翼 MapReduce 服务（MRS）

组件操作指南（普通版）

天翼云科技有限公司

目 录

1 使用 Alluxio

1.1 配置底层存储系统

用户想要通过统一的客户端 API 和全局命名空间访问包括 HDFS 和 OBS 在内的持久化存储系统，从而实现了对计算和存储的分离时，可以在 MRS Manager 页面中配置 Alluxio 的底层存储系统来实现。集群创建后，默认的底层存储地址是 hdfs://hacluster/，即将 HDFS 的根目录映射到 Alluxio。

前提条件

- 已安装 Alluxio 服务的集群。
- 获取用户“admin”帐号密码。“admin”密码在创建 MRS 集群时由用户指定。

配置 HDFS 作为 Alluxio 的底层文件系统

说明

开启 Kerberos 认证的安全集群不支持该功能。

步骤 1 请参考修改集群服务配置参数，进入 Alluxio 的“全部配置”页面。

步骤 2 在左侧边栏中选择“Alluxio > 底层存储系统”，修改参数“alluxio.master.mount.table.root.ufs”的值为“hdfs://hacluster/XXX/”。

例如：若想将“HDFS 根目录/alluxio/”作为 alluxio 的根目录，则修改参数“alluxio.master.mount.table.root.ufs”的值为“hdfs://hacluster/alluxio/”。

步骤 3 单击“保存配置”，并在弹出窗口中勾选“重新启动受影响的服务和实例。”

步骤 4 单击“确定”重启 Alluxio 服务。

----结束

1.2 通过数据应用访问 Alluxio

访问 Alluxio 文件系统的端口号是 19998，即地址为 alluxio://<alluxio 的 master 节点 ip>:19998/<PATH>，本节将通过示例介绍如何通过数据应用（Spark、Hive、Hadoop MapReduce 和 Presto）访问 Alluxio。

使用 Alluxio 作为 Spark 应用程序的输入和输出

步骤 1 以 root 用户登录集群的 Master 节点，密码为用户创建集群时设置的 root 密码。

步骤 2 执行如下命令，配置环境变量。

source /opt/client/bigdata_env

步骤 3 如果当前集群已启用 Kerberos 认证，执行如下命令认证当前用户。如果当前集群未启用 Kerberos 认证，则无需执行此命令。

kinit *MRS 集群用户*

例如, **kinit admin**

步骤 4 准备输入文件，将本地数据复制到 Alluxio 文件系统中。

如在本地/home 目录下准备一个输入文件 test_input.txt，然后执行如下命令，将 test_input.txt 文件放入 Alluxio 中。

alluxio fs copyFromLocal /home/test_input.txt /input

步骤 5 执行如下命令启动 spark-shell。

spark-shell

步骤 6 在 spark-shell 中运行如下命令。

val s = sc.textFile("alluxio://<Alluxio 的节点名称>:19998/input")

val double = s.map(line => line + line)

double.saveAsTextFile("alluxio://<Alluxio 的节点名称>:19998/output")

说明

<Alluxio 的节点名称>:19998，请根据实际情况替换为 AlluxioMaster 实例所在所有节点的节点名称与端口号，各个名称与端口之间以英文逗号间隔，例如：node-ana-coremspb.mrs-m0va.com:19998,node-master2kiww.mrs-m0va.com:19998,node-master1cqwv.mrs-m0va.com:19998

步骤 7 使用“Ctrl + C”退出 spark-shell。

步骤 8 通过 alluxio 命令行 **alluxio fs ls /**查看 alluxio 根目录下存在一个输出目录/output，其中包含了输入文件 input 的双倍内容。

----结束

在 Alluxio 上创建 Hive 表

步骤 1 以 root 用户登录集群的 Master 节点，密码为用户创建集群时设置的 root 密码。

步骤 2 执行如下命令，配置环境变量。

source /opt/client/bigdata_env

步骤 3 如果当前集群已启用 Kerberos 认证，执行如下命令认证当前用户。如果当前集群未启用 Kerberos 认证，则无需执行此命令。

kinit *MRS 集群用户*

例如, **kinit admin**

步骤 4 准备输入文件，如在本地/home 目录下准备一个输入文件 hive_load.txt， 内容为

```
1, Alice, company A
2, Bob, company B
```

步骤 5 执行如以下命令，将 hive_load.txt 文件放入 Alluxio 中。

alluxio fs copyFromLocal /home/hive_load.txt /hive_input

步骤 6 执行如下命令启动 hive beeline。

beeline

步骤 7 在 beeline 中运行如下命令，根据 Alluxio 中的输入文件进行创表。

CREATE TABLE u_user(id INT, name STRING, company STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE;

LOAD DATA INPATH 'alluxio://<Alluxio 的节点名称>:19998/hive_input' INTO TABLE u_user;

说明

<Alluxio 的节点名称>:19998，请根据实际情况替换为 AlluxioMaster 实例所在所有节点的节点名称与端口号，各个名称与端口之间以英文逗号间隔，例如：node-ana-coremspb.mrs-m0va.com:19998,node-master2kiww.mrs-m0va.com:19998,node-master1cqwv.mrs-m0va.com:19998

步骤 8 执行如下命令查看创建的表。

select * from u_user;

----结束

在 Alluxio 上运行 Hadoop Wordcount

步骤 1 以 root 用户登录集群的 Master 节点，密码为用户创建集群时设置的 root 密码。

步骤 2 执行如下命令，配置环境变量。

source /opt/client/bigdata_env

步骤 3 如果当前集群已启用 Kerberos 认证，执行如下命令认证当前用户。如果当前集群未启用 Kerberos 认证，则无需执行此命令。

kinit *MRS 集群用户*

例如, **kinit admin**

步骤 4 准备输入文件，将本地数据复制到 Alluxio 文件系统中。

如在本地/home 目录下准备一个输入文件 test_input.txt，然后执行如下命令，将 test_input.txt 文件放入 Alluxio 中。

alluxio fs copyFromLocal /home/test_input.txt /input

步骤 5 通过 yarn jar 执行 wordcount 作业。

yarn jar /opt/share/hadoop-mapreduce-examples-<hadoop 版本号>-mrs-<mrs 集群版本号>/hadoop-mapreduce-examples-<hadoop 版本号>-mrs-<mrs 集群版本号>.jar wordcount alluxio://<Alluxio 的节点名称>:19998/input alluxio://<Alluxio 的节点名称>:19998/output

说明

- <hadoop 版本号>请根据实际情况替换。
- <mrs 集群版本号>替换为 MRS 的大版本号，如 MRS 1.9.2 版本集群此处为 **mrs-1.9.0**。
- <Alluxio 的节点名称>:19998，请根据实际情况替换为 AlluxioMaster 实例所在所有节点的节点名称与端口号，各个名称与端口之间以英文逗号间隔，例如：node-ana-coremspb.mrs-m0va.com:19998,node-master2kiww.mrs-m0va.com:19998,node-master1cqwv.mrs-m0va.com:19998

步骤 6 通过 alluxio 命令行 **alluxio fs ls /**查看 alluxio 根目录下存在一个输出目录/output，包含了 wordcount 的结果。

----结束

使用 Presto 在 Alluxio 上查询表

步骤 1 以 root 用户登录集群的 Master 节点，密码为用户创建集群时设置的 root 密码。

步骤 2 执行如下命令，配置环境变量。

source /opt/client/bigdata_env

步骤 3 如果当前集群已启用 Kerberos 认证，执行如下命令认证当前用户。如果当前集群未启用 Kerberos 认证，则无需执行此命令。

kinit *MRS 集群用户*

例如, **kinit admin**

步骤 4 启动 hive beeline 在 alluxio 上创建表。

beeline

CREATE TABLE u_user (id int, name string, company string) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LOCATION 'alluxio://<Alluxio 的节点名称>:19998/u_user';

insert into u_user values(1,'Alice','Company A'),(2, 'Bob', 'Company B');

说明

<Alluxio 的节点名称>:19998，请根据实际情况替换为 AlluxioMaster 实例所在所有节点的节点名称与端口号，各个名称与端口之间以英文逗号间隔，例如：node-ana-coremspb.mrs-m0va.com:19998,node-master2kiww.mrs-m0va.com:19998,node-master1cqwv.mrs-m0va.com:19998

步骤 5 启动 Presto 客户端，具体请参见使用客户端执行查询语句的步骤 2~步骤 8。

步骤 6 在 Presto 客户端中执行查询语句 **select * from hive.default.u_user;** 查询 alluxio 上创建表。

图1-1 Presto 查询 alluxio 上创建的表

```
presto> select * from hive.default.u_user;
 id | name  |  company
----+-------+-----------
  1 | Alice | Company A
  2 | Bob   | Company B
(2 rows)
```

----结束

1.3 Alluxio 常用操作

前期准备

1. 创建安装 Alluxio 组件的集群。
2. 以 root 用户登录集群的主 Master 节点，密码为用户创建集群时设置的 root 密码。
3. 执行如下命令，配置环境变量。
   source /opt/client/bigdata_env

使用 Alluxio Shell

Alluxio shell 包含多种与 Alluxio 交互的命令行操作。

- 要查看文件系统操作命令列表。
  alluxio fs
- 使用 **ls** 命令列出 Alluxio 里的文件。例如列出根目录下所有文件。
  alluxio fs ls /
- 使用 copyFromLocal 命令可以复制本地文件到 Alluxio 中。
  alluxio fs copyFromLocal /home/test_input.txt /test_input.txt
  命令执行后回显：

  ```
  Copied file:///home/test_input.txt to /test_input.txt
  ```
- 再次使用 ls 命令列出 Alluxio 中的文件，可以看到刚刚拷贝的 test_input.txt 文件。
  alluxio fs ls /
  命令执行后回显：

```
12      PERSISTED 11-28-2019 17:10:17:449 100% /test_input.txt
```

输出显示 test_input.txt 文件在 Alluxio 中，各参数含义为文件的大小、是否被持久化、创建日期、Alluxio 中这个文件的缓存占比、文件名。

- 使用 cat 命令打印文件的内容。

  alluxio fs cat /test_input.txt

  命令执行后回显：

```
Test Alluxio
```

Alluxio 中的挂载功能

Alluxio 通过统一命名空间的特性统一了对存储系统的访问。详情请参考：https://docs.alluxio.io/os/user/2.0/cn/advanced/Namespace-Management.html

这个特性允许用户挂载不同的存储系统到 Alluxio 命名空间中并且通过 Alluxio 命名空间无缝地跨存储系统访问文件。

1. 在 Alluxio 中创建一个目录作为挂载点。

   alluxio fs mkdir /mnt

```
Successfully created directory /mnt
```

2. 挂载一个已有的 OBS 文件系统到 Alluxio（前提：给集群配置有 OBS OperateAccess 权限的委托）。此处以 obs-mrstest 文件系统为例，请根据实际情况替换文件系统名。

   alluxio fs mount /mnt/obs obs://obs-mrstest/data

```
Mounted obs://obs-mrstest/data at /mnt/obs
```

3. 通过 Alluxio 命名空间列出 OBS 文件系统中的文件。使用 **ls** 命令列出 OBS 挂载目录下的文件。

   alluxio fs ls /mnt/obs

```
38      PERSISTED 11-28-2019 17:42:54:554   0% /mnt/obs/hive_load.txt
12      PERSISTED 11-28-2019 17:43:07:743   0% /mnt/obs/test_input.txt
```

   新挂载的文件和目录也可以通过 Alluxio WebUI 查看。

4. 挂载完成后，通过 Alluxio 统一命名空间，可以无缝地从不同存储系统中交互数据。例如，使用 **ls -R** 命令，递归地列举出一个目录下的所有文件。

   alluxio fs ls -R /

```
       0      PERSISTED 11-28-2019 11:15:19:719  DIR /app-logs
       1      PERSISTED 11-28-2019 11:18:36:885  DIR /apps
       1      PERSISTED 11-28-2019 11:18:40:209  DIR /apps/templeton
239440292      PERSISTED 11-28-2019 11:18:40:209   0%
/apps/templeton/hive.tar.gz
.....
       1      PERSISTED 11-28-2019 19:00:23:879  DIR /mnt
       2      PERSISTED 11-28-2019 19:00:23:879  DIR /mnt/obs
      38      PERSISTED 11-28-2019 17:42:54:554   0% /mnt/obs/hive load.txt
      12      PERSISTED 11-28-2019 17:43:07:743   0% /mnt/obs/test input.txt
.....
```

   输出显示了 Alluxio 文件系统根目录（默认值是 HDFS 的根目录，即 hdfs://hacluster/）中来源于挂载存储系统的所有文件。/app-logs 和/apps 目录在 HDFS 文件系统中，/mnt/obs/目录在 OBS 中。

用 Alluxio 加速数据访问

由于 Alluxio 利用内存存储数据，它可以加速数据的访问。例如：

1. 上传一个文件 test_data.csv（文件是一份记录了食谱的样本）到 obs-mrstest 文件系统的/data 目录下。通过 **ls** 命令显示文件状态：

   alluxio fs ls /mnt/obs/test_data.csv

   ```
   294520189      PERSISTED 11-28-2019 19:38:55:000   0% /mnt/obs/test_data.csv
   ```

   输出显示了该文件在 Alluxio 中缓存占比为 0%，即不在 Alluxio 内存中。

2. 统计该文件中单词"milk"出现的次数，并计算耗时。

   time alluxio fs cat /mnt/obs/test_data.csv | grep -c milk

   ```
   52180

   real    0m10.765s
   user    0m5.540s
   sys     0m0.696s
   ```

3. 第一次读取数据后会将数据放在内存中，Alluxio 再次读取时可以提高访问该数据的速度。例如：在通过 **cat** 命令获取文件后，用 **ls** 命令再查看文件的状态。

   alluxio fs ls /mnt/obs/test_data.csv

   ```
   294520189      PERSISTED 11-28-2019 19:38:55:000 100% /mnt/obs/test_data.csv
   ```

   输出显示文件已经 100% 被加载到 Alluxio 中。

4. 再次访问该文件，统计单词“eggs”出现的次数，并计算耗时。

   time alluxio fs cat /mnt/obs/test_data.csv | grep -c eggs

   ```
   59510

   real    0m5.777s
   user    0m5.992s
   sys     0m0.592s
   ```

   对比两次耗时可以看出存储在 Alluxio 内存中的数据，数据访问耗时明显缩短。

2 使用 CarbonData（MRS 3.x 之前版本）

2.1 从零开始使用 CarbonData

MRS 3.x 之前版本参考本章节，MRS 3.x 及后续版本请参考使用 CarbonData（MRS 3.x 及之后版本）。

本章节介绍使用 Spark CarbonData 的基本流程，所有任务场景基于 spark-beeline 环境。CarbonData 快速入门包含以下任务：

1. 连接到 Spark

   在对 CarbonData 进行任何操作之前，需要先连接到 Spark。

2. 创建 CarbonData 表

   连接 CarbonData 之后，需要创建 CarbonData Table，用于加载数据和执行查询操作。

3. 加载数据到 CarbonData 表

   用户从 HDFS 中的 CSV 文件加载数据到所创建的表中。

4. 在 CarbonData 中查询数据

   在 CarbonData 表加载数据之后，用户可以执行所需的查询操作，例如 groupby 或者 where 等。

前提条件

已安装客户端，具体参见使用 MRS 客户端。

操作步骤

步骤 1 连接到 Spark CarbonData。

1. 根据业务情况，准备好客户端，使用 **root** 用户登录安装客户端的节点。

   例如在 Master2 节点更新客户端，则在该节点登录客户端，具体参见使用 MRS 客户端。

2. 切换用户与配置环境变量。

   sudo su - omm

   source /opt/client/bigdata_env

3. 启用 Kerberos 认证的集群，执行以下命令认证用户身份。未启用 Kerberos 认证集群无需执行。

   kinit *Spark 组件用户名*

   说明

   用户需要加入用户组 hadoop、hive，主组 hadoop。

4. 执行以下命令，连接到 Spark 运行环境：

   spark-beeline

步骤 2 执行命令创建 CarbonData 表。

CarbonData 表可用于加载数据和执行查询操作，例如执行以下命令创建 CarbonData 表：

CREATE TABLE x1 (imei string, deviceInformationId int, mac string, productdate timestamp, updatetime timestamp, gamePointId double, contractNumber double)

STORED BY 'org.apache.carbondata.format'

TBLPROPERTIES ('DICTIONARY_EXCLUDE'='mac','DICTIONARY_INCLUDE'='deviceInformationId');

命令执行结果如下：

```
+---------+--+
| result  |
+---------+--+
+---------+--+
No rows selected (1.551 seconds)
```

步骤 3 从 CSV 文件加载数据到 CarbonData 表。

根据所要求的参数运行命令从 CSV 文件加载数据，且仅支持 CSV 文件。**LOAD** 命令中配置的 CSV 列名，需要和 CarbonData 表列名相同，顺序也要对应。CSV 文件中的数据的列数，以及数据格式需要和 CarbonData 表匹配。

文件需要保存在 HDFS 中。用户可以将文件上传到 OBS，并在 MRS 管理控制台“文件管理”将文件从 OBS 导入 HDFS。

如果集群启用了 Kerberos 认证，则需要在工作环境准备 CSV 文件，然后可以使用开源 HDFS 命令，参考 5 将文件从工作环境导入 HDFS，并设置 Spark 组件用户在 HDFS 中对文件有读取和执行的权限。

例如，HDFS 的“tmp”目录有一个文件“data.csv”，内容如下：

```
x123,111,dd,2017-04-20 08:51:27,2017-04-20 07:56:51,2222,33333
```

执行导入命令：

LOAD DATA inpath 'hdfs://hacluster/tmp/data.csv' into table x1 options('DELIMITER'=',','QUOTECHAR'='"','FILEHEADER'='imei, deviceinformationid,mac,productdate,updatetime,gamepointid,contractnumber');

命令执行结果如下：

```
+---------+--+
| Result  |
+---------+--+
+---------+--+
No rows selected (3.039 seconds)
```

步骤 4 在 CarbonData 中查询数据。

- **获取记录数**

  为了获取在 CarbonData table 中的记录数，可以执行以下命令。

  select count(*) from x1;

- **使用 Groupby 查询**

  为了获取不重复的“deviceinformationid”记录数，可以执行以下命令。

  select deviceinformationid,count (distinct deviceinformationid) from x1 group by deviceinformationid;

- **使用条件查询**

  为了获取特定 **deviceinformationid** 的记录，可以执行以下命令。

  select * from x1 where deviceinformationid='111';

  说明

  在执行数据查询操作后，如果查询结果中某一列的结果含有中文字等其他非英文字符，会导致查询结果中的列不能对齐，这是由于不同语言的字符在显示时所占的字宽不尽相同。

步骤 5 执行以下命令退出 Spark 运行环境。

!quit

----结束

2.2 CarbonData 表简介

简介

CarbonData 表与 RDBMS 中的表类似，RDBMS 数据存储在由行和列构成的表中。CarbonData 表存储的也是结构化的数据，具有固定列和数据类型。CarbonData 中的数据存储在表实体文件中。

支持的数据类型

CarbonData 表支持以下数据类型：

- Int
- String
- BigInt
- Decimal
- Double
- TimeStamp

表 2-1 对所支持的数据类型和对应的范围进行了详细说明。

表2-1 CarbonData 数据类型

数据类型	描述
Int	4 字节有符号整数，从-2,147,483,648 到 2,147,483,647。 说明 非字典列如果是 Int 类型，会在内部存储为 BigInt 类型。
String	最大支持字符长度为 100000。
BigInt	使用 64-bit 存储数据，支持从-9,223,372,036,854,775,808 到 9,223,372,036,854,775,807。
Decimal	默认值是(10,0)，最大值是(38,38)。 说明 当进行带过滤条件的查询时，为了得到准确的结果，需要在数字后面加上 BD。例如，**select * from carbon_table where num = 1234567890123456.22BD。**
Double	使用 64-bit 存储数据，从 4.9E-324 到 1.7976931348623157E308。
TimeStamp	默认格式为“yyyy-MM-dd HH:mm:ss”。

说明

所有 Integer 类型度量均以 BigInt 类型进行处理与显示。

2.3 创建 CarbonData 表

操作场景

使用 CarbonData 前需先创建表，才可从表中加载数据和查询数据。

使用自定义列创建表

可通过指定各列及其数据类型来创建表。启用 Kerberos 认证的分析集群创建 CarbonData 表时，如果用户需要在默认数据库“default”以外的数据库创建新表，则需要在 Hive 角色管理中为用户绑定的角色添加指定数据库的“Create”权限。

命令示例：

CREATE TABLE IF NOT EXISTS productdb.productSalesTable (

productNumber Int,

productName String,

storeCity String,

storeProvince String,

revenue Int)

STORED BY 'org.apache.carbondata.format'

TBLPROPERTIES (

'table_blocksize'='128',

'DICTIONARY_EXCLUDE'='productName',

'DICTIONARY_INCLUDE'='productNumber');

上述命令所创建的表的详细信息如下：

表2-2 表信息定义

参数	描述
productSalesTable	待创建的表的名称。该表用于加载数据进行分析。 表名由字母、数字、下划线组成。
productdb	数据库名称。该数据库将与其中的表保持逻辑连接以便于识别和管理。 数据库名称由字母、数字、下划线组成。
productNumber productName storeCity storeProvince revenue	表中的列，代表执行分析所需的业务实体。 列名（字段名）由字母、数字、下划线组成。 说明 CarbonData 暂不支持设置列是否允许为空、默认值以及主键。
table_blocksize	CarbonData 表使用的数据文件的 block 大小，默认值为 1024，取值范围为 1～2048，单位为 MB。 • 如果“table_blocksize”值太小，数据加载时将生成过多的小数据文件，可能会影响 HDFS 的使用性能。 • 如果“table_blocksize”值太大，数据查询时索引匹配的 block 数据量较大，导致读取并发度不高，从而降低查询性能。 一般情况下，建议根据数据量级别来选择大小。例如：GB 级别用 256，TB 级别用 512，PB 级别用 1024。
DICTIONARY_EXCLUDE	设置指定列不生成字典，适用于数值复杂度高的列。系统默认为 String 类型的列做字典编码，但是如果字典值过多，会导致字典转换操作增加造成性能下降。 一般情况下，列的数值复杂度高于 5 万，可以被认定为高复杂度，则需要排除掉字典编码，该参数为可选参数。 说明 在非字典列中，只支持 String 和 Timestamp 数据类型。
DICTIONARY_INCLU	设置指定列生成字典，适用于数值复杂度低的列，可以提

参数	描述
DE	升字典列上的 **groupby** 性能，为可选参数。一般情况下，字典列的复杂度不应该高于 5 万。

2.4 删除 CarbonData 表

操作场景

用户根据业务使用情况，可以删除不再使用的 CarbonData 表。删除表后，其所有的元数据以及表中已加载的数据都会被删除。

操作步骤

步骤 1 运行如下命令删除表。

DROP TABLE *[IF EXISTS] [db_name.]table_name*;

“db_name”为可选参数。如果没有指定“db_name”，那么将会删除当前数据库下名为“table_name”的表。

例如执行命令，删除数据库“productdb”下的表“productSalesTable”：

DROP TABLE productdb.productSalesTable;

步骤 2 执行以下命令查询表是否被删除：

SHOW TABLES;

----结束

3 使用 CarbonData（MRS 3.x 及之后版本）

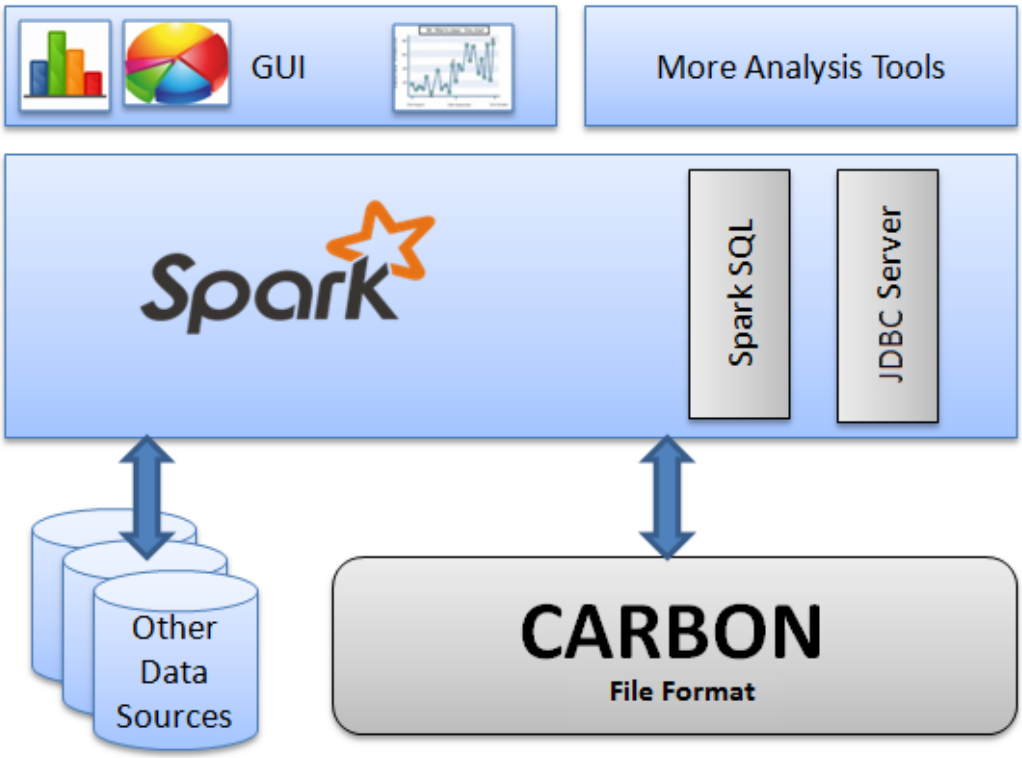
3.1 概述

MRS 3.x 及后续版本参考本章节，MRS 3.x 之前版本请参考使用 CarbonData（MRS 3.x 之前版本）。

3.1.1 CarbonData 简介

CarbonData 是一种新型的 Apache Hadoop 本地文件格式，使用先进的列式存储、索引、压缩和编码技术，以提高计算效率，有助于加速超过 PB 数量级的数据查询，可用于更快的交互查询。同时，CarbonData 也是一种将数据源与 Spark 集成的高性能分析引擎。

图3-1 CarbonData 基本架构



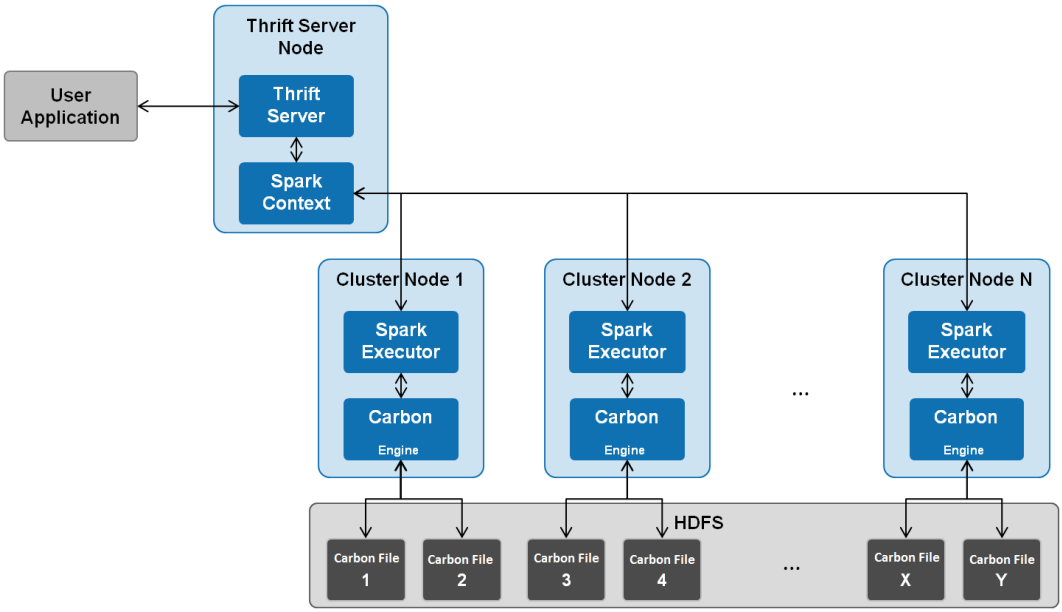
使用 CarbonData 的目的是对大数据即席查询提供超快速响应。从根本上说，CarbonData 是一个 OLAP 引擎，采用类似于 RDBMS 中的表来存储数据。用户可将大量（10TB 以上）的数据导入以 CarbonData 格式创建的表中，CarbonData 将以压缩的多维索引列格式自动组织和存储数据。数据被加载到 CarbonData 后，就可以执行即席查询，CarbonData 将对数据查询提供秒级响应。

CarbonData 将数据源集成到 Spark 生态系统，用户可使用 Spark SQL 执行数据查询和分析。也可以使用 Spark 提供的第三方工具 JDBCServer 连接到 Spark SQL。

CarbonData 结构

CarbonData 作为 Spark 内部数据源运行，不需要额外启动集群节点中的其他进程，CarbonData Engine 在 Spark Executor 进程之中运行。

图3-2 CarbonData 结构



存储在 CarbonData Table 中的数据被分成若干个 CarbonData 数据文件，每一次数据查询时，CarbonData Engine 模块负责执行数据集的读取、过滤等实际任务。CarbonData Engine 作为 Spark Executor 进程的一部分运行，负责处理数据文件块的一个子集。

Table 数据集数据存储在 HDFS 中。同一 Spark 集群内的节点可以作为 HDFS 的数据节点。

CarbonData 特性

- SQL 功能：CarbonData 与 Spark SQL 完全兼容，支持所有可以直接在 Spark SQL 上运行的 SQL 查询操作。
- 简单的 Table 数据集定义：CarbonData 支持易于使用的 DDL(数据定义语言)语句来定义和创建数据集。CarbonData DDL 十分灵活、易于使用，并且足够强大，可以定义复杂类型的 Table。
- 便捷的数据管理：CarbonData 为数据加载和维护提供多种数据管理功能。CarbonData 支持加载历史数据以及增量加载新数据。加载的数据可以基于加载时间进行删除，也可以撤销特定的数据加载操作。
- CarbonData 文件格式是 HDFS 中的列式存储格式。该格式具有许多新型列存储文件的特性，例如，分割表和数据压缩。CarbonData 具有以下独有的特点：
  - 伴随索引的数据存储：由于在查询中设置了过滤器，可以显著加快查询性能，减少 I/O 扫描次数和 CPU 资源占用。CarbonData 索引由多个级别的索引组成，处理框架可以利用这个索引来减少需要安排和处理的任务，也可以通过在任务扫描中以更精细的单元（称为 blocklet）进行 skip 扫描来代替对整个文件的扫描。
  - 可选择的数据编码：通过支持高效的数据压缩，可基于压缩/编码数据进行查询，在将结果返回给用户之前，才将编码转化为实际数据，这被称为“延迟物化”。

- 支持一种数据格式应用于多种用例场景：例如，交互式 OLAP-style 查询，顺序访问（big scan），随机访问（narrow scan）。

CarbonData 关键技术和优势

- 快速查询响应：高性能查询是 CarbonData 关键技术的优势之一。CarbonData 查询速度大约是 Spark SQL 查询的 10 倍。CarbonData 使用的专用数据格式围绕高性能查询进行设计，其中包括多种索引技术和多次的 Push down 优化，从而对 TB 级数据查询进行最快响应。
- 高效率数据压缩：CarbonData 使用轻量级压缩和重量级压缩的组合压缩算法压缩数据，可以减少 60%~80%数据存储空间，很大程度上节省硬件存储成本。

3.1.2 CarbonData 主要规格

CarbonData 主要规格

表3-1 CarbonData 主要规格

实体	测试值	测试环境
表数	10000	3 个节点，每个 executor 4 个 CPU 核，20GB。Driver 内存 5GB，3 个 Executor。 总列数：107 String：75 Int：13 BigInt：7 Timestamp：6 Double：6
表的列数	2000	3 个节点，每个 executor4 个 CPU 核，20GB。Driver 内存 5GB，3 个 Executor。
原始 CSV 文件大小的最大值	200GB	17 个 cluster 节点，每个 executor 150GB，25 个 CPU 核。Driver 内存 10 GB，17 个 Executor。
每个文件夹的 CSV 文件数	100 个文件夹，每个文件夹 10 个文件，每个文件大小 50MB。	3 个节点，每个 executor4 个 CPU 核，20GB。Driver 内存 5GB，3 个 Executor。
加载文件夹数	10000	3 个节点，每个 executor4 个 CPU 核，20GB。Driver 内存 5GB，3 个 Executor。

数据加载所需的内存取决于以下因素：

- 列数
- 列值大小
- 并发（使用“carbon.number.of.cores.while.loading”进行配置）
- 在内存中排序的大小（使用“carbon.sort.size”进行配置）
- 中间缓存（使用“carbon.graph.rowset.size”进行配置）

加载包含 1000 万条记录和 300 列的 8 GB CSV 文件的数据，每行大小约为 0.8KB 的 8GB CSV 文件的数据，需要约为 10GB 的 executor 执行内存，也就是说，“carbon.sort.size” 配置为“100000”，所有其他前面的配置保留默认值。

二级索引表规格

表3-2 二级索引表规格

实体	测试值
二级索引表数量	10
二级索引表中的组合列的列数	5
二级索引表中的列名长度（单位：字符）	120
二级索引表名长度（单位：字符）	120
表中所有二级索引表的表名+列名的累积长度*（单位：字符）	3800**

说明

- * Hive 允许的上限值或可用资源的上限值。
- ** 二级索引表使用 hive 注册，并以 json 格式的值存储在 HiveSERDEPROPERTIES 中。由 hive 支持的 SERDEPROPERTIES 的最大字符数为 4000 个字符，无法更改。

3.2 配置参考

本章节介绍 CarbonData 所有配置的详细信息。

表3-3 carbon.properties 中的系统配置

参数	默认值	描述
carbon.ddl.base.hdfs.url	hdfs://hacluster/opt/data	此属性用于从 HDFS 基本路径配置 HDFS 相对路径，在“fs.defaultFS”中进行配置。在“carbon.ddl.base.hdfs.url”中配置的路径将被追加

参数	默认值	描述
		到在“fs.defaultFS”中配置的 HDFS 路径中。如果配置了这个路径，则用户不需要通过完整路径加载数据。 例如：如果 CSV 文件的绝对路径是“hdfs://10.18.101.155:54310/data/cnbc/2016/xyz.csv”，其中，路径“hdfs://10.18.101.155:54310”来源于属性“fs.defaultFS”并且用户可以把“/data/cnbc/”作为“carbon.ddl.base.hdfs.url”配置。 当前，在数据加载时，用户可以指定 CSV 文件为“/2016/xyz.csv”。
carbon.badRecords.location	-	指定 Bad records 的存储路径。此路径为 HDFS 路径。默认值为 Null。如果启用了 bad records 日志记录或者 bad records 操作重定向，则该路径必须由用户进行配置。
carbon.bad.records.action	fail	以下是 bad records 的四种行为类型： FORCE：通过将 bad records 存储为 NULL 来自动更正数据。 REDIRECT：Bad records 被写入原始 CSV 文件而不是被加载。 IGNORE：Bad records 既不被加载也不被写入原始 CSV 文件。 FAIL：如果找到任何 bad records，则数据加载失败。
carbon.update.sync.folder	/tmp/carbondata	modifiedTime.mdt 文件路径，可以设置为已有路径或新路径。 说明 如果设置为已有路径，需确保所有用户都可以访问该路径，且该路径具有 777 权限。

表3-4 carbon.properties 中的性能配置

参数	默认值	描述
数据加载配置		
carbon.sort.file.write.buffer.size	16384	为了限制内存的使用，CarbonData 会将数据排序并写入临时文件中。该参数控制读取和写入临时文件过程使用的缓存大小。单位：字节。 取值范围为：10240~10485760。
carbon.graph.r	100000	数据加载图步骤之间交换的行集大小。

参数	默认值	描述
owset.size		最小值=500，最大值=1000000
carbon.number.of.cores.while.loading	6	数据加载时所使用的核数。配置的核数越大压缩性能越好。如果 CPU 资源充足可以增加此值。
carbon.sort.size	500000	内存排序的数据大小。
carbon.enableXXHash	true	用于 hashkey 计算的 hashmap 算法。
carbon.number.of.cores.block.sort	7	数据加载时块排序所使用的核数。
carbon.max.driver.lru.cache.size	-1	在 driver 端加载数据所达到的最大 LRU 缓存大小。以 MB 为单位，默认值为-1，表示缓存没有内存限制。只允许使用大于 0 的整数值。
carbon.max.executor.lru.cache.size	-1	在 executor 端加载数据所达到的最大 LRU 缓存大小。以 MB 为单位，默认值为-1，表示缓存没有内存限制。只允许使用大于 0 的整数值。如果未配置该参数，则将考虑参数“carbon.max.driver.lru.cache.size”的值。
carbon.merge.sort.prefetch	true	在数据加载过程中，从排序的临时文件中读取数据进行合并排序时，启用数据预取。
carbon.update.persist.enable	true	启用此参数将考虑持久化数据，减少 UPDATE 操作的执行时间。
enable.unsafe.sort	true	指定在数据加载期间是否使用非安全排序。非安全的排序减少了数据加载操作期间的垃圾回收（GC），从而提高了性能。默认值为“true”，表示启用非安全排序功能。
enable.offheap.sort	true	在数据加载期间启用堆排序。
offheap.sort.chunk.size.inmb	64	指定需要用于排序的数据块的大小。最小值为 1MB，最大值为 1024MB。
carbon.unsafe.working.memory.in.mb	512	指定非安全工作内存的大小。这将用于排序数据，存储列页面等。单位是 MB。 数据加载所需内存： (“carbon.number.of.cores.while.loading”的值[默认值 = 6]) x 并行加载数据的表格 x (“offheap.sort.chunk.size.inmb”的值[默认值 = 64 MB] + “carbon.blockletgroup.size.in.mb”的值[默认值 = 64 MB] + 当前的压缩率[64 MB/3.5]) = ~900 MB 每表格

参数	默认值	描述
		数据查询所需内存： (SPARK_EXECUTOR_INSTANCES. [默认值 = 2]) x (carbon.blockletgroup.size.in.mb [默认值 = 64 MB] + “carbon.blockletgroup.size.in.mb”解压内容[默认值 = 64 MB * 3.5]) x (每个执行器核数[默认值 = 1]) = ~ 600 MB
carbon.sort.inmemory.storage.size.in.mb	512	指定要存储在内存中的中间排序数据的大小。达到该指定的值，系统会将数据写入磁盘。单位是 MB。
sort.inmemory.size.inmb	1024	指定要保存在内存中的中间排序数据的大小。达到该指定值后，系统会将数据写入磁盘。单位：MB。 如果配置了“carbon.unsafe.working.memory.in.mb”和“carbon.sort.inmemory.storage.size.in.mb”，则不需要配置该参数。如果此时也配置了该参数，那么这个内存的20%将用于工作内存“carbon.unsafe.working.memory.in.mb”，80%将用于排序存储内存“carbon.sort.inmemory.storage.size.in.mb”。 说明 Spark 配置参数“spark.yarn.executor.memoryOverhead”的值应该大于 CarbonData 配置参数“sort.inmemory.size.inmb”的值，否则如果堆外（off heap）访问超出配置的 executor 内存，则 YARN 可能会停止 executor。
carbon.blockletgroup.size.in.mb	64	数据作为 blocklet group 被系统读入。该参数指定 blocklet group 的大小。较高的值会有更好的顺序 IO 访问性能。 最小值为 16MB，任何小于 16MB 的值都将重置为默认值（64MB）。 单位：MB。
enable.inmemory.merge.sort	false	指定是否启用内存合并排序（inmemorymerge sort）。
use.offheap.in.query.processing	true	指定是否在查询处理中启用 offheap。
carbon.load.sort.scope	local_sort	指定加载操作的排序范围。支持两种类型的排序，batch_sort 和 local_sort。选择 batch_sort 将提升加载性能，但会降低查询性能。
carbon.batch.sort.size.inmb	-	指定在数据加载期间为批处理排序而考虑的数据大小。推荐值为小于总排序数据的 45%。该值以 MB 为单位。 说明

参数	默认值	描述
		如果没有设置参数值，那么默认情况下其大约等于“sort.inmemory.size.inmb”参数值的 45%。
enable.unsafe.columnpage	true	指定在数据加载或查询期间，是否将页面数据保留在堆内存中，以避免垃圾回收受阻。
carbon.use.local.dir	false	是否使用 YARN 本地目录加载多个磁盘的数据。设置为 true，则使用 YARN 本地目录加载多个磁盘的数据，以提高数据加载性能。
carbon.use.multiple.temp.dir	false	是否使用多个临时目录存储临时文件以提高数据加载性能。
carbon.load.datamaps.parallel.db_name.table_name	NA	值为 true 或者 false。可以设置数据库名和表名，使得该表的首次查询性能得到提升。
压缩配置		
carbon.number.of.cores.while.compacting	2	在压缩过程中用于写入数据所使用的核数。配置的核数越大压缩性能越好。如果 CPU 资源充足可以增加此值。
carbon.compaction.level.threshold	4,3	该属性用于 Minor 压缩，决定合并 segment 的数量。 例如：如果被设置为“2,3”，则将每 2 个 segment 触发一次 Minor 压缩。“3”是 Level 1 压缩的 segment 个数，这些 segment 将进一步被压缩为新的 segment。 有效值为 0-100。
carbon.major.compaction.size	1024	使用该参数配置 Major 压缩的大小。总数低于该阈值的 segment 将被合并。 单位为 MB。
carbon.horizontal.compaction.enable	true	该参数用于配置打开/关闭水平压缩。在每个 DELETE 和 UPDATE 语句之后，如果增量（DELETE / UPDATE）文件超过指定的阈值，则可能发生水平压缩。默认情况下，该参数值设置为“true”，打开水平压缩功能，可将参数值设置为“false”来关闭水平压缩功能。
carbon.horizontal.update.compaction.threshold	1	该参数指定 segment 内的 UPDATE 增量文件数的阈值限制。在增量文件数量超过阈值的情况下，segment 内的 UPDATE 增量文件变得适合水平压缩，并压缩为单个 UPDATE 增量文件。默认情况下，该参数值设置为 1。可以设置为 1 到 10000 之间的值。
carbon.horizontal.delete.compaction.thres	1	该参数指定 segment 的 block 中的 DELETE 增量文件数量的阈值限制。在增量文件数量超过阈值的情况下，segment 特定 block 的 DELETE 增量文件变得适合水平

参数	默认值	描述
hold		压缩，并压缩为单个 DELETE 增量文件。 默认情况下，该参数值设置为 1。可以设置为 1 到 10000 之间的值。
查询配置		
carbon.number.of.cores	4	查询时所使用的核数。
carbon.limit.block.distribution.enable	false	当查询语句中包含关键字 limit 时，启用或禁用 CarbonData 块分布。默认值为“false”，将对包含关键字 limit 的查询语句禁用块分布。此参数调优请参考性能调优的相关配置。
carbon.custom.block.distribution	false	指定是使用 Spark 还是 CarbonData 的块分配功能。默认情况下，其配置值为“false”，表明启用 Spark 块分配。若要使用 CarbonData 块分配，请将配置值更改为“true”。
carbon.infilter.subquery.pushdown.enable	false	如果启用此参数，并且用户在具有 subquery 的过滤器中触发 Select 查询，则执行子查询，并将输出作为 IN 过滤器广播到左表，否则将执行 SortMergeSemiJoin。建议在 IN 过滤器子查询未返回太多记录时启用此参数。例如，IN 子句子查询返回 10k 或更少的记录时，启用此参数将更快地给出查询结果。 示例：***select * from flow_carbon_256b where cus_no in (select cus_no from flow_carbon_256b where dt>='20260101' and dt<='20260701' and txn_bk='tk_1' and txn_br='tr_1') limit 1000;***
carbon.scheduler.minRegisteredResourcesRatio	0.8	启动块分布所需的最小资源（executor）比率。默认值为“0.8”，表示所请求资源的 80%被分配用于启动块分布。
carbon.dynamicAllocation.schedulerTimeout	5	此参数值指示调度器等待 executors 处于活动状态的最长时间。默认值为“5”秒，允许的最大值为“15”秒。
enable.unsafe.in.query.processing	true	指定在查询操作期间是否使用非安全排序。非安全排序减少查询操作期间的垃圾回收（GC），从而提高性能。默认值为“true”，表示启用非安全排序功能。
carbon.enable.vector.reader	true	为结果收集（result collection）启用向量处理，以增强查询性能。
carbon.query.show.datamaps	true	**SHOW TABLES** 会展示所有的表包含主表和 datamap。如果需要过滤掉 datamap，将该配置设置为 false。
二级索引配置		

参数	默认值	描述
carbon.secondary.index.creation.threads	1	该参数用于配置启动二级索引创建期间并行处理 segments 的线程数。当表的 segments 数较多时，该参数有助于微调系统生成二级索引的速度。该参数值范围为 1 到 50。
carbon.si.lookup.partialstring	true	• 当配置为 true 时，它包括开始，结尾和包含。 • 当配置为 false 时，它只包括从二级索引开始。
carbon.si.segment.merge	true	开启这个配置后会合并二级索引表 segment 内的**.carbondata** 文件。合并发生在导入操作后，在二级索引表导入操作的最后，会检查小文件并合并他们。 **说明** Table Block Size 会用作合并小文件的大小阈值。

表3-5 carbon.properties 中的其它配置

参数	默认值	描述
数据加载配置		
carbon.lock.type	HDFSLOCK	该配置指定了表上并发操作过程中所要求的锁的类型。 有以下几种类型锁实现方式： • LOCALLOCK：基于本地文件系统的文件来创建的锁。该锁只适用于一台机器上只运行一个 Spark Driver（或者 JDBCServer）的情况。 • HDFSLOCK：基于 HDFS 文件系统上的文件来创建的锁。该锁适用于集群上有多个运行的 Spark 应用而且没有可用的 ZooKeeper 的情况。
carbon.sort.intermediate.files.limit	20	中间文件的最小数量。生成中间文件后开始排序合并。此参数调优请参考性能调优的相关配置。
carbon.csv.read.buffersize.byte	1048576	CSV 读缓冲区大小。
carbon.merge.sort.reader.thread	3	用于读取中间文件进行最终合并的最大线程数。
carbon.concurrent.lock.retries	100	指定获取并发操作锁的最大重试次数。该参数用于并发加载。

参数	默认值	描述
carbon.concurrent.lock.retry.timeout.sec	1	指定获取并发操作的锁重试之间的间隔。
carbon.lock.retries	3	指定除导入操作外其他所有操作尝试获取锁的次数。
carbon.lock.retry.timeout.sec	5	指定除导入操作外其他所有操作尝试获取锁的时间间隔。
carbon.tempstore.location	/opt/Carbon/TempStoreLoc	临时存储位置。默认情况下，采用“System.getProperty("java.io.tmpdir")”方法获取。此参数调优请参考性能调优的相关配置中关于“carbon.use.local.dir”的描述。
carbon.load.log.counter	500000	数据加载记录计数日志。
SERIALIZATION_NULL_FORMAT	\N	指定需要替换为 NULL 的值。
carbon.skip.empty.line	false	设置此属性将在数据加载期间忽略 CSV 文件中的空行。
carbon.load.datamaps.parallel	false	该配置项将会开启对所有会话所有表的 datamap 并行加载。该配置项通过将导入 datamap 到内存的工作分发给所有的 executor 来缩短时间，进而提升查询性能。
合并配置		
carbon.numberof.preserve.segments	0	若用户希望从被合并的 segment 中保留一定数量的 segment，可设置该属性参数。 例如：“carbon.numberof.preserve.segments”=“2”，那么合并的 segement 中将不包含最新的 2 个 segment。 默认保留 No segment 的状态。
carbon.allowed.compaction.days	0	合并将合并在配置的指定天数中加载的 segment。 例如：如果配置值为“2”，那么只有在 2 天时间框架中加载的 segment 被合并。2 天以外被加载的 segment 不会被合并。 该参数默认为禁用。
carbon.enable.auto.load.merge	false	在数据加载时启用压缩。
carbon.merge.index.in.segm	true	如果设置，则 Segment 内的所有 Carbon 索引文件（.carbonindex）将合并为单个 Carbon 索引合并文

参数	默认值	描述
ent		件（.carbonindexmerge）。 这增强了首次查询性能
查询配置		
max.query.execution.time	60	单次查询允许的最大时间。 单位为分钟。
carbon.enable MinMax	true	MinMax 用于提高查询性能。设置为 false 可禁用该功能。
carbon.lease.recovery.retry.count	5	需要为恢复文件租约所需的最大尝试次数。 最小值：1 最大值：50
carbon.lease.recovery.retry.interval	1000 (ms)	尝试在文件上进行租约恢复之后的间隔（Interval）或暂停（Pause）时间。 最小值：1000（ms） 最大值：10000（ms）

表3-6 spark-defaults.conf 中的 Spark 配置参考

参数	默认值	描述
spark.driver.memory	4G	指定用于 driver 端进程的内存，其中 SparkContext 已初始化。 说明 在客户端模式下，不要使用 SparkConf 在应用程序中设置该参数，因为驱动程序 JVM 已经启动。要配置该参数，请在--driver-memory 命令行选项或默认属性文件中进行配置。
spark.executor.memory	4GB	指定每个执行程序进程使用的内存。
spark.sql.crossJoin.enabled	true	如果查询包含交叉连接，请启用此属性，以便不会抛出错误，此时使用交叉连接而不是连接，可实现更好的性能。

在 Spark Driver 端的“spark-defaults.conf”文件中配置以下参数。

- 在 spark-sql 模式下配置：

表3-7 spark-sql 模式下的配置参数

参数	配置值	描述
spark.driver.extraJavaOptions	-Dlog4j.configuration=file:/opt/client/Spark2x/spark/conf/log4j.properties -Djetty.version=x.y.z -Dzookeeper.server.principal=zookeeper/hadoop.*<系统域名>* -Djava.security.krb5.conf=/opt/client/KrbClient/kerberos/var/krb5kdc/krb5.conf -Djava.security.auth.login.config=/opt/client/Spark2x/spark/conf/jaas.conf -Dorg.xerial.snappy.tempdir=/opt/client/Spark2x/tmp -Dcarbon.properties.filepath=/opt/client/Spark2x/spark/conf/carbon.properties -Djava.io.tmpdir=/opt/client/Spark2x/tmp	默认值中“/opt/client/Spark2x/spark”为客户端的 CLIENT_HOME，且该默认值是追加到参数“spark.driver.extraJavaOptions”其他值之后的，此参数用于指定 Driver 端的“carbon.properties”文件路径。 说明 请注意“=”两边不要有空格。
spark.sql.session.state.builder	org.apache.spark.sql.hive.FIHiveACLSessionStateBuilder	指定会话状态构造器。
spark.carbon.sqlastbuilder.classname	org.apache.spark.sql.hive.CarbonInternalSqlAstBuilder	指定 AST 构造器。
spark.sql.catalog.class	org.apache.spark.sql.hive.HiveACLExternalCatalog	指定 Hive 的外部目录实现。启用 Spark ACL 时必须提供。
spark.sql.hive.implementation	org.apache.spark.sql.hive.HiveACLClientImpl	指定 Hive 客户端调用的实现。启用 Spark ACL 时必须提供。
spark.sql.hive.Client.isolation.enabled	false	启用 Spark ACL 时必须提供。

- 在 JDBCServer 服务中配置：

表3-8 JDBCServer 服务中的配置参数

参数	配置值	描述
spark.driver.extraJavaOptions	-Xloggc:${SPARK_LOG_DIR}/indexserver-omm-%p-gc.log -XX:+PrintGCDetail	默认值中${SPARK_CONF_DIR}需视具体的集群而定，且该默认值是追加到参数“spark.driver.extraJavaOptions”其他值之后的，此参数用于指定 Driver 端的“carbon.properties”文件路径。

参数	配置值	描述
	s -XX:-OmitStackTraceInFastThrow -XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps -XX:MaxDirectMemorySize=512M -XX:MaxMetaspaceSize=512M -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=20 -XX:GCLogFileSize=10M -XX:OnOutOfMemoryError='kill -9 %p' -Djetty.version=x.y.z -Dorg.xerial.snappy.tempdir=${BIGDATA_HOME}/tmp/spark2x/JDBCServer/snappy_tmp -Djava.io.tmpdir=${BIGDATA_HOME}/tmp/spark2x/JDBCServer/io_tmp -Dcarbon.properties.filepath=${SPARK_CONF_DIR}/carbon.properties -Djdk.tls.ephemeralDHKeySize=2048 -Dspark.ssl.keyStore=${SPARK_CONF_DIR}/child.keystore #{java_stack_prefer}	说明 请注意“=”两边不要有空格。
spark.sql.session.state.builder	org.apache.spark.sql.hive.FIHiveACLSessionStateBuilder	指定会话状态构造器。
spark.carbon.sqlastbuilder.classname	org.apache.spark.sql.hive.CarbonInternalSqlAstBuilder	指定 AST 构造器。
spark.sql.catalog.class	org.apache.spark.sql.hive.HiveACLExternalCatalog	指定 Hive 的外部目录实现。启用 Spark ACL 时必须提供。

参数	配置值	描述
spark.sql.hive.implementation	org.apache.spark.sql.hive.HiveACLClientImpl	指定 Hive 客户端调用的实现。启用 Spark ACL 时必须提供。
spark.sql.hiveClient.isolation.enabled	false	启用 Spark ACL 时必须提供。

3.3 CarbonData 操作指导

3.3.1 CarbonData 快速入门

本章节介绍创建 CarbonData table、加载数据，以及查询数据的快速入门流程。该快速入门提供基于 Spark Beeline 客户端的操作。如果使用 Spark shell，需将查询命令写在 spark.sql()的括号中。

本操作以从 CSV 文件加载数据到 CarbonData Table 为例

表3-9 CarbonData 快速入门

操作	说明
准备 CSV 文件	准备加载到 CarbonData Table 的 CSV 文件。
连接到 CarbonData	在对 CarbonData 进行任何一种操作之前，首先需要连接到 CarbonData。
创建 CarbonData Table	连接到 CarbonData 之后，需要创建 CarbonData table 用于加载数据和执行查询操作。
加载数据到 CarbonData Table	创建 CarbonData table 之后，可以从 CSV 文件加载数据到所创建的 table 中。
在 CarbonData 中查询数据	创建 CarbonData table 并加载数据之后，可以执行所需的查询操作，例如 filters，groupby 等。

准备 CSV 文件

1. 在本地准备 CSV 文件，文件名为：test.csv，样例如下：

```
13418592122,1001,MAC地址,2017-10-23 15:32:30,2017-10-24 15:32:30,62.50,74.56
13418592123,1002,MAC地址,2017-10-23 16:32:30,2017-10-24 16:32:30,17.80,76.28
13418592124,1003,MAC地址,2017-10-23 17:32:30,2017-10-24 17:32:30,20.40,92.94
13418592125,1004,MAC地址,2017-10-23 18:32:30,2017-10-24 18:32:30,73.84,8.58
13418592126,1005,MAC地址,2017-10-23 19:32:30,2017-10-24 19:32:30,80.50,88.02
13418592127,1006,MAC地址,2017-10-23 20:32:30,2017-10-24 20:32:30,65.77,71.24
13418592128,1007,MAC地址,2017-10-23 21:32:30,2017-10-24 21:32:30,75.21,76.04
```

```
13418592129,1008,MAC地址,2017-10-23 22:32:30,2017-10-24 22:32:30,63.30,94.40
13418592130,1009,MAC地址,2017-10-23 23:32:30,2017-10-24 23:32:30,95.51,50.17
13418592131,1010,MAC地址,2017-10-24 00:32:30,2017-10-25 00:32:30,39.62,99.13
```

2. 使用 WinSCP 工具将 CSV 文件导入客户端节点，例如“/opt”目录下。
3. 登录 FusionInsight Manager 页面，选择“系统 > 权限 > 用户”，添加人机用户 sparkuser，用户组（hadoop、hive），主组（hadoop）。
4. 进入客户端目录，加载环境变量并认证用户：

   cd /*客户端安装目录*

   source ./bigdata_env

   source ./Spark2x/component_env

   kinit sparkuser
5. 上传 CSV 中的文件到 HDFS 的“/data”目录：

   hdfs dfs -put /opt/test.csv /data/

连接到 CarbonData

- 使用 Spark SQL 或 Spark shell 连接到 Spark 并执行 Spark SQL 命令。
- 开启 JDBCServer 并使用 JDBC 客户端（例如，Spark Beeline）连接。

  执行如下命令：

  cd ./Spark2x/spark/bin

  ./spark-beeline

创建 CarbonData Table

在 Spark Beeline 被连接到 JDBCServer 之后，需要创建一个 CarbonData table 用于加载数据和执行查询操作。下面是创建一个简单的表的命令。

create table x1 (imei string, deviceInformationId int, mac string, productdate timestamp, updatetime timestamp, gamePointId double, contractNumber double) STORED AS carbondata TBLPROPERTIES ('SORT_COLUMNS'='imei,mac');

命令执行结果如下：

```
+---------+
| Result  |
+---------+
+---------+
No rows selected (1.093 seconds)
```

加载数据到 CarbonData Table

创建 CarbonData table 之后，可以从 CSV 文件加载数据到所创建的表中。

用所要求的参数运行以下命令从 CSV 文件加载数据。该表的列名需要与 CSV 文件的列名匹配。

LOAD DATA inpath 'hdfs://hacluster/data/*test.csv***' into table** ***x1*** **options('DELIMITER'=',', 'QUOTECHAR'='"','FILEHEADER'='imei, deviceinformationid,mac, productdate,updatetime, gamepointid,contractnumber');**

其中，“test.csv”为准备 CSV 文件的 CSV 文件，“x1”为示例的表名。

CSV 样例内容如下：

```
13418592122,1001,MAC地址,2017-10-23 15:32:30,2017-10-24 15:32:30,62.50,74.56
13418592123,1002,MAC地址,2017-10-23 16:32:30,2017-10-24 16:32:30,17.80,76.28
13418592124,1003,MAC地址,2017-10-23 17:32:30,2017-10-24 17:32:30,20.40,92.94
13418592125,1004,MAC地址,2017-10-23 18:32:30,2017-10-24 18:32:30,73.84,8.58
13418592126,1005,MAC地址,2017-10-23 19:32:30,2017-10-24 19:32:30,80.50,88.02
13418592127,1006,MAC地址,2017-10-23 20:32:30,2017-10-24 20:32:30,65.77,71.24
13418592128,1007,MAC地址,2017-10-23 21:32:30,2017-10-24 21:32:30,75.21,76.04
13418592129,1008,MAC地址,2017-10-23 22:32:30,2017-10-24 22:32:30,63.30,94.40
13418592130,1009,MAC地址,2017-10-23 23:32:30,2017-10-24 23:32:30,95.51,50.17
13418592131,1010,MAC地址,2017-10-24 00:32:30,2017-10-25 00:32:30,39.62,99.13
```

命令执行结果如下：

```
+-------------+
|Segment ID  |
+-------------+
|0          |
+-------------+
No rows selected (3.039 seconds)
```

在 CarbonData 中查询数据

创建 CarbonData table 并加载数据之后，可以执行所需的数据查询操作。以下为一些查询操作举例。

- **获取记录数**

  为了获取在 CarbonData table 中的记录数，可以运行以下命令。

  select count(*) from x1;

- **使用 Groupby 查询**

  为了获取不重复的 deviceinformationid 记录数，可以运行以下命令。

  select deviceinformationid,count (distinct deviceinformationid) from x1 group by deviceinformationid;

- **用 Filter 查询**

  为了获取特定 deviceinformationid 的记录，可以运行以下命令。

  select * from x1 where deviceinformationid='1010';

说明

在执行数据查询操作后，如果查询结果中某一列的结果含有中文字等非英文字符，会导致查询结果中的列不能对齐，这是由于不同语言的字符在显示时所占的字宽不尽相同。

在 Spark-shell 上使用 CarbonData

用户若需要在 Spark-shell 上使用 CarbonData，需通过如下方式创建 CarbonData Table，加载数据到 CarbonData Table 和在 CarbonData 中查询数据的操作。

```
spark.sql("CREATE TABLE x2(imei string, deviceInformationId int, mac string,
productdate timestamp, updatetime timestamp, gamePointId double, contractNumber
double) STORED AS carbondata")
```

```
spark.sql("LOAD DATA inpath 'hdfs://hacluster/data/x1_without_header.csv' into
table x2 options('DELIMITER'=',', 'QUOTECHAR'='\"','FILEHEADER'='imei,
deviceinformationid,mac, productdate,updatetime, gamepointid,contractnumber')")
spark.sql("SELECT * FROM x2").show()
```

3.3.2 管理 CarbonData Table

3.3.2.1 CarbonData Table 简介

简介

CarbonData 中的数据存储在 table 实体中。CarbonData table 与 RDBMS 中的表类似。RDBMS 数据存储在由行和列构成的表中。CarbonData table 存储的也是结构化的数据，拥有固定列和数据类型。

支持数据类型

CarbonData 支持以下数据类型：

- Int
- String
- BigInt
- Smallint
- Char
- Varchar
- Boolean
- Decimal
- Double
- TimeStamp
- Date
- Array
- Struct
- Map

下表对所支持的数据类型及其各自的范围进行了详细说明。

表3-10 CarbonData 数据类型

数据类型	范围
Int	4 字节有符号整数，从-2,147,483,648 到 2,147,483,647 说明 非字典列如果是 Int 类型，会在内部存储为 BigInt 类型。
String	100000 字符 说明 如果在 **CREATE TABLE** 中使用 Char 或 Varchar 数据类型，则这

数据类型	范围
	两种数据类型将自动转换为 String 数据类型。 如果存在字符长度超过 32000 的列，需要在建表时，将该列加入到 tblproperties 的 LONG_STRING_COLUMNS 属性里。
BigInt	64-bit，从-9,223,372,036,854,775,808 到 9,223,372,036,854,775,807
SmallInt	范围-32,768 到 32,767
Char	范围 A 到 Z&a 到 z
Varchar	范围 A 到 Z&a 到 z&0 到 9
Boolean	范围 true 或者 false
Decimal	默认值是(10,0) ，最大值是(38,38) 说明 当进行带过滤条件的查询时，为了得到准确的结果，需要在数字后面加上 BD。例如，*select * from carbon_table where num = 1234567890123456.22BD*.
Double	64-bit，从 4.9E-324 到 1.7976931348623157E308
TimeStamp	NA，默认格式为“yyyy-MM-dd HH:mm:ss”。
Date	DATE 数据类型用于存储日历日期。 默认格式为“yyyy-MM-dd”。
Array<data_type>	NA 说明 现仅支持 2 层复杂类型的嵌套。
Struct<col_name: data_type COMMENT col_comment, ...>	
Map<primitive_type, data_type>	

3.3.2.2 新建 CarbonData Table

操作场景

使用 CarbonData 前需先创建表，才可在其中加载数据和查询数据。可通过 **Create Table** 命令来创建表。该命令支持使用自定义列创建表。

使用自定义列创建表

可通过指定各列及其数据类型来创建表。

命令示例：

CREATE TABLE *IF NOT EXISTS productdb.productSalesTable (*

productNumber Int,

productName String,

storeCity String,

storeProvince String,

productCategory String,

productBatch String,

saleQuantity Int,

revenue Int)

STORED AS *carbondata*

TBLPROPERTIES (

'table_blocksize'='128');

上述命令所创建的表的详细信息如下：

表3-11 表信息定义

参数	描述
productSalesTable	待创建的表的名称。该表用于加载数据进行分析。 表名由字母、数字、下划线组成。
productdb	数据库名称。该数据库将与其中的表保持逻辑连接以便于识别和管理。 数据库名称由字母、数字、下划线组成。
productName storeCity storeProvince procuctCategory productBatch saleQuantity revenue	表中的列，代表执行分析所需的业务实体。 列名（字段名）由字母、数字、下划线组成。
table_blocksize	CarbonData 表使用的数据文件的 block 大小，默认值为 1024，最小值为 1，最大值为 2048，单位为 MB。 如果“table_blocksize”值太小，数据加载时，生成过多的小数据文件，可能会影响 HDFS 的使用性能。 如果“table_blocksize”值太大，数据查询时，索引匹配的 block 数据量较大，某些 block 会包含较多的 blocklet，导致读取并发度不高，从而降低查询性能。 一般情况下，建议根据数据量级别来选择大小。例如：GB 级别用 256，TB 级别用 512，PB 级别用 1024。

说明

- 所有 Integer 类型度量均以 BigInt 类型进行处理与显示。
- CarbonData 遵循严格解析，因此任何不可解析的数据都会被保存为 null。例如，在 BigInt 列中加载 double 值（3.14），将会保存为 null。
- 在 Create Table 中使用的 Short 和 Long 数据类型在 DESCRIBE 命令中分别显示为 Smallint 和 Bigint。
- 可以使用 DESCRIBE 格式化命令查看表数据大小和表索引大小。

操作结果

根据命令创建表。

3.3.2.3 删除 CarbonData Table

操作场景

可使用 **DROP TABLE** 命令删除表。删除表后，所有 metadata 以及表中已加载的数据都会被删除。

操作步骤

运行如下命令删除表。

命令：

DROP TABLE *[IF EXISTS] [db_name.]table_name;*

一旦执行该命令，将会从系统中删除表。命令中的“db_name”为可选参数。如果没有指定“db_name”，那么将会删除当前数据库下名为“table_name”的表。

示例：

DROP TABLE *productdb.productSalesTable;*

通过上述命令，删除数据库“productdb”下的表“productSalesTable”。

操作结果

从系统中删除命令中指定的表。删除完成后，可通过 **SHOW TABLES** 命令进行查询，确认所需删除的表是否成功被删除，详见 SHOW TABLES。

3.3.2.4 修改 CarbonData Table

SET 和 UNSET

当使用 set 命令时，所有新 set 的属性将会覆盖已存在的旧的属性。

- SORT SCOPE

SET SORT SCOPE 命令示例：

ALTER TABLE *tablename* **SET TBLPROPERTIES('SORT_SCOPE'**=*'no_sort'*)

当 UNSET SORT SCOPE 后，会使用默认值 NO_SORT。

UNSET SORT SCOPE 命令示例：

ALTER TABLE *tablename* **UNSET TBLPROPERTIES('SORT_SCOPE'**)

- SORT COLUMNS

  SET SORT COLUMNS 命令示例：

  ALTER TABLE *tablename* **SET TBLPROPERTIES('SORT_COLUMNS'**=*'column1'*)

  在执行该命令后，新的导入会使用新的 SORT_COLUMNS 配置值。 用户可以根据查询的情况来调整 SORT_COLUMNS，但是不会直接影响旧的数据。所以对历史的 segments 的查询性能不会受到影响，因为历史的 segments 不是按照新的 SORT_COLUMNS。

  不支持 UNSET 命令，但是可以使用 set SORT_COLUMNS 等于空字符串来代替 UNSET 命令。

  ALTER TABLE *tablename* **SET TBLPROPERTIES('SORT_COLUMNS'**=*''*)

说明

- 后续版本会加强自定义合并来对旧的 segment 重新排序。
- 流式表不支持修改 SORT_COLUMNS。
- 如果 inverted index 的列从 SORT_COLUMNS 里面移除了，该列不会再创建 inverted index。但是旧的 INVERTED_INDEX 配置值不会变化。

3.3.3 管理 CarbonData Table 数据

3.3.3.1 加载数据

操作场景

CarbonData table 创建成功后，可使用 **LOAD DATA** 命令在表中加载数据，并可供查询。触发数据加载后，数据以 CarbonData 格式进行编码，并将多维列式存储格式文件压缩后复制到存储 CarbonData 文件的 HDFS 路径下供快速分析查询使用。HDFS 路径可以配置在 carbon.properties 文件中。具体请参考配置参考。

3.3.3.2 删除 Segments

操作场景

如果用户将错误数据加载到表中，或者数据加载后出现许多错误记录，用户希望修改并重新加载数据时，可删除对应的 segment。可使用 segment ID 来删除 segment，也可以使用加载数据的时间来删除 segment。

说明

删除 segment 操作只能删除未合并的 segment，已合并的 segment 可以通过 **CLEAN FILES** 命令清除 segment。

通过 Segment ID 删除

每个 Segment 都有与其关联的唯一 Segment ID。使用这个 Segment ID 可以删除该 Segment。

步骤 1 运行如下命令获取 Segment ID。

命令：

SHOW SEGMENTS FOR Table *dbname.tablename LIMIT number_of_loads;*

示例：

SHOW SEGMENTS FOR TABLE *carbonTable;*

上述命令可显示 tablename 为 carbonTable 的表的所有 Segment 信息。

SHOW SEGMENTS FOR TABLE *carbonTable LIMIT 2;*

上述命令可显示 *number_of_loads* 规定条数的 Segment 信息。

输出结果如下：

```
+-----+----------+--------------------------+-----------------+------------+------
------+-------------+--------------+--+
| ID  |  Status  |    Load Start Time       | Load Time Taken  | Partition  | Data Size
| Index Size  | File Format  |
+-----+----------+--------------------------+-----------------+------------+------
------+-------------+--------------+--+
| 3   | Success  | 2020-09-28 22:53:26.336  | 3.726S          | {}         | 6.47KB
| 3.30KB      | columnar v3  |
| 2   | Success  | 2020-09-28 22:53:01.702  | 6.688S          | {}         | 6.47KB
| 3.30KB      | columnar v3  |
+-----+----------+--------------------------+-----------------+------------+------
------+-------------+--------------+--+
```

说明

SHOW SEGMENTS 命令输出包括 ID、Status、Load Start Time、Load Time Taken、Partition、Data Size、Index Size、File Format。最新的加载信息在输出中第一行显示。

步骤 2 获取到需要删除的 Segment 的 Segment ID 后，执行如下命令删除对应 Segment：

命令：

DELETE FROM TABLE tableName WHERE SEGMENT.ID IN (load_sequence_id1, load_sequence_id2, ...);

示例：

DELETE FROM TABLE carbonTable WHERE SEGMENT.ID IN (1,2,3);

详细信息，请参阅 DELETE SEGMENT by ID。

----结束

通过加载数据的时间删除

用户可基于特定的加载时间删除数据。

命令：

DELETE FROM TABLE db_name.table_name WHERE SEGMENT.STARTTIME BEFORE date_value;

示例：

DELETE FROM TABLE carbonTable WHERE SEGMENT.STARTTIME BEFORE '2017-07-01 12:07:20';

上述命令可删除'2017-07-01 12:07:20'之前的所有 segment。

有关详细信息，请参阅 DELETE SEGMENT by DATE。

删除结果

数据对应的 segment 被删除，数据将不能再被访问。可通过 **SHOW SEGMENTS** 命令显示 segment 状态，查看是否成功删除。

说明

- 调用 **DELETE SEGMENT** 命令时，物理上而言，Segment 并没有从文件系统中被删除。使用命令 **SHOW SEGMENTS** 查看 Segment 信息，可看见被删除的 Segment 的状态被标识为 "Marked for Delete"。但使用 **SELECT * FROM tablename** 命令查询时，不会显示被删除的 Segment 的内容。
- 下一次加载数据且达到最大查询执行时间（由“max.query.execution.time”配置，默认为“60分钟”）后，Segment 才会从文件系统中真正删除。
- 如果用户想要强制删除物理 Segment 文件，那么可以使用 **CLEAN FILES** 命令。

  示例：

  CLEAN FILES FOR TABLE table1;

  该命令将从物理上删除状态为“Marked for delete”的 Segment 文件。

  如果在“max.query.execution.time”规定的时间到达之前使用该命令，可能会导致查询失败。“max.query.execution.time”可在“carbon.properties”文件中设置，表示一次查询允许花费的最长时间。

3.3.3.3 合并 Segments

操作场景

频繁的数据获取导致在存储目录中产生许多零碎的 CarbonData 文件。由于数据排序只在每次加载时进行，所以，索引也只在每次加载时执行。这意味着，对于每次加载都会产生一个索引，随着数据加载数量的增加，索引的数量也随之增加。由于每个索引只在一次加载时工作，索引的性能被降低。CarbonData 提供加载压缩。压缩过程通过合并排序各 segment 中的数据，将多个 segment 合并为一个大的 segment。

前提条件

已经加载了多次数据。

操作描述

有 Minor 合并、Major 合并和 Custom 合并三种类型。

- Minor 合并：

  在 Minor 合并中，用户可指定合并数据加载的数量。如果设置了参数“carbon.enable.auto.load.merge”，每次数据加载都可触发 Minor 合并。如果任意 segment 均可合并，那么合并将于数据加载时并行进行。

  Minor 合并有两个级别。

  - Level 1：合并未合并的 segment。
  - Level 2：合并已合并的 segment，以形成更大的 segment。

- Major 合并：

  在 Major 合并中，许多 segment 可以合并为一个大的 segment。用户将指定合并尺寸，将对未达到该尺寸的 segment 进行合并。Major 合并通常在非高峰时段进行。

- Custom 合并：

  在 Custom 合并中，用户可以指定几个 segment 的 id 合并为一个大的 segment。所有指定的 segment 的 id 必须存在并且有效，否则合并将会失败。Custom 合并通常在非高峰时段进行。

具体的命令操作，请参考 ALTER TABLE COMPACTION。

表3-12 合并参数

参数	默认值	应用类型	描述
carbon.enable.auto.load.merge	false	Minor	数据加载时启用合并。 “true”：数据加载时自动触发 segment 合并。 “false”：数据加载时不触发 segment 合并。
carbon.compaction.level.threshold	4,3	Minor	对于 Minor 合并，该属性参数决定合并 segment 的数量。 例如，如果该参数设置为“2,3”，在 Level 1，每 2 个 segment 触发一次 Minor 合并。在 Level2，每 3 个 Level 1 合并的 segment 将被再次合并为新的 segment。 合并策略根据实际的数据大小和可用资源决定。 有效值为 0-100。
carbon.major.compaction.size	1024mb	Major	通过配置该参数可配置 Major 合并。低于该阈值的 segment 之和将被合并。 例如，如果该阈值是 1024MB，且有 5 个大小依次为 300MB，400MB，500MB，200MB，100MB 的 segment

参数	默认值	应用类型	描述
			用于 Major 合并，那么只有相加的总数小于阈值的 segment 会被合并，也就是 300+400+200+100 = 1000MB 的 segment 会被合并，而 500MB 的 segment 将会被跳过。
carbon.numberof.preserve.segments	0	Minor/Major	如果用户希望从被合并的 segment 中保留一定数量的 segment，可通过该属性参数进行设置。 例如，“carbon.numberof.preserve.segments”=“2”，那么最新的 2 个 segment 将不会包含在合并中。 默认不保留任何 segment。
carbon.allowed.compaction.days	0	Minor/Major	合并将合并在指定的配置天数中加载的 segment。 例如，如果配置为“2”，那么只有在 2 天的时间框架中被加载的 segment 可以被合并。在 2 天以外被加载的 segment 将不被合并。 默认为禁用。
carbon.number.of.cores.while.compacting	2	Minor/Major	在合并过程中写入数据时所用的核数。配置的核数越大合并性能越好。如果 CPU 资源充足可以增加此值。
carbon.merge.index.in.segment	true	SEGMENT_INDEX	如果设置为 true，则一个 segment 中所有 Carbon 索引文件（.carbonindex）将合并为单个 Carbon 索引合并文件（.carbonindexmerge）。 这增强了首次查询性能。

参考信息

建议避免对历史数据进行 minor compaction，请参考如何避免对历史数据进行 minor compaction？

3.3.4 迁移 CarbonData 数据

操作场景

如果用户需要快速从一个集群中将 CarbonData 的数据迁移到另外一个集群的 CarbonData 中，可以使用 CarbonData 的数据备份与恢复命令来完成该任务。使用此方法迁移数据，无需在新集群执行数据导入的过程，可以减少迁移的时间。

前提条件

两个集群已安装 Spark2x 客户端，例如安装目录为“/opt/client”。假设原始数据所在集群为 A，需要迁移到集群 B。

操作步骤

步骤 1 使用客户端安装用户登录 A 集群客户端所在节点。

步骤 2 使用客户端用户执行以下命令，配置环境变量。

source /opt/client/bigdata_env

source /opt/client/Spark2x/component_env

步骤 3 如果集群为安全模式，执行以下命令进行用户认证。普通模式集群无需执行用户认证。

kinit *carbondatauser*

carbondatauser 需要为原始数据的使用者，即拥有表的读写权限。

说明

该用户需要加入 hadoop，hive 组，主组选择 hadoop 组，并关联角色 System_administrator。

步骤 4 执行以下命令连接数据库，并查看表的数据在 HDFS 保存的位置：

spark-beeline

desc formatted ***原始数据的表名称*****;**

查看系统显示的信息中“Location”表示数据文件所在目录。

步骤 5 使用客户端安装用户登录 B 集群客户端所在节点，并执行以下命令，配置环境变量。

source /opt/client/bigdata_env

source /opt/client/Spark2x/component_env

步骤 6 如果集群为安全模式，执行以下命令进行用户认证。普通模式集群无需执行用户认证。

kinit *carbondatauser2*

carbondatauser2 需要为上传数据的用户。

说明

该用户需要加入 hadoop，hive 组，主组选择 hadoop 组，并关联角色 System_administrator。

步骤 7 执行 **spark-beeline** 命令连接数据库。

步骤 8 原始数据对应的数据库是否存在？

- 是，执行步骤 9。
- 否，执行命令 **create database** *数据库名称*，创建一个同名的数据库，然后执行步骤 9。

步骤 9 将原始数据从集群 A 的 HDFS 目录中，拷贝数据到集群 B 的 HDFS 中。

在集群 B 上传数据时，上传目录中需要存在与原始目录有相同的数据库以及表名目录，且上传用户需要有在此目录写入数据的权限。上传后该用户将拥有数据的读写权限。

例如，原始数据保存在“/user/carboncadauser/warehouse/db1/tb1”，则在新集群中数据可以保存在“/user/carbondatauser2/warehouse/db1/tb1”中：

1. 下载原始数据到集群 A 的“/opt/backup”目录下：

   hdfs dfs -get /user/carboncadauser/warehouse/db1/tb1 /opt/backup

2. 拷贝集群 A 的原始数据到集群 B 客户端节点的“/opt/backup”目录下：

   scp /opt/backup root@*集群 B 客户端节点 IP*:/opt/backup

3. 上传拷贝到集群 B 的数据到 HDFS 中：

   hdfs dfs -put /opt/backup /user/carbondatauser2/warehouse/db1/tb1

步骤 10 在集群 B 的客户端环境执行以下命令，在 Hive 中生成原始数据对应的表所关联的元数据：

REFRESH TABLE *$dbName.$tbName;*

$dbName 和$tbName 分别表示数据对应的数据库名称以及表名称。

步骤 11 如果原表存在索引表，执行步骤 9 和步骤 10，将集群 A 的索引表目录迁移到集群 B。

步骤 12 执行以下命令，为 CarbonData 表注册索引表（注意，如果原表没有创建索引表，则不需要执行此步骤）：

REGISTER INDEX TABLE *$tableName* ON *$maintable*;

$tableName 和$maintable 分别表示索引表名称和主表名称。

----结束

3.3.5 迁移 Spark1.5 的 Carbondata 数据到 Spark2x 的 Carbondata 中

迁移方案概览

本次迁移目标是将 Spark1.5 的 CarbonData 表数据迁移到 Spark2x 的 CarbonData 表中。

说明

执行本操作前需要将 spark1.5 的 carbondata 表入库业务中断，将数据一次性迁移至 spark2x 的 carbondata 表，完成迁移后使用 spark2x 进行业务操作。

迁移思路：

1. 先通过 Spark1.5 将历史数据迁移至中间表。
2. 再通过 Spark2x 将中间表的数据迁移至目标表，然后将目标表名修改为原表名。
3. 迁移完成后使用 Spark2x 操作 CarbonData 表中的数据。

具体迁移方案和命令

历史数据迁移

步骤 1 中断 CarbonData 的数据入库业务，用 Spark1.5 的 spark-beeline，查看 CarbonData 表当前最新的 Segment 的 ID 和时间，并记录此 Segment 的 ID。

show segments for table dbname.tablename;

步骤 2 用创建原 CarbonData 表的用户，执行 Spark1.5 的 spark-beeline，创建 ORC（或 PARQUET）格式的中间表，然后将原 CarbonData 表中的数据导入该中间表，导入完成后即可恢复 CarbonData 表的业务。

创建 ORC 表：

CREATE TABLE dbname.mid_tablename_orc STORED AS ORC as select * from dbname.tablename;

创建 PARQUET 表：

CREATE TABLE dbname.mid_tablename_parq STORED AS PARQUET as select * from dbname.tablename;

其中 dbname 指数据库名，tablename 指原 CarbonData 表名。

步骤 3 用创建原 CarbonData 表的用户，执行 Spark2x 的 spark-beeline；然后通过旧表的建表语句，创建新的 CarbonData 表。

说明

创建新表的语句中，字段顺序和类型必须和旧表的完全一致，如此才能保留原表的索引列等结构，且可以避免后续插入数据时因用到了 select * 而出错。

通过 Spark1.5 的 spark-beeline 命令查看旧表的建表语句：**SHOW CREATE TABLE dbname.tablename**;

创建新 CarbonData 表，名称为：**dbname.new_tablename**

步骤 4 用创建原 CarbonData 表的用户，执行 Spark2x 的 spark-beeline，将步骤 2 中创建的 ORC（或 PARQUET）格式的中间表的数据，加载到步骤 3 创建的新表中。此步骤可能耗时较长（200G 数据耗时约 2 小时）。加载数据的命令以 ORC 格式的中间表举例：

insert into dbname.new_tablename select *

from dbname. mid_tablename_orc;

步骤 5 用创建原 CarbonData 表的用户，执行 Spark2x 的 spark-beeline，对新表的数据进行查询检验，确认数据无误后，将原 CarbonData 表修改为其他名称，再将新 CarbonData 表修改为原 CarbonData 表的名称。

ALTER TABLE dbname.tablename RENAME TO dbname.old_tablename;

ALTER TABLE dbname.new_tablename RENAME TO dbname.tablename;

步骤 6 迁移完成。此时即可通过 Spark2x 对新表进行查询、重建二级索引等操作。

----结束

3.4 CarbonData 性能调优

3.4.1 调优指导

查询性能调优

CarbonData 可以通过调整各种参数来提高查询性能。大部分参数聚焦于增加并行性处理和更好地使用系统资源。

- Spark Executor 数量：Executor 是 Spark 并行性的基础实体。通过增加 Executor 数量，集群中的并行数量也会增加。关于如何配置 Executor 数量，请参考 Spark 资料。
- Executor 核：每个 Executor 内，并行任务数受 Executor 核的配置控制。通过增加 Executor 核数，可增加并行任务数，从而提高性能。
- HDFS block 容量：CarbonData 通过给不同的处理器分配不同的 block 来分配查询任务。所以一个 HDFS block 是一个分区单元。另外，CarbonData 在 Spark 驱动器中，支持全局 block 级索引，这有助于减少需要被扫描的查询 block 的数量。设置较大的 block 容量，可提高 I/O 效率，但是会降低全局索引效率；设置较小的 block 容量，意味着更多的 block 数量，会降低 I/O 效率，但是会提高全局索引效率，同时，对于索引查询会要求更多的内存。
- 扫描线程数量：扫描仪（Scanner）线程控制每个任务中并行处理的数据块的数量。通过增加扫描仪线程数，可增加并行处理的数据块的数量，从而提高性能。可使用“carbon.properties”文件中的“carbon.number.of.cores”属性来配置扫描仪线程数。例如，“carbon.number.of.cores = 4”。
- B-Tree 缓存：为了获得更好的查询特性，可以通过 B-tree LRU（least recently used，最近最少使用）缓存来优化缓存内存。在 driver 中，B-Tree LRU 缓存配置将有助于通过释放未被访问或未使用的表 segments 来释放缓存。类似地，在 executor 中，B-Tree LRU 缓存配置将有助于释放未被访问或未使用的表 blocks。具体可参考表 3-4 中的参数“carbon.max.driver.lru.cache.size”和“carbon.max.executor.lru.cache.size”的详细描述。

CarbonData 查询流程

当 CarbonData 首次收到对某个表（例如表 A）的查询任务时，系统会加载表 A 的索引数据到内存中，执行查询流程。当 CarbonData 再次收到对表 A 的查询任务时，系统则不需要再加载其索引数据。

在 CarbonData 中执行查询时，查询任务会被分成几个扫描任务。即，基于 CarbonData 数据存储的 HDFS block 对扫描任务进行分割。扫描任务由集群中的执行器执行。扫描任务可以并行、部分并行，或顺序处理，具体采用的方式取决于执行器的数量以及配置的执行器核数。

查询任务的某些部分可在独立的任务级上处理，例如 select 和 filter。查询任务的某些部分可在独立的任务级上进行部分处理，例如 group-by、count、distinct count 等。

某些操作无法在任务级上处理，例如 Having Clause（分组后的过滤），sort 等。这些无法在任务级上处理，或只能在任务级上部分处理的操作需要在集群内跨执行器来传输数据（部分结果）。这个传送操作被称为 shuffle。

任务数量越多，需要 shuffle 的数据就越多，会对查询性能产生不利影响。

由于任务数量取决于 HDFS block 的数量，而 HDFS block 的数量取决于每个 block 的大小，因此合理选择 HDFS block 的大小很重要，需要在提高并行性，进行 shuffle 操作的数据量和聚合表的大小之间达到平衡。

分割和 Executors 的关系

如果分割数小于等于 Executor 数乘以 Executor 核数，那么任务将以并行方式运行。否则，某些任务只有在其他任务完成之后才能开始。因此，要确保 Executor 数乘以 Executor 核数大于等于分割数。同时，还要确保有足够的分割数，这样一个查询任务可被分为足够多的子任务，从而确保并行性。

配置扫描仪线程

扫描仪线程属性决定了每个分割的数据被划分的可并行处理的数据块的数量。如果数量过多，会产生很多小数据块，性能会受到影响。如果数量过少，并行性不佳，性能也会受到影响。因此，决定扫描仪线程数时，最好考虑一个分割内的平均数据大小，选择一个使数据块不会很小的值。经验法则是将单个块大小（MB）除以 250 得到的值作为扫描仪线程数。

增加并行性还需考虑的重要一点是集群中实际可用的 CPU 核数，确保并行计算数不超过实际 CPU 核数的 75%至 80%。

CPU 核数约等于：

并行任务数 x 扫描仪线程数。其中并行任务数为分割数和执行器数 x 执行器核数两者之间的较小值。

数据加载性能调优

数据加载性能调优与查询性能调优差异很大。跟查询性能一样，数据加载性能也取决于可达到的并行性。在数据加载情况下，工作线程的数量决定并行的单元。因此，更多的执行器就意味着更多的执行器核数，每个执行器都可以提高数据加载性能。

同时，为了得到更好的性能，可在 HDFS 中配置如下参数。

表3-13 HDFS 配置

参数	建议值
dfs.datanode.drop.cache.behind.reads	false
dfs.datanode.drop.cache.behind.writes	false
dfs.datanode.sync.behind.writes	true

压缩调优

CarbonData 结合少数轻量级压缩算法和重量级压缩算法来压缩数据。虽然这些算法可处理任何类型的数据，但如果数据经过排序，相似值在一起出现时，就会获得更好的压缩率。

CarbonData 数据加载过程中，数据基于 Table 中的列顺序进行排序，从而确保相似值在一起出现，以获得更好的压缩率。

由于 CarbonData 按照 Table 中定义的列顺序将数据进行排序，因此列顺序对于压缩效率起重要作用。如果低 cardinality 维度位于左边，那么排序后的数据分区范围较小，压缩效率较高。如果高 cardinality 维度位于左边，那么排序后的数据分区范围较大，压缩效率较低。

内存调优

CarbonData 为内存调优提供了一个机制，其中数据加载会依赖于查询中需要的列。不论何时，接收到一个查询命令，将会获取到该查询中的列，并确保内存中这些列有数据加载。在该操作期间，如果达到内存的阈值，为了给查询需要的列提供内存空间，最少使用加载级别的文件将会被删除。

3.4.2 创建 CarbonData Table 的建议

操作场景

本章节根据超过 50 个测试用例总结得出建议，帮助用户创建拥有更高查询性能的 CarbonData 表。

表3-14 CarbonData 表中的列

Column name	Data type	Cardinality	Attribution
msisdn	String	3 千万	dimension
BEGIN_TIME	bigint	1 万	dimension
host	String	1 百万	dimension
dime_1	String	1 千	dimension
dime_2	String	500	dimension
dime_3	String	800	dimension
counter_1	numeric(20,0)	NA	measure
...	...	NA	measure
counter_100	numeric(20,0)	NA	measure

操作步骤

- 如果待创建的表有一个常用于过滤的列 ，例如 80%以上的场景使用此列过滤。

  针对此类场景，调优方法如下：

  将常用于过滤的列放在 sort_columns 第一列。

  例如，msisdn 作为过滤条件在查询中使用的最多，则将其放在第一列。创建表的命令如下，其中采用 msisdn 作为过滤条件的查询性能将会很好。

  ```
  create table carbondata_table(
     msisdn String,
     ...
     )STORED AS carbondata TBLPROPERTIES ('SORT_COLUMS'='msisdn');
  ```

- 如果待创建的表有多个常用于过滤的列。

  针对此类场景，调优方法如下：

  为常用的过滤列创建索引。

  例如，如果 msisdn，host 和 dime_1 是过滤经常使用的列，根据 cardinality，sort_columns 列的顺序是 dime_1-> host-> msisdn…。创建表命令如下，以下命令可提高 dime_1，host 和 msisdn 上的过滤性能。

  ```
  create table carbondata_table(
     dime_1 String,
     host String,
     msisdn String,
     dime_2 String,
     dime_3 String,
     ...
     )STORED AS carbondata
  TBLPROPERTIES ('SORT_COLUMS'='dime_1,host,msisdn');
  ```

- 如果每个用于过滤的列的频率相当。

  针对此类场景，调优方法如下：

  sort_columns 按照 cardinality 从低到高的顺序排列。

  创建表的命令如下：

  ```
  create table carbondata_table(
     Dime 1 String,
     BEGIN TIME bigint,
     HOST String,
     MSISDN String,
     ...
     )STORED AS carbondata
  TBLPROPERTIES ('SORT_COLUMS'='dime_2,dime_3,dime_1, BEGIN_TIME,host,msisdn');
  ```

- 按照维度的 cardinality 从低到高创建表后，再为高 Cardinality 列创建 SECONDARY INDEX。创建索引的语句如下：

  ```
  create index carbondata_table_index_msidn on tablecarbondata_table (
  MSISDN String) as 'carbondata' PROPERTIES ('table_blocksize'='128');
  create index carbondata_table_index_host on tablecarbondata_table (
  host String) as 'carbondata' PROPERTIES ('table_blocksize'='128');
  ```

- 对于不需要高精度的度量，无需使用 numeric (20,0)数据类型，建议使用 double 数据类型来替换 numeric (20,0)数据类型，以提高查询性能。

在一个测试用例中，使用 double 来替换 numeric (20, 0)，查询时间从 15 秒降低到 3 秒，查询速度提高了 5 倍。创建表命令如下：

```
create table carbondata_table(
    Dime_1 String,
    BEGIN_TIME bigint,
    HOST String,
    MSISDN String,
    counter_1 double,
    counter_2 double,
    ...
    counter_100 double,
    )STORED AS carbondata
;
```

- 如果列值总是递增的，如 start_time。

  例如，每天将数据加载到 CarbonData，start_time 是每次加载的增量。对于这种情况，建议将 start_time 列放在 sort_columns 的最后，因为总是递增的值可以始终使用最小/最大索引。创建表命令如下：

```
create table carbondata_table(
    Dime_1 String,
    HOST String,
    MSISDN String,
    counter_1 double,
    counter_2 double,
    BEGIN_TIME bigint,
    ...
    counter_100 double,
    )STORED AS carbondata
    TBLPROPERTIES ( 'SORT_COLUMS'='dime_2,dime_3,dime_1..BEGIN_TIME');
```

3.4.3 性能调优的相关配置

操作场景

CarbonData 的性能与配置参数相关，本章节提供了能够提升性能的相关配置介绍。

操作步骤

用于 CarbonData 查询的配置介绍，详情请参见表 3-15 和表 3-16。

表3-15 Shuffle 过程中，启动 Task 的个数

参数	spark.sql.shuffle.partitions
所属配置文件	spark-defaults.conf
适用于	数据查询
场景描述	Spark shuffle 时启动的 Task 个数。
如何调优	一般建议将该参数值设置为执行器核数的 1 到 2 倍。例如，在聚合场景中，将 task 个数从 200 减少到 32，有些查询的性能可提升

	2 倍。

表3-16 设置用于 CarbonData 查询的 Executor 个数、CPU 核数以及内存大小

参数	spark.executor.cores spark.executor.instances spark.executor.memory
所属配置文件	spark-defaults.conf
适用于	数据查询
场景描述	设置用于 CarbonData 查询的 Executor 个数、CPU 核数以及内存大小。
如何调优	在银行方案中，为每个执行器提供 4 个 CPU 内核和 15GB 内存，可以获得良好的性能。这 2 个值并不意味着越多越好，在资源有限的情况下，需要正确配置。例如，在银行方案中，每个节点有足够的 32 个 CPU 核，而只有 64GB 的内存，这个内存是不够的。例如，当每个执行器有 4 个内核和 12GB 内存，有时在查询期间发生垃圾收集（GC），会导致查询时间从 3 秒增加到超过 15 秒。在这种情况下需要增加内存或减少 CPU 内核。

用于 CarbonData 数据加载的配置参数，详情请参见表 3-17、表 3-18 和表 3-19。

表3-17 设置数据加载使用的 CPU core 数量

参数	carbon.number.of.cores.while.loading
所属配置文件	carbon.properties
适用于	数据加载
场景描述	数据加载过程中，设置处理数据使用的 CPU core 数量。
如何调优	如果有更多的 CPU 个数，那么可以增加 CPU 值来提高性能。例如，将该参数值从 2 增加到 4，那么 CSV 文件读取性能可以增加大约 1 倍。

表3-18 是否使用 YARN 本地目录进行多磁盘数据加载

参数	carbon.use.local.dir
所属配置文件	carbon.properties
适用于	数据加载
场景描述	是否使用 YARN 本地目录进行多磁盘数据加载。

如何调优	如果将该参数值设置为“true”，CarbonData 将使用 YARN 本地目录进行多表加载磁盘负载平衡，以提高数据加载性能。

表3-19 加载时是否使用多路径

参数	carbon.use.multiple.temp.dir
所属配置文件	carbon.properties
适用于	数据加载
场景描述	是否使用多个临时目录存储 sort 临时文件。
如何调优	设置为 true，则数据加载时使用多个临时目录存储 sort 临时文件。此配置能提高数据加载性能并避免磁盘单点故障。

用于 CarbonData 数据加载和数据查询的配置参数，详情请参见表 3-20。

表3-20 设置数据加载和查询使用的 CPU core 数量

参数	carbon.compaction.level.threshold
所属配置文件	carbon.properties
适用于	数据加载和查询
场景描述	对于 minor 压缩，在阶段 1 中要合并的 segment 数量和在阶段 2 中要合并的已压缩的 segment 数量。
如何调优	每次 CarbonData 加载创建一个 segment，如果每次加载的数据量较小，将在一段时间内生成许多小文件，影响查询性能。配置该参数将小的 segment 合并为一个大的 segment，然后对数据进行排序，可提高查询性能。 压缩的策略根据实际的数据大小和可用资源决定。如某银行 1 天加载一次数据，且加载数据选择在晚上无查询时进行，有足够的资源，压缩策略可选择为 6、5。

表3-21 使用索引缓存服务器时是否开启数据预加载

参数	carbon.indexserver.enable.prepriming
所属配置文件	carbon.properties
适用于	数据加载
场景描述	使用索引缓存服务器过程中开启数据预加载可以提升首次查询的性能。

如何调优	用户可以将该参数设置为 true 来开启预加载。默认情况，该参数为 false。

3.5 CarbonData 访问控制

下表提供了对 CarbonData Table 执行相应操作所需的 Hive ACL 特权的详细信息。

前提条件

已经设置了表 3-7 或表 3-8 中 Carbon 相关参数。

Hive ACL 权限

表3-22 CarbonData 表级操作所需的 Hive ACL 权限

场景	所需权限
DESCRIBE TABLE	SELECT (of table)
SELECT	SELECT (of table)
EXPLAIN	SELECT (of table)
CREATE TABLE	CREATE (of database)
CREATE TABLE As SELECT	CREATE (on database), INSERT (on table), RW on data file, and SELECT (on table)
LOAD	INSERT (of table) RW on data file
DROP TABLE	OWNER (of table)
DELETE SEGMENTS	DELETE (of table)
SHOW SEGMENTS	SELECT (of table)
CLEAN FILES	DELETE (of table)
INSERT OVERWRITE / INSERT INTO	INSERT (of table) RW on data file and SELECT (of table)
CREATE INDEX	OWNER (of table)
DROP INDEX	OWNER (of table)
SHOW INDEXES	SELECT (of table)
ALTER TABLE ADD COLUMN	OWNER (of table)
ALTER TABLE DROP COLUMN	OWNER (of table)
ALTER TABLE CHANGE DATATYPE	OWNER (of table)

场景	所需权限
ALTER TABLE RENAME	OWNER (of table)
ALTER TABLE COMPACTION	INSERT (on table)
FINISH STREAMING	OWNER (of table)
ALTER TABLE SET STREAMING PROPERTIES	OWNER (of table)
ALTER TABLE SET TABLE PROPERTIES	OWNER (of table)
UPDATE CARBON TABLE	UPDATE (of table)
DELETE RECORDS	DELETE (of table)
REFRESH TABLE	OWNER (of main table)
REGISTER INDEX TABLE	OWNER (of table)
SHOW PARTITIONS	SELECT (on table)
ALTER TABLE ADD PARTITION	OWNER (of table)
ALTER TABLE DROP PARTITION	OWNER (of table)

说明

- 如果数据库下的表由多个用户创建，那么执行 Drop database 命令会失败，即使执行的用户是数据库的拥有者。
- 在二级索引中，当父表（parent table）触发时，insert 和 compaction 将在索引表上触发。 如果选择具有过滤条件匹配索引表列的查询，用户应该为父表和索引表提供选择权限。
- LockFiles 文件夹和 LockFiles 文件夹中创建的锁定文件将具有完全权限，因为 LockFiles 文件夹不包含任何敏感数据。
- 如果使用 ACL，确保不要为 DDL 或 DML 配置任何被其他进程使用中的路径，建议创建新路径。

  以下配置项需要配置路径：

  1) carbon.badRecords.location

  2) 创建数据库时 Db_Path 及其他。
- 对于非安全集群中的 Carbon ACL 权限，hive-site.xml 中的参数 hive.server2.enable.doAs 必须设置为 false。 将此属性设置为 false，查询将以 hiveserver2 进程运行的用户身份运行。

3.6 CarbonData 语法参考

3.6.1 DDL

3.6.1.1 CREATE TABLE

命令功能

CREATE TABLE 命令通过指定带有表属性的字段列表来创建 CarbonData Table。

命令格式

CREATE TABLE *[IF NOT EXISTS] [db_name.]table_name*

[(col_name data_type, ...)]

STORED AS *carbondata*

[TBLPROPERTIES (property_name=property_value, ...)];

所有表的附加属性都会放到 TBLPROPERTIES 中来定义。

参数描述

表3-23 CREATE TABLE 参数描述

参数	描述
db_name	Database 名称，由字母、数字和下划线（_）组成。
col_name data_type	以逗号分隔的带数据类型的列表。列名由字母、数字和下划线（_）组成。 说明 在 CarbonData 表创建过程中，不允许使用 tupleId，PositionId 和 PositionReference 为列命名，因为具有这些名称的列由二级索引命令在内部使用。
table_name	Database 中的表名，由字母、数字和下划线（_）组成。
STORED AS	参数 carbondata，定义和创建 CarbonData table。
TBLPROPERTIES	CarbonData table 属性列表。

注意事项

以下是表格属性的使用。

- Block 大小

  单个表的数据文件 block 大小可以通过 TBLPROPERTIES 进行定义，系统会选择数据文件实际大小和设置的 blocksize 大小中的较大值，作为该数据文件在 HDFS

上存储的实际 blocksize 大小。单位为 MB，默认值为 1024MB，范围为 1MB~2048MB。若设置值不在[1, 2048]之间，系统将会报错。

一旦 block 大小达到配置值，写入程序将启动新的 CarbonData 数据的 block。数据以页面大小（32000 个记录）的倍数写入，因此边界在字节级别上不严格。 如果新页面跨越配置 block 的边界，则不会将其写入当前 block，而是写入新的 block。

TBLPROPERTIES('table_blocksize'='128')

说明

- 当在 CarbonData 表中配置了较小的 blocksize，而加载的数据生成的数据文件比较大时，在 HDFS 上显示的 blocksize 会与设置值不同。这是因为，对于每一个本地 block 文件的首次写入，即使待写入数据的大小大于 blocksize 的配置值，也直接将待写入数据写入此 block。所以，HDFS 上 blocksize 的实际值为待写入数据大小与 blocksize 配置值中的较大值。
- 当 CarbonData 表中的数据文件 block.num 小于任务并行度（parellelism）时，CarbonData 数据文件的 block 会被切为新的 block，使得 blocks.num 大于 parellelism，这样所有 core 均可被使用。这种优化称为 block distribution。

- SORT_SCOPE：指定表创建时的排序范围。如下为四种排序范围。
  - GLOBAL_SORT：它提高了查询性能，特别是点查询。
    TBLPROPERTIES('SORT_SCOPE'='GLOBAL_SORT')
  - LOCAL_SORT：数据会本地排序（任务级别排序）。
  - NO_SORT：默认排序。它将以不排序的方式加载数据，这将显着提升加载性能。
- SORT_COLUMNS

  此表属性指定排序列的顺序。

  TBLPROPERTIES('SORT_COLUMNS'='column1, column3')

  说明

  - 如果未指定此属性，则默认情况下，没有列会被排序。
  - 如果指定了此属性，但具有空参数，则表将被加载而不进行排序。例如，*('SORT_COLUMNS'='')*。
  - SORT_COLUMNS 将接受 string，date，timestamp，short，int，long, byte 和 boolean 数据类型。
- RANGE_COLUMN

  此表属性指定一列，该列将会按照一个范围值来对输入的数据进行分区。仅可配置一列。在数据导入过程中，可以使用“global_sort_partitions”或者“scale_factor”来避免生成小文件。

  TBLPROPERTIES('RANGE_COLUMN'='column1')
- LONG_STRING_COLUMNS

  普通 String 类型的长度不能超过 32000 字符，如果需要存储超过 32000 字符的字符串，指定 LONG_STRING_COLUMNS 配置为该列。

  TBLPROPERTIES('LONG_STRING_COLUMNS'='column1, column3')

说明

LONG_STRING_COLUMNS 仅可以设置 string/char/varchar 类型的列，并且不能为 SORT_COLUMNS 和复杂列。

使用场景

通过指定列创建表

CREATE TABLE 命令与 Hive DDL 相同。CarbonData 的额外配置将作为表格属性给出。

CREATE TABLE *[IF NOT EXISTS] [db_name.]table_name*

[(col_name data_type , ...)]

STORED AS *carbondata*

[TBLPROPERTIES (property_name=property_value, ...)];

示例

CREATE TABLE *IF NOT EXISTS productdb.productSalesTable (*

productNumber Int,

productName String,

storeCity String,

storeProvince String,

productCategory String,

productBatch String,

saleQuantity Int,

revenue Int)

STORED AS carbondata

TBLPROPERTIES (

'table_blocksize'='128',

'SORT_COLUMNS'='productBatch, productName')

系统响应

Table 创建成功，创建成功的消息将被记录在系统日志中。

3.6.1.2 CREATE TABLE As SELECT

命令功能

CREATE TABLE As SELECT 命令通过指定带有表属性的字段列表来创建 CarbonData Table。

命令格式

CREATE TABLE *[IF NOT EXISTS] [db_name.]table_name* ***STORED AS carbondata*** *[TBLPROPERTIES (key1=val1, key2=val2, ...)] AS select_statement;*

参数描述

表3-24 CREATE TABLE 参数描述

参数	描述
db_name	Database 名称，由字母、数字和下划线（_）组成。
table_name	Database 中的表名，由字母、数字和下划线（_）组成。
STORED AS	使用 CarbonData 数据格式存储数据。
TBLPROPERTIES	CarbonData table 属性列表。详细信息，见注意事项。

注意事项

NA

示例

CREATE TABLE ctas_select_parquet ***STORED AS*** carbondata as select * from parquet_ctas_test;

系统响应

该命令会从 Parquet 表上创建一个 Carbon 表，同时导入所有 Parquet 表的数据。

3.6.1.3 DROP TABLE

命令功能

DROP TABLE 的功能是用来删除已存在的 Table。

命令格式

DROP TABLE *[IF EXISTS] [db_name.]table_name;*

参数描述

表3-25 DROP TABLE 参数描述

参数	描述
db_name	Database 名称。如果未指定，将选择当前 database。

参数	描述
table_name	需要删除的 Table 名称。

注意事项

在该命令中，IF EXISTS 和 db_name 是可选配置。

示例

DROP TABLE IF EXISTS productDatabase.productSalesTable;

系统响应

Table 将被删除。

3.6.1.4 SHOW TABLES

命令功能

SHOW TABLES 命令用于显示所有在当前 database 中的 table，或所有指定 database 的 table。

命令格式

SHOW TABLES *[IN db_name];*

参数描述

表3-26 SHOW TABLES 参数描述

参数	描述
IN db_name	Database 名称，仅当需要显示指定 Database 的所有 Table 时配置。

注意事项

IN db_Name 为可选配置。

示例

SHOW TABLES IN ProductDatabase;

系统响应

显示所有 Table。

3.6.1.5 ALTER TABLE COMPACTION

命令功能

ALTER TABLE COMPACTION 命令将合并指定数量的 segment 为一个 segment。这将提高该表的查询性能。

命令格式

ALTER TABLE*[db_name.]table_name COMPACT 'MINOR/MAJOR/SEGMENT_INDEX';*

ALTER TABLE*[db_name.]table_name COMPACT 'CUSTOM' WHERE SEGMENT.ID IN (id1, id2, ...);*

参数描述

表3-27 ALTER TABLE COMPACTION 参数描述

Parameter	Description
db_name	数据库名。若未指定，则选择当前数据库。
table_name	表名。
MINOR	Minor 合并，详见合并 Segments。
MAJOR	Major 合并，详见合并 Segments。
SEGMENT_INDEX	这会将一个 segment 内的所有 Carbon 索引文件（.carbonindex）合并为一个 Carbon 索引合并文件（.carbonindexmerge）。 这增强了首次查询性能。详见表 3-12。
CUSTOM	Custom 合并，详见合并 Segments。

注意事项

NA

示例

ALTER TABLE ProductDatabase COMPACT 'MINOR';

ALTER TABLE ProductDatabase COMPACT 'MAJOR';

ALTER TABLE ProductDatabase COMPACT 'SEGMENT_INDEX';

ALTER TABLE ProductDatabase COMPACT 'CUSTOM' WHERE SEGMENT.ID IN (0, 1);

系统响应

由于为后台运行，**ALTER TABLE COMPACTION** 命令不会显示压缩响应。

如果想要查看 MINOR 合并和 MAJOR 合并的响应结果，用户可以检查日志或运行 ***SHOW SEGMENTS*** 命令查看。

示例：

```
+------+------------+--------------------------+------------------+------------+---
---------+-------------+--------------+--+
|  ID  |   Status   |    Load Start Time      | Load Time Taken  | Partition  | Data
Size  | Index Size  | File Format  |
+------+------------+--------------------------+------------------+------------+---
---------+-------------+--------------+--+
| 3    | Success    | 2020-09-28 22:53:26.336  | 3.726S           | {}         | 6.47KB
| 3.30KB      | columnar_v3  |
| 2    | Success    | 2020-09-28 22:53:01.702  | 6.688S           | {}         | 6.47KB
| 3.30KB      | columnar_v3  |
| 1    | Compacted  | 2020-09-28 22:51:15.242  | 5.82S            | {}         | 6.50KB
| 3.43KB      | columnar_v3  |
| 0.1  | Success    | 2020-10-30 20:49:24.561  | 16.66S           | {}         | 12.87KB
| 6.91KB      | columnar_v3  |
| 0    | Compacted  | 2020-09-28 22:51:02.6    | 6.819S           | {}         | 6.50KB
| 3.43KB      | columnar v3  |
+------+------------+--------------------------+------------------+------------+---
---------+-------------+--------------+--+
```

其中，

- Compacted 表示该数据已被合并。
- 0.1 表示 segment0 与 segment1 合并之后的结果。

数据合并前后的其他操作没有差别。

被合并的 segments（例如 segment0 和 segment1）即成为无用的 segments，会占用空间，因此建议合并之后使用 ***CLEAN FILES*** 命令进行彻底删除，再进行其他操作。**CLEAN FILES** 命令的使用方法可参考 CLEAN FILES。

3.6.1.6 TABLE RENAME

命令功能

RENAME 命令用于重命名现有表。

命令语法

ALTER TABLE *[db_name.]table_name* **RENAME TO** *new_table_name*;

参数描述

表3-28 RENAME 参数描述

参数	描述
db_name	数据库名。若未指定，则选择当前数据库。
table_name	现有表名。

参数	描述
new_table_name	现有表名的新表名。

注意事项

- 并行运行的查询（需要使用表名获取路径，以读取 CarbonData 存储文件）可能会在此操作期间失败。
- 不允许二级索引表重命名。

示例

ALTER TABLE *carbon* **RENAME TO** *carbondata*;

ALTER TABLE *test_db.carbon* **RENAME TO** *test_db.carbondata*;

系统响应

CarbonData 库中的文件夹将显示新表名称，可以通过运行 SHOW TABLES 显示新表名称。

3.6.1.7 ADD COLUMNS

命令功能

ADD COLUMNS 命令用于为现有表添加新列。

命令语法

ALTER TABLE *[db_name.]table_name* **ADD COLUMNS** *(col_name data_type,...)* **TBLPROPERTIES***("COLUMNPROPERTIES.columnName.shared_column'='sharedFolder.sharedColumnName,...', 'DEFAULT.VALUE.COLUMN_NAME'='default_value')*;

参数描述

表3-29 ADD COLUMNS 参数描述

参数	描述
db_name	数据库名。若未指定，则选择当前数据库。
table_name	表名。
col_name data_type	带数据类型且用逗号分隔的列的名称。列名称包含字母，数字和下划线（_）。 说明 创建 CarbonData 表时，不要将列名命名为 tupleId，PositionId 和 PositionReference，因为将在 UPDATE，DELETE 和二级索引命令内部

参数	描述
	使用这些名称。

注意事项

- 除了 shared_column 和 default_value 之外，将不会读取其他属性。如果指定了任何其他属性名称，则不会抛出错误，其他属性将被忽略。
- 如果未指定默认值，则新列的默认值将被视为 null。
- 如果在该列上应用 filter，则在排序期间不会考虑新增列，新增列可能会影响查询性能。

示例

- **ALTER TABLE** *carbon* **ADD COLUMNS** *(a1 INT, b1 STRING)*;
- **ALTER TABLE** *carbon* **ADD COLUMNS** *(a1 INT, b1 STRING)* **TBLPROPERTIES***('COLUMNPROPERTIES.b1.shared_column'='sharedFolder.b1')*;
- ALTER TABLE *carbon* **ADD COLUMNS** *(a1 INT, b1 STRING)* **TBLPROPERTIES***('DEFAULT.VALUE.a1'='10')*;

系统响应

通过运行 DESCRIBE 命令，可显示新添加的列。

3.6.1.8 DROP COLUMNS

命令功能

DROP COLUMNS 命令用于删除表中现有的列或多个列。

命令语法

ALTER TABLE *[db_name.]table_name* **DROP COLUMNS** *(col_name, ...)*;

参数描述

表3-30 DROP COLUMNS 参数描述

参数	描述
db_name	数据库名。若未指定，则选择当前数据库。
table_name	表名。
col_name	表中的列名称。支持多列。列名称包含字母，数字和下划线（_）。

注意事项

对于删除列操作，至少要有一个 key 列在删除操作后存在于 schema 中，否则将显示出错信息，删除列操作将失败。

示例

假设表包含 4 个列，分别命名为 a1，b1，c1 和 d1。

- 删除单个列：
  ALTER TABLE *carbon* **DROP COLUMNS** *(b1)*;
  ALTER TABLE *test_db.carbon* **DROP COLUMNS** *(b1)*;
- 删除多个列：
  ALTER TABLE *carbon* **DROP COLUMNS** *(b1,c1)*;
  ALTER TABLE *test_db.carbon* **DROP COLUMNS** *(b1,c1)*;

系统响应

运行 DESCRIBE 命令，将不会显示已删除的列。

3.6.1.9 CHANGE DATA TYPE

命令功能

CHANGE 命令用于将数据类型从 INT 更改为 BIGINT 或将 Decimal 精度从低精度改为高精度。

命令语法

ALTER TABLE *[db_name.]table_name* **CHANGE** *col_name col_name changed_column_type*;

参数描述

表3-31 CHANGE DATA TYPE 参数描述

参数	描述
db_name	数据库名。若未指定，则选择当前数据库。
table_name	表名。
col_name	表中的列名称。列名称包含字母，数字和下划线（_）。
changed_column_type	所要更改为的新数据类型。

注意事项

- 仅在没有数据丢失的情况下支持将 Decimal 数据类型从较低精度更改为较高精度

例如：

- 无效场景：将 Decimal 数据精度从（10,2）更改为（10,5）无效，因为在这种情况下，只有 scale 增加，但总位数保持不变。
- 有效场景：将 Decimal 数据精度从（10,2）更改为（12,3）有效，因为总位数增加 2，但是 scale 仅增加 1，这不会导致任何数据丢失。

- 将 Decimal 数据类型从较低精度更改为较高精度，其允许的最大精度(precision, scale)范围为(38,38)，并且只适用于不会导致数据丢失的有效提升精度的场景。

示例

- 将列 a1 的数据类型从 INT 更改为 BIGINT。

  ALTER TABLE *test_db.carbon* **CHANGE** *a1 a1 BIGINT*;
- 将列 a1 的精度从 10 更改为 18。

  ALTER TABLE *test_db.carbon* **CHANGE** *a1 a1 DECIMAL(18,2)*;

系统响应

通过运行 DESCRIBE 命令，将显示被修改列变更后的数据类型。

3.6.1.10 REFRESH TABLE

命令功能

REFRESH TABLE 命令用于将已有的 Carbon 表数据注册到 Hive 元数据库中。

命令语法

REFRESH TABLE *db_name.table_name*;

参数描述

表3-32 REFRESH TABLE 参数描述

参数	描述
db_name	数据库名。若未指定，则选择当前数据库。
table_name	表名。

注意事项

- 在执行此命令之前，应将旧表的表结构定义 schema 和数据复制到新数据库位置。
- 对于旧版本仓库，源集群和目的集群的时区应该相同。
- 新的数据库和旧数据库的名字应该相同。
- 执行命令前，旧表的表结构定义 schema 和数据应该复制到新的数据库位置。
- 如果表是聚合表，则应将所有聚合表复制到新的数据库位置。

- 如果旧集群使用 HIVE 元数据库来存储表结构，则刷新将不起作用，因为文件系统中不存在表结构定义 schema 文件。

示例

REFRESH TABLE *dbcarbon.productSalesTable*;

系统响应

通过运行该命令，已有的 Carbon 表数据会被注册到 Hive 元数据库中。

3.6.1.11 REGISTER INDEX TABLE

命令功能

REGISTER INDEX TABLE 命令用于将索引表注册到主表。

命令语法

REGISTER INDEX TABLE *indextable_name* ON *db_name.maintable_name*;

参数描述

表3-33 REFRESH INDEX TABLE 参数描述

参数	描述
db_name	数据库名。若未指定，则选择当前数据库。
indextable_name	索引表名。
maintable_name	主表名。

注意事项

在执行此命令之前，使用 REFRESH TABLE 将主表和二级所索引表都注册到 Hive 元数据中。

示例

create database productdb;

use productdb;

CREATE TABLE productSalesTable(a int,b string,c string) stored as carbondata;

create index productNameIndexTable on table productSalesTable(c) as 'carbondata';

insert into table productSalesTable select 1,'a','aaa';

create database productdb2;

使用 hdfs 命令将 productdb 数据库下的 productSalesTable 和 productNameIndexTable 拷贝到 productdb2。

refresh table productdb2.productSalesTable ;

refresh table productdb2.productNameIndexTable ;

explain select * from productdb2.productSalesTable where c = 'aaa'; //可以发现该查询命令没有使用索引表

REGISTER INDEX TABLE productNameIndexTable ON productdb2.productSalesTable;

explain select * from productdb2.productSalesTable where c = 'aaa'; //可以发现该查询命令使用了索引表

系统响应

通过运行该命令，索引表会被注册到主表。

3.6.2 DML

3.6.2.1 LOAD DATA

命令功能

LOAD DATA 命令以 CarbonData 特定的数据存储类型加载原始的用户数据，这样，CarbonData 可以在查询数据时提供良好的性能。

说明

仅支持加载位于 HDFS 上的原始数据。

命令格式

LOAD DATA *INPATH 'folder_path' INTO TABLE [db_name.]table_name OPTIONS(property_name=property_value, ...);*

参数描述

表3-34 LOAD DATA 参数描述

参数	描述
folder_path	原始 CSV 数据文件夹或者文件的路径。
db_name	Database 名称。若未指定，则使用当前 database。
table_name	所提供的 database 中的表的名称。

注意事项

以下是可以在加载数据时使用的配置选项：

- DELIMITER：可以在加载命令中提供分隔符和引号字符。默认值为,。

  OPTIONS('DELIMITER'=',' , 'QUOTECHAR'='"')

  可使用'DELIMITER'='\t'来表示用制表符 tab 对 CSV 数据进行分隔。

  OPTIONS('DELIMITER'='\t')

  CarbonData 也支持\001 和\017 作为分隔符。

  说明

  对于 CSV 数据，分隔符为单引号（'）时，单引号必须在双引号（" "）内。例如：'DELIMITER='"'"。

- QUOTECHAR：可以在加载命令中提供分隔符和引号字符。默认值为"。

  OPTIONS('DELIMITER'=',' , 'QUOTECHAR'='"')

- COMMENTCHAR：可以在加载命令中提供注释字符。在加载操作期间，如果在行的开头遇到注释字符，那么该行将被视为注释，并且不会被加载。默认值为＃。

  OPTIONS('COMMENTCHAR'='#')

- FILEHEADER：如果源文件中没有表头，可在 LOAD DATA 命令中提供表头。

  OPTIONS('FILEHEADER'='column1,column2')

- ESCAPECHAR：如果用户想在 CSV 上对 Escape 字符进行严格验证，可以提供 Escape 字符。默认值为\。

  OPTIONS('ESCAPECHAR'='\')

  说明

  如果在 CSV 数据中输入 ESCAPECHAR，该 ESCAPECHAR 必须在双引号（" "）内。例如："a\b"。

- Bad Records 处理：

  为了使数据处理应用程序为用户增值，不可避免地需要对数据进行某种程度的集成。在大多数情况下，数据质量问题源于生成源数据的上游（主要）系统。

  有两种完全不同的方式处理 Bad Data：

  - 按照数据原本的样子加载所有数据，之后进行除错处理。
  - 在进入数据源的过程中，可以清理或擦除 Bad Data，或者在发现 Bad Data 时让数据加载失败。

  有多个选项可用于在 CarbonData 数据加载过程中清除源数据。对于 CarbonData 数据中的 Bad Records 管理，请参见表 3-35。

表3-35 Bad Records Logger

配置项	默认值	描述
BAD_RECORDS_LOGGER_ENABLE	false	若设置为 true，则将创建 Bad Records 日志文件，其中包含 Bad Records 的详细信息。

配置项	默认值	描述
BAD_RECORDS_ACTION	FAIL	以下为 Bad Records 的四种操作类型： • FORCE：通过将 Bad Records 存储为 NULL 来自动校正数据。 • REDIRECT：无法加载 Bad Records，并将其写入原始 CSV 文件。 • IGNORE：既不加载 Bad Records 也不将其写入原始 CSV 文件。 • FAIL：如果发现存在 Bad Records，数据加载将会失败。 说明 在加载数据时，如果所有记录都是 Bad Records，则参数 BAD_RECORDS_ACTION 将不起作用，加载数据操作将会失败。
IS_EMPTY_DATA_BAD_RECORD	false	如果设置为“false”，则空（""或"或,,）数据将不被视为 Bad Records，如果设置为“true”，则空数据将被视为 Bad Records。
BAD_RECORD_PATH	-	指定存储 Bad Records 的 HDFS 路径。默认值为 Null。 如果启用了 Bad Records 日志记录或者 Bad Records 操作重定向，则该路径必须由用户进行配置。

示例：

LOAD DATA INPATH *'filepath.csv'* **INTO TABLE** *tablename OPTIONS('BAD_RECORDS_LOGGER_ENABLE'='true', 'BAD_RECORD_PATH'='hdfs://hacluster/tmp/carbon', 'BAD_RECORDS_ACTION'='REDIRECT', 'IS_EMPTY_DATA_BAD_RECORD'='false');*

说明

使用“REDIRECT”选项，CarbonData 会将所有的 Bad Records 添加到单独的 CSV 文件中，但是该文件内容不能用于后续的数据加载，因为其内容可能无法与源记录完全匹配。用户必须清理原始源记录以便于进一步的数据提取。该选项的目的只是让用户知道哪些记录被视为 Bad Records。

- MAXCOLUMNS：该可选参数指定了在一行中，由 CSV 解析器解析的最大列数。
  OPTIONS('MAXCOLUMNS'='400')

表3-36 MAXCOLUMNS

可选参数名称	默认值	最大值
MAXCOLUMNS	2000	20000

表3-37 MAXCOLUMNS 可选参数的行为图

MAXCOLUMNS 值	在文件 Header 选项中的列数	考虑的最终值
在加载项中未指定	5	2000
在加载项中未指定	6000	6000
40	7	文件 header 列数与 MAXCOLUMNS 值，两者中的最大值
22000	40	20000
60	在加载项中未指定	CSV 文件第一行的列数与 MAXCOLUMNS 值，两者中的最大值

说明

对于设置 MAXCOLUMNS Option 的最大值，要求 executor 具有足够的内存，否则，数据加载会由于内存不足的错误而失败。

- 如果在创建表期间将 SORT_SCOPE 定义为 GLOBAL_SORT，则可以指定在对数据进行排序时要使用的分区数。如果未配置或配置小于 1，则将使用 map 任务的数量作为 reduce 任务的数量。 建议每个 reduce 任务处理 512MB - 1GB 数据。

  OPTIONS('GLOBAL_SORT_PARTITIONS'='2')

说明

增加分区数可能需要增加“spark.driver.maxResultSize”，因为在 driver 中收集的采样数据随着分区的增加而增加。

- DATEFORMAT：此选项用于指定表的日期格式。

  OPTIONS('DATEFORMAT'='dateFormat')

说明

日期格式由日期模式字符串指定。 Carbon 中的日期模式字母与 JAVA 中的日期模式字母相同。

- TIMESTAMPFORMAT：此选项用于指定表的时间戳格式。
- *OPTIONS('TIMESTAMPFORMAT'='timestampFormat')*
- SKIP_EMPTY_LINE：数据加载期间，此选项将忽略 CSV 文件中的空行。

  OPTIONS('SKIP_EMPTY_LINE'='TRUE/FALSE')

- **可选:** SCALE_FACTOR：针对 RANGE_COLUMN，SCALE_FACTOR 用来控制分区的数量，根据如下公式：

```
splitSize = max(blocklet_size, (block_size - blocklet_size)) *
scale_factor
numPartitions = total size of input data / splitSize
```

  默认值为 3，range 的范围为[1, 300]。

  OPTIONS('SCALE_FACTOR'='10')

说明

- 如果 GLOBAL_SORT_PARTITIONS 和 SCALE_FACTOR 同时使用，只有 GLOBAL_SORT_PARTITIONS 生效。
- RANGE_COLUMN 合并默认使用 LOCAL_SORT。

使用场景

可使用下列语句从 CSV 文件加载 CarbonData table。

LOAD DATA *INPATH 'folder path' INTO TABLE tablename OPTIONS(property_name=property_value, ...);*

示例

data.csv 源文件数据如下所示：

```
ID,date,country,name,phonetype,serialname,salary
4,2014-01-21 00:00:00,china,aaa4,phone2435,ASD66902,15003
5,2014-01-22 00:00:00,china,aaa5,phone2441,ASD90633,15004
6,2014-03-07 00:00:00,china,aaa6,phone294,ASD59961,15005
```

CREATE TABLE carbontable(ID int, date Timestamp, country String, name String, phonetype String, serialname String,salary int) STORED AS carbondata;

LOAD DATA *inpath 'hdfs://hacluster/tmp/data.csv' INTO table carbontable*

options('DELIMITER'=',');

系统响应

可在 driver 日志中查看命令运行成功或失败。

3.6.2.2 UPDATE CARBON TABLE

命令功能

UPDATE 命令根据列表达式和可选的过滤条件更新 CarbonData 表。

命令格式

- 格式 1：

UPDATE <CARBON TABLE> SET (column_name1, column_name2, ... column_name n) = (column1_expression , column2_expression , column3_expression ... column n_expression) [WHERE { <filter_condition> }];

- 格式 2：

  UPDATE <CARBON TABLE> SET (column_name1, column_name2,) = (select sourceColumn1, sourceColumn2 from sourceTable [WHERE { <filter_condition> }]) [WHERE { <filter_condition> }];

参数描述

表3-38 UPDATE 参数

参数	描述
CARBON TABLE	在其中执行更新操作的 CarbonData 表的名称。
column_name	待更新的目标列。
sourceColumn	需在目标表中更新的源表的列值。
sourceTable	将其记录更新到目标 CarbonData 表中的表。

注意事项

以下是使用 UPDATE 命令的条件：

- 如果源表中的多个输入行与目标表中的单行匹配，则 UPDATE 命令失败。
- 如果源表生成空记录，则 UPDATE 操作将在不更新表的情况下完成。
- 如果源表的行与目标表中任何已有的行不对应，则 UPDATE 操作将完成，不更新表。
- 具有二级索引的表不支持 UPDATE 命令。
- 在子查询中，如果源表和目标表相同，则 UPDATE 操作失败。
- 如果在 UPDATE 命令中使用的子查询包含聚合函数或 group by 子句，则 UPDATE 操作失败。

  例如，**update t_carbn01 a set (a.item_type_code, a.profit) = (select b.item_type_cd, sum(b.profit) from t_carbn01b b where item_type_cd =2 group by item_type_code);**

  其中，在子查询中使用聚合函数 sum(b.profit)和 group by 子句，因此 UPDATE 操作失败。
- 如果查询的表设置了 carbon.input.segments 属性，则 UPDATE 操作失败。要解决该问题，在查询前执行以下语句。

  语法：

  SET carbon.input.segments. <database_name>. <table_name>=*;

示例

- 示例 1：

update carbonTable1 d set (d.column3,d.column5) = (select s.c33 ,s.c55 from sourceTable1 s where d.column1 = s.c11) where d.column1 = 'country' exists(select * from table3 o where o.c2 > 1);

- 示例 2：

  update carbonTable1 d set (c3) = (select s.c33 from sourceTable1 s where d.column1 = s.c11) where exists(select * from iud.other o where o.c2 > 1);

- 示例 3：

  update carbonTable1 set (c2, c5) = (c2 + 1, concat(c5 , "y"));

- 示例 4：

  update carbonTable1 d set (c2, c5) = (c2 + 1, "xyx") where d.column1 = 'india';

- 示例 5：

  update carbonTable1 d set (c2, c5) = (c2 + 1, "xyx") where d.column1 = 'india' and exists(select * from table3 o where o.column2 > 1);

系统响应

可在 driver 日志和客户端中查看命令运行成功或失败。

3.6.2.3 DELETE RECORDS from CARBON TABLE

命令功能

DELETE RECORDS 命令从 CarbonData 表中删除记录。

命令格式

DELETE FROM CARBON_TABLE [WHERE expression];

参数描述

表3-39 DELETE RECORDS 参数

参数	描述
CARBON TABLE	在其中执行删除操作的 CarbonData 表的名称。

注意事项

- 删除 segment 将删除相应 segment 的所有二级索引。
- 如果查询的表设置了 carbon.input.segments 属性，则 DELETE 操作失败。要解决该问题，在查询前执行以下语句。

  语法：

  SET carbon.input.segments. <database_name>.<table_name>=*;

示例

- 示例 1：

  delete from columncarbonTable1 d where d.column1 = 'country';

- 示例 2：

  delete from dest where column1 IN ('country1', 'country2');

- 示例 3：

  delete from columncarbonTable1 where column1 IN (select column11 from sourceTable2);

- 示例 4：

  delete from columncarbonTable1 where column1 IN (select column11 from sourceTable2 where column1 = 'USA');

- 示例 5：

  delete from columncarbonTable1 where column2 >= 4;

系统响应

可在 driver 日志和客户端中查看命令运行成功或失败。

3.6.2.4 INSERT INTO CARBON TABLE

命令功能

INSERT 命令用于将 SELECT 查询结果加载到 CarbonData 表中。

命令格式

INSERT INTO [CARBON TABLE] [select query];

参数描述

表3-40 INSERT INTO 参数

参数	描述
CARBON TABLE	需要执行 INSERT 命令的 CarbonData 表的名称。
select query	Source 表上的 SELECT 查询（支持 CarbonData、Hive 和 Parquet 表）。

注意事项

- 表必须已经存在。
- 用户应属于数据加载组以执行数据加载操作。默认情况下，数据加载组被命名为“ficommon”。
- CarbonData 表不支持 Overwrite。

- 源表和目标表的数据类型应该相同，否则原表中的数据将被视为 Bad Records。
- **INSERT INTO** 命令不支持部分成功（partial success），如果存在 Bad Records，该命令会失败。
- 在从源表插入数据到目标表的过程中，无法在源表中加载或更新数据。
  若要在 INSERT 操作期间启用数据加载或更新，请将以下参数配置为“true”。
  “carbon.insert.persist.enable”=“true”
  默认上述参数配置为“false”。

说明

启用该参数将降低 INSERT 操作的性能。

示例

create table carbon01(a int,b string,c string) stored as carbondata;

insert into table carbon01 values(1,'a','aa'),(2,'b','bb'),(3,'c','cc');

create table carbon02(a int,b string,c string) stored as carbondata;

INSERT INTO carbon02 select * from carbon01 where a > 1;

系统响应

可在 driver 日志中查看命令运行成功或失败。

3.6.2.5 DELETE SEGMENT by ID

命令功能

DELETE SEGMENT by ID 命令是使用 Segment ID 来删除 segment。

命令格式

DELETE FROM TABLE db_name.table_name WHERE SEGMENT.ID IN (segment_id1,segment_id2);

参数描述

表3-41 DELETE LOAD 参数描述

参数	描述
segment_id	将要删除的 Segment 的 ID。
db_name	Database 名称，若未指定，则使用当前 database。
table_name	在给定的 database 中的表名。

注意事项

流式表不支持删除 segment。

示例

DELETE FROM TABLE CarbonDatabase.CarbonTable WHERE SEGMENT.ID IN (0);

DELETE FROM TABLE CarbonDatabase.CarbonTable WHERE SEGMENT.ID IN (0,5,8);

系统响应

操作成功或失败会在 CarbonData 日志中被记录。

3.6.2.6 DELETE SEGMENT by DATE

命令功能

DELETE SEGMENT by DATE 命令用于通过加载日期删除 CarbonData segment，在特定日期之前创建的 segment 将被删除。

命令格式

DELETE FROM TABLE db_name.table_name WHERE SEGMENT.STARTTIME BEFORE date_value;

参数描述

表3-42 DELETE SEGMENT by DATE 参数描述

参数	描述
db_name	Database 名称，若未指定，则使用当前 database。
table_name	给定 database 中的表名。
date_value	有效 Segment 加载启动时间。在这个指定日期前的 Segment 将被删除。

注意事项

流式表不支持删除 segment。

示例

DELETE FROM TABLE db_name.table_name WHERE SEGMENT.STARTTIME BEFORE '2017-07-01 12:07:20';

其中，STARTTIME 是不同负载的加载启动时间。

系统响应

操作成功或失败会在 CarbonData 日志中被记录。

3.6.2.7 SHOW SEGMENTS

命令功能

SHOW SEGMENTS 命令是用来向用户展示 CarbonData table 的 Segment。

命令格式

SHOW SEGMENTS FOR TABLE *[db_name.]table_name* **LIMIT** *number_of_loads;*

参数描述

表3-43 SHOW SEGMENTS FOR TABLE 参数描述

参数	描述
db_name	Database 名，若未指定，则使用当前 database。
table_name	在给定 database 中的表名。
number_of_loads	加载数的限制。

注意事项

无。

示例

create table carbon01(a int,b string,c string) stored as carbondata;

insert into table carbon01 select 1,'a','aa';

insert into table carbon01 select 2,'b','bb';

insert into table carbon01 select 3,'c','cc';

SHOW SEGMENTS FOR TABLE carbon01 LIMIT 2;

系统响应

```
+-----+----------+--------------------------+------------------+------------+------
------+-------------+--------------+--+
| ID  |  Status  |    Load Start Time       | Load Time Taken  | Partition  | Data Size
| Index Size  | File Format  |
+-----+----------+--------------------------+------------------+------------+------
------+-------------+--------------+--+
| 3   | Success  | 2020-09-28 22:53:26.336  | 3.726S           | {}         | 6.47KB
| 3.30KB      | columnar_v3  |
```

```
| 2   | Success  | 2020-09-28 22:53:01.702  | 6.688S          | {}        | 6.47KB
| 3.30KB     | columnar_v3  |
+-----+----------+--------------------------+-----------------+-----------+------
------+-------------+--------------+--+
```

3.6.2.8 CREATE SECONDARY INDEX

命令功能

该命令用于在 CarbonData 表中创建二级索引表。

命令格式

CREATE INDEX *index_name*

ON TABLE *[db_name.]table_name (col_name1, col_name2)*

AS *'carbondata'*

PROPERTIES *('table_blocksize'='256')*;

参数描述

表3-44 CREATE SECONDARY INDEX 参数

参数	描述
index_name	索引表的名称。表名称应由字母数字字符和下划线（_）特殊字符组成。
db_name	数据库的名称。数据库名称应由字母数字字符和下划线（_）特殊字符组成。
table_name	数据库中的表名称。表名称应由字母数字字符和下划线（_）特殊字符组成。
col_name	表中的列名称。支持多列。列名称应由字母数字字符和下划线（_）特殊字符组成。
table_blocksize	数据文件的 block 大小。更多详细信息，请参考•Block 大小。

注意事项

db_name 为可选项。

示例

create table productdb.productSalesTable(id int,price int,productName string,city string) stored as carbondata;

CREATE INDEX productNameIndexTable on table productdb.productSalesTable (productName,city) as 'carbondata' ;

上述示例将创建名为“productdb.productNameIndexTable”的二级表并加载所提供列的索引信息。

系统响应

将创建二级索引表，加载与所提供的列相关的索引信息到二级索引表中，并将成功消息记录在系统日志中。

3.6.2.9 SHOW SECONDARY INDEXES

命令功能

该命令用于在所提供的 CarbonData 表中显示所有的二级索引表。

命令格式

SHOW INDEXES ON db_name.table_name;

参数描述

表3-45 SHOW SECONDARY INDEXES 参数

参数	描述
db_name	数据库的名称。数据库名称应由字母数字字符和下划线（_）特殊字符组成
table_name	数据库中的表名称。表名称应由字母数字字符和下划线（_）特殊字符组成。

注意事项

db_name 为可选项。

示例

create table productdb.productSalesTable(id int,price int,productName string,city string) stored as carbondata;

CREATE INDEX productNameIndexTable on table productdb.productSalesTable (productName,city) as 'carbondata' ;

SHOW INDEXES ON productdb.productSalesTable;

系统响应

显示列出给定 CarbonData 表中的所有索引表和相应的索引列。

3.6.2.10 DROP SECONDARY INDEX

命令功能

该命令用于删除给定表中存在的二级索引表。

命令格式

DROP INDEX [IF EXISTS] index_name ON [db_name.]table_name;

参数描述

表3-46 DROP SECONDARY INDEX 参数

参数	描述
index_name	索引表的名称。表名称应由字母数字字符和下划线（_）特殊字符组成。
db_name	数据库的名称。若未指定，选择当前默认数据库。
table_name	需要删除的表的名称。

注意事项

该命令中 IF EXISTS 和 db_name 为可选项。

示例

DROP INDEX if exists productNameIndexTable ON productdb.productSalesTable;

系统响应

二级索引表将被删除，索引信息将在 CarbonData 表中被清除，删除成功的消息将记录在系统日志中。

3.6.2.11 CLEAN FILES

命令功能

DELETE SEGMENT 命令会将被删除的 segments 标识为 delete 状态；segment 合并后，旧的 segments 状态会变为 compacted。这些 segments 的数据文件不会从物理上删除。如果用户希望强制删除这些文件，可以使用 **CLEAN FILES** 命令。

但是，使用该命令可能会导致查询命令执行失败。

命令格式

CLEAN FILES FOR TABLE *[db_name.]table_name* ;

参数描述

表3-47 CLEAN FILES FOR TABLE 参数描述

参数	描述
db_name	数据库名称。数据库名称由字母，数字和下划线组成。
table_name	数据库中的表的名称。表名由字母，数字和下划线组成。

注意事项

无。

示例

添加 carbon 配置参数

```
carbon.clean.file.force.allowed = true
```

create table carbon01(a int,b string,c string) stored as carbondata;

insert into table carbon01 select 1,'a','aa';

insert into table carbon01 select 2,'b','bb';

delete from table carbon01 where segment.id in (0);

show segments for table carbon01;

CLEAN FILES FOR TABLE carbon01 options('force'='true');

show segments for table carbon01;

上述命令将从物理上删除所有 DELETE SEGMENT 命令删除的 segment 和合并后的旧的 segment。

系统响应

可在 driver 日志中查看命令运行成功或失败。

3.6.2.12 SET/RESET

命令功能

此命令用于动态 Add，Update，Display 或 Reset CarbonData 参数，而无需重新启动 driver。

命令格式

- Add 或 Update 参数值：

  SET *parameter_name=parameter_value*

  此命令用于添加或更新“parameter_name”的值。

- Display 参数值：

  SET *parameter_name*

  此命令用于显示指定的“parameter_name”的值。
- Display 会话参数：

  SET

  此命令显示所有支持的会话参数。
- Display 会话参数以及使用细节：

  SET -v

  此命令显示所有支持的会话参数及其使用细节。
- Reset 参数值：

  RESET

  此命令清除所有会话参数。

参数描述

表3-48 SET 参数描述

参数	描述
parameter_name	其值需要被动态添加（add），更新（update）或显示（display）的参数名称。
parameter_value	将要设置的“parameter_name”的新值。

注意事项

以下为分别使用 SET 和 RESET 命令进行动态设置或清除操作的属性：

表3-49 属性描述

属性	描述
carbon.options.bad.records.logger.enable	启用或禁用 bad record 日志记录。
carbon.options.bad.records.action	指定 bad record 操作，例如，强制（force），重定向（redirect），失败（fail）或忽略（ignore）。有关详细信息，请参阅•Bad Records 处理：
carbon.options.is.empty.data.bad.record	指定空数据是否被视为 bad record。有关详细信息，请参阅 Bad Records 处理：
carbon.options.sort.scope	指定数据加载期间排序的范围。
carbon.options.bad.record.path	指定需要存储 bad record 的 HDFS 路径。

属性	描述
carbon.custom.block.distribution	指定是否使用 Spark 或 CarbonData 的块分布功能。
enable.unsafe.sort	指定在数据加载期间是否使用不安全的排序。不安全的排序可减少数据加载操作期间的垃圾回收，从而实现更好的性能。
carbon.si.lookup.partialstring	当参数设置为 TRUE 时，二级索引采用 starts-with、ends-with、contains 和 LIKE 分区条件字符串。 当参数设置为 FALSE 时，二级索引只采用 starts-with 分区条件字符串。
carbon.input.segments	指定要查询的段 ID。 此属性允许您查询指定表的指定段。CarbonScan 将仅从指定的段 ID 读取数据。 语法： “carbon.input.segments. <database_name>. <table_name> = < list of segment ids >” 如果用户想在多线程模式下查询指定段，可使用 **CarbonSession.threadSet** 代替 **SET** 语句。 语法： “CarbonSession.threadSet ("carbon.input.segments. <database_name>. <table_name>","< list of segment ids >");” 说明 不建议在 **carbon.properties** 文件中设置该属性，因为所有会话都包含段列表，除非发生会话级或线程级覆盖。

示例

- 添加（Add）或更新（Update）：
  SET *enable.unsafe.sort=true*
- 显示（Display）属性值：
  SET *enable.unsafe.sort*
- 显示段 ID 列表，段状态和其他所需详细信息的示例，然后指定要读取的段列表：
  SHOW SEGMENTS FOR *TABLE carbontable1;*
  SET *carbon.input.segments.db.carbontable1 = 1, 3, 9;*
- 多线程模式查询指定段示例如下：

CarbonSession.threadSet
(*"carbon.input.segments.default.carbon_table_MulTI_THread", "1,3"*);

- 在多线程环境中使用 **CarbonSession.threadSet** 查询段示例如下（以 Scala 代码为例）：

```
def main(args: Array[String]) {
 Future
{           CarbonSession.threadSet("carbon.input.segments.default.carbon_tabl
e_MulTI_THread", "1")
     spark.sql("select count(empno) from carbon_table_MulTI_THread").show()
   }
}
```

- 重置（Reset）：

  RESET

系统响应

- 若运行成功，将记录在 driver 日志中。
- 若出现故障，将显示在用户界面（UI）中。

3.6.3 操作并发

DDL 和 DML 中的操作，执行前，需要获取对应的锁，各操作需要获取锁的情况见表 1 操作获取锁一览表，√表示需要获取该锁，一个操作仅在获取到所有需要获取的锁后，才能继续执行。

任意两个操作是否可以并发执行，可以通过如下方法确定：表 3-50 两行代表两个操作，这两行没有任意一列都标记√，即不存在某一列两行全为√。

表3-50 操作获取锁一览表

操作	METADATA_LOCK	COMPACTION_LOCK	DROP_TABLE_LOCK	DELETE_SEGMENT_LOCK	CLEAN_FILES_LOCK	ALTER_PARTITION_LOCK	UPDATE_LOCK	STREAMING_LOCK	CONCURRENT_LOAD_LOCK	SEGMENT_LOCK
CREATE TABLE	-	-	-	-	-	-	-	-	-	-
CREATE TABLE As SELECT	-	-	-	-	-	-	-	-	-	-
DRO	√	-	√	-	-	-	-	√	-	-

操作	METADATA_LOCK	COMPACTION_LOCK	DROP_TABLE_LOCK	DELETE_SEGMENT_LOCK	CLEAN_FILES_LOCK	ALTER_PARTITION_LOCK	UPDATE_LOCK	STREAMING_LOCK	CONCURRENT_LOAD_LOCK	SEGMENT_LOCK
PTABLE										
ALTER TABLE COMPACTION	-	√	-	-	-	-	√	-	-	-
TABLE RENAME	-	-	-	-	-	-	-	-	-	-
ADD COLUMNS	√	√	-	-	-	-	-	-	-	-
DROP COLUMNS	√	√	-	-	-	-	-	-	-	-
CHANGE DATA TYPE	√	√	-	-	-	-	-	-	-	-
REFRESH TABLE	-	-	-	-	-	-	-	-	-	-
REGISTER INDEX TABLE	√	-	-	-	-	-	-	-	-	-

操作	METADATA_LOCK	COMPACTION_LOCK	DROP_TABLE_LOCK	DELETE_SEGMENT_LOCK	CLEAN_FILES_LOCK	ALTER_PARTITION_LOCK	UPDATE_LOCK	STREAMING_LOCK	CONCURRENT_LOAD_LOCK	SEGMENT_LOCK
REFRESH INDEX	-	√	-	-	-	-	-	-	-	-
LOAD DATA/INSERT INTO	-	-	-	-	-	-	-	-	√	√
UPDATE CARBONTABLE	√	√	-	-	-	-	√	-	-	-
DELETE RECORDS from CARBONTABLE	√	√	-	-	-	-	√	-	-	-
DELETE SEGMENT by ID	-	-	-	√	√	-	-	-	-	-
DELETE SEGMENT by DATE	-	-	-	√	√	-	-	-	-	-
SHOW	-	-	-	-	-	-	-	-	-	-

操作	METADATA_LOCK	COMPACTION_LOCK	DROP_TABLE_LOCK	DELETE_SEGMENT_LOCK	CLEAN_FILES_LOCK	ALTER_PARTITION_LOCK	UPDATE_LOCK	STREAMING_LOCK	CONCURRENT_LOAD_LOCK	SEGMENT_LOCK
SEGMENTS										
CREATE SECONDARY INDEX	√	√	-	√	-	-	-	-	-	-
SHOW SECONDARY INDEXES	-	-	-	-	-	-	-	-	-	-
DROP SECONDARY INDEX	√	-	√	-	-	-	-	-	-	-
CLEAN FILES	-	-	-	-	-	-	-	-	-	-
SET/RESET	-	-	-	-	-	-	-	-	-	-
Add Hive Partition	-	-	-	-	-	-	-	-	-	-
Drop Hive Partition	√	√	√	√	√	√	-	-	-	-
Drop Partiti	√	√	√	√	√	√	-	-	-	-

操作	METADATA_LOCK	COMPACTION_LOCK	DROP_TABLE_LOCK	DELETE_SEGMENT_LOCK	CLEAN_FILES_LOCK	ALTER_PARTITION_LOCK	UPDATE_LOCK	STREAMING_LOCK	CONCURRENT_LOAD_LOCK	SEGMENT_LOCK
on										
Alter table set	√	√	-	-	-	-	-	-	-	-

3.6.4 API

本章节描述 Segment 的 API 以及使用方法，所有方法在 org.apache.spark.util.CarbonSegmentUtil 类中。

如下方法已废弃：

```
/**
* Returns the valid segments for the query based on the filter condition
* present in carbonScanRdd.
*
* @param carbonScanRdd
* @return Array of valid segments
*/
@deprecated def getFilteredSegments(carbonScanRdd: CarbonScanRDD[InternalRow]):
Array[String];
```

使用方法

使用如下方法从查询语句中获得 CarbonScanRDD：

```
val df=carbon.sql("select * from table where age='12'")
val myscan=df.queryExecution.sparkPlan.collect {
case scan: CarbonDataSourceScan if scan.rdd.isInstanceOf[CarbonScanRDD[InternalRow]]
=> scan.rdd
case scan: RowDataSourceScanExec if
scan.rdd.isInstanceOf[CarbonScanRDD[InternalRow]] => scan.rdd
}.head
val carbonrdd=myscan.asInstanceOf[CarbonScanRDD[InternalRow]]
```

例子：

```
CarbonSegmentUtil.getFilteredSegments(carbonrdd)
```

可以通过传入 sql 语句来获取过滤后的 segment：

```
/**
* Returns an array of valid segment numbers based on the filter condition provided
in the sql
* NOTE: This API is supported only for SELECT Sql (insert into,ctas,.., is not
```

```
supported)
*
* @param sql
* @param sparkSession
* @return Array of valid segments
* @throws UnsupportedOperationException because Get Filter Segments API supports if
and only
* if only one carbon main table is present in query.
*/
def getFilteredSegments(sql: String, sparkSession: SparkSession): Array[String];
```

例子：

```
CarbonSegmentUtil.getFilteredSegments("select * from table where age='12'”,
sparkSession)
```

传入数据库名和表名，获取会被合并的 segment 列表，得到的 segment 列表可以当做 getMergedLoadName 函数的参数传入：

```
/**
* Identifies all segments which can be merged with MAJOR compaction type.
* NOTE: This result can be passed to getMergedLoadName API to get the merged load
name.
*
* @param sparkSession
* @param tableName
* @param dbName
* @return list of LoadMetadataDetails
*/
def identifySegmentsToBeMerged(sparkSession: SparkSession,
tableName: String,
dbName: String) : util.List[LoadMetadataDetails];
```

例子：

```
CarbonSegmentUtil.identifySegmentsToBeMerged(sparkSession, "table_test","default")
```

传入数据库名、表名和自定义的 segment 列表，获取自定义合并操作会被合并的 segment 列表，得到的 segment 列表可以当做 getMergedLoadName 函数的参数传入：

```
/**
* Identifies all segments which can be merged with CUSTOM compaction type.
*  NOTE: This result can be passed to getMergedLoadName API to get the merged load
name.
*
* @param sparkSession
* @param tableName
* @param dbName
* @param customSegments
* @return list of LoadMetadataDetails
* @throws UnsupportedOperationException if customSegments is null or empty.
* @throws MalformedCarbonCommandException if segment does not exist or is not valid
*/
def identifySegmentsToBeMergedCustom(sparkSession: SparkSession,
tableName: String,
dbName: String,
customSegments: util.List[String]): util.List[LoadMetadataDetails];
```

例子：

```
val customSegments = new util.ArrayList[String]()
customSegments.add("1")
customSegments.add("2")
CarbonSegmentUtil.identifySegmentsToBeMergedCustom(sparkSession,
"table_test","default", customSegments)
```

给定 segment 列表，返回合并后新的导入名称：

```
/**
* Returns the Merged Load Name for given list of segments
*
* @param list of segments
* @return Merged Load Name
* @throws UnsupportedOperationException if list of segments is less than 1
*/
def getMergedLoadName(list: util.List[LoadMetadataDetails]): String;
```

例子：

```
val carbonTable = CarbonEnv.getCarbonTable(Option(databaseName),
tableName)(sparkSession)
val loadMetadataDetails =
SegmentStatusManager.readLoadMetadata(carbonTable.getMetadataPath)
CarbonSegmentUtil.getMergedLoadName(loadMetadataDetails.toList.asJava)
```

3.6.5 空间索引

快速示例

```
create table IF NOT EXISTS carbonTable
(
COLUMN1    BIGINT,
LONGITUDE    BIGINT,
LATITUDE    BIGINT,
COLUMN2    BIGINT,
COLUMN3    BIGINT
)
STORED AS carbondata
TBLPROPERTIES
('SPATIAL_INDEX.mygeohash.type'='geohash','SPATIAL_INDEX.mygeohash.sourcecolumns'='
longitude,
latitude','SPATIAL_INDEX.mygeohash.originLatitude'='39.850713','SPATIAL_INDEX.mygeo
hash.gridSize'='50','SPATIAL_INDEX.mygeohash.minLongitude'='115.828503','SPATIAL_IN
DEX.mygeohash.maxLongitude'='720.000000','SPATIAL_INDEX.mygeohash.minLatitude'='39.
850713','SPATIAL_INDEX.mygeohash.maxLatitude'='720.000000','SPATIAL_INDEX'='mygeoha
sh','SPATIAL_INDEX.mygeohash.conversionRatio'='1000000','SORT_COLUMNS'='column1,col
umn2,column3,latitude,longitude');
```

空间索引介绍

空间数据包括多维点、线、矩形、立方体、多边形和其他几何对象。空间数据对象占据空间的某一区域，称为空间范围，通过其位置和边界描述。空间数据可以是点数据，也可以是区域数据。

- 点数据：一个点具有一个空间范围，仅通过其位置描述。它不占用空间，没有相关的边界。点数据由二维空间中的点的集合组成。点可以存储为一对经纬度。
- 区域数据：一个区域有空间范围，有位置和边界。位置可以看作是一个定点在区域内的位置，例如它的质心。在二维中，边界可以可视化为一条线（有限区域，闭环）。区域数据包含一系列区域。

目前仅限于支持点数据，存储点数据。

经纬度可以编码为唯一的 GeoID。Geohash 是 Gustavo Niemeyer 发明的公共域地理编码系统，它将地理位置编码为一串由字母和数字组成的短字符串。它是一种分层的空间数据结构，把空间细分为网格形状的桶，是被称为 Z 阶曲线和通常称为空间填充曲线的许多应用之一。

点在多维中的 Z 值是简单地通过交织其坐标值的二进制表示来计算的，如下图所示。使用 Geohash 创建 GeoID 时，数据按照 GeoID 排序，而不是按照经纬度排序，数据按照空间就近性排序存储。

x: / y:	0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0 000	000000	000001	000100	000101	010000	010001	010100	010101
1 001	000010	000011	000110	000111	010010	010011	010110	010111
2 010	001000	001001	001100	001101	011000	011001	011100	011101
3 011	001010	001011	001110	001111	011010	011011	011110	011111
4 100	100000	100001	100100	100101	110000	110001	110100	110101
5 101	100010	100011	100110	100111	110010	110011	110110	110111
6 110	101000	101001	101100	101101	111000	111001	111100	111101
7 111	101010	101011	101110	101111	111010	111011	111110	111111

建表

GeoHash 编码：

```
create table IF NOT EXISTS carbonTable
(
...
`LONGITUDE`     BIGINT,
`LATITUDE`      BIGINT,
...
)
STORED AS carbondata
```

```
TBLPROPERTIES
('SPATIAL_INDEX.mygeohash.type'='geohash','SPATIAL_INDEX.mygeohash.sourcecolumns'='
longitude,
latitude','SPATIAL_INDEX.mygeohash.originLatitude'='xx.xxxxxx','SPATIAL_INDEX.mygeo
hash.gridSize'='xx','SPATIAL_INDEX.mygeohash.minLongitude'='xxx.xxxxxx','SPATIAL_IN
DEX.mygeohash.maxLongitude'='xxx.xxxxxx','SPATIAL_INDEX.mygeohash.minLatitude'='xx.
xxxxxx','SPATIAL_INDEX.mygeohash.maxLatitude'='xxx.xxxxxx','SPATIAL_INDEX'='mygeoha
sh','SPATIAL_INDEX.mygeohash.conversionRatio'='1000000','SORT_COLUMNS'='column1,col
umn2,column3,latitude,longitude');
```

SPATIAL_INDEX：自定义索引处理器。此处理程序允许用户从表结构列集合中创建新的列。新创建的列名与处理程序名相同。处理程序的 type 和 sourcecolumns 属性是必需的属性。目前，type 属性只支持“geohash”。Carbon 提供一个简单的默认实现类。用户可以通过扩展默认实现类来挂载 geohash 的自定义实现类。该默认处理程序还需提供以下的表属性：

- SPATIAL_INDEX.xxx.originLatitude：Double 类型，坐标原点纬度
- SPATIAL_INDEX.xxx.gridSize：Int 类型，栅格长度（米）
- SPATIAL_INDEX.xxx.minLongitude：Double 类型，最小经度
- SPATIAL_INDEX.xxx.maxLongitude：Double 类型，最大经度
- SPATIAL_INDEX.xxx.minLatitude：Double 类型，最小纬度
- SPATIAL_INDEX.xxx.maxLatitude：Double 类型，最大纬度
- SPATIAL_INDEX.xxx.conversionRatio：Int 类型，将经纬度小数值转换为整型值

用户可以按照上述格式为处理程序添加自己的表属性，并在自定义实现类中访问它们。originLatitude，gridSize 及 conversionRatio 是必选参数，其余属性在 Carbon 中都是可选的。可以使用“SPATIAL_INDEX.xxx.class”属性指定它们的实现类。

默认实现类可以为每一行的 sourcecolumns 生成 handler 列值，并且支持基于 sourcecolumns 的过滤条件查询。生成的 handler 列对用户不可见。除 SORT_COLUMNS 表属性外，任何 DDL 命令和属性都不允许包含 handler 列。

说明

- 生成的 handler 列默认被视为排序列。如果 SORT_COLUMNS 不包含任何 sourcecolumns，则将 handler 列追加到现有的 SORT_COLUMNS 最后。如果在 SORT_COLUMNS 中已经指定了该 handler 列，则它在 SORT_COLUMNS 的顺序将保持不变。
- 如果 SORT_COLUMNS 包含任意的 sourcecolumns，但是没有包含 handler 列，则 handler 列将自动插入到 SORT_COLUMNS 中的 sourcecolumns 之前。
- 如果 SORT_COLUMNS 需要包含任意的 sourcecolumns，那么需要保证 handler 列出现在 sourcecolumns 之前，这样 handler 列才能在排序中生效。

GeoSOT 编码：

```
CREATE TABLE carbontable(
…
longitude DOUBLE,
latitude DOUBLE,
…)
STORED AS carbondata
TBLPROPERTIES ('SPATIAL_INDEX'='xxx',
```

```
'SPATIAL_INDEX.xxx.type'='geosot',
'SPATIAL_INDEX.xxx.sourcecolumns'='longitude, latitude',
'SPATIAL_INDEX.xxx.level'='21',
'SPATIAL_INDEX.xxx.class'='org.apache.carbondata.geo.GeoSOTIndex')
```

表3-51 参数说明

参数	说明
SPATIAL_INDEX	指定表属性”SPATIAL_INDEX”，空间索引列，列名与该属性的值相同。
SPATIAL_INDEX.xxx.type	必填参数，值为 geosot。
SPATIAL_INDEX.xxx.sourcecolumns	必填参数，空间索引列属性，指定计算空间索引的源数据列，需为 2 个存在的列，且类型为 double。
SPATIAL_INDEX.xxx.level	可选参数，用于计算空间索引列。默认值为 17，因为该值可以计算出足够精确的结果，同时拥有良好的性能。
SPATIAL_INDEX.xxx.class	可选参数，用于指定 geo 的实现类，默认为“org.apache.carbondata.geo.GeoSOTIndex”。

使用示例：

```
create table geosot(
timevalue bigint,
longitude double,
latitude double)
stored as carbondata
TBLPROPERTIES ('SPATIAL INDEX'='mygeosot',
'SPATIAL INDEX.mygeosot.type'='geosot',
'SPATIAL INDEX.mygeosot.level'='21',
'SPATIAL_INDEX.mygeosot.sourcecolumns'='longitude, latitude');
```

准备数据

- 准备数据文件 1：geosotdata.csv

```
timevalue,longitude,latitude
1575428400000,116.285807,40.084087
1575428400000,116.372142,40.129503
1575428400000,116.187332,39.979316
1575428400000,116.337069,39.951887
1575428400000,116.359102,40.154684
1575428400000,116.736367,39.970323
1575428400000,116.720179,40.009893
1575428400000,116.346961,40.13355
1575428400000,116.302895,39.930753
1575428400000,116.288955,39.999101
1575428400000,116.17609,40.129953
1575428400000,116.725575,39.981115
1575428400000,116.266922,40.179415
```

```
1575428400000,116.353706,40.156483
1575428400000,116.362699,39.942444
1575428400000,116.325378,39.963129
```

- 准备数据文件 2：geosotdata2.csv

```
timevalue,longitude,latitude
1575428400000,120.17708,30.326882
1575428400000,120.180685,30.326327
1575428400000,120.184976,30.327105
1575428400000,120.189311,30.327549
1575428400000,120.19446,30.329698
1575428400000,120.186965,30.329133
1575428400000,120.177481,30.328911
1575428400000,120.169713,30.325614
1575428400000,120.164563,30.322243
1575428400000,120.171558,30.319613
1575428400000,120.176365,30.320687
1575428400000,120.179669,30.323688
1575428400000,120.181001,30.320761
1575428400000,120.187094,30.32354
1575428400000,120.193574,30.323651
1575428400000,120.186192,30.320132
1575428400000,120.190055,30.317464
1575428400000,120.195376,30.318094
1575428400000,120.160786,30.317094
1575428400000,120.168211,30.318057
1575428400000,120.173618,30.316612
1575428400000,120.181001,30.317316
1575428400000,120.185162,30.315908
1575428400000,120.192415,30.315871
1575428400000,120.161902,30.325614
1575428400000,120.164306,30.328096
1575428400000,120.197093,30.325985
1575428400000,120.19602,30.321651
1575428400000,120.198638,30.32354
1575428400000,120.165421,30.314834
```

导入数据

GeoHash 默认实现类扩展自定义索引抽象类。如果没有配置 handler 属性为自定义的实现类，则使用默认的实现类。用户可以通过扩展默认实现类来挂载 geohash 的自定义实现类。自定义索引抽象类方法包括：

- Init 方法，用来提取、验证和存储 handler 属性。在失败时抛出异常，并显示错误信息。
- Generate 方法，用来生成索引。它为每行数据生成一个索引数据。
- Query 方法，用来对给定输入生成索引值范围列表。

导入命令同普通 Carbon 表：

LOAD DATA inpath '/tmp/*geosotdata.csv*' INTO TABLE geosot OPTIONS ('DELIMITER'= ',');

LOAD DATA inpath '/tmp/*geosotdata2.csv*' INTO TABLE geosot OPTIONS ('DELIMITER'= ',');

说明

geosotdata.csv 和 geosotdata2.csv 表请参考准备数据。

不规则空间集合的聚合查询

查询语句及 Filter UDF

- 根据 polygon 过滤数据

  IN_POLYGON(pointList)

  UDF 输入参数：

参数	类型	说明
pointList	String	将多个点输入为一个字符串，每个点以 **longitude latitude** 表示。经纬度间用空格分隔，每对经纬度用逗号分隔，字符串首尾经纬度一致。

  UDF 输出参数：

参数	类型	说明
inOrNot	Boolean	判断数据是否在指定的 polygon_list 之内。

  使用示例：

```
select longitude, latitude from geosot where IN_POLYGON('116.321011 40.123503, 116.137676 39.947911, 116.560993 39.935276, 116.321011 40.123503');
```

- 根据 polygon 列表过滤数据。

  IN_POLYGON_LIST(polygonList, opType)

  UDF 输入参数：

参数	类型	说明
polygonList	String	将多个 polygon 输入为一个字符串，每个 polygon 以 **POLYGON ((longitude1 latitude1, longitude2 latitude2, …))**表示。注意“POLYGON”后有空格，经纬度间用空格分隔，每对经纬度用逗号分隔，一个 polygon 的首尾经纬度一致。IN_POLYGON_LIST 必须输入 2 个以上 polygon。 一个 polygon 示例： `POLYGON ((116.137676 40.163503, 116.137676 39.935276, 116.560993 39.935276, 116.137676 40.163503))`

参数	类型	说明
opType	String	对多个 polygon 进行并交差操作。 目前支持的操作类型： • OR：A U B U C (假设输入了三个 POLYGON，A、B、C) • AND：A ∩ B ∩ C

UDF 输出参数：

参数	类型	说明
inOrNot	Boolean	判断数据是否在指定的 polygon_list 之内。

使用示例：

```
select longitude, latitude from geosot where IN_POLYGON_LIST('POLYGON
((120.176433 30.327431,120.171283 30.322245,120.181411 30.314540, 120.190509
30.321653,120.185188 30.329358,120.176433 30.327431)), POLYGON ((120.191603
30.328946,120.184179 30.327465,120.181819 30.321464, 120.190359
30.315388,120.199242 30.324464,120.191603 30.328946))', 'OR');
```

- 根据 polyline 列表过滤数据。

  IN_POLYLINE_LIST(polylineList, bufferInMeter)

  UDF 输入参数：

参数	类型	说明
polylineList	String	将多个 polyline 输入为一个字符串，每个 polyline 以 **LINESTRING (longitude1 latitude1, longitude2 latitude2, …)**表示。注意“LINESTRING”后有空格，经纬度间用空格分隔，每组经纬度用逗号分隔。 对多个 polyline 区域内的数据会输出并集结果。 一个 polyline 示例： `LINESTRING (116.137676 40.163503, 116.137676 39.935276, 116.260993 39.935276)`
bufferInMeter	Float	polyline 的 buffer 距离，单位为米。末端使用直角创建缓冲区。

UDF 输出参数：

参数	类型	说明
inOrNot	Boolean	判断数据是否在指定的 polyline_list 之内。

使用示例：

```
select longitude, latitude from geosot where IN_POLYLINE_LIST('LINESTRING (120.184179 30.327465, 120.191603 30.328946, 120.199242 30.324464, 120.190359 30.315388)', 65);
```

- 根据 GeoId 区间列表过滤数据。

  IN_POLYGON_RANGE_LIST(polygonRangeList, opType)

  UDF 输入参数：

参数	类型	说明
polygonRangeList	String	将多个 rangeList 输入为一个字符串，每个 rangeList 以 **RANGELIST (startGeoId1 endGeoId1, startGeoId2 endGeoId2, …)**表示。注意“RANGELIST”后有空格，首尾 GeoId 间用空格分隔，每组 GeoId range 用逗号分隔。 一个 rangeList 示例： `RANGELIST (855279368848 855279368850, 855280799610 855280799612, 855282156300 855282157400)`
opType	String	对多个 rangeList 进行并交差操作。 目前支持的操作类型： • OR：A ∪ B ∪ C (假设输入了三个 RANGELIST，A、B、C) • AND：A ∩ B ∩ C

UDF 输出参数：

参数	类型	说明
inOrNot	Boolean	判断数据是否在指定的 polyRange_list 之内。

使用示例：

```
select mygeosot, longitude, latitude from geosot where
IN_POLYGON_RANGE_LIST('RANGELIST (526549722865860608 526549722865860618, 532555655580483584 532555655580483594)', 'OR');
```

- polygon 连接查询

  IN_POLYGON_JOIN(GEO_HASH_INDEX_COLUMN, POLYGON_COLUMN)

  两张表做 join 查询，一张表为空间数据表（有经纬度列和 GeoHashIndex 列），另一张表为维度表，保存 polygon 数据。

  查询使用 IN_POLYGON_JOIN UDF，参数 GEO_HASH_INDEX_COLUMN 和 polygon 表的 POLYGON_COLUMN。Polygon_column 列是一系列的点（经纬度列）。Polygon 表的每一行的第一个点和最后一个点必须是相同的。Polygon 表的每一行的所有点连接起来形成一个封闭的几何对象。

  UDF 输入参数：

参数	类型	说明
GEO_HASH_INDEX_COLUMN	Long	空间数据表的 GeoHashIndex 列。
POLYGON_COLUMN	String	Polygon 表的 polygon 列，数据为 polygon 的字符串表示。例如，一个 polygon 是 POLYGON ((longitude1 latitude1, longitude2 latitude2, …))

  使用示例：

```
CREATE TABLE polygonTable(
polygon string,
poiType string,
poiId String)
STORED AS carbondata;

insert into polygonTable select 'POLYGON ((120.176433 30.327431,120.171283
30.322245, 120.181411 30.314540,120.190509 30.321653,120.185188
30.329358,120.176433 30.327431))','abc','1';

insert into polygonTable select 'POLYGON ((120.191603 30.328946,120.184179
30.327465, 120.181819 30.321464,120.190359 30.315388,120.199242
30.324464,120.191603 30.328946))','abc','2';

select t1.longitude,t1.latitude from geosot t1
inner join
(select polygon,poiId from polygonTable where poitype='abc') t2
on in_polygon_join(t1.mygeosot,t2.polygon) group by t1.longitude,t1.latitude;
```

- range_list 连接查询

  IN_POLYGON_JOIN_RANGE_LIST(GEO_HASH_INDEX_COLUMN, POLYGON_COLUMN)

  同 IN_POLYGON_JOIN，使用 IN_POLYGON_JOIN_RANGE_LIST UDF 关联空间数据表和 polygon 维度表，关联基于 Polygon_RangeList。直接使用 range list 可以避免 polygon 到 range list 的转换。

  UDF 输入参数：

参数	类型	说明

参数	类型	说明
GEO_HASH_INDEX_COLUMN	Long	空间数据表的 GeoHashIndex 列。
POLYGON_COLUMN	String	Polygon 表的 rangelist 列，数据为 rangeList 的字符串。例如，一个 rangelist 是 RANGELIST (startGeoId1 endGeoId1, startGeoId2 endGeoId2, …)

使用示例：

```
CREATE TABLE polygonTable(
polygon string,
poiType string,
poiId String)
STORED AS carbondata;

insert into polygonTable select 'RANGELIST (526546455897309184
526546455897309284, 526549831217315840 526549831217315850, 532555655580483534
532555655580483584)','xyz','2';

select t1.*
from geosot t1
inner join
(select polygon,poiId from polygonTable where poitype='xyz') t2
on in_polygon_join_range_list(t1.mygeosot,t2.polygon);
```

空间索引工具类 UDF

- GeoID 转栅格行列号。

  GeoIdToGridXy(geoId)

  UDF 输入参数：

参数	类型	说明
geoId	Long	根据 GeoId 计算栅格行列号。

UDF 输出参数：

参数	类型	说明
gridArray	Array[Int]	返回该 geoid 所包含的栅格行列号，以数组的方式返回，第一位为行，第二位为列。

使用示例：

```
select longitude, latitude, mygeohash, GeoIdToGridXy(mygeohash) as GridXY from
geoTable;
```

- 经纬度转 GeoID。

  LatLngToGeoId(latitude, longitude oriLatitude, gridSize)

  UDF 输入参数：

参数	类型	说明
longitude	Long	经度，注：转换后的整数类型。
latitude	Long	纬度，注：转换后的整数类型。
oriLatitude	Double	原点纬度，计算 GeoId 需要参数。
gridSize	Int	栅格大小，计算 GeoId 需要参数。

  UDF 输出参数：

参数	类型	说明
geoId	Long	通过编码获得一个表示经纬度的数。

  使用示例：

```
select longitude, latitude, mygeohash, LatLngToGeoId(latitude, longitude,
39.832277, 50) as geoId from geoTable;
```

- GeoID 转经纬度。

  GeoIdToLatLng(geoId, oriLatitude, gridSize)

  UDF 输入参数：

参数	类型	说明
geoId	Long	根据 GeoId 计算经纬度。
oriLatitude	Double	原点纬度，计算经纬度需要参数。
gridSize	Int	栅格大小，计算经纬度需要参数。

说明

由于 GeoId 由栅格坐标生成，坐标为栅格中心点，则计算出的经纬度是栅格中心点经纬度，与生成该 GeoId 的经纬度可能有[0 度~半个栅格度数]的误差。

  UDF 输出参数：

参数	类型	说明
latitudeAndLongitude	Array[Double]	返回该 geoid 所表示的栅格的中心点的经纬度坐标，以数组的方式返回，第一位为 latitude，第二位为 longitude。

使用示例：

```
select longitude, latitude, mygeohash, GeoIdToLatLng(mygeohash, 39.832277, 50)
as LatitudeAndLongitude from geoTable;
```

- 计算金字塔模型向上汇聚一层的 GeoID。

  ToUpperLayerGeoId(geoId)

  UDF 输入参数：

参数	类型	说明
geoId	Long	根据输入 GeoId 计算金字塔模型上一层 GeoId。

UDF 输出参数：

参数	类型	说明
geoId	Long	金字塔模型上一层 GeoId。

使用示例：

```
select longitude, latitude, mygeohash, ToUpperLayerGeoId(mygeohash) as
upperLayerGeoId from geoTable;
```

- 输入 polygon 获得 GeoID 范围列表。

  ToRangeList(polygon, oriLatitude, gridSize)

  UDF 输入参数：

参数	类型	说明
polygon	String	输入 polygon 字符串，用一组经纬度表示。 经纬度间用空格分隔，每对经纬度间用逗号分隔，首尾经纬度一致。
oriLatitude	Double	原点纬度，计算 GeoId 需要参数。
gridSize	Int	栅格大小，计算 GeoId 需要参数。

UDF 输出参数：

参数	类型	说明
geoIdList	Buffer[Array[Long]]	将 polygon 转换为一串 geoid 的范围列表。

使用示例：

```
select ToRangeList('116.321011 40.123503, 116.137676 39.947911, 116.560993
39.935276, 116.321011 40.123503', 39.832277, 50) as rangeList from geoTable;
```

- 计算金字塔模型向上汇聚一层的 longitude。

  ToUpperLongitude (longitude, gridSize, oriLat)

  UDF 输入参数：

参数	类型	说明
longitude	Long	输入 longitude，用一个长整型表示。
gridSize	Int	栅格大小，计算 longitude 需要参数。
oriLatitude	Double	原点纬度，计算 longitude 需要参数。

  UDF 输出参数：

参数	类型	说明
longitude	Long	返回上一层的 longitude。

  使用示例：

```
select ToUpperLongitude (-23575161504L, 50, 39.832277) as upperLongitude from
geoTable;
```

- 计算金字塔模型向上汇聚一层的 Latitude。

  ToUpperLatitude(Latitude, gridSize, oriLat)

  UDF 输入参数：

参数	类型	说明
latitude	Long	输入 latitude，用一个长整型表示。
gridSize	Int	栅格大小，计算 latitude 需要参数。
oriLatitude	Double	原点纬度，计算 latitude 需要参数。

  UDF 输出参数：

参数	类型	说明
Latitude	Long	返回上一层的 latitude。

  使用示例：

```
select ToUpperLatitude (-23575161504L, 50, 39.832277) as upperLatitude from
geoTable;
```

- 经纬度转 GeoSOT

  LatLngToGridCode(latitude, longitude, level)

UDF 输入参数：

参数	类型	说明
latitude	Double	输入 latitude。
longitude	Double	输入 longitude。
level	Int	输入 level，值区间[0-32]。

UDF 输出参数：

参数	类型	说明
geoId	Long	通过 GeoSOT 编码获得一个表示经纬度的数。

使用示例：

```
select LatLngToGridCode(39.930753, 116.302895, 21) as geoId;
```

3.7 CarbonData 故障处理

3.7.1 当在 Filter 中使用 Big Double 类型数值时，过滤结果与 Hive 不一致

现象描述

当在 filter 中使用更高精度的 double 数据类型的数值时，过滤结果没有按照所使用的 filter 的要求返回正确的值。

可能原因

如果 filter 使用更高精度的 double 数据类型的数值，系统将会对该值四舍五入进行比较，因此在这种情况下，即使小数部分不同，系统仍然会认为 double 数据类型的值是相同的。

定位思路

无。

处理步骤

当需要高精度的数据比较时，可以使用 Decimal 数据类型的数值，例如，在财务应用程序中，equality 和 inequality 检查，以及取整运算，均可使用 Decimal 数据类型的数值。

参考信息

无。

3.7.2 查询性能下降

现象描述

在不同的查询周期内运行查询功能，查询性能会有起伏。

可能原因

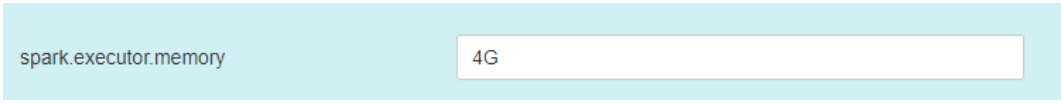
在处理数据加载时，为每个 executor 程序实例配置的内存不足，可能会产生更多的 Java GC（垃圾收集）。当 GC 发生时，会发现查询性能下降。

定位思路

在 Spark UI 上，会发现某些 executors 的 GC 时间明显比其他 executors 高，或者所有的 executors 都表现出高 GC 时间。

处理步骤

登录 Manager 页面，选择“集群 > 服务 > Spark2x > 配置 > 全部配置”，在搜索框搜索“spark.executor.memory”，通过参数“spark.executor.memory”配置更高的内存值。



参考信息

无。

3.8 CarbonData FAQ

3.8.1 为什么对 decimal 数据类型进行带过滤条件的查询时会出现异常输出？

问题

当对 decimal 数据类型进行带过滤条件的查询时，输出结果不正确。

例如，

select * from carbon_table where num = 1234567890123456.22;

输出结果：

```
+------+----------------------+--+
| name |       num        |
+------+----------------------+--+
| IAA  | 1234567890123456.22 |
| IAA  | 1234567890123456.21 |
+------+----------------------+--+
```

回答

为了得到准确的输出结果，需在数字后面加上“BD”。

例如，

select * from carbon_table where num = 1234567890123456.22BD;

输出结果：

```
+------+----------------------+--+
| name |       num        |
+------+----------------------+--+
| IAA  | 1234567890123456.22 |
+------+----------------------+--+
```

3.8.2 如何避免对历史数据进行 minor compaction？

问题

如何避免对历史数据进行 minor compaction？

回答

如果要先加载历史数据，后加载增量数据，则以下步骤可避免对历史数据进行 minor compaction：

1. 加载所有历史数据。
2. 将 major compaction 大小配置为小于历史数据 segment 大小的值。
3. 对历史数据进行一次 major compaction，之后将不会考虑这些 segments 进行 minor compaction。
4. 加载增量数据。
5. 用户可以根据自己的需要配置 minor compaction 阈值。

配置示例和预期输出：

1. 用户将所有历史数据加载到 CarbonData，此数据的一个 segment 的大小假定为 500GB。
2. 用户设置 major compaction 参数的阈值：“carbon.major.compaction.size” = “491520（480gb * 1024）”。其中，*491520* 可配置。
3. 运行 major compaction。由于每个 segment 的大小超过配置值的大小，因此这些 segments 将会被压缩。
4. 加载增量负载。
5. 配置 minor compaction 参数的阈值：“compaction.level.threshold” = “6,6”。

6. 运行 minor compaction。此时只考虑增量负载。

3.8.3 如何在 CarbonData 数据加载时修改默认的组名？

问题

如何在 CarbonData 数据加载时修改默认的组名？

回答

CarbonData 数据加载时，默认的组名为“ficommon”。可以根据需要修改默认的组名。

1. 编辑“carbon.properties”文件。
2. 根据需要修改关键字“carbon.dataload.group.name”的值。其默认值为“ficommon”。

3.8.4 为什么 INSERT INTO CARBON TABLE 失败？

问题

为什么 ***INSERT INTO CARBON TABLE*** 命令无法在日志文件中记录以下信息？

```
Data load failed due to bad record
```

回答

在以下场景中，**INSERT INTO CARBON TABLE** 命令会失败：

- 当源表和目标表的列数据类型不同时，源表中的数据将被视为 Bad Records，则 INSERT INTO 命令会失败。
- 源列上的 aggregartion 函数的结果超过目标列的最大范围，则 INSERT INTO 命令会失败。

  解决方法：

  在进行插入操作时，可在对应的列上使用 cast 函数。

  示例：

  a. 使用 DESCRIBE 命令查询目标表和源表。

  DESCRIBE *newcarbontable*;

  结果：

  ```
  col1 int
  col2 bigint
  ```

  DESCRIBE *sourcetable*;

  结果：

  ```
  col1 int
  col2 int
  ```

  b. 添加 cast 函数以将 BigInt 类型数据转换为 Integer 类型数据。

  INSERT INTO *newcarbontable select col1, cast(col2 as integer) from sourcetable*;

3.8.5 为什么含转义字符的输入数据记录到 Bad Records 中的值与原始数据不同？

问题

为什么含转义字符的输入数据记录到 Bad Records 中的值与原始数据不同？

回答

转义字符以反斜线"\"开头，后跟一个或几个字符。如果输入记录包含类似\t，\b，\n，\r，\f，\'，\"，\\的转义字符，Java 将把转义符'\'和它后面的字符一起处理得到转义后的值。

例如：如果 CSV 数据类似“2010\\10,test”，将这两列插入“String,int”类型时，因为“test”无法转换为 int 类型，表会将这条记录重定向到 Bad Records 中。但记录到 Bad Records 中的值为“2010\10”，Java 会将原始数据中的“\\”转义为“\”。

3.8.6 为什么 Bad Records 导致数据加载性能降低？

问题

为什么 Bad Records 导致数据加载性能降低？

回答

如果数据中存在 Bad Records，并且“BAD_RECORDS_LOGGER_ENABLE”参数值为“true”或“BAD_RECORDS_ACTION”参数值为“redirect”，则由于将失败原因写入日志文件中或将 Bad Records 重定向到原始 CSV 文件中导致的额外的 I/O 开销，数据加载性能就会降低 。

3.8.7 当初始 Executor 为 0 时，为什么 INSERT INTO/LOAD DATA 任务分配不正确，打开的 task 少于可用的 Executor？

问题

当初始 Executor 为 0 时，为什么 INSERT INTO/LOAD DATA 任务分配不正确，打开的 task 少于可用的 Executor？

回答

在这种场景下，CarbonData 会给每个节点分配一个 INSERT INTO 或 LOAD DATA 任务。如果 Executor 不是不同的节点分配的，CarbonData 将会启动较少的 task。

解决措施：

您可以适当增大 Executor 内存和 Executor 核数，以便 YARN 可以在每个节点上启动一个 Executor。具体的配置方法如下：

1. 配置 Executor 核数。

- 将“spark-defaults.conf”中的“spark.executor.cores”配置项或者“spark-env.sh”中的“SPARK_EXECUTOR_CORES”配置项设置为合适大小。
- 在使用 spark-submit 命令时，添加“--executor-cores NUM”参数设置核数。

2. 配置 Executor 内存。
   - 将“spark-defaults.conf”中的“spark.executor.memory”配置项或者“spark-env.sh”中的“SPARK_EXECUTOR_MEMORY”配置项设置为合适大小。
   - 在使用 spark-submit 命令时，添加“--executor-memory MEM”参数设置内存。

3.8.8 为什么并行度大于待处理的 block 数目时，CarbonData 仍需要额外的 executor？

问题

为什么并行度大于待处理的 block 数目时，CarbonData 仍需要额外的 executor？

回答

CarbonData 块分布对于数据处理进行了如下优化：

1. 优化数据处理并行度。
2. 优化了读取块数据的并行性。

为了优化并行数据处理及并行读取块数据，CarbonData 根据块的局域性申请 executor，因此 CarbonData 可获得所有节点上的 executor。

为了优化并行数据处理及并行读取块数据，运用动态分配的用户需配置以下特性。

1. 使用参数“spark.dynamicAllocation.executorIdleTimeout”并将此参数值设置为 15min（或平均查询时间）。
2. 正确配置参数“spark.dynamicAllocation.maxExecutors”，不推荐使用默认值（2048），否则 CarbonData 将申请最大数量的 executor。
3. 对于更大的集群，配置参数“carbon.dynamicAllocation.schedulerTimeout”为 10～15sec，默认值为 5sec。
4. 配置参数“carbon.scheduler.minRegisteredResourcesRatio”为 0.1～1.0，默认值为 0.8。只要达到此参数值，块分布可启动。

3.8.9 为什么在 off heap 时数据加载失败？

问题

为什么在 off heap 时数据加载失败？

回答

YARN Resource Manager 将（Java 堆内存 + “spark.yarn.am.memoryOverhead”）作为内存限制，因此在 off heap 时，内存可能会超出此限制。您需配置参数“spark.yarn.am.memoryOverhead”以增加 memory。

3.8.10 为什么创建 Hive 表失败？

问题

为什么创建 Hive 表失败？

回答

当源表或子查询具有大数据量的 Partition 时，创建 Hive 表失败。执行查询需要很多的 task，此时输出的文件数就会很多，从而导致 driver OOM。

可以在创建 Hive 表的语句中增加 ***distribute by*** 子句来解决这个问题，其中 ***distribute by*** 的字段要选取合适的 cardinality（即 distinct 值的个数）。

distribute by 子句限制了 Hive 表的 Partition 数量。增加 ***distribute by*** 子句后，最终的输出文件数取决于指定列的 cardinality 和“spark.sql.shuffle.partitions”参数值。但如果 distribute by 的字段的 cardinality 值很小，例如，“spark.sql.shuffle.partitions”参数值为 200，但 ***distribute by*** 字段的 cardinality 只有 100，则输出的 200 个文件中，只有其中 100 个文件有数据，剩下的 100 个文件为空文件。也就是说，如果选取的字段的 cardinality 过低，如 1，则会造成严重的数据倾斜，从而严重影响查询性能。

因此，建议选取的 distribute by 字段的 cardinality 个数要大于“spark.sql.shuffle.partitions”参数，可大于 2~3 倍。

示例：

create table hivetable1 as select * from sourcetable1 distribute by col_age;

3.8.11 为什么在 V100R002C50RC1 版本中创建的 CarbonData 表不具有 Hive 特权为非所有者提供的特权？

问题

为什么在 V100R002C50RC1 版本中创建的 CarbonData 表不具有 Hive 特权为非所有者提供的特权？

回答

Hive ACL 在 V100R002C50RC1 版本之后实现，因此无法体现 Hive ACL 特权。

为了支持在 V100R002C50RC1 中创建的 CarbonData 表的 HIVE ACL 特权，必须由表的所有者执行以下两个 **ALTER TABLE** 命令。

ALTER TABLE *$dbname.$tablename SET LOCATION '$carbon.store/$dbname/$tablename';*

ALTER TABLE *$dbname.$tablename SET SERDEPROPERTIES ('path'='$carbon.store/$dbname/$tablename');*

示例：

假设数据库名称为"carbondb"，表名为"carbontable"，Carbon 的储存位置为“hdfs://hacluster/user/hive/warehouse/carbon.store”，执行如下命令：

ALTER TABLE *carbondb.carbontable SET LOCATION 'hdfs://hacluster/user/hive/warehouse/carbon.store/carbondb/carbontable';*

ALTER TABLE *carbondb.carbontable SET SERDEPROPERTIES ('path'='hdfs://hacluster/user/hive/warehouse/carbon.store/carbondb/carbontable');*

3.8.12 如何在不同的 namespaces 上逻辑地分割数据

问题

如何在不同的 namespaces 上逻辑地分割数据？

回答

- 配置：

  要在不同 namespaces 之间逻辑地分割数据，必须更新 HDFS，Hive 和 Spark 的“core-site.xml”文件中的以下配置。

  说明

  改变 Hive 组件将改变 carbonstore 的位置和 warehouse 的位置。

  - HDFS 中的配置
    - fs.defaultFS - 默认文件系统的名称。URI 模式必须设置为“viewfs”。当使用“viewfs”模式时，权限部分必须是“ClusterX”。
    - fs.viewfs.mountable.ClusterX.homedir - 主目录基本路径。每个用户都可以使用在“FileSystem/FileContext”中定义的 getHomeDirectory()方法访问其主目录。
    - fs.viewfs.mountable.default.link.<dir_name> - ViewFS 安装表。

    示例：

    ```
    <property>
    <name>fs.defaultFS</name>
    <value>viewfs://ClusterX/</value>
    </property>
    <property>
    <name>fs.viewfs.mounttable.ClusterX.link./folder1</name>
    <value>hdfs://NS1/folder1</value>
    </property>
    <property>
    <name>fs.viewfs.mounttable.ClusterX.link./folder2</name>
    <value>hdfs://NS2/folder2</value>
    </property>
    ```

  - Hive 和 Spark 中的配置

fs.defaultFS - 默认文件系统的名称。URI 模式必须设置为“viewfs”。 当使用“viewfs”模式时，权限部分必须是“ClusterX”。

- 命令格式：

  LOAD DATA INPATH *'path to data' INTO TABLE table_name OPTIONS ('...');*

  说明

  每当 Spark 配置有 viewFS 文件系统时，当尝试从 HDFS 加载数据时，用户必须在 ***LOAD*** 语句中指定如“viewfs://”这样的路径或相对路径作为文件路径。

- 示例：
  - viewFS 路径举例：

    LOAD DATA INPATH *'viewfs://ClusterX/dir/data.csv' INTO TABLE table_name OPTIONS ('...');*
  - 相对路径举例：

    LOAD DATA INPATH *'/apps/input_data1.txt'* ***INTO TABLE*** *table_name*;

3.8.13 为什么 drop 数据库抛出 Missing Privileges 异常？

问题

为什么 drop 数据库抛出以下异常？

```
Error: org.apache.spark.sql.AnalysisException: Missing Privileges;(State=,code=0)
```

回答

当数据库的所有者执行 ***drop database <database_name> cascade*** 命令（包含其他用户创建的表）时，会抛出此错误。

3.8.14 为什么在 Spark Shell 中不能执行更新命令？

问题

为什么在 Spark Shell 中不能执行更新命令？

回答

本文档中给出的语法和示例是关于 Beeline 的命令，而不是 Spark Shell 中的命令。

若要在 Spark Shell 中使用更新命令，可以使用以下语法。

- 语法 1

  <carbon_context>.sql("UPDATE <CARBON TABLE> SET (column_name1, column_name2, ... column_name n) = (column1_expression , column2_expression , column3_expression ... column n_expression) [WHERE { <filter_condition> }];").show
- 语法 2

<carbon_context>.sql("UPDATE <CARBON TABLE> SET (column_name1, column_name2,) = (select sourceColumn1, sourceColumn2 from sourceTable [WHERE { <filter_condition> }]) [WHERE { <filter_condition> }];").show

示例：

如果 CarbonData 的 context 是“carbon”，那么更新命令如下：

carbon.sql("update carbonTable1 d set (d.column3,d.column5) = (select s.c33 ,s.c55 from sourceTable1 s where d.column1 = s.c11) where d.column1 = 'country' exists(select * from table3 o where o.c2 > 1);").show

3.8.15 如何在 CarbonData 中配置非安全内存？

问题

如何在 CarbonData 中配置非安全内存？

回答

在 Spark 配置中，“spark.yarn.executor.memoryOverhead”参数的值应大于 CarbonData 配置参数“sort.inmemory.size.inmb” 与“Netty offheapmemory required”参数值的总和，或者“carbon.unsafe.working.memory.in.mb” 、“carbon.sort.inememory.storage.size.in.mb” 与 “Netty offheapmemory required”参数值的总和。否则，如果堆外（off heap）访问超出配置的 executor 内存，则 YARN 可能会停止 executor。

“Netty offheapmemory required”说明：当“spark.shuffle.io.preferDirectBufs”设为 true 时，Spark 中 netty 传输服务从"spark.yarn.executor.memoryOverhead"中拿掉部分堆内存[~ 384 MB or 0.1 x 执行器内存]。

详细信息请参考常见配置 executor 堆外内存大小。

3.8.16 设置了 HDFS 存储目录的磁盘空间配额，CarbonData 为什么会发生异常？

问题

设置了 HDFS 存储目录的磁盘空间配额，CarbonData 为什么会发生异常。

回答

创建、加载、更新表或进行其他操作时，数据会被写入 HDFS。若 HDFS 目录的磁盘空间配额不足，则操作失败并抛出以下异常。

```
org.apache.hadoop.hdfs.protocol.DSQuotaExceededException: The DiskSpace quota of /user/tenant is exceeded: quota = 314572800 B = 300 MB but diskspace consumed = 402653184 B = 384 MB at org.apache.hadoop.hdfs.server.namenode.DirectoryWithQuotaFeature.verifyStoragespaceQuota(DirectoryWithQuotaFeature.java:211) at org.apache.hadoop.hdfs.server.namenode.DirectoryWithQuotaFeature.verifyQuota(DirectoryWithQuotaFeature.java:239) at
```

```
org.apache.hadoop.hdfs.server.namenode.FSDirectory.verifyQuota(FSDirectory.java:941)
at
org.apache.hadoop.hdfs.server.namenode.FSDirectory.updateCount(FSDirectory.java:745)
```

若发生此异常，请为租户配置足够的磁盘空间配额。

例如：

需要的磁盘空间配置可以按照如下方法计算：

如果 HDFS 的副本数为 3， HDFS 默认的块大小为 128MB，则最小需要 384MB 的磁盘空间用于写表的 schema 文件到 HDFS 上。计算公式：no. of block x block_size x replication_factor of the schema file = 1 x 128 x 3 = 384 MB

说明

数据加载时，由于默认块大小为 1024MB，每个 fact 文件需要的最小空间为 3072MB。

3.8.17 为什么数据查询/加载失败，且抛出“org.apache.carbondata.core.memory.MemoryException: Not enough memory”异常？

问题

为什么数据查询/加载失败，且抛出“org.apache.carbondata.core.memory.MemoryException: Not enough memory”异常？

回答

当执行器中此次数据查询和加载所需要的堆外内存不足时，便会抛出此异常。

在这种情况下,请增大“carbon.unsafe.working.memory.in.mb”和“spark.yarn.executor.memoryOverhead”的值。

详细信息请参考如何在 CarbonData 中配置非安全内存？

该内存被数据查询和加载共享。所以如果加载和查询需要同时进行，建议将“carbon.unsafe.working.memory.in.mb”和“spark.yarn.executor.memoryOverhead”的值配置为 2048 MB 以上。

可以使用以下公式进行估算：

数据加载所需内存：

(“carbon.number.of.cores.while.loading”的值[默认值 = 6]) x 并行加载数据的表格 x (“offheap.sort.chunk.size.inmb”的值[默认值 = 64 MB] + “carbon.blockletgroup.size.in.mb”的值[默认值 = 64 MB] + 当前的压缩率[64 MB/3.5])

= ~900 MB 每表格

数据查询所需内存：

(SPARK_EXECUTOR_INSTANCES. [默认值 = 2]) x (carbon.blockletgroup.size.in.mb [默认值 = 64 MB] +“carbon.blockletgroup.size.in.mb”解压内容[默认值 = 64 MB * 3.5]) x (每个执行器核数[默认值 = 1])

= ~ 600 MB

3.8.18 开启防误删下，为什么 Carbon 表没有执行 drop table 命令，回收站中也会存在该表的文件？

问题

开启防误删下，为什么 Carbon 表没有执行 drop table 命令，回收站中也会存在该表的文件？

回答

在 Carbon 适配防误删后，调用文件删除命令，会将删除的文件放入回收站中。在 insert、load 等命令中会有中间文件.carbonindex 文件的删除，所以在未执行 drop table 命令的时候，回收站中也可能会存在该表的文件。如果这个时候再执行 drop table 命令，那么按照回收站机制，会生成一个带时间戳的该表目录，该目录中的文件是完整的。

4 使用 ClickHouse

4.1 从零开始使用 ClickHouse

ClickHouse 是面向联机分析处理的列式数据库，支持 SQL 查询，且查询性能好，特别是基于大宽表的聚合分析查询性能非常优异，比其他分析型数据库速度快一个数量级。

前提条件

已安装客户端，例如安装目录为“/opt/hadoopclient”。以下操作的客户端目录只是举例，请根据实际安装目录修改。在使用客户端前，需要先下载并更新客户端配置文件，确认 Manager 的主管理节点后才能使用客户端。

操作步骤

步骤 1 以客户端安装用户，登录安装客户端的节点。

步骤 2 执行以下命令，切换到客户端安装目录。

cd /opt/hadoopclient

步骤 3 执行以下命令配置环境变量。

source bigdata_env

步骤 4 如果当前集群已启用 Kerberos 认证，执行以下命令认证当前用户，当前用户需要具有创建 ClickHouse 表的权限，具体请参见“组件操作指南 > 使用 ClickHouse > 用户管理及认证 > ClickHouse 用户及权限管理”章节，为用户绑定对应角色。如果当前集群未启用 Kerberos 认证，则无需执行本步骤。

1. 如果是 MRS 3.1.0 版本集群，则需要先执行：**export CLICKHOUSE_SECURITY_ENABLED=true**
2. **kinit** *组件业务用户*

   例如，**kinit** clickhouseuser。

步骤 5 执行 ClickHouse 组件的客户端命令。

执行 **clickhouse -h**，查看 ClickHouse 组件命令帮助。

回显信息如下：

```
Use one of the following commands:
clickhouse local [args]
clickhouse client [args]
clickhouse benchmark [args]
clickhouse server [args]
clickhouse performance-test [args]
clickhouse extract-from-config [args]
clickhouse compressor [args]
clickhouse format [args]
clickhouse copier [args]
clickhouse obfuscator [args]
...
```

详细命令使用请参考：https://clickhouse.tech/docs/zh/operations/。

MRS 3.1.0 及之后版本，使用 **clickhouse client** 命令连接 ClickHouse 服务端：

- 例如，当前集群未启用 Kerberos 认证，使用 ssl 安全方式登录：

  clickhouse client --host *ClickHouse 的实例 IP* **--user** *用户名* **--password** *密码* **--port** 9440 **--secure**

- 例如，当前集群已启用 Kerberos 认证，使用 ssl 安全方式登录。

  Kerberos 集群场景下没有默认用户，必须在 Manager 上创建用户，详细参考“组件操作指南 > 使用 ClickHouse > 用户管理及认证 > ClickHouse 用户及权限管理”。

  使用 kinit 认证成功后，客户端登录时可以不携带--user 和--password 参数，即使用 kinit 认证的用户登录。

  clickhouse client --host *ClickHouse 的实例 IP* **--port** 9440 **--secure**

  相关参数使用说明如下表：

表4-1 clickhouse client 命令行参数说明

参数名	参数说明
--host	服务端的 host 名称，默认是 localhost。您可以选择使用 ClickHouse 实例所在节点主机名或者 IP 地址。 **说明** ClickHouse 的实例 IP 地址可登录集群 FusionInsight Manager，然后选择“集群 > 服务 > ClickHouse > 实例”，获取 ClickHouseServer 实例对应的业务 IP 地址。
--port	连接的端口。 • 如果使用 ssl 安全连接则默认端口为 9440，并且需要携带参数--secure。具体的端口值可通过 ClickHouseServer 实例配置搜索“tcp_port_secure”参数获取。 • 如果使用非 ssl 安全连接则默认端口为 9000，不需要携带参数--secure。具体的端口值可通过 ClickHouseServer 实例配置搜索“tcp_port”参数获取。

参数名	参数说明
--user	用户名。 可以在 Manager 上创建该用户名并绑定对应的角色权限，具体可以参考“组件操作指南 > 使用 ClickHouse > 用户管理及认证 > ClickHouse 用户及权限管理”。 • 如果当前集群已启用 Kerberos 认证，使用 kinit 认证成功后，客户端登录时可以不携带--user 和--password 参数，即使用 kinit 认证的用户登录。Kerberos 集群场景下没有默认用户，必须在 Manager 上创建该用户名。 • 如果当前集群未启用 Kerberos 认证，客户端登录时可以指定 Manager 上创建的用户和密码。不携带用户和密码参数时则默认使用 **default** 用户登录。
--password	密码。 默认值：空字符串。该参数和--user 参数配套使用，可以在 Manager 上创建用户名时设置该密码。
--query	使用非交互模式查询。
--database	默认当前操作的数据库。默认值：服务端默认的配置（默认是 default）。
--multiline	如果指定，允许多行语句查询（Enter 仅代表换行，不代表查询语句完结）。
--multiquery	如果指定，允许处理用;号分隔的多个查询，只在非交互模式下生效。
--format	使用指定的默认格式输出结果。
--vertical	如果指定，默认情况下使用垂直格式输出结果。在这种格式中，每个值都在单独的行上打印，适用显示宽表的场景。
--time	如果指定，非交互模式下会打印查询执行的时间到 stderr 中。
--stacktrace	如果指定，如果出现异常，会打印堆栈跟踪信息。
--config-file	配置文件的名称。
--secure	如果指定，将通过 ssl 安全模式连接到服务器。
--history_file	存放命令历史的文件的路径。
--param_<name>	带有参数的查询，并将值从客户端传递给服务器。具体用法详见 https://clickhouse.tech/docs/zh/interfaces/cli/#cli-queries-with-parameters。

----结束

4.2 ClickHouse 表引擎介绍

背景介绍

表引擎在 ClickHouse 中的作用十分关键，不同的表引擎决定了：

- 数据存储和读取的位置
- 支持哪些查询方式
- 能否并发式访问数据
- 能不能使用索引
- 是否可以执行多线程请求
- 数据复制使用的参数

其中 MergeTree 和 Distributed 是 ClickHouse 表引擎中最重要，也是最常使用的两个引擎，本文将重点进行介绍。

其他表引擎详细可以参考官网链接：https://clickhouse.tech/docs/en/engines/table-engines。

MergeTree 系列引擎

MergeTree 用于高负载任务的最通用和功能最强大的表引擎，其主要有以下关键特征：

- 基于分区键（partitioning key）的数据分区分块存储
- 数据索引排序（基于 primary key 和 order by）
- 支持数据复制（带 Replicated 前缀的表引擎）
- 支持数据抽样

在写入数据时，该系列引擎表会按照分区键将数据分成不同的文件夹，文件夹内每列数据为不同的独立文件，以及创建数据的序列化索引排序记录文件。该结构使得数据读取时能够减少数据检索时的数据量，极大的提高查询效率。

- MergeTree

  建表语法：

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]
(
    name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1] [TTL expr1],
    name2 [type2] [DEFAULT|MATERIALIZED|ALIAS expr2] [TTL expr2],
    ...
    INDEX index_name1 expr1 TYPE type1(...) GRANULARITY value1,
    INDEX index_name2 expr2 TYPE type2(...) GRANULARITY value2
) ENGINE = MergeTree()
ORDER BY expr
[PARTITION BY expr]
[PRIMARY KEY expr]
[SAMPLE BY expr]
[TTL expr [DELETE|TO DISK 'xxx'|TO VOLUME 'xxx'], ...]
[SETTINGS name=value, ...]
```

  使用示例：

```
CREATE TABLE default.test (
  name1 DateTime,
  name2 String,
  name3 String,
  name4 String,
  name5 Date,
  ...
) ENGINE = MergeTree()
PARTITION BY toYYYYMM(name5)
ORDER BY (name1, name2)
SETTINGS index_granularity = 8192
```

示例参数说明如下：

- **ENGINE = MergeTree()**：MergeTree 表引擎。
- **PARTITION BY** toYYYYMM(name4)：分区，示例数据将以月份为分区，每个月份一个文件夹。
- **ORDER BY**：排序字段，支持多字段的索引排序，第一个相同的时候按照第二个排序依次类推。
- **index_granularity = 8192**：排序索引的颗粒度，每 8192 条数据记录一个排序索引值。

如果被查询的数据存在于分区或排序字段中，能极大降低数据查找时间。

- ReplacingMergeTree

  该引擎和 MergeTree 的不同之处在于它会删除排序键值相同的重复项。ReplacingMergeTree 适合于清除重复数据节省存储空间，但是它不保证重复数据不出现，一般不建议使用。

  建表语法：

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]
(
   name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1],
   name2 [type2] [DEFAULT|MATERIALIZED|ALIAS expr2],
   ...
) ENGINE = ReplacingMergeTree([ver])
[PARTITION BY expr]
[ORDER BY expr]
[SAMPLE BY expr]
[SETTINGS name=value, ...]
```

- SummingMergeTree

  当合并 SummingMergeTree 表的数据片段时，ClickHouse 会把所有具有相同主键的行合并为一行，该行包含了被合并的行中具有数值数据类型的列的汇总值。如果主键的组合方式使得单个键值对应于大量的行，则可以显著的减少存储空间并加快数据查询的速度。

  建表语法：

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]
(
   name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1],
   name2 [type2] [DEFAULT|MATERIALIZED|ALIAS expr2],
   ...
) ENGINE = SummingMergeTree([columns])
[PARTITION BY expr]
```

```
[ORDER BY expr]
[SAMPLE BY expr]
[SETTINGS name=value, ...]
```

使用示例：

创建一个 SummingMergeTree 表 testTable：

```
CREATE TABLE testTable
(
   id UInt32,
   value UInt32
)
ENGINE = SummingMergeTree()
ORDER BY id
```

插入表数据：

```
INSERT INTO testTable Values(5,9),(5,3),(4,6),(1,2),(2,5),(1,4),(3,8);
INSERT INTO testTable
Values(88,5),(5,5),(3,7),(3,5),(1,6),(2,6),(4,7),(4,6),(43,5),(5,9),(3,6);
```

在未合并 parts 查询所有数据：

```
SELECT * FROM testTable
┌─id─┬─value─┐
│  1 │     6 │
│  2 │     5 │
│  3 │     8 │
│  4 │     6 │
│  5 │    12 │
└────┴───────┘
┌─id─┬─value─┐
│  1 │     6 │
│  2 │     6 │
│  3 │    18 │
│  4 │    13 │
│  5 │    14 │
│ 43 │     5 │
│ 88 │     5 │
└────┴───────┘
```

ClickHouse 还没有汇总所有行，如果需要通过 ID 进行汇总聚合，需要用到 sum 和 GROUP BY 子句：

```
SELECT id, sum(value) FROM testTable GROUP BY id
┌─id─┬─sum(value)─┐
│  4 │         19 │
│  3 │         26 │
│ 88 │          5 │
│  2 │         11 │
│  5 │         26 │
│  1 │         12 │
│ 43 │          5 │
└────┴────────────┘
```

手工执行合并操作：

```
OPTIMIZE TABLE testTable
```

此时再查询 testTable 表数据：

```
SELECT * FROM testTable
┌─id─┬─value─┐
│  1 │    12 │
│  2 │    11 │
│  3 │    26 │
│  4 │    19 │
│  5 │    26 │
│ 43 │     5 │
│ 88 │     5 │
└────┴───────┘
```

SummingMergeTree 根据 ORDER BY 排序键作为聚合数据的条件 Key。即如果排序 key 是相同的，则会合并成一条数据，并对指定的合并字段进行聚合。

后台执行合并操作时才会进行数据的预先聚合，而合并操作的执行时机无法预测，所以可能存在部分数据已经被预先聚合、部分数据尚未被聚合的情况。因此，在执行聚合计算时，SQL 中仍需要使用 GROUP BY 子句。

- AggregatingMergeTree

AggregatingMergeTree 是预先聚合引擎的一种，用于提升聚合计算的性能。AggregatingMergeTree 引擎能够在合并分区时，按照预先定义的条件聚合数据，同时根据预先定义的聚合函数计算数据并通过二进制的格式存入表内。

建表语法：

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]
(
    name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1],
    name2 [type2] [DEFAULT|MATERIALIZED|ALIAS expr2],
    ...
) ENGINE = AggregatingMergeTree()
[PARTITION BY expr]
[ORDER BY expr]
[SAMPLE BY expr]
[TTL expr]
[SETTINGS name=value, ...]
```

使用示例：

AggregatingMergeTree 无单独参数设置，在分区合并时，在每个数据分区内，会按照 ORDER BY 聚合，使用何种聚合函数，对哪些列字段计算，则是通过定义 AggregateFunction 函数类型实现，例如：

```
create table test_table (
name1 String,
name2 String,
name3 AggregateFunction(uniq,String),
name4 AggregateFunction(sum,Int),
name5 DateTime
) ENGINE = AggregatingMergeTree()
PARTITION BY toYYYYMM(name5)
ORDER BY (name1,name2)
PRIMARY KEY name1;
```

AggregateFunction 类型的数据在写入和查询时需要分别调用*state、*merge 函数，*表示定义字段类型时使用的聚合函数。如上示例表 test_table 定义的 name3、name4 字段分别使用了 uniq、sum 函数，那么在写入数据时需要调用 uniqState、sumState 函数，并使用 INSERT SELECT 语法。

```
insert into test_table select
'8','test1',uniqState('name1'),sumState(toInt32(100)),'2021-04-30 17:18:00';
insert into test_table select
'8','test1',uniqState('name1'),sumState(toInt32(200)),'2021-04-30 17:18:00';
```

在查询数据时也需要调用对应的函数 uniqMerge、sumMerge：

```
select name1,name2,uniqMerge(name3),sumMerge(name4) from test_table group by
name1,name2;
┌name1┬─name2─┬─uniqMerge(name3)─┬─sumMerge(name4)─┐
│ 8      │   test1 │                1 │             300 │
└─────┴──────┴──────────────┴──────────┘
```

AggregatingMergeTree 更常用的方式是结合物化视图使用，物化视图即其它数据表上层的一种查询视图。详细可以参考：https://clickhouse.tech/docs/en/engines/table-engines/mergetree-family/aggregatingmergetree/

- CollapsingMergeTree

  CollapsingMergeTree 它通过定义一个 sign 标记位字段记录数据行的状态。如果 sign 标记为 1，则表示这是一行有效的数据；如果 sign 标记为-1，则表示这行数据需要被删除。

  建表语法：

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]
(
    name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1],
    name2 [type2] [DEFAULT|MATERIALIZED|ALIAS expr2],
    ...
) ENGINE = CollapsingMergeTree(sign)
[PARTITION BY expr]
[ORDER BY expr]
[SAMPLE BY expr]
[SETTINGS name=value, ...]
```

  使用示例：

  具体的使用示例可以参考：https://clickhouse.tech/docs/en/engines/table-engines/mergetree-family/collapsingmergetree/。

- VersionedCollapsingMergeTree

  VersionedCollapsingMergeTree 表引擎在建表语句中新增了一列 version，用于在乱序情况下记录状态行与取消行的对应关系。主键相同，且 Version 相同、Sign 相反的行，在 Compaction 时会被删除。

  建表语法：

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]
(
    name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1],
    name2 [type2] [DEFAULT|MATERIALIZED|ALIAS expr2],
    ...
) ENGINE = VersionedCollapsingMergeTree(sign, version)
[PARTITION BY expr]
[ORDER BY expr]
[SAMPLE BY expr]
[SETTINGS name=value, ...]
```

  使用示例：

具体的使用示例可以参考：https://clickhouse.tech/docs/en/engines/table-engines/mergetree-family/versionedcollapsingmergetree/。

- GraphiteMergeTree

  GraphiteMergeTree 引擎用来存储时序数据库 Graphite 的数据。

  建表语法：

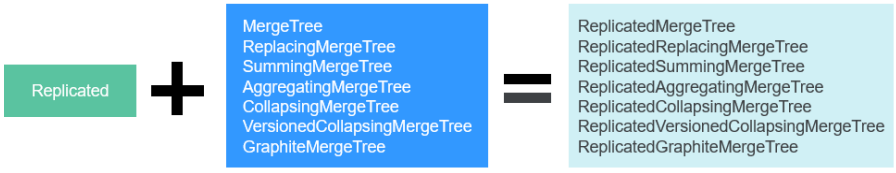
```
CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]
(
    Path String,
    Time DateTime,
    Value <Numeric_type>,
    Version <Numeric_type>
    ...
) ENGINE = GraphiteMergeTree(config_section)
[PARTITION BY expr]
[ORDER BY expr]
[SAMPLE BY expr]
[SETTINGS name=value, ...]
```

  使用示例：

  具体的使用示例可以参考：https://clickhouse.tech/docs/en/engines/table-engines/mergetree-family/graphitemergetree/。

Replicated*MergeTree 引擎

ClickHouse 中的所有 MergeTree 家族引擎前面加上 Replicated 就成了支持副本的合并树引擎。



Replicated 系列引擎借助 ZooKeeper 实现数据的同步，创建 Replicated 复制表时通过注册到 ZooKeeper 上的信息实现同一个分片的所有副本数据进行同步。

Replicated 表引擎的创建模板：

```
ENGINE = Replicated*MergeTree('ZooKeeper 存储路径','副本名称', ...)
```

Replicated 表引擎需指定两个参数：

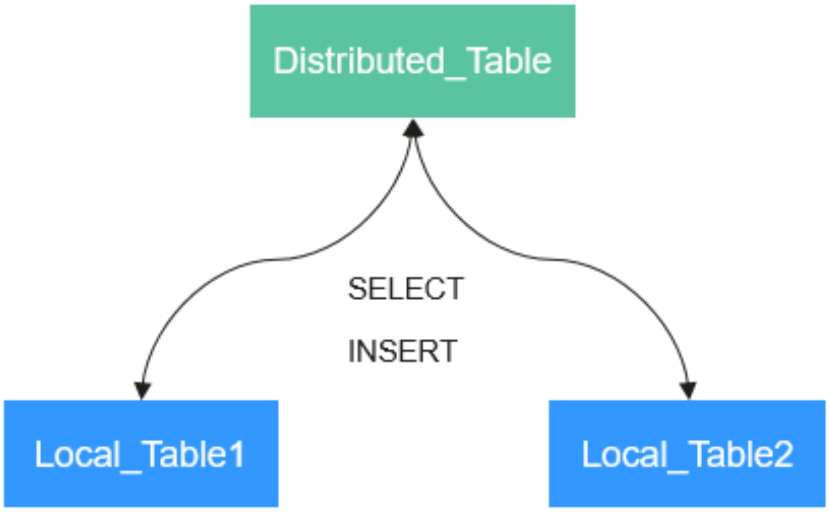
- ZooKeeper 存储路径：ZooKeeper 中该表相关数据的存储路径，建议规范化，如：*/clickhouse/tables/{shard}/数据库名/表名*。
- 副本名称，一般用**{replica}**即可。

Replicated 表引擎使用示例可以参考：ClickHouse 表创建。

Distributed 表引擎

Distributed 表引擎本身不存储任何数据，而是作为数据分片的透明代理，能够自动路由数据到集群中的各个节点，分布式表需要和其他本地数据表一起协同工作。分布式表会将接收到的读写任务分发到各个本地表，而实际上数据的存储在各个节点的本地表中。

图4-1 Distributed 原理图



Distributed 表引擎的创建模板：

```
ENGINE = Distributed(cluster_name, database_name, table_name, [sharding_key])
```

Distributed 表参数解析如下：

- cluster_name：集群名称，在对分布式表执行读写的过程中，使用集群的配置信息查找对应的 ClickHouse 实例节点。
- database_name：数据库名称。
- table_name：数据库下对应的本地表名称，用于将分布式表映射到本地表上。
- sharding_key：分片键（可选参数），分布式表会按照这个规则，将数据分发到各个本地表中。

Distributed 表引擎使用示例：

```
--先创建一个表名为 test 的 ReplicatedMergeTree 本地表
CREATE TABLE default.test ON CLUSTER default_cluster_1
(
    `EventDate` DateTime,
    `id` UInt64
)
ENGINE = ReplicatedMergeTree('/clickhouse/tables/{shard}/default/test', '{replica}')
```

```
PARTITION BY toYYYYMM(EventDate)
ORDER BY id

--基于本地表 test 创建表名为 test_all 的 Distributed 表
CREATE TABLE default.test_all ON CLUSTER default_cluster_1
(
    `EventDate` DateTime,
    `id` UInt64
)
ENGINE = Distributed(default_cluster_1, default, test, rand())
```

分布式表创建规则：

- 创建 Distributed 表时需加上 **on cluster** *cluster_name*，这样建表语句在某一个 ClickHouse 实例上执行一次即可分发到集群中所有实例上执行。
- 分布式表通常以本地表加“_all”命名。它与本地表形成一对多的映射关系，之后可以通过分布式表代理操作多张本地表。
- 分布式表的表结构尽量和本地表的结构一致。如果不一致，在建表时不会报错，但在查询或者插入时可能会抛出异常。

4.3 ClickHouse 表创建

ClickHouse 依靠 ReplicatedMergeTree 引擎与 ZooKeeper 实现了复制表机制，用户在创建表时可以通过指定引擎选择该表是否高可用，每张表的分片与副本都是互相独立的。

同时 ClickHouse 依靠 Distributed 引擎实现了分布式表机制，在所有分片（本地表）上建立视图进行分布式查询，使用很方便。ClickHouse 有数据分片（shard）的概念，这也是分布式存储的特点之一，即通过并行读写提高效率。

CPU 架构为鲲鹏计算的 ClickHouse 集群表引擎不支持使用 HDFS 和 Kafka。

查看 ClickHouse 服务 cluster 等环境参数信息

步骤 1 参考从零开始使用 ClickHouse 使用 ClickHouse 客户端连接到 ClickHouse 服务端。

步骤 2 查询集群标识符 cluster 等其他环境参数信息。

select cluster,shard_num,replica_num,host_name from system.clusters;

```
SELECT
   cluster,
   shard_num,
   replica_num,
   host_name
FROM system.clusters

┌─cluster─────────────┬─shard_num─┬─replica_num─┬─host_name────────┐
│ default_cluster_1   │         1 │           1 │ node-master1dOnG │
│ default_cluster_1   │         1 │           2 │ node-group-1tXED0001
│
│ default_cluster_1   │         2 │           1 │ node-master2OXQS │
│ default_cluster_1   │         2 │           2 │ node-group-1tXED0002
```

```
|
| default_cluster_1           |       3   |         1   | node-master3QsRI          |
| default_cluster_1           |       3   |         2   | node-group-1tXED0003
|
└─────────────── ┴─────── ┴──────── ┴──────────────┘

6 rows in set. Elapsed: 0.001 sec.
```

步骤 3 查询分片标识符 shard 和副本标识符 replica。

select * from system.macros;

```
SELECT *
FROM system.macros

┌─macro────┬─substitution─────┐
| id         | 76                 |
| replica    | node-master3QsRI      |
| shard      | 3                  |
└──────┴──────────────┘

3 rows in set. Elapsed: 0.001 sec.
```

----结束

创建本地复制表和分布式表

步骤 1 客户端登录 ClickHouse 节点，例如：**clickhouse client --host** *node-master3QsRI* **--multiline --port 9440 --secure;**

说明

node-master3QsRI 参数为查看 ClickHouse 服务 cluster 等环境参数信息中步骤 2 对应的 host_name 参数的值。

步骤 2 使用 ReplicatedMergeTree 引擎创建复制表。

详细的语法说明请参考：https://clickhouse.tech/docs/zh/engines/table-engines/mergetree-family/replication/#creating-replicated-tables。

例如，如下在 default_cluster_1 集群节点上和 default 数据库下创建表名为 test 的 ReplicatedMergeTree 表：

CREATE TABLE *default.test* **ON CLUSTER** *default_cluster_1*

(

`EventDate` DateTime,

`id` UInt64

)

ENGINE = ReplicatedMergeTree('*/clickhouse/tables/{shard}/default/test***', '***{replica}***')**

PARTITION BY toYYYYMM(EventDate)

ORDER BY id;

参数说明如下：

- ON CLUSTER 语法表示分布式 DDL，即执行一次就可在集群所有实例上创建同样的本地表。
- default_cluster_1 为查看 ClickHouse 服务 cluster 等环境参数信息中步骤 2 查询到的 cluster 集群标识符。

⚠ 注意

ReplicatedMergeTree 引擎族接收两个参数：

- ZooKeeper 中该表相关数据的存储路径。

  该路径必须在/clickhouse 目录下，否则后续可能因为 ZooKeeper 配额不够导致数据插入失败。

  为了避免不同表在 ZooKeeper 上数据冲突，目录格式必须按照如下规范填写：

  /clickhouse/tables/{shard}/default/test，其中/clickhouse/tables/{shard}为固定值，default 为数据库名，test 为创建的表名。

- 副本名称，一般用**{replica}**即可。

```
CREATE TABLE default.test ON CLUSTER default_cluster_1
(
    `EventDate` DateTime,
    `id` UInt64
)
ENGINE = ReplicatedMergeTree('/clickhouse/tables/{shard}/default/test', '{replica}')
PARTITION BY toYYYYMM(EventDate)
ORDER BY id

┌─host──────────────────┬─port─┬─status─┬─error─┬─num_hosts_remaining─┬─num_hosts_ac
tive─┐
│ node-group-1tXED0002  │ 9000 │      0 │       │                   5
│                   3   │
│ node-group-1tXED0003  │ 9000 │      0 │       │                   4
│                   3   │
│ node-master1dOnG      │ 9000 │      0 │       │                   3
│                   3   │
└───────────────────────┴──────┴────────┴───────┴─────────────────────┴──────────┘
┌─host──────────────────┬─port─┬─status─┬─error─┬─num hosts remaining─┬─num hosts ac
tive─┐
│ node-master3QsRI      │ 9000 │      0 │       │                   2
│                   0   │
│ node-group-1tXED0001  │ 9000 │      0 │       │                   1
│                   0   │
│ node-master2OXQS      │ 9000 │      0 │       │                   0
│                   0   │
└───────────────────────┴──────┴────────┴───────┴─────────────────────┴──────────┘

6 rows in set. Elapsed: 0.189 sec.
```

步骤 3　使用 Distributed 引擎创建分布式表。

例如，以下将在 default_cluster_1 集群节点上和 default 数据库下创建名为 test_all 的 Distributed 表：

CREATE TABLE *default.test_all* **ON CLUSTER** *default_cluster_1*

(

`EventDate` DateTime,

`id` UInt64

)

ENGINE = Distributed(*default_cluster_1, default, test, rand()***);**

```
CREATE TABLE default.test_all ON CLUSTER default_cluster_1
(
    `EventDate` DateTime,
    `id` UInt64
)
ENGINE = Distributed(default_cluster_1, default, test, rand())

┌─host──────────────────┬─port─┬─status─┬─error─┬─num_hosts_remaining─┬─num_hosts_ac
tive─┐
│ node-group-1tXED0002  │ 9000 │      0 │       │                   5 
│                    0 │
│ node-master3QsRI      │ 9000 │      0 │       │                   4 
│                    0 │
│ node-group-1tXED0003  │ 9000 │      0 │       │                   3 
│                    0 │
│ node-group-1tXED0001  │ 9000 │      0 │       │                   2 
│                    0 │
│ node-master1dOnG      │ 9000 │      0 │       │                   1 
│                    0 │
│ node-master2OXQS      │ 9000 │      0 │       │                   0 
│                    0 │
└───────────────────────┴──────┴────────┴───────┴─────────────────────┴──────────┘

6 rows in set. Elapsed: 0.115 sec.
```

说明

Distributed 引擎需要以下几个参数：

- default_cluster_1 为查看 ClickHouse 服务 cluster 等环境参数信息中步骤 2 查询到的 cluster 集群标识符。
- default 本地表所在的数据库名称。
- test 为本地表名称，该例中为步骤 2 中创建的表名。
- （可选的）分片键（sharding key）

该键与 config.xml 中配置的分片权重（weight）一同决定写入分布式表时的路由，即数据最终落到哪个物理表上。它可以是表中一列的原始数据（如 site_id），也可以是函数调用的结果，如上面的 SQL 语句采用了随机值 rand()。注意该键要尽量保证数据均匀分布，另外一个常用的操作是采用区分度较高的列的哈希值，如 intHash64(user_id)。

----结束

ClickHouse 表数据操作

步骤 1 客户端登录 ClickHouse 节点。例如：

clickhouse client --host *node-master3QsRI* **--multiline --port 9440 --secure;**

说明

node-master3QsRI 参数为查看 ClickHouse 服务 cluster 等环境参数信息中步骤 2 对应的 host_name 参数的值。

步骤 2 参考创建本地复制表和分布式表创建表后，可以插入数据到本地表。

例如插入数据到本地表：test

insert into test values(toDateTime(now()), rand());

步骤 3 查询本地表信息。

例如查询步骤 2 中的表 test 数据信息：

select * from test;

```
SELECT *
FROM test

┌─────────EventDate─┬──────────id─┐
│ 2020-11-05 21:10:42           │ 1596238076          │
└──────────────────── ┴─────────────┘


1 rows in set. Elapsed: 0.002 sec.
```

步骤 4 查询 Distributed 分布式表。

例如步骤 3 中因为分布式表 test_all 基于 test 创建，所以 test_all 表也能查询到和 test 相同的数据。

select * from test_all;

```
SELECT *
FROM test_all

┌─────────EventDate─┬──────────id─┐
│ 2020-11-05 21:10:42           │ 1596238076          │
└──────────────────── ┴─────────────┘


1 rows in set. Elapsed: 0.004 sec.
```

步骤 5 切换登录节点为相同 shard_num 的 shard 节点，并且查询当前表信息，能查询到相同的表数据。

例如，退出原有登录节点：**exit;**

切换到节点 node-group-1tXED0003：

clickhouse client --host *node-group-1tXED0003* **--multiline --port 9440 --secure;**

说明

通过步骤 2 可以看到 node-group-1tXED0003 和 node-master3QsRI 的 shard_num 值相同。

show tables;

```
SHOW TABLES


┌─name─────┐
│ test     │
│ test all │
└──────────┘
```

步骤 6 查询本地表数据。例如在节点 node-group-1tXED0003 查询 test 表数据。

select * from test;

```
SELECT *
FROM test


┌─────────EventDate─┬──────id─┐
│ 2020-11-05 21:10:42 │ 1596238076 │
└─────────────────── ┴────────────┘


1 rows in set. Elapsed: 0.005 sec.
```

步骤 7 切换到不同 shard_num 的 shard 节点，并且查询之前创建的表数据信息。

例如退出之前的登录节点 node-group-1tXED0003：

exit;

切换到 node-group-1tXED0001 节点。通过步骤 2 可以看到 node-group-1tXED0001 和 node-master3QsRI 的 shard_num 值不相同。

clickhouse client --host *node-group-1tXED0001* **--multiline --port 9440 --secure;**

查询 test 本地表数据，因为 test 是本地表所以在不同分片节点上查询不到数据。

select * from test;

```
SELECT *
FROM test


Ok.
```

查询 test_all 分布式表数据，能正常查询到数据信息。

select * from test_all;

```
SELECT *
FROM test

┌─────────────EventDate─┬──────────id─┐
│ 2020-11-05 21:12:19           │ 3686805070         │
└──────────────────── ┴─────────────┘

1 rows in set. Elapsed: 0.002 sec.
```

----结束

4.4 ClickHouse 常用 SQL 语法

4.4.1 CREATE DATABASE 创建数据库

本章节主要介绍 ClickHouse 创建数据库的 SQL 基本语法和使用说明。

基本语法

CREATE DATABASE [IF NOT EXISTS] *database_name* **[ON CLUSTER** *ClickHouse 集群名***]**

说明

ON CLUSTER ClickHouse 集群名的语法，使得该 DDL 语句执行一次即可在集群中所有实例上都执行。集群名信息可以使用以下语句的 **cluster** 字段获取：

select cluster,shard_num,replica_num,host_name from system.clusters;

使用示例

```
--创建数据库名为 test 的数据库
CREATE DATABASE test ON CLUSTER default_cluster;
--创建成功后，通过查询命令验证
show databases;
┌─name───┐
│ default    │
│ system     │
│ test       │
└────────┘
```

4.4.2 CREATE TABLE 创建表

本章节主要介绍 ClickHouse 创建表的 SQL 基本语法和使用说明。

基本语法

- 方法一：在指定的“database_name”数据库中创建一个名为“table_name ”的表。

  如果建表语句中没有包含“database_name”，则默认使用客户端登录时选择的数据库作为数据库名称。

CREATE TABLE [IF NOT EXISTS] *[database_name.]table_name* **[ON CLUSTER** *ClickHouse 集群名***]**

(

name1 **[***type1***] [DEFAULT|MATERIALIZED|ALIAS** *expr1***],**

name2 **[***type2***] [DEFAULT|MATERIALIZED|ALIAS** *expr2***],**

...

) *ENGINE = engine_name()*

[**PARTITION BY** *expr_list*]

[**ORDER BY** *expr_list*]

⚠ 注意

ClickHouse 在创建表时建议携带 **PARTITION BY** 创建表分区。因为 ClickHouse 数据迁移工具是基于表的分区做数据迁移，在创建表时如果不携带 **PARTITION BY** 创建表分区，则在使用 ClickHouse 数据迁移工具界面无法对该表进行数据迁移。

- 方法二：创建一个与 database_name2.table_name2 具有相同结构的表，同时可以对其指定不同的表引擎声明。

  如果没有表引擎声明，则创建的表将与 database_name2.table_name2 使用相同的表引擎。

  CREATE TABLE [IF NOT EXISTS] *[database_name.]table_name* **AS** [*database_name2*.]*table_name2* [ENGINE = *engine_name*]
- 方法三：使用指定的引擎创建一个与 SELECT 子句的结果具有相同结构的表，并使用 SELECT 子句的结果填充它。

  CREATE TABLE [IF NOT EXISTS] *[database_name.]table_name ENGINE = engine_name* **AS SELECT** ...

使用示例

```
--在 default 数据库和 default_cluster 集群下创建名为 test 表
CREATE TABLE default.test ON CLUSTER default_cluster
(
    `EventDate` DateTime,
    `id` UInt64
)
ENGINE = ReplicatedMergeTree('/clickhouse/tables/{shard}/default/test', '{replica}')
PARTITION BY toYYYYMM(EventDate)
ORDER BY id
```

4.4.3 INSERT INTO 插入表数据

本章节主要介绍 ClickHouse 插入表数据的 SQL 基本语法和使用说明。

基本语法

- 方法一：标准格式插入数据。

INSERT INTO *[database_name.]table* [(*c1, c2, c3*)] **VALUES** (*v11, v12, v13*), (*v21, v22, v23*), ...

- 方法二：使用 SELECT 的结果写入。

  INSERT INTO *[database_name.]table* [(c1, c2, c3)] **SELECT** ...

使用示例

```
--给 test2 表插入数据
insert into test2 (id, name) values (1, 'abc'), (2, 'bbbb');
--查询 test2 表数据
select * from test2;
┌─id─┬─name─┐
│  1 │ abc  │
│  2 │ bbbb │
└────┴──────┘
```

4.4.4 SELECT 查询表数据

本章节主要介绍 ClickHouse 查询表数据的 SQL 基本语法和使用说明。

基本语法

SELECT [**DISTINCT**] expr_list

[**FROM** [*database_name.*]*table* | (subquery) | table_function] [**FINAL**]

[SAMPLE sample_coeff]

[ARRAY **JOIN** ...]

[**GLOBAL**] [**ANY**|**ALL**|**ASOF**] [**INNER**|**LEFT**|**RIGHT**|**FULL**|**CROSS**]
[**OUTER**|SEMI|ANTI] **JOIN** (subquery)|**table** (**ON** <expr_list>)|(**USING** <column_list>)

[PREWHERE expr]

[**WHERE** expr]

[**GROUP BY** expr_list] [**WITH** TOTALS]

[**HAVING** expr]

[**ORDER BY** expr_list] [**WITH** FILL] [**FROM** expr] [**TO** expr] [STEP expr]

[**LIMIT** [offset_value,]n **BY** columns]

[**LIMIT** [n,]m] [**WITH** TIES]

[**UNION ALL** ...]

[**INTO** OUTFILE filename]

[FORMAT format]

使用示例

```
--查看 ClickHouse 集群信息
select * from system.clusters;
--显示当前节点设置的宏
select * from system.macros;
```

```
--查看数据库容量
select
sum(rows) as "总行数",
formatReadableSize(sum(data_uncompressed_bytes)) as "原始大小",
formatReadableSize(sum(data_compressed_bytes)) as "压缩大小",
round(sum(data_compressed_bytes) / sum(data_uncompressed_bytes) * 100,
0) "压缩率"
from system.parts;
--查询 test 表容量。where 条件根据实际情况添加修改
select
sum(rows) as "总行数",
formatReadableSize(sum(data_uncompressed_bytes)) as "原始大小",
formatReadableSize(sum(data_compressed_bytes)) as "压缩大小",
round(sum(data_compressed_bytes) / sum(data_uncompressed_bytes) * 100,
0) "压缩率"
from system.parts
where table in ('test')
and partition like '2020-11-%'
group by table;
```

4.4.5 ALTER TABLE 修改表结构

本章节主要介绍 ClickHouse 修改表结构的 SQL 基本语法和使用说明。

基本语法

ALTER TABLE [*database_name*].*name* [**ON CLUSTER** *cluster*]
ADD|**DROP**|**CLEAR**|**COMMENT**|**MODIFY COLUMN** ...

说明

ALTER 仅支持 *MergeTree ， Merge 以及 Distributed 等引擎表。

使用示例

```
--给表 t1 增加列 test01
ALTER TABLE t1 ADD COLUMN test01 String DEFAULT 'defaultvalue';
--查询修改后的表 t1
desc t1
┌─name────┬─type─┬─default_type─┬─default_expression
┬─comment─┬─codec_expression─┬─ttl_expression─┐
│ id      │ UInt8 │              │                  │         │                │
│ name    │ String │             │                  │         │                │
│ address │ String │             │                  │         │
│ test01  │ String │ DEFAULT     │ 'defaultvalue'   │         │
└─────────┴──────┴──────────── ┴──────┴──────────┴──────────┘
--修改表 t1 列 name 类型为 UInt8
ALTER TABLE t1 MODIFY COLUMN name UInt8;
--查询修改后的表 t1
desc t1
┌─name────┬─type─┬─default_type─┬─default_expression
```

```
┬─comment─┬─codec_expression─┬─ttl_expression─┐
│ id         │ UInt8  │              │                  │          │                │
│
│ name        │ UInt8  │              │                  │          │                │
│
│ address      │ String │              │                  │          │
│              │
│ test01       │ String │ DEFAULT      │ 'defaultvalue'     │          │
│              │
└────────┴────┴────────┴──────────┴─────┴───────────┴──────────┘
--删除表 t1 的列 test01
ALTER TABLE t1 DROP COLUMN test01;
--查询修改后的表 t1
desc t1
┌─name───────┬─type───┬─default_type─┬─default_expression
┬─comment─┬─codec_expression─┬─ttl_expression─┐
│ id         │ UInt8  │              │                  │          │                │
│
│ name        │ UInt8  │              │                  │          │                │
│
│ address      │ String │              │                  │          │
│              │
└────────┴────┴────────┴──────────┴─────┴───────────┴──────────┘
```

4.4.6 DESC 查询表结构

本章节主要介绍 ClickHouse 查询表结构的 SQL 基本语法和使用说明。

基本语法

DESC|**DESCRIBE TABLE** [*database_name*.]*table* [**INTO** OUTFILE filename] [FORMAT format]

使用示例

```
--查询表 t1 的表结构
desc t1;
┌─name───────┬─type───┬─default_type─┬─default_expression
┬─comment─┬─codec_expression─┬─ttl_expression─┐
│ id         │ UInt8  │              │                  │          │                │
│
│ name        │ UInt8  │              │                  │          │                │
│
│ address      │ String │              │                  │          │
│              │
└────────┴────┴────────┴──────────┴─────┴───────────┴──────────┘
```

4.4.7 DROP 删除表

本章节主要介绍 ClickHouse 删除表的 SQL 基本语法和使用说明。

基本语法

DROP [**TEMPORARY**] **TABLE** [**IF EXISTS**] [*database_name*.]*name* [**ON CLUSTER** *cluster*]

使用示例

```
--删除表 t1
drop table t1;
```

4.4.8 SHOW 显示数据库和表信息

本章节主要介绍 ClickHouse 显示数据库和表信息的 SQL 基本语法和使用说明。

基本语法

show databases

show tables

使用示例

```
--查询数据库
show databases;
┌─name───┐
│ default     │
│ system      │
│ test        │
└────────┘
--查询表信息
show tables;
┌─name─┐
│ t1      │
│ test    │
│ test2   │
│ test5   │
└─────┘
```

4.5 ClickHouse 数据迁移

4.5.1 ClickHouse 数据导入导出

使用 ClickHouse 客户端导入导出数据

本章节主要介绍使用 ClickHouse 客户端导入导出文件数据的基本语法和使用说明。

- CSV 格式数据导入

  clickhouse client --host *主机名/ClickHouse 实例 IP 地址* **--database** *数据库名* **--port** *端口号* **--secure --format_csv_delimiter="***csv 文件数据分隔符***" --query="INSERT INTO** *数据表名* **FORMAT CSV"** < *csv 文件所在主机路径*

  使用示例：

```
clickhouse client --host 10.5.208.5 --database testdb --port 9440 --secure --
format_csv_delimiter="," --query="INSERT INTO testdb.csv_table FORMAT CSV" <
/opt/data.csv
```

数据表需提前创建好。

- CSV 格式数据导出

⚠ 注意

导出数据为 CSV 格式的文件，可能存在 CSV 注入的安全风险，请谨慎使用。

---

clickhouse client --host *主机名/ClickHouse 实例 IP 地址* **--database** *数据库名* **--port** *端口号* **-m --secure --query="SELECT * FROM** *表名*" > *csv 文件导出路径*

使用示例：

```
clickhouse client --host 10.5.208.5 --database testdb --port 9440 -m --secure -
-query="SELECT * FROM test_table" > /opt/test.csv
```

- parquet 格式数据导入

  cat *parquet 格式文件* **| clickhouse client --host** *主机名/ClickHouse 实例 IP* **--database** *数据库名* **--port** *端口号* **-m --secure --query="INSERT INTO** *表名* **FORMAT Parquet"**

  使用示例：

```
cat /opt/student.parquet | clickhouse client --host 10.5.208.5 --database
testdb --port 9440 -m --secure --query="INSERT INTO parquet tab001 FORMAT
Parquet"
```

- parquet 格式数据导出

  clickhouse client --host *主机名/ClickHouse 实例 IP* **--database** *数据库名* **--port** *端口号* **-m --secure --query="select * from** *表名* **FORMAT Parquet**" > *parquet 格式文件输出路径*

  使用示例：

```
clickhouse client --host 10.5.208.5 --database testdb --port 9440 -m --secure -
-query="select * from test_table FORMAT Parquet" > /opt/student.parquet
```

- ORC 格式数据导入

  cat *orc 格式文件路径* **| clickhouse client --host** *主机名/ClickHouse 实例 IP* **--database** *数据库名* **--port** *端口号* **-m --secure --query="INSERT INTO** *表名* **FORMAT ORC"**

  使用示例：

```
cat /opt/student.orc | clickhouse client --host 10.5.208.5 --database testdb --
port 9440 -m --secure --query="INSERT INTO orc_tab001 FORMAT ORC"
#orc 格式文件格式文件数据可以从 HDFS 中导出，例如：
hdfs dfs -cat /user/hive/warehouse/hivedb.db/emp_orc/000000_0_copy_1 |
clickhouse client --host 10.5.208.5 --database testdb --port 9440 -m --secure -
-query="INSERT INTO orc_tab001 FORMAT ORC"
```

- ORC 格式数据导出

clickhouse client --host *主机名/ClickHouse 实例 IP* **--database** *数据库名* **--port** *端口* -m **--secure --query="select * from** *表名* **FORMAT ORC**" > *输出的 ORC 格式文件路径*

使用示例：

```
clickhouse client --host 10.5.208.5 --database testdb --port 9440 -m --secure --query="select * from csv_tab001 FORMAT ORC" > /opt/student.orc
```

- JSON 格式数据导入

  INSERT INTO *表名* **FORMAT JSONEachRow** *JSON 格式字符串 1 JSON 格式字符串 2*

  使用示例：

```
INSERT INTO test_table001 FORMAT JSONEachRow {"PageViews":5,
"UserID":"4324182021466249494", "Duration":146,"Sign":-1}
{"UserID":"4324182021466249494","PageViews":6,"Duration":185,"Sign":1}
```

- JSON 格式数据导出

  clickhouse client --host *主机名/ClickHouse 实例 IP* **--database** *数据库名* **--port** *端口号* **-m --secure --query="SELECT * FROM** *表名* **FORMAT JSON|JSONEachRow|JSONCompact|...**" > *json 文件输出路径*

  使用示例

```
#导出 json
clickhouse client --host 10.5.208.5 --database testdb --port 9440 -m --secure --query="SELECT * FROM test_table FORMAT JSON" > /opt/test.json

#导出 json(JSONEachRow)
clickhouse client --host 10.5.208.5 --database testdb --port 9440 -m --secure --query="SELECT * FROM test table FORMAT JSONEachRow" >
/opt/test jsoneachrow.json

#导出 json(JSONCompact)
clickhouse client --host 10.5.208.5 --database testdb --port 9440 -m --secure --query="SELECT * FROM test table FORMAT JSONCompact" >
/opt/test_jsoncompact.json
```

4.5.2 将 Kafka 数据同步至 ClickHouse

您可以通过创建 Kafka 引擎表将 Kafka 数据自动同步至 ClickHouse 集群，具体操作详见本章节描述。

前提条件

- 已创建 Kafka 集群。已安装 Kafka 客户端。
- 已创建 ClickHouse 集群，并且 ClickHouse 集群和 Kafka 集群在同一 VPC 下，网络可以互通。已安装 ClickHouse 客户端。

约束限制

当前 ClickHouse 不支持和开启安全模式的 Kafka 集群进行对接。

Kafka 引擎表使用语法说明

- **语法**

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]
(
    name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1],
    name2 [type2] [DEFAULT|MATERIALIZED|ALIAS expr2],
    ...
) ENGINE = Kafka()
SETTINGS
    kafka_broker_list = 'host1:port1,host2:port2',
    kafka_topic_list = 'topic1,topic2,...',
    kafka_group_name = 'group_name',
    kafka_format = 'data_format';
    [kafka_row_delimiter = 'delimiter_symbol',]
    [kafka_schema = '',]
    [kafka_num_consumers = N]
```

- **参数说明**

表4-2 Kafka 引擎表参数说明

参数名	是否必选	参数说明
kafka_broker_list	是	Kafka 集群 broker 实例的 IP 和端口列表。例如：*kafka 集群 broker 实例 IP1*:**9092**,*kafka 集群 broker 实例 IP2*:**9092**,*kafka 集群 broker 实例 IP3*:**9092**。 Kafka 集群 broker 实例 IP 获取方法如下： • MRS 3.x 之前版本，单击集群名称，登录集群详情页面，选择“组件管理 > Kafka”。单击“实例”，查看 Kafka 角色实例的 IP 地址。 说明 若集群详情页面没有“组件管理”页签，请先完成 IAM 用户同步（在集群详情页的“概览”页签，单击“IAM 用户同步”右侧的“同步”进行 IAM 用户同步）。 • MRS 3.x 及后续版本，登录 FusionInsight Manager，然后选择“集群 > *待操作的集群名称* > 服务 > Kafka”。单击“实例”，查看 Kafka 角色实例的 IP 地址。
kafka_topic_list	是	Kafka 的 topic 列表。
kafka_group_name	是	Kafka 的 Consumer Group 名称，可以自己指定。
kafka_format	是	Kafka 消息体格式。例如 JSONEachRow、CSV、XML 等。
kafka_row_delimiter	否	每个消息体（记录）之间的分隔符。
kafka_sche	否	如果解析格式需要一个 schema 时，此参数必填。

参数名	是否必选	参数说明
ma		
kafka_num_consumers	否	单个表的消费者数量。默认值是：1，如果一个消费者的吞吐量不足，则指定更多的消费者。消费者的总数不应该超过 topic 中分区的数量，因为每个分区只能分配一个消费者。

Kafka 数据同步至 ClickHouse 操作示例

步骤 1 参考使用 Kafka 客户端，切换到 Kafka 客户端安装目录。

1. 以 Kafka 客户端安装用户，登录 Kafka 安装客户端的节点。
2. 执行以下命令，切换到客户端安装目录。

   cd /opt/client
3. 执行以下命令配置环境变量。

   source bigdata_env
4. 如果当前集群已启用 Kerberos 认证，执行以下命令认证当前用户。如果当前集群未启用 Kerberos 认证，则无需执行此命令。
   a. 如果是 MRS 3.1.0 版本集群，则需要先执行：**export CLICKHOUSE_SECURITY_ENABLED=true**
   b. **kinit** *组件业务用户*

步骤 2 执行以下命令，创建 Kafka 的 Topic。详细的命令使用可以参考管理 Kafka 主题。

kafka-topics.sh --topic *kafkacktest2* **--create --zookeeper** *ZooKeeper 角色实例 IP:2181*/**kafka --partitions** *2* **--replication-factor** *1*

说明

- **--topic** 参数值为要创建的 Topic 名称，本示例创建的名称为 kafkacktest2 。
- **--zookeeper**：ZooKeeper 角色实例所在节点 IP 地址，填写三个角色实例其中任意一个的 IP 地址即可。ZooKeeper 角色实例所在节点 IP 获取参考如下。
- MRS 3.x 之前版本，单击集群名称，登录集群详情页面，选择“组件管理 > ZooKeeper > 实例”。查看 ZooKeeper 角色实例的 IP 地址。
- MRS 3.x 及后续版本，登录 FusionInsight Manager，具体请参见访问 FusionInsight Manager (MRS 3.x 及之后版本)。然后选择“集群 > *待操作的集群名称* > 服务 > ZooKeeper > 实例”。查看 ZooKeeper 角色实例的 IP 地址。
- **--partitions** 主题分区数和**--replication-factor** 主题备份个数不能大于 Kafka 角色实例数量。

步骤 3 参考从零开始使用 ClickHouse 登录 ClickHouse 客户端。

1. 执行以下命令，切换到客户端安装目录。

   cd /opt/Bigdata/client
2. 执行以下命令配置环境变量。

source bigdata_env

3. 如果当前集群已启用 Kerberos 认证，执行以下命令认证当前用户，当前用户需要具有创建 ClickHouse 表的权限，具体请参见 ClickHouse 用户及权限管理章节，为用户绑定对应角色。如果当前集群未启用 Kerberos 认证，则无需执行本步骤。

   kinit *组件业务用户*

   例如，**kinit** clickhouseuser。

4. 执行以下命令连接到要导入数据的 ClickHouse 实例节点。

   clickhouse client --host *ClickHouse 的实例 IP* **--user** *登录名* **--password** *密码* **--port** *ClickHouse 的端口号* **--database** *数据库名* **--multiline**

步骤 4 参考 Kafka 引擎表使用语法说明，在 ClickHouse 中创建 Kafka 引擎表。例如，如下建表语句在 default 数据库下，创建表名为 kafka_src_tbl3，Topic 名为 kafkacktest2、消息格式为 JSONEachRow 的 Kafka 引擎表。

```
create table kafka_src_tbl3 on cluster default_cluster
(id UInt32, age UInt32, msg String)
ENGINE=Kafka()
SETTINGS
 kafka broker list='kafka 集群 broker 实例 IP1:9092,kafka 集群 broker 实例 IP2:9092,kafka 集群 broker 实例 IP3:9092',
 kafka topic list='kafkacktest2',
 kafka group name='cg12',
 kafka_format='JSONEachRow';
```

步骤 5 创建 ClickHouse 本地复制表。例如，如下创建表名为 kafka_dest_tbl3 的 ReplicatedMergeTree 表。

```
create table kafka_dest_tbl3 on cluster default_cluster
( id UInt32, age UInt32, msg String )
engine = ReplicatedMergeTree('/clickhouse/tables/{shard}/default/kafka_dest_tbl3',
'{replica}')
partition by age
order by id;
```

步骤 6 创建 MATERIALIZED VIEW，该视图会在后台转换 Kafka 引擎中的数据并将其放入创建的 ClickHouse 表中。

```
create materialized view consumer3 on cluster default_cluster to kafka_dest_tbl3 as
select * from kafka_src_tbl3;
```

步骤 7 再次执行步骤 1，进入 Kafka 客户端安装目录。

步骤 8 执行以下命令，在 Kafka 的 Topic 中产生消息。例如，如下命令向步骤 2 中创建的 Topic 发送消息。

kafka-console-producer.sh --broker-list *kafka 集群 broker 实例 IP1***:9092,***kafka 集群 broker 实例 IP2***:9092,***kafka 集群 broker 实例 IP3***:9092 --topic** *kafkacktest2*

```
>{"id":31, "age":30, "msg":"31 years old"}
>{"id":32, "age":30, "msg":"31 years old"}
>{"id":33, "age":30, "msg":"31 years old"}
>{"id":35, "age":30, "msg":"31 years old"}
```

步骤 9 使用 ClickHouse 客户端登录步骤 3 中 ClickHouse 实例节点，查询 ClickHouse 表数据。例如，查询 kafka_dest_tbl3 本地复制表，Kafka 消息中的数据已经同步到该表。

```
select * from kafka_dest_tbl3;
```

```
ClickHouseKXya.mrs-2xxk.com :) select * from kafka_dest_tbl3;

SELECT *
FROM kafka_dest_tbl3

Query id: 386ed9db-26b6-4823-8ce5-5b2de14384bb

┌─id─┬─age─┬─msg──────────┐
│ 31 │  30 │ 31 years old │
└────┴─────┴──────────────┘
┌─id─┬─age─┬─msg──────────┐
│ 32 │  30 │ 31 years old │
└────┴─────┴──────────────┘
┌─id─┬─age─┬─msg──────────┐
│ 33 │  30 │ 31 years old │
└────┴─────┴──────────────┘
┌─id─┬─age─┬─msg──────────┐
│ 35 │  30 │ 31 years old │
└────┴─────┴──────────────┘

4 rows in set. Elapsed: 0.003 sec.
```

----结束

4.5.3 使用 ClickHouse 数据迁移工具

ClickHouse 数据迁移工具可以将某几个 ClickHouseServer 实例节点上的一个或多个 MergeTree 引擎分区表的部分分区迁移至其他 ClickHouseServer 节点上相同的表中。在扩容场景中，可以使用该工具将原节点上的部分数据迁移至新增节点上，从而达到扩容后的数据均衡。

前提条件

- ClickHouse 服务运行正常，Zookeeper 服务运行正常，迁入、迁出节点的 ClickHouseServer 实例状态正常。
- 请确保迁入节点已有待迁移数据表，且确保该表是 MergeTree 系列引擎的分区表。
- 创建迁移任务前请确保所有对待迁移数据表的写入任务已停止，且任务启动后，只允许对待迁移数据表进行查询操作，禁止对该表进行写入、删除等操作，否则可能会造成迁移前后数据不一致。
- 迁入节点的 ClickHouse 数据目录有足够的空间。

操作步骤

步骤 1 登录 Manager，选择“集群 > 服务 > ClickHouse”， 在 ClickHouse 服务界面单击“数据迁移”页签，进入数据迁移界面。

步骤 2 单击“创建迁移任务”。

步骤 3 在创建迁移任务界面，填写迁移任务的相关参数，具体参考如下表 4-3。

表4-3 迁移任务参数说明

参数名	参数取值说明
任务名称	填写具体的任务名称。可由字母、数组及下划线组成，长度为 1~50 位，且不能与已有的迁移任务相同。

参数名	参数取值说明
任务类型	• 定时任务：选择定时任务时，可以设置“开始时间”参数，设定任务在当前时间以后的某个时间点执行。 • 即时任务：任务启动后立即开始执行。
开始时间	在“任务类型”参数选择“定时任务”时填写，有效值为当前时间以后的某个时间（最长为 90 天以后）。

步骤 4 在选择迁移节点界面，填写“迁入节点主机名”、“迁出节点主机名”，单击“下一步”。

说明

- “迁入节点主机名”与“迁出节点主机名”只能各填写一个主机名，不支持多节点迁移。

  具体的参数值可以在 ClickHouse 服务界面单击“实例”页签，查看当前 ClickHouseServer 实例所在“主机名称”列获取。

- “带宽上限”为可选参数，若不填写则为无上限，最大可设置为 10000MB/s。

步骤 5 在选择迁移数据表界面，单击“数据库”后的 ，选择待迁出节点上存在的数据库，在“数据表”处选择待迁移的数据表，数据表下拉列表中展示的是所选数据库中的 MergeTree 系列引擎的分区表。“节点信息”中展示的为当前迁入节点、迁出节点上 ClickHouse 服务数据目录的空间使用情况，单击“下一步”。

步骤 6 确认任务信息，确认无误后可以单击“提交”提交任务。

数据迁移工具将根据待迁移数据表的大小自动计算需要迁移的分区，数据迁移量则是计算出的需要迁移的分区总大小。

步骤 7 提交迁移任务成功后，单击操作列的“启动”。如果任务类型是即时任务则开始执行任务，如果是定时任务则开始倒计时。

步骤 8 迁移任务执行过程中，可单击“取消”取消正在执行的迁移任务，若取消任务，则会回退掉迁入节点上已迁移的数据。

可以单击“更多 > 详情”查看迁移过程中的日志信息。

步骤 9 迁移完成后，选择“更多 > 结果”查看迁移结果；选择“更多 > 删除”清理 ZooKeeper 以及迁出节点上该迁移任务相关的目录。

----结束

4.6 用户管理及认证

4.6.1 ClickHouse 用户及权限管理

用户权限模型

ClickHouse 用户权限管理实现了对集群中各个 ClickHouse 实例上用户、角色、权限的统一管理。通过 Manager UI 的权限管理模块进行创建用户、创建角色、绑定 ClickHouse 访问权限配置等操作，通过用户绑定角色的方式，实现用户权限控制。

管理资源：Clickhouse 权限管理支持的资源如表 4-4 所示。

资源权限：ClickHouse 支持的资源权限如表 4-5 所示。

表4-4 ClickHouse 支持的权限管理对象

资源列表	是否集成	备注
数据库	是（一级）	-
表	是（二级）	-
视图	是（二级）	与表一致

表4-5 资源权限列表

资源对象	可选权限	备注
数据库（DATABASE）	CREATE	CREATE DATABASE/TABLE/VIEW/DICTIONARY 权限
表/视图（TABLE/VIEW）	SELECT/INSERT	-

前提条件

- ClickHouse 服务运行正常，Zookeeper 服务运行正常。
- 用户在集群中创建数据库或者表时使用 ON CLUSTER 语句，保证各个 ClickHouse 节点上数据库、表的元信息相同。

说明

ClickHouse 赋权成功后，权限生效时间大约为 1 分钟。

添加 ClickHouse 角色

步骤 1 登录 Manager，选择“系统 > 权限 > 角色”，在“角色”界面单击“添加角色”按钮，进入添加角色页面。

步骤 2 在添加角色界面输入“角色名称”，在配置资源权限处单击集群名称，进入服务列表页面，单击 ClickHouse 服务，进入 ClickHouse 权限资源页面。

根据业务需求确定是否要创建具有管理员权限的角色。

说明

- ClickHouse 管理员权限为：除去对 user/role 的创建、删除和修改之外的所有数据库操作权限。
- 对于用户和角色的管理，仅有 ClickHouse 的内置用户 **clickhouse** 具有权限。

- 是，执行步骤 3。
- 否，执行步骤 4。

步骤 3 勾选“ClickHouse 管理员权限”，单击“确定”操作结束。

步骤 4 单击“ClickHouse Scope”，进入 ClickHouse 数据库资源列表。勾选“创建”权限，则该角色将拥有该数据库下的创建（CREATE）权限。

根据业务需求确定是否赋权。

- 是，单击“确定”操作结束。
- 否，执行步骤 5。

步骤 5 单击“资源名称 > *待操作的数据库资源名称*”，进入表、视图页面，根据业务需要，勾选“读”（SELECT 权限）或者“写”（INSERT 权限）权限，单击“确定”。

----结束

添加用户并将 ClickHouse 对应角色绑定到该用户

步骤 1 登录 Manager，选择“系统 > 权限 > 用户”，单击“添加用户”，进入添加用户页面。

步骤 2 “用户类型”选择“人机”，在“密码”和“确认密码”参数设置该用户对应的密码。

说明

- 用户名：添加的用户名不能包含字符“-”，否则会导致认证失败。
- 密码：设置的密码不能携带“$”、“.”、“#”特殊字符，否则会导致认证失败。

步骤 3 在“角色”处单击“添加”，在弹框中选择具有 ClickHouse 权限的角色，单击“确定”添加到角色，单击“确定”完成操作。

步骤 4 登录 ClickHouse 客户端安装节点，使用新添加的用户及设置的密码连接 ClickHouse 服务。

- 执行以下命令，切换到客户端安装目录。

  cd /opt/*客户端安装目录*

- 执行以下命令配置环境变量。

  source bigdata_env

- 如果当前集群已启用 Kerberos 认证，执行以下命令认证当前用户，当前用户需要具有创建 ClickHouse 表的权限，具体请参见添加 ClickHouse 角色，为用户绑定对应角色。如果当前集群未启用 Kerberos 认证，则无需执行本步骤。
  a. 如果是 MRS 3.1.0 版本集群，则需要先执行：**export CLICKHOUSE_SECURITY_ENABLED=true**
  b. **kinit** *组件业务用户*
- 使用新添加的用户登录验证。

  clickhouse client --host *ClickHouse 的实例 IP* **--multiline --user** ***步骤 1*** *中添加的用户* **--password** *步骤 2 中设置的用户密码* **--port** *ClickHouse 的端口号* **--secure**

----结束

异常场景下登录客户端操作赋权

ClickHouse 集群默认每个节点上的表元信息是相同的，因此在 Manager 的权限管理页面上默认采集的是任意 ClickHouse 节点的表信息，如果有个别节点上创建 DATABASE/TABLE 时未使用 ON CLUSTER 语句，则权限操作可能无法展示该资源，不保证可以对其赋权。对于这样单个 ClickHouse 节点中的本地表，如果需要赋权，则可以通过后台客户端进行操作。

说明

以下操作，需要提前获取到需要赋权的角色、数据库或表名称、对应的 ClickHouseServer 实例所在的节点 IP。

- ClickHouseServer 的实例 IP 地址可登录集群 FusionInsight Manager，然后选择“集群 > 服务 > ClickHouse > 实例”，获取 ClickHouseServer 实例对应的业务 IP 地址。
- 系统域名：默认为 hadoop.com。可登录集群 FusionInsight Manager，单击“系统 > 权限 > 域和互信”，“本端域”参数值即为系统域名。在执行命令时改为小写。

步骤 1 以 root 用户登录 ClickHouseServer 实例所在的节点。

步骤 2 执行以下命令获取“clickhouse.keytab”文件路径。

ls ${BIGDATA_HOME}/FusionInsight_ClickHouse_*/install/FusionInsight-ClickHouse-*/clickhouse/keytab/clickhouse.keytab

步骤 3 以客户端安装用户，登录安装客户端的节点。

步骤 4 执行以下命令，切换到客户端安装目录。

cd /opt/client

步骤 5 执行以下命令配置环境变量。

source bigdata_env

如果是 MRS 3.1.0 版本集群并且集群已启用 Kerberos 认证，则还需要执行：

export CLICKHOUSE_SECURITY_ENABLED=true

步骤 6 执行如下命令使用客户端命令连接 ClickHouseServer 实例。

如果当前集群已启用 Kerberos 认证，执行以下命令：

clickhouse client --host *ClickHouseServer 实例所在节点 IP* **--user clickhouse/hadoop.**<*系统域名*> **--password** *步骤 2 中获取的 clickhouse.keytab 路径* **--port** *ClickHouse 的端口号* **--secure**

如果当前集群未启用 Kerberos 认证，执行以下命令：

clickhouse client --host *ClickHouseServer 实例所在节点 IP* **--user clickhouse --port** *ClickHouse 的端口号*

步骤 7 对某 DATABASE 进行赋权操作，执行如下命令。

授权操作语法，其中 *DATABASE* 为要操作的数据库名称，*role* 为需要操作的角色。

GRANT [ON CLUSTER *cluster_name***]** *privilege* **ON** *{DATABASE|TABLE}* **TO** *{user | role]*

例如，给用户 testuser 授予数据库 t2 的 CREATE 权限：

GRANT CREATE ON *m2* **to** *testuser***;**

步骤 8 对 TABLE/VIEW 进行赋权操作，执行如下命令，其中 *TABLE* 为要操作的表或视图名称，*user* 为需要操作的角色。

对某数据库下的表赋予查询权限：

GRANT SELECT ON *TABLE* **TO** *user;*

对某数据库下的表赋予写入权限：

GRANT INSERT ON *TABLE* **TO** *user*；

说明

更多 ClickHouse 授权操作及详细权限说明可参考 https://clickhouse.tech/docs/zh/sql-reference/statements/grant/。

步骤 9 执行如下命令，退出客户端。

quit;

----结束

4.6.2 ClickHouse 使用 OpenLDAP 认证

ClickHouse 支持和 OpenLDAP 进行对接，通过在 ClickHouse 上添加 OpenLDAP 服务器配置和创建用户，实现帐号和权限的统一集中管理和权限控制等操作。此方案适合从 OpenLDAP 服务器中批量向 ClickHouse 中导入用户。

本章节操作仅支持 MRS 3.1.0 及以上集群版本。

前提条件

- MRS 集群及 ClickHouse 实例运行正常，已安装 ClickHouse 客户端。
- OpenLDAP 已安装且状态正常。

对接 OpenLDAP 服务器创建 ClickHouse 用户

步骤 1 登录集群 Manager 页面，选择“集群 > 服务 > ClickHouse > 配置 > 全部配置”。

步骤 2 参考下图图 4-2，选择“ClickHouseServer（角色）> 自定义”，在“clickhouse-config-customize”配置项中添加如下 OpenLDAP 配置参数。

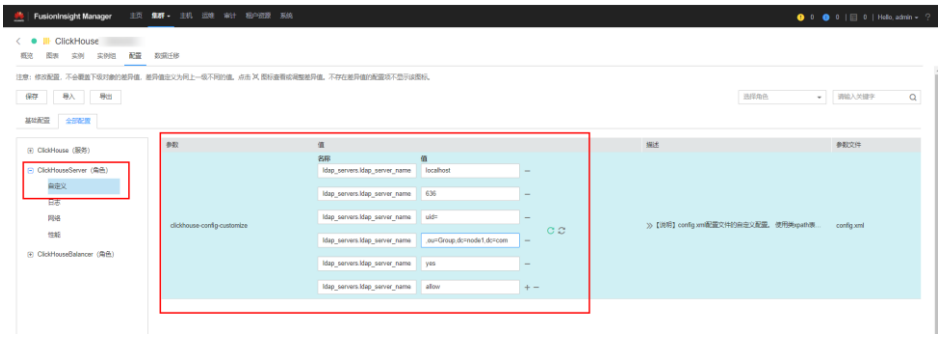
表4-6 OpenLDAP 参数说明

参数名	参数值说明	参数取值参考
ldap_servers.ldap_server_name.host	OpenLDAP 服务器主机名或 IP，不能为空。	localhost
ldap_servers.ldap_server_name.port	OpenLDAP 服务器端口。 如果 enable_tls 参数设置为 true，则默认端口号为 636，否则为 389。	636
ldap_servers.ldap_server_name.auth_dn_prefix	用于构造要绑定到的 DN 的前缀和后缀。 生成的 DN 将被构造为 auth_dn_prefix + escape(user_name) + auth_dn_suffix 字符串。 auth_dn_suffix 通常应将逗号“,”作为其第一个非空格字符。	uid=
ldap_servers.ldap_server_name.auth_dn_suffix		,ou=Group,dc=node1,dc=com
ldap_servers.ldap_server_name.enable_tls	触发使用 OpenLDAP 服务器安全连接的标志。 • 纯文本（ldap://）协议指定“no”（不推荐）。 • LDAP over SSL/TLS (ldaps://) 协议指定“yes”。	yes
ldap_servers.ldap_server_name.tls_require_cert	SSL/TLS 对端证书校验行为。 取值范围为： 'never'、'allow'、'try'、'require'。	allow

说明

其他参数说明详细可以参考<ldap_servers>配置参数详解。

图4-2 OpenLDAP 配置



步骤 3 添加完配置后，单击“保存”，在弹出对话框中单击“确定”，配置保存成功后，单击“完成”。

步骤 4 Manager 页面，单击“实例”，选择 ClickHouseServer 实例，单击“更多 > 重启实例 ”，弹出对话框输入密码，单击“确定”。重启实例对话框，单击“确定”，根据界面提示信息确认实例重启成功，单击“完成”重启操作完成。

步骤 5 登录 ClickHouseServer 实例所在主机节点，进入“${BIGDATA_HOME}/FusionInsight_ClickHouse_*版本号*/*x_x*_ClickHouseServer/etc”目录。

cd ${BIGDATA_HOME}/FusionInsight_ClickHouse_*/*x_x*_ClickHouseServer/etc

步骤 6 执行以下命令，查看配置文件 config.xml，确认 OpenLDAP 参数是否配置成功。

cat config.xml

```
[root@k                    3 etc]# cat config.xml
<yandex>
  <ldap_servers>
    <ldap_server_name>
      <auth_dn_prefix>uid=</auth_dn_prefix>
      <port>636</port>
      <host>localhost</host>
      <enable_tls>yes</enable_tls>
      <tls_require_cert>allow</tls_require_cert>
      <auth_dn_suffix>,ou=Group,dc=node1,dc=com</auth_dn_suffix>
    </ldap_server_name>
  </ldap_servers>
  <zookeeper>
```

步骤 7 以 root 用户登录 ClickHouseServer 实例所在的节点。

步骤 8 执行以下命令获取“clickhouse.keytab”文件路径。

ls ${BIGDATA_HOME}/FusionInsight_ClickHouse_*/install/FusionInsight-ClickHouse-*/clickhouse/keytab/clickhouse.keytab

步骤 9 以客户端安装用户，登录安装客户端的节点。

步骤 10 执行以下命令，切换到 ClickHouse 客户端安装目录。

cd /opt/client

步骤 11 执行以下命令配置环境变量。

source bigdata_env

步骤 12 执行如下命令使用客户端命令连接 ClickHouseServer 实例。

- 如果当前集群已启用 Kerberos 认证，使用 clickhouse.keytab 连接 ClickHouseServer 实例：

  clickhouse client --host *ClickHouseServer 实例所在节点 IP* **--user clickhouse/hadoop.**<*系统域名*> **--password** *步骤 8 中获取的 clickhouse.keytab 路径* **--port** *ClickHouse 的端口号*

  说明

  系统域名：默认为 hadoop.com。具体可登录集群 FusionInsight Manager，单击“系统 > 权限 > 域和互信”，“本端域”参数值即为系统域名。在执行命令时改为小写。

- 如果当前集群未启用 Kerberos 认证，使用 clickhouse 管理员用户连接 ClickHouseServer 实例：

  clickhouse client --host *ClickHouseServer 实例所在节点 IP* **--user clickhouse --port** *ClickHouse 的端口号*

步骤 13 创建 OpenLDAP 中的普通用户。

如以下语句，在集群 default_cluster 上创建 testUser 用户，设置 ldap_server 为步骤 6 中<ldap_servers>标签下的 OpenLDAP 服务名，本示例为 ldap_server_name。

CREATE USER *testUser* **ON CLUSTER** *default_cluster* **IDENTIFIED WITH ldap_server BY** 'ldap_server_name'**;**

testUser 用户为 OpenLDAP 中已有的用户名，请根据实际情况修改。

步骤 14 退出客户端，使用新建的用户登录验证配置是否成功。

exit;

clickhouse client --host *ClickHouseServer 实例 IP* **--user** *testUser* **--password** *testUser 对应的密码* **--port** *ClickHouse 的端口号*

----结束

<ldap_servers>配置参数详解

- host

  OpenLDAP 服务器主机名或 IP，必选参数，不能为空。

- port

  OpenLDAP 服务器端口，如果 enable_tls 参数设置为 true，则默认为 636，否则为 389。

- auth_dn_prefix，auth_dn_suffix

  用于构造要绑定到的 DN 的前缀和后缀。

  实际上，生成的 DN 将被构造为 auth_dn_prefix + escape(user_name) + auth_dn_suffix 字符串。

  注意，这意味着 auth_dn_suffix 通常应将逗号“,”作为其第一个非空格字符。

- enable_tls

  触发使用 OpenLDAP 服务器安全连接的标志。

为纯文本（ldap://）协议指定“no”（不推荐）。

为 LDAP over SSL/TLS (ldaps://)协议指定“yes”（建议为默认值）。

- tls_minimum_protocol_version

  SSL/TLS 的最小协议版本。

  接受的值是： 'ssl2'、'ssl3'、'tls1.0'、'tls1.1'、'tls1.2'（默认值）。
- tls_require_cert

  SSL/TLS 对端证书校验行为。

  接受的值是： 'never'、'allow'、'try'、'require'（默认值）。
- tls_cert_file

  证书文件。
- tls_key_file

  证书密钥文件。
- tls_ca_cert_file

  CA 证书文件。
- tls_ca_cert_dir

  CA 证书所在的目录。
- tls_cipher_suite

  允许加密套件。

4.7 ClickHouse 慢查询语句和复制表数据同步指标监控

4.7.1 慢查询语句监控

操作场景

在 ClickHouse 上执行 SQL 语句查询时，常因为 SQL 语句的分区、where 条件以及索引等设置不合理问题，导致 SQL 查询很慢，影响数据库的整体性能。针对该场景，MRS 提供了 ClickHouse 慢查询语句的监控功能。

该功能仅支持 MRS 3.1.2 及以后版本集群。

正在进行的慢查询

当前还在执行没有返回结果的慢 SQL 语句信息可以通过该界面查询。

- **慢查询菜单路径**

  登录 Manager，选择“集群 > 服务 > ClickHouse > 查询管理”，单击“正在进行的慢查询”页签。

- **慢查询参数说明**

表4-7 慢查询参数说明

参数	参数说明
Server 节点 IP	ClickHouseServer 实例的 IP。具体可以到 Manager，选择“集群 > 服务 > ClickHouse > 实例”，ClickHouseServer 角色的 IP。
查询 id	内部生成的唯一 ID。
查询语句	具体慢查询的 SQL 语句。
开始时间	慢查询的 SQL 语句的执行开始时间。
结束时间	慢查询的 SQL 语句的执行结束时间。
查询时长 （s）	慢查询的 SQL 语句当前累计执行的时间，单位是秒。
用户	执行慢查询的 SQL 语句的 ClickHouse 用户。
客户端 IP	提交该慢查询 SQL 语句的客户端 IP。
占用的内存空间（MB）	慢查询 SQL 语句占用的内存大小统计，单位是 MB。
操作	当前查询出来的慢 SQL 语句，可以单击“终止”结束该慢 SQL 语句查询。

- **慢查询过滤条件**

  选择对应的过滤条件，输入查询条件值进行过滤查询。

表4-8 慢查询界面过滤条件

条件	参数说明
慢查询运行时长大于	按照慢 SQL 查询语句查询累计时长过滤查询。 支持时长大于：3(s)、9(s)、15(s)、25(s)
按查询 id	根据查询界面对应慢查询语句的“查询 id”字段过滤查询。 支持按照“查询 id”的部分值进行模糊查询，例如，查询 ID 为“111-222-333-444-555”，则输入“111-222”或“-222-333”也能查询到。
按用户查询	对应执行慢 SQL 的 ClickHouse 用户。 支持按照用户名的部分值进行模糊查询，例如，用户为“cktest123”，则输入“cktest”或“cktest12”也能查询到。
按客户端 IP 查询	对应慢查询 SQL 语句的客户端 IP。 支持按照客户端 IP 的部分值进行模糊查询，例如，客户端 IP 为“172.1.0.1”，则输入“172.1”或“172.1.0”也能查询到。

已经结束的查询

已经执行完成并且已返回结果的慢 SQL 语句信息可以通过该界面查询。

界面访问路径：登录 Manager，选择“集群 > 服务 > ClickHouse > 查询管理”，单击“已经结束的查询”页签。

界面的参数说明参考表 4-7，过滤条件说明参考表 4-8 说明。

4.7.2 复制表数据同步监控

操作场景

Replicated*MergeTree 系列引擎表同分片下的多个副本数据相互进行同步，MRS 针对该场景下的表数据同步进行了状态监控。

该功能仅支持 MRS 3.1.2 及以后版本集群。

约束限制

当前只支持 Replicated*MergeTree 系列引擎表并且建表语句携带 **ON CLUSTER** 关键字的表监控查询。

复制表数据同步

- **数据同步菜单路径**

  登录 Manager，选择“集群 > 服务 > ClickHouse > 数据同步状态”。

  图4-3 数据同步状态查询

- **数据同步参数说明**

表4-9 数据状态同步参数说明

参数	参数说明
数据表	Replicated*MergeTree 系列引擎表表名。
所属数据库	数据表所在的数据库。
分片信息	数据表所在的 ClickHouse 分片。
同步状态	分为以下几种状态。 • 无数据：当前分片节点上该表没有数据。 • 已同步：当前分片节点上该表有数据，并且分片下多个副本实例间的数据一致。 • 未同步：当前分片节点上该表有数据，当分片下多个副本实例间的表数据不一致。
详情	数据表在对应 ClickHouseServer 实例上的表数据同步详情。

- **过滤条件**

  选择“按数据表查询”，搜索框输入对应的数据表表名进行过滤查询。

4.8 通过数据文件备份恢复 ClickHouse 数据

操作场景

本章节主要介绍通过把 ClickHouse 中的表数据导出到 CSV 文件进行备份，后续可以通过备份的 CSV 文件数据再进行恢复操作。

前提条件

- 已安装 ClickHouse 客户端。
- 在 Manager 已创建具有 ClickHouse 相关表权限的用户。
- 已准备好备份服务器。

备份数据

步骤 1 以客户端安装用户，登录安装客户端的节点。

步骤 2 执行以下命令，切换到客户端安装目录。

cd /opt/client

步骤 3 执行以下命令配置环境变量。

source bigdata_env

步骤 4 如果当前集群已启用 Kerberos 认证，执行以下命令认证当前用户，当前用户需要具有创建 ClickHouse 表的权限。如果当前集群未启用 Kerberos 认证，则无需执行本步骤。

1. 如果是 MRS 3.1.0 版本集群，则需要先执行：

   export CLICKHOUSE_SECURITY_ENABLED=true

2. **kinit** *组件业务用户*

   例如，**kinit** clickhouseuser。

步骤 5 执行 ClickHouse 组件的客户端命令，将要备份 ClickHouse 表数据导出到指定目录下。

clickhouse client --host *主机名/实例 IP* **--secure --port 9440 --query="***表查询语句***" >** *输出的 csv 格式文件路径*

例如，如下是在 ClickHouse 实例 10.244.225.167 下备份 test 表数据到 default_test.csv 文件中。

clickhouse client --host 10.244.225.167 --secure --port 9440 --query="select * from default.test FORMAT CSV" > /opt/clickhouse/default_test.csv

步骤 6 将导出的 csv 数据文件上传至备份服务器。

----结束

恢复数据

步骤 1 将备份服务器上的备份数据文件上传到 ClickHouse 客户端所在目录。

例如，上传 default_test.csv 备份文件到：/opt/clickhouse 目录下。

步骤 2 以客户端安装用户，登录安装客户端的节点。

步骤 3 执行以下命令，切换到客户端安装目录。

cd /opt/client

步骤 4 执行以下命令配置环境变量。

source bigdata_env

步骤 5 如果当前集群已启用 Kerberos 认证，执行以下命令认证当前用户，当前用户需要具有创建 ClickHouse 表的权限。如果当前集群未启用 Kerberos 认证，则无需执行本步骤。

1. 如果是 MRS 3.1.0 版本集群，则需要先执行：

   export CLICKHOUSE_SECURITY_ENABLED=true

2. **kinit** *组件业务用户*

   例如，**kinit** clickhouseuser。

步骤 6 执行 ClickHouse 组件的客户端命令，登录 ClickHouse 集群。

clickhouse client --host *主机名/实例 IP* **--secure --port 9440**

步骤 7 创建与 CSV 备份数据文件格式对应的表。

CREATE TABLE [IF NOT EXISTS] *[database_name.]table_name* **[ON CLUSTER** *Cluster 名***]**

(

name1 **[***type1***] [DEFAULT|***materialized***|ALIAS** *expr1***],**

name2 **[***type2***] [DEFAULT|***materialized***|ALIAS** *expr2***],**

...

) *ENGINE = engine*

步骤 8 将备份数据文件中的内容导入到步骤 7 创建的表中进行数据恢复。

clickhouse client --host *主机名/实例 IP* **--secure --port 9440 --query="insert into** *表信息* **FORMAT CSV" <** *csv 文件路径*

例如，如下在 ClickHouse 实例 10.244.225.167 中，恢复 default_test.csv 备份文件数据到 test_cpy 表中。

clickhouse client --host 10.244.225.167 --secure --port 9440 --query="insert into default.test_cpy FORMAT CSV" < /opt/clickhouse/default_test.csv

----结束

4.9 ClickHouse 日志介绍

日志描述

日志路径：ClickHouse 相关日志的默认存储路径为“${BIGDATA_LOG_HOME}/clickhouse”。

日志归档规则：ClickHouse 日志启动了自动压缩归档功能，缺省情况下，当日志大小超过 100MB 的时候，会自动压缩，压缩后的日志文件名规则为：“<原有日志名>.[编号].gz”。默认最多保留最近的 10 个压缩文件，压缩文件保留个数可以在 Manager 界面中配置。

表4-10 ClickHouse 日志列表

日志类型	日志文件名	描述
运行日志	/var/log/Bigdata/clickhouse/clickhouseServer/clickhouse-server.err.log	ClickHouseServer 服务运行错误日志文件路径。
	/var/log/Bigdata/clickhouse/clickhouseServer/checkService.log	ClickHouseServer 服务运行关键日志文件路径。
	/var/log/Bigdata/clickhouse/clickhouseServer/clickhouse-server.log	
	/var/log/Bigdata/clickhouse/balance/start.log	ClickHouseBalancer 服务启动日志文件路径。
	/var/log/Bigdata/clickhouse/balance/error.log	ClickHouseBalancer 服务运行错误日志文件路径。
	/var/log/Bigdata/clickhouse/balance/access_http.log	ClickHouseBalancer 服务运行日志文件路径。
数据迁移日志	/var/log/Bigdata/clickhouse/migration/数据迁移任务名/clickhouse-copier_{timestamp}_{processId}/copier.log	参考使用 ClickHouse 数据迁移工具使用迁移工具时产生的运行日志。
	/var/log/Bigdata/clickhouse/migration/数据迁移任务名/clickhouse-copier_{timestamp}_{processId}/copier.err.log	参考使用 ClickHouse 数据迁移工具使用迁移工具时产生的错误日志。

日志级别

ClickHouse 提供了如表 4-11 所示的日志级别。

运行日志的级别优先级从高到低分别是 error、warning、trace、information、debug，程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。

表4-11 日志级别

日志类型	级别	描述
运行日志	error	error 表示系统运行的错误信息。
	warning	warning 表示当前事件处理存在异常信息。

日志类型	级别	描述
	trace	trace 表示当前事件处理跟踪信息。
	information	information 表示记录系统及各事件正常运行状态信息。
	debug	debug 表示记录系统及系统的调试信息。

如果您需要修改日志级别，请执行如下操作：

步骤 1 登录 FusionInsight Manager 系统。

步骤 2 选择“集群 > 服务 > ClickHouse > 配置”。

步骤 3 单击“全部配置”。

步骤 4 左边菜单栏中选择所需修改的角色所对应的日志菜单。

步骤 5 选择所需修改的日志级别。

步骤 6 单击“保存”，然后单击“确定”，成功后配置生效。

----结束

说明

配置完成后即生效，不需要重启服务。

日志格式

ClickHouse 的日志格式如下所示：

表4-12 日志格式

日志类型	格式	示例
运行日志	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<产生该日志的线程名字>|<log 中的 message>|<日志事件的发生位置>	2021.02.23 15:26:30.691301 [6085] {} <Error> DynamicQueryHandler: Code: 516, e.displayText() = DB::Exception: default: Authentication failed: password is incorrect or there is no user with such name, Stack trace (when copying this message, always include the lines below):

日志类型	格式	示例
		0. Poco::Exception::Exception(std::__1::basic_string<char, std::__1::char_traits<char>, std::__1::allocator<char> > const&, int) @ 0x1250e59c

5 使用 DBService

5.1 DBService 日志介绍

日志描述

日志存储路径：DBService 相关日志的默认存储路径为“/var/log/Bigdata/dbservice”。

- gaussDB：“/var/log/Bigdata/dbservice/DB”（gaussDB 运行日志目录），
  “/var/log/Bigdata/dbservice/scriptlog/gaussdbinstall.log”（gaussDB 安装日志），
  “/var/log/gaussdbuninstall.log”（gaussDB 卸载日志）。
- HA：“/var/log/Bigdata/dbservice/ha/runlog”（HA 运行日志目录），
  “/var/log/Bigdata/dbservice/ha/scriptlog”（HA 脚本日志目录）。
- DBServer：“/var/log/Bigdata/dbservice/healthCheck”（服务进程健康状态检查日志目录）。

  “/var/log/Bigdata/dbservice/scriptlog”（运行日志目录），
  “/var/log/Bigdata/audit/dbservice/”（审计日志目录）。

日志归档规则：DBService 的日志启动了自动压缩归档功能，缺省情况下，当日志大小超过 1MB 的时候，会自动压缩，压缩后的日志文件名规则为：“*<原有日志名>*_*<编号>*.gz”。最多保留最近的 20 个压缩文件。

说明

日志归档规则用户不能修改。

表5-1 DBService 日志列表

日志类型	日志文件名	描述
DBServer 运行相关日志	dbservice_serviceCheck.log	服务检查脚本运行日志
	dbservice_processCheck.log	进程检查脚本运行日志
	backup.log	备份恢复操作运行日志（需执行 DBService 备份恢复操作）

日志类型	日志文件名	描述
	checkHaStatus.log	HA 检查日志
	cleanupDBService.log	卸载日志（需执行 DBService 卸载日志操作）
	componentUserManager.log	数据库用户添加删除操作日志 （需添加依赖 DBService 的服务）
	install.log	安装日志
	preStartDBService.log	预启动日志
	start_dbserver.log	DBServer 启动操作日志（需执行启动 DBService 服务的操作）
	stop_dbserver.log	DBServer 停止操作日志（需执行停止 DBService 服务的操作）
	status_dbserver.log	DBServer 状态检查日志（需执行 **$DBSERVICE_HOME/sbin/status-dbserver.sh**）
	modifyPassword.log	DBService 修改密码脚本运行日志（需执行 **$DBSERVICE_HOME/sbin/modifyDBPwd.sh**）
	modifyDBPwd_yyyy-mm-dd.log	修改密码工具运行日志 （需执行 **$DBSERVICE_HOME/sbin/modifyDBPwd.sh**）
	dbserver_switchover.log	DBServer 执行主备倒换脚本的日志（需执行主备倒换操作）
GAUSSDB 运行日志	gaussdb.log	记录数据库运行信息
	gs_ctl-current.log	记录 gs_ctl 工具的操作
	gs_guc-current.log	记录 gs_guc 工具的操作，主要是参数修改
	gaussdbinstall.log	gaussDB 安装日志
	gaussdbuninstall.log	gaussDB 卸载日志

日志类型	日志文件名	描述
HA 脚本相关运行日志	floatip_ha.log	Floatip 资源脚本日志
	gaussDB_ha.log	gaussDB 资源脚本日志
	ha_monitor.log	HA 进程监控日志
	send_alarm.log	告警发送日志
	ha.log	HA 运行日志
DBService 审计日志	dbservice_audit.log	dbservice 操作审计日志（例如：备份恢复操作）

日志格式

DBService 的日志格式如下所示：

表5-2 日志格式

日志类型	格式	示例
运行日志	[<yyyy-MM-dd HH:mm:ss>] <Log Level>： [*<产生该日志的脚本名称：行号>*]： *<log 中的 message>*	[2020-12-19 15:56:42] INFO [postinstall.sh:653] Is cloud flag is false. (main)
审计日志	[<yyyy-MM-dd HH:mm:ss,SSS>] UserName:*<用户名称>* UserIP:*<用户 IP>* Operation:*<操作内容>* Result:*<操作结果>* Detail:*<具体信息>*	[2020-05-26 22:00:23] UserName:omm UserIP:192.168.10.21 Operation:DBService data backup Result: SUCCESS Detail: DBService data backup is successful.

6 使用 Flink

6.1 从零开始使用 Flink

本节提供使用 Flink 运行 wordcount 作业的操作指导。

前提条件

- MRS 集群中已安装 Flink 组件。
- 集群正常运行，已安装集群客户端，例如安装目录为“/opt/hadoopclient”。以下操作的客户端目录只是举例，请根据实际安装目录修改。

使用 Flink 客户端（MRS 3.x 之前版本）

步骤 1 以客户端安装用户，登录安装客户端的节点。

步骤 2 执行以下命令，切换到客户端安装目录。

cd /opt/hadoopclient

步骤 3 执行如下命令初始化环境变量。

source /opt/hadoopclient/bigdata_env

步骤 4 若集群开启 Kerberos 认证，需要执行以下步骤，若集群未开启 Kerberos 认证请跳过该步骤。

1. 准备一个提交 Flink 作业的用户。
2. 登录 Manager，下载认证凭据。

   登录集群的 Manager 界面，具体请参见访问 MRS Manager（MRS 3.x 之前版本），选择“系统设置 > 用户管理”，在已增加用户所在行的“操作”列，选择“更多 > 下载认证凭据”。
3. 将下载的认证凭据压缩包解压缩，并将得到的 user.keytab 文件拷贝到客户端节点中，例如客户端节点的“/opt/hadoopclient/Flink/flink/conf”目录下。如果是在集群外节点安装的客户端，需要将得到的 krb5.conf 文件拷贝到该节点的“/etc/”目录下。
4. 配置安全认证，在“/opt/hadoopclient/Flink/flink/conf/flink-conf.yaml”配置文件中的对应配置添加 keytab 路径以及用户名。

security.kerberos.login.keytab: *<user.keytab 文件路径>*

security.kerberos.login.principal: *<用户名>*

例如：

security.kerberos.login.keytab: /opt/hadoopclient/Flink/flink/conf/user.keytab

security.kerberos.login.principal: test

5. 参考“组件操作指南 > 使用 Flink > 参考 > 签发证书样例”章节生成“generate_keystore.sh”脚本并放置在 Flink 的客户端 bin 目录下，执行如下命令进行安全加固，请参考“组件操作指南 > 使用 Flink > 安全加固 > 认证和加密”，password 请重新设置为一个用于提交作业的密码。

   sh generate_keystore.sh <*password*>

   该脚本会自动替换“/opt/hadoopclient/Flink/flink/conf/flink-conf.yaml”中关于 SSL 的值，针对 MRS2.x 及之前版本，安全集群默认没有开启外部 SSL，用户如果需要启用外部 SSL，请参考“组件操作指南 > 使用 Flink > 安全加固”进行配置后再次运行该脚本即可。

   说明

   - generate_keystore.sh 脚本无需手动生成。
   - 执行认证和加密后会将生成的 flink.keystore、flink.truststore、security.cookie 自动填充到“flink-conf.yaml”对应配置项中。

6. 客户端访问 flink.keystore 和 flink.truststore 文件的路径配置。
   - 绝对路径：执行该脚本后，在 flink-conf.yaml 文件中将 flink.keystore 和 flink.truststore 文件路径自动配置为绝对路径“/opt/hadoopclient/Flink/flink/conf/”，此时需要将 conf 目录中的 flink.keystore 和 flink.truststore 文件分别放置在 Flink Client 以及 Yarn 各个节点的该绝对路径上。
   - 相对路径：请执行如下步骤配置 flink.keystore 和 flink.truststore 文件路径为相对路径，并确保 Flink Client 执行命令的目录可以直接访问该相对路径。
     i. 在“/opt/hadoopclient/Flink/flink/conf/”目录下新建目录，例如 ssl。

        cd /opt/hadoopclient/Flink/flink/conf/

        mkdir ssl

     ii. 移动 flink.keystore 和 flink.truststore 文件到“/opt/hadoopclient/Flink/flink/conf/ssl/”中。

        mv flink.keystore ssl/

        mv flink.truststore ssl/

     iii. 修改 flink-conf.yaml 文件中如下两个参数为相对路径。

```
security.ssl.internal.keystore: ssl/flink.keystore
security.ssl.internal.truststore: ssl/flink.truststore
```

步骤 5 运行 wordcount 作业。

须知

用户在 Flink 提交作业或者运行作业时，应具有如下权限：

- 如果启用 Ranger 鉴权，当前用户必须属于 hadoop 组或者已在 Ranger 中为该用户添加“/flink”的读写权限。
- 如果停用 Ranger 鉴权，当前用户必须属于 hadoop 组。

- 普通集群（未开启 Kerberos 认证）
  - 执行如下命令启动 session，并在 session 中提交作业。

    yarn-session.sh -nm "*session-name***"**

    flink run /opt/hadoopclient/Flink/flink/examples/streaming/WordCount.jar
  - 执行如下命令在 Yarn 上提交单个作业。

    flink run -m yarn-cluster /opt/hadoopclient/Flink/flink/examples/streaming/WordCount.jar
- 安全集群（开启 Kerberos 认证）
  - flink.keystore 和 flink.truststore 文件路径为绝对路径时：
    - 执行如下命令启动 session，并在 session 中提交作业。

      yarn-session.sh -nm "*session-name***"**

      flink run /opt/hadoopclient/Flink/flink/examples/streaming/WordCount.jar
    - 执行如下命令在 Yarn 上提交单个作业。

      flink run -m yarn-cluster /opt/hadoopclient/Flink/flink/examples/streaming/WordCount.jar
  - flink.keystore 和 flink.truststore 文件路径为相对路径时：
    - 在“ssl”的同级目录下执行如下命令启动 session，并在 session 中提交作业，其中“ssl”是相对路径，如“ssl”所在目录是“opt/hadoopclient/Flink/flink/conf/”，则在“opt/hadoopclient/Flink/flink/conf/”目录下执行命令。

      yarn-session.sh -t ssl/ -nm "*session-name***"**

      flink run /opt/hadoopclient/Flink/flink/examples/streaming/WordCount.jar
    - 执行如下命令在 Yarn 上提交单个作业。

      flink run -m yarn-cluster -yt ssl/ /opt/hadoopclient/Flink/flink/examples/streaming/WordCount.jar

步骤 6 作业提交成功后，客户端界面显示如下。

图6-1 在 Yarn 上提交作业成功

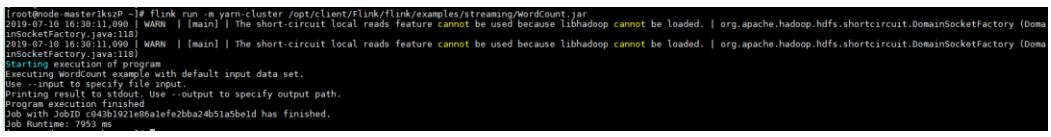
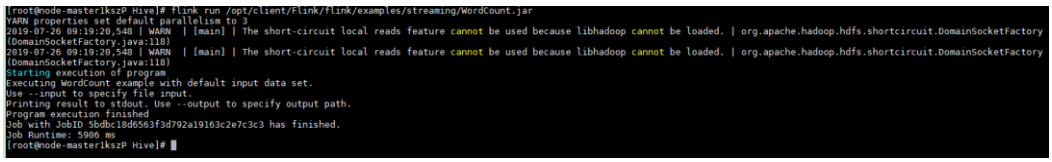


图6-2 启动 session 成功

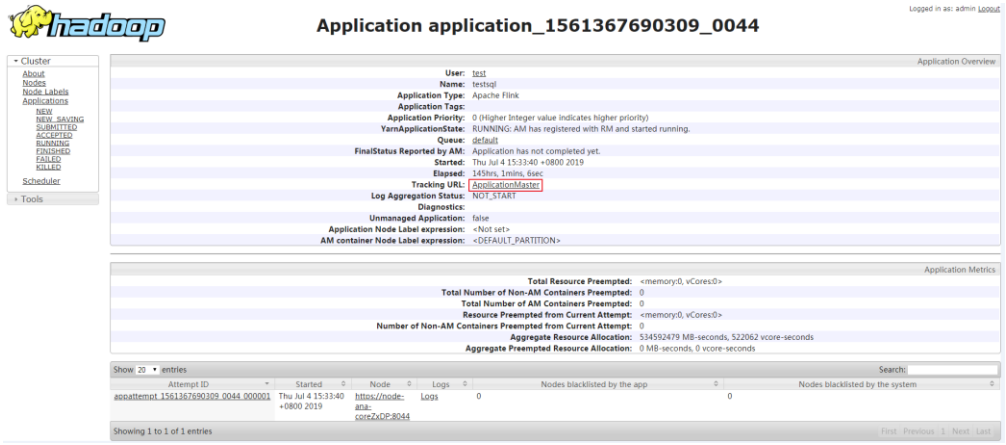
图6-3 在 session 中提交作业成功



步骤 7 使用运行用户进入 Yarn 服务的原生页面，具体操作参考“组件操作指南 > 使用 Flink > 查看 Flink 作业”，找到对应作业的 application，单击 application 名称，进入到作业详情页面。

- 若作业尚未结束，可单击“Tracking URL”链接进入到 Flink 的原生页面，查看作业的运行信息。
- 若作业已运行结束，对于在 session 中提交的作业，可以单击“Tracking URL”链接登录 Flink 原生页面查看作业信息。

图6-4 application



----结束

使用 Flink 客户端（MRS 3.x 及之后版本）

步骤 1 以客户端安装用户，登录安装客户端的节点。

步骤 2 执行以下命令，切换到客户端安装目录。

cd /opt/hadoopclient

步骤 3 执行如下命令初始化环境变量。

source /opt/hadoopclient/bigdata_env

步骤 4 若集群开启 Kerberos 认证，需要执行以下步骤，若集群未开启 Kerberos 认证请跳过该步骤。

1. 准备一个提交 Flink 作业的用户。
2. 登录 Manager，下载认证凭据。

   登录集群的 Manager 界面，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本），选择“系统 > 权限 > 用户”，在已增加用户所在行的“操作”列，选择“更多 > 下载认证凭据”。
3. 将下载的认证凭据压缩包解压缩，并将得到的 user.keytab 文件拷贝到客户端节点中，例如客户端节点的“/opt/hadoopclient/Flink/flink/conf”目录下。如果是在集群外节点安装的客户端，需要将得到的 krb5.conf 文件拷贝到该节点的“/etc/”目录下。
4. 将客户端安装节点的业务 IP、Manager 的浮动 IP 和 Master 节点 IP 添加到配置文件“/opt/hadoopclient/Flink/flink/conf/flink-conf.yaml”中的“jobmanager.web.access-control-allow-origin”和“jobmanager.web.allow-access-address”配置项中，IP 地址之间使用英文逗号分隔。

   ```
   jobmanager.web.access-control-allow-origin:
   xx.xx.xxx.xxx,xx.xx.xxx.xxx,xx.xx.xxx.xxx
   jobmanager.web.allow-access-address: xx.xx.xxx.xxx,xx.xx.xxx.xxx,xx.xx.xxx.xxx
   ```

   说明

   - 客户端安装节点的业务 IP 获取方法：
   - 集群内节点：

     登录 MapReduce 服务管理控制台，选择“集群列表 > 现有集群”，选中当前的集群并单击集群名，进入集群信息页面。

     在“节点管理”中查看安装客户端所在的节点 IP。
   - 集群外节点：安装客户端所在的弹性云主机的 IP。
   - Manager 的浮动 IP 获取方法：
   - 登录 MapReduce 服务管理控制台，选择“集群列表 > 现有集群”，选中当前的集群并单击集群名，进入集群信息页面。

     在“节点管理”中查看节点名称，名称中包含“master1”的节点为 Master1 节点，名称中包含“master2”的节点为 Master2 节点。
   - 远程登录 Master2 节点，执行“ifconfig”命令，系统回显中“eth0:wsom”表示 MRS Manager 浮动 IP 地址，请记录“inet”的实际参数值。如果在 Master2 节点无法查询到 MRS Manager 的浮动 IP 地址，请切换到 Master1 节点查询并记录。如果只有一个 Master 节点时，直接在该 Master 节点查询并记录。
5. 配置安全认证，在“/opt/hadoopclient/Flink/flink/conf/flink-conf.yaml”配置文件中的对应配置添加 keytab 路径以及用户名。

   security.kerberos.login.keytab: <*user.keytab 文件路径*>

   security.kerberos.login.principal: <*用户名*>

   例如：

security.kerberos.login.keytab: /opt/hadoopclient/Flink/flink/conf/user.keytab

security.kerberos.login.principal: test

6. 参考“组件操作指南 > 使用 Flink > 参考 > 签发证书样例”章节生成“generate_keystore.sh”脚本并放置在 Flink 的客户端 bin 目录下，执行如下命令进行安全加固，请参考“组件操作指南 > 使用 Flink > 安全加固 > 认证和加密”，password 请重新设置为一个用于提交作业的密码。

   sh generate_keystore.sh <*password*>

   该脚本会自动替换“/opt/hadoopclient/Flink/flink/conf/flink-conf.yaml”中关于 SSL 的值。

   sh generate_keystore.sh <*password*>

   说明

   执行认证和加密后会在 Flink 客户端的“conf”目录下生成“flink.keystore”和“flink.truststore”文件，并且在客户端配置文件“flink-conf.yaml”中将以下配置项进行了默认赋值：

   - 将配置项“security.ssl.keystore”设置为“flink.keystore”文件所在绝对路径。
   - 将配置项“security.ssl.truststore”设置为“flink.truststore”文件所在的绝对路径。
   - 将配置项“security.cookie”设置为“generate_keystore.sh”脚本自动生成的一串随机规则密码。
   - 默认“flink-conf.yaml”中“security.ssl.encrypt.enabled: false”，“generate_keystore.sh”脚本将配置项“security.ssl.key-password”、“security.ssl.keystore-password”和“security.ssl.truststore-password”的值设置为调用“generate_keystore.sh”脚本时输入的密码。

7. 客户端访问 flink.keystore 和 flink.truststore 文件的路径配置。
   - 绝对路径：执行该脚本后，在 flink-conf.yaml 文件中将 flink.keystore 和 flink.truststore 文件路径自动配置为绝对路径“/opt/hadoopclient/Flink/flink/conf/”，此时需要将 conf 目录中的 flink.keystore 和 flink.truststore 文件分别放置在 Flink Client 以及 Yarn 各个节点的该绝对路径上。
   - 相对路径：请执行如下步骤配置 flink.keystore 和 flink.truststore 文件路径为相对路径，并确保 Flink Client 执行命令的目录可以直接访问该相对路径。
     i. 在“/opt/hadoopclient/Flink/flink/conf/”目录下新建目录，例如 ssl。

        cd /opt/hadoopclient/Flink/flink/conf/

        mkdir ssl

     ii. 移动 flink.keystore 和 flink.truststore 文件到“/opt/hadoopclient/Flink/flink/conf/ssl/”中。

        mv flink.keystore ssl/

        mv flink.truststore ssl/

     iii. 修改 flink-conf.yaml 文件中如下两个参数为相对路径。

```
security.ssl.keystore: ssl/flink.keystore
security.ssl.truststore: ssl/flink.truststore
```

步骤 5 运行 wordcount 作业。

须知

用户在 Flink 提交作业或者运行作业时，应具有如下权限：

- 如果启用 Ranger 鉴权，当前用户必须属于 hadoop 组或者已在 Ranger 中为该用户添加“/flink”的读写权限。
- 如果停用 Ranger 鉴权，当前用户必须属于 hadoop 组。

- 普通集群（未开启 Kerberos 认证）
  - 执行如下命令启动 session，并在 session 中提交作业。

    yarn-session.sh -nm "*session-name*"

    flink run /opt/hadoopclient/Flink/flink/examples/streaming/WordCount.jar
  - 执行如下命令在 Yarn 上提交单个作业。

    flink run -m yarn-cluster /opt/hadoopclient/Flink/flink/examples/streaming/WordCount.jar
- 安全集群（开启 Kerberos 认证）
  - flink.keystore 和 flink.truststore 文件路径为绝对路径时：
    - 执行如下命令启动 session，并在 session 中提交作业。

      yarn-session.sh -nm "*session-name*"

      flink run /opt/hadoopclient/Flink/flink/examples/streaming/WordCount.jar
    - 执行如下命令在 Yarn 上提交单个作业。

      flink run -m yarn-cluster /opt/hadoopclient/Flink/flink/examples/streaming/WordCount.jar
  - flink.keystore 和 flink.truststore 文件路径为相对路径时：
    - 在“ssl”的同级目录下执行如下命令启动 session，并在 session 中提交作业，其中“ssl”是相对路径，如“ssl”所在目录是“opt/hadoopclient/Flink/flink/conf/”，则在“opt/hadoopclient/Flink/flink/conf/”目录下执行命令。

      yarn-session.sh -t ssl/ -nm "*session-name*"

      flink run /opt/hadoopclient/Flink/flink/examples/streaming/WordCount.jar
    - 执行如下命令在 Yarn 上提交单个作业。

      flink run -m yarn-cluster -yt ssl/ /opt/hadoopclient/Flink/flink/examples/streaming/WordCount.jar

步骤 6 作业提交成功后，客户端界面显示如下。

图6-5 在 Yarn 上提交作业成功

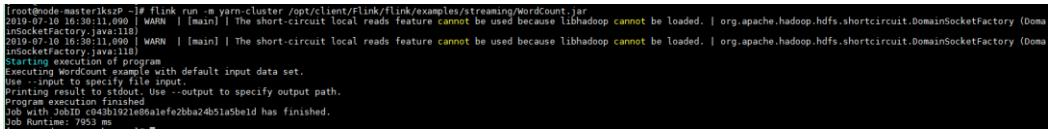
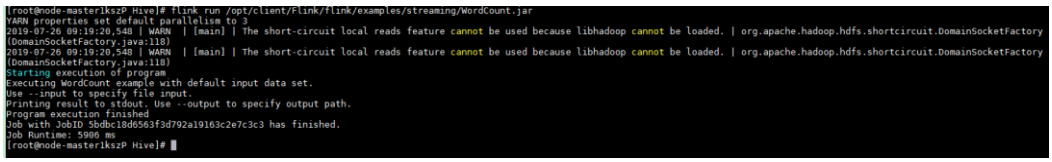


图6-6 启动 session 成功

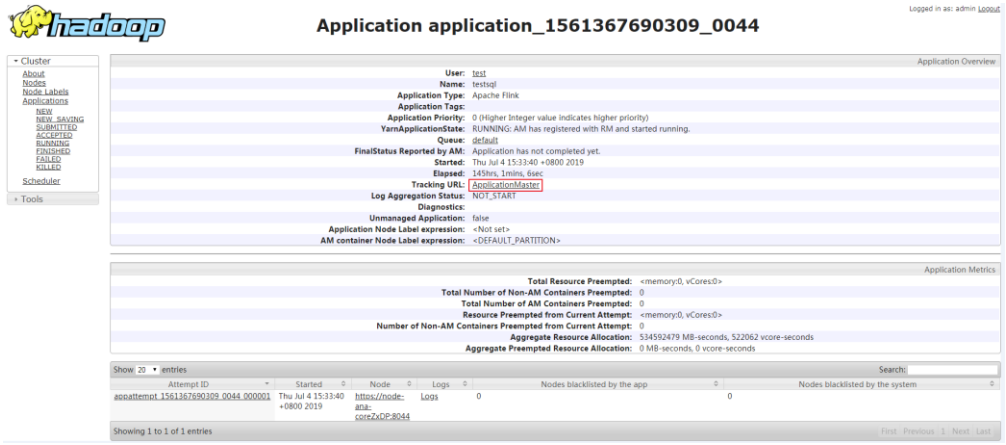
图6-7 在 session 中提交作业成功



步骤 7 使用运行用户进入 Yarn 服务的原生页面，具体操作参考“组件操作指南 > 使用 Flink > 查看 Flink 作业”，找到对应作业的 application，单击 application 名称，进入到作业详情页面

- 若作业尚未结束，可单击“Tracking URL”链接进入到 Flink 的原生页面，查看作业的运行信息。
- 若作业已运行结束，对于在 session 中提交的作业，可以单击“Tracking URL”链接登录 Flink 原生页面查看作业信息。

图6-8 application



----结束

6.2 查看 Flink 作业信息

用户可以通过 Yarn 的 WebUI，在图形化界面查看 Flink 作业的相关信息。

前提条件

集群已安装 Flink 服务。

访问 Yarn 的 WebUI

步骤 1 进入 Yarn 服务页面：

- MRS 3.x 之前版本，单击集群名称，登录集群详情页面，选择“组件管理 > Yarn > 概述”。

  说明

  若集群详情页面没有“组件管理”页签，请先完成 IAM 用户同步（在集群详情的“概览”页签，单击“IAM 用户同步”右侧的“同步”进行 IAM 用户同步）。

- MRS 3.x 及后续版本，登录 FusionInsight Manager，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本）。然后选择“集群 > *待操作的集群名称* > 服务 > Yarn > 概览”。

步骤 2 单击“ResourceManager WebUI”后面对应的链接，进入 Yarn 的 WebUI 页面。

----结束

6.3 配置管理 Flink

6.3.1 配置参数路径

Flink 所有的配置参数都需要在客户端侧进行配置，配置文件路径：*客户端安装路径*/Flink/flink/conf/flink-conf.yaml。

说明

- 建议用户直接修改客户端的“flink-conf.yaml”配置文件进行配置，YAML 文件的配置格式为 *key: value*。

  例：**taskmanager.heap.size: 1024mb**

  注意配置项 key:与 value 之间需有空格分隔。

- 如果在 Flink 服务的配置中修改了参数，配置完成之后需要重新下载安装客户端。

6.3.2 JobManager & TaskManager

配置场景

JobManager 和 TaskManager 是 Flink 的主要组件，针对各种安全场景和性能场景，可以在客户端侧配置相关参数。

配置描述

主要配置项包括通信端口，内存管理，连接重试等。

针对 MRS 3.x 之前版本，参数说明见表表 6-1。

表6-1 参数说明

参数	是否必选	默认值	描述
taskmanager.rpc.port	否	默认值为32326-32390。	TaskManager 的 IPC 端口范围。
taskmanager.data.port	否	默认值为32391-32455。	TaskManager 数据交换端口范围。
taskmanager.data.ssl.enabled	否	false	TaskManager 之间数据传输是否使用 SSL 加密，仅在全局开关 security.ssl 开启时有效。
taskmanager.numberOfTaskSlots	否	3	TaskManager 占用的 slot 数，一般配置成物理机的核数，yarn-session 模式下只能使用-s 参数传递，yarn-cluster 模式下只能使用-ys 参数传递。
parallelism.default	否	1	Job 各个算子运行的并发数。
taskmanager.memory.size	否	0	TaskManager 在 JVM 堆内存中保留空间的大小，此内存用于排序，哈希表和中间状态的缓存。如果未指定，则会使用 JVM 堆内存乘以比例 taskmanager.memory.fraction。单位：MB。
taskmanager.memory.fraction	否	0.7	TaskManager 在 JVM 堆内存中保留空间的比例，此内存用于排序，哈希表和中间状态的缓存。
taskmanager.memory.off-heap	是	false	TaskManager 是否使用堆外内存，此内存用于排序，哈希表和中间状态的缓存。建议对于大内存，开启此配置提高内存操作的效率。
taskmanager.memory.segment-size	否	32768	TaskManager 中 memory segment 大小，这是保留内存空间的基本单位，以及用于配置网络缓存栈。单位：bytes。
taskmanager.memory.preallocate	否	false	TaskManager 是否在启动时分配保留内存空间。当开启堆外内存时，建议开启此配置项。
taskmanager.registration.initial-backoff	否	500 ms	两次连续注册的初始间隔时间。单位：ms/s/m/h/d。

参数	是否必选	默认值	描述
			说明 时间数值和单位之间有半角字符空格。 ms/s/m/h/d 表示毫秒、秒、分钟、小时、天。
taskmanager.registration.refused-backoff	否	5 min	JobManager 拒绝注册后到允许再次注册的间隔时间。
task.cancellation.interval	否	30000	两次连续任务取消操作的间隔时间。

针对 MRS 3.x 及之后版本，参数说明见表 6-2。

表6-2 参数说明

参数	描述	默认值	是否必选配置
taskmanager.rpc.port	TaskManager 的 IPC 端口范围。	默认值为 32326-32390。	否
client.rpc.port	Flink client 端 Akka system 监听端口。	默认值为 32651-32720。	否
taskmanager.data.port	TaskManager 数据交换端口范围。	默认值为 32391-32455。	否
taskmanager.data.ssl.enabled	TaskManager 之间数据传输是否使用 SSL 加密，仅在全局开关 security.ssl 开启时有效。	false	否
jobmanager.heap.size	JobManager 堆内存大小，yarn-session 模式下只能使用-jm 参数传递，yarn-cluster 模式下只能使用-yjm 参数传递，如果小于 YARN 配置文件中 yarn.scheduler.minimum-allocation-mb 大小，则使用 YARN 配置中的值。单位：B/KB/MB/GB/TB。	1024mb	否
taskmanager.heap.size	TaskManager 堆内存大小，yarn-session 模式下只能使用-tm 参数传递，yarn-cluster 模式下只能使用-ytm 参数传递，如果小于 YARN 配置文件中 yarn.scheduler.minimum-allocation-mb 大小，则使用 YARN 配置中的值。单位：B/KB/MB/GB/TB。	1024mb	否
taskmanager.nu	TaskManager 占用的 slot 数，一般配置成物	1	否

参数	描述	默认值	是否必选配置
mberOfTaskSlots	理机的核数，yarn-session 模式下只能使用-s 参数传递，yarn-cluster 模式下只能使用-ys 参数传递。		
parallelism.default	默认并行度，用于未指定并行度的作业。	1	否
taskmanager.network.numberOfBuffers	TaskManager 网络传输缓冲栈数量，如果作业运行中出错提示系统中可用缓冲不足，可以增加这个配置项的值。	2048	否
taskmanager.memory.fraction	TaskManager 在 JVM 堆内存中保留空间的比例，此内存用于排序，哈希表和中间状态的缓存。	0.7	否
taskmanager.memory.off-heap	TaskManager 是否使用堆外内存，此内存用于排序，哈希表和中间状态的缓存。建议对于大内存，开启此配置提高内存操作的效率。	false	是
taskmanager.memory.segment-size	内存管理器和网络堆栈使用的内存缓冲区大小。单位：bytes。	32768	否
taskmanager.memory.preallocate	TaskManager 是否在启动时分配保留内存空间。当开启堆外内存时，建议开启此配置项。	false	否
taskmanager.debug.memory.startLogThread	调试 Flink 内存和 GC 相关问题时可开启，TaskManager 会定时采集内存和 GC 的统计信息，包括当前堆内，堆外，内存池的使用率和 GC 时间。	false	否
taskmanager.debug.memory.logIntervalMs	TaskManager 定时采集内存和 GC 的统计信息的采集间隔。	0	否
taskmanager.maxRegistrationDuration	TaskManager 向 JobManager 注册自己的最长时间，如果超过时间，TaskManager 会关闭。	5 min	否
taskmanager.initial-registration-pause	两次连续注册的初始间隔时间。该值需带一个时间单位（ms/s/min/h/d）（比如 5 秒）。	500ms 说明 时间数值和单位之间有半角字符空格。 ms/s/m/h/d	否

参数	描述	默认值	是否必选配置
		表示毫秒、秒、分钟、小时、天。	
taskmanager.max-registration-pause	TaskManager 注册失败最大重试间隔。单位：ms/s/m/h/d。	30s	否
taskmanager.refused-registration-pause	TaskManager 注册连接被 JobManager 拒绝后的重试间隔。单位：ms/s/m/h/d。	10s	否
task.cancellation.interval	两次连续任务取消操作的间隔时间。单位：ms。	30000	否
classloader.resolve-order	从用户代码加载类时定义类解析策略，这意味着是首先检查用户代码 jar（“child-first”）还是应用程序类路径（“parent-first”）。默认设置指示首先从用户代码 jar 加载类，这意味着用户代码 jar 可以包含和加载不同于 Flink 使用的（依赖）依赖项。	child-first	否
slot.idle.timeout	Slot Pool 中空闲 Slot 的超时时间（以 ms 为单位）。	50000	否
slot.request.timeout	从 Slot Pool 请求 Slot 的超时（以 ms 为单位）。	300000	否
task.cancellation.timeout	取消任务超时时间（以 ms 为单位），超时后会触发 TaskManager 致命错误。设置为 0，取消任务卡住则不会报错。	180000	否
taskmanager.network.detailed-metrics	启用网络队列长度的详细指标监控。	false	否
taskmanager.network.memory.buffers-per-channel	每个传出/传入通道（子分区/输入通道）使用的最大网络缓冲区数.在基于信用的流量控制模式下，这表示每个输入通道中有多少信用。它应配置至少 2 以获得良好的性能。1 个缓冲区用于接收子分区中的飞行中数据，1 个缓冲区用于并行序列化。	2	否
taskmanager.network.memory.floating-buffers-per-gate	每个输出/输入门（结果分区/输入门）使用的额外网络缓冲区数。在基于信用的流量控制模式中，这表示在所有输入通道之间共享多少浮动信用。浮动缓冲区基于积压（子分	8	否

参数	描述	默认值	是否必选配置
	区中的实时输出缓冲区）反馈来分布，并且可以帮助减轻由子分区之间的不平衡数据分布引起的背压。如果节点之间的往返时间较长和/或群集中的机器数量较多，则应增加此值。		
taskmanager.network.memory.fraction	用于网络缓冲区的 JVM 内存的占比。这决定了 TaskManager 可以同时拥有多少流数据交换通道以及通道缓冲的程度。如果作业被拒绝或者收到系统没有足够缓冲区的警告，请增加此值或“taskmanager.network.memory.min”和“taskmanager.network.memory.max”。另请注意，“taskmanager.network.memory.min”和“taskmanager.network.memory.max”可能会覆盖此占比。	0.1	否
taskmanager.network.memory.max	网络缓冲区的最大内存大小。该值需带一个大小单位（B/KB/MB/GB/TB）。	1 GB	否
taskmanager.network.memory.min	网络缓冲区的最小内存大小。该值需带一个大小单位（B/KB/MB/GB/TB）。	64 MB	否
taskmanager.network.request-backoff.initial	输入通道的分区请求的最小退避。	100	否
taskmanager.network.request-backoff.max	输入通道的分区请求的最大退避。	10000	否
taskmanager.registration.timeout	TaskManager 注册的超时时间，在该时间内未成功注册，TaskManager 将终止。该值需带一个时间单位（ms/s/min/h/d）。	5 min	否
resourcemanager.taskmanager-timeout	释放空闲 TaskManager 的超时（以 ms 为单位）。	30000	否

6.3.3 Blob

配置场景

JobManager 节点上的 Blob 服务端是用于接收用户在客户端上传的 Jar 包，或将 Jar 包发送给 TaskManager，传输 log 文件等。Flink 提供配置 Blob 服务端的一些配置项，用户请在“flink-conf.yaml”配置文件中进行配置。

配置描述

用户可以配置端口，SSL，重试次数，并发等配置项。

表6-3 参数说明

参数	描述	默认值	是否必选配置
blob.server.port	blob 服务器端口。	默认值为 32456-32520。	否
blob.service.ssl.enabled	blob 传输通道是否加密传输，仅在全局开关 security.ssl 开启时有。	true	是
blob.fetch.retries	TaskManager 从 JobManager 下载 blob 文件的重试次数。	50	否
blob.fetch.num-concurrent	JobManager 支持的下载 blob 的并发数。	50	否
blob.fetch.backlog	JobManager 支持的 blob 下载队列大小，比如下载 Jar 包等。单位：个。	1000	否
library-cache-manager.cleanup.interval	当用户取消 flink job 后，jobmanager 删除 HDFS 上存放用户 jar 包的时间，单位为 s。	3600	否

说明

针对 MRS 3.x 之前版本，不支持配置 library-cache-manager.cleanup.interval 参数项。

6.3.4 Distributed Coordination (via Akka)

配置场景

Flink 客户端与 JobManager 的通信，JobManager 与 TaskManager 的通信和 TaskManager 与 TaskManager 的通信都基于 Akka actor 模型。Flink 提供 Akka 连接参数的配置项，

配置项请在“flink-conf.yaml”配置文件中进行配置，用户可以根据网络环境或调优策略再进行配置。

配置描述

配置项包括消息发送和等待的超时设置，akka 监听机制 Deathwatch 的相关配置等。

针对 MRS 3.x 之前版本，参数说明见表 6-4。

表6-4 参数说明

参数	是否必选	默认值	描述
akka.ask.timeout	否	10 s	akka 所有异步请求和阻塞请求的超时时间。如果 Flink 发生超时失败，可以增大这个值。当机器处理速度慢或者网络阻塞时会发生超时。单位：ms/s/m/h/d。
akka.lookup.timeout	否	10 s	查找 JobManager actor 对象的超时时间。单位：ms/s/m/h/d。
akka.framesize	否	10485760b	JobManager 和 TaskManager 间最大消息传输大小。当 Flink 出现消息大小超过限制的错误时，可以增大这个值。单位：b/B/KB/MB。
akka.watch.heartbeat.interval	否	10 s	Akka DeathWatch 机制检测失联 TaskManager 的心跳间隔。如果 TaskManager 经常发生由于心跳消息丢失或延误而被错误标记为失联的情况，可以增大这个值。单位：ms/s/m/h/d。 **说明** DeathWatch 的详细解释可以参考 akka 官网：http://doc.akka.io/docs/akka/snapshot/scala/remoting.html#failure-detector。
akka.watch.heartbeat.pause	否	60 s	Akka DeathWatch 可接受的心跳暂停时间，较小的数值表示不允许不规律的心跳。单位：ms/s/m/h/d。 **说明** DeathWatch 的详细解释可以参考 akka 官网：http://doc.akka.io/docs/akka/snapshot/scala/remoting.html#failure-detector。
akka.watch.threshold	否	12	DeathWatch 失败检测阈值，较小的数值容易把正常 TaskManager 标记为失败，较大的值增加了失败检测的时间。 **说明** DeathWatch 的详细解释可以参考 akka 官网：http://doc.akka.io/docs/akka/snapshot/scala/remoting.html#failure-detector。

参数	是否必选	默认值	描述
akka.tcp.timeout	否	20 s	发送连接 TCP 超时时间，如果经常发生满网络环境下连接 TaskManager 超时，可以增大这个值。单位：ms/s/m/h/d。
akka.throughput	否	15	Akka 批量处理消息的数量，一次操作完后把处理线程归还线程池。较小的数值代表 actor 消息处理的公平调度，较大的值以牺牲调度公平的代价提高整体性能。
akka.log.lifecycle.events	否	false	Akka 远程时间日志开关，当需要调试时可打开此开关。
akka.startup-timeout	否	默认与 akka.ask.timeout 的值一致	Akka 启动 remote 组件的超时时间。单位：ms/s/m/h/d。
akka.ssl.enabled	是	true	Akka 通信 SSL 开关，仅在全局开关 security.ssl 开启时有。

针对 MRS 3.x 及之后版本，参数说明见表 6-5。

表6-5 参数说明

参数	描述	默认值	是否必选配置
akka.ask.timeout	akka 所有异步请求和阻塞请求的超时时间。如果 Flink 发生超时失败，可以增大这个值。当机器处理速度慢或者网络阻塞时会发生超时。单位：ms/s/m/h/d。	10s	否
akka.lookup.timeout	查找 JobManager actor 对象的超时时间。单位：ms/s/m/h/d。	10s	否
akka.framesize	JobManager 和 TaskManager 间最大消息传输大小。当 Flink 出现消息大小超过限制的错误时，可以增大这个值。单位：b/B/KB/MB。	10485760b	否
akka.watch.heartbeat.interval	Akka DeathWatch 机制检测失联 TaskManager 的心跳间隔。如果 TaskManager 经常发生由于心跳消息丢失或延误而被错误标记为失联的情况，可以增大这个值。单位：ms/s/m/h/d。 说明	10s	否

参数	描述	默认值	是否必选配置
	DeathWatch 的详细解释可以参考 akka 官网：http://doc.akka.io/docs/akka/snapshot/scala/remoting.html#failure-detector。		
akka.watch.heartbeat.pause	Akka DeathWatch 可接受的心跳暂停时间，较小的数值表示不允许不规律的心跳。单位：ms/s/m/h/d。 说明 DeathWatch 的详细解释可以参考 akka 官网：http://doc.akka.io/docs/akka/snapshot/scala/remoting.html#failure-detector。	60s	否
akka.watch.threshold	DeathWath 失败检测阈值，较小的数值容易把正常 TaskManager 标记为失败，较大的值增加了失败检测的时间。 说明 DeathWatch 的详细解释可以参考 akka 官网：http://doc.akka.io/docs/akka/snapshot/scala/remoting.html#failure-detector。	12	否
akka.tcp.timeout	发送连接 TCP 超时时间，如果经常发生满网络环境下连接 TaskManager 超时，可以增大这个值。单位：ms/s/m/h/d。	20s	否
akka.throughput	Akka 批量处理消息的数量，一次操作完后把处理线程归还线程池。较小的数值代表 actor 消息处理的公平调度，较大的值以牺牲调度公平的代价提高整体性能。	15	否
akka.log.lifecycle.events	Akka 远程时间日志开关，当需要调试时可打开此开关。	false	否
akka.startup-timeout	远程组件启动失败前的超时时间。该值需带一个时间单位（ms/s/min/h/d）	默认与 akka.ask.timeout 的值一致	否
akka.ssl.enabled	Akka 通信 SSL 开关，仅在全局开关 security.ssl 开启时有。	true	是
akka.client-socket-worker-pool.pool-size-factor	计算线程池大小的因子，计算公式：ceil（可用处理器*因子），计算结果限制在 pool-size-min 和 pool-size-max 之间。	1.0	否
akka.client-socket-worker-pool.pool-size-	基于因子计算的线程数上限。	2	否

参数	描述	默认值	是否必选配置
max			
akka.client-socket-worker-pool.pool-size-min	基于因子计算的线程数下限。	1	否
akka.client.timeout	【说明】客户端超时时间。该值需带一个时间单位（ms/s/min/h/d）。	60s	否
akka.server-socket-worker-pool.pool-size-factor	【说明】计算线程池大小的因子，计算公式：ceil（可用处理器*因子），计算结果限制在 pool-size-min 和 pool-size-max 之间。	1.0	否
akka.server-socket-worker-pool.pool-size-max	基于因子计算的线程数上限。	2	否
akka.server-socket-worker-pool.pool-size-min	基于因子计算的线程数下限。	1	否

6.3.5 SSL

配置场景

当需要配置安全 Flink 集群时，需要配置 SSL 相关配置项。

配置描述

配置项包括 SSL 开关，证书，密码，加密算法等。

针对 MRS 3.x 之前版本，参数说明见表 6-6。

表6-6 参数说明

参数	是否必选	默认值	描述
security.ssl.internal.enabled	是	按照集群的安装模式自动配置。 • 安全模式：默认为 true。 • 普通模式：默认为 false。	内部通信 SSL 总开关。

参数	是否必选	默认值	描述
security.ssl.internal.keystore	是	-	Java keystore 文件。
security.ssl.internal.keystore-password	是	-	keystore 文件解密密码。
security.ssl.internal.key-password	是	-	keystore 文件中服务端 key 的解密密码。
security.ssl.internal.truststore	是	-	truststore 文件包含公共 CA 证书。
security.ssl.internal.truststore-password	是	-	truststore 文件解密密码。
security.ssl.protocol	是	TLSv1.2	SSL 传输的协议版本。
security.ssl.algorithms	是	默认值为“TLS_RSA_WITH_AES_128_CBC_SHA256,TLS_DHE_RSA_WITH_AES_128_CBC_SHA256,TLS_DHE_DSS_WITH_AES_128_CBC_SHA256”	支持的 SSL 标准算法，具体可参考 java 官网：http://docs.oracle.com/javase/8/docs/technotes/guides/security/StandardNames.html#ciphersuites 。
security.ssl.rest.enabled	是	按照集群的安装模式自动配置。 • 安全模式：默认为 true。 • 普通模式：默认为 false。	外部通信 SSL 总开关。
security.ssl.rest.keystore	是	-	Java keystore 文件。
security.ssl.rest.keystore-password	是	-	keystore 文件解密密码。
security.ssl.rest.key-password	是	-	keystore 文件中服务端 key 的解密密码。
security.ssl.rest.truststore	是	-	truststore 文件包含公共 CA 证书。
security.ssl.rest.truststore-password	是	-	truststore 文件解密密码。

针对 MRS 3.x 及之后版本，参数说明见表 6-7。

表6-7 参数说明

参数	描述	默认值	是否必选配置
security.ssl.enabled	内部通信 SSL 总开关。	按照集群的安装模式自动配置。 • 安全模式：默认为 true。 • 非安全模式：默认为 false。	是
security.ssl.keystore	Java keystore 文件。	-	是
security.ssl.keystore-password	keystore 文件解密密码。	-	是
security.ssl.key-password	keystore 文件中服务端 key 的解密密码。	-	是
security.ssl.truststore	truststore 文件包含公共 CA 证书。	-	是
security.ssl.truststore-password	truststore 文件解密密码。	-	是
security.ssl.protocol	SSL 传输的协议版本。	TLSv1.2	是
security.ssl.algorithms	支持的 SSL 标准算法，具体可参考 java 官网：http://docs.oracle.com/javase/8/docs/technotes/guides/security/StandardNames.html#ciphersuites。	默认值为 "TLS_DHE_RSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_DHE_RSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384"	是

6.3.6 Network communication (via Netty)

配置场景

Flink 运行 Job 时，Task 之间的数据传输和反压检测都依赖 Netty，某些环境下可能需要对 Netty 参数进行配置。

配置描述

对于高级调优，可调整以下 Netty 配置项，默认配置已可满足大规模集群并发高吞吐量的任务，参数详情可参考 Netty 官网：http://netty.io/。

表6-8 参数说明

参数	描述	默认值	是否必选配置
taskmanager.network.netty.num-arenas	Netty 内存块数。	1	否
taskmanager.network.netty.server.numThreads	Netty 服务器线程的数量。	1	否
taskmanager.network.netty.client.numThreads	Netty 客户端线程数。	1	否
taskmanager.network.netty.client.connectTimeoutSec	Netty 客户端连接超时。单位：s。	120	否
taskmanager.network.netty.sendReceiveBufferSize	Netty 发送和接收缓冲区大小。默认为系统缓冲区大小（cat / proc / sys / net / ipv4 / tcp_ [rw] mem），在现代 Linux 中为 4MB。单位：bytes。	4096	否
taskmanager.network.netty.transport	Netty 传输类型，“nio”或“epoll”。	nio	否

6.3.7 JobManager Web Frontend

配置场景

JobManager 启动时，会在同一进程内启动 web 服务器。

- 用户可以访问 web 服务器获取当前 Flink 集群的信息，包括 JobManager，TaskManager 及集群内运行的 Job。
- 用户可以对 web 服务器参数进行配置。

配置描述

配置包括端口，临时目录，显示项目，错误重定向，安全相关等。

针对 MRS 3.x 之前版本，参数说明见表 6-9。

表6-9 参数说明

参数	是否必选	默认值	描述
jobmanager.web.port	否	32261-32325	web 端口，支持范围：32261-32325。
jobmanager.web.allow-access-address	是	*	web 访问白名单，ip 以逗号隔开。只有在白名单中的 ip 才能访问 web。

针对 MRS 3.x 及之后版本，参数说明见表 6-10。

表6-10 参数说明

参数	描述	默认值	是否必选配置
flink.security.enable	用户安装 Flink 集群时，需要选择“安全模式”或“普通模式”。 • 当选择“安全模式”，配置项“flink.security.enable”被自动配置为“true”。 • 当选择“普通模式”，配置项“flink.security.enable”被自动配置为“false”。 对于已经安装好的 Flink 集群，用户可以通过查看配置项“flink.security.enable”的值来区分当前安装的是安全模式还是普通模式。	按照集群的安装模式自动配置	否
rest.bind-port	web 端口，支持范围：32261-32325。	32261-32325	否
jobmanager.web.history	显示“flink.security.enable”最近的 job 数目。	5	否
jobmanager.web.checkpoints.disable	禁用 checkpoint 统计。	false	否
jobmanager.web.checkpoints.history	Checkpoint 统计记录数。	10	否
jobmanager.web.backpressure.cleanup-interval	未访问反压记录清理周期。单位：ms。	600000	否
jobmanager.web.backpressure.refresh-interval	反压记录刷新周期。单位：ms。	60000	否
jobmanager.web.backpressure.num-	计算反压使用的堆栈跟踪记录数。	100	否

参数	描述	默认值	是否必选配置
samples			
jobmanager.web.backpressure.delay-between-samples	计算反压的采样间隔。单位：ms	50	否
jobmanager.web.ssl.enabled	web 是否使用 SSL 加密传输，仅在全局开关 security.ssl 开启时有。	false	是
jobmanager.web.accesslog.enable	web 操作日志使能开关，日志会存放在 webaccess.log 中。	true	是
jobmanager.web.x-frame-options	http 安全头 X-Frame-Options 的值，可选范围为：SAMEORIGIN、DENY、ALLOW-FROM uri。	DENY	是
jobmanager.web.cache-directive	web 页面是否支持缓存。	no-store	是
jobmanager.web.expires-time	web 页面缓存过期时长。单位：ms。	0	是
jobmanager.web.allow-access-address	web 访问白名单，ip 以逗号隔开。只有在白名单中的 ip 才能访问 web。	*	是
jobmanager.web.access-control-allow-origin	网页同源策略，防止跨域攻击。	*	是
jobmanager.web.refresh-interval	web 网页刷新时间。单位：ms。	3000	是
jobmanager.web.logout-timer	配置无操作情况下自动登出时间间隔。单位：ms。	600000	是
jobmanager.web.403-redirect-url	web403 页面，访问若遇到 403 错误，则会重定向到配置的页面。	自动配置	是
jobmanager.web.404-redirect-url	web404 页面，访问若遇到 404 错误，则会重定向到配置的页面。	自动配置	是
jobmanager.web.415-redirect-url	web415 页面，访问若遇到 415 错误，则会重定向到配置的页面。	自动配置	是
jobmanager.web.500-redirect-url	web500 页面，访问若遇到 500 错误，则会重定向到配置的页面。	自动配置	是
rest.await-leader-timeout	客户端等待 Leader 地址的时间（以 ms 为单位）。	30000	否
rest.client.max-content-length	客户端处理的最大内容长度（以字节为单位）。	104857600	否

参数	描述	默认值	是否必选配置
rest.connection-timeout	客户端建立 TCP 连接的最长时间（以 ms 为单位）。	15000	否
rest.idleness-timeout	连接保持空闲状态的最长时间（以 ms 为单位）。	300000	否
rest.retry.delay	客户端在连续重试之间等待的时间（以 ms 为单位）。	3000	否
rest.retry.max-attempts	如果可重试算子操作失败，客户端将尝试重试的次数。	20	否
rest.server.max-content-length	服务端处理的最大内容长度（以字节为单位）。	104857600	否
rest.server.numThreads	异步处理请求的最大线程数。	4	否
web.timeout	web 监控超时时间（以 ms 为单位）。	10000	否

6.3.8 File Systems

配置场景

task 运行中会创建结果文件，Flink 支持对文件创建行为进行配置。

配置描述

配置项包括文件覆盖策略，目录创建。

表6-11 参数说明

参数	描述	默认值	是否必选配置
fs.overwrite-files	文件输出写操作是否默认覆盖已有文件。	false	否
fs.output.always-create-directory	当文件写入程序的并行度大于 1 时，输出文件的路径下会创建一个目录，并将不同的结果文件（每个并行写程序任务一个）放入该目录。 • 如果此选项设置为 true，那么并行度为 1 的写入程序也将创建一个目录并将一个结果文件放入其中。 • 如果该选项设置为 false，则并行度为 1 的写入程序将直接在输出路径	false	否

参数	描述	默认值	是否必选配置
	中创建文件，而不再创建目录。		

6.3.9 State Backend

配置场景

Flink 提供了 HA 和作业的异常恢复，并且提供版本升级时作业的暂停恢复。对于作业状态的存储，Flink 依赖于 state backend，作业的重启依赖于重启策略，用户可以对这两部分进行配置。

配置描述

配置项包括 state backend 类型，存储路径，重启策略等。

表6-12 参数说明

参数	描述	默认值	是否必选配置
state.backend.fs.checkpointdir	当 backend 为 filesystem 时的路径，路径必须能够被 JobManager 访问到，本地路径只支持 local 模式，集群模式下请使用 HDFS 路径。	hdfs:///flink/checkpoints	否
state.savepoints.dir	Flink 用于恢复和更新作业的保存点存储目录。当触发保存点的时候，保存点元数据信息将会保存到该目录中。	hdfs:///flink/savepoint	安全模式下必配
restart-strategy	默认重启策略，用于未指定重启策略的作业。三个值可选： • fixed-delay • failure-rate • none	none	否
restart-strategy.fixed-delay.attempts	fixed-delay 策略重试次数，具体策略的介绍请参见：https://ci.apache.org/projects/flink/flink-docs-release-1.12/dev/task_failure_recove	• 作业中开启了checkpoint，则默认值为Integer.MAX_VALUE。 • 作业中未开启	否

参数	描述	默认值	是否必选配置
	ry.html。	checkpoint，默认值为 3。	
restart-strategy.fixed-delay.delay	fixed-delay 策略重试间隔时间。单位：ms/s/m/h/d。	• 作业中开启了checkpoint，默认值是 10 s。 • 作业中不开启checkpoint，默认值和配置项akka.ask.timeout的值一致。	否
restart-strategy.failure-rate.max-failures-per-interval	故障率策略下作业失败前给定时间段内的最大重启次数。具体策略的介绍请参见：https://ci.apache.org/projects/flink/flink-docs-release-1.12/dev/task_failure_recovery.html。	1	否
restart-strategy.failure-rate.failure-rate-interval	failure-rate 策略重试时间。单位：ms/s/m/h/d。	60 s	否
restart-strategy.failure-rate.delay	failure-rate 策略重试间隔时间。单位：ms/s/m/h/d。	默认值和akka.ask.timeout 配置值一样，请参见Distributed Coordination (via Akka)	否

6.3.10 Kerberos-based Security

配置场景

Flink 安全模式下必须配置 Kerberos 相关配置项。

配置描述

配置项包括 kerberos 的 keytab、principal 等。

表6-13 参数说明

参数	描述	默认值	是否必选配置
security.kerberos.login.keytab	该参数为客户端参数，keytab 路径。	根据实际业务配置	是
security.kerberos.login.principal	该参数为客户端参数，如果 keytab 和 principal 都设置，默认会使用 keytab 认证。	根据实际业务配置	否
security.kerberos.login.contexts	该参数为服务器端参数，flink 生成 jass 文件的 contexts。	Client、KafkaClient	是

6.3.11 HA

配置场景

Flink 的 HA 模式依赖于 ZooKeeper，所以必须配置 ZooKeeper 相关配置。

配置描述

配置项包括 ZooKeeper 地址，路径，安全认证等。

表6-14 参数说明

参数	描述	默认值	是否必选配置
high-availability	HA 模式，是启用 HA 还是非 HA 模式。当前支持两种模式： 1. none，只运行单个 jobManager，jobManager 的状态不进行 Checkpoint。 2. ZooKeeper。 • 非 YARN 模式下，支持多个 jobManager，通过选举产生 leader。 • YARN 模式下只存在一个 jobManager。	zookeeper	否
high-availability.zookeeper.quorum	ZooKeeper quorum 地址。	自动配置	否
high-	Flink 在 ZooKeeper 上创建的根目	/flink	否

参数	描述	默认值	是否必选配置
availability.zookeeper.path.root	录，存放 HA 模式必须的元数据。		
high-availability.storageDir	存放 state backend 中 JobManager 元数据，ZooKeeper 只保存实际数据的指针。	hdfs:///flink/recovery	否
high-availability.zookeeper.client.session-timeout	ZooKeeper 客户端会话超时时间。单位：ms。	60000	否
high-availability.zookeeper.client.connection-timeout	ZooKeeper 客户端连接超时时间。单位：ms。	15000	否
high-availability.zookeeper.client.retry-wait	ZooKeeper 客户端重试等待时间。单位：ms。	5000	否
high-availability.zookeeper.client.max-retry-attempts	ZooKeeper 客户端最大重试次数。	3	否
high-availability.job.delay	当 jobManager 恢复后重启 job 的延迟时间。	默认值和 akka.ask.timeout 配置值保持一致。	否
high-availability.zookeeper.client.acl	设置 ZooKeeper 节点的 ACL (open creator)。设置 ACL 选项请参考：https://zookeeper.apache.org/doc/r3.5.1-alpha/zookeeperProgrammers.html#sc_BuiltinACLSchemes。	按照集群的安装模式自动配置： • 安全模式：creator • 非安全模式：open	是
zookeeper.sasl.disable	基于 SASL 认证的使能开关。	按照集群的安装模式自动配置： • 安全模式：false • 非安全模式：true	是
zookeeper.sasl.service-name	• 如果 ZooKeeper 服务端配置了不同于“ZooKeeper”的服务名，可以设置此配置项。 • 如果客户端和服务端的服务名不	zookeeper	是

参数	描述	默认值	是否必选配置
	一致，认证会失败。		

说明

针对 MRS 3.x 之前版本，不支持 high-availability.job.delay 配置参数。

6.3.12 Environment

配置场景

对于 JVM 配置有特定要求的场景，可以通过配置项传递 JVM 参数到客户端，JobMananger，TaskManager 等。

配置描述

可配置 JVM 参数。

表6-15 参数说明

参数	描述	默认值	是否必选配置
env.java.opts	JVM 参数，会传递到启动脚本，JobManager，TaskManager，Yarn 客户端。比如传递远程调试的参数等。	-Xloggc:<LOG_DIR>/gc.log -XX:+PrintGCDetails -XX:-OmitStackTraceInFastThrow -XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=20 -XX:GCLogFileSize=20M -Djdk.tls.ephemeralDHKeySize=2048 -Djava.library.path=${HADOOP_COMMON_HOME}/lib/native -Djava.net.preferIPv4Stack=true -Djava.net.preferIPv6Addresses=false -Dbeetle.application.home.path=/opt/xxx/Bigdata/common/runtime/security/config	否

6.3.13 Yarn

配置场景

Flink 运行在 Yarn 集群上时，JobManager 运行在 Application Master 上。JobManager 的一些配置参数依赖于 Yarn，通过配置 YARN 相关的配置，使 Flink 更好的运行在 Yarn 上。

配置描述

配置项包括 yarn container 的内存，虚拟内核，端口等。

表6-16 参数说明

参数	描述	默认值	是否必选配置
yarn.maximum-failed-containers	当 TaskManager 所属容器出错后，重新申请 container 次数。默认值为 Flink 集群启动时 TaskManager 的数量。	5	否
yarn.application-attempts	Application master 重启次数，次数是算在一个 validity interval 的最大次数，validity interval 在 flink 中设置为 akka 的 timeout。重启后 AM 的地址和端口会变化，client 需要手动连接。	2	否
yarn.heartbeat-delay	Application Master 和 YARN Resource Manager 心跳的时间间隔。单位：seconds	5	否
yarn.containers.vcores	每个 Yarn 容器的虚拟核数。	默认值是 TaskManager 的 slot 数	否
yarn.application-master.port	Application Master 端口号设置，支持端口范围。	32586-32650	否

6.3.14 Pipeline

配置场景

为适应某些场景对降低时延的需求，设计多个 Job 间采用 Netty 直接相连的方式传递数据，即分别使用 NettySink 用于 Server 端、NettySource 用于 Client 端进行数据传输。

本章节适用于 MRS 3.x 及之后版本。

配置描述

配置项包括 NettySink 的信息存放路径、NettySink 的端口监听范围、连接是否通过 SSL 加密以及 NettySink 监听所使用的网络所在域等。

表6-17 参数说明

参数	描述	默认值	是否必选配置
nettyconnector.registerserver.topic.storage	设置 NettySink 的 IP、端口及并发度信息在第三方注册服务器上的路径。建议用户使用 ZooKeeper 进行存储。	/flink/nettyconnector	否，当使用 pipeline 特性为必选
nettyconnector.sinkserver.port.range	设置 NettySink 的端口范围。	MRS 集群下默认设置为 28444-28843	否，当使用 pipeline 特性为必选
nettyconnector.ssl.enabled	设置 NettySink 与 NettySource 之间通信是否配置 SSL 加密。其中加密密钥以及加密协议等请参见 SSL。	false	否，当使用 pipeline 特性为必选
nettyconnector.message.delimiter	用来配置 nettysink 发送给 nettysource 消息的分隔符，长度为 2-4 个字节，不可包含“\n”，“ ”，“#” 。	默认使用“$_”。	否，当使用 pipeline 特性为必选

6.4 安全配置

6.4.1 安全特性描述

Flink 主要完成如下安全特性：

- Flink 集群中，各部件支持认证。
  - Flink 集群内部各部件和外部部件之间，支持和外部部件如 YARN、HDFS、ZooKeeprer 进行 kerberors 认证。
  - Flink 集群内部各部件之间，如 Flink client 和 JobManager、JobManager 和 TaskManager、TaskManager 和 TaskManager 之间支持 security cookie 认证。
- Flink 集群中，各部件支持 SSL 加密传输。
- Flink 集群内部各部件之间，如 Flink client 和 JobManager、JobManager 和 TaskManager、TaskManager 和 TaskManager 之间支持 SSL 加密传输。
- Flink web 安全加固。
  - 支持白名单过滤，Flink web 只能通过 YARN 代理访问。
  - 安全头域增强。
- Flink 集群中，各部件的监听端口支持范围可配置。
- 在 HA 模式下，支持 ACL 控制。

6.4.2 配置对接 Kafka

Flink 样例工程的数据存储在 Kafka 组件中。向 Kafka 组件发送数据（需要有 Kafka 权限用户），并从 Kafka 组件接收数据。

步骤 1 确保集群安装完成，包括 HDFS、Yarn、Flink 和 Kafka。

步骤 2 创建 Topic。

- 用户使用 Linux 命令行创建 topic，执行命令前需要使用 kinit 命令进行人机认证，如 **kinit flinkuser**。

  说明

  flinkuser 需要用户自己创建，并拥有创建 Kafka 的 topic 权限。

  创建 topic 的命令格式：{zkQuorum}表示 ZooKeeper 集群信息，格式为 IP:port。{Topic}表示 Topic 名称。

  bin/kafka-topics.sh --create --zookeeper {zkQuorum}/kafka --replication-factor 1 --partitions 5 --topic {Topic}

  例如此处以 topic1 的数据为例：

  ```
  /opt/client/Kafka/kafka/bin/kafka-topics.sh --create --zookeeper
  10.96.101.32:2181,10.96.101.251:2181,10.96.101.177:2181,10.91.8.160:2181/kafka
  --replication-factor 1 --partitions 5 --topic topic1
  ```

- 服务端 topic 权限配置。

  将 Kafka 的 Broker 配置参数“allow.everyone.if.no.acl.found”的值修改为“true”。

步骤 3 安全认证。

安全认证的方式有三种：Kerberos 认证、SSL 加密认证和 Kerberos+SSL 模式认证，用户在使用的时候可任选其中一种方式进行认证。

说明

针对 MRS 3.x 之前版本，安全认证的方式只支持 Kerberos 认证。

- **Kerberos 认证配置**
  - 客户端配置。

    在 Flink 配置文件“flink-conf.yaml”中，增加 kerberos 认证相关配置（主要在“contexts”项中增加“KafkaClient”），示例如下：

    ```
    security.kerberos.login.keytab: /home/demo/keytab/flinkuser.keytab
    security.kerberos.login.principal: flinkuser
    security.kerberos.login.contexts: Client,KafkaClient
    security.kerberos.login.use-ticket-cache: false
    ```

    说明

    针对 MRS 3.x 之前版本，配置 security.kerberos.login.keytab 示例为：/home/demo/flink/release/keytab/flinkuser.keytab。

  - 运行参数。

    关于“SASL_PLAINTEXT”协议的运行参数示例如下：

```
--topic topic1 --bootstrap.servers 10.96.101.32:21007 --security.protocol
SASL_PLAINTEXT  --sasl.kerberos.service.name kafka //10.96.101.32:21007 表示
kafka 服务器的 IP:port
```

- **SSL 加密配置**
  - 服务端配置。

    登录 FusionInsight Manager 页面，选择“集群 > 服务 > Kafka > 配置”，参数类别设置为“全部配置”，搜索“ssl.mode.enable”并配置为“true”。
  - 客户端配置。

    i. 登录集群的 FusionInsight Manager，选择“集群 > *待操作的集群名称* > 服务 > Kafka > 更多 > 下载客户端”，下载客户端压缩文件到本地机器。

    ii. 使用客户端根目录中的“ca.crt”证书文件生成客户端的“truststore”。

    执行命令如下：

    ```
    keytool -noprompt -import -alias myservercert -file ca.crt -keystore
    truststore.jks
    ```

    命令执行结果查看：

    ```
    drwx------.  5 zgd  users     4096 Feb  4 16:22 .
    drwxr-xr-x. 10 zgd  users     4096 Jan 22 17:38 ..
    -rwx------.  1 zgd  users      135 Jan 22 17:31 application.properties
    -rwx------.  1 zgd  users      790 Jan 22 17:31 bigdata_env.sample
    -rw-------.  1 zgd  users     1322 Jan 22 17:31 ca.crt
    -rwx------.  1 zgd  users     4508 Jan 22 17:31 conf.py
    -rw-------.  1 zgd  users      120 Jan 22 17:31 hosts
    -rwx------.  1 zgd  users      745 Jan 22 17:31 install.bat
    -rwx------.  1 zgd  users    15082 Jan 22 17:31 install.sh
    drwx------.  2 zgd  users     4096 Jan 22 17:38 JDK
    -rwx------.  1 zgd  users 37021723 Jan 22 17:31 jython-standalone-2.7.0.jar
    drwx------.  5 zgd  users     4096 Jan 22 17:38 Kafka
    drwx------.  3 zgd  users     4096 Jan 22 17:38 KrbClient
    -rwx------.  1 zgd  users      473 Jan 22 17:31 log4j.properties
    -rwx------.  1 zgd  users     2107 Jan 22 17:31 README
    -rwx------.  1 zgd  users     6949 Jan 22 17:31 refreshConfig.sh
    -rwx------.  1 zgd  users     1736 Jan 22 17:31 switchuser.py
    -rw-r--r--   1 root root      1004 Feb  4 16:22 truststore.jks
    ```

    iii. 运行参数。

    “ssl.truststore.password”参数内容需要跟创建“truststore”时输入的密码保持一致，执行以下命令运行参数。

    ```
    --topic topic1 --bootstrap.servers 10.96.101.32:9093 --
    security.protocol SSL --ssl.truststore.location
    /home/zgd/software/FusionInsight_Kafka_ClientConfig/truststore.jks --
    ssl.truststore.password XXX
    ```

- **Kerberos+SSL 模式配置**

  完成上文中 Kerberos 和 SSL 各自的服务端和客户端配置后，只需要修改运行参数中的端口号和协议类型即可启动 Kerberos+SSL 模式。

  ```
  --topic topic1 --bootstrap.servers 10.96.101.32:21009 --security.protocol
  SASL SSL  --sasl.kerberos.service.name kafka --ssl.truststore.location
  /home/zgd/software/FusionInsight Kafka ClientConfig/truststore.jks --
  ssl.truststore.password XXX
  ```

----结束

6.4.3 配置 Pipeline

本章节适用于 MRS 3.x 及之后版本。

1. 配置文件。
   - nettyconnector.registerserver.topic.storage：设置 NettySink 的 IP、端口及并发度信息在第三方注册服务器上的路径（必填），例如：

     ```
     nettyconnector.registerserver.topic.storage: /flink/nettyconnector
     ```

   - nettyconnector.sinkserver.port.range：设置 NettySink 的端口范围（必填），例如：

     ```
     nettyconnector.sinkserver.port.range: 28444-28843
     ```

   - nettyconnector.ssl.enabled：设置 NettySink 与 NettySource 之间通信是否 SSL 加密（默认为 false），例如：

     ```
     nettyconnector.ssl.enabled: true
     ```

2. 安全认证配置。
   - Zookeeper 的 SASL 认证，依赖“flink-conf.yaml”中有关 HA 的相关配置。
   - SSL 的 keystore、truststore、keystore password、truststore password 以及 password 等也使用“flink-conf.yaml”的相关配置，具体配置请参见加密传输。

6.5 安全加固

6.5.1 认证和加密

安全认证

Flink 整个系统有三种认证方式：

- 使用 kerberos 认证：Flink yarn client 与 Yarn Resource Manager、JobManager 与 Zookeeper、JobManager 与 HDFS、TaskManager 与 HDFS、Kafka 与 TaskManager、TaskManager 和 Zookeeper。
- 使用 security cookie 进行认证：Flink yarn client 与 Job Manager、JobManager 与 TaskManager、TaskManager 与 TaskManager。
- 使用 YARN 内部的认证机制：Yarn Resource Manager 与 Application Master（简称 AM）。

说明

- Flink 的 JobManager 与 YARN 的 AM 是在同一个进程下。
- 如果用户集群开启 Kerberos 认证需要使用 kerberos 认证。
- 针对 MRS 3.x 之前版本，Flink 不支持使用 security cookie 方式进行认证。

表6-18 安全认证方式

安全认证方式	说明	配置方法

Kerberos认证	当前只支持keytab认证方式。	1. 从KDC服务器上下载用户keytab，并将keytab放到Flink客户端所在主机的某个文件夹下。
		2. 在“flink-conf.yaml”上配置：
		a. keytab路径。
		`security.kerberos.login.keytab: /home/flinkuser/keytab/abc222.keytab`
		说明：
		“/home/flinkuser/keytab/abc222.keytab”表示的是用户目录。
		b. principal名。
		`security.kerberos.login.principal: abc222`
		c. 对于HA模式，如果配置了ZooKeeper，还需要设置ZK kerberos认证相关的配置。配置如下：
		`zookeeper.sasl.disable: false`
		`security.kerberos.login.contexts: Client`
		d. 如果用户对于Kafka client和Kafka broker之间也需要做kerberos认证，配置如下：
		`security.kerberos.login.contexts: Client,KafkaClient`

Security Cookie认证	-	1. 参考签发证书样例章节生成“generate_keystore.sh”脚本并放置在Flink客户端的“bin”目录下，调用“generate_keystore.sh”脚本，生成“Security Cookie”、“flink.keystore”文件和“flink.truststore”文件。 执行 **sh generate_keystore.sh**，输入用户自定义密码。密码不允许包含#。 说明 执行脚本后，在Flink客户端的“conf”目录下生成“flink.keystore”和“flink.truststore”文件，并且在客户端配置文件“flink-conf.yaml”中将以下配置项进行了默认赋值。 • 将配置项“security.ssl.keystore”设置为“flink.keystore”文件所在绝对路径。 • 将配置项“security.ssl.truststore”设置为“flink.truststore”文件所在的绝对路径。 • 将配置项“security.cookie”设置为“generate_keystore.sh”脚本自动生成的一串随机规则密码。 • 默认“flink-conf.yaml”中“security.ssl.encrypt.enabled: false”，“generate_keystore.sh”脚本将配置项“security.ssl.key-password”、“security.ssl.keystore-password”和“security.ssl.truststore-password”的值设置为调用“generate_keystore.sh”脚本时输入的密码。

安全认证方式	说明	配置方法
		2. 打开“Security Cookie”开关，配置 flink-conf.yaml 文件中的“security.enable: true”，查看“security cookie”是否已配置成功，例如： `security.cookie: ae70acc9-9795-4c48-ad35-8b5adc8071744f605d1d-2726-432e-88ae-dd39bfec40a9` 说明 用户需要获取 SSL 证书，放置到 Flink 客户端中。具体操作可参考签发证书样例。
YARN 内部认证方式	该方式是 YARN 内部的认证方式，不需要用户配置。	-

说明

当前一个 Flink 集群只支持一个用户，一个用户可以创建多个 Flink 集群。

加密传输

Flink 整个系统有三种加密传输方式：

- 使用 Yarn 内部的加密传输方式：Flink yarn client 与 Yarn Resource Manager、Yarn Resource Manager 与 Job Manager。
- SSL：Flink yarn client 与 JobManager、JobManager 与 TaskManager、TaskManager 与 TaskManager。
- 使用 Hadoop 内部的加密传输方式：JobManager 和 HDFS、TaskManager 和 HDFS、JobManager 与 ZooKeeper、TaskManager 与 ZooKeeper。

说明

Yarn 内部和 Hadoop 内部都不需要用户配置加密，用户只需要配置 SSL 加密传输方式。

配置 SSL 传输，用户主要在客户端的“flink-conf.yaml”文件中做如下配置：

1. 打开 SSL 开关和设置 SSL 加密算法，针对 MRS 3.x 及之后版本，配置参数如表 6-19 所示，请根据实际情况修改对应参数值。

表6-19 参数描述

参数	参数值示例	描述

参数	参数值示例	描述
security.ssl.enabled	true	打开 SSL 总开关。
akka.ssl.enabled	true	打开 akka SSL 开关。
blob.service.ssl.enabled	true	打开 blob 通道 SSL 开关。
taskmanager.data.ssl.enabled	true	打开 taskmanager 之间通信的 SSL 开关。
security.ssl.algorithms	TLS_DHE_RSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_DHE_RSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	设置 SSL 加密的算法。

针对 MRS 3.x 之前版本，配置参数如表 6-20 所示。

表6-20 参数描述

参数	参数值示例	描述
security.ssl.internal.enabled	true	打开内部 SSL 开关。
akka.ssl.enabled	true	打开 akka SSL 开关。
blob.service.ssl.enabled	true	打开 blob 通道 SSL 开关。
taskmanager.data.ssl.enabled	true	打开 taskmanager 之间通信的 SSL 开关。
security.ssl.algorithms	TLS_RSA_WITH_AES128CBC_SHA256	设置 SSL 加密的算法。

针对 MRS 3.x 之前版本，如下参数见表 6-21，在 MRS 的 Flink 默认配置中不存在，用户如果开启外部连接 SSL，则需要添加以下参数。开启外部连接 SSL 后，因为 YARN 目前的开源版本无法代理 HTTPS 请求，所以无法通过 YARN 代理访问 Flink 的原生页面，用户可以在集群的同一个 VPC 下，创建 windows 虚拟机，在该虚拟机中访问 Flink 原生页面。

表6-21 参数描述

参数	参数值示例	描述
security.ssl.rest.enabled	true	打开外部 SSL 开关，若该参数配置为“true”，请参考表 6-21 配置相关参数。

参数	参数值示例	描述
security.ssl.rest.keystore	${path}/flink.keystore	keystore 的存放路径。
security.ssl.rest.keystore-password	-	keystore 的 password，-表示需要用户输入自定义设置的密码值。
security.ssl.rest.key-password	-	ssl key 的 password，-表示需要用户输入自定义设置的密码值。
security.ssl.rest.truststore	${path}/flink.truststore	truststore 存放路径。
security.ssl.rest.truststore-password	-	truststore 的 password，-表示需要用户输入自定义设置的密码值。

说明

如果打开 Task Manager 之间 data 传输通道的 SSL，对性能会有较大影响，需要用户从安全和性能综合考虑。

2. 参考签发证书样例章节生成“generate_keystore.sh”脚本并放置在 Flink 客户端的 bin 目录下，执行命令 **sh generate_keystore.sh** <*password*>，请参考认证和加密，针对 MRS 3.x 及之后版本，表 6-22 中的配置项会被默认赋值，用户也可以手动配置。

表6-22 参数描述

参数	参数值示例	描述
security.ssl.keystore	${path}/flink.keystore	keystore 的存放路径，“flink.keystore”表示用户通过 generate_keystore.sh*工具生成的 keystore 文件名称。
security.ssl.keystore-password	-	keystore 的 password，-表示需要用户输入自定义设置的密码值。
security.ssl.key-password	-	ssl key 的 password，-表示需要用户输入自定义设置的密码值。
security.ssl.truststore	${path}/flink.truststore	truststore 存放路径，“flink.truststore”表示用户通过 generate_keystore.sh*工具生成的 truststore 文件名称。
security.ssl.truststore-password	-	truststore 的 password，-表示需要用户输入自定义设置的密码值。

针对 MRS 3.x 之前版本，***generate_keystore.sh*** 不需手动生成，表 6-23 中的配置项会被默认赋值，用户也可以手动配置。

表6-23 参数描述

参数	参数值示例	描述
security.ssl.internal.keystore	${path}/flink.keystore	keystore 的存放路径，“flink.keystore”表示用户通过 generate_keystore.sh*工具生成的 keystore 文件名称。
security.ssl.internal.keystore-password	-	keystore 的 password，表示需要用户输入自定义设置的密码值。
security.ssl.internal.key-password	-	ssl key 的 password，表示需要用户输入自定义设置的密码值。
security.ssl.internal.truststore	${path}/flink.truststore	truststore 存放路径，“flink.truststore”表示用户通过 generate_keystore.sh*工具生成的 truststore 文件名称。
security.ssl.internal.truststore-password	-	truststore 的 password，表示需要用户输入自定义设置的密码值。

针对 MRS 3.x 之前版本，如果开启外部连接 SSL，即 security.ssl.rest.enabled 配置为 true，则如下参数见表 6-24，用户需要配置。

表6-24 参数说明

参数	参数值示例	描述
security.ssl.rest.enabled	true	打开外部 SSL 开关，若该参数配置为“true”，请参考表 6-24 配置相关参数。
security.ssl.rest.keystore	${path}/flink.keystore	keystore 的存放路径
security.ssl.rest.keystore-password	-	keystore 的 password，表示需要用户输入自定义设置的密码值。
security.ssl.rest.key-password	-	ssl key 的 password，表示需要用户输入自定义设置的密码值。
security.ssl.rest.truststore	${path}/flink.truststore	truststore 存放路径
security.ssl.rest.truststore-password	-	truststore 的 password，表示需要用户输入自定义设置的密码值。

说明

path”目录是用来存放 SSL keystore、truststore 相关配置文件，该目录是由用户自定义创建。相对路径和绝对路径的不同导致执行命令存在差异，详细说明在 3 和 4 中说明。

3. 配置 keystore 或 truststore 文件路径为相对路径时，Flink Client 执行命令的目录需要可以直接访问该相对路径。Flink 有两种执行方式来传输 keystore 和 truststore 文件。
   - 在 Flink 的 CLI yarn-session.sh 命令中增加“-t”选项来传输 keystore 和 truststore 文件到各个执行节点。如：

     ```
     ./bin/yarn-session.sh -t ssl/
     ```

   - 在 Flink run 命令中增加“-yt”选项来传输 keystore 和 truststore 文件到各个执行节点。如：

     ```
     ./bin/flink run -yt ssl/ -ys 3  -m yarn-cluster -c
     org.apache.flink.examples.java.wordcount.WordCount
     /opt/client/Flink/flink/examples/batch/WordCount.jar
     ```

说明

- 在举例当中的“ssl/”是 Flink Client 端目录下的子目录，该目录是用来存放 SSL keystore、truststore 相关配置文件。
- Flink Client 执行命令的当前路径需要能访问到“ssl/”相对路径。

4. 配置 keystore 或 truststore 文件路径为绝对路径时，需要在 Flink Client 以及各个节点的该绝对路径上放置 keystore 和 truststore 文件。

说明

针对 MRS 3.x 之前版本，提交作业的用户需要具有读取 keystore 和 truststore 文件的权限。

Flink 有两种方式执行应用程序，且执行命令中不需要使用“-t”或“-yt”来传输 keystore 和 truststore 文件。

- 使用 Flink 的 CLI yarn-session.sh 命令执行应用程序。如：

  ```
  ./bin/yarn-session.sh
  ```

- 使用 Flink run 命令执行应用程序。如：

  ```
  ./bin/flink run  -ys 3 -m yarn-cluster -c
  org.apache.flink.examples.java.wordcount.WordCount
  /opt/client/Flink/flink/examples/batch/WordCount.jar
  ```

6.5.2 ACL 控制

Flink 在 HA 模式下，支持用 ZooKeeper 来管理集群和发现服务。ZooKeeper 支持 SASL ACL 控制，即只有通过 SASL（kerberos）认证的用户，才有往 ZK 上操作文件的权限。如果要在 Flink 上使用 SASL ACL 控制，需要在 Flink 配置文件中设置如下配置：

```
high-availability.zookeeper.client.acl: creator
zookeeper.sasl.disable: false
```

具体配置项介绍请参考表 6-14。

6.5.3 web 安全

编码规范

说明：Web Service 客户端和服务器间使用相同的编码方式，是为了防止出现乱码现象，也是实施输入校验的基础。

安全加固：web server 响应消息统一采用 UTF-8 字符编码。

支持 IP 白名单过滤

说明：防止非法用户登录，需在 web server 侧添加 IP Filter 过滤源 IP 非法的请求。

安全：支持 IP Filter 实现 Web 白名单配置，配置项是“jobmanager.web.allow-access-address”，默认情况下只支持 YARN 用户接入。

说明

安装客户端之后需要将客户端节点 IP 追加到 jobmanager.web.allow-access-address 配置项中。

禁止将文件绝对路径发送到客户端

说明：文件绝对路径发送到客户端会暴露服务端的目录结构信息，有助于攻击者遍历了解系统，为攻击者攻击提供帮助。

安全加固：Flink 配置文件中所有配置项中如果包含以/开头的，则删掉第一级目录。

同源策略

同源策略适用于 MRS 3.x 及之后版本。

如果两个 URL 的协议，主机和端口均相同，则它们同源；如果不同源，默认不能相互访问；除非被访问者在其服务端显示指定访问者的来源。

安全加固：响应头“Access-Control-Allow-Origin”头域默认配置为 YARN 集群 ResourceManager 的 IP 地址，如果源不是来自 YARN 的，则不能互相访问。

防范敏感信息泄露

防范敏感信息泄露适用于 MRS 3.x 及之后版本。

带有敏感数据的 Web 页面都应该禁止缓存，以防止敏感信息泄漏或通过代理服务器上网的用户数据互窜现象。

安全加固：添加“Cache-control”、“Pragma”、“Expires”安全头域，默认值为：“Cache-Control：no-store”，“Pragma ：no-cache”，“Expires ： 0”。

实现了安全加固，Flink 和 web server 交互的内容将不会被缓存。

防止劫持

防止劫持适用于 MRS 3.x 及之后版本。

由于点击劫持（ClickJacking）和框架盗链都利用到框架技术，所以需要采用安全措施。

安全加固：添加“X-Frame-Options”安全头域，给浏览器提供允许一个页面可否在“iframe”、“frame”或“object”网站中的展现页面的指示，如果默认配置为“X-Frame-Options: DENY”，则确保任何页面都不能被嵌入到别的“iframe”、“frame”或“object”网站中，从而避免了点击劫持 (clickjacking) 的攻击。

对 Web Service 接口调用记录日志

对 Web Service 接口调用记录日志适用于 MRS 3.x 及之后版本。

对“Flink webmonitor restful”接口调用进行日志记录。

安全加固：“access log”支持配置：“jobmanager.web.accesslog.enable”，默认为“true”。且日志保存在单独的“webaccess.log”文件中。

跨站请求（CSRF）伪造防范

跨站请求（CSRF）伪造防范适用于 MRS 3.x 及之后版本。

在 B/S 应用中，对于涉及服务器端数据改动（如增加、修改、删除）的操作必须进行跨站请求伪造的防范。跨站请求伪造是一种挟制终端用户在当前已登录的 Web 应用程序上执行非本意的操作的攻击方法。

安全加固：现有请求修改的接口有 2 个 post，1 个 delete，其余均是 get 请求，非 get 请求的接口均已删除。

异常处理

异常处理适用于 MRS 3.x 及之后版本。

应用程序出现异常时，捕获异常，过滤返回给客户端的信息，并在日志中记录详细的错误信息。

安全加固：

- 默认的错误提示页面，进行信息过滤，并在日志中记录详细的错误信息。
- 新加四个配置项，默认配置为 FusionInsight 提供的跳转 URL，错误提示页面跳转到固定配置的 URL 中，防止暴露不必要的信息。

表6-25 四个配置项参数介绍

参数	描述	默认值	是否必选配置
jobmanager.web.403-redirect-url	web403 页面，访问若遇到 403 错误，则会重定向到配置的页面。	-	是
jobmanager.web.404-redirect-url	web404 页面，访问若遇到 404 错误，则会重定向到配置的页面。	-	是
jobmanager.web.	web415 页面，访问若遇到 415 错误，则会重	-	是

参数	描述	默认值	是否必选配置
415-redirect-url	定向到配置的页面。		
jobmanager.web.500-redirect-url	web500 页面，访问若遇到 500 错误，则会重定向到配置的页面。	-	是

HTML5 安全

HTML5 安全适用于 MRS 3.x 及之后版本。

HTML5 是下一代的 Web 开发规范，为开发者提供了许多新的功能并扩展了标签。这些新的标签及功能增加了攻击面，存在被攻击的风险（例如跨域资源共享、客户端存储、WebWorker、WebRTC、WebSocket 等）。

安全加固：添加“Access-Control-Allow-Origin”配置，如运用到跨域资源共享功能，可对 HTTP 响应头的“Access-Control-Allow-Origin”属性进行控制。

说明

Flink 不涉及如客户端存储、WebWorker、WebRTC、WebSocket 等安全风险。

6.6 安全声明

- Flink 的安全都为开源社区提供和自身研发。有些是需要用户自行配置的安全特性，如认证、SSL 传输加密等，这些特性可能对性能和使用方便性造成一定影响。
- Flink 作为大数据计算和分析平台，对客户输入的数据是否包含敏感信息无法感知，因此需要客户保证输入数据是脱敏的。
- 客户可以根据应用环境，权衡配置安全与否。
- 任何与安全有关的问题，请联系运维人员。

6.7 使用 Flink WebUI

6.7.1 概述

6.7.1.1 Flink WebUI 应用简介

Flink WebUI 提供基于 Web 的可视化开发平台，用户只需要编写 SQL 即可开发作业，极大降低作业开发门槛。同时通过作业平台能力开放，支持业务人员自行编写 SQL 开发作业来快速应对需求，大大减少 Flink 作业开发工作量。

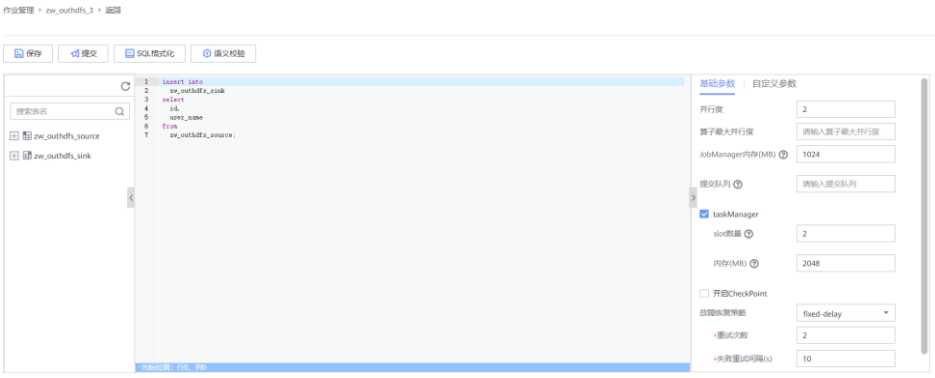
说明

Flink WebUI 功能仅支持 MRS 3.1.0 及之后版本。

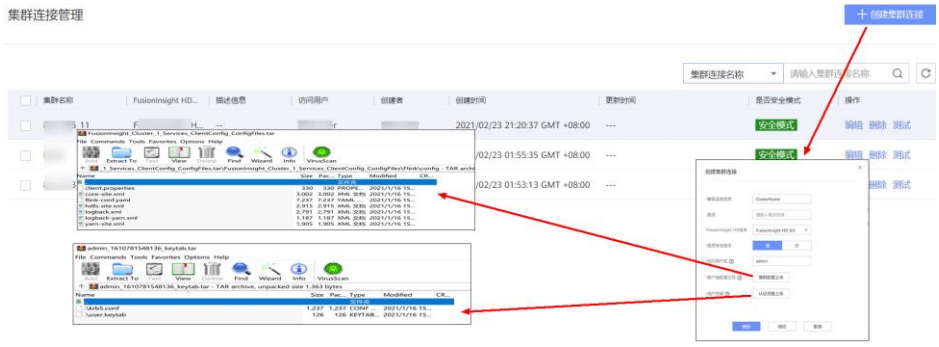
Flink WebUI 特点

Flink WebUI 主要有以下特点：

- 企业级可视化运维：运维管理界面化、作业监控、作业开发 Flink SQL 标准化等。



- 快速建立集群连接：通过集群连接功能配置访问一个集群，需要客户端配置、用户认证密钥文件。



- 快速建立数据连接：通过数据连接功能配置访问一个组件。创建“数据连接类型”为“HDFS”类型时需创建集群连接，其他数据连接类型的“认证类型”为“KERBEROS”需创建集群连接，“认证类型”为“SIMPLE”不需创建集群连接。

说明

“数据连接类型”为“Kafka”时，认证类型不支持“KERBEROS”。

- 可视化开发平台：支持自定义输入/输出映射表，满足不同输入来源、不同输出目标端的需求。
- 图形化作业管理：简单易用。

Flink WebUI 关键能力

FlinkWebUI 关键能力如表 6-26：

表6-26 Flink WebUI 关键能力

关键能力分类	描述
批流一体	• 支持一套 Flink SQL 定义批作业和流作业。
Flink SQL 内核能力	• Flink SQL 支持自定义大小窗、24 小时以内流计算、超出 24 小时批处理。 • Flink SQL 支持 Kafka、HDFS 读取；支持写入 Kafka 和 HDFS。 • 支持同一个作业定义多个 Flink SQL，多个指标合并在一个作业计算。当一个作业是相同主键、相同的输入和输出时，该作业支持多个窗口的计算。 • 支持 AVG、SUM、COUNT、MAX 和 MIN 统计方法。
Flink SQL 可视化定义	• 集群连接管理，配置 Kafka、HDFS 等服务所属的集群信息。 • 数据连接管理，配置 Kafka、HDFS 等服务信息。 • 数据表管理，定义 Sql 访问的数据表信息，用于生成 DDL 语句。 • Flink SQL 作业定义，根据用户输入的 Sql，校验、解析、优化、转换成 Flink 作业并提交运行。
Flink 作业可视化管理	• 支持可视化定义流作业和批作业。 • 支持作业资源、故障恢复策略、Checkpoint 策略可视化配置。 • 流作业和批作业的状态监控。 • Flink 作业运维能力增强，包括原生监控页面跳转。
性能&可靠性	• 流处理支持 24 小时窗口聚合计算，毫秒级性能。 • 批处理支持 90 天窗口聚合计算，分钟级计算完成。 • 支持对流处理和批处理的数据进行过滤配置，过滤无效数据。 • 读取 HDFS 数据时，提前根据计算周期过滤。 • 作业定义平台故障、服务降级，不支持再定义作业，但是不影响已有作业计算。 • 作业故障有自动重启机制，重启策略可配置。

6.7.1.2 Flink WebUI 应用流程

Flink WebUI 应用流程参考如下步骤：

图6-9 Flink WebUI 应用流程

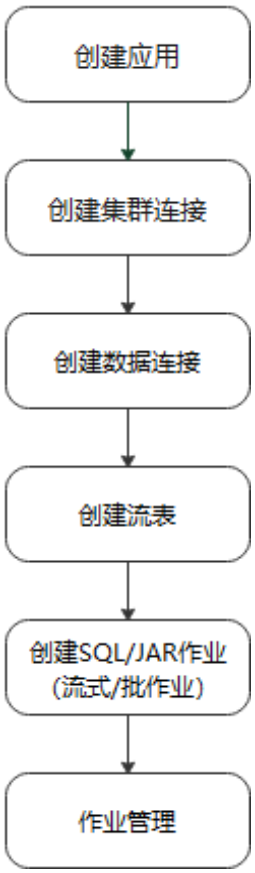


表6-27 Flink WebUI 应用流程说明

阶段	说明	参考章节
创建应用	通过应用来隔离不同的上层业务。	在 Flink WebUI 创建应用
创建集群连接	通过集群连接配置访问不同的集群。	在 Flink WebUI 创建集群连接
创建数据连接	通过数据连接，访问不同的数据服务，包括 HDFS、Kafka 等。	在 Flink WebUI 创建数据连接
创建流表	通过数据表，定义源表、维表、输出表的基本属性和字段信息。	使用 Flink WebUI 的流表管理
创建 SQL/JAR 作业（流式/批作业）	定义 Flink 作业的 API，包括 Flink SQL 和 Flink Jar 作业。	使用 Flink WebUI 的作业管理
作业管理	管理创建的作业，包括作业启动、开发、停止、删除和编辑等。	使用 Flink WebUI 的作业管理

6.7.2 FlinkServer 权限管理

6.7.2.1 概述

Manager 的 **admin** 用户没有 FlinkServer 的业务操作权限，使用 FlinkServer 的业务操作需要给用户赋予相关权限。

FlinkServer 中应用（租户）是最大管理范围，包含集群连接管理、数据连接管理、应用管理、流表和作业管理等。

FlinkServer 中有如表 6-28 所示三种资源权限：

表6-28 FlinkServer 资源权限

权限名称	权限描述	备注
管理员权限	具有所有应用的编辑、查看权限。	是 FlinkServer 的最高权限。如果已经具有管理员权限，则会自动具备所有应用的权限。
应用编辑权限	具有当前应用编辑权限的用户，可以执行创建、编辑和删除集群连接、数据连接，创建流表、创建作业及运行作业等操作。	同时具有当前应用查看权限。
应用查看权限	具有当前应用查看权限的用户，可以查看应用。	-

6.7.2.2 基于用户和角色的鉴权

该任务指导系统管理员在 Manager 创建并设置 FlinkServer 的角色。FlinkServer 角色可设置管理员权限以及应用的编辑和查看权限。

用户需要在 FlinkServer 中对指定的用户设置权限，才能够更新数据、查询数据和删除数据等。

前提条件

管理员已根据业务需要规划权限。

操作步骤

步骤 1 登录 Manager。

步骤 2 选择“系统 > 权限 > 角色”。

步骤 3 单击“添加角色”，然后在“角色名称”和“描述”输入角色名字与描述。

步骤 4 设置角色“配置资源权限”。

FlinkServer 权限类型：

- FlinkServer 管理员权限：是最高权限，具有 FlinkServer 所有应用的业务操作权限。
- FlinkServer 应用权限：可设置对应用的“应用查看”、“应用编辑”权限。

表6-29 设置角色

任务场景	角色授权操作
设置管理员权限	在“配置资源权限”的表格中选择“*待操作集群的名称* > Flink”，勾选“FlinkServer 管理操作权限”。
设置用户对应用的指定权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > Flink > FlinkServer 应用”。 2. 在“权限”列，勾选“应用查看”或“应用编辑”。

步骤 5 单击“确定”完成，返回角色管理。

说明

FlinkServer 角色创建成功后，可参考“用户指南 > FusionInsight Manager 操作指导 > 系统设置> 权限设置> 用户管理> 创建用户”章节创建一个 FlinkServer 用户并绑定角色和用户组。

----结束

6.7.3 访问 Flink WebUI

操作场景

MRS 集群安装 Flink 组件后，用户可以通过 Flink 的 WebUI，在图形化界面进行集群连接、数据连接、流表管理和作业管理等。

该任务指导用户在 MRS 集群中访问 Flink WebUI。

说明

Internet Explorer 浏览器可能存在兼容性问题，建议使用 Google Chrome 浏览器 50 及以上版本访问 Flink WebUI。

对系统的影响

第一次访问 Manager 和 Flink WebUI，需要在浏览器中添加站点信任以继续访问 Flink WebUI。

操作步骤

步骤 1 使用具有 FlinkServer 管理员权限的用户登录 FusionInsight Manager，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本），选择“集群 > 服务 > Flink”。

步骤 2 在“Flink WebUI”右侧，单击链接，访问 Flink 的 WebUI。

Flink WebUI 支持以下功能：

- 使用系统管理可以支持以下功能：
  - 使用集群连接管理可以创建、查看、编辑、测试和删除集群连接。
  - 使用数据连接管理可以创建、查看、编辑、测试和删除数据连接。数据连接类型包含 HDFS、Kafka 等。
  - 使用应用管理可以创建、查看、删除应用。
- 使用流表管理可以新建、查看、编辑和删除流表。
- 使用作业管理可以新建、查看、启动、开发、编辑、停止和删除作业等。

----结束

6.7.4 在 Flink WebUI 创建应用

操作场景

通过应用来隔离不同的上层业务。

创建应用

步骤 1 使用具有 FlinkServer 管理员权限的用户访问 Flink WebUI，请参考访问 Flink WebUI。

步骤 2 选择“系统管理 > 应用管理”，进入应用管理页面。

步骤 3 单击“创建应用”，在弹出的页面中参考表 6-30 填写信息，单击“确定”，完成应用创建。

表6-30 创建应用信息

参数名称	参数描述
应用名称	应用名称。只能包含英文字母、数字和下划线，且不能多于 32 个字符。
描述信息	应用描述信息。不能多于 85 个字符。

应用创建成功后，在 Flink WebUI 左上角即可切换待操作的应用，然后进行相关的作业开发。

----结束

6.7.5 在 Flink WebUI 创建集群连接

操作场景

通过集群连接配置访问不同的集群。

创建集群连接

步骤 1 访问 Flink WebUI，请参考访问 Flink WebUI。

步骤 2 选择“系统管理 > 集群连接管理”，进入集群连接管理页面。

步骤 3 单击“创建集群连接”，在弹出的页面中参考表 6-31 填写信息，单击“确定”，完成集群连接创建。

表6-31 创建集群连接信息

参数名称	参数描述
集群连接名称	集群连接的名称，只能包含英文字母、数字和下划线，且不能多于 100 个字符。
描述	集群连接名称描述信息。
版本	选择集群版本。
是否安全版本	• 是，安全集群选择是。需要输入访问用户名和上传用户凭证； • 否，非安全集群选择否。
访问用户名	访问用户需要包含访问集群中服务所需要的最小权限。只能包含英文字母、数字和下划线，且不能多于 100 个字符。 “是否安全版本”选择“是”时存在此参数。
客户端配置文件	集群客户端配置文件，格式为 tar。
用户凭据	FusionInsight Manager 中用户的认证凭据，格式为 tar。 “是否安全版本”选择“是”时存在此参数。 输入访问用户名后才可上传文件。

说明

集群客户端配置文件获取方法：

1. 登录 FusionInsight Manager，选择“集群 > 概览”。
2. 选择“更多 > 下载客户端 > 仅配置文件”，选择平台类型后单击“确定”。

用户凭据获取方法：

1. 登录 FusionInsight Manager，单击“系统”。
2. 在对应用户的“操作”列，选择“更多 > 下载认证凭据”，选择集群后单击“确定”。

----结束

编辑集群连接

步骤 1 访问 Flink WebUI，请参考访问 Flink WebUI。

步骤 2 选择“系统管理 > 集群连接管理”，进入集群连接管理页面。

步骤 3 在待修改项的“操作”列单击“编辑”，在弹出的页面中参考表 6-31 修改连接信息，单击“确定”完成修改。

----结束

测试集群连接

步骤 1 访问 Flink WebUI，请参考访问 Flink WebUI。

步骤 2 选择“系统管理 > 集群连接管理”，进入集群连接管理页面。

步骤 3 在待测试项的“操作”列单击“测试”进行测试。

----结束

搜索集群连接

步骤 1 访问 Flink WebUI，请参考访问 Flink WebUI。

步骤 2 选择“系统管理 > 集群连接管理”，进入集群连接管理页面。

步骤 3 在页面右上角，用户可以根据“集群连接名称”，输入查询条件后进行搜索和查看集群连接。

----结束

删除集群连接

步骤 1 访问 Flink WebUI，请参考访问 Flink WebUI。

步骤 2 选择“系统管理 > 集群连接管理”，进入集群连接管理页面。

步骤 3 在待删除项的“操作”列单击“删除”，在弹出的页面中单击“确定”完成删除。

----结束

6.7.6 在 Flink WebUI 创建数据连接

操作场景

通过数据连接，访问不同的数据服务，当前 FlinkServer 支持 HDFS、Kafka 类型的数据连接。

创建数据连接

步骤 1 访问 Flink WebUI，请参考访问 Flink WebUI。

步骤 2 选择“系统管理 > 数据连接管理”，进入数据连接管理页面。

步骤 3 单击“创建数据连接”，在弹出的页面中选择数据连接类型，参考表 6-32 填写信息，单击“确定”，完成数据连接创建。

表6-32 创建数据连接信息

参数名称	参数描述	示例
数据连接类型	选择数据连接的类型，包含 HDFS、Kafka。	-
数据连接名称	数据连接的名称。只能包含英文字母、数字和下划线，且不能多于 100 个字符。	-
集群连接	配置管理里的集群连接名称。	-
Kafka broker	Kafka Broker 实例的连接信息，格式为“IP 地址:端口”，多个实例之间通过逗号分割。 Kafka 类型数据连接需配置该参数。	192.168.0.1:21005,192.168.0.2:21005
认证类型	• SIMPLE：表示对接的服务是非安全模式，无需认证。 • KERBEROS：表示对接的服务是安全模式，安全模式的服务统一使用 Kerberos 认证协议进行安全认证。	-

----结束

编辑数据连接

步骤 1 访问 Flink WebUI，请参考访问 Flink WebUI。

步骤 2 选择“系统管理 > 数据连接管理”，进入数据连接管理页面。

步骤 3 在待修改项的“操作”列单击“编辑”，在弹出的页面中参考表 6-32 修改连接信息，单击“确定”完成修改。

----结束

测试数据连接

步骤 1 访问 Flink WebUI，请参考访问 Flink WebUI。

步骤 2 选择“系统管理 > 数据连接管理”，进入数据连接管理页面。

步骤 3 在待测试项的“操作”列单击“测试”进行测试。

----结束

搜索数据连接

步骤 1 访问 Flink WebUI，请参考访问 Flink WebUI。

步骤 2 选择“系统管理 > 数据连接管理”，进入数据连接管理页面。

步骤 3 在页面右上角，用户可以根据“名称”进行搜索查看数据连接。

----结束

删除数据连接

步骤 1 访问 Flink WebUI，请参考访问 Flink WebUI。

步骤 2 选择“系统管理 > 数据连接管理”，进入数据连接管理页面。

步骤 3 在待删除项的“操作”列单击“删除”，在弹出的页面中单击“确定”完成删除。

----结束

6.7.7 使用 Flink WebUI 的流表管理

操作场景

通过数据表，定义源表、维表、输出表的基本属性和字段信息。

新建流表

步骤 1 访问 Flink WebUI，请参考访问 Flink WebUI。

步骤 2 单击“流表管理”进入流表管理页面。

步骤 3 单击“新建流表”，在新建流表页面参考表 6-33 填写信息，单击“确定”，完成流表创建。

表6-33 新建流表信息

参数名称	参数描述	备注
流/表名称	流/表的名称，只能包含英文字母、数字和下划线，且长度为 1～64 个字符。	例如：flink_sink
描述	流/表的描述信息，且长度为 1～1024 个字符。	-
映射表类型	Flink SQL 本身不带有数据存储功能，所有涉及表创建的操作，实际上均是对于外部数据表、存储的引用映射。 类型包含 Kafka、HDFS。	-
类型	包含数据源表 Source，数据结果表 Sink。不同映射表类型包含的表如下所示。 • Kafka：Source、Sink	-

参数名称	参数描述	备注
	• HDFS：Source、Sink	
数据连接	选择数据连接。	-
Topic	读取的 Kafka 的 topic，支持从多个 Kakfa topic 中读取，topic 之间使用英文分隔符进行分隔。 “映射表类型”选择“Kafka”时存在此参数。	-
文件路径	要传输的 HDFS 目录或单个文件路径。 “映射表类型”选择“HDFS”时存在此参数。	例如： “/user/sqoop/ ”或“/user/sqoop/example.csv”
编码	选择不同“映射表类型”对应的编码如下： • Kafka：CSV、JSON • HDFS：CSV	-
前缀	“映射表类型”选择“Kafka”，且“类型”选择“Source”，“编码”选择“JSON”时含义为：多层嵌套 json 的层级前缀，使用英文逗号(,)进行分隔。	例如：data,info 表示取嵌套 json 中 data，info 下的内容，作为 json 格式数据输入
分隔符	选择不同“映射表类型”对应的含义为：用于指定 CSV 字段分隔符。当数据“编码”为“CSV”时存在此参数。	例如：“,”
行分隔符	文件中的换行符，包含“\r”、“\n”、“\r\n”。 “映射表类型”选择“HDFS”时存在此参数。	-
列分隔符	文件中的字段分隔符。 “映射表类型”选择“HDFS”时存在此参数。	例如：“,”
流/表结构	填写流/表结构，包含名称，类型。	-
Proctime	指系统时间，与数据本身的时间戳无关，即在 Flink 算子内计算完成的时间。 “类型”选择“Source”时存在此参数。	-
Event Time	指事件产生的时间，即数据产生时自带时间戳。 “类型”选择“Source”时存在此参数。	-

----结束

编辑流表

步骤 1 访问 Flink WebUI，请参考访问 Flink WebUI。

步骤 2 单击“流表管理”进入流表管理页面。

步骤 3 在待修改项的“操作”列单击“编辑”，在弹出的页面中参考表 6-33 修改流表信息，单击“确定”完成修改。

----结束

搜索流表

步骤 1 访问 Flink WebUI，请参考访问 Flink WebUI。

步骤 2 单击“流表管理”进入流表管理页面。

步骤 3 在页面右上角，用户可以输入关键字搜索查看流表信息。

----结束

删除流表

步骤 1 访问 Flink WebUI，请参考访问 Flink WebUI。

步骤 2 单击“流表管理”进入流表管理页面。

步骤 3 在待删除项的“操作”列单击“删除”，在弹出的页面单击“确定”完成删除。

----结束

6.7.8 使用 Flink WebUI 的作业管理

操作场景

定义 Flink 的作业，包括 Flink SQL 和 Flink Jar 作业。

新建作业

步骤 1 访问 Flink WebUI，请参考访问 Flink WebUI。

步骤 2 单击“作业管理”进入作业管理页面。

步骤 3 单击“新建作业”，在新建作业页面参考表 6-34 填写信息，单击“确定”，创建作业成功并进入作业开发界面。

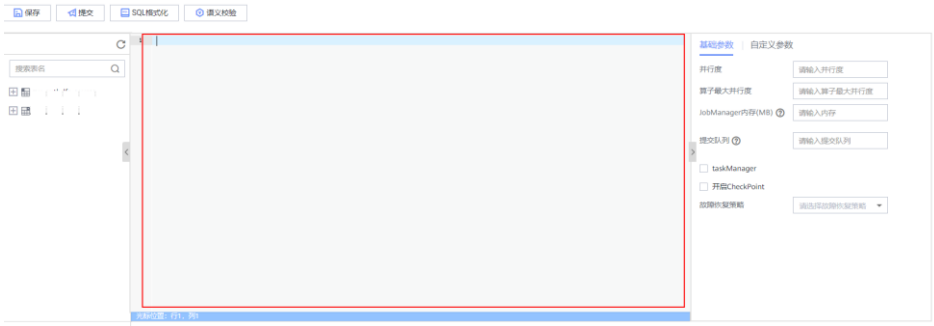
表6-34 新建作业信息

参数名称	参数描述
类型	作业类型，包括 Flink SQL 和 Flink Jar。
名称	作业名称，只能包含英文字母、数字和下划线，且不能多于

参数名称	参数描述
	64 个字符。
作业类型	作业数据来源类型，包括流作业和批作业。
描述	作业描述，不能超过 100 个字符。

步骤 4 （可选）如果需要立即进行作业开发，可以在作业开发界面进行作业配置。

- 新建 Flink SQL 作业
  a. 在作业开发界面进行作业开发。



  b. 可以单击上方“语义校验”对输入内容校验，单击“SQL 格式化”对 SQL 语句进行格式化。
  c. 作业 SQL 开发完成后，请参考表 6-35 设置基础参数，还可根据需要设置自定义参数，然后单击“保存”。

表6-35 基础参数

参数名称	参数描述
并行度	并行数量，只能填写正整数，且不能多于 64 字符。
算子最大并行度	算子最大的并行度，只能填写正整数，且不能多于 64 字符。
JobManager 内存（MB）	JobManager 的内存。输入值最小为 512，且不能超过 64 个字符。
提交队列	作业提交队列。不填默认提交到 default。只能包含英文字母，数字和下划线，且不能超过 30 字符。
taskManager	taskManager 运行参数。该参数需配置以下内容： • slot 数量：不填默认是 1； • 内存（MB）：输入值最小为 512。
开启 CheckPoint	是否开启 CheckPoint。开启后，需配置以下内容： • 时间间隔（ms）：必填； • 模式：必填； 可选项为：EXACTLY_ONCE、AT_LEAST_ONCE；

参数名称	参数描述
	• 最小间隔（ms）：输入值最小为 10； • 超时时间：输入值最小为 10； • 最大并发量：正整数，且不能超过 64 个字符； • 是否清理：是/否； • 是否开启增量 Checkpoint：是/否。
故障恢复策略	作业的故障恢复策略，包含以下三种。 • fixed-delay：需配置“重试次数”和“失败重试间隔（s）”； • failure-rate：需配置“最大重试次数”、“时间间隔（min）”和“失败重试间隔（s）”； • none：无。

d. 单击左上角“提交”提交作业。

- 新建 Flink Jar 作业

a. 单击“选择”，上传本地 Jar 文件，并参考表 6-36 配置参数或添加自定义参数。

图6-10 新建 Flink Jar 作业

表6-36 参数配置

参数名称	参数描述
本地 jar 文件	上传 jar 文件。直接上传本地文件，大小不能超过 10M。
Main Class	Main-Class 类型。 • 默认：默认根据 Jar 包文件的 Mainfest 文件指定类名。 • 指定：手动指定类名。

参数名称	参数描述
类名	类名。 “Main Class”选择“指定”时存在该参数。
类参数	类参数，为 Main-Class 的参数（参数间用空格分隔）。
并行度	并行数量，只能填写正整数，且不能多于 64 字符。
JobManager 内存（MB）	JobManager 的内存。输入值最小为 512，且不能超过 64 个字符。
提交队列	作业提交队列。不填默认提交到 default。只能包含英文字母，数字和下划线，且不能超过 30 字符。
taskManager	taskManager 运行参数。该参数需配置以下内容： • slot 数量：不填默认是 1； • 内存（MB）：输入值最小为 512。

b. 单击“保存”保存配置，单击“提交”提交作业。

步骤 5 返回作业管理页面，可以查看到已创建的作业名称、类型、状态、作业种类和描述等信息。

说明

若要使用其他用户在节点上读取已提交的作业相关文件，需确保该用户与提交作业的用户具有相同的用户组和具有对应的 FlinkServer 应用管理权限角色，如参考基于用户和角色的鉴权勾选“应用查看”。

----结束

启动作业

步骤 1 访问 Flink WebUI，请参考访问 Flink WebUI。

步骤 2 单击“作业管理”进入作业管理页面。

步骤 3 在待启动项的“操作”列单击“启动”运行作业。作业状态为“草稿”、“保存”、“提交失败”、“运行成功”、“运行失败”和“停止”的作业可以启动。

----结束

开发作业

步骤 1 访问 Flink WebUI，请参考访问 Flink WebUI。

步骤 2 单击“作业管理”进入作业管理页面。

步骤 3 在待开发项的“操作”列单击“开发”进入作业开发页面，参考步骤 4 进行作业开发，在左侧列表可以查看已创建的流表及字段。

----结束

编辑作业名称和描述

步骤 1 访问 Flink WebUI，请参考访问 Flink WebUI。

步骤 2 单击“作业管理”进入作业管理页面。

步骤 3 在待修改项的“操作”列单击“编辑”，修改“描述”，修改完成后单击“确定”保存修改。

----结束

查看作业详情

步骤 1 访问 Flink WebUI，请参考访问 Flink WebUI。

步骤 2 单击“作业管理”进入作业管理页面。

步骤 3 在待查看项的“操作”列选择“更多 > 作业详情”可以查看作业运行详情。

说明

只能查看状态为“运行中”的作业详情。

----结束

Checkpoint 故障恢复

步骤 1 访问 Flink WebUI，请参考访问 Flink WebUI。

步骤 2 单击“作业管理”进入作业管理页面。

步骤 3 在待恢复项的“操作”列选择“更多 > Checkpoint 故障恢复”进行 Checkpoint 故障恢复。作业状态为“运行失败”、“运行成功”和“停止”的作业可以进行 Checkpoint 故障恢复。

----结束

筛选/搜索作业

步骤 1 访问 Flink WebUI，请参考访问 Flink WebUI。

步骤 2 单击“作业管理”进入作业管理页面。

步骤 3 在页面右上角，用户可以根据作业名称进行筛选，或输入关键字搜索查看作业信息。

----结束

停止作业

步骤 1 访问 Flink WebUI，请参考访问 Flink WebUI。

步骤 2 单击“作业管理”进入作业管理页面。

步骤 3 在待停止项的“操作”列单击“停止”，停止作业运行。作业状态为“提交中”、“提交成功”和“运行中”的作业可以停止。

----结束

删除作业

步骤 1 访问 Flink WebUI，请参考访问 Flink WebUI。

步骤 2 单击“作业管理”进入作业管理页面。

步骤 3 在待删除项的“操作”列单击“删除”在弹出的页面单击“确定”删除作业。作业状态为“草稿”、“保存”、“提交失败”、“运行成功”、“运行失败”和“停止”状态的作业可以删除。

----结束

6.7.9 使用 Flink WebUI 管理 UDF

6.7.9.1 使用 Flink WebUI 管理 UDF

本章节适用于 MRS 3.1.2 及之后的版本。

用户可以自定义一些函数，用于扩展 SQL 以满足个性化的需求，这类函数称为 UDF。用户可以在 Flink WebUI 界面中上传并管理 UDF jar 包，然后在运行作业时调用相关 UDF 函数。

Flink 支持以下 3 类自定义函数，如表 6-37。

表6-37 函数分类

分类	描述
UDF（User Defined Scalar Function）	自定义函数，支持一个或多个输入参数，返回一个结果值。详情请参考 UDF java 代码及 SQL 样例。
UDAF（User Defined Aggregation Function）	自定义聚合函数，将多条记录聚合成一个值。详情请参考 UDAF java 代码及 SQL 样例。
UDTF（User Defined Table-valued Function）	自定义表值函数，支持一个或多个输入参数，可返回多行多列。详情请参考 UDTF java 代码及 SQL 样例。

前提条件

准备 UDF jar 文件，大小不能超过 200MB。

上传 UDF

步骤 1 访问 Flink WebUI，请参考访问 Flink WebUI。

步骤 2 单击“UDF 管理”进入 UDF 管理页面。

步骤 3 单击“添加 UDF”，在“本地 Jar 文件”参数后选择并上传本地已准备好的 UDF jar 文件。

步骤 4 填写 UDF 名称以及描述信息后，单击“确定”。

说明

“UDF 名称”最多可添加 10 项，“名称”可自定义，“类名”需与上传的 UDF jar 文件中 UDF 函数一一对应。

步骤 5 在 UDF 列表中，可查看当前应用内所有的 UDF 信息。

步骤 6 （可选）如果需要立即运行或开发作业，可在“作业管理”进行相关作业配置。

单击“作业管理”进入作业管理页面。

- 启动 UDF 作业：在 UDF 列表的“操作”列，单击“启动”。
- 开发 UDF 作业：在 UDF 列表的“操作”列，单击“开发”，相关参数设置可参考新建 Flink SQL 作业。
- 停止 UDF 作业：在 UDF 列表的“操作”列，单击“停止”。
- 删除 UDF 作业：在 UDF 列表的“操作”列，单击“删除”，仅可删除作业状态为“停止”的作业。
- 编辑 UDF 作业：在 UDF 列表的“操作”列，单击“编辑”，仅可修改作业“描述”。
- 查看作业详情：在 UDF 列表的“操作”列，选择“更多 > 作业详情”。
- Checkpoint 故障恢复：在待恢复项 UDF 列表的“操作”列选择“更多 > Checkpoint 故障恢复”进行 Checkpoint 故障恢复。作业状态为“运行失败”、“运行成功”和“停止”的作业可以进行 Checkpoint 故障恢复。

----结束

编辑 UDF

步骤 1 参考上传 UDF，上传 UDF jar 包。

步骤 2 在 UDF 列表的“操作”列，单击“编辑”，进入“编辑 UDF”页面。

步骤 3 修改信息，单击“确定”完成修改。

----结束

删除 UDF

步骤 1 参考上传 UDF，上传 UDF jar 包。

步骤 2 在 UDF 列表的“操作”列，单击“删除”，进入“删除 UDF”页面。

步骤 3 确认待删除的 UDF 信息，单击“确定”完成删除。

说明

只能删除未被使用的 UDF 项。

----结束

6.7.9.2 UDF java 代码及 SQL 样例

UDF java 使用样例

```
package com.xxx.udf;
import org.apache.flink.table.functions.ScalarFunction;
public class UdfClass_UDF extends ScalarFunction {
   public int eval(String s) {
      return s.length();
   }
}
```

UDF SQL 使用样例

```
CREATE TEMPORARY FUNCTION udf as 'com.xxx.udf.UdfClass_UDF';
CREATE TABLE udfSource (a VARCHAR) WITH ('connector' = 'datagen','rows-per-
second'='1');
CREATE TABLE udfSink (a VARCHAR,b int) WITH ('connector' = 'print');
INSERT INTO
  udfSink
SELECT
  a,
  udf(a)
FROM
  udfSource;
```

6.7.9.3 UDAF java 代码及 SQL 样例

UDAF java 使用样例

```
package com.xxx.udf;
import org.apache.flink.table.functions.AggregateFunction;
public class UdfClass_UDAF {
   public static class AverageAccumulator  {
      public int sum;
   }
   public static class Average extends AggregateFunction<Integer,
AverageAccumulator> {
      public void accumulate(AverageAccumulator acc, Integer value) {
         acc.sum += value;
      }
      @Override
      public Integer getValue(AverageAccumulator acc) {
         return acc.sum;
      }
      @Override
      public AverageAccumulator createAccumulator() {
         return new AverageAccumulator();
```

```
        }
    }
}
```

UDAF SQL 使用样例

```
CREATE TEMPORARY FUNCTION udaf as 'com.xxx.udf.UdfClass_UDAF$Average';
CREATE TABLE udfSource (a int) WITH ('connector' = 'datagen','rows-per-
second'='1','fields.a.min'='1','fields.a.max'='3');
CREATE TABLE udfSink (b int,c int) WITH ('connector' = 'print');
INSERT INTO
  udfSink
SELECT
  a,
  udaf(a)
FROM
  udfSource group by a;
```

6.7.9.4 UDTF java 代码及 SQL 样例

UDTF java 使用样例

```
package com.xxx.udf;
import org.apache.flink.api.java.tuple.Tuple2;
import org.apache.flink.table.functions.TableFunction;
public class UdfClass_UDTF extends TableFunction<Tuple2<String, Integer>> {
    public void eval(String str) {
        Tuple2<String, Integer> tuple2 = Tuple2.of(str, str.length());
        collect(tuple2);
    }
}
```

UDTF SQL 使用样例

```
CREATE TEMPORARY FUNCTION udtf as 'com.xxx.udf.UdfClass_UDTF';
CREATE TABLE udfSource (a VARCHAR) WITH ('connector' = 'datagen','rows-per-
second'='1');
CREATE TABLE udfSink (b VARCHAR,c int) WITH ('connector' = 'print');
INSERT INTO
  udfSink
SELECT
  str,
  strLength
FROM
  udfSource,lateral table(udtf(udfSource.a)) as T(str,strLength);
```

6.7.10 FlinkServer 对接外部组件

6.7.10.1 FlinkServer 对接 ClickHouse

本章节适用于 MRS 3.1.2 及之后的版本。

操作场景

Flink 通过对接 ClickHouse 的 ClickHouseBalancer 实例进行读写，有效避免 ClickHouse 流量分发问题。

前提条件

- 集群中已安装 ClickHouse、HDFS、Yarn、Flink 和 HBase 等服务。
- 客户端已安装，例如安装路径为：/opt/Bigdata/client。

FlinkSQL 与 ClickHouse 数据类型对应关系

FlinkSQL 数据类型	ClickHouse 数据类型
BOOLEAN	UInt8
TINYINT	Int8
SMALLINT	Int16
INTEGER	Int32
BIGINT	Int64
FLOAT	Float32
DOUBLE	Float64
CHAR	String
VARCHAR	String
VARBINARY	FixedString
DATE	Date
TIMESTAMP	DateTime
DECIMAL	Decimal

操作步骤

步骤 1 使用 **root** 用户登录安装客户端的节点。

步骤 2 执行以下命令，切换到客户端安装目录。

cd /opt/Bigdata/client

步骤 3 执行以下命令配置环境变量。

source bigdata_env

步骤 4 如果当前集群已启用 Kerberos 认证，执行以下命令认证当前用户，当前用户需要具有创建 ClickHouse 表的权限。如果当前集群未启用 Kerberos 认证，则无需执行此命令。

kinit *组件业务用户*

例如，**kinit clickhouseuser**。

步骤 5 连接 ClickHouse 客户端，可参考从零开始使用 ClickHouse。

- 普通模式：

  clickhouse client --host *ClickHouse 的实例 IP* **--user** *登录名* **--password** '密码' **--port** *ClickHouse 的端口号* **--multiline**

- 安全模式：

  clickhouse client --host *ClickHouse 的实例 IP* **--user** *登录名* **--password** '*密码*' **--port** *ClickHouse 的端口号* **--secure --multiline**

步骤 6 执行以下命令创建复制表和分布式表。

1. 创建复制表“default.test1”。

```
CREATE TABLE default.test1 on cluster default_cluster
(
`pid` Int8,
`uid` UInt8,
`Int_16` Int16,
`Int_32` Int32,
`Int_64` Int64,
`String_x` String,
`String_y` String,
`float_32` Float32,
`float_64` Float64,
`Decimal_x` Decimal32(2),
`Date_x` Date,
`DateTime_x` DateTime
)
ENGINE =
ReplicatedReplacingMergeTree('/clickhouse/tables/{shard}/test1','{replica}')
PARTITION BY pid
ORDER BY (pid, DateTime_x);
```

2. 创建分布式表“test1_all”。

```
CREATE TABLE test1_all ON CLUSTER default_cluster
(
`pid` Int8,
`uid` UInt8,
`Int_16` Int16,
`Int_32` Int32,
`Int_64` Int64,
`String_x` String,
`String_y` String,
`float_32` Float32,
`float_64` Float64,
`Decimal_x` Decimal32(2),
`Date_x` Date,
`DateTime_x` DateTime
)
ENGINE = Distributed(default_cluster, default, test1, rand());
```

步骤 7 登录 Manager，选择“集群 > 服务 > Flink”，在“Flink WebUI”右侧，单击链接，访问 Flink 的 WebUI。

步骤 8 参考新建作业，新建 Flink SQL 作业，作业类型选择“流作业”。在作业开发界面进行如下作业配置，并启动作业。需勾选“基础参数”中的“开启 CheckPoint”，“时间间隔（ms）”可设置为“60000”，“模式”可使用默认值。

- 如果当前 MRS 集群为安全模式，执行以下操作：

```
create table kafkasource(
`pid` TINYINT,
`uid` BOOLEAN,
`Int_16` SMALLINT,
`Int_32` INTEGER,
`Int_64` BIGINT,
`String_x` CHAR,
`String_y` VARCHAR(10),
`float_32` FLOAT,
`float_64` DOUBLE,
`Decimal_x` DECIMAL(9,2),
`Date_x` DATE,
`DateTime_x` TIMESTAMP
) with(
  'connector' = 'kafka',
  'topic' = 'input',
  'properties.bootstrap.servers' = 'Kafka 的 Broker 实例业务 IP:Kafka 端口号',
  'properties.group.id' = 'group1',
  'scan.startup.mode' = 'earliest-offset',
  'format' = 'json',
  'properties.sasl.kerberos.service.name' = 'kafka',
  'properties.security.protocol' = 'SASL_PLAINTEXT',
  'properties.kerberos.domain.name' = 'hadoop.hadoop.com'
);
CREATE TABLE cksink (
`pid` TINYINT,
`uid` BOOLEAN,
`Int_16` SMALLINT,
`Int_32` INTEGER,
`Int_64` BIGINT,
`String_x` CHAR,
`String_y` VARCHAR(10),
`float_32` FLOAT,
`float_64` DOUBLE,
`Decimal_x` DECIMAL(9,2),
`Date_x` DATE,
`DateTime_x` TIMESTAMP
) WITH (
'connector' = 'jdbc',
'url' = 'jdbc:clickhouse://ClickHouseBalancer 实例
IP:21422/default?ssl=true&sslmode=none',
'username' = 'ClickHouse 用户，详见说明',
'password' = 'ClickHouse 用户密码，详见说明',
'table-name' = 'test1_all',
'driver' = 'ru.yandex.clickhouse.ClickHouseDriver',
'sink.buffer-flush.max-rows' = '0',
'sink.buffer-flush.interval' = '60s'
);
Insert into cksink
select
```

```
*
from
kafkasource;
```

- 如果当前 MRS 集群为普通模式，执行以下操作：

```
create table kafkasource(
`pid` TINYINT,
`uid` BOOLEAN,
`Int_16` SMALLINT,
`Int_32` INTEGER,
`Int_64` BIGINT,
`String_x` CHAR,
`String_y` VARCHAR(10),
`float_32` FLOAT,
`float_64` DOUBLE,
`Decimal_x` DECIMAL(9,2),
`Date_x` DATE,
`DateTime_x` TIMESTAMP
) with(
  'connector' = 'kafka',
  'topic' = 'kinput',
  'properties.bootstrap.servers' = 'Kafka 的 Broker 实例业务 IP:Kafka 端口号',
  'properties.group.id' = 'kafka_test',
  'scan.startup.mode' = 'earliest-offset',
  'format' = 'json'
);
CREATE TABLE cksink (
`pid` TINYINT,
`uid` BOOLEAN,
`Int 16` SMALLINT,
`Int 32` INTEGER,
`Int 64` BIGINT,
`String x` CHAR,
`String y` VARCHAR(10),
`float_32` FLOAT,
`float_64` DOUBLE,
`Decimal_x` DECIMAL(9,2),
`Date_x` DATE,
`DateTime_x` TIMESTAMP
) WITH (
'connector' = 'jdbc',
'url' = 'jdbc:clickhouse://ClickHouseBalancer 实例 IP:21425/default',
'table-name' = 'test1_all',
'driver' = 'ru.yandex.clickhouse.ClickHouseDriver',
'sink.buffer-flush.max-rows' = '0',
'sink.buffer-flush.interval' = '60s'
);
Insert into cksink
select
*
from
kafkasource;
```

说明

- MRS 集群安全模式下，创建的 cksink 表中 username、password 参数填写的用户为具有 ClickHouse 相应表权限的用户及密码，详见 ClickHouse 用户及权限管理。
- Kafka 端口号：
- 集群的“认证模式”为“安全模式”时为“sasl.port”的值，默认为“21007”。
- 集群的“认证模式”为“普通模式”时为“port”的值，默认为“9092”。如果配置端口号为 9092，则需要配置“allow.everyone.if.no.acl.found”参数为 true，具体操作如下：

  登录 FusionInsight Manager 系统，选择“集群 > 服务 > Kafka > 配置 > 全部配置”，搜索“allow.everyone.if.no.acl.found”配置，修改参数值为 true，保存配置即可。
- 21422：ClickHouseBalancer 实例 IP 的 https 端口。
- 21425：ClickHouseBalancer 实例 IP 的 http 端口。
- 其他 jdbc connector 参数请参考 Flink 官网：http://flink.apache.org/
- 攒批写参数：Flink 会将数据先放入内存，到达触发条件时再 flush 到数据库表中。相关配置如下。

  sink.buffer-flush.max-rows：攒批写 ClickHouse 的行数，默认 100。

  sink.buffer-flush.interval：攒批写入的间隔时间，默认 1s。

  这两个条件只要有一个满足，就会触发一次 sink，即到达触发条件时再 flush 到数据库表中。
- 情况一：60s sink 一次

  'sink.buffer-flush.max-rows' = '0',
  'sink.buffer-flush.interval' = '60s'
- 情况二：100 条 sink 一次

  'sink.buffer-flush.max-rows' = '100',
  'sink.buffer-flush.interval' = '0s'
- 情况三：数据不 sink

  'sink.buffer-flush.max-rows' = '0',
  'sink.buffer-flush.interval' = '0s'

步骤 9 查看作业管理界面，作业状态为“运行中”。

步骤 10 参考管理 Kafka 主题中的消息，向 kafka 中写入数据。

sh kafka-console-producer.sh --broker-list *Kafka 角色实例所在节点的 IP 地址:Kafka 端口号* **--topic** *主题名称* **--producer.config** *客户端目录*
/Kafka/kafka/config/producer.properties

例如本示例使用主题名称为 kinput：**sh kafka-console-producer.sh --broker-list** *Kafka 角色实例所在节点的 IP 地址:Kafka 端口号* **--topic kinput --producer.config**
/opt/Bigdata/client/Kafka/kafka/config/producer.properties

输入消息内容：

```
{"pid": "3","uid":false,"Int_16": "6533","Int_32": "429496294","Int_64":
"1844674407370955614","String_x": "abc1","String_y": "abc1defghi","float_32":
"0.1234","float_64": "95.1","Decimal_x": "0.451236414","Date_x": "2021-05-
```

```
29","DateTime_x": "2021-05-21 10:05:10"}
{"pid": "4","uid":false,"Int_16": "6533","Int_32": "429496294","Int_64":
"1844674407370955614","String_x": "abc1","String_y": "abc1defghi","float_32":
"0.1234","float_64": "95.1","Decimal_x": "0.4512314","Date_x": "2021-05-
29","DateTime_x": "2021-05-21 10:05:10"}
```

输入完成后按回车发送消息。

步骤 11 连接 ClickHouse 查询表数据。

clickhouse client --host *ClickHouse 的实例 IP* **--user** *登录名* **--password** '*密码*'**--port** *ClickHouse 的端口号* **--secure --multiline**

执行查询命令查询 ClickHouse 表是否已写入数据。例如，当前 ClickHouse 表为 test1_all。

select * from test1_all;

----结束

6.7.10.2 FlinkServer 对接 HBase

本章节适用于 MRS 3.1.2 及之后的版本。

操作场景

FlinkServer 支持对接 HBase，详情如下：

- 支持对接维表、Sink 表。
- 当 HBase 与 Flink 为同一集群或互信的集群，支持 FlinkServer 对接 HBase。
- 当 HBase 与 Flink 不在同一集群或不互信的集群，则只支持 Flink 和 HBase 均为普通模式集群的对接。

前提条件

- 集群已安装，包括 HDFS、Yarn、Flink 和 HBase。
- 包含 HBase 服务的客户端已安装，安装路径如：/opt/Bigdata/client。
- 参考使用 HBase 客户端，登录 HBase 客户端，使用 **create 'dim_province', "f1"**创建 dim_province 表。

操作步骤

步骤 1 以客户端安装用户登录安装客户端的节点，拷贝 HBase 的“/opt/Bigdata/client/HBase/hbase/conf/”目录下的所有配置文件至部署 FlinkServer 的所有节点的一个空目录，如“/tmp/client/HBase/hbase/conf/”。

修改 FlinkServer 节点上面配置文件目录及其上层目录属主为 omm。

chown omm: /tmp/client/HBase/ -R

说明

- FlinkServer 节点：

登录 Manager，选择“集群 > 服务 > Flink > 实例”，查看 FlinkServer 所在的“业务 IP”。

- 若 FlinkServer 实例所在节点与包含 HBase 服务客户端的安装节点相同，则该节点不执行此步骤。

步骤 2 登录 Manager，选择“集群 > 服务 > Flink > 配置 > 全部配置”，搜索“HBASE_CONF_DIR”参数，在该参数的“值”中填写步骤 1 中拷贝了 HBase 配置文件的 FlinkServer 的目录，如“/tmp/client/HBase/hbase/conf/”。

说明

若 FlinkServer 实例所在节点与包含 HBase 服务客户端的安装节点相同，则在 HBASE_CONF_DIR”参数的“值”填写 HBase 的“/opt/Bigdata/client/HBase/hbase/conf/”目录。

步骤 3 填写完成后单击“保存”，确认修改配置后单击“确定”。

步骤 4 单击“实例”，勾选所有 FlinkServer 实例，选择“更多 > 重启实例”，输入密码，单击“确定”重启实例。

步骤 5 登录 Manager，选择“集群 > 服务 > Flink”，在“Flink WebUI”右侧，单击链接，访问 Flink 的 WebUI。

步骤 6 参考新建作业，新建 Flink SQL 作业，作业类型选择“流作业”。在作业开发界面进行如下作业配置并启动作业。

需勾选“基础参数”中的“开启 CheckPoint”，“时间间隔（ms）”可设置为“60000”，“模式”可使用默认值。

安全集群参考如下样例，建立 Flink SQL 作业。

```
CREATE TABLE ksource1 (
user_id STRING,
item_id STRING,
proctime as PROCTIME()
) WITH (
'connector' = 'kafka',
'topic' = 'ksource1',
'properties.group.id' = 'group1',
'properties.bootstrap.servers' = 'Kafka 的 Broker 实例业务 IP1:Kafka 端口号,Kafka 的 Broker 实例业务 IP2:Kafka 端口号',
'format' = 'json',
'properties.sasl.kerberos.service.name' = 'kafka',
'properties.security.protocol' = 'SASL_PLAINTEXT',
'properties.kerberos.domain.name' = 'hadoop.hadoop.com'
);

CREATE TABLE hsink1 (
rowkey STRING,
f1 ROW < item_id STRING >,
PRIMARY KEY (rowkey) NOT ENFORCED
) WITH (
'connector' = 'hbase-2.2',
'table-name' = 'dim_province',
'zookeeper.quorum' = 'ZooKeeper 的 quorumpeer 实例业务 IP1:ZooKeeper 客户端端口号,ZooKeeper 的 quorumpeer 实例业务 IP2:ZooKeeper 客户端端口号'
```

```
);

INSERT INTO
hsink1
SELECT
user_id as rowkey,
ROW(item_id) as f1
FROM
ksource1;
```

说明

- Kafka 端口号：
- 集群的“认证模式”为“安全模式”时为“sasl.port”的值，默认为“21007”。
- 集群的“认证模式”为“普通模式”时为“port”的值，默认为“9092”。如果配置端口号为9092，则需要配置“allow.everyone.if.no.acl.found”参数为 true，具体操作如下：

  登录 FusionInsight Manager 系统，选择“集群 > 服务 > Kafka > 配置 > 全部配置”，搜索“allow.everyone.if.no.acl.found”配置，修改参数值为 true，保存配置即可。
- ZooKeeper 的 quorumpeer 实例业务 IP：

  ZooKeeper 服务所有 quorumpeer 实例业务 IP。登录 FusionInsight Manager，选择“集群 > 服务 > ZooKeeper > 实例”，可查看所有 quorumpeer 实例所在主机业务 IP 地址。
- ZooKeeper 客户端端口号：

  登录 FusionInsight Manager，选择“集群 > 服务 > ZooKeeper”，在“配置”页签查看“clientPort”的值。默认为 24002。

步骤 7 查看作业管理界面，作业状态为“运行中”。

步骤 8 参考管理 Kafka 主题中的消息，向 kafka 中写入数据。

sh kafka-console-producer.sh --broker-list *Kafka 角色实例所在节点的 IP 地址:Kafka 端口号* **--topic** ***主题名称*** **--producer.config** *客户端目录* **/Kafka/kafka/config/producer.properties**

例如本示例使用主题名称为 ksource1：**sh kafka-console-producer.sh --broker-list** *Kafka 角色实例所在节点的 IP 地址:Kafka 端口号* **--topic ksource1 --producer.config /opt/Bigdata/client/Kafka/kafka/config/producer.properties**

输入消息内容：

```
{"user_id": "3","item_id":"333333"}
{"user_id": "4","item_id":"44444444"}
```

输入完成后按回车发送消息。

步骤 9 参考使用 HBase 客户端，登录 HBase 客户端，查看表数据信息。

hbase shell

scan 'dim_province'

----结束

应用端提交作业

- 若使用 Flink run 模式，推荐使用 export HBASE_CONF_DIR=hbase 的配置目录，例如：export HBASE_CONF_DIR=/opt/hbaseconf。
- 若使用 Flink run-application 模式，则有如下两种方式。
  - 在建表语句中添加如下配置（推荐）

配置	说明
'properties.hbase.rpc.protection' = 'authentication'	需和 HBase 服务端的配置一致
'properties.zookeeper.znode.parent' = '/hbase'	多服务场景中，会存在 hbase1，hbase2，需明确要访问的集群
'properties.hbase.security.authorization' = 'true'	开启鉴权
'properties.hbase.security.authentication' = 'kerberos'	开启 kerberos 认证

    示例：

```
CREATE TABLE hsink1 (
    rowkey STRING,
    f1 ROW < q1 STRING >,
    PRIMARY KEY (rowkey) NOT ENFORCED
   ) WITH (
     'connector' = 'hbase-2.2',
     'table-name' = 'cc',
     'zookeeper.quorum' = 'x.x.x.x:24002',
     'properties.hbase.rpc.protection' = 'authentication',
     'properties.zookeeper.znode.parent' = '/hbase',
     'properties.hbase.security.authorization' = 'true',
     'properties.hbase.security.authentication' = 'kerberos'
  );
```

  - 提交作业时将 HBase 的配置添加到 yarnShip 中。

    例如：Dyarn.ship-files=/opt/hbaseconf。

6.7.10.3 FlinkServer 对接 HDFS

本章节适用于 MRS 3.1.2 及之后的版本。

操作场景

本章节介绍 HDFS 作为 sink 表的 DDL 定义，以及创建 sink 表时使用的 WITH 参数和代码示例，并指导如何在 FlinkServer 作业管理页面操作。

本示例以安全模式 Kafka 为例。

前提条件

- 集群中已安装 HDFS、Yarn、Flink 服务。

- 包含 HDFS 服务的客户端已安装，安装路径如：/opt/Bigdata/client。
- 参考基于用户和角色的鉴权创建一个具有 FlinkServer 管理员权限的用户用于访问 Flink WebUI，如：flink_admin。

操作步骤

步骤 1 使用 **flink_admin** 登录 Manager，选择“集群 > 服务 > Flink”，在“Flink WebUI”右侧，单击链接，访问 Flink 的 WebUI。

步骤 2 参考新建作业，新建 Flink SQL 流作业，参考如下内容在作业开发界面进行作业开发，配置完成后启动作业。

需勾选“基础参数”中的“开启 CheckPoint”，“时间间隔（ms）”可设置为“60000”，“模式”可使用默认值。

```
CREATE TABLE kafka_table (
  user_id STRING,
  order_amount DOUBLE,
  log_ts TIMESTAMP(3),
  WATERMARK FOR log_ts AS log_ts - INTERVAL '5' SECOND
) WITH (
  'connector' = 'kafka',
  'topic' = 'user_source',
  'properties.bootstrap.servers' = 'Kafka 的 Broker 实例业务 IP:Kafka 端口号',
  'properties.group.id' = 'testGroup',
  'scan.startup.mode' = 'latest-offset',
  'format' = 'csv',
   --跳过解析失败的 csv 数据
  'csv.ignore-parse-errors' = 'true',--如果是 json 数据格式，设置'json.ignore-parse-errors' = 'true'
  'properties.sasl.kerberos.service.name' = 'kafka',
  'properties.security.protocol' = 'SASL_PLAINTEXT',
  'properties.kerberos.domain.name' = 'hadoop.hadoop.com'

);

CREATE TABLE fs_table (
  user_id STRING,
  order_amount DOUBLE,
  dt STRING,
  `hour` STRING
) PARTITIONED BY (dt, `hour`) WITH ( --根据日期进行文件分区
  'connector'='filesystem',
  'path'='hdfs:///sql/parquet',
  'format'='parquet',
  'sink.partition-commit.delay'='0 s',--该延迟时间之前分区不会被提交。如果是按天分区，可以设置为'1 d'，如果是按小时分区，应设置为'1 h'
  'sink.partition-commit.policy.kind'='success-file'
);
-- streaming sql, insert into file system table
INSERT INTO fs_table SELECT user_id, order_amount, DATE_FORMAT(log_ts, 'yyyy-MM-dd'), DATE_FORMAT(log_ts, 'HH') FROM kafka_table;
```

说明

Kafka 端口号：

- 集群的“认证模式”为“安全模式”时为“sasl.port”的值，默认为“21007”。
- 集群的“认证模式”为“普通模式”时为“port”的值，默认为“9092”。如果配置端口号为9092，则需要配置“allow.everyone.if.no.acl.found”参数为 true，具体操作如下：

  登录 FusionInsight Manager 系统，选择“集群 > 服务 > Kafka > 配置 > 全部配置”，搜索“allow.everyone.if.no.acl.found”配置，修改参数值为 true，保存配置即可。

步骤 3 查看作业管理界面，作业状态为“运行中”。

步骤 4 参考管理 Kafka 主题中的消息，查看 Topic 并向 Kafka 中写入数据。

./kafka-topics.sh --list --zookeeper *ZooKeeper 的 quorumpeer 实例业务 IP*:*ZooKeeper 客户端端口号***/kafka**

sh kafka-console-producer.sh --broker-list *Kafka 角色实例所在节点的 IP 地址*:*Kafka 端口号* **--topic** *主题名称* **--producer.config** *客户端目录***/Kafka/kafka/config/producer.properties**

例如本示例使用主题名称为 user_source：**sh kafka-console-producer.sh --broker-list** *Kafka 角色实例所在节点的 IP 地址*:*Kafka 端口号* **--topic user_source --producer.config /opt/Bigdata/client/Kafka/kafka/config/producer.properties**

输入消息内容：

```
3,3333,"2021-09-10 14:00"
4,4444,"2021-09-10 14:01"
```

输入完成后按回车发送消息。

说明

- ZooKeeper 的 quorumpeer 实例业务 IP：

  ZooKeeper 服务所有 quorumpeer 实例业务 IP。登录 FusionInsight Manager，选择“集群 > 服务 > ZooKeeper > 实例”，可查看所有 quorumpeer 实例所在主机业务 IP 地址。

- ZooKeeper 客户端端口号：

  登录 FusionInsight Manager，选择“集群 > 服务 > ZooKeeper”，在“配置”页签查看“clientPort”的值。默认为 24002。

步骤 5 执行以下命令查看 Sink 表中是否接收到数据，即 HDFS 目录是否正常写入文件。

hdfs dfs -ls -R /sql/parquet

----结束

Flink 对接 HDFS 分区

- Flink 对接 HDFS 支持自定义分区。

  Flink 文件系统分区支持使用标准的 Hive 格式。不需要将分区预先注册到表目录中，分区是根据目录结构推断。

例如，根据下面的目录分区的表将被推断为包含日期时间和小时分区。

```
path
└── datetime=2021-09-03
    └── hour=11
        ├── part-0.parquet
        ├── part-1.parquet
    └── hour=12
        ├── part-0.parquet
└── datetime=2021-09-24
    └── hour=6
        ├── part-0.parquet
```

- 分区文件的滚动策略。

  分区目录中的数据被拆分为 part 文件，每个分区将至少包含一个 part 文件，用于接收 sink 的子任务的数据写入。

  如下参数介绍分区文件如何进行滚动。

配置项	默认值	类型	描述
sink.rolling-policy.file-size	128MB	MemorySize	分区文件达到该阈值后，进行滚动。
sink.rolling-policy.rollover-interval	30min	Duration	分区文件在滚动前可以保持打开的最长持续时间。
sink.rolling-policy.check-interval	1min	Duration	检查基于时间的滚动策略的时间间隔。

- 分区目录的文件合并。

  支持文件压缩，允许应用程序具有更小的检查点间隔，而无需生成大量文件。

  说明

  仅压缩单个检查点中的文件，即生成的文件数量至少与检查点数量相同。合并前的文件是不可见的，因此文件的可见性是：检查点间隔+压缩时间之后。如果压缩时间太长，将延长检查点的时间段。

配置项	默认值	类型	描述
auto-compaction	false	Boolean	是否启用自动压缩。数据将写入临时文件。检查点完成后，检查点生成的临时文件将被压缩。压缩前临时文件不可见。
compaction.file-size	none	MemorySize	压缩目标文件大小，默认值为滚动文件大小。

- 分区文件的提交。

文件写入分区后，通常需要通知下游应用程序。如将分区添加到 Hive 元存储中，或在目录中写入_SUCCESS 文件。分区文件的提交操作基于触发器和策略的组合方式。

- 分区文件提交触发器相关配置

配置项	默认值	类型	描述
sink.partition-commit.trigger	process-time	String	• process-time：基于计算节点的系统时间，它既不需要分区时间提取，也不需要生成 watermark。即“当前系统时间”超过“分区创建时的系统时间”加上“延迟”时间，就提交分区。 • partition-time：基于从分区提取的时间，它需要生成 watermark。即“watermark 时间”超过“从分区提取的时间”加上“延迟”时间，就提交分区。
sink.partition-commit.delay	0 s	Duration	分区在延迟时间之前不会提交。如果是每日分区，则应为“1 d”，如果是每小时分区，则应为“1 h”。

- 分区问文件提交策略相关配置

配置项	默认值	类型	描述
sink.partition-commit.policy.kind	-	String	提交分区的策略。 • metastore：将分区添加到元存储。只有 hive 表支持元存储策略，文件系统通过目录结构管理分区。 • success-file：将 success-file 文件添加到目录中。 • 两者可以同时配置，即：'sink.partition-commit.policy.kind'='metastore,success-file'。
sink.partition-commit.policy.class	-	String	用于实现分区提交策略接口的分区提交策略类。 仅在自定义提交策略中生效。
sink.partition-commit.success-file.name	_SUCCESS	String	success-file 分区提交策略的文件名，默认值为_SUCCESS。

6.7.10.4 FlinkServer 对接 Hive

本章节适用于 MRS 3.1.2 及之后的版本。

操作场景

目前 FlinkServer 对接 Hive 使用对接 metaStore 的方式，所以需要 Hive 开启 MetaStore 功能。Hive 可以作为 source，sink 和维表。

本示例以安全模式 Kafka 为例。

前提条件

- 集群已安装 HDFS、Yarn、Kafka、Flink 和 Hive 等服务。
- 包含 Hive 服务的客户端已安装，安装路径如：/opt/Bigdata/client。
- Flink 支持 1.12.2 及以后版本，Hive 支持 3.1.0 及以后版本。
- 参考基于用户和角色的鉴权创建一个具有 FlinkServer 管理员权限的用户用于访问 Flink WebUI，如：flink_admin。
- 参考创建集群连接中的“说明”获取访问 Flink WebUI 用户的客户端配置文件及用户凭据。

操作步骤

以映射表类型为 Kafka 对接 Hive 流程为例。

步骤 1 使用 **flink_admin** 访问 Flink WebUI，请参考访问 Flink WebUI。

步骤 2 新建集群连接，如：flink_hive。

1. 选择“系统管理 > 集群连接管理”，进入集群连接管理页面。
2. 单击“创集集群连接”，在弹出的页面中参考表 6-38 填写信息，单击“测试”，测试连接成功后单击“确定”，完成集群连接创建。

表6-38 创建集群连接信息

参数名称	参数描述	取值样例
集群连接名称	集群连接的名称，只能包含英文字母、数字和下划线，且不能多于 100 个字符。	flink_hive
描述	集群连接名称描述信息。	-
版本	选择集群版本。	MRS 3
是否安全版本	• 是，安全集群选择是。需要输入访问用户名和上传用户凭证； • 否，非安全集群选择否。	是
访问用户名	访问用户需要包含访问集群中服务所需要的最小权限。只能包含英文字母、数字和下划线，且不能多于 100 个字符。 “是否安全版本”选择“是”时存在此参数。	flink_admin
客户端配置文件	集群客户端配置文件，格式为 tar。	-

参数名称	参数描述	取值样例
用户凭据	FusionInsight Manager 中用户的认证凭据，格式为 tar。 “是否安全版本”选择“是”时存在此参数。 输入访问用户名后才可上传文件。	flink_admin 的用户凭据

步骤 3 新建 Flink SQL 流作业，如：flinktest1。

1. 单击“作业管理”进入作业管理页面。
2. 单击“新建作业”，在新建作业页面参考表 6-39 填写信息，单击“确定”，创建作业成功并进入作业开发界面。

表6-39 新建作业信息

参数名称	参数描述	取值样例
类型	作业类型，包括 Flink SQL 和 Flink Jar。	Flink SQL
名称	作业名称，只能包含英文字母、数字和下划线，且不能多于 64 个字符。	flinktest1
作业类型	作业数据来源类型，包括流作业和批作业。	流作业
描述	作业描述，不能超过 100 个字符。	-

步骤 4 在作业开发界面进行作业开发，输入如下语句，可以单击上方“语义校验”对输入内容校验。

```
CREATE TABLE test kafka (
  user id varchar,
  item_id varchar,
  cat_id varchar,
  zw_test timestamp
) WITH (
  'properties.bootstrap.servers' = 'Kafka 的 Broker 实例业务 IP:Kafka 端口号',
  'format' = 'json',
  'topic' = 'zw_tset_kafka',
  'connector' = 'kafka',
  'scan.startup.mode' = 'latest-offset',
  'properties.sasl.kerberos.service.name' = 'kafka',
  'properties.security.protocol' = 'SASL_PLAINTEXT',
  'properties.kerberos.domain.name' = 'hadoop.hadoop.com'

);
CREATE CATALOG myhive WITH (
  'type' = 'hive',
  'hive-version' = '3.1.0',
  'default-database' = 'default',
  'cluster.name' = 'flink_hive'
```

```
);
use catalog myhive;
set table.sql-dialect = hive;create table user_behavior_hive_tbl_no_partition (
    user_id STRING,
    item_id STRING,
    cat_id STRING,
    ts timestamp
  ) PARTITIONED BY (dy STRING, ho STRING, mi STRING) stored as textfile
TBLPROPERTIES (
    'partition.time-extractor.timestamp-pattern' = '$dy $ho:$mi:00',
    'sink.partition-commit.trigger' = 'process-time',
    'sink.partition-commit.delay' = '0S',
    'sink.partition-commit.policy.kind' = 'metastore,success-file'
  );
INSERT into
  user_behavior_hive_tbl_no_partition
SELECT
  user_id,
  item_id,
  cat_id,
  zw_test,
  DATE_FORMAT(zw_test, 'yyyy-MM-dd'),
  DATE_FORMAT(zw_test, 'HH'),
  DATE_FORMAT(zw_test, 'mm')
FROM
  default_catalog.default_database.test_kafka;
```

说明

- Kafka 端口号：
- 集群的“认证模式”为“安全模式”时为“sasl.port”的值，默认为“21007”。
- 集群的“认证模式”为“普通模式”时为“port”的值，默认为“9092”。如果配置端口号为9092，则需要配置“allow.everyone.if.no.acl.found”参数为 true，具体操作如下：

  登录 FusionInsight Manager 系统，选择“集群 > 服务 > Kafka > 配置 > 全部配置”，搜索“allow.everyone.if.no.acl.found”配置，修改参数值为 true，保存配置即可。
- 'cluster.name' = 'flink_hive'的值为步骤 2 新建的集群连接名称。
- 相关参数可参考 Flink 官网：http://flink.apache.org/。

步骤 5 作业 SQL 开发完成后，请勾选“基础参数”中的“开启 CheckPoint”，“时间间隔（ms）”可设置为“60000”，“模式”可使用默认值。

步骤 6 单击左上角“提交”提交作业。

步骤 7 作业运行成功后，选择“更多 > 作业详情”可查看作业运行详情。

步骤 8 参考管理 Kafka 主题中的消息，查看 Topic 并向 Kafka 中写入数据。

./kafka-topics.sh --list --zookeeper *ZooKeeper 的 quorumpeer 实例业务 IP*:*ZooKeeper 客户端端口号***/kafka**

sh kafka-console-producer.sh --broker-list *Kafka 角色实例所在节点的 IP 地址*:*Kafka 端口号* **--topic** *主题名称* **--producer.config** *客户端目录***/Kafka/kafka/config/producer.properties**

例如本示例使用主题名称为 zw_tset_kafka：**sh kafka-console-producer.sh --broker-list *Kafka 角色实例所在节点的 IP 地址*:*Kafka 端口号* --topic zw_tset_kafka --producer.config /opt/Bigdata/client/Kafka/kafka/config/producer.properties**

输入消息内容：

```
{"user_id": "3","item_id":"333333","cat_id":"cat333","zw_test":"2021-09-08
09:08:01"}
{"user_id": "4","item_id":"444444","cat_id":"cat444","zw_test":"2021-09-08
09:08:01"}
```

输入完成后按回车发送消息。

说明

- ZooKeeper 的 quorumpeer 实例业务 IP：

  ZooKeeper 服务所有 quorumpeer 实例业务 IP。登录 FusionInsight Manager，选择“集群 > 服务 > ZooKeeper > 实例”，可查看所有 quorumpeer 实例所在主机业务 IP 地址。

- ZooKeeper 客户端端口号：

  登录 FusionInsight Manager，选择“集群 > 服务 > ZooKeeper”，在“配置”页签查看“clientPort”的值。默认为 24002。

步骤 9 执行以下命令查看 Sink 表中是否接收到数据，即 Hive 表是否正常写入数据。

beeline

select * from user_behavior_hive_tbl_no_partition;

----结束

6.7.10.5 FlinkServer 对接 Hudi

本章节适用于 MRS 3.1.2 及之后的版本。

操作场景

本指南通过使用 FlinkServer 写 FlinkSQL 对接 Hudi。

前提条件

- 集群已安装 HDFS、Yarn、Flink 和 Hudi 等服务。
- 包含 Hudi 服务的客户端已安装，例如安装路径为：/opt/Bigdata/client。
- Flink 要求 1.12.2 及以后版本，Hudi 要求 0.9.0 及以后版本。
- 参考基于用户和角色的鉴权创建一个具有 FlinkServer 管理员权限的用户用于访问 Flink WebUI，如：flink_admin。

Flink 对 Hudi 表的读写支持

Flink 对 Hudi 表的 COW 表、MOR 表类型读写支持详情见表 6-40。

表6-40 Flink 对 Hudi 表的读写支持

Flink SQL	COW 表	MOR 表
批量写	支持	支持
批量读	支持	支持
流式写	支持	支持
流式读	支持	支持

说明

1. 目前 FlinkSQL 对接 Hudi 仅支持 Snapshot mode 和 Read Optimized mode 两种模式。
2. 如果需要 bulkinsert 方式写入 Hudi 分区表并且使用 Date、TimeStamp 类型作为分区健时，请使用 Spark 的 bulkinsert 功能写入 Hudi。

操作步骤

步骤 1 使用 **flink_admin** 登录 Manager，选择“集群 > 服务 > Flink”，在“Flink WebUI”右侧，单击链接，访问 Flink 的 WebUI。

步骤 2 参考新建作业，新建 Flink SQL 流作业，在作业开发界面进行如下作业配置。并启动作业。

需勾选“基础参数”中的“开启 CheckPoint”，“时间间隔（ms）”可设置为“60000”，“模式”可使用默认值。

说明

- 由于 FlinkSQL 作业在触发 CheckPoint 时才会往 Hudi 表中写数据，所以需要在 Flink WebUI 界面中开启 CheckPoint。CheckPoint 间隔根据业务需要调整，建议间隔调大。
- 如果 CheckPoint 间隔太短，数据来不及刷新会导致作业异常；建议 CheckPoint 间隔为分钟级。
- FlinkSQL 作业写 MOR 表时需要做异步 compaction，控制 compaction 间隔的参数，见 Hudi 官网：https://hudi.apache.org/docs/configurations.html

- FlinkSQL 流式写入 MOR 表，仅支持 kafka json 格式。

```
CREATE TABLE stream_mor(
uuid VARCHAR(20),
name VARCHAR(10),
age INT,
ts INT,
`p` VARCHAR(20)
) PARTITIONED BY (`p`) WITH (
'connector' = 'hudi',
'path' = 'hdfs://hacluster/tmp/hudi/stream_mor',
'table.type' = 'MERGE_ON_READ'
);
```

```
CREATE TABLE kafka(
uuid VARCHAR(20),
name VARCHAR(10),
age INT,
ts INT,
`p` VARCHAR(20)
) WITH (
'connector' = 'kafka',
'topic' = 'writehudi',
'properties.bootstrap.servers' = 'Kafka的Broker实例业务IP:Kafka端口号',
'properties.group.id' = 'testGroup1',
'scan.startup.mode' = 'latest-offset',
'format' = 'json'
);

insert into
stream_mor
select
*
from
kafka;
```

- FlinkSQL 流式写入 COW 表

```
CREATE TABLE stream_write_cow(
uuid VARCHAR(20),
name VARCHAR(10),
age INT,
ts INT,
`p` VARCHAR(20)
) PARTITIONED BY (`p`) WITH (
'connector' = 'hudi',
'path' = 'hdfs://hacluster/tmp/hudi/stream cow'
);

CREATE TABLE kafka(
uuid VARCHAR(20),
name VARCHAR(10),
age INT,
ts INT,
`p` VARCHAR(20)
) WITH (
'connector' = 'kafka',
'topic' = 'writehudi',
'properties.bootstrap.servers' = 'Kafka的Broker实例业务IP:Kafka端口号',
'properties.group.id' = 'testGroup1',
'scan.startup.mode' = 'latest-offset',
'format' = 'json'
);

insert into
stream_write_cow
select
*
from
kafka;
```

- FlinkSQL 读取 MOR 表

```
CREATE TABLE hudi_read_spark_mor(
uuid VARCHAR(20),
name VARCHAR(10),
age INT,
ts INT,
`p` VARCHAR(20)
) PARTITIONED BY (`p`) WITH (
'connector' = 'hudi',
'path' = 'hdfs://hacluster/tmp/default/tb_hudimor',
'table.type' = 'MERGE_ON_READ'
);

CREATE TABLE kafka(
uuid VARCHAR(20),
name VARCHAR(10),
age INT,
ts timestamp(6)INT,
`p` VARCHAR(20)
) WITH (
'connector' = 'kafka',
'topic' = 'writehudi',
'properties.bootstrap.servers' = 'Kafka 的 Broker 实例业务 IP:Kafka 端口号',
'properties.group.id' = 'testGroup1',
'scan.startup.mode' = 'latest-offset',
'format' = 'json'
);

insert into
hudi_read_spark_mor
select
*
from
kafka;
```

说明

Kafka 端口号：

- 集群的“认证模式”为“安全模式”时为“sasl.port”的值，默认为“21007”。
- 集群的“认证模式”为“普通模式”时为“port”的值，默认为“9092”。如果配置端口号为9092，则需要配置“allow.everyone.if.no.acl.found”参数为 true，具体操作如下：

  登录 FusionInsight Manager 系统，选择“集群 > 服务 > Kafka > 配置 > 全部配置”，搜索“allow.everyone.if.no.acl.found”配置，修改参数值为 true，保存配置即可。

步骤 3 FlinkSQL 写入 Hudi 表数据后，通过 Spark、Hive 读该数据时，需要使用 run_hive_sync_tool.sh 将 Hudi 表数据同步到 Hive 中。同步方法请参考将 Hudi 表数据同步到 Hive。

须知

同步前需要保证不再新增分区，同步后新增的分区将不能被读取。

----结束

6.7.10.6 FlinkServer 对接 Kafka

本章节适用于 MRS 3.1.2 及之后的版本。

操作场景

本章节介绍 Kafka 作为 source 表或者 sink 表的 DDL 定义，以及创建表时使用的 WITH 参数和代码示例，并指导如何在 FlinkServer 作业管理页面操作。

本示例以安全模式 Kafka 为例。

前提条件

- 集群中已安装 HDFS、Yarn、Kafka 和 Flink 服务。
- 包含 Kafka 服务的客户端已安装，例如安装路径为：/opt/Bigdata/client
- 参考基于用户和角色的鉴权创建一个具有 FlinkServer 管理员权限的用户用于访问 Flink WebUI，如：flink_admin。

操作步骤

步骤 1 使用 **flink_admin** 登录 Manager，选择“集群 > 服务 > Flink”，在“Flink WebUI”右侧，单击链接，访问 Flink 的 WebUI。

步骤 2 参考新建作业，新建 Flink SQL 流作业，在作业开发界面进行作业开发，配置完成后启动作业。

需勾选“基础参数”中的“开启 CheckPoint”，“时间间隔（ms）”可设置为“60000”，“模式”可使用默认值。

```
CREATE TABLE KafkaSource (
  `user_id` VARCHAR,
  `user_name` VARCHAR,
  `age` INT
) WITH (
  'connector' = 'kafka',
  'topic' = 'test_source',
  'properties.bootstrap.servers' = 'Kafka 的 Broker 实例业务 IP:Kafka 端口号',
  'properties.group.id' = 'testGroup',
  'scan.startup.mode' = 'latest-offset',
  'format' = 'csv',
  'properties.sasl.kerberos.service.name' = 'kafka',
  'properties.security.protocol' = 'SASL_PLAINTEXT',
  'properties.kerberos.domain.name' = 'hadoop.hadoop.com'
);
CREATE TABLE KafkaSink(
  `user_id` VARCHAR,
```

```
  `user_name` VARCHAR,
  `age` INT
) WITH (
  'connector' = 'kafka',
  'topic' = 'test_sink',
  'properties.bootstrap.servers' = 'Kafka 的 Broker 实例业务 IP:Kafka 端口号',
  'scan.startup.mode' = 'latest-offset',
  'value.format' = 'csv',
  'properties.sasl.kerberos.service.name' = 'kafka',
  'properties.security.protocol' = 'SASL_PLAINTEXT',
  'properties.kerberos.domain.name' = 'hadoop.hadoop.com'
);
Insert into
  KafkaSink
select
  *
from
  KafkaSource;
```

说明

Kafka 端口号：

- 集群的“认证模式”为“安全模式”时为“sasl.port”的值，默认为“21007”。
- 集群的“认证模式”为“普通模式”时为“port”的值，默认为“9092”。如果配置端口号为9092，则需要配置“allow.everyone.if.no.acl.found”参数为 true，具体操作如下：

  登录 FusionInsight Manager 系统，选择“集群 > 服务 > Kafka > 配置 > 全部配置”，搜索“allow.everyone.if.no.acl.found”配置，修改参数值为 true，保存配置即可。

步骤 3 查看作业管理界面，作业状态为“运行中”。

步骤 4 参考管理 Kafka 主题中的消息，执行以下命令查看 Sink 表中是否接收到数据，即步骤 5 执行完成后查看 Kafka topic 是否正常写入数据。

sh kafka-console-consumer.sh --topic test_sink --bootstrap-server *Kafka 的 Broker 实例业务 IP:Kafka 端口号* **--consumer.config /opt/Bigdata/client/Kafka/kafka/config/consumer.properties**

步骤 5 参考管理 Kafka 主题中的消息，查看 Topic 并向 Kafka 中写入数据，输入完成后可在步骤 4 中的窗口查看执行结果。

./kafka-topics.sh --list --zookeeper *ZooKeeper 的 quorumpeer 实例业务 IP*:*ZooKeeper 客户端端口号*/**kafka**

sh kafka-console-producer.sh --broker-list *Kafka 角色实例所在节点的 IP 地址:Kafka 端口号* **--topic** *主题名称* **--producer.config** *客户端目录*/**Kafka/kafka/config/producer.properties**

例如本示例使用主题名称为 test_source：**sh kafka-console-producer.sh --broker-list** *Kafka 角色实例所在节点的 IP 地址:Kafka 端口号* **--topic test_source --producer.config /opt/Bigdata/client/Kafka/kafka/config/producer.properties**

输入消息内容：

```
1,clw,33
```

输入完成后按回车发送消息。

说明

- ZooKeeper 的 quorumpeer 实例业务 IP：

  ZooKeeper 服务所有 quorumpeer 实例业务 IP。登录 FusionInsight Manager，选择“集群 > 服务 > ZooKeeper > 实例”，可查看所有 quorumpeer 实例所在主机业务 IP 地址。

- ZooKeeper 客户端端口号：

  登录 FusionInsight Manager，选择“集群 > 服务 > ZooKeeper”，在“配置”页签查看“clientPort”的值。默认为 24002。

----结束

WITH 主要参数说明

配置项	是否必选	类型	描述
connector	必选	String	指定要使用的连接器，Kafka 使用“kafka”
topic	• kafka 作为 sink，必选 • kafka 作为 source，可选	String	主题名称 • 当表用作 source 时，要从中读取数据的主题名称。支持主题列表，通过按分号分隔主题，如“主题-1；主题-2” • 当表用作 sink 时，主题名称为写入数据的主题。sink 不支持主题列表
topic-pattern	kafka 作为 source 时可选	String	主题模式 当表用作 source 时可设置该参数，主题名称需使用正则表达式 说明 不能同时设置“topic-pattern”和“topic”。
properties.bootstrap.servers	必选	String	Kafka broker 列表，以逗号分隔
properties.group.id	kafka 作为 source 时必选	String	Kafka 的使用者组 ID
format	必选	String	用于反序列化和序列化 Kafka 消息的值部分的格式
properties.*	可选	String	安全模式下需增加认证相关的参数

6.7.10.7 FlinkServer 对接 Redis

本章节适用于 MRS 3.1.2 及之后的版本。

操作场景

本章节介绍 Redis 作为 sink 表或者维表的 DDL 定义，以及创建表时使用的 WITH 参数和代码示例，并指导如何在 FlinkServer 作业管理页面操作。

本示例以安全模式 Kafka 为例。

前提条件

- 集群中已安装 HDFS、Yarn、Redis 和 Flink 服务。
- 包含 Redis 服务的客户端已安装，例如安装路径为：/opt/Bigdata/client
- 参考基于用户和角色的鉴权创建一个具有 FlinkServer 管理员权限的用户用于访问 Flink WebUI，如：flink_admin。

操作步骤

场景一：Redis 作为 sink 表。

步骤 1 使用 **flink_admin** 登录 Manager，选择“集群 > 服务 > Flink”，在“Flink WebUI”右侧，单击链接，访问 Flink 的 WebUI。

步骤 2 参考新建作业，新建 Flink SQL 流作业，在作业开发界面进行作业开发，配置完成后启动作业。

需勾选“基础参数”中的“开启 CheckPoint”，“时间间隔（ms）”可设置为“60000”，“模式”可使用默认值。

```
CREATE TABLE kafka source (
  account varchar(10),
  costs int,
  ts AS PROCTIME()
) WITH (
  'connector' = 'kafka',
  'topic' = 'user_source',
  'properties.bootstrap.servers' = 'Kafka 的 Broker 实例业务 IP:Kafka 端口号',
  'properties.group.id' = 'testGroup',
  'scan.startup.mode' = 'latest-offset',
  'format' = 'json',
  'properties.sasl.kerberos.service.name' = 'kafka',
  'properties.security.protocol' = 'SASL_PLAINTEXT',
  'properties.kerberos.domain.name' = 'hadoop.系统域名'
);
CREATE table redis_sink(
  account varchar,
  costs int,
  PRIMARY KEY(account) NOT ENFORCED
) WITH (
  'connector' = 'redis',
  'need-kerberos-auth' = 'true',
  'service-kerberos-name' = 'redis/hadoop.系统域名',
```

```
  'login-context-name' = 'Client',
  'host' = '10.10.10.169',
  'port' = '22400',
  'data-type' = 'string',
  'namespace' = 'redis_table_2'

);
INSERT INTO
  redis_sink
SELECT
  account,
  SUM(costs)
FROM
  kafka_source
GROUP BY
  TUMBLE(ts, INTERVAL '90' SECOND),
  --为了快速看到计算结果
  account;
```

说明

- Kafka 端口号：
- 集群的“认证模式”为“安全模式”时为“sasl.port”的值，默认为“21007”。
- 集群的“认证模式”为“普通模式”时为“port”的值，默认为“9092”。如果配置端口号为9092，则需要配置“allow.everyone.if.no.acl.found”参数为 true，具体操作如下：

  登录 FusionInsight Manager 系统，选择“集群 > 服务 > Kafka > 配置 > 全部配置”，搜索“allow.everyone.if.no.acl.found”配置，修改参数值为 true，保存配置即可。
- host、port：分别为 Redis 集群的其中一个实例 IP（业务平面）和端口号。

  Redis 实例的端口计算方式为：22400+该实例的 ID-1。

  实例 ID 可以通过在 FusionInsight Manager 中选择“集群 > *待操作集群的名称* > 服务 > Redis > Redis 管理”，单击 Redis 集群名称查看。

  例如 Redis 集群内角色 R1 对应的 Redis 实例的端口为 22400+1-1=22400。
- namespace：用于拼接 Redis 数据库的键，格式为“namespace 的值:account 的值”。如发送数据的 account 的值为“A1”，namespace 的值为“redis_table_2”，那么此数据在 Redis 数据库中的键为 redis_table_2:A1。
- service-kerberos-name：为“**redis/hadoop**.*系统域名*”。可登录 FusionInsight Manager 界面，选择“系统 > 域和互信”中查看集群实际域名。

步骤 3 查看作业管理界面，作业状态为“运行中”。

步骤 4 参考管理 Kafka 主题中的消息，执行以下命令查看 Sink 表中是否接收到数据，即步骤 5 执行完成后查看 Kafka topic 是否正常写入数据。

sh kafka-console-consumer.sh --topic *主题名称* **--bootstrap-server** *Kafka 的 Broker 实例业务 IP:Kafka 端口号* **--consumer.config /opt/Bigdata/client/Kafka/kafka/config/consumer.properties**

步骤 5 参考管理 Kafka 主题中的消息，查看 Topic 并向 Kafka 中写入数据，输入完成后可在步骤 4 中的窗口查看执行结果。

./kafka-topics.sh --list --zookeeper *ZooKeeper 的 quorumpeer 实例业务 IP*:*ZooKeeper 客户端端口号***/kafka**

sh kafka-console-producer.sh --broker-list *Kafka 角色实例所在节点的 IP 地址*:*Kafka 端口号* **--topic** *主题名称* **--producer.config** *客户端目录* **/Kafka/kafka/config/producer.properties**

例如本示例使用主题名称为 user_source：**sh kafka-console-producer.sh --broker-list** *Kafka 角色实例所在节点的 IP 地址*:*Kafka 端口号* **--topic user_source --producer.config /opt/Bigdata/client/Kafka/kafka/config/producer.properties**

输入消息内容：

```
{"account": "A1","costs":"11"}
{"account": "A1","costs":"22"}
{"account": "A2","costs":"33"}
{"account": "A3","costs":"44"}
```

输入完成后按回车发送消息。

说明

- ZooKeeper 的 quorumpeer 实例业务 IP：

  ZooKeeper 服务所有 quorumpeer 实例业务 IP。登录 FusionInsight Manager，选择“集群 > 服务 > ZooKeeper > 实例”，可查看所有 quorumpeer 实例所在主机业务 IP 地址。

- ZooKeeper 客户端端口号：

  登录 FusionInsight Manager，选择“集群 > 服务 > ZooKeeper”，在“配置”页签查看“clientPort”的值。默认为 24002。

步骤 6 执行以下命令登录 Redis 客户端查询结果，以查询“redis_table_2:A1”为例。

redis-cli -c -h *Redis 集群中其中一个实例业务 IP* **-p** *Redis 端口号*

get *redis_table_2:A1*

----结束

场景二：Redis 作为维表。

步骤 1 使用 **flink_admin** 登录 Manager，选择“集群 > 服务 > Flink”，在“Flink WebUI”右侧，单击链接，访问 Flink 的 WebUI。

步骤 2 参考新建作业，新建 Flink SQL 流作业，在作业开发界面进行作业开发，配置完成后启动作业。

需勾选“基础参数”中的“开启 CheckPoint”，“时间间隔（ms）”可设置为“60000”，“模式”可使用默认值。

```
CREATE TABLE KafkaSource ( -- Kafka 作为 source 表
  `user id` VARCHAR,
  `user name` VARCHAR,
  `age` INT,
   proctime as proctime()
) WITH (
```

```
  'connector' = 'kafka',
  'topic' = 'user_source',
  'properties.bootstrap.servers' = 'Kafka 的 Broker 实例业务 IP:Kafka 端口号',
  'properties.group.id' = 'testGroup',
  'scan.startup.mode' = 'latest-offset',
  'value.format' = 'csv',
  'properties.sasl.kerberos.service.name' = 'kafka',
  'properties.security.protocol' = 'SASL_PLAINTEXT',
  'properties.kerberos.domain.name' = 'hadoop.系统域名'
);
CREATE TABLE KafkaSink (  -- Kafka 作为 sink 表
  `user_id` VARCHAR,
  `user_name` VARCHAR,
  `age` INT,
  `phone_number` VARCHAR,
  `address` VARCHAR
) WITH (
  'connector' = 'kafka',
  'topic' = 'user_sink',
  'properties.bootstrap.servers' = 'Kafka 的 Broker 实例业务 IP:Kafka 端口号',
  'properties.group.id' = 'testGroup',
  'scan.startup.mode' = 'latest-offset',
  'value.format' = 'csv',
  'properties.sasl.kerberos.service.name' = 'kafka',
  'properties.security.protocol' = 'SASL_PLAINTEXT',
  'properties.kerberos.domain.name' = 'hadoop.系统域名'
);
CREATE TABLE RedisTable ( -- Redis 作为维表
  user_name VARCHAR,
  score INT,
  phone_number VARCHAR,
  address VARCHAR
) WITH (

  'connector' = 'redis',
  'need-kerberos-auth' = 'true',
  'service-kerberos-name' = 'redis/hadoop.系统域名',
  'login-context-name' = 'Client',
  'zset-score-column' = 'score',
  'host' = '10.10.10.169',
  'port' = '22400',
  'key-ttl-mode' = 'no-ttl',
  'data-type' = 'sorted-set',
  'namespace' = 'dkdz_redis_table_1',
  'zset-delimiter' = ',',
  'key-column' = 'user_name'
);

INSERT INTO
  KafkaSink
SELECT
  t.user_id,
  t.user_name,
  t.age,
  d.phone_number,
```

```
  d.address
FROM
  KafkaSource as t
  JOIN RedisTable FOR SYSTEM_TIME AS OF t.proctime as d ON t.user_name =
d.user_name;
  -- 必须加上 FOR SYSTEM_TIME AS OF t.proctime，表示 JOIN 维表当前时刻所看到的每条数据
```

步骤 3 执行以下命令向 Redis 维表中写入测试数据。

cd /opt/Bigdata/client/Redis/bin

./redis-cli -h *10.10.10.11* **-p** *22400* **-c**

输入消息内容：

```
ZADD redis_zset:zhangsan 80 153xxxx1111,xian
ZADD redis_zset:lisi 70 153xxxx2222,shenzhen
ZADD redis_zset:wangwu 90 153xxxx3333,shanghai
```

说明

若 Redis 启用通道加密，使用命令：**./redis-cli -h** *10.10.10.11* **-p** *22400* **--tls -c**

登录 Manager，选择“集群 > 服务 > Redis > 配置 > 全部配置”，搜索“REDIS_SSL_ON”，将参数“值”设置为“true”，Redis 启用 SSL 通道加密。

步骤 4 生产数据，写入 Kafka Source 表中。

sh kafka-console-producer.sh --broker-list *Kafka 角色实例所在节点的 IP 地址:Kafka 端口号* **--topic** *主题名称* **--producer.config** *客户端目录* **/Kafka/kafka/config/producer.properties**

输入消息内容：

```
1,zhangsan,20
2,lisi,25
3,wangwu,28
```

步骤 5 执行以下命令查看 Sink 表中是否接收到数据，即查看 Kafka topic 是否正常写入数据。

sh kafka-console-consumer.sh --topic *主题名称* **--bootstrap-server** *Kafka 的 Broker 实例业务 IP:Kafka 端口号* **--consumer.config** *客户端目录* **/Kafka/kafka/config/consumer.properties**

结果如下：

```
1,zhangsan,20,153xxxx1111,xian
2,lisi,25,153xxxx2222,shenzhen
3,wangwu,28,153xxxx3333,shanghai
```

----结束

WITH 主要参数说明

配置项	是否必选	类型	描述
zSetScoreC	可选	Stri	Redis 作为维表时，ZSet 格式 score 字段对应的列名

配置项	是否必选	类型	描述
olumn		ng	
hashKeyColumn	可选	String	Hash 格式，Hash 字段对应的列名
host	必选	String	Redis 集群连接 IP，为 Redis 集群的实例 IP（业务平面）
port	必选	String	端口为对应的 Redis 实例的端口 Redis 实例的端口计算方式为：22400+该实例的 ID-1 实例 ID 可以通过在 FusionInsight Manager 中选择“集群 > *待操作集群的名称* > 服务 > Redis > Redis 管理”，单击 Redis 集群名称查看 例如 Redis 集群内角色 R1 对应的 Redis 实例的端口为 22400+1-1=22400
separator	可选	String	Redis 作为维表时，value 中的字段分割符，示例：“(,)”、“(\u200b)”
ttlDeadlineSecond	可选	String	TTL 截止时间，Sink 专用，例如：1615305600，表示写入的 key 将在 2021-03-10 00:00:00 数据过期
ttlDurationSecond	可选	String	TTL 时间，Sink 专用，例如：1000，表示写入的 key 将在写入 1000 秒后过期
keyPrefix	可选	String	Redis key 的前缀

6.8 Flink 任务运行残留信息清理

本章节适用于 MRS 3.1.2 及之后的版本。

操作场景

Flink 任务异常停止时会在 ZooKeeper、HDFS 中残留目录，开启 FlinkServer 目录残留清理功能可以清理残留目录。

前提条件

集群已安装 FlinkServer 实例并运行正常。

配置步骤

步骤 1 登录 Manager 页面。

步骤 2 选择“集群 > 服务 > Flink > 配置 > 全部配置”，搜索参数“ClearUpEnabled”并将值设置为“true”开启目录残留清理功能，相关参数详情请见表 6-41。

表6-41 FlinkServer 目录残留清理参数

参数	描述	默认值	取值范围
ClearUpEnabled	FlinkServer 是否开启目录残留清理功能。	true	true、false
ClearUpPeriod	FlinkServer 残留目录清理周期。单位：分钟	1440	1440~2147483647
TrashDirectoryRetentionPeriod	FlinkServer 保留残留目录的周期。单位：分钟	10080	10080~2147483647

步骤 3 配置完成后，单击左上角“保存”并确定保存。

须知

- 该特性只会清理 ZooKeeper 的“/flink_base”目录和 HDFS 的“/flink/recovery”目录下的残留目录，用户自定义修改的目录不会清理。
- HDFS 中的“checkpoints”目录需用户手动删除，该特性不会删除。

----结束

6.9 Flink 日志介绍

日志描述

日志存储路径：

- Flink 作业运行日志：“${BIGDATA_DATA_HOME}/hadoop/data${i}/nm/containerlogs/application_${appid}/container_{$contid}”。

  说明

  运行中的任务日志存储在以上路径中，运行结束后会基于 Yarn 的配置确定是否汇聚到 HDFS 目录中。

- FlinkResource 运行日志：“/var/log/Bigdata/flink/flinkResource”。

日志归档规则：

1. FlinkResource 运行日志：
   - 服务日志默认 20MB 滚动存储一次，最多保留 20 个文件，不压缩。

说明

针对 MRS 3.x 之前版本，Executor 日志默认 30MB 滚动存储一次，最多保留 20 个文件，不压缩。

- 日志大小和压缩文件保留个数可以在 Manager 界面中配置或者修改客户端“/opt/client/Flink/flink/conf/”中的 log4j-cli.properties、log4j.properties、log4j-session.properties 中对应的配置项。其中“/opt/client”为客户端安装目录。

表6-42 FlinkResource 日志列表

日志类型	日志文件名	描述
FlinkResource 运行日志	checkService.log	健康检查日志。
	kinit.log	初始化日志。
	postinstall.log	服务安装日志。
	prestart.log	prestart 脚本日志。
	start.log	启动日志。

日志级别

Flink 中提供了如表 6-43 所示的日志级别。日志级别优先级从高到低分别是 ERROR、WARN、INFO、DEBUG。程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。

表6-43 日志级别

级别	描述
ERROR	ERROR 表示当前时间处理存在错误信息。
WARN	WARN 表示当前事件处理存在异常信息。
INFO	INFO 表示记录系统及各事件正常运行状态信息。
DEBUG	DEBUG 表示记录系统及系统的调试信息。

如果您需要修改日志级别，请执行如下操作：

步骤 1 请参考修改集群服务配置参数，进入 Flink 的“全部配置”页面。

步骤 2 左边菜单栏中选择所需修改的角色所对应的日志菜单。

步骤 3 选择所需修改的日志级别。

步骤 4 保存配置，在弹出窗口中单击“确定”使配置生效。

----结束

说明

- 配置完成后不需要重启服务，重新下载客户端使配置生效。
- 也可以直接修改客户端“/opt/client/Flink/flink/conf/”中 log4j-cli.properties、log4j.properties、log4j-session.properties 文件中对应的日志级别配置项。其中“/opt/client”为客户端安装目录。
- 通过客户端提交作业时会在客户端 log 文件夹中生成相应日志文件，由于系统默认 umask 值是 0022，所以日志默认权限为 644；如果需要修改文件权限，需要修改 umask 值；例如修改 omm 用户 umask 值：
- 在“/home/omm/.baskrc”文件末尾添加“umask 0026”；
- 执行命令 **source /home/omm/.baskrc** 使文件权限生效。

日志格式

表6-44 日志格式

日志类型	格式	示例
运行日志	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<产生该日志的线程名字>|<log 中的 message>|<日志事件的发生位置>	2019-06-27 21:30:31,778 | INFO | [flink-akka.actor.default-dispatcher-3] | TaskManager container_e10_1498290698388_0004_02_000007 has started. | org.apache.flink.yarn.YarnFlinkResourceManager (FlinkResourceManager.java:368)

6.10 Flink 性能调优

6.10.1 DataStream 调优

6.10.1.1 配置内存

操作场景

Flink 是依赖内存计算，计算过程中内存不够对 Flink 的执行效率影响很大。可以通过监控 GC（Garbage Collection），评估内存使用及剩余情况来判断内存是否变成性能瓶颈，并根据情况优化。

监控节点进程的 YARN 的 Container GC 日志，如果频繁出现 Full GC，需要优化 GC。

说明

GC 的配置：在客户端的“conf/flink-conf.yaml”配置文件中，在“env.java.opts”配置项中添加参数：“-Xloggc:<LOG_DIR>/gc.log -XX:+PrintGCDetails -XX:-OmitStackTraceInFastThrow -XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=20 -XX:GCLogFileSize=20M”。 此处默认已经添加 GC 日志。

操作步骤

- 优化 GC。

  调整老年代和新生代的比值。在客户端的“conf/flink-conf.yaml”配置文件中，在“env.java.opts”配置项中添加参数：“-XX:NewRatio”。如“ -XX:NewRatio=2”，则表示老年代与新生代的比值为 2:1，新生代占整个堆空间的 1/3，老年代占 2/3。

- 开发 Flink 应用程序时，优化 DataStream 的数据分区或分组操作。
  - 当分区导致数据倾斜时，需要考虑优化分区。
  - 避免非并行度操作，有些对 DataStream 的操作会导致无法并行，例如 WindowAll。
  - keyBy 尽量不要使用 String。

6.10.1.2 设置并行度

操作场景

并行度控制任务的数量，影响操作后数据被切分成的块数。调整并行度让任务的数量和每个任务处理的数据与机器的处理能力达到最优。

查看 CPU 使用情况和内存占用情况，当任务和数据不是平均分布在各节点，而是集中在个别节点时，可以增大并行度使任务和数据更均匀的分布在各个节点。增加任务的并行度，充分利用集群机器的计算能力。

操作步骤

任务的并行度可以通过以下四种层次（按优先级从高到低排列）指定，用户可以根据实际的内存、CPU、数据以及应用程序逻辑的情况调整并行度参数。

- 算子层次

  一个算子、数据源和 sink 的并行度可以通过调用 setParallelism()方法来指定，例如

```
final StreamExecutionEnvironment env =
StreamExecutionEnvironment.getExecutionEnvironment();

DataStream<String> text = [...]
DataStream<Tuple2<String, Integer>> wordCounts = text
    .flatMap(new LineSplitter())
    .keyBy(0)
    .timeWindow(Time.seconds(5))
    .sum(1).setParallelism(5);

wordCounts.print();

env.execute("Word Count Example");
```

- 执行环境层次

  Flink 程序运行在执行环境中。执行环境为所有执行的算子、数据源、data sink 定义了一个默认的并行度。

  执行环境的默认并行度可以通过调用 setParallelism()方法指定。例如：

  ```
  final StreamExecutionEnvironment env =
  StreamExecutionEnvironment.getExecutionEnvironment();
  env.setParallelism(3);
  DataStream<String> text = [...]
  DataStream<Tuple2<String, Integer>> wordCounts = [...]
  wordCounts.print();
  env.execute("Word Count Example");
  ```

- 客户端层次

  并行度可以在客户端将 job 提交到 Flink 时设定。对于 CLI 客户端，可以通过“-p”参数指定并行度。例如：

  ```
  ./bin/flink run -p 10 ../examples/*WordCount-java*.jar
  ```

- 系统层次

  在系统级可以通过修改 Flink 客户端 conf 目录下的“flink-conf.yaml”文件中的“parallelism.default”配置选项来指定所有执行环境的默认并行度。

6.10.1.3 配置进程参数

操作场景

Flink on YARN 模式下，有 JobManager 和 TaskManager 两种进程。在任务调度和运行的过程中，JobManager 和 TaskManager 承担了很大的责任。

因而 JobManager 和 TaskManager 的参数配置对 Flink 应用的执行有着很大的影响意义。用户可通过如下操作对 Flink 集群性能做优化。

操作步骤

步骤 1 配置 JobManager 内存。

JobManager 负责任务的调度，以及 TaskManager、RM 之间的消息通信。当任务数变多，任务平行度增大时，JobManager 内存都需要相应增大。

您可以根据实际任务数量的多少，为 JobManager 设置一个合适的内存。

- 在使用 yarn-session 命令时，添加“-jm MEM”参数设置内存。
- 在使用 yarn-cluster 命令时，添加“-yjm MEM”参数设置内存。

步骤 2 配置 TaskManager 个数。

每个 TaskManager 每个核同时能跑一个 task，所以增加了 TaskManager 的个数相当于增大了任务的并发度。在资源充足的情况下，可以相应增加 TaskManager 的个数，以提高运行效率。

步骤 3 配置 TaskManager Slot 数。

每个 TaskManager 多个核同时能跑多个 task，相当于增大了任务的并发度。但是由于所有核共用 TaskManager 的内存，所以要在内存和核数之间做好平衡。

- 在使用 yarn-session 命令时，添加“-s NUM”参数设置 SLOT 数。
- 在使用 yarn-cluster 命令时，添加“-ys NUM”参数设置 SLOT 数。

步骤 4 配置 TaskManager 内存。

TaskManager 的内存主要用于任务执行、通信等。当一个任务很大的时候，可能需要较多资源，因而内存也可以做相应的增加。

- 将在使用 yarn-session 命令时，添加“-tm MEM”参数设置内存。
- 将在使用 yarn-cluster 命令时，添加“-ytm MEM”参数设置内存。

----结束

6.10.1.4 设计分区方法

操作场景

合理的设计分区依据，可以优化 task 的切分。在程序编写过程中要尽量分区均匀，这样可以实现每个 task 数据不倾斜，防止由于某个 task 的执行时间过长导致整个任务执行缓慢。

操作步骤

以下是几种分区方法。

- **随机分区：**将元素随机地进行分区。

```
dataStream.shuffle();
```

- **Rebalancing (Round-robin partitioning)：**基于 round-robin 对元素进行分区，使得每个分区负责均衡。对于存在数据倾斜的性能优化是很有用的。

```
dataStream.rebalance();
```

- **Rescaling：**以 round-robin 的形式将元素分区到下游操作的子集中。如果你想要将数据从一个源的每个并行实例中散发到一些 mappers 的子集中，用来分散负载，但是又不想要完全的 rebalance 介入（引入`rebalance()`），这会非常有用。

```
dataStream.rescale();
```

- **广播：**广播每个元素到所有分区。

```
dataStream.broadcast();
```

- **自定义分区：**使用一个用户自定义的 Partitioner 对每一个元素选择目标 task，由于用户对自己的数据更加熟悉，可以按照某个特征进行分区，从而优化任务执行。

  简单示例如下所示：

```
// fromElements 构造简单的 Tuple2 流
DataStream<Tuple2<String, Integer>> dataStream =
env.fromElements(Tuple2.of("hello",1), Tuple2.of("test",2),
Tuple2.of("world",100));

// 定义用于分区的 key 值，返回即属于哪个 partition 的，该值加 1 就是对应的子任务的 id 号
Partitioner<Tuple2<String, Integer>> strPartitioner = new
Partitioner<Tuple2<String, Integer>>() {
    @Override
```

```
    public int partition(Tuple2<String, Integer> key, int numPartitions) {
        return (key.f0.length() + key.f1) % numPartitions;
    }
};

// 使用 Tuple2 进行分区的 key 值
dataStream.partitionCustom(strPartitioner, new KeySelector<Tuple2<String,
Integer>, Tuple2<String, Integer>>() {
    @Override
    public Tuple2<String, Integer> getKey(Tuple2<String, Integer> value) throws
Exception {
        return value;
    }
}).print();
```

6.10.1.5 配置 netty 网络通信

操作场景

Flink 通信主要依赖 netty 网络，所以在 Flink 应用执行过程中，netty 的设置尤为重要，网络通信的好坏直接决定着数据交换的速度以及任务执行的效率。

操作步骤

以下配置均可在客户端的“conf/flink-conf.yaml”配置文件中进行修改适配，默认已经是相对较优解，请谨慎修改，防止性能下降。

- “taskmanager.network.netty.num-arenas”： 默认是“taskmanager.numberOfTaskSlots”，表示 netty 的域的数量。
- “taskmanager.network.netty.server.numThreads”和“taskmanager.network.netty.client.numThreads”：默认是“taskmanager.numberOfTaskSlots”，表示 netty 的客户端和服务端的线程数目设置。
- “taskmanager.network.netty.client.connectTimeoutSec”：默认是 120s，表示 taskmanager 的客户端连接超时的时间。
- “taskmanager.network.netty.sendReceiveBufferSize”：默认是系统缓冲区大小(cat /proc/sys/net/ipv4/tcp_[rw]mem) ，一般为 4MB，表示 netty 的发送和接收的缓冲区大小。
- “taskmanager.network.netty.transport”：默认为“nio”方式，表示 netty 的传输方式，有“nio”和“epoll”两种方式。

6.10.1.6 经验总结

数据倾斜

当数据发生倾斜（某一部分数据量特别大），虽然没有 GC（Gabage Collection，垃圾回收），但是 task 执行时间严重不一致。

- 需要重新设计 key，以更小粒度的 key 使得 task 大小合理化。
- 修改并行度。

- 调用 rebalance 操作，使数据分区均匀。

缓冲区超时设置

- 由于 task 在执行过程中存在数据通过网络进行交换，数据在不同服务器之间传递的缓冲区超时时间可以通过 setBufferTimeout 进行设置。
- 当设置“setBufferTimeout(-1)”，会等待缓冲区满之后才会刷新，使其达到最大吞吐量；当设置“setBufferTimeout(0)”时，可以最小化延迟，数据一旦接收到就会刷新；当设置“setBufferTimeout”大于 0 时，缓冲区会在该时间之后超时，然后进行缓冲区的刷新。

  示例可以参考如下：

```
env.setBufferTimeout(timeoutMillis);

env.generateSequence(1,10).map(new MyMapper()).setBufferTimeout(timeoutMillis);
```

6.11 Flink 常见 Shell 命令

本章节适用于 MRS 3.x 及之后版本。

在使用 Flink 的 Shell 脚本前，首先需要执行以下操作：

步骤 1 安装 Flink 客户端，例如安装目录为“/opt/client”。

步骤 2 初始化环境变量。

source /opt/client/bigdata_env

步骤 3 如果当前集群已启用 Kerberos 认证，执行以下命令认证当前用户。如果当前集群未启用 Kerberos 认证，则无需执行此命令。

kinit *业务用户*

步骤 4 参考表 6-45 运行相关命令。

表6-45 Flink Shell 命令参考

命令	参数说明	描述
yarn-session.sh	-at,--applicationType <arg>：为 Yarn application 自定义类型。 -D <property=value>：动态参数配置。 -d,--detached：关闭交互模式，启动一个分离的 Flink YARN session。 -h,--help: 显示 Yarn session CLI 的帮助。 -id,--applicationId <arg>：绑定到一个已经运行的 Yarn session。 -j,--jar <arg>：设置用户 jar 包路径。 -jm,--jobManagerMemory <arg>：为 JobManager 设置内存。	启动一个常驻的 Flink 集群，接受来自 Flink 客户端的任务。

命令	参数说明	描述
	-m,--jobmanager <arg>：要连接的 JobManager 的地址，使用该参数可以连接特定的 JobManager。 -nl,--nodeLabel <arg>: 指定 YARN application 的 nodeLabel 。 -nm,--name <arg>：为 Yarn application 自定义名称。 -q,--query：查询可用的 Yarn 资源。 -qu,--queue <arg>：指定 YARN 队列。 -s,--slots <arg>：设置每个 Taskmanager 的 SLOT 个数。 -t,--ship <arg>：指定待发送文件的目录。 -tm,--taskManagerMemory <arg>：为 TaskManager 设置内存。 -yd,--yarndetached：以分离模式启动。 -z,--zookeeperNamespace <args>：指定 zookeeper 的 namespace。 -h：获取帮助。	
flink run	-c,--class <classname>：指定一个类作为程序运行的入口点。 -C,--classpath <url>：指定 classpath。 -d,--detached：以分离方式运行 job。 -files,--dependencyFiles <arg>：Flink 程序依赖的文件。 -n,--allowNonRestoredState： 从快照点恢复时允许跳过不能恢复的状态。比如删除了程序中某个操作符，那么在恢复快照点时需要增加该参数。 -m,--jobmanager <host:port>：指定 JobManager。 -p,--parallelism <parallelism>：指定 job 并行度，会覆盖配置文件中配置的并行度参数。 -q,--sysoutLogging：禁止 flink 日志输出至控制台。 -s,--fromSavepoint <savepointPath>：指定用于恢复 job 的 savepoint 路径。 -z,--zookeeperNamespace <zookeeperNamespace>：指定 zookeeper 的 namespace。 -yat,--yarnapplicationType <arg>：为 Yarn	Flink 提交作业。 1."-y*"参数是指 yarn-cluster 模式下使用。 2.非"-y*"参数用户在用该命令提交任务前需要先用 yarn-session 启动 Flink 集群。

命令	参数说明	描述
	application 自定义类型。 -yD <arg>：动态参数配置。 -yd,--yarndetached：以分离模式启动。 -yh,--yarnhelp：获取 yarn 帮助。 -yid,--yarnapplicationId <arg>：绑定到 yarn session 运行 job。 -yj,--yarnjar <arg>：设置 Flink jar 文件路径。 -yjm,--yarnjobManagerMemory <arg>：为 JobManager 设置内存（MB）。 -ynm,--yarnname <arg>：为 Yarn application 自定义名称。 -yq,--yarnquery：查询可用的 YARN 资源（内存、CPU）。 -yqu,--yarnqueue <arg>：指定 YARN 队列。 -ys,--yarnslots：设置每个 TaskManager 的 SLOT 个数。 -yt,--yarnship <arg>：指定待发送文件的路径。 -ytm,--yarntaskManagerMemory <arg>：为 TaskManager 设置内存（MB）。 -yz,--yarnzookeeperNamespace <arg>：指定 zookeeper 的 namespace，需与 yarn-session.sh -z 保持一致。 -h：获取帮助。	
flink info	-c,--class <classname>：指定一个类作为程序运行的入口点。 -p,--parallelism <parallelism>：指定程序运行的并行度。 -h：获取帮助。	显示所运行程序的执行计划（JSON）
flink list	-a,--all：显示所有的 Job。 -m,--jobmanager <host:port>：指定 JobManager。 -r,--running：仅显示 running 状态的 Job。 -s,--scheduled：仅显示 scheduled 状态的 Job。 -z,--zookeeperNamespace <zookeeperNamespace>：指定 zookeeper 的 namespace。 -yid,--yarnapplicationId <arg>：绑定 YARN session。	查询集群中运行的程序。

命令	参数说明	描述
	-h：获取帮助。	
flink stop	-d,--drain：在触发 savepoint 和停止作业之前，发送 MAX_WATERMARK。 -p,--savepointPath <savepointPath>：savepoint 的储存路径，默认目录 state.savepoints.dir。 -m,--jobmanager <host:port>：指定 JobManager。 -z,--zookeeperNamespace <zookeeperNamespace>：指定 zookeeper 的 namespace。 -yid,--yarnapplicationId <arg>：绑定 YARN session。 -h：获取帮助。	强制停止一个运行中的 Job（仅支持 streaming jobs、业务代码 source 端需要 implements StoppableFunction）
flink cancel	-m,--jobmanager <host:port>：指定 JobManager。 -s,--withSavepoint <targetDirectory>：取消 Job 时触发 savepoint，默认目录 state.savepoints.dir -z,--zookeeperNamespace <zookeeperNamespace>：指定 zookeeper 的 namespace。 -yid,--yarnapplicationId <arg>：绑定 YARN session。 -h：获取帮助。	取消一个运行中 Job
flink savepoint	-d,--dispose <arg>：指定 savepoint 的保存目录。 -m,--jobmanager <host:port>：指定 JobManager。 -z,--zookeeperNamespace <zookeeperNamespace>：指定 zookeeper 的 namespace。 -yid,--yarnapplicationId <arg>：绑定 YARN session。 -h：获取帮助。	触发一个 savepoint
source *客户端安装目录* **/bigdata_env**	无	导入客户端环境变量。 使用限制：如果用户使用自定义脚本（例如 A.sh）并在脚本中调用该命令，则脚本 A.sh 不能传入参数。如果确实需要给 A.sh

命令	参数说明	描述
		传入参数，则需采用二次调用方式。 例如 A.sh 中调用 B.sh，在 B.sh 中调用该命令。A.sh 可以传入参数，B.sh 不能传入参数。
start-scala-shell.sh	local | remote <host> <port> | yarn：运行模式	scala shell 启动脚本
sh generate_keystore.sh	-	用户调用“generate_keystore.sh”脚本工具生成“Security Cookie”、“flink.keystore”和“flink.truststore”。需要输入自定义密码（不能包含#）。

----结束

6.12 参考

6.12.1 签发证书样例

将该样例代码生成 generate_keystore.sh 脚本，放置在 Flink 客户端的 bin 目录下。

```
#!/bin/bash

KEYTOOL=${JAVA_HOME}/bin/keytool
KEYSTOREPATH="$FLINK_HOME/conf/"
CA_ALIAS="ca"
CA_KEYSTORE_NAME="ca.keystore"
CA_DNAME="CN=Flink_CA"
CA_KEYALG="RSA"
CLIENT_CONF_YAML="$FLINK_HOME/conf/flink-conf.yaml"
KEYTABPRINCEPAL=""

function getConf()
{
   if [ $# -ne 2 ]; then
      echo "invalid parameters for getConf"
      exit 1
   fi

   confName="$1"
```

```
    if [ -z "$confName" ]; then
        echo "conf name is empty."
        exit 2
    fi

    configFile=$FLINK_HOME/conf/client.properties
    if [ ! -f $configFile ]; then
        echo $configFile" is not exist."
        exit 3
    fi

    defaultValue="$2"
    cnt=$(grep $1 $configFile | wc -l)
    if [ $cnt -gt 1 ]; then
        echo $confName" has multi values in "$configFile
        exit 4
    elif [ $cnt -lt 1 ]; then
        echo $defaultValue
    else
        line=$(grep $1 $configFile)
        confValue=$(echo "${line#*=}")
        echo "$confValue"
    fi
}

function createSelfSignedCA()
{
    #varible from user input
    keystorePath=$1
    storepassValue=$2
    keypassValue=$3

    #generate ca keystore
    rm -rf $keystorePath/$CA_KEYSTORE_NAME
    $KEYTOOL -genkeypair -alias $CA_ALIAS -keystore $keystorePath/$CA_KEYSTORE_NAME
-dname $CA_DNAME -storepass $storepassValue -keypass $keypassValue -validity 3650 -
keyalg $CA_KEYALG -keysize 3072 -ext bc=ca:true
    if [ $? -ne 0 ]; then
        echo "generate ca.keystore failed."
        exit 1
    fi

    #generate ca.cer
    rm -rf "$keystorePath/ca.cer"
    $KEYTOOL -keystore "$keystorePath/$CA_KEYSTORE_NAME" -storepass
"$storepassValue" -alias $CA_ALIAS -validity 3650 -exportcert >
"$keystorePath/ca.cer"
    if [ $? -ne 0 ]; then
        echo "generate ca.cer failed."
        exit 1
    fi

    #generate ca.truststore
    rm -rf "$keystorePath/flink.truststore"
    $KEYTOOL -importcert -keystore "$keystorePath/flink.truststore" -alias $CA_ALIAS
```

```
-storepass "$storepassValue" -noprompt -file "$keystorePath/ca.cer"
    if [ $? -ne 0 ]; then
        echo "generate ca.truststore failed."
        exit 1
    fi
}

function generateKeystore()
{
    #get path/pass from input
    keystorePath=$1
    storepassValue=$2
    keypassValue=$3

    #get value from conf
    aliasValue=$(getConf "flink.keystore.rsa.alias" "flink")
    validityValue=$(getConf "flink.keystore.rsa.validity" "3650")
    keyalgValue=$(getConf "flink.keystore.rsa.keyalg" "RSA")
    dnameValue=$(getConf "flink.keystore.rsa.dname" "CN=flink.xxx.com")
    SANValue=$(getConf "flink.keystore.rsa.ext" "ip:127.0.0.1")
    SANValue=$(echo "$SANValue" | xargs)
    SANValue="ip:$(echo "$SANValue"| sed 's/,/,ip:/g')"

    #generate keystore
    rm -rf $keystorePath/flink.keystore
    $KEYTOOL -genkeypair -alias $aliasValue -keystore $keystorePath/flink.keystore -
dname $dnameValue -ext SAN=$SANValue -storepass $storepassValue -keypass
$keypassValue -keyalg $keyalgValue -keysize 3072 -validity 3650
    if [ $? -ne 0 ]; then
        echo "generate flink.keystore failed."
        exit 1
    fi

    #generate cer
    rm -rf $keystorePath/flink.csr
    $KEYTOOL -certreq -keystore $keystorePath/flink.keystore -storepass
$storepassValue -alias $aliasValue -file $keystorePath/flink.csr
    if [ $? -ne 0 ]; then
        echo "generate flink.csr failed."
        exit 1
    fi

    #generate flink.cer
    rm -rf $keystorePath/flink.cer
    $KEYTOOL -gencert -keystore $keystorePath/ca.keystore -storepass $storepassValue
-alias $CA_ALIAS -ext SAN=$SANValue -infile $keystorePath/flink.csr -outfile
$keystorePath/flink.cer -validity 3650
    if [ $? -ne 0 ]; then
        echo "generate flink.cer failed."
        exit 1
    fi

    #import cer into keystore
    $KEYTOOL -importcert -keystore $keystorePath/flink.keystore -storepass
$storepassValue -file $keystorePath/ca.cer -alias $CA_ALIAS -noprompt
```

```
    if [ $? -ne 0 ]; then
        echo "importcert ca."
        exit 1
    fi

    $KEYTOOL -importcert -keystore $keystorePath/flink.keystore -storepass
$storepassValue -file $keystorePath/flink.cer -alias $aliasValue -noprompt;
    if [ $? -ne 0 ]; then
        echo "generate flink.truststore failed."
        exit 1
    fi
}

function configureFlinkConf()
{
    # set config
    if [ -f "$CLIENT_CONF_YAML" ]; then
        SSL_ENCRYPT_ENABLED=$(grep "security.ssl.encrypt.enabled"
"$CLIENT_CONF_YAML" | awk '{print $2}')
        if [ "$SSL_ENCRYPT_ENABLED" = "false" ];then

            sed -i s/"security.ssl.key-password:".*/"security.ssl.key-password:"\
"${keyPass}"/g "$CLIENT_CONF_YAML"
            if [ $? -ne 0 ]; then
                echo "set security.ssl.key-password failed."
                return 1
            fi

            sed -i s/"security.ssl.keystore-password:".*/"security.ssl.keystore-
password:"\ "${storePass}"/g "$CLIENT_CONF_YAML"
            if [ $? -ne 0 ]; then
                echo "set security.ssl.keystore-password failed."
                return 1
            fi

            sed -i s/"security.ssl.truststore-password:".*/"security.ssl.truststore-
password:"\ "${storePass}"/g "$CLIENT_CONF_YAML"
            if [ $? -ne 0 ]; then
                echo "set security.ssl.keystore-password failed."
                return 1
            fi

            echo "security.ssl.encrypt.enabled is false, set security.ssl.key-
password security.ssl.keystore-password security.ssl.truststore-password success."
        else
            echo "security.ssl.encrypt.enabled is true, please enter
security.ssl.key-password security.ssl.keystore-password security.ssl.truststore-
password encrypted value in flink-conf.yaml."
        fi

        keystoreFilePath="${keystorePath}"/flink.keystore
        sed -i 's#'"security.ssl.keystore:".*'#'"security.ssl.keystore:"\
"$keystoreFilePath"'#g' "$CLIENT_CONF_YAML"
        if [ $? -ne 0 ]; then
            echo "set security.ssl.keystore failed."
```

```
          return 1
       fi


       truststoreFilePath="${keystorePath}/flink.truststore"
       sed -i 's#'"security.ssl.truststore:".*'#'"security.ssl.truststore:"\
"$truststoreFilePath"'#g' "$CLIENT_CONF_YAML"
       if [ $? -ne 0 ]; then
          echo "set security.ssl.truststore failed."
          return 1
       fi

       command -v sha256sum >/dev/null
       if [ $? -ne 0 ];then
          echo "sha256sum is not exist, it will produce security.cookie with date
+%F-%H-%M-%s-%N."
          cookie=$(date +%F-%H-%M-%s-%N)
       else
          cookie="$(echo "${KEYTABPRINCEPAL}"| sha256sum | awk '{print $1}')"
       fi

       sed -i s/"security.cookie:".*/"security.cookie:"\ "${cookie}"/g
"$CLIENT_CONF_YAML"
       if [ $? -ne 0 ]; then
          echo "set security.cookie failed."
          return 1
       fi
    fi
    return 0;
}

main()
{
    #check environment variable is set or not
    if [ -z ${FLINK_HOME+x} ]; then
       echo "errro: environment variables are not set."
       exit 1
    fi
    stty -echo
    read -rp "Enter password:" password
    stty echo
    echo


    KEYTABPRINCEPAL=$(grep "security.kerberos.login.principal" "$CLIENT_CONF_YAML" |
awk '{print $2}')
    if [ -z "$KEYTABPRINCEPAL" ];then
       echo "please config security.kerberos.login.principal info first."
       exit 1
    fi


    #get input
    keystorePath="$KEYSTOREPATH"
    storePass="$password"
```

```
    keyPass="$password"

    #generate self signed CA
    createSelfSignedCA "$keystorePath" "$storePass" "$keyPass"
    if [ $? -ne 0 ]; then
        echo "create self signed ca failed."
        exit 1
    fi

    #generate keystore
    generateKeystore "$keystorePath" "$storePass" "$keyPass"
    if [ $? -ne 0 ]; then
        echo "create keystore failed."
        exit 1
    fi

    echo "generate keystore/truststore success."

    # set flink config
    configureFlinkConf "$keystorePath" "$storePass" "$keyPass"
    if [ $? -ne 0 ]; then
        echo "configure Flink failed."
        exit 1
    fi

    return 0;
}

#the start main
main "$@"

exit 0
```

说明

执行命令“sh generate_keystore.sh <password>”即可，<password>由用户自定义输入

- 若<password>中包含特殊字符"$"，应使用如下方式，以防止被转义，“sh generate_keystore.sh 'Bigdata_2013'”
- 密码不允许包含“#”。
- 使用该 generate_keystore.sh 脚本前需要在客户端目录下执行 source bigdata_env。
- 使用该 generate_keystore.sh 脚本会自动将 security.ssl.keystore、security.ssl.truststore 的绝对路径填写到 flink-conf.yaml 中，所以需要用户根据实际情况手动修改为相对路径。例如：
- 将 security.ssl.keystore: /opt/client/Flink/flink/conf//flink.keystore 修改为 security.ssl.keystore: ssl/flink.keystore；
- 将 security.ssl.truststore: /opt/client/Flink/flink/conf//flink.truststore 修改为 security.ssl.truststore: ssl/flink.truststore；
- 需要在 Flink 客户端环境中任意目录下创建 ssl 文件夹，如在“/opt/client/Flink/flink/conf/”目录下新建目录 ssl，将 flink.keystore、flink.truststore 文件放入 ssl 文件夹中；

- 执行 yarn-session 或者 flink run -m yarn-cluster 命令时需要在 ssl 文件夹同级目录下执行：yarn-session.sh -t ssl -d 或者 flink run -m yarn-cluster -yt ssl -d WordCount.jar 。

7 使用 Flume

7.1 从零开始使用 Flume

操作场景

Flume 支持将采集的日志信息导入到 Kafka。

前提条件

- 已创建启用 Kerberos 认证的流集群。
- 已在日志生成节点安装 Flume 客户端，例如安装目录为“/opt/Flumeclient”，客户端安装请参见“组件操作指南 > 使用 Flume > 安装 Flume 客户端”。以下操作的客户端目录只是举例，请根据实际安装目录修改。
- 已配置网络，使日志生成节点与流集群互通。

使用 Flume 客户端（MRS 3.x 之前版本）

说明

普通集群不需要执行步骤 1-步骤 5。

步骤 1 将 Master1 节点上的认证服务器配置文件，复制到安装 Flume 客户端的节点，保存到 Flume 客户端中“Flume 客户端安装目录/fusioninsight-flume-*Flume 组件版本号*/conf”目录下。

文件完整路径为${BIGDATA_HOME}/MRS_Current/1_*X*_KerberosClient/etc/kdc.conf。

其中“X”为随机生成的数字，请根据实际情况修改。同时文件需要以 Flume 客户端安装用户身份保存，例如 **root** 用户。

步骤 2 查看任一部署 Flume 角色节点的“业务 IP”。

登录集群详情页面，选择“集群 > 组件管理 > Flume > 实例”，查看任一部署 Flume 角色节点的“业务 IP”。

说明

若集群详情页面没有“组件管理”页签，请先完成 IAM 用户同步（在集群详情页的“概览”页签，单击“IAM 用户同步”右侧的“同步”进行 IAM 用户同步）。

步骤 3 将此节点上的用户认证文件，复制到安装 Flume 客户端的节点，保存到 Flume 客户端中“Flume 客户端安装目录/fusioninsight-flume-*Flume 组件版本号*/conf”目录下。

文件完整路径为${BIGDATA_HOME}/MRS_*XXX*/install/FusionInsight-Flume-*Flume 组件版本号*/flume/conf/flume.keytab。

其中“XXX”为产品版本号，请根据实际情况修改。同时文件需要以 Flume 客户端安装用户身份保存，例如 **root** 用户。

步骤 4 将此节点上的配置文件“jaas.conf”，复制到安装 Flume 客户端的节点，保存到 Flume 客户端中“conf”目录。

文件完整路径为${BIGDATA_HOME}/MRS_Current/1_*X*_Flume/etc/jaas.conf。

其中“X”为随机生成的数字，请根据实际情况修改。同时文件需要以 Flume 客户端安装用户身份保存，例如 **root** 用户。

步骤 5 登录安装 Flume 客户端节点，切换到客户端安装目录，执行以下命令修改文件：

vi conf/jaas.conf

修改参数“keyTab”定义的用户认证文件完整路径即步骤 3 中保存用户认证文件的目录：“Flume 客户端安装目录/fusioninsight-flume-*Flume 组件版本号*/conf”，然后保存并退出。

步骤 6 执行以下命令，修改 Flume 客户端配置文件“flume-env.sh”：

vi *Flume 客户端安装目录*/**fusioninsight-flume-***Flume 组件版本号*/**conf/flume-env.sh**

在“-XX:+UseCMSCompactAtFullCollection”后面，增加以下内容：

```
-Djava.security.krb5.conf=Flume 客户端安装目录/fusioninsight-flume-1.9.0/conf/kdc.conf
-Djava.security.auth.login.config=Flume 客户端安装目录/fusioninsight-flume-
1.9.0/conf/jaas.conf -Dzookeeper.request.timeout=120000
```

例如：**"-XX:+UseCMSCompactAtFullCollection -Djava.security.krb5.conf=/opt/FlumeClient/fusioninsight-flume-***Flume 组件版本号***/conf/kdc.conf -Djava.security.auth.login.config=/opt/FlumeClient/fusioninsight-flume-***Flume 组件版本号***/conf/jaas.conf -Dzookeeper.request.timeout=120000"**

请根据实际情况，修改“Flume 客户端安装目录”，然后保存并退出。

步骤 7 执行以下命令，重启 Flume 客户端：

cd *Flume 客户端安装目录*/**fusioninsight-flume-***Flume 组件版本号*/**bin**

./flume-manage.sh restart

例如：

cd /opt/FlumeClient/fusioninsight-flume-*Flume 组件版本号*/**bin**

./flume-manage.sh restart

步骤 8 执行以下命令，根据实际业务需求，在 Flume 客户端配置文件“properties.properties”中配置并保存作业。

vi *Flume 客户端安装目录*/**fusioninsight-flume-***Flume 组件版本号*
/conf/properties.properties

以配置 SpoolDir Source+File Channel+Kafka Sink 为例：

```
################################################################################################
client.sources = static_log_source
client.channels = static_log_channel
client.sinks = kafka_sink
################################################################################################
#LOG_TO_HDFS_ONLINE_1

client.sources.static_log_source.type = spooldir
client.sources.static_log_source.spoolDir = 监控目录
client.sources.static_log_source.fileSuffix = .COMPLETED
client.sources.static_log_source.ignorePattern = ^$
client.sources.static_log_source.trackerDir = 传输过程中元数据存储路径
client.sources.static_log_source.maxBlobLength = 16384
client.sources.static_log_source.batchSize = 51200
client.sources.static_log_source.inputCharset = UTF-8
client.sources.static_log_source.deserializer = LINE
client.sources.static_log_source.selector.type = replicating
client.sources.static_log_source.fileHeaderKey = file
client.sources.static_log_source.fileHeader = false
client.sources.static_log_source.basenameHeader = true
client.sources.static_log_source.basenameHeaderKey = basename
client.sources.static_log_source.deletePolicy = never

client.channels.static_log_channel.type = file
client.channels.static_log_channel.dataDirs = 数据缓存路径，设置多个路径可提升性能，中间用逗号分开
client.channels.static_log_channel.checkpointDir = 检查点存放路径
client.channels.static_log_channel.maxFileSize = 2146435071
client.channels.static_log_channel.capacity = 1000000
client.channels.static_log_channel.transactionCapacity = 612000
client.channels.static_log_channel.minimumRequiredSpace = 524288000

client.sinks.kafka_sink.type = org.apache.flume.sink.kafka.KafkaSink
client.sinks.kafka_sink.kafka.topic = 数据写入的topic，如flume_test
client.sinks.kafka_sink.kafka.bootstrap.servers = XXX.XXX.XXX.XXX:kafka端口号,XXX.XXX.XXX.XXX:kafka端口号,XXX.XXX.XXX.XXX:kafka端口号
client.sinks.kafka_sink.flumeBatchSize = 1000
client.sinks.kafka_sink.kafka.producer.type = sync
client.sinks.kafka_sink.kafka.security.protocol = SASL_PLAINTEXT
client.sinks.kafka_sink.kafka.kerberos.domain.name = Kafka Domain名称，安全集群必填，如hadoop.xxx.com
client.sinks.kafka sink.requiredAcks = 0

client.sources.static log source.channels = static log channel
client.sinks.kafka_sink.channel = static_log_channel
```

说明

- client.sinks.kafka_sink.kafka.topic：数据写入的 topic。若 kafka 中该 topic 不存在，默认情况下会自动创建该 topic。
- client.sinks.kafka_sink.kafka.bootstrap.servers：Kafkabrokers 列表，多个用英文逗号分隔。默认情况下，安全集群端口 21007，普通集群对应端口 9092。
- client.sinks.kafka_sink.kafka.security.protocol：安全集群为 SASL_PLAINTEXT，普通集群为 PLAINTEXT。
- client.sinks.kafka_sink.kafka.kerberos.domain.name：

  普通集群无需配置此参数。安全集群对应此参数的值为 Kafka 集群中“kerberos.domain.name”对应的值。

  具体可到 Broker 实例所在节点上查看 ${BIGDATA_HOME}/MRS_Current/1_*X*_Broker/etc/server.properties。

  其中 X 为随机生成的数字，请根据实际情况修改。同时文件需要以 Flume 客户端安装用户身份保存，例如 **root** 用户。

  具体可到 Broker 实例所在节点上查看“${BIGDATA_HOME}/FusionInsight_Current/1_X_Broker/etc/server.properties”。

步骤 9 参数配置并保存后，Flume 客户端将自动加载“properties.properties”中配置的内容。当 spoolDir 生成新的日志文件，文件内容将发送到 Kafka 生产者，并支持 Kafka 消费者消费。

----结束

使用 Flume 客户端（MRS 3.x 及之后版本）

说明

普通集群不需要执行步骤 1-步骤 5。

步骤 1 将 Master1 节点上的认证服务器配置文件，复制到安装 Flume 客户端的节点，保存到 Flume 客户端中 *Flume 客户端安装目录*/fusioninsight-flume-*Flume 组件版本号*/conf 目录下。

文件完整路径为“${BIGDATA_HOME}/FusionInsight_BASE_*XXX*/1_*X*_KerberosClient/etc/kdc.conf”。其中“XXX”为产品版本号，“X”为随机生成的数字，请根据实际情况修改。同时文件需要以 Flume 客户端安装用户身份保存，例如 **root** 用户。

步骤 2 查看任一部署 Flume 角色节点的“业务 IP”。

登录 FusionInsight Manager 页面，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本），选择“集群 > 服务 > Flume > 实例”。查看任一部署 Flume 角色节点的“业务 IP”。

说明

若集群详情页面没有“组件管理”页签，请先完成 IAM 用户同步（在集群详情页的“概览”页签，单击“IAM 用户同步”右侧的“同步”进行 IAM 用户同步）。

步骤 3 将此节点上的用户认证文件，复制到安装 Flume 客户端的节点，保存到 Flume 客户端中“Flume 客户端安装目录/fusioninsight-flume-*Flume 组件版本号*/conf”目录下。

文件完整路径为${BIGDATA_HOME}/FusionInsight_Porter_*XXX*/install/FusionInsight-Flume-*Flume 组件版本号*/flume/conf/flume.keytab。

其中“XXX”为产品版本号，请根据实际情况修改。同时文件需要以 Flume 客户端安装用户身份保存，例如 **root** 用户。

步骤 4 将此节点上的配置文件“jaas.conf”，复制到安装 Flume 客户端的节点，保存到 Flume 客户端中“conf”目录。

文件完整路径为${BIGDATA_HOME}/FusionInsight_Current/1_*X*_Flume/etc/jaas.conf。

其中“X”为随机生成的数字，请根据实际情况修改。同时文件需要以 Flume 客户端安装用户身份保存，例如 **root** 用户。

步骤 5 登录安装 Flume 客户端节点，切换到客户端安装目录，执行以下命令修改文件：

vi conf/jaas.conf

修改参数“keyTab”定义的用户认证文件完整路径即步骤 3 中保存用户认证文件的目录：“Flume 客户端安装目录/fusioninsight-flume-*Flume 组件版本号*/conf”，然后保存并退出。

步骤 6 执行以下命令，修改 Flume 客户端配置文件“flume-env.sh”：

vi *Flume 客户端安装目录*/**fusioninsight-flume-***Flume 组件版本号*/**conf/flume-env.sh**

在“-XX:+UseCMSCompactAtFullCollection”后面，增加以下内容：

```
-Djava.security.krb5.conf=Flume 客户端安装目录/fusioninsight-flume-1.9.0/conf/kdc.conf
-Djava.security.auth.login.config=Flume 客户端安装目录/fusioninsight-flume-
1.9.0/conf/jaas.conf -Dzookeeper.request.timeout=120000
```

例如：**"-XX:+UseCMSCompactAtFullCollection -Djava.security.krb5.conf=/opt/FlumeClient/fusioninsight-flume-***Flume 组件版本号***/conf/kdc.conf -Djava.security.auth.login.config=/opt/FlumeClient/fusioninsight-flume-***Flume 组件版本号***/conf/jaas.conf -Dzookeeper.request.timeout=120000"**

请根据实际情况，修改“Flume 客户端安装目录”，然后保存并退出。

步骤 7 执行以下命令，重启 Flume 客户端：

cd *Flume 客户端安装目录*/**fusioninsight-flume-***Flume 组件版本号*/**bin**

./flume-manage.sh restart

例如：

cd /opt/FlumeClient/fusioninsight-flume-*Flume 组件版本号*/**bin**

./flume-manage.sh restart

步骤 8 根据实际业务场景配置作业。

- MRS 3.x 及之后版本部分参数可直接在 Manager 界面配置。
- 在“properties.properties”文件中配置，以配置 SpoolDir Source+File Channel+Kafka Sink 为例。

  在安装 Flume 客户端的节点执行以下命令，根据实际业务需求，在 Flume 客户端配置文件“properties.properties”中配置并保存作业。

  vi *Flume 客户端安装目录*/**fusioninsight-flume-**_Flume 组件版本号_**/conf/properties.properties**

```
################################################################################
##########
client.sources = static_log_source
client.channels = static_log_channel
client.sinks = kafka_sink
################################################################################
##########
#LOG_TO_HDFS_ONLINE_1

client.sources.static_log_source.type = spooldir
client.sources.static_log_source.spoolDir = 监控目录
client.sources.static_log_source.fileSuffix = .COMPLETED
client.sources.static_log_source.ignorePattern = ^$
client.sources.static_log_source.trackerDir = 传输过程中元数据存储路径
client.sources.static_log_source.maxBlobLength = 16384
client.sources.static_log_source.batchSize = 51200
client.sources.static_log_source.inputCharset = UTF-8
client.sources.static_log_source.deserializer = LINE
client.sources.static_log_source.selector.type = replicating
client.sources.static_log_source.fileHeaderKey = file
client.sources.static_log_source.fileHeader = false
client.sources.static_log_source.basenameHeader = true
client.sources.static_log_source.basenameHeaderKey = basename
client.sources.static_log_source.deletePolicy = never

client.channels.static_log_channel.type = file
client.channels.static log channel.dataDirs = 数据缓存路径，设置多个路径可提升性能，
中间用逗号分开
client.channels.static log channel.checkpointDir = 检查点存放路径
client.channels.static log channel.maxFileSize = 2146435071
client.channels.static log channel.capacity = 1000000
client.channels.static log channel.transactionCapacity = 612000
client.channels.static log channel.minimumRequiredSpace = 524288000

client.sinks.kafka sink.type = org.apache.flume.sink.kafka.KafkaSink
client.sinks.kafka sink.kafka.topic = 数据写入的topic，如flume test
client.sinks.kafka sink.kafka.bootstrap.servers = XXX.XXX.XXX.XXX:kafka端口
号,XXX.XXX.XXX.XXX:kafka端口号,XXX.XXX.XXX.XXX:kafka端口号
client.sinks.kafka_sink.flumeBatchSize = 1000
client.sinks.kafka_sink.kafka.producer.type = sync
client.sinks.kafka_sink.kafka.security.protocol = SASL_PLAINTEXT
client.sinks.kafka_sink.kafka.kerberos.domain.name = Kafka Domain名称，安全集群必
填，如hadoop.xxx.com
client.sinks.kafka_sink.requiredAcks = 0
```

```
client.sources.static_log_source.channels = static_log_channel
client.sinks.kafka_sink.channel = static_log_channel
```

说明

- client.sinks.kafka_sink.kafka.topic：数据写入的 topic。若 kafka 中该 topic 不存在，默认情况下会自动创建该 topic。
- client.sinks.kafka_sink.kafka.bootstrap.servers：Kafkabrokers 列表，多个用英文逗号分隔。默认情况下，安全集群端口 21007，普通集群对应端口 9092。
- client.sinks.kafka_sink.kafka.security.protocol：安全集群为 SASL_PLAINTEXT，普通集群为 PLAINTEXT。
- client.sinks.kafka_sink.kafka.kerberos.domain.name：

  普通集群无需配置此参数。安全集群对应此参数的值为 Kafka 集群中“kerberos.domain.name”对应的值。

  具体可到 Broker 实例所在节点上查看 ${BIGDATA_HOME}/MRS_Current/1_*X*_Broker/etc/server.properties。

  其中 X 为随机生成的数字，请根据实际情况修改。同时文件需要以 Flume 客户端安装用户身份保存，例如 **root** 用户。

  具体可到 Broker 实例所在节点上查看“${BIGDATA_HOME}/FusionInsight_Current/1_X_Broker/etc/server.properties”。

步骤 9 参数配置并保存后，Flume 客户端将自动加载“properties.properties”中配置的内容。当 spoolDir 生成新的日志文件，文件内容将发送到 Kafka 生产者，并支持 Kafka 消费者消费。

----结束

7.2 使用简介

Flume 是一个分布式、可靠和高可用的海量日志聚合的系统。它能够将不同数据源的海量日志数据进行高效收集、聚合、移动，最后存储到一个中心化数据存储系统中。支持在系统中定制各类数据发送方，用于收集数据。同时，提供对数据进行简单处理，并写到各种数据接受方（可定制）的能力。

Flume 分为客户端和服务端，两者都是 FlumeAgent。服务端对应着 FlumeServer 实例，直接部署在集群内部。而客户端部署更灵活，可以部署在集群内部，也可以部署在集群外。它们之间没有必然联系，都可以独立工作，并且提供的功能是一样的。

Flume 客户端需要单独安装，支持将数据直接导到集群中的 HDFS 和 Kafka 等组件上，也可以结合 Flume 服务端一起使用。

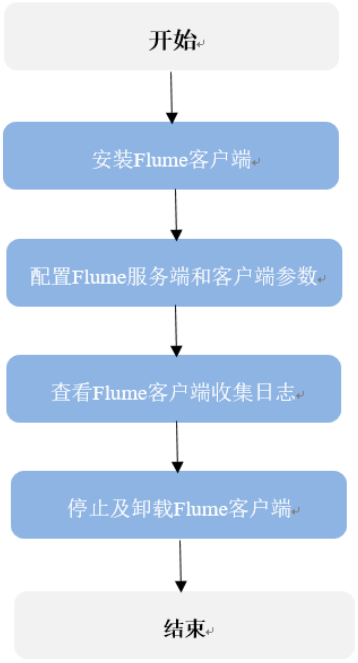
使用流程

通过 Flume 采集日志的流程如下所示。

1. 安装 Flume 客户端。

2. 配置 Flume 服务端和客户端参数。
3. 查看 Flume 客户端收集日志。
4. 停止及卸载 Flume 客户端。

图7-1 Flume 使用流程



Flume 客户端介绍

Flume 客户端由 Source、Channel、Sink 组成，数据先进入 Source 然后传递到 Channel，最后由 Sink 发送到客户端外部。各模块说明见表 7-1。

表7-1 模块说明

名称	说明
Source	Source 负责接收数据或产生数据，并将数据批量放到一个或多个 Channel。Source 有两种类型：数据驱动和轮询。 典型的 Source 样例如下： • 和系统集成并接收数据的 Sources：Syslog、Netcat。 • 自动生成事件数据的 Sources：Exec、SEQ。 • 用于 Agent 和 Agent 之间通信的 IPC Sources：Avro。 Source 必须至少和一个 Channel 关联。
Channel	Channel 位于 Source 和 Sink 之间，用于缓存 Source 传递的数据，当 Sink 成功将数据发送到下一跳的 Channel 或最终数据处理端，缓存数据将自动从 Channel 移除。

名称	说明
	不同类型的 Channel 提供的持久化水平也是不一样的： • Memory Channel：非持久化 • File Channel：基于预写式日志（Write-Ahead Logging，简称 WAL）的持久化实现 • JDBC Channel：基于嵌入 Database 的持久化实现 Channel 支持事务特性，可保证简易的顺序操作，同时可以配合任意数量的 Source 和 Sink 共同工作。
Sink	Sink 负责将数据传输到下一跳或最终目的，成功完成后将数据从 Channel 移除。 典型的 Sink 样例如下： • 存储数据到最终目的终端 Sink，比如：HDFS、Kafka • 自动消耗的 Sinks，比如：Null Sink • 用于 Agent 和 Agent 之间通信的 IPC sink：Avro Sink 必须关联到一个 Channel。

Flume 客户端可以配置成多个 Source、Channel、Sink，即一个 Source 将数据发送给多个 Channel，再由多个 Sink 发送到客户端外部。

Flume 还支持多个 Flume 客户端配置级联，即 Sink 将数据再发送给 Source。

补充说明

1. Flume 可靠性保障措施。
   - Source 与 Channel、Channel 与 Sink 之间支持事务机制。
   - Sink Processor 支持配置 failover、load_balance 机制。

     例如 load_balance 示例如下：

     ```
     server.sinkgroups=g1
     server.sinkgroups.g1.sinks=k1 k2
     server.sinkgroups.g1.processor.type=load_balance
     server.sinkgroups.g1.processor.backoff=true
     server.sinkgroups.g1.processor.selector=random
     ```
2. Flume 多客户端聚合级联时的注意事项。
   - 级联时需要走 Avro 或者 Thrift 协议进行级联。
   - 聚合端存在多个节点时，连接配置尽量配置均衡，不要聚合到单节点上。
3. Flume 客户端可以包含多个独立的数据流，即在一个配置文件 properties.properties 中配置多个 Source、Channel、Sink。这些组件可以链接以形成多个流。

   例如在一个配置中配置两个数据流，示例如下：

   ```
   server.sources = source1 source2
   server.sinks = sink1 sink2
   server.channels = channel1 channel2
   ```

```
#dataflow1
server.sources.source1.channels = channel1
server.sinks.sink1.channel = channel1

#dataflow2
server.sources.source2.channels = channel2
server.sinks.sink2.channel = channel2
```

7.3 安装 Flume 客户端

7.3.1 安装 MRS 3.x 之前版本 Flume 客户端

操作场景

使用 Flume 搜集日志时，需要在日志主机上安装 Flume 客户端。用户可以创建一个新的 ECS 并安装 Flume 客户端。

本章节适用于 MRS 3.x 之前版本。

前提条件

- 已创建包含 Flume 组件的流集群。
- 日志主机需要与 MRS 集群在相同的 VPC 和子网。
- 已获取日志主机的登录方式。

操作步骤

步骤 1 根据前提条件，创建一个满足要求的弹性云主机。

步骤 2 登录集群详情页面，选择“组件管理”。

说明

若集群详情页面没有“组件管理”页签，请先完成 IAM 用户同步（在集群详情页的“概览”页签，单击“IAM 用户同步”右侧的“同步”进行 IAM 用户同步）。

步骤 3 单击“下载客户端”。

1. 在“客户端类型”选择“完整客户端”。
2. 在“下载路径”选择“远端主机”。
3. 将“主机 IP”设置为 ECS 的 IP 地址，设置“主机端口”为“22”，并将“保存路径”设置为“/tmp”。
   - 如果使用 SSH 登录 ECS 的默认端口“22”被修改，请将“主机端口”设置为新端口。
   - “保存路径”最多可以包含 256 个字符。
4. “登录用户”设置为“root”。

   如果使用其他用户，请确保该用户对保存目录拥有读取、写入和执行权限。
5. 在“登录方式”选择“密码”或“SSH 私钥”。

- 密码：输入创建集群时设置的 root 用户密码。
- SSH 私钥：选择并上传创建集群时使用的密钥文件。

6. 单击“确定”开始生成客户端文件。

   若界面显示以下提示信息表示客户端包已经成功保存。

   ```
   下载客户端文件到远端主机成功。
   ```

   若界面显示以下提示信息，请检查用户名密码及远端主机的安全组配置，确保用户名密码正确，及远端主机的安全组已增加 SSH(22)端口的入方向规则。然后从步骤 3 执行重新开始下载客户端。

   ```
   连接到服务器失败，请检查网络连接或参数设置。
   ```

步骤 4 选择“Flume”服务，单击“实例”，查看任意一个 Flume 实例和两个 MonitorServer 实例的“业务 IP”。

步骤 5 使用 VNC 方式，登录弹性云主机。参见弹性云主机《用户指南》的**远程登录（VNC 方式）**章节（“实例 > 登录 Linux 弹性云主机 > 远程登录（VNC 方式）”）。

所有镜像均支持 Cloud-init 特性。Cloud-init 预配置的用户名“root”，密码为创建集群时设置的密码。首次登录建议修改。

步骤 6 在弹性云主机，切换到 **root** 用户，并将安装包复制到目录“/opt”。

sudo su - root

cp /tmp/MRS_Flume_Client.tar /opt

步骤 7 在“/opt”目录执行以下命令，解压压缩包获取校验文件与客户端配置包。

tar -xvf MRS_Flume_Client.tar

步骤 8 执行以下命令，校验文件包。

sha256sum -c MRS_Flume_ClientConfig.tar.sha256

界面显示如下信息，表明文件包校验成功：

```
MRS_Flume_ClientConfig.tar: OK
```

步骤 9 执行以下命令，解压“MRS_Flume_ClientConfig.tar”。

tar -xvf MRS_Flume_ClientConfig.tar

步骤 10 执行以下命令，安装客户端运行环境到新的目录，例如“/opt/Flumeenv”。安装时自动生成目录。

sh /opt/MRS_Flume_ClientConfig/install.sh /opt/Flumeenv

查看安装输出信息，如有以下结果表示客户端运行环境安装成功：

```
Components client installation is complete.
```

步骤 11 执行以下命令，配置环境变量。

source /opt/Flumeenv/bigdata_env

步骤 12 执行以下命令，解压 Flume 客户端。

cd /opt/MRS_Flume_ClientConfig/Flume

tar -xvf FusionInsight-Flume-1.6.0.tar.gz

步骤 13 执行以下命令，查看当前用户密码是否过期。

chage -l root

“Password expires”时间早于当前则表示过期。此时需要修改密码，或执行 **chage -M -1 root** 设置密码为未过期状态。

步骤 14 执行以下命令，安装 Flume 客户端到新目录，例如“/opt/FlumeClient”。安装时自动生成目录。

sh /opt/MRS_Flume_ClientConfig/Flume/install.sh -d /opt/FlumeClient -f *MonitorServer 实例的业务 IP 地址* -c Flume 配置文件路径 -l /var/log/ -e *Flume 的业务 IP 地址* -n *Flume 客户端名称*

各参数说明如下：

- “-d”：表示 Flume 客户端安装路径。
- “-f”：可选参数，表示两个 MonitorServer 角色的业务 IP 地址，中间用英文逗号分隔，若不设置则 Flume 客户端将不向 MonitorServer 发送告警信息，同时在 MRS Manager 界面上看不到该客户端的相关信息。
- “-c”：可选参数，表示 Flume 客户端在安装后默认加载的配置文件“properties.properties”。如不添加参数，默认使用客户端安装目录的“fusioninsight-flume-1.6.0/conf/properties.properties”。客户端中配置文件为空白模板，根据业务需要修改后 Flume 客户端将自动加载。
- “-l”：可选参数，表示日志目录，默认值为“/var/log/Bigdata”。
- “-e”：可选参数，表示 Flume 实例的业务 IP 地址，主要用于接收客户端上报的监控指标信息。
- “-n”：可选参数，表示自定义的 Flume 客户端的名称。
- IBM 的 JDK 不支持“-Xloggc”，需要修改“flume/conf/flume-env.sh”，将“-Xloggc”修改为“-Xverbosegclog”，若 JDK 为 32 位，“-Xmx”不能大于 3.25GB。
- “flume/conf/flume-env.sh”中，“-Xmx”默认为 4GB。若客户端机器内存过小，可调整为 512M 甚至 1GB。

例如执行：**sh install.sh -d /opt/FlumeClient**

系统显示以下结果表示客户端运行环境安装成功：

```
install flume client successfully.
```

----结束

7.3.2 安装 MRS 3.x 及之后版本 Flume 客户端

操作场景

使用 Flume 搜集日志时，需要在日志主机上安装 Flume 客户端。用户可以创建一个新的 ECS 并安装 Flume 客户端。

本章节适用于 MRS 3.x 及之后版本。

前提条件

- 已创建包含 Flume 组件的集群。
- 日志主机需要与 MRS 集群在相同的 VPC 和子网。
- 已获取日志主机的登录方式。
- 安装目录可以不存在，会自动创建。但如果存在，则必须为空。目录路径不能包含空格。

操作步骤

步骤 1 获取软件包。

登录 FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > Flume”进入 Flume 服务界面，在右上角选择“更多 > 下载客户端”，选择“选择客户端类型”为“完整客户端”，下载 Flume 服务客户端文件。

客户端文件名称为“FusionInsight_Cluster_<*集群 ID*>_Flume_Client.tar”，本章节以“FusionInsight_Cluster_1_Flume_Client.tar”为例进行描述。

步骤 2 上传软件包。

以 **user** 用户将软件包上传到将要安装 Flume 服务客户端的节点目录上，例如“/opt/client”。

说明

user 用户为安装和运行 Flume 客户端的用户。

步骤 3 解压软件包。

以 **user** 用户登录将要安装 Flume 服务客户端的节点。进入安装包所在目录，例如“/opt/client”，执行如下命令解压安装包到当前目录。

cd /opt/client

tar -xvf FusionInsight_Cluster_1_Flume_Client.tar

步骤 4 校验软件包。

执行 **sha256sum -c** 命令校验解压得到的文件，返回“OK”表示校验通过。例如：

sha256sum -c FusionInsight_Cluster_1_Flume_ClientConfig.tar.sha256

```
FusionInsight_Cluster_1_Flume_ClientConfig.tar: OK
```

步骤 5 解压文件。

tar -xvf FusionInsight_Cluster_1_Flume_ClientConfig.tar

步骤 6 在 Flume 客户端安装目录下执行以下命令，安装客户端到指定目录（绝对路径），例如安装到“/opt/FlumeClient”目录。客户端安装成功后安装结束。

cd /opt/client/FusionInsight_Cluster_1_Flume_ClientConfig/Flume/FlumeClient

./install.sh -d */opt/FlumeClient* **-f** *MonitorServer 角色的业务 IP 或主机名* **-c** *用户业务配置文件 properties.properties 放置路径* **-s** *cpu 阈值* **-l /var/log/Bigdata -e** *FlumeServer 的业务 IP 或主机名* **-n** *Flume*

说明

- “-d”：Flume 客户端安装路径。
- “-f”（可选）：两个 MonitorServer 角色的业务 IP 或主机名，中间用逗号分隔，若不设置则 Flume 客户端将不向 MonitorServer 发送告警信息，同时在 FusionInsight Manager 界面上看不到该客户端的相关信息。
- “-c”（可选）：指定业务配置文件，该文件需要用户根据自己业务生成，具体操作可在 Flume 服务端中“配置工具”页面参考 Flume 业务配置指南章节生成，并上传到待安装客户端节点上的任一目录下。若安装时未指定（即不配置该参数），可在安装后上传已经生成的业务配置文件 properties.properties 到“/opt/FlumeClient/fusioninsight-flume-1.9.0/conf”目录下。
- “-s”（可选）：Cgroup 阈值，阈值取值范围为 1~100*N 之间的整数，N 表示机器 cpu 核数。默认阈值为“-1”，表示加入到 Cgroup 的进程不受 cpu 使用率限制。
- “-l”（可选）：日志路径，默认值为“/var/log/Bigdata”（“user”用户需要对此目录有写权限）。首次安装客户端会生成名为 flume-client 的子目录，之后安装会依次生成名为“flume-client-*n*”的子目录，n 代表一个序号，从 1 依次递增。在 Flume 客户端安装目录下的 conf 目录中，编辑 ENV_VARS 文件，搜索 FLUME_LOG_DIR 属性，可查看客户端日志路径。
- “-e”（可选）：FlumeServer 的业务 IP 地址或主机名，主要用于接收客户端上报的监控指标信息。
- “-n”（可选）：Flume 客户端的名称，可以通过在 FusionInsight Manager 上选择“集群 > *待操作集群名称* > 服务 > Flume > Flume 管理”查看对应节点上客户端的名称。
- 若产生以下错误提示，可执行命令 **export JAVA_HOME**=*JDK 路径*进行处理。

  ```
  JAVA_HOME is null in current user,please install the JDK and set the JAVA_HOME
  ```

- IBM 的 JDK 不支持“-Xloggc”，需要修改“flume/conf/flume-env.sh”，将“-Xloggc”修改为“-Xverbosegclog”，若 JDK 为 32 位，“-Xmx”不能大于 3.25GB。
- 集群混搭时，安装跨平台客户端时，请进入 /opt/client/FusionInsight_Cluster_1_Flume_ClientConfig/Flume/FusionInsight-Flume-1.9.0.tar.gz 路径下进行 Flume 客户端安装。

----结束

7.4 查看 Flume 客户端日志

操作场景

查看日志以便定位问题。

前提条件

Flume 客户端已经正确安装。

操作步骤

步骤 1 进入 Flume 客户端日志目录，默认为“/var/log/Bigdata”。

步骤 2 执行如下命令查看日志文件列表。

ls -lR flume-client-*

日志文件示例如下：

```
flume-client-1/flume:
total 7672
-rw-------. 1 root root      0 Sep  8 19:43 Flume-audit.log
-rw-------. 1 root root 1562037 Sep 11 06:05 FlumeClient.2017-09-11_04-05-
09.[1].log.zip
-rw-------. 1 root root 6127274 Sep 11 14:47 FlumeClient.log
-rw-------. 1 root root   2935 Sep  8 22:20 flume-root-20170908202009-pid72456-
gc.log.0.current
-rw-------. 1 root root   2935 Sep  8 22:27 flume-root-20170908202634-pid78789-
gc.log.0.current
-rw-------. 1 root root   4382 Sep  8 22:47 flume-root-20170908203137-pid84925-
gc.log.0.current
-rw-------. 1 root root   4390 Sep  8 23:46 flume-root-20170908204918-pid103920-
gc.log.0.current
-rw-------. 1 root root   3196 Sep  9 10:12 flume-root-20170908215351-pid44372-
gc.log.0.current
-rw-------. 1 root root   2935 Sep  9 10:13 flume-root-20170909101233-pid55119-
gc.log.0.current
-rw-------. 1 root root   6441 Sep  9 11:10 flume-root-20170909101631-pid59301-
gc.log.0.current
-rw-------. 1 root root      0 Sep  9 11:10 flume-root-20170909111009-pid119477-
gc.log.0.current
-rw-------. 1 root root  92896 Sep 11 13:24 flume-root-20170909111126-pid120689-
gc.log.0.current
-rw-------. 1 root root   5588 Sep 11 14:46 flume-root-20170911132445-pid42259-
gc.log.0.current
-rw-------. 1 root root   2576 Sep 11 13:24 prestartDetail.log
-rw-------. 1 root root   3303 Sep 11 13:24 startDetail.log
-rw-------. 1 root root   1253 Sep 11 13:24 stopDetail.log

flume-client-1/monitor:
total 8
-rw-------. 1 root root  141 Sep  8 19:43 flumeMonitorChecker.log
-rw-------. 1 root root 2946 Sep 11 13:24 flumeMonitor.log
```

其中 **FlumeClient.log** 即为 Flume 客户端的运行日志。

----结束

7.5 停止或卸载 Flume 客户端

操作场景

指导运维工程师停止、启动 Flume 客户端，以及在不需要 Flume 数据采集通道时，卸载 Flume 客户端。

操作步骤

- 停止 Flume 角色的客户端。

  假设 Flume 客户端安装路径为“/opt/FlumeClient”，执行以下命令，停止 Flume 客户端：

  cd /opt/FlumeClient/fusioninsight-flume-*Flume 组件版本号*/bin

  ./flume-manage.sh stop

  执行脚本后，显示如下信息，说明成功的停止了 Flume 客户端：

  ```
  Stop Flume PID=120689 successful..
  ```

  说明

  Flume 客户端停止后会自动重启，如果不需自动重启，请执行以下命令：

  ./flume-manage.sh stop force

  需要启动时，可执行以下命令：

  ./flume-manage.sh start force

- 卸载 Flume 角色的客户端。

  假设 Flume 客户端安装路径为“/opt/FlumeClient”，执行以下命令，卸载 Flume 客户端：

  cd /opt/FlumeClient/fusioninsight-flume-*Flume 组件版本号*/inst

  ./uninstall.sh

7.6 使用 Flume 客户端加密工具

操作场景

安装 Flume 客户端后，配置文件的部分参数可能需要填写加密的字符，Flume 客户端中提供了加密工具。

前提条件

已完成客户端安装。

操作步骤

步骤 1 登录安装 Flume 客户端的节点，并切换到客户端安装目录。例如“/opt/FlumeClient”。

步骤 2 切换到以下目录

cd fusioninsight-flume-*Flume 组件版本号*/bin

步骤 3 执行以下命令，加密原始信息：

./genPwFile.sh

输入两次待加密信息。

步骤 4 执行以下命令，查看加密后的信息：

cat password.property

说明

如果加密参数是用于 Flume Server，那么需要到相应的 Flume Server 所在节点执行加密。需要使用 omm 用户执行加密脚本进行加密。

- 针对 MRS 3.x 之前版本加密路径为“/opt/Bigdata/MRS_XXX/install/FusionInsight-Flume-*Flume 组件版本号*/flume/bin/genPwFile.sh”。
- 针对 MRS 3.x 及之后版本加密路径为“/opt/Bigdata/FusionInsight_Porter_XXX/install/FusionInsight-Flume-*Flume 组件版本号*/flume/bin/genPwFile.sh”。其中 XXX 为产品的版本号。

----结束

7.7 Flume 业务配置指南

本章节适用于 MRS 3.x 及之后版本。

该操作指导用户完成 Flume 常用业务的配置。其他一些不太常用的 Source、Channel、Sink 的配置请参考 Flume 社区提供的用户手册（http://flume.apache.org/releases/1.9.0.html）。

说明

- 各个表格中所示参数，黑体加粗的参数为必选参数。
- Sink 的 BatchSize 参数必须小于 Channel 的 transactionCapacity。
- 集群 Flume 配置工具界面篇幅有限，Source、Channel、Sink 只展示部分参数，详细请参考如下常用配置。
- 集群 Flume 配置工具界面上所展示 Customer Source、Customer Channel 及 Customer Sink 需要用户根据自己开发的代码来进行配置，下述常用配置不再展示。

常用 Source 配置

- **Avro Source**

  Avro Source 监听 Avro 端口，接收外部 Avro 客户端数据并放入配置的 Channel 中。常用配置如下表所示：

表7-2 Avro Source 常用配置

参数	默认值	描述
channels	-	与之相连的 channel，可以配置多个。
type	avro	avro source 的类型，必须为 avro。
bind	-	监听主机名/IP。
port	-	绑定监听端口，该端口需未被占用。
threads	-	source 工作的最大线程数。
compression-type	none	消息压缩格式：“none”或“deflate”。“none”表示不压缩，“deflate”表示压缩。
compression-level	6	数据压缩级别（1-9），数值越高，压缩率越高。
ssl	false	是否使用 SSL 加密。设置为 true 时还必须指定“秘钥(keystore)”和“秘钥存储密码(keystore-password)”。
truststore-type	JKS	Java 信任库类型，“JKS”或“PKCS12”。 说明 JKS 的秘钥库和私钥采用不同的密码进行保护，而 PKCS12 的秘钥库和私钥采用相同密码进行保护。
truststore	-	Java 信任库文件。
truststore-password	-	Java 信任库密码。
keystore-type	JKS	ssl 启用后密钥存储类型，“JKS”或“PKCS12”。 说明 JKS 的秘钥库和私钥用不同的密码进行保护，而 PKCS12 的秘钥库和私钥用相同密码进行保护。

参数	默认值	描述
keystore	-	ssl 启用后密钥存储文件路径，开启 ssl 后，该参数必填。
keystore-password	-	ssl 启用后密钥存储密码，开启 ssl 后，该参数必填。
trust-all-certs	false	是否关闭 SSL server 证书检查。设置为“true”时将不会检查远端 source 的 SSL server 证书，不建议在生产中使用。
exclude-protocols	SSLv3	排除的协议列表，用空格分开。默认排除 SSLv3 协议。
ipFilter	false	是否开启 ip 过滤。
ipFilter.rules	-	定义 N 网络的 ipFilters，多个主机或 IP 地址用逗号分割。ipFilter 设置为“true”时，配置规则有允许和禁止两种，配置格式如下： ipFilterRules=allow:ip:127.*, allow:name:localhost, deny:ip:*

- **SpoolDir Source**

  Spool Dir Source 监控并传输目录下新增的文件，可实现实时数据传输。常用配置如下表所示：

表7-3 Spooling Directory Source 常用配置

参数	默认值	描述
channels	-	与之相连的 channel，可以配置多个。
type	spooldir	spooling source 的类型，必须设置为 spooldir。
spoolDir	-	Spooldir source 的监控目录，flume 运行用户需要对该目录具有可读可写可执行权限。
monTime	0（不开启）	线程监控阈值，更新时间超过阈值后，重新启动该 Source，单位：秒。
fileSuffix	.COMPLETED	文件传输完成后添加的后缀。

参数	默认值	描述
deletePolicy	never	文件传输完成后源文件删除策略，never 或 immediate。“never”表示不删除已完成传输的源文件，“immediate”表示传输完成后立刻删除源文件。
ignorePattern	^$	忽略文件的正则表达式表示。默认为“^$”，表示忽略空格。
includePattern	^.*$	包含文件的正则表达式表示。可以与 ignorePattern 同时使用，如果一个文件既满足 ignorePattern 也满足 includePattern,则该文件会被忽略。另外，以“.”开头的文件不会被过滤。
trackerDir	.flumespool	传输过程中元数据存储路径。
batchSize	1000	批次写入 Channel 的 Event 数量。
decodeErrorPolicy	FAIL	编码错误策略。 说明 如果文件中有编码错误，请配置“decodeErrorPolicy”为“REPLACE”或“IGNORE”，Flume 遇到编码错误将跳过编码错误，继续采集后续日志。
deserializer	LINE	文件解析器，值为“LINE”或“BufferedLine”。 • 配置为“LINE”时，对从文件读取的字符逐个转码。 • 配置为“BufferedLine”时，对文件读取的一行或多行的字符进行批量转码，性能更优。
deserializer.maxLineLength	2048	按行解析最大长度。
deserializer.maxBatchLine	1	按行解析最多行数，如果行数设置为多行，maxLineLength 也应该设置为相应的倍数。 说明 用户设置 Interceptor 时，需要考虑多行合并后的场景，否则会造成数据丢失。如果 Interceptor 无法处理多行合并场景，请将该配置设置为 1。
selector.type	replicating	选择器类型，“replicating”或“multiplexing”。“replicating”表示将数据复制多份，分别传递给每一个 channel，每个 channel 接收到的数据都是相同的，而“multiplexing”表示根据 event 中 header 的 value 来选择特定的 channel，每个 channel 中的数据是不同的。
interceptors	-	拦截器。多个拦截器用空格分开。

参数	默认值	描述
inputCharset	UTF-8	读取文件的编码格式。须与读取数据源文件编码格式相同，否则字符解析可能会出错。
fileHeader	false	是否把文件名（包含路径）添加到 event 的 header 中。
fileHeaderKey	-	设置 header 中数据存储结构为<key,value>模式，需要 fileHeaderKey 与 fileHeader 配合使用。若 fileHeader 设置为 true，可参考如下示例。 示例：将 fileHeaderKey 定义为 file，当读取到文件名为/root/a.txt 的内容时，header 中以 file=/root/a.txt 的形式存在。
basenameHeader	false	是否把文件名（不包含路径）添加到 event 的 header 中。
basenameHeaderKey	-	设置 header 中数据存储结构为<key,value>模式，需要 basenameHeaderKey 与 basenameHeader 配合使用。若 basenameHeader 设置为 true，可参考如下示例。 示例：将 basenameHeaderKey 定义为 file，当读取到文件名为 a.txt 的内容时，header 中以 file=a.txt 的形式存在。
pollDelay	500	轮询监控目录下新文件时的时延。单位：毫秒。
recursiveDirectorySearch	false	是否监控配置的目录下子目录中的新文件。
consumeOrder	oldest	监控目录下文件的消耗次序。如果配置为 oldest 或者 youngest，会根据监控目录下文件的最后修改时间来决定，当目录下有大量文件时，会消耗较长时间去寻找 oldest 或者 youngest 的文件。需要注意的是，如果配置为 random，创建比较早的文件有可能长时间未被读取。如果配置为 oldest 或者 youngest，那么进程会需要较多时间来查找最新的或最旧的文件。可选值：random，youngest，oldest。
maxBackoff	4000	当 Channel 满了以后，尝试再次去写 Channel 所等待的最大时间。超过这个时间，则会抛出异常。对应的 Source 会以一个较小的时间开始，然后每尝试一次，该时间数字指数增长直到达到当前指定的值，如果还不能成功写入，则认为失败。时间单位：秒。
emptyFileEvent	true	是否采集空文件信息发送到 Sink 端，默认值为 true，表示将空文件信息发送到 Sink 端。该参数只对 HDFS Sink 有效，其他 Sink 该参数无效。以 HDFS Sink 为例，当参数为 true 时，如果 spoolDir

参数	默认值	描述
		路径下存在空文件，那么 HDFS 的 hdfs.path 路径下就会创建一个同名的空文件。

说明

SpoolDir Source 在按行读取过程中会忽略掉每一个 event 的最后一个换行符，该换行符所占用的数据量指标不会被 Flume 统计。

- **Kafka Source**

  Kafka Source 从 Kafka 的 topic 中消费数据，可以设置多个 Source 消费同一个 topic 的数据，每个 Source 会消费 topic 的不同 partitions。常用配置如下表所示：

表7-4 Kafka Source 常用配置

参数	默认值	描述
channels	-	与之相连的 channel，可以配置多个。
type	org.apache.flume.source.kafka.KafkaSource	kafka source 的类型，必须设置为 org.apache.flume.source.kafka.KafkaSource。
kafka.bootstrap.servers	-	Kafka 的 bootstrap 地址端口列表。如果集群已安装 Kafka 并且配置已经同步，服务端可以不配置此项，默认值为 Kafka 集群中所有的 broker 列表。客户端必须配置该项，多个值用逗号分隔。端口和安全协议的匹配规则必须为：21007 匹配安全模式（SASL_PLAINTEXT），9092 匹配普通模式（PLAINTEXT）。
kafka.topics	-	订阅的 Kafka topic 列表，用逗号分隔。
kafka.topics.regex	-	符合正则表达式的 topic 会被订阅，优先级高于“kafka.topics”，如果存在将覆盖“kafka.topics”。
monTime	0（不开启）	线程监控阈值，更新时间超过阈值后，重新启动该 Source，单位：秒。
nodatatime	0（不开启）	告警阈值，从 Kafka 中订阅不到数据的时长超过阈值时发送告警，单位：秒。该参数可在配置文件 properties.properties 进行设置。
batchSize	1000	批次写入 Channel 的 Event 数量。
batchDurationMil	1000	批次消费 topic 数据的最大时长，单

参数	默认值	描述
lis		位：ms。
keepTopicInHeader	false	是否在 Event Header 中保存 topic。设置为 true，则 Kafka Sink 配置的 topic 将无效。
setTopicHeader	true	当设置为 true 时，会将“topicHeader”中定义的 topic 名称存储到 Header 中。
topicHeader	topic	当 setTopicHeader 属性设置为 true，此参数用于定义存储接收的 topic 名称。如果与 Kafka Sink 的 topicHeader 属性结合使用，应该注意，避免将消息循环发送到同一主题。
useFlumeEventFormat	false	默认情况下，event 会以字节的形式从 kafka topic 传递到 event 的 body 体中。设置为 true，则会以 Flume 的 Avro 二进制格式来读取 Event。与 KafkaSink 或 KakfaChannel 中同名的 parseAsFlumeEvent 参数一起使用时，会保留从数据源产生的任何设定的 Header。
keepPartitionInHeader	false	是否在 Event Header 中保存 partitionID。设置为 true，则 Kafka Sink 将写入对应的 Partition。
kafka.consumer.group.id	flume	Kafka 消费组 ID。多个源或代理中设置相同的 ID 表示它们是同一个 consumer group。
kafka.security.protocol	SASL_PLAINTEXT	Kafka 安全协议，普通模式集群下须配置为“PLAINTEXT”。端口和安全协议的匹配规则必须为：21007 匹配安全模式（SASL_PLAINTEXT），9092 匹配普通模式（PLAINTEXT）。
Other Kafka Consumer Properties	-	其他 Kafka 配置，可以接受任意 Kafka 支持的消费配置，配置需要加前缀“kafka.”。

- **Taildir Source**

  Taildir Source 监控目录下文件的变化并自动读取文件内容，可实现实时数据传输，常用配置如下表所示：

表7-5 Taildir Source 常用配置

参数	默认值	描述
channels	-	与之相连的 channel，可以配置多个。
type	TAILDIR	taildir source 的类型，必须为 TAILDIR。
filegroups	-	设置采集文件目录分组名字，分组名字中间使用空格间隔。
filegroups.<filegroupName>.parentDir	-	父目录，需要配置为绝对路径。
filegroups.<filegroupName>.filePattern	-	相对父目录的文件路径，可以包含目录，支持正则表达式，须与父目录联合使用。
positionFile	-	传输过程中元数据存储路径。
headers.<filegroupName>.<headerKey>	-	设置某一个分组采集数据时 event 中的 key-value 值。
byteOffsetHeader	false	是否在每一个 event 头中携带该 event 在源文件中的位置信息。设置为 true，则该信息保存在 byteoffset 变量中。
maxBatchCount	Long.MAX_VALUE	控制从一个文件中连续读取的最大批次。如果监控目录会一直读取多个文件，且其中一个文件以非常快的速率在写入，那么其他文件可能会无法处理。因为高速写入的这个文件会陷入无限读取的循环中。这种情况下，应该降低此值。
skipToEnd	false	Flume 在重启后是否直接定位到文件最新的位置处读取最新的数据。设置为 true，则重启后直接定位到文件最新位置读取最新数据。
idleTimeout	120000	设置读取文件的空闲时间，单位：毫秒，如果在该时间内文件内容没有变更，关闭掉该文件，关闭后如果该文件有数据写入，重新打开并读取数据。
writePosInterval	3000	设置将元数据写入到文件的周期，单位：毫秒。
batchSize	1000	批次写入 Channel 的 Event 数量。
monTime	0（不开启）	线程监控阈值，更新时间超过阈值后，重新启动该 Source，单位：秒。
fileHeader	false	是否把文件名（包含路径）添加到 event 的 header 中。
fileHeaderKey	file	设置 header 中数据存储结构为<key,value>

参数	默认值	描述
		模式，需要 fileHeaderKey 与 fileHeader 配合使用。若 fileHeader 设置为 true，可参考如下示例。 示例：将 fileHeaderKey 定义为 file，当读取到文件名为/root/a.txt 的内容时，header 中以 file=/root/a.txt 的形式存在。

- **Http Source**

  Http Source 接收外部 HTTP 客户端发送过来的数据，并放入配置的 Channel 中，常用配置如下表所示：

表7-6 Http Source 常用配置

参数	默认值	描述
channels	-	与之相连的 channel，可以配置多个。
type	http	http source 的类型，必须为 http。
bind	-	监听主机名/IP。
port	-	绑定监听端口，该端口需未被占用。
handler	org.apache.flume.source.http.JSONHandler	http 请求的消息解析方式，支持 Json 格式解析（org.apache.flume.source.http.JSONHandler）和二进制 Blob 块解析（org.apache.flume.sink.solr.morphline.BlobHandler）。
handler.*	-	设置 handler 的参数。
exclude-protocols	SSLv3	排除的协议列表，用空格分开。默认排除 SSLv3 协议。
include-cipher-suites	-	包含的协议列表，用空格分开。如果设置为空，则默认支持所有协议。
enableSSL	false	http 协议是否启用 SSL。设置为 true 时还必须指定“秘钥(keystore)”和“秘钥存储密码(keystore-password)”。
keystore-type	JKS	Keystore 类型，可以为 JKS 或者 PKCS12。
keystore	-	http 启用 SSL 后设置 keystore 的路径。
keystorePassword	-	http 启用 SSL 后设置 keystore 的密

参数	默认值	描述
		码。

- **Thrift Source**

  Thrift Source 监听 thrift 端口，接收外部 Thrift 客户端数据并放入配置的 Channel 中。常用配置如下表所示：

参数	默认值	描述
channels	-	与之相连的 channel，可以配置多个。
type	thrift	thrift source 的类型，必须设置为 thrift。
bind	-	监听主机名/IP。
port	-	绑定监听端口，该端口需未被占用。
threads	-	允许运行的最大的 worker 线程数目。
kerberos	false	是否启用 Kerberos 认证。
agent-keytab	-	服务端使用的 keytab 文件地址，必须使用机机帐号。建议使用 Flume 服务安装目录下 flume/conf/flume_server.keytab。
agent-principal	-	服务端使用的安全用户的 Principal，必须使用机机帐户。建议使用 Flume 服务默认用户 flume_server/hadoop.<*系统域名*>@<*系统域名*> 说明 “flume_server/hadoop.<*系统域名*>”为用户名，用户的用户名所包含的系统域名所有字母为小写。例如“本端域”参数为“9427068F-6EFA-4833-B43E-60CB641E5B6C.COM”，用户名为“flume_server/hadoop.9427068f-6efa-4833-b43e-60cb641e5b6c.com”。
compression-type	none	消息压缩格式：“none”或“deflate”。“none”表示不压缩，“deflate”表示压缩。
ssl	false	是否使用 SSL 加密。设置为 true 时还必须指定“秘钥(keystore)”和“秘钥存储密码(keystore-password)”。
keystore-type	JKS	SSL 启用后密钥存储类型。
keystore	-	SSL 启用后密钥存储文件路径，开启 SSL 后，该参数必填。
keystore-password	-	SSL 启用后密钥存储密码，开启 ssl 后，该参数必填。

常用 Channel 配置

- **Memory Channel**

  Memory Channel 使用内存作为缓存区，Events 存放在内存队列中。常用配置如下表所示：

表7-7 Memory Channel 常用配置

参数	默认值	描述
type	-	memory channel 的类型，必须设置为 memory。
capacity	10000	缓存在 channel 中的最大 Event 数。
transactionCapacity	1000	每次存取的最大 Event 数。 说明 • 此参数值需要大于 source 和 sink 的 batchSize。 • 事务缓存容量必须小于或等于 Channel 缓存容量。
channelfullcount	10	channel full 次数，达到该次数后发送告警。
keep-alive	3	当事务缓存或 Channel 缓存满时，Put、Take 线程等待时间。单位：秒。
byteCapacity	JVM 最大内存的 80%	channel 中最多能容纳所有 event body 的总字节数，默认是 JVM 最大可用内存（-Xmx ）的 80%，单位：bytes。
byteCapacityBufferPercentage	20	channel 中字节容量百分比（%）。

- **File Channel**

  File Channel 使用本地磁盘作为缓存区，Events 存放在设置的 dataDirs 配置项文件夹中。常用配置如下表所示：

表7-8 File Channel 常用配置

参数	默认值	描述
type	-	file channel 的类型，必须

参数	默认值	描述
		设置为 file。
checkpointDir	${BIGDATA_DATA_HOME}/hadoop/data1~N/flume/checkpoint 说明 此路径随自定义数据路径变更。	检查点存放路径。
dataDirs	${BIGDATA_DATA_HOME}/hadoop/data1~N/flume/data 说明 此路径随自定义数据路径变更。	数据缓存路径，设置多个路径可提升性能，中间用逗号分开。
maxFileSize	2146435071	单个缓存文件的最大值，单位：bytes。
minimumRequiredSpace	524288000	缓冲区空闲空间最小值，单位：bytes。
capacity	1000000	缓存在 channel 中的最大 Event 数。
transactionCapacity	10000	每次存取的最大 Event 数。 说明 • 此参数值需要大于 source 和 sink 的 batchSize。 • 事务缓存容量必须小于或等于 Channel 缓存容量。
channelfullcount	10	channel full 次数，达到该次数后发送告警。
useDualCheckpoints	false	是否备份检查点。设置为“true”时，必须设置 backupCheckpointDir 的参数值。
backupCheckpointDir	-	备份检查点路径。
checkpointInterval	30000	检查点间隔时间，单位：秒。
keep-alive	3	当事务缓存或 Channel 缓存满时，Put、Take 线程等待时间。单位：秒。

参数	默认值	描述
use-log-replay-v1	false	是否启用旧的回复逻辑。
use-fast-replay	false	是否使用队列回复。
checkpointOnClose	true	channel 关闭时是否创建检查点。

- **Memory File Channel**

  Memory File Channel 同时使用内存和本地磁盘作为缓存区，消息可持久化，性能优于 File Channel，接近 Memory Channel 的性能。此 Channel 目前处于试验阶段，可靠性不够高，不建议在生产环境使用。常用配置如下表所示：

表7-9 Memory File Channel 常用配置

参数	默认值	描述
type	org.apache.flume.channel.MemoryFileChannel	memory file channel 的类型，必须设置为“org.apache.flume.channel.MemoryFileChannel”。
capacity	50000	Channel 缓存容量：缓存在 Channel 中的最大 Event 数。
transactionCapacity	5000	事务缓存容量：一次事务能处理的最大 Event 数。 说明 • 此参数值需要大于 source 和 sink 的 batchSize。 • 事务缓存容量必须小于或等于 Channel 缓存容量。
subqueueByteCapacity	20971520	每个 subqueue 最多保存多少 byte 的 Event，单位：byte。 Memory File Channel 采用 queue 和 subqueue 两级缓存，event 保存在 subqueue，subqueue 保存在 queue。 subqueue 能保存多少 event，由“subqueueCapacity”和“subqueueInterval”两个参数决定，“subqueueCapacity”限制 subqueue 内的 Event 总容量，“subqueueInterval”限制 subqueue 保存 Event 的时长，只有 subqueue 达到“subqueueCapacity”或“subqueueInterval”上限时，subqueue 内的 Event 才会发往目的地。

参数	默认值	描述
		说明 “subqueueByteCapacity”必须大于一个 batchsize 内的 Event 总容量。
subqueueInterval	2000	每个 subqueue 最多保存一段多长时间的 Event，单位：毫秒。
keep-alive	3	当事务缓存或 Channel 缓存满时，Put、Take 线程等待时间。 单位：秒。
dataDir	-	缓存本地文件存储目录。
byteCapacity	JVM 最大内存的 80%	Channel 缓存容量。 单位：bytes。
compression-type	None	消息压缩格式：“none”或“deflate”。 “none”表示不压缩，“deflate”表示压缩。
channelfullcount	10	channel full 次数，达到该次数后发送告警。

Memory File Channel 配置样例：

```
server.channels.c1.type = org.apache.flume.channel.MemoryFileChannel
server.channels.c1.dataDir = /opt/flume/mfdata
server.channels.c1.subqueueByteCapacity = 20971520
server.channels.c1.subqueueInterval=2000
server.channels.c1.capacity = 500000
server.channels.c1.transactionCapacity = 40000
```

- **Kafka Channel**

  Kafka Channel 使用 Kafka 集群缓存数据，Kafka 提供高可用、多副本，以防 Flume 或 Kafka Broker 崩溃，Channel 中的数据会立即被 Sink 消费。

表7-10 Kafka channel 常用配置

Parameter	Default Value	Description
type	-	kafka channel 的类型，必须设置为“org.apache.flume.channel.kafka.KafkaChannel”。
kafka.bootstrap.servers	-	Kafka 的 bootstrap 地址端口列表。 如果集群已安装 Kafka 并且配置已经同步，则服务端可以不配置此项，默认值为 Kafka 集群中所有的 broker 列表。客户端必须

Parameter	Default Value	Description
		配置该项，多个值用逗号分隔。端口和安全协议的匹配规则必须为：21007匹配安全模式（SASL_PLAINTEXT），9092匹配普通模式（PLAINTEXT）。
kafka.topic	flume-channel	channel用来缓存数据的topic。
kafka.consumer.group.id	flume	从kafka中获取数据的组标识，此参数不能为空。
parseAsFlumeEvent	true	是否解析为Flume event。
migrateZookeeperOffsets	true	当Kafka没有存储offset时，是否从ZooKeeper中查找，并提交到Kafka。
kafka.consumer.auto.offset.reset	latest	当没有offset记录时从什么位置消费，可选为“earliest”、“latest”或“none”。“earliest”表示将offset重置为初始点，“latest”表示将offset置为最新位置点，“none”表示若没有offset则抛出异常。
kafka.producer.security.protocol	SASL_PLAINTEXT	Kafka生产安全协议。端口和安全协议的匹配规则必须为：21007匹配安全模式（SASL_PLAINTEXT），9092匹配普通模式（PLAINTEXT）。 说明 若该参数没有显示，请单击弹窗左下角的"+"显示全部参数。
kafka.consumer.security.protocol	SASL_PLAINTEXT	同上，但用于消费。端口和安全协议的匹配规则必须为：21007匹配安全模式（SASL_PLAINTEXT），9092匹配普通模式（PLAINTEXT）。

Parameter	Default Value	Description
pollTimeout	500	consumer 调用 poll()函数能接受的最大超时时间，单位：毫秒。
ignoreLongMessage	false	是否丢弃超大消息。
messageMaxLength	1000012	Flume 写入 Kafka 的消息的最大长度。

常用 Sink 配置

- **HDFS Sink**

  HDFS Sink 将数据写入 Hadoop 分布式文件系统（HDFS）。常用配置如下表所示：

表7-11 HDFS Sink 常用配置

参数	默认值	描述
channel	-	与之相连的 channel。
type	hdfs	hdfs sink 的类型，必须设置为 hdfs。
hdfs.path	-	HDFS 上数据存储路径，必须以“hdfs://hacluster/”开头。
monTime	0（不开启）	线程监控阈值，更新时间超过阈值后，重新启动该 Sink，单位：秒。
hdfs.inUseSuffix	.tmp	正在写入的 hdfs 文件后缀。
hdfs.rollInterval	30	按时间滚动文件，单位：秒。
hdfs.rollSize	1024	按大小滚动文件，单位：bytes。
hdfs.rollCount	10	按 Event 个数滚动文件。 说明 参数“rollInterval”、“rollSize”和“rollCount”可同时配置，三个参数采取优先原则，哪个参数值先满足，优先按照哪个参数进行压缩。
hdfs.idleTimeout	0	自动关闭空闲文件超时时间，单位：秒。
hdfs.batchSize	1000	批次写入 HDFS 的 Event 个数。
hdfs.kerberosPrincipal	-	认证 HDFS 的 Kerberos principal，普通模式集群不配置，安全模式集群必须配置。
hdfs.kerberosKeytab	-	认证 HDFS 的 Kerberos keytab，普通模式集群不配置，安全模式集群中，用户必须对

参数	默认值	描述
		jaas.cof 文件中的 keyTab 路径有访问权限。
hdfs.fileCloseByEndEvent	true	收到源文件的最后一个 Event 时是否关闭 hdfs 文件。
hdfs.batchCallTimeout	-	批次写入 HDFS 超时控制时间，单位：毫秒。 当不配置此参数时，对每个 Event 写入 HDFS 进行超时控制。当“hdfs.batchSize”大于 0 时，配置此参数可以提升写入 HDFS 性能。 说明 “hdfs.batchCallTimeout”设置多长时间需要考虑“hdfs.batchSize”的大小，“hdfs.batchSize”越大，“hdfs.batchCallTimeout”也要调整更长时间，设置过短时间容易导致写 HDFS 失败。
serializer.appendNewline	true	将一个 Event 写入 HDFS 后是否追加换行符（'\n'），如果追加该换行符，该换行符所占用的数据量指标不会被 HDFS Sink 统计。
hdfs.filePrefix	over_%{basename}	数据写入 hdfs 后文件名的前缀。
hdfs.fileSuffix	-	数据写入 hdfs 后文件名的后缀。
hdfs.inUsePrefix	-	正在写入的 hdfs 文件前缀。
hdfs.fileType	DataStream	hdfs 文件格式，包括“SequenceFile”、“DataStream”以及“CompressedStream”。 说明 “SequenceFile”和“DataStream”不压缩输出文件，不能设置参数“codeC”，“CompressedStream”压缩输出文件，必须设置“codeC”参数值配合使用。
hdfs.codeC	-	文件压缩格式，包括 gzip、bzip2、lzo、lzop、snappy。
hdfs.maxOpenFiles	5000	最大允许打开的 hdfs 文件数，当打开的文件数达到该值时，最早打开的文件将会被关闭。
hdfs.writeFormat	Writable	文件写入格式，“Writable”或者“Text”。
hdfs.callTimeout	10000	写入 HDFS 超时控制时间，单位：毫秒。
hdfs.threadsPoolSize	-	每个 HDFS sink 用于 HDFS io 操作的线程数。

参数	默认值	描述
hdfs.rollTimerPoolSize	-	每个 HDFS sink 用于调度定时文件滚动的线程数。
hdfs.round	false	时间戳是否四舍五入。若设置为 true，则会影响所有基于时间的转义序列（%t 除外）。
hdfs.roundUnit	second	时间戳四舍五入单位，可选为“second”、“minute”或“hour”，分别对应为秒、分钟和小时。
hdfs.useLocalTimeStamp	true	是否启用本地时间戳，建议设置为“true”。
hdfs.closeTries	0	hdfs sink 尝试关闭重命名文件的最大次数。默认为 0 表示 sink 会一直尝试重命名，直至重命名成功。
hdfs.retryInterval	180	尝试关闭 hdfs 文件的时间间隔，单位：秒。 说明 每个关闭请求都会有多个 RPC 往返 Namenode，因此设置的太低可能导致 Namenode 超负荷。如果设置 0，如果第一次尝试失败的话，该 Sink 将不会尝试关闭文件，并且把文件打开，或者用“.tmp”作为扩展名。
hdfs.failcount	10	数据写入 hdfs 失败的次数。该参数作为 sink 写入 hdfs 失败次数的阈值，当超过该阈值后上报数据传输异常告警。

- **Avro Sink**

  Avro Sink 把 events 转化为 Avro events 并发送到配置的主机的监听端口。常用配置如下表所示：

表7-12 Avro Sink 常用配置

参数	默认值	描述
channel	-	与之相连的 channel。
type	-	avro sink 的类型，必须设置为 avro。
hostname	-	绑定的主机名/IP。
port	-	监听端口，该端口需未被占用。
batch-size	1000	批次发送的 Event 个数。
client.type	DEFAULT	客户端实例类型，根据所

参数	默认值	描述
		配置的模型实际使用到的通信协议设置。该值可选值包括： • DEFAULT，返回 AvroRPC 类型的客户端实例。 • OTHER，返回 NULL。 • THRIFT，返回 Thrift RPC 类型的客户端实例。 • DEFAULT_LOADBALANCING， 返回 LoadBalancing RPC 客户端实例。 • DEFAULT_FAILOVER， 返回 Failover RPC 客户端实例。
ssl	false	是否使用 SSL 加密。设置为 true 时还必须指定“秘钥(keystore)”和“秘钥存储密码(keystore-password)”。
truststore-type	JKS	Java 信任库类型，“JKS”或“PKCS12”。 说明 JKS 的秘钥库和私钥采用不同的密码进行保护，而 PKCS12 的秘钥库和私钥采用相同密码进行保护。
truststore	-	Java 信任库文件。
truststore-password	-	Java 信任库密码。
keystore-type	JKS	ssl 启用后密钥存储类型。
keystore	-	ssl 启用后密钥存储文件路径，开启 ssl 后，该参数必填。
keystore-password	-	ssl 启用后密钥存储密码，开启 ssl 后，该参数必填。
connect-timeout	20000	第一次连接的超时时间，

参数	默认值	描述
		单位：毫秒。
request-timeout	20000	第一次请求后一次请求的最大超时时间，单位：毫秒。
reset-connection-interval	0	一次断开连接后，等待多少时间后进行重新连接，单位：秒。默认为 0 表示不断尝试。
compression-type	none	批数据压缩类型，“none”或“deflate”，“none”表示不压缩，“deflate”表示压缩。该值必须与 AvroSource 的 compression-type 匹配。
compression-level	6	批数据压缩级别（1-9），数值越高，压缩率越高。
exclude-protocols	SSLv3	排除的协议列表，用空格分开。默认排除 SSLv3 协议。

- **HBase Sink**

  HBase Sink 将数据写入到 HBase 中。常用配置如下表所示：

表7-13 HBase Sink 常用配置

参数	默认值	描述
channel	-	与之相连的 channel。
type	-	hbase sink 的类型，必须设置为 hbase。
table	-	HBase 表名称。
columnFamily	-	HBase 列族。
monTime	0（不开启）	线程监控阈值，更新时间超过阈值后，重新启动该 Sink，单位：秒。
batchSize	1000	批次写入 HBase 的 Event 个数。
kerberosPrincipal	-	认证 HBase 的 Kerberos principal，普通模式集群不配置，安全模式集群必须配置。
kerberosKeytab	-	认证 HBase 的 Kerberos keytab，普通模式集群不配置，安全模式集群中，flume 运行用户必须对

参数	默认值	描述
		jaas.cof 文件中的 keyTab 路径有访问权限。
coalesceIncrements	true	是否在同一个处理批次中，合并对同一个 hbase cell 多个操作。设置为 true 有利于提高性能。

- **Kafka Sink**

  Kafka Sink 将数据写入到 Kafka 中。常用配置如下表所示：

表7-14 Kafka Sink 常用配置

参数	默认值	描述
channel	-	与之相连的 channel。
type	-	kafka sink 的类型，必须设置为 org.apache.flume.sink.kafka.KafkaSink。
kafka.bootstrap.servers	-	Kafka 的 bootstrap 地址端口列表。如果集群安装有 kafka 并且配置已经同步，服务端可以不配置此项，默认值为 Kafka 集群中所有的 broker 列表，客户端必须配置该项，多个用逗号分隔。端口和安全协议的匹配规则必须为：21007 匹配安全模式（SASL_PLAINTEXT），9092 匹配普通模式（PLAINTEXT）。
monTime	0（不开启）	线程监控阈值，更新时间超过阈值后，重新启动该 Sink，单位：秒。
kafka.producer.acks	1	必须收到多少个 replicas 的确认信息才认为写入成功。0 表示不需要接收确认信息，1 表示只等待 leader 的确认信息。-1 表示等待所有的 relicas 的确认信息。设置为-1，在某些 leader 失败的场景中可以避免数据丢失。
kafka.topic	-	数据写入的 topic，必须填写。
flumeBatchSize	1000	批次写入 Kafka 的 Event 个数。
kafka.security.protocol	SASL_PLAINTEXT	Kafka 安全协议，普通模式集群下须配置为“PLAINTEXT”。端口和安全协议的匹配规则必须为：21007 匹配安全模式（SASL_PLAINTEXT），9092 匹配普通模式（PLAINTEXT）。
ignoreLongMessage	false	是否丢弃超大消息的开关。
messageMaxLength	1000012	Flume 写入 Kafka 的消息的最大长度。
defaultPartitionId	-	用于指定 channel 中的 events 被传输到哪一个

参数	默认值	描述
		Kafka partition ID ，此值会被 partitionIdHeader 覆盖。默认情况下，如果此参数不设置，会由 Kafka Producer's partitioner 进行 events 分发(可以通过指定 key 或者 kafka.partitioner.class 自定义的 partitioner)。
partitionIdHeader	-	设置时，对应的 Sink 将从 Event 的 Header 中获取使用此属性的值命名的字段的值，并将消息发送到主题的指定分区。 如果该值无对应的有效分区，则会抛出 EventDeliveryException。 如果 Header 值已经存在，则此设置将覆盖参数 defaultPartitionId。
Other Kafka Producer Properties	-	其他 Kafka 配置，可以接受任意 Kafka 支持的生产配置，配置需要加前缀 .kafka。

- **Thrift Sink**

  Thrift Sink 把 events 转化为 Thrift events 并发送到配置的主机的监听端口。常用配置如下表所示：

表7-15 Thrift Sink 常用配置

参数	默认值	描述
channel	-	与之相连的 channel。
type	thrift	thrift sink 的类型，必须设置为 thrift。
hostname	-	绑定的主机名/IP。
port	-	监听端口，该端口需未被占用。
batch-size	1000	批次发送的 Event 个数。
connect-timeout	20000	第一次连接的超时时间，单位：毫秒。
request-timeout	20000	第一次请求后一次请求的最大超时时间，单位：毫秒。
kerberos	false	是否启用 Kerberos 认证。
client-keytab	-	客户端使用的 keytab 文件地址，flume 运行用户必须对认证文件具有访问权

参数	默认值	描述
		限。
client-principal	-	客户端使用的安全用户的Principal。
server-principal	-	服务端使用的安全用户的Principal。
compression-type	none	Flume 发送数据的压缩类型，“none”或“deflate”，“none”表示不压缩，“deflate”表示压缩。
maxConnections	5	Flume 发送数据时的最大连接池大小。
ssl	false	是否使用 SSL 加密。
truststore-type	JKS	Java 信任库类型。
truststore	-	Java 信任库文件。
truststore-password	-	Java 信任库密码。
reset-connection-interval	0	一次断开连接后，等待多少时间后进行重新连接，单位：秒。默认为 0 表示不断尝试。

注意事项

- Flume 可靠性保障措施有哪些？
  - Source&Channel、Channel&Sink 之间的事务机制。
  - Sink Processor 支持配置 failover、load_blance 机制，例如负载均衡示例如下，详细参考 http://flume.apache.org/releases/1.9.0.html。

```
server.sinkgroups=g1
server.sinkgroups.g1.sinks=k1 k2
server.sinkgroups.g1.processor.type=load_balance
server.sinkgroups.g1.processor.backoff=true
server.sinkgroups.g1.processor.selector=random
```

- Flume 多 agent 聚合级联时的注意事项？
  - 级联时需要使用 Avro 或者 Thrift 协议进行级联。
  - 聚合端存在多个节点时，连接配置尽量配置均衡，不要聚合到单节点上。

7.8 Flume 配置参数说明

MRS 3.x 之前版本需在“properties.properties”文件中配置。

MRS 3.x 及之后版本，部分参数可在 Manager 界面配置。

基本介绍

使用 Flume 需要配置 Source、Channel 和 Sink，各模块配置参数说明可通过本节内容了解。

MRS 3.x 及之后版本部分参数可通过 Manager 界面配置，选择“集群 > 服务 > Flume > 配置工具”，选择要使用的 Source、Channel 以及 Sink，将其拖到右侧的操作界面中，双击对应的 Source、Channel 以及 Sink，根据实际环境可配置 Source、Channel 和 Sink 参数。“channels”、“type”等参数仅在客户端配置文件“properties.properties”中进行配置，配置文件路径为“*Flume 客户端安装目录*/**fusioninsight-flume-***Flume 组件版本号*/**conf/properties.properties**”。

说明

部分配置可能需要填写加密后的信息，请参见使用 Flume 客户端加密工具。

常用 Source 配置

- **Avro Source**

  Avro Source 监听 Avro 端口，接收外部 Avro 客户端数据并放入配置的 Channel 中。常用配置如表 7-16 所示：

表7-16 Avro Source 常用配置

参数	默认值	描述
channels	-	与之相连的 Channel，可以配置多个。用空格隔开。 在单个代理流程中，是通过 channel 连接 sources 和 sinks。一个 source 实例对应多个 channels，但一个 sink 实例只能对应一个 channel。 格式如下： **<Agent >.sources.<Source>.channels = <channel1> <channel2> <channel3>...** **<Agent >.sinks.<Sink>.channels = <channel1>** 仅可在“properties.properties”文件中配置。
type	avro	类型，需设置为“avro”。每一种 source 的类型都为相应的固定值。 仅可在“properties.properties”文件中配

参数	默认值	描述
		置。
bind	-	绑定和 source 关联的主机名或 IP 地址。
port	-	绑定端口号。
ssl	false	是否使用 SSL 加密。 • true • false
truststore-type	JKS	Java 信任库类型。填写 JKS 或其他 java 支持的 truststore 类型。
truststore	-	Java 信任库文件。
truststore-password	-	Java 信任库密码。
keystore-type	JKS	密钥存储类型。填写 JKS 或其他 java 支持的 truststore 类型。
keystore	-	密钥存储文件。
keystore-password	-	密钥存储密码。

- **SpoolDir Source**

  SpoolDir Source 监控并传输目录下新增的文件，可实现准实时数据传输。常用配置如表 2 Spooling Source 常用配置所示：

表7-17 SpoolDir Source 常用配置

参数	默认值	描述
channels	-	与之相连的 Channel，可以配置多个。 仅可在“properties.properties”文件中配置。
type	spooldir	类型，需设置为“spooldir”。 仅可在“properties.properties”文件中配置。
monTime	0（不开启）	线程监控阈值，更新时间大于阈值时会重新启动该 Source，单位：秒。
spoolDir	-	监控目录。
fileSuffix	.COMPLETED	文件传输完成后添加的后缀。
deletePolicy	never	文件传输完成后源文件删除策略，支持“never”或“immediate”。分别别是从不删除和立即删除。

参数	默认值	描述
ignorePattern	^$	忽略文件的正则表达式表示。
trackerDir	.flumespool	传输过程中元数据存储路径。
batchSize	1000	Source 传输粒度。
decodeErrorPolicy	FAIL	编码错误策略。仅可在“properties.properties”文件中配置。 可选 FAIL、REPLACE、IGNORE。 FAIL：抛出异常并让解析失败。 REPLACE：将不能识别的字符用其它字符代替，通常是字符 U+FFFD。 IGNORE：直接丢弃不能解析的字符串。 说明 如果文件中有编码错误，请配置“decodeErrorPolicy”为“REPLACE”或“IGNORE”，Flume 遇到编码错误将跳过编码错误，继续采集后续日志。
deserializer	LINE	文件解析器，值为“LINE”或“BufferedLine”。 • 配置为“LINE”时，对从文件读取的字符逐个转码。 • 配置为“BufferedLine”时，对文件读取的一行或多行的字符进行批量转码，性能更优。
deserializer.maxLineLength	2048	按行解析最大长度。0 到 2,147,483,647。
deserializer.maxBatchLine	1	按行解析最多行数，如果行数设置为多行，“maxLineLength”也应该设置为相应的倍数。例如 maxBatchLine 设置为 2，“maxLineLength”相应的设置为 2048*2 为 4096。
selector.type	replicating	选择器类型，支持“replicating”或“multiplexing”。 • “replicating”表示同样的内容会发给每一个 channel。 • “multiplexing”表示根据分发规则，有选择地发给某些 channel。
interceptors	-	拦截器配置。详细配置可参考 flume 官方文档。 仅可在“properties.properties”文件中配

参数	默认值	描述
		置。

说明

Spooling Source 在按行读取过程中，会忽略掉每一个 Event 的最后一个换行符，该换行符所占用的数据量指标不会被 Flume 统计。

- **Kafka Source**

  Kafka Source 从 Kafka 的 topic 中消费数据，可以设置多个 Source 消费同一个 topic 的数据，每个 Source 会消费 topic 的不同 partitions。常用配置如表 3 Kafka Source 常用配置所示：

表7-18 Kafka Source 常用配置

参数	默认值	描述
channels	-	与之相连的 Channel，可以配置多个。 仅可在“properties.properties”文件中配置。
type	org.apache.flume.source.kafka.KafkaSource	类型，需设置为“org.apache.flume.source.kafka.KafkaSource”。 仅可在“properties.properties”文件中配置。
monTime	0（不开启）	线程监控阈值，更新时间大于阈值时重新启动该 Source，单位：秒。
nodatatime	0（不开启）	告警阈值，从 Kafka 中订阅不到数据的时长大于阈值时发送告警，单位：秒。
batchSize	1000	每次写入 Channel 的 Event 数量。
batchDurationMillis	1000	每次消费 topic 数据的最大时长，单位：毫秒。
keepTopicInHeader	false	是否在 Event Header 中保存 topic，如果保存，Kafka Sink 配置的 topic 将无效。 • true • false 仅可在“properties.properties”文件中配置。
keepPartitionInHeader	false	是否在 Event Header 中保存 partitionID，如果保存，Kafka Sink 将写入对应的 Partition。 • true

参数	默认值	描述
		• false 仅可在“properties.properties”文件中配置。
kafka.bootstrap.servers	-	brokers 地址列表，多个地址用英文逗号分隔。
kafka.consumer.group.id	-	Kafka 消费者组 ID。
kafka.topics	-	订阅的 kafka topic 列表，用英文逗号分隔。
kafka.topics.regex	-	符合正则表达式的 topic 会被订阅，优先级高于“kafka.topics”，如果配置将覆盖“kafka.topics”。
kafka.security.protocol	SASL_PLAINTEXT	Kafka 安全协议，未启用 Kerberos 集群中须配置为“PLAINTEXT”。
kafka.kerberos.domain.name	-	此参数的值为 Kafka 集群中 kerberos 的“default_realm”，仅安全集群需要配置。 仅可在“properties.properties”文件中配置。
Other Kafka Consumer Properties	-	其他 Kafka 配置，可以接受任意 Kafka 支持的消费参数配置，配置需要加前缀“.kafka”。 仅可在“properties.properties”文件中配置。

- **Taildir Source**

  Taildir Source 监控目录下文件的变化并自动读取文件内容，可实现实时数据传输，常用配置如表 7-19 所示：

表7-19 Taildir Source 常用配置

参数	默认值	描述
channels	-	与之相连的 Channel，可以配置多个。 仅可在“properties.properties”文件中配置。
type	taildir	类型，需配置为“taildir”。 仅可在“properties.properties”文件中配置。
filegroups	-	设置采集文件目录分组名字，分组名字

参数	默认值	描述
		中间使用空格间隔。
filegroups.<filegroupName>.parentDir	-	父目录，需要配置为绝对路径。 仅可在“properties.properties”文件中配置。
filegroups.<filegroupName>.filePattern	-	相对父目录的文件路径，可以包含目录，支持正则表达式，须与父目录联合使用。 仅可在“properties.properties”文件中配置。
positionFile	-	传输过程中元数据存储路径。
headers.<filegroupName>.<headerKey>	-	设置某一个分组采集数据时 Event 中的 key-value 值。 仅可在“properties.properties”文件中配置。
byteOffsetHeader	false	是否在每一个 Event 头中携带该 Event 在源文件中的位置信息，该信息保存在“byteoffset”变量中。
skipToEnd	false	Flume 在重启后是否直接定位到文件最新的位置处，以读取最新的数据。
idleTimeout	120000	设置读取文件的空闲时间，单位：毫秒。如果在该时间内文件内容没有变更，关闭掉该文件，关闭后如果该文件有数据写入，重新打开并读取数据。
writePosInterval	3000	设置将元数据写入到文件的周期，单位：毫秒。
batchSize	1000	批次写入 Channel 的 Event 数量。
monTime	0（不开启）	线程监控阈值，更新时间大于阈值时重新启动该 Source，单位：秒。

- **Http Source**

  Http Source 接收外部 HTTP 客户端发送过来的数据，并放入配置的 Channel 中，常用配置如表 7-20 所示：

表7-20 Http Source 常用配置

参数	默认值	描述
channels	-	与之相连的 Channel，可以配置多个。仅可在“properties.properties”文件中配

参数	默认值	描述
		置。
type	http	类型，需配置为“http”。仅可在“properties.properties”文件中配置。
bind	-	绑定关联的主机名或 IP 地址。
port	-	绑定端口。
handler	org.apache.flume.source.http.JSONHandler	http 请求的消息解析方式，支持以下两种： • “org.apache.flume.source.http.JSONHandler”：表示 Json 格式解析。 • “org.apache.flume.sink.solr.morphline.BlobHandler”：表示二进制 Blob 块解析。
handler.*	-	设置 handler 的参数。
enableSSL	false	http 协议是否启用 SSL。
keystore	-	http 启用 SSL 后设置 keystore 的路径。
keystorePassword	-	http 启用 SSL 后设置 keystore 的密码。

常用 Channel 配置

- **Memory Channel**

  Memory Channel 使用内存作为缓存区，Events 存放在内存队列中。常用配置如表 7-21 所示：

表7-21 Memory Channel 常用配置

参数	默认值	描述
type	-	类型，需配置为“memory”。仅可在“properties.properties”文件中配置。
capacity	10000	缓存在 Channel 中的最大 Event 数。
transactionCapacity	1000	每次存取的最大 Event 数。
channelfullcount	10	Channel full 次数，达到该次数后发送告警。

- **File Channel**

  File Channel 使用本地磁盘作为缓存区，Events 存放在设置的“dataDirs”配置项文件夹中。常用配置如表 7-22 所示：

表7-22 File Channel 常用配置

参数	默认值	描述
type	-	类型，需配置为“file”。仅可在“properties.properties”文件中配置。
checkpointDir	${BIGDATA_DATA_HOME}/flume/checkpoint	检查点存放路径。
dataDirs	${BIGDATA_DATA_HOME}/flume/data	数据缓存路径，设置多个路径可提升性能，中间用逗号分开。
maxFileSize	2146435071	单个缓存文件的最大值，单位：字节。
minimumRequiredSpace	524288000	缓冲区空闲空间最小值，单位：字节。
capacity	1000000	缓存在 Channel 中的最大 Event 数。
transactionCapacity	10000	每次存取的最大 Event 数。
channelfullcount	10	Channel full 次数，达到该次数后发送告警。

- **Kafka Channel**

  Kafka Channel 使用 kafka 集群缓存数据，Kafka 提供高可用、多副本，以防 Flume 或 Kafka Broker 崩溃，Channel 中的数据会立即被 Sink 消费。常用配置如表 10 Kafka Channel 常用配置所示：

表7-23 Kafka Channel 常用配置

参数	默认值	描述
type	-	类型，需配置为“org.apache.flume.channel.kafka.KafkaChannel”.。 仅可在“properties.properties”文件中配置。
kafka.bootstrap.servers	-	kafka broker 列表。
kafka.topic	flume-channel	Channel 用来缓存数据的 topic。
kafka.consumer.group.id	flume	Kafka 消费者组 ID。
parseAsFlumeEvent	true	是否解析为 Flume event。
migrateZookeeperOffsets	true	当 Kafka 没有存储 offset 时，是否从 ZooKeeper 中查找，并提交到 Kafka。

参数	默认值	描述
kafka.consumer.auto.offset.reset	latest	当没有 offset 记录时，从指定的位置消费数据。
kafka.producer.security.protocol	SASL_PLAINTEXT	Kafka 生产者安全协议。
kafka.consumer.security.protocol	SASL_PLAINTEXT	Kafka 消费者安全协议。

常用 Sink 配置

- **HDFS Sink**

  HDFS Sink 将数据写入 HDFS。常用配置如表 7-24 所示：

表7-24 HDFS Sink 常用配置

参数	默认值	描述
channel	-	与之相连的 Channel。仅可在“properties.properties”文件中配置。
type	hdfs	类型，需配置为“hdfs”。仅可在“properties.properties”文件中配置。
monTime	0（不开启）	线程监控阈值，更新时间大于阈值时重新启动该 Sink，单位：秒。
hdfs.path	-	HDFS 路径。
hdfs.inUseSuffix	.tmp	正在写入的 HDFS 文件后缀。
hdfs.rollInterval	30	按时间滚动文件，单位：秒。
hdfs.rollSize	1024	按大小滚动文件，单位：字节。
hdfs.rollCount	10	按 Event 个数滚动文件。
hdfs.idleTimeout	0	自动关闭空闲文件超时时间，单位：秒。
hdfs.batchSize	1000	每次写入 HDFS 的 Event 个数。
hdfs.kerberosPrincipal	-	认证 HDFS 的 Kerberos 用户名，未启用 Kerberos 认证集群不配置。
hdfs.kerberosKeytab	-	认证 HDFS 的 Kerberos keytab 路径，未启用 Kerberos 认证集群不配置
hdfs.fileCloseByEndEvent	true	收到最后一个 Event 时是否关闭文件。
hdfs.batchCallTimeout	-	每次写入 HDFS 超时控制时间，单位：

参数	默认值	描述
		毫秒。 当不配置此参数时，对每个 Event 写入 HDFS 进行超时控制。当“hdfs.batchSize”大于 0 时，配置此参数可以提升写入 HDFS 性能。 说明 “hdfs.batchCallTimeout”设置多长时间需要考虑“hdfs.batchSize”的大小，“hdfs.batchSize”越大，“hdfs.batchCallTimeout”也要调整更长时间，设置过短时间容易导致数据写入 HDFS 失败。
serializer.appendNewline	true	将一个 Event 写入 HDFS 后是否追加换行符（'\n'），如果追加该换行符，该换行符所占用的数据量指标不会被 HDFS Sink 统计。

- **Avro Sink**

  Avro Sink 把 events 转化为 Avro events 并发送到配置的主机的监听端口。常用配置如表 7-25 所示：

表7-25 Avro Sink 常用配置

参数	默认值	描述
channel	-	与之相连的 Channel。仅可在“properties.properties”文件中配置。
type	-	类型，需配置为“avro”。仅可在“properties.properties”文件中配置。
hostname	-	绑定关联的主机名或 IP 地址。
port	-	监听端口。
batch-size	1000	批次发送的 Event 个数。
ssl	false	是否使用 SSL 加密。
truststore-type	JKS	Java 信任库类型。
truststore	-	Java 信任库文件。
truststore-password	-	Java 信任库密码。
keystore-type	JKS	密钥存储类型。

参数	默认值	描述
keystore	-	密钥存储文件。
keystore-password	-	密钥存储密码

- **HBase Sink**

  HBase Sink 将数据写入到 HBase 中。常用配置如表 7-26 所示：

表7-26 HBase Sink 常用配置

参数	默认值	描述
channel	-	与之相连的 Channel。仅可在“properties.properties”文件中配置。
type	-	类型，需配置为“hbase”。仅可在“properties.properties”文件中配置。
table	-	HBase 表名称。
monTime	0（不开启）	线程监控阈值，更新时间大于阈值时重新启动该 Sink，单位：秒。
columnFamily	-	HBase 列族名称。
batchSize	1000	每次写入 HBase 的 Event 个数。
kerberosPrincipal	-	认证 HBase 的 Kerberos 用户名，未启用 Kerberos 认证集群不配置。
kerberosKeytab	-	认证 HBase 的 Kerberos keytab 路径，未启用 Kerberos 认证集群不配置。

- **Kafka Sink**

  Kafka Sink 将数据写入到 Kafka 中。常用配置如表 7-27 所示：

表7-27 Kafka Sink 常用配置

参数	默认值	描述
channel	-	与之相连的 Channel。仅可在“properties.properties”文件中配置。
type	-	类型，需配置为“org.apache.flume.sink.kafka.KafkaSink”。 仅可在“properties.properties”文件中配置。

参数	默认值	描述
kafka.bootstrap.servers	-	Kafkabrokers 列表，多个用英文逗号分隔。
monTime	0（不开启）	线程监控阈值，更新时间大于阈值时重新启动该 Sink，单位：秒。
kafka.topic	default-flume-topic	数据写入的 topic。
flumeBatchSize	1000	每次写入 Kafka 的 Event 个数。
kafka.security.protocol	SASL_PLAINTEXT	Kafka 安全协议，未启用 Kerberos 认证集群下须配置为“PLAINTEXT”。
kafka.kerberos.domain.name	-	Kafka Domain 名称。安全集群必填。仅可在“properties.properties”文件中配置。
Other Kafka Producer Properties	-	其他 Kafka 配置，可以接受任意 Kafka 支持的生产参数配置，配置需要加前缀“.kafka”。 仅可在“properties.properties”文件中配置。

7.9 在配置文件 properties.properties 中使用环境变量

操作场景

本章节描述如何在配置文件“properties.properties”中使用环境变量。

本章节适用于 MRS 3.x 及之后版本。

前提条件

Flume 服务运行正常并已成功安装 Flume 客户端。

操作步骤

步骤 1 以 **root** 用户登录安装 Flume 客户端所在节点。

步骤 2 切换到以下目录。

cd *Flume 客户端安装目录*/**fusioninsight-flume**-*Flume 组件版本号*/**conf**

步骤 3 在该目录下的“flume-env.sh”文件中添加环境变量。

- 格式：

```
export 变量名=变量值
```

- 示例：

```
JAVA_OPTS="-Xms2G -Xmx4G -XX:CMSFullGCsBeforeCompaction=1 -
XX:+UseConcMarkSweepGC -XX:+CMSParallelRemarkEnabled -
XX:+UseCMSCompactAtFullCollection -
DpropertiesImplementation=org.apache.flume.node.EnvVarResolverProperties"
export TAILDIR_PATH=/tmp/flumetest/201907/20190703/1/.*log.*
```

步骤 4 重启 Flume 实例进程。

1. 登录 FusionInsight Manager。
2. 选择“集群 > 服务 > Flume > 实例”，勾选 Flume 实例，选择“更多 > 重启实例”输入密码，单击“确定”等待实例重启成功。

须知

服务端 flume-env.sh 生效后不能通过 Manager 界面重启整个 Flume 服务，否则用户自定义环境变量丢失，仅需在 Manager 界面重启对应实例即可。

步骤 5 在“*Flume 客户端安装目录*/**fusioninsight-flume**-*Flume 组件版本号*/**conf/properties.properties**”配置文件中使用“${变量名}”格式引用变量，示例如下：

```
client.sources.s1.type = TAILDIR
client.sources.s1.filegroups = f1
client.sources.s1.filegroups.f1 = ${TAILDIR_PATH}
client.sources.s1.positionFile =
/tmp/flumetest/201907/20190703/1/taildir_position.json
client.sources.s1.channels = c1
```

须知

- 必须保证“flume-env.sh”生效之后，再执行步骤 5 配置“properties.properties”文件。
- 若在本地配置该文件，配置完成后可参考如下步骤在 Manager 界面上传配置文件。若操作顺序不规范，可能造成用户自定义环境变量丢失。

- 登录 FusionInsight Manager。

1. 选择“集群 > 服务 > Flume > 配置”，勾选 Flume 实例，在“flume.config.file”后单击“上传文件”，上传“properties.properties”文件。

----结束

7.10 非加密传输

7.10.1 配置非加密传输

操作场景

该操作指导安装工程师在集群及 Flume 服务安装完成后，分别配置 Flume 服务的服务端和客户端参数，使其可以正常工作。

本章节适用于 MRS 3.x 及之后版本。

说明

本配置默认集群网络环境是安全的，数据传输过程不需要启用 SSL 认证。如需使用加密方式，请参考配置加密传输。

前提条件

- 已成功安装集群及 Flume 服务。
- 确保集群网络环境安全。

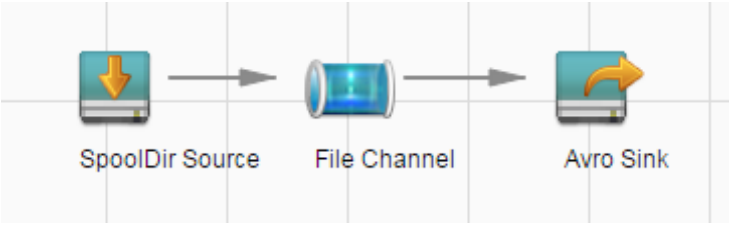
操作步骤

步骤 1 配置 Flume 角色客户端参数。

1. 使用 FusionInsight Manager 界面中的 Flume 配置工具来配置 Flume 角色客户端参数并生成配置文件。
   a. 登录 FusionInsight Manager，选择“集群 > 服务 > Flume > 配置工具”。
   b. “Agent 名”选择“client”，然后选择要使用的 Source、Channel 以及 Sink，将其拖到右侧的操作界面中并将其连接。

      例如采用 SpoolDir Source、File Channel 和 Avro Sink，如图 7-2 所示。

      图7-2 Flume 配置工具示例

   c. 双击对应的 Source、Channel 以及 Sink，根据实际环境并参考表 7-28 设置对应的配置参数。

说明

- 如果对应的 Flume 角色之前已经配置过客户端参数，为保证与之前的配置保持一致，可以到“*客户端安装目录*/fusioninsight-flume-1.9.0/conf/properties.properties”获取已有的客户端参数配置文件。然后登录 FusionInsight Manager，选择“集群 > 服务 > Flume > 配置 > 导入”，将该文件导入后再修改非加密传输的相关配置项即可。
- 导入配置文件时，建议配置 Source/Channel/Sink 的各自的个数都不要超过 40 个，否则可能导致界面响应时间过长。

表7-28 Flume 角色客户端所需修改的参数列表

参数名称	参数值填写规则	参数样例
ssl	是否启用 SSL 认证（基于安全要求，建议启用此功能） 只有“Avro”类型的 Source 才有此配置项 • true 表示启用 • false 表示不启用	false

d. 单击“导出”，将配置文件“properties.properties”保存到本地。

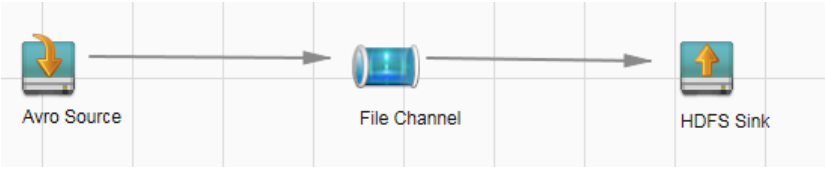
2. 将“properties.properties”文件上传到 Flume 客户端安装目录下的“flume/conf/”下。

步骤 2 配置 Flume 角色的服务端参数，并将配置文件上传到集群。

1. 使用 FusionInsight Manager 界面中的 Flume 配置工具来配置服务端参数并生成配置文件。
   a. 登录 FusionInsight Manager，选择“集群 > 服务 > Flume > 配置工具”。
   b. “Agent 名”选择“server”，然后选择要使用的 source、channel 以及 sink，将其拖到右侧的操作界面中并将其连接。

      例如采用 Avro Source、File Channel 和 HDFS Sink，如图 7-3 所示。

      图7-3 Flume 配置工具示例

   c. 双击对应的 source、channel 以及 sink，根据实际环境并参考表 7-29 设置对应的配置参数。

说明

- 如果对应的 Flume 角色之前已经配置过服务端参数，为保证与之前的配置保持一致，在 FusionInsight Manager 界面选择“集群 > 服务 > Flume > 实例”，选择相应的 Flume 角色实例，单击“实例配置 ”页面“flume.config.file”参数后的“下载文件”，可获取已有的服务端参数配置文件。然后选择“集群 > 服务 > Flume > 配置 > 导入”，将该文件导入后再修改非加密传输的相关配置项即可。
- 导入配置文件时，建议配置 source/channel/sink 的各自的个数都不要超过 40 个，否则可能导致界面响应时间过长。
- 不同的 File Channel 均需要配置一个不同的 checkpoint 目录。

表7-29 Flume 角色服务端所需修改的参数列表

参数名称	参数值填写规则	参数样例
ssl	是否启用 SSL 认证（基于安全要求，建议启用此功能） 只有“Avro”类型的 Source 才有此配置项 • true 表示启用 • false 表示不启用	false

d. 单击“导出”，将配置文件“properties.properties”保存到本地。

2. 登录 FusionInsight Manager，选择“集群 > 服务 > Flume”，在“实例”下单击“Flume”角色。
3. 选择准备上传配置文件的节点行的“Flume”角色，单击“实例配置 ”页面“flume.config.file”参数后的“上传文件”，选择“properties.properties”文件完成操作。

说明

- 每个 Flume 实例均可以上传单独的服务端配置文件。
- 更新配置文件需要按照此步骤操作，后台修改配置文件是不规范操作，同步配置时后台做的修改将会被覆盖。

4. 单击“保存”，单击“确定”。
5. 单击“完成”完成操作。

----结束

7.10.2 典型场景：从本地采集静态日志保存到 Kafka

操作场景

该任务指导用户使用 Flume 客户端从本地采集静态日志保存到 Kafka 的 Topic 列表（test1）。

本章节适用于 MRS 3.x 及之后版本。

说明

本配置默认集群网络环境是安全的，数据传输过程不需要启用 SSL 认证。如需使用加密方式，请参考配置加密传输。该配置为只用一个 Flume 场景，例如：Spooldir Source+Memory Channel+Kafka Sink.

前提条件

- 已成功安装集群，包含 Kafka 及 Flume 服务。
- 已安装 flume 客户端，客户端安装请参见“组件操作指南 > 使用 Flume > 安装 Flume 客户端”。
- 确保集群网络环境安全。
- 系统管理员已明确业务需求，并准备一个 Kafka 管理员用户 **flume_kafka**。

操作步骤

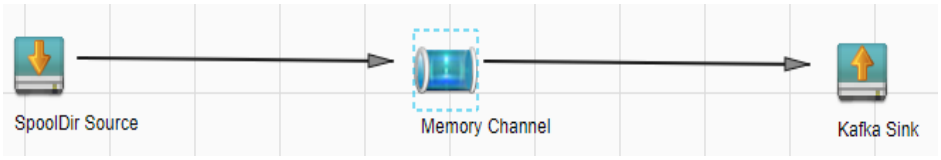
步骤 1 配置 Flume 的参数。

使用 Manager 界面中的 Flume 配置工具来配置 Flume 角色客户端参数并生成配置文件。

1. 登录 FusionInsight Manager，选择“集群 > 服务 > Flume > 配置工具”。
2. “Agent 名”选择“client”，然后选择要使用的 source、channel 以及 sink，将其拖到右侧的操作界面中并将其连接。

   采用 SpoolDir Source、Memory Channel 和 Kafka Sink，如图 7-4 所示。

   图7-4 Flume 配置工具示例

3. 双击对应的 source、channel 以及 sink，根据实际环境并参考表 7-30 设置对应的配置参数。

说明

- 如果想在之前的“properties.propretites”文件上进行修改后继续使用，则登录 Manager，选择“集群 > 服务 > Flume > 配置工具 > 导入”，将该文件导入后再修改非加密传输的相关配置项即可。
- 导入配置文件时，建议配置 Source/Channel/Sink 的各自的个数都不要超过 40 个，否则可能导致界面响应时间过长。

表7-30 Flume 角色客户端所需修改的参数列表

参数名称	参数值填写规则	参数样例
名称	不能为空，必须唯一	test
spoolDir	待采集的文件所在的目录路径，此参数不能为空。该路径需存在，且对 flume 运行用户有读写执行权限。	/srv/BigData/hadoop/data1/zb
trackerDir	flume 采集文件信息元数据保存路径。	/srv/BigData/hadoop/data1/tracker
batchSize	Flume 一次发送的事件个数（数据条数）。增大会提升性能，降低实时性；反之降低性能，提升实时性。	61200
kafka.topics	订阅的 Kafka topic 列表，多个 topic 用逗号分隔，此参数不能为空。	test1
kafka.bootstrap.servers	Kafka 的 bootstrap 地址端口列表，默认值为 Kafka 集群中所有的 Kafkabrokers。	192.168.101.10:21007

4. 单击“导出”，将配置文件“properties.properties”保存到本地。

步骤 2 上传配置文件。

将步骤 1.4 导出的文件上传至集群的“*Flume 客户端安装目录*/fusioninsight-flume-*Flume 组件版本号*/conf”目录下。

步骤 3 验证日志是否传输成功。

1. 登录 Kafka 客户端：

   cd *Kafka 客户端安装目录*/**Kafka/kafka**

   kinit flume_kafka（输入密码）

2. 读取 KafkaTopic 中的数据（修改命令中的中文为实际参数）。

bin/kafka-console-consumer.sh --topic *主题名称* **--bootstrap-server** *Kafka 角色实例所在节点的业务 IP 地址*:**21007 --consumer.config config/consumer.properties --from-beginning**

系统显示待采集文件目录下的内容：

```
[root@host1 kafka]# bin/kafka-console-consumer.sh --topic test1 --bootstrap-server 192.168.101.10:21007 --consumer.config config/consumer.properties --from-beginning
Welcome to flume
```

----结束

7.10.3 典型场景：从本地采集静态日志保存到 HDFS

操作场景

该任务指导用户使用 Flume 客户端从本地采集静态日志保存到 HDFS 上“/flume/test”目录下。

本章节适用于 MRS 3.x 及之后版本。

说明

本配置默认集群网络环境是安全的，数据传输过程不需要启用 SSL 认证。如需使用加密方式，请参考配置加密传输。该配置为只用一个 Flume 场景，例如：Spooldir Source+Memory Channel+HDFS Sink。

前提条件

- 已成功安装集群，包含 HDFS 及 Flume 服务。
- 已安装 flume 客户端，客户端安装请参见“组件操作指南 > 使用 Flume > 安装 Flume 客户端”。
- 确保集群网络环境安全。
- 已创建用户 **flume_hdfs** 并授权验证日志时操作的 HDFS 目录和数据。

操作步骤

步骤 1 在 FusionInsight Manager 管理界面，选择“系统 > 权限 > 用户”，选择用户 **flume_hdfs**，选择“更多 > 下载认证凭据”下载 Kerberos 证书文件并保存在本地。

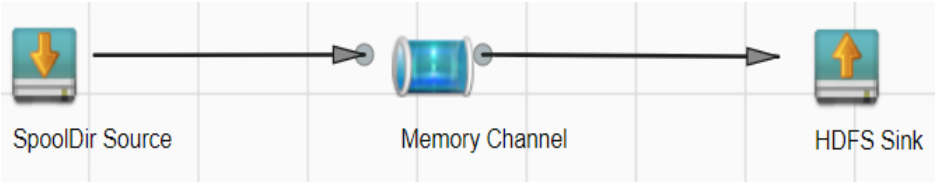
步骤 2 配置 Flume 参数。

使用 FusionInsight Manager 界面中的 Flume 来配置 Flume 角色客户端参数并生成配置文件。

1. 登录 FusionInsight Manager，选择“集群 > 服务 > Flume > 配置工具”。
2. “Agent 名”选择“client”，然后选择要使用的 source、channel 以及 sink，将其拖到右侧的操作界面中并将其连接。

   采用 SpoolDir Source、Memory Channel 和 HDFS Sink，如图 7-5 所示。

图7-5 Flume 配置工具示例



3. 双击对应的 source、channel 以及 sink，根据实际环境并参考表 7-31 设置对应的配置参数。

说明

- 如果想在之前的“properties.propretites”文件上进行修改后继续使用，则登录 FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > Flume > 配置工具 > 导入”，将该文件导入后再修改非加密传输的相关配置项即可。
- 导入配置文件时，建议配置 source/channel/sink 的各自的个数都不要超过 40 个，否则可能导致界面响应时间过长。

表7-31 Flume 角色客户端所需修改的参数列表

参数名称	参数值填写规则	参数样例
名称	不能为空，必须唯一。	test
spoolDir	待采集的文件所在的目录路径，此参数不能为空。该路径需存在，且对 flume 运行用户有读写执行权限。	/srv/BigData/hadoop/data1/zb
trackerDir	flume 采集文件信息元数据保存路径。	/srv/BigData/hadoop/data1/tracker
batchSize	Flume 一次发送数据的最大事件数。	61200
hdfs.path	写入 HDFS 的目录，此参数不能为空。	hdfs://hacluster/flume/test
hdfs.filePrefix	数据写入 HDFS 后文件名的前缀。	TMP_
hdfs.batchSize	一次写入 HDFS 的最大事件数目。	61200
hdfs.kerberosPrincipal	kerberos 认证时用户，在安全版本下必须填写。安全集群需要配置此项，普通模式集群无需配置。	flume_hdfs
hdfs.kerberosKeytab	kerberos 认证时 keytab 文件路径，在安全版本下必须填写。	/opt/test/conf/user.keytab

参数名称	参数值填写规则	参数样例
	安全集群需要配置此项，普通模式集群无需配置。	说明 user.keytab 文件从下载用户 **flume_hdfs** 的 kerberos 证书文件中获取，另外，确保用于安装和运行 Flume 客户端的用户对 user.keytab 文件有读写权限。
hdfs.useLocalTimeStamp	是否使用本地时间，取值为"true"或者"false"。	true

4. 单击“导出”，将配置文件“properties.properties”保存到本地。

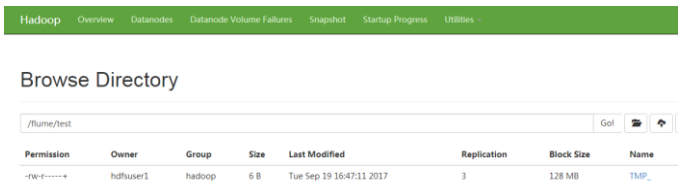
步骤 3 上传配置文件。

将步骤 2.4 导出的文件上传至集群的“*Flume 客户端安装目录*/fusioninsight-flume-*Flume 组件版本号*/conf”目录下。

步骤 4 验证日志是否传输成功。

1. 以具有 HDFS 组件管理权限的用户登录 FusionInsight Manager，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本）。在 FusionInsight Manager 界面选择“集群 > 服务 > HDFS”，单击“NameNode(主)”对应的链接，打开 HDFS WebUI，然后选择“Utilities > Browse the file system”。
2. 观察 HDFS 上“/flume/test”目录下是否有产生数据。

图7-6 查看 HDFS 目录和文件



----结束

7.10.4 典型场景：从本地采集动态日志保存到 HDFS

操作场景

该任务指导用户使用 Flume 客户端从本地采集动态日志保存到 HDFS 上“/flume/test”目录下。

本章节适用于 MRS 3.x 及之后版本。

说明

本配置默认集群网络环境是安全的，数据传输过程不需要启用 SSL 认证。如需使用加密方式，请参考配置加密传输。该配置为只用一个 Flume 场景，例如：Taildir Source+Memory Channel+HDFS Sink。

前提条件

- 已成功安装集群，包含 HDFS 及 Flume 服务。
- 已安装 flume 客户端，客户端安装请参见“组件操作指南 > 使用 Flume > 安装 Flume 客户端”。
- 确保集群网络环境安全。
- 已创建用户 **flume_hdfs** 并授权验证日志时操作的 HDFS 目录和数据。

操作步骤

步骤 1 在 FusionInsight Manager 管理界面，选择“系统 > 权限 > 用户”，选择“更多 > 下载认证凭据”下载用户 **flume_hdfs** 的 kerberos 证书文件并保存在本地。

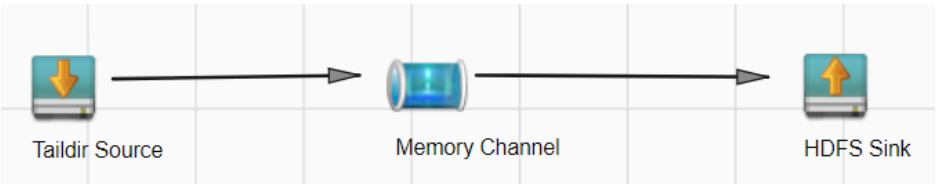
步骤 2 配置 Flume 参数。

使用 FusionInsight Manager 界面中的 Flume 配置工具来配置 Flume 角色客户端参数并生成配置文件。

1. 登录 FusionInsight Manager，选择“集群 > 服务 > Flume > 配置工具”。
2. “Agent 名”选择“client”，然后选择要使用的 source、channel 以及 sink，将其拖到右侧的操作界面中并将其连接。

   采用 Taildir Source、Memory Channel 和 HDFS Sink，如图 7-7 所示。

   图7-7 Flume 配置工具示例



3. 双击对应的 Source、Channel 以及 Sink，根据实际环境并参考表 7-32 设置对应的配置参数。

说明

- 如果想在之前的“properties.propretites”文件上进行修改后继续使用，则登录 FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > Flume > 配置工具 > 导入”，将该文件导入后再修改非加密传输的相关配置项即可。
- 导入配置文件时，建议配置 Source/Channel/Sink 的各自的个数都不要超过 40 个，否则可能导致界面响应时间过长。

表7-32 Flume 角色客户端所需修改的参数列表

参数名称	参数值填写规则	参数样例
名称	不能为空，必须唯一。	test
filegroups	文件分组列表名，此参数不能为空。该值包含如下两项参数： • 名称：文件分组列表名。 • filegroups：动态日志文件绝对路径。	-
positionFile	保存当前采集文件信息（文件名和已经采集的位置），此参数不能为空。该文件不需要手工创建，但其上层目录需对 flume 运行用户可写。	/home/omm/flume/positionfile
batchSize	Flume 一次发送数据的最大事件数。	61200
hdfs.path	写入 HDFS 的目录，此参数不能为空。	hdfs://hacluster/flume/test
hdfs.filePrefix	数据写入 HDFS 后文件名的前缀。	TMP_
hdfs.batchSize	一次写入 HDFS 的最大事件数目。	61200
hdfs.kerberosPrincipal	kerberos 认证时用户，在安全版本下必须填写。安全集群需要配置此项，普通模式集群无需配置。	flume_hdfs
hdfs.kerberosKeytab	kerberos 认证时 keytab 文件路径，在安全版本下必须填写。安全集群需要配置此项，普通模式集群无需配置。	/opt/test/conf/user.keytab 说明 user.keytab 文件从下载用户 **flume_hdfs** 的 kerberos 证书文件中获取，另外，确保用于安装和运行 Flume 客户端的用户对 user.keytab 文件有读写权限。
hdfs.useLocalTimeStamp	是否使用本地时间，取值为"true"或者"false"。	true

4. 单击“导出”，将配置文件“properties.properties”保存到本地。

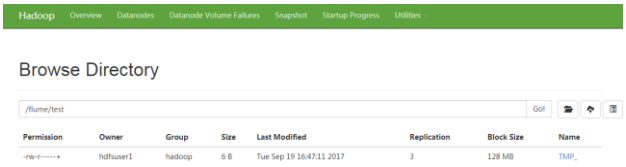
步骤 3 上传配置文件。

将步骤 2.4 导出的文件上传至集群的“*Flume 客户端安装目录*/fusioninsight-flume-*Flume 组件版本号*/conf”目录下。

步骤 4 验证日志是否传输成功。

1. 以具有 HDFS 组件管理权限的用户登录 FusionInsight Manager，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本）。在 FusionInsight Manager 界面选择“集群 > 服务 > HDFS”，单击“NameNode(*节点名称*，主)”对应的链接，打开 HDFS WebUI，然后选择“Utilities > Browse the file system”。
2. 观察 HDFS 上“/flume/test”目录下是否有产生数据。

图7-8 查看 HDFS 目录和文件



----结束

7.10.5 典型场景：从 Kafka 采集日志保存到 HDFS

操作场景

该任务指导用户使用 Flume 客户端从 Kafka 的 Topic 列表(test1)采集日志保存到 HDFS 上“/flume/test”目录下。

本章节适用于 MRS 3.x 及之后版本。

说明

本配置默认集群网络环境是安全的，数据传输过程不需要启用 SSL 认证。如需使用加密方式，请参考配置加密传输。该配置为只用一个 Flume 场景，例如：Kafka Source+Memory Channel+HDFS Sink.

前提条件

- 已成功安装集群，包含 HDFS、Kafka 及 Flume 服务。
- 已安装 flume 客户端，客户端安装请参见“组件操作指南 > 使用 Flume > 安装 Flume 客户端”。
- 确保集群网络环境安全。
- 已创建用户 **flume_hdfs** 并授权验证日志时操作的 HDFS 目录和数据。

操作步骤

步骤 1 在 FusionInsight Manager 管理界面，选择“系统 > 权限 > 用户”，选择“更多 > 下载认证凭据”下载用户 **flume_hdfs** 的 kerberos 证书文件并保存在本地。

步骤 2 配置 Flume 角色客户端参数。

使用 FusionInsight Manager 界面中的 Flume 配置工具来配置 Flume 角色客户端参数并生成配置文件。

1. 登录 FusionInsight Manager，选择“集群 > 服务 > Flume > 配置工具”。
2. “Agent 名”选择“client”，然后选择要使用的 source、channel 以及 sink，将其拖到右侧的操作界面中并将其连接。

   例如采用 Kafka Source、Memory Channel 和 HDFS Sink，如图 7-9 所示。

   图7-9 Flume 配置工具示例

3. 双击对应的 source、channel 以及 sink，根据实际环境并参考表 7-33 设置对应的配置参数。

说明

- 如果想在之前的“properties.propretites”文件上进行修改后继续使用，则登录 FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > Flume > 配置工具 > 导入”，将该文件导入后再修改非加密传输的相关配置项即可。
- 导入配置文件时，建议配置 source/channel/sink 的各自的个数都不要超过 40 个，否则可能导致界面响应时间过长。

表7-33 Flume 角色客户端所需修改的参数列表

参数名称	参数值填写规则	参数样例
名称	不能为空，必须唯一。	test
kafka.topics	订阅的 Kafka topic 列表，用逗号分隔，此参数不能为空。	test1
kafka.consumer.group.id	从 Kafka 中获取数据的组标识，此参数不能为空。	flume
kafka.bootstrap.servers	Kafka 的 bootstrap 地址端口列表,默认值为 Kafka 集群中所有的 Kafka 列表。如果集群安装有 Kafka 并且配置已经同步，可以不配置此项。	192.168.101.10:9092
batchSize	Flume 一次发送的事件个数（数据条数）。	61200
hdfs.path	写入 HDFS 的目录，此参数	hdfs://hacluster/flume/test

参数名称	参数值填写规则	参数样例
	不能为空。	
hdfs.filePrefix	数据写入 HDFS 后文件名的前缀。	TMP_
hdfs.batchSize	一次写入 HDFS 的最大事件数目。	61200
hdfs.kerberosPrincipal	kerberos 认证时用户，在安全版本下必须填写。安全集群需要配置此项，普通模式集群无需配置。	flume_hdfs
hdfs.kerberosKeytab	kerberos 认证时 keytab 文件路径，在安全版本下必须填写。安全集群需要配置此项，普通模式集群无需配置。	/opt/test/conf/user.keytab **说明** user.keytab 文件从下载用户 **flume_hdfs** 的 kerberos 证书文件中获取，另外，确保用于安装和运行 Flume 客户端的用户对 user.keytab 文件有读写权限。
hdfs.useLocalTimeStamp	是否使用本地时间，取值为"true"或者"false"。	true

4. 单击“导出”，将配置文件“properties.properties”保存到本地。

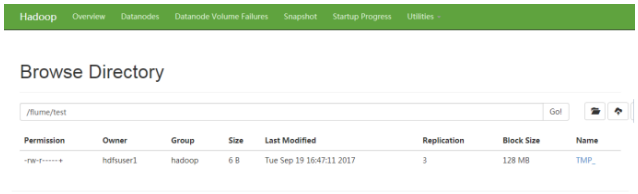
步骤 3 上传配置文件。

将步骤 2.4 导出的文件上传至集群的“*Flume 客户端安装目录*/fusioninsight-flume-*Flume 组件版本号*/conf”目录下。

步骤 4 验证日志是否传输成功。

1. 以具有 HDFS 组件管理权限的用户登录 FusionInsight Manager，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本）。在 FusionInsight Manager 界面选择“集群 > 服务 > HDFS”，单击“NameNode(*节点名称*，主)”对应的链接，打开 HDFS WebUI，然后选择“Utilities > Browse the file system”。
2. 观察 HDFS 上“/flume/test”目录下是否有产生数据。

图7-10 查看 HDFS 目录和文件



----结束

7.10.6 典型场景：从 Kafka 客户端采集日志经 Flume 客户端保存到 HDFS

操作场景

该任务指导用户使用 Flume 客户端从 Kafka 客户端的 Topic 列表(test1)采集日志保存到 HDFS 上“/flume/test”目录下。

本章节适用于 MRS 3.x 及之后版本。

说明

本配置默认集群网络环境是安全的，数据传输过程不需要启用 SSL 认证。如需使用加密方式，请参考配置加密传输。

前提条件

- 已成功安装集群，包含 HDFS、Kafka 及 Flume 服务。
- 已安装 flume 客户端，客户端安装请参见“组件操作指南 > 使用 Flume > 安装 Flume 客户端”。
- 已创建用户 **flume_hdfs** 并授权验证日志时操作的 HDFS 目录和数据。
- 确保集群网络环境安全。

操作步骤

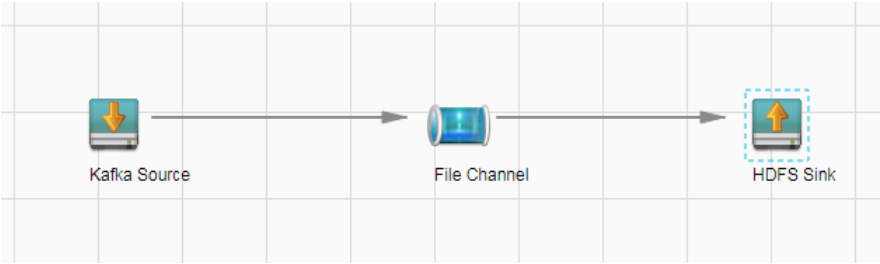
步骤 1 在 FusionInsight Manager 管理界面，选择“系统 > 权限 > 用户”，选择“更多 > 下载认证凭据”下载用户 **flume_hdfs** 的 kerberos 证书文件并保存在本地。

步骤 2 配置 Flume 角色客户端参数。

1. 使用 FusionInsight Manager 界面中的 Flume 配置工具来配置 Flume 角色客户端参数并生成配置文件。
   a. 登录 FusionInsight Manager，选择“集群 > 服务 > Flume > 配置工具”。
   b. “Agent 名”选择“client”，然后选择要使用的 source、channel 以及 sink，将其拖到右侧的操作界面中并将其连接。

      例如采用 Kafka Source、File Channel 和 HDFS Sink，如图 7-11 所示。

图7-11 Flume 配置工具示例



c. 双击对应的 source、channel 以及 sink，根据实际环境并参考表 7-34 设置对应的配置参数。

说明

- 如果想在之前的“properties.propretites”文件上进行修改后继续使用，则登录 FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > Flume > 配置工具 > 导入”，将该文件导入后再修改非加密传输的相关配置项即可。
- 导入配置文件时，建议配置 source/channel/sink 的各自的个数都不要超过 40 个，否则可能导致界面响应时间过长。

表7-34 Flume 角色客户端所需修改的参数列表

参数名称	参数值填写规则	参数样例
名称	不能为空，必须唯一。	test
kafka.topics	订阅的 Kafka topic 列表，用逗号分隔，此参数不能为空。	test1
kafka.consumer.group.id	从 Kafka 中获取数据的组标识，此参数不能为空。	flume
kafka.bootstrap.servers	Kafka 的 bootstrap 地址端口列表,默认值为 Kafka 集群中所有的 Kafka 列表。如果集群安装有 Kafka 并且配置已经同步，可以不配置此项。	192.168.101.10:21007
batchSize	Flume 一次发送的事件个数（数据条数）。	61200
dataDirs	缓冲区数据保存目录，默认为运行目录。配置多个盘上的目录可以提升传输效率，多个目录使用逗号分隔。如果为集群内，则可以指定在如下目录 /srv/BigData/hadoop/dataX/	/srv/BigData/hadoop/data1/flume/data

参数名称	参数值填写规则	参数样例
	flume/data，dataX 为 data1~dataN。如果为集群外，则需要单独规划。	
checkpointDir	checkpoint 信息保存目录，默认在运行目录下。如果为集群内，则可以指定在如下目录 /srv/BigData/hadoop/dataX/flume/checkpoint，dataX 为 data1~dataN。如果为集群外，则需要单独规划。	/srv/BigData/hadoop/data1/flume/checkpoint
transactionCapacity	事务大小：即当前 channel 支持事务处理的事件个数，建议和 Source 的 batchSize 设置为同样大小，不能小于 batchSize。	61200
hdfs.path	写入 HDFS 的目录，此参数不能为空。	hdfs://hacluster/flume/test
hdfs.filePrefix	数据写入 HDFS 后文件名的前缀。	TMP_
hdfs.batchSize	一次写入 HDFS 的最大事件数目。	61200
hdfs.kerberosPrincipal	kerberos 认证时用户，在安全版本下必须填写。安全集群需要配置此项，普通模式集群无需配置。	flume_hdfs
hdfs.kerberosKeytab	kerberos 认证时 keytab 文件路径，在安全版本下必须填写。安全集群需要配置此项，普通模式集群无需配置。	/opt/test/conf/user.keytab **说明** user.keytab 文件从下载用户 **flume_hdfs** 的 kerberos 证书文件中获取，另外，确保用于安装和运行 Flume 客户端的用户对 user.keytab 文件有读写权限。
hdfs.useLocalTimeStamp	是否使用本地时间，取值为"true"或者"false"	true

d. 单击“导出”，将配置文件“properties.properties”保存到本地。

2. 将“properties.properties”文件上传到 Flume 客户端安装目录下的“flume/conf/”下。
3. Flume 客户端连接到 HDFS，还需要补充如下配置：
   a. 通过“用户”下载用户 **flume_hdfs** 的 kerberos 证书文件获取 krb5.conf 配置文件，并上传至客户端所在节点安装目录的“fusioninsight-flume-1.9.0/conf/”下。
   b. 新建 jaas.conf 配置文件到客户端所在节点安装目录的“fusioninsight-flume-1.9.0/conf/”下。

      vi jaas.conf

      ```
      KafkaClient {
      com.sun.security.auth.module.Krb5LoginModule required
      useKeyTab=true
      keyTab="/opt/test/conf/user.keytab"
      principal="flume_hdfs@<系统域名>"
      useTicketCache=false
      storeKey=true
      debug=true;
      };
      ```

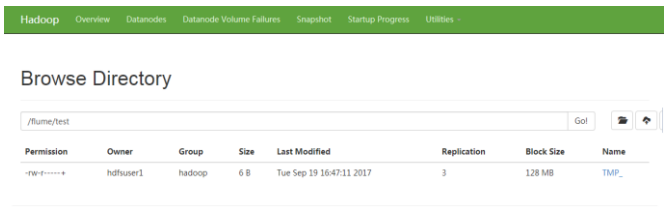
      参数 keyTab 和 principal 根据实际情况修改。
   c. 从/opt/FusionInsight_Cluster_<*集群 ID*>_Flume_ClientConfig/Flume/config 目录下获取 core-site.xml 和 hdfs-site.xml 配置文件，并上传至客户端所在节点安装目录的“fusioninsight-flume-1.9.0/conf/”下。
4. 执行以下命令重启 Flume 进程。

   flume-manager.sh restart

步骤 3 验证日志是否传输成功。

1. 以具有 HDFS 组件管理权限的用户登录 FusionInsight Manager，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本）。在 FusionInsight Manager 界面选择“集群 > 服务 > HDFS”，单击“NameNode(*节点名称*，主)”对应的链接，打开 HDFS WebUI，然后选择“Utilities > Browse the file system”。
2. 观察 HDFS 上“/flume/test”目录下是否有产生数据。

图7-12 查看 HDFS 目录和文件



----结束

7.10.7 典型场景：从本地采集静态日志保存到 HBase

操作场景

该任务指导用户使用 Flume 客户端从本地采集静态日志保存到 HBase 表：flume_test。该场景介绍的是多级 agent 串联操作

本章节适用于 MRS 3.x 及之后版本。

说明

本配置默认集群网络环境是安全的，数据传输过程不需要启用 SSL 认证。如需使用加密方式，请参考配置加密传输。该配置可以只用一个 Flume 场景，例如 Server：Spooldir Source+File Channel+HBase Sink。

前提条件

- 已成功安装集群，包含 HBase 及 Flume 服务。
- 已安装 flume 客户端，客户端安装请参见“组件操作指南 > 使用 Flume > 安装 Flume 客户端”。
- 确保集群网络环境安全。
- 已创建 HBase 表：**create 'flume_test', 'cf'**。
- 系统管理员已明确业务需求，并准备一个 HBase 管理员用户 **flume_hbase**。

操作步骤

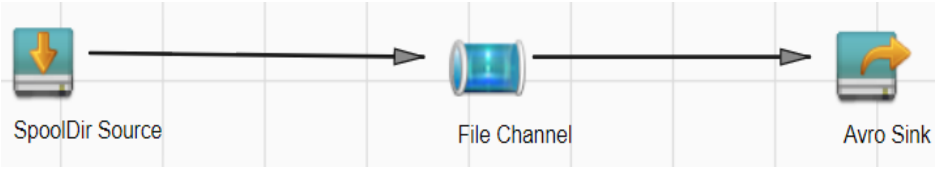
步骤 1 在 FusionInsight Manager 管理界面，选择“系统 > 权限 > 用户”，选择“更多 > 下载认证凭据”下载用户 **flume_hbase** 的 kerberos 证书文件并保存在本地。

步骤 2 配置 Flume 角色客户端参数。

1. 使用 FusionInsight Manager 界面中的 Flume 配置工具来配置 Flume 角色客户端参数并生成配置文件。
   a. 登录 FusionInsight Manager，选择“集群 > 服务 > Flume > 配置工具”。
   b. “Agent 名”选择“client”，然后选择要使用的 source、channel 以及 sink，将其拖到右侧的操作界面中并将其连接。

      采用 SpoolDir Source、File Channel 和 Avro Sink，如图 7-13 所示。

      图7-13 Flume 配置工具示例

   c. 双击对应的 source、channel 以及 sink，根据实际环境并参考表 7-35 设置对应的配置参数。

说明

- 如果想在之前的“properties.propretites”文件上进行修改后继续使用，则登录 FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > Flume > 配置工具 > 导入”，将该文件导入后再修改非加密传输的相关配置项即可。
- 导入配置文件时，建议配置 source/channel/sink 的各自的个数都不要超过 40 个，否则可能导致界面响应时间过长。

表7-35 Flume 角色客户端所需修改的参数列表

参数名称	参数值填写规则	参数样例
名称	不能为空，必须唯一。	test
spoolDir	待采集的文件所在的目录路径，此参数不能为空。该路径需存在，且对 flume 运行用户有读写执行权限。	/srv/BigData/hadoop/data1/zb
trackerDir	flume 采集文件信息元数据保存路径。	/srv/BigData/hadoop/data1/tracker
batchSize	Flume 一次发送的事件个数（数据条数）。增大会提升性能，降低实时性；反之降低性能，提升实时性。	61200
dataDirs	缓冲区数据保存目录，默认为运行目录。配置多个盘上的目录可以提升传输效率，多个目录使用逗号分隔。如果为集群内，则可以指定在如下目录 /srv/BigData/hadoop/dataX/flume/data，dataX 为 data1~dataN。如果为集群外，则需要单独规划。	/srv/BigData/hadoop/data1/flume/data
checkpointDir	checkpoint 信息保存目录，默认在运行目录下。如果为集群内，则可以指定在如下目录 /srv/BigData/hadoop/dataX/flume/checkpoint，dataX 为 data1~dataN。如果为集群外，则需要单独规划。	/srv/BigData/hadoop/data1/flume/checkpoint
transactionCapacity	事务大小：即当前 channel 支持事务处理的事件个	61200

参数名称	参数值填写规则	参数样例
	数，建议和 Source 的 batchSize 设置为同样大小，不能小于 batchSize。	
hostname	要发送数据的主机名或者IP，此参数不能为空。须配置为与之相连的 avro source 所在的主机名或IP。	192.168.108.11
port	要发送数据的端口，此参数不能为空。须配置为与之相连的 avro source 监听的端口。	21154
ssl	是否启用 SSL 认证（基于安全要求，建议启用此功能）。 只有“Avro”类型的 Source 才有此配置项。 • true 表示启用 • false 表示不启用	false

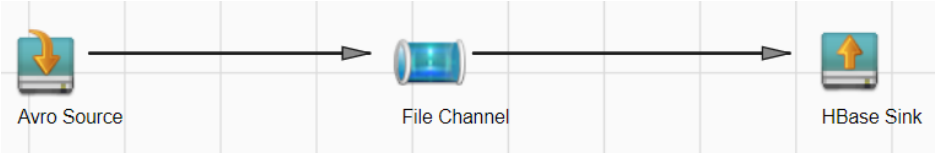
d. 单击“导出”，将配置文件“properties.properties”保存到本地。

2. 将“properties.properties”文件上传到 Flume 客户端安装目录下的“flume/conf/”下。

步骤 3 配置 Flume 角色的服务端参数，并将配置文件上传到集群。

1. 使用 FusionInsight Manager 界面中的 Flume 配置工具来配置服务端参数并生成配置文件。
   a. 登录 FusionInsight Manager，选择“集群 > 服务 > Flume > 配置工具”。
   b. “Agent 名”选择“server”，然后选择要使用的 source、channel 以及 sink，将其拖到右侧的操作界面中并将其连接。
      采用 Avro Source、File Channel 和 HBase Sink，如图 7-14 所示。

      图7-14 Flume 配置工具示例

   c. 双击对应的 source、channel 以及 sink，根据实际环境并参考表 7-36 设置对应的配置参数。

说明

- 如果对应的 Flume 角色之前已经配置过服务端参数，为保证与之前的配置保持一致，在 FusionInsight Manager 界面选择“集群 > *待操作集群的名称* > 服务 > Flume > 实例”，选择相应的 Flume 角色实例，单击“实例配置”页面“flume.config.file”参数后的“下载文件”，可获取已有的服务端参数配置文件。然后选择“集群 > *待操作集群的名称* > 服务 > Flume > 配置工具 > 导入”，将该文件导入后再修改非加密传输的相关配置项即可。
- 导入配置文件时，建议配置 source/channel/sink 的各自的个数都不要超过 40 个，否则可能导致界面响应时间过长。
- 不同的 File Channel 均需要配置一个不同的 checkpoint 目录。

表7-36 Flume 角色服务端所需修改的参数列表

参数名称	参数值填写规则	参数样例
名称	不能为空，必须唯一。	test
bind	avro source 绑定的 ip 地址，此参数不能为空。须配置为服务端配置文件即将要上传的主机 IP。	192.168.108.11
port	avro source 监听的端口，此参数不能为空。须配置为未被使用的端口。	21154
ssl	是否启用 SSL 认证（基于安全要求，建议启用此功能）。 只有“Avro”类型的 Source 才有此配置项。 • true 表示启用 • false 表示不启用	false
dataDirs	缓冲区数据保存目录，默认为运行目录。配置多个盘上的目录可以提升传输效率，多个目录使用逗号分隔。如果为集群内，则可以指定在如下目录 /srv/BigData/hadoop/dataX/flume/data，dataX 为 data1~dataN。如果为集群外，则需要单独规划。	/srv/BigData/hadoop/data1/flumeserver/data
checkpointDir	checkpoint 信息保存目录，默认在运行目录下。如果为集群内，则可以指定在如下目录 /srv/BigData/hadoop/dataX/flume/checkpoint，dataX 为 data1~dataN。如果为集群外，则需要单独规划。	/srv/BigData/hadoop/data1/flumeserver/checkpoint
transactionCapacity	事务大小：即当前 channel 支持事务处理的事件个数。建议和 Source 的 batchSize 设置为同样大小，不能小于	61200

参数名称	参数值填写规则	参数样例
	batchSize。	
table	HBase 表名，此参数不能为空。	flume_test
columnFamily	HBase 列族名，此参数不能为空。	cf
batchSize	Flume 一次写入 HBase 中的最大事件数。	61200
kerberosPrincipal	kerberos 认证时用户,在安全版本下必须填写。安全集群需要配置此项，普通模式集群无需配置。	flume_hbase
kerberosKeytab	kerberos 认证时文件路径，,在安全版本下必须填写。安全集群需要配置此项，普通模式集群无需配置。	/opt/test/conf/user.keytab 说明 user.keytab 文件从下载用户 **flume_hbase** 的 kerberos 证书文件中获取，另外，确保用于安装和运行 Flume 客户端的用户对 user.keytab 文件有读写权限。

d. 单击“导出”，将配置文件“properties.properties”保存到本地。

2. 登录 FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > Flume”，在“实例”下单击“Flume”角色。
3. 选择准备上传配置文件的节点行的“Flume”角色，单击“实例配置”页面“flume.config.file”参数后的“上传文件”，选择“properties.properties”文件完成操作。

说明

- 每个 Flume 实例均可以上传单独的服务端配置文件。
- 更新配置文件需要按照此步骤操作，后台修改配置文件是不规范操作，同步配置时后台做的修改将会被覆盖。

4. 单击“保存”，单击“确定”。
5. 单击“完成”完成操作。

步骤 4 验证日志是否传输成功。

1. 进入 HBase 客户端目录：

   cd */客户端安装目录***/HBase/hbase**

   kinit flume_hbase（输入密码）
2. 执行 **hbase shell** 进入 HBase 客户端。
3. 执行语句：**scan 'flume_test'** ，可以看到日志按行写入 HBase 列族里。

```
hbase(main):001:0> scan 'flume_test'
ROW                                                   COLUMN+CELL
2017-09-18 16:05:36,394 INFO  [hconnection-0x415a3f6a-shared--pool2-t1]
ipc.AbstractRpcClient: RPC Server Kerberos principal name for
service=ClientService is hbase/hadoop.<系统域名>@<系统域名>
 default4021ff4a-9339-4151-a4d0-00f20807e76d              column=cf:pCol,
timestamp=1505721909388, value=Welcome to flume
 incRow                                                  column=cf:iCol,
timestamp=1505721909461, value=\x00\x00\x00\x00\x00\x00\x00\x01
2 row(s) in 0.3660 seconds
```

----结束

7.11 加密传输

7.11.1 配置加密传输

操作场景

该操作指导安装工程师在集群安装完成后，分别设置 Flume 服务（包括 Flume 角色和 MonitorServer 角色）的服务端和客户端参数，使其可以正常工作。

本章节适用于 MRS 3.x 及之后版本。

前提条件

已成功安装集群及 Flume 服务。

操作步骤

步骤 1 分别生成 Flume 角色服务端和客户端的证书和信任列表。

1. 使用 ECM 远程以 **omm** 用户登录将要安装 Flume 服务端的节点。进入“${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/bin”目录。

   cd ${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/bin

   说明

   此处版本号 8.1.0.1 为示例，具体以实际环境的版本号为准。

2. 执行以下命令，生成并导出 Flume 角色服务端、客户端证书。

   sh geneJKS.sh -f *xxx* **-g** *xxx*

   生成的证书在“${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/conf”路径下。其中：

   - “flume_sChat.jks”是 Flume 角色服务端的证书库，“flume_sChat.crt”是“flume_sChat.jks”证书的导出文件，“-f”配置项是证书和证书库的密码；

- “flume_cChat.jks”是 Flume 角色客户端的证书库，“flume_cChat.crt”是“flume_cChat.jks”证书的导出文件，“-g”配置项是证书和证书库的密码；
- “flume_sChatt.jks”和“flume_cChatt.jks”分别为 Flume 服务端、客户端 SSL 证书信任列表。

说明

本章节涉及到所有的用户自定义密码（如 xxx），需满足以下复杂度要求：

- 至少包含大写字母、小写字母、数字、特殊符号 4 种类型字符。
- 至少 8 位，最多 64 位。
- 出于安全考虑，建议用户定期更换自定义密码（例如三个月更换一次），并重新生成各项证书和信任列表。

步骤 2 配置 Flume 角色的服务端参数，并将配置文件上传到集群。

1. 使用 ECM 远程，以 **omm** 用户登录任意一个 Flume 角色所在的节点。执行以下命令进入“${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/bin”。

   cd ${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/bin

2. 执行以下命令，生成并得到 Flume 服务端密钥库密码、信任列表密码和 keystore-password 加密的私钥信息。连续输入两次密码并确认，该密码是 *flume_sChat.jks* 证书库的密码。

   ./genPwFile.sh

   cat password.property

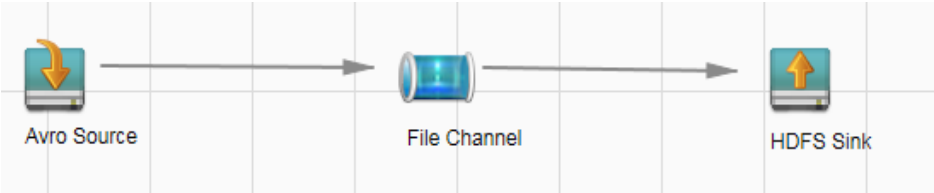
3. 使用 FusionInsight Manager 界面中的 Flume 配置工具来配置服务端参数并生成配置文件。

   a. 登录 FusionInsight Manager，选择“ 服务 > Flume > 配置工具”。

   b. “Agent 名”选择“server”，然后选择要使用的 source、channel 以及 sink，将其拖到右侧的操作界面中并将其连接。

   例如采用 Avro Source、File Channel 和 HDFS Sink，如图 7-15 所示。

   图7-15 Flume 配置工具示例

   c. 双击对应的 source、channel 以及 sink，根据实际环境并参考表 7-37 设置对应的配置参数。

说明

- 如果对应的 Flume 角色之前已经配置过服务端参数，为保证与之前的配置保持一致，在 FusionInsight Manager 界面选择“ 服务 > Flume > 实例”，选择相应的 Flume 角色实例，单击“实例配置”页面“flume.config.file”参数后的“下载文件”，可获取已有的服务端参数配置文件。然后选择“ 服务 > Flume > 导入”，将该文件导入后再修改加密传输的相关配置项即可。
- 导入配置文件时，建议配置 Source/Channel/Sink 的各自的个数都不要超过 40 个，否则可能导致界面响应时间过长。

d. 单击“导出”，将配置文件“properties.properties”保存到本地。

表7-37 Flume 角色服务端所需修改的参数列表

参数名称	参数值填写规则	参数样例
ssl	是否启用 SSL 认证（基于安全要求，建议启用此功能）。 • true 表示启用。 • false 表示不启用。	true
keystore	服务端证书。	${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/conf/flume_sChat.jks
keystore-password	密钥库密码，获取 keystore 信息所需密码。 输入步骤 2.2 中获取的“password”值。	-
truststore	服务端的 SSL 证书信任列表。	${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/conf/flume_sChatt.jks
truststore-password	信任列表密码，获取 truststore 信息所需密码。 输入步骤 2.2 中获取的“password”值。	-

4. 登录 FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > Flume”服务，在“角色”下单击“Flume”角色。

5. 选择准备上传配置文件的节点行的“Flume”角色，单击“实例配置”页面“flume.config.file”参数后的“上传文件”，选择“properties.properties”文件完成操作。

说明

- 每个 Flume 实例均可以上传单独的服务端配置文件。
- 更新配置文件需要按照此步骤操作，后台修改配置文件是不规范操作，同步配置时后台做的修改将会被覆盖。

6. 单击“保存”，单击“确定”。单击“完成”完成操作。

步骤 3 设置 Flume 角色客户端参数。

1. 执行以下命令将生成的客户端证书（flume_cChat.jks）和客户端信任列表（flume_cChatt.jks）复制到客户端目录下，如“/opt/flume-client/fusionInsight-flume-1.9.0/conf/”（要求已安装 Flume 客户端），其中 **10.196.26.1** 为客户端所在节点业务平面的 IP 地址。

   scp ${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/conf/flume_cChat.jks user@10.196.26.1:/opt/flume-client/fusionInsight-flume-1.9.0/conf/

   scp ${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/conf/flume_cChatt.jks user@10.196.26.1:/opt/flume-client/fusionInsight-flume-1.9.0/conf/

   说明

   复制过程中需要输入客户端所在主机（如 **10.196.26.1**）**user** 用户的密码。

2. 以 **user** 用户登录解压 Flume 客户端的节点。执行以下命令进入客户端目录“opt/flume-client/fusionInsight-flume-1.9.0/bin”。

   cd opt/flume-client/fusionInsight-flume-1.9.0/bin

3. 执行以下命令，生成并得到 Flume 客户端密钥库密码、信任列表密码和 keystore-password 加密的私钥信息。连续输入两次密码并确认，该密码是别名为 *flumechatclient* 的证书和 *flume_cChat.jks* 证书库的密码。

   ./genPwFile.sh

   cat password.property

   说明

   若产生以下错误提示，可执行命令 **export JAVA_HOME=***JDK 路径*进行处理。

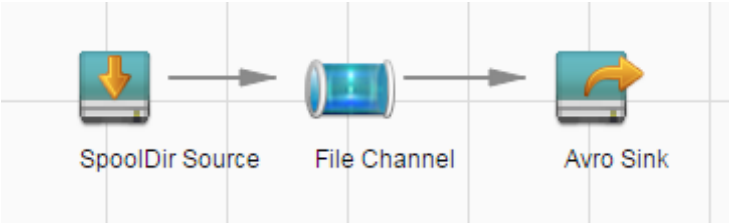
   ```
   JAVA_HOME is null in current user,please install the JDK and set the JAVA_HOME
   ```

4. 执行 **echo $SCC_PROFILE_DIR** 检查 SCC_PROFILE_DIR 环境变量是否为空。
   - 是，执行 **source .sccfile**。
   - 否，执行步骤 3.5。

5. 使用 FusionInsight Manager 界面中的 Flume 配置工具来配置 Flume 角色客户端参数并生成配置文件。
   a. 登录 FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > Flume > 配置工具”。
   b. “Agent 名”选择“client”，然后选择要使用的 source、channel 以及 sink，将其拖到右侧的操作界面中并将其连接。

例如采用 SpoolDir Source、File Channel 和 Avro Sink，如图 7-16 所示。

图7-16 Flume 配置工具示例



c. 双击对应的 source、channel 以及 sink，根据实际环境并参考表 7-38 设置对应的配置参数。

说明

- 如果对应的 Flume 角色之前已经配置过客户端参数，为保证与之前的配置保持一致，可以到“客户端安装目录/fusioninsight-flume-1.9.0/conf/properties.properties”获取已有的客户端参数配置文件。然后登录 FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > Flume > 配置工具 > 导入”，将该文件导入后再修改加密传输的相关配置项即可。
- 导入配置文件时，建议配置 source/channel/sink 的各自的个数都不要超过 40 个，否则可能导致界面响应时间过长。
- 不同的 File Channel 均需要配置一个不同的 checkpoint 目录。

d. 单击“导出”，将配置文件“properties.properties”保存到本地。

表7-38 Flume 角色客户端所需修改的参数列表

参数名称	参数值填写规则	参数样例
ssl	是否启用 SSL 认证（基于安全要求，建议用户启用此功能）。 • true 表示启用。 • false 表示不启用。	true
keystore	客户端证书。	/opt/flume-client/fusionInsight-flume-1.9.0/conf/flume_cChat.jks
keystore-password	密钥库密码，获取 keysore 信息所需密码。 输入步骤 3.3 中获取的“password”值。	-
truststore	客户端的 SSL 证书信任列表。	/opt/flume-client/fusionInsight-flume-1.9.0/conf/flume_cChatt.jks
truststore-password	信任列表密码，获取	-

参数名称	参数值填写规则	参数样例
	truststore 信息所需密码。 输入步骤 3.3 中获取的“password”值。	

6. 将“properties.properties”文件上传到 Flume 客户端安装目录下的“flume/conf/”下。

步骤 4 分别生成 MonitorServer 角色服务端和客户端的证书和信任列表。

1. 使用 ECM 以 **omm** 用户登录 MonitorServer 角色所在主机。

   进入“${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/bin”目录。

   cd ${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/bin

2. 执行以下命令，生成并导出 MonitorServer 角色服务端、客户端证书。

   sh geneJKS.sh -m *xxx* **-n** ***xxx***

   生成的证书在“${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/conf”路径下，其中：

   - “ms_sChat.jks”是 MonitorServer 角色服务端的证书库，“ms_sChat.crt”是“ms_sChat.jks”证书的导出文件，“-m”配置项是证书和证书库的密码；
   - “ms_cChat.jks”是 MonitorServer 角色客户端的证书库，“ms_cChat.crt”是“ms_cChat.jks”证书的导出文件，“-n”配置项是证书和证书库的密码；
   - “ms_sChatt.jks”、“ms_cChatt.jks”分别为 MonitorServer 服务端、客户端 SSL 证书信任列表。

步骤 5 配置 MonitorServer 角色服务端参数。

1. 执行以下命令，生成并得到 MonitorServer 服务端密钥库密码、信任列表密码和 keystore-password 加密的私钥信息。连续输入两次密码并确认，该密码是别名为 *mschatserver* 的证书和 *ms_sChat.jks* 证书库的密码。

   ./genPwFile.sh

   cat password.property

2. 使用以下命令打开“${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/conf/service/application.properties”文件。根据表 7-39 中的说明，修改相关参数，并保存退出。

   vi ${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1**/install/FusionInsight-Flume-1.9.0/flume/conf/service/application.properties**

表7-39 MonitorServer 角色服务端所需修改的参数列表

参数名称	参数值填写规则	参数样例
ssl_need_kspassw	是否开启自定义密钥加解密功	true

参数名称	参数值填写规则	参数样例
d_decrypt_key	能（基于安全要求，建议启用此功能）。 • true 表示启用。 • false 表示不启用。	
ssl_server_enable	是否启用 SSL 认证（基于安全要求，建议用户启用此功能）。 • true 表示启用。 • false 表示不启用。	true
ssl_server_key_store	根据具体的存放位置进行修改。	${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/conf/ms_sChat.jks
ssl_server_trust_key_store	根据具体的存放位置进行修改。	${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/conf/ms_sChatt.jks
ssl_server_key_store_password	keystore 密码，根据具体制作证书的实际情况修改（生成证书的明文密钥） 输入步骤 5.1 中获取的“password”值。	-
ssl_server_trust_key_store_password	krustkeystore 密码，根据具体制作证书的实际情况修改（生成信任列表的明文密钥）。 输入步骤 5.1 中获取的“password”值。	-
ssl_need_client_auth	是否启用客户端认证（基于安全要求，建议用户启用此功能）。 • true 表示启用。 • false 表示不启用。	true

3. 重启 MonitorServer 实例。选择“ 服务 > Flume > 实例 > MonitorServer”，勾选配置的“MonitorServer”实例，选择“更多 > 重启实例”。输入管理员密码，单击“确定”，重启完成后单击“完成”完成操作。

步骤 6 配置 MonitorServer 角色客户端参数。

1. 执行以下命令将生成的客户端证书（ms_cChat.jks）和客户端信任列表（ms_cChatt.jks）复制到客户端的“/opt/flume-client/fusionInsight-flume-1.9.0/conf/”目录下，其中 **10.196.26.1** 为客户端所在节点业务平面的 IP 地址。

   scp ${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/conf/ms_cChat.jks user@10.196.26.1:/opt/flume-client/fusionInsight-flume-1.9.0/conf/

   scp ${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/conf/ms_cChatt.jks user@10.196.26.1:/opt/flume-client/fusionInsight-flume-1.9.0/conf/

2. 以 **user** 用户登录 Flume 客户端所在的节点。执行以下命令进入客户端目录“/opt/flume-client/fusionInsight-flume-1.9.0/bin”。

   cd /opt/flume-client/fusionInsight-flume-1.9.0/bin

3. 执行以下命令，生成并得到 MonitorServer 客户端密钥库密码、信任列表密码和 keystore-password 加密的私钥信息。连续输入两次密码并确认，该密码是别名为 *mschatclient* 的证书和 *ms_cChat.jks* 证书库的密码。

   ./genPwFile.sh

   cat password.property

4. 使用以下命令打开“/opt/flume-client/fusionInsight-flume-1.9.0/conf/service/application.properties”文件（“/opt/flume-client/fusionInsight-flume-1.9.0”为客户端软件安装后的目录）。根据表 7-40 中的说明，修改相关参数，并保存退出。

   vi /opt/flume-client/fusionInsight-flume-1.9.0/flume/conf/service/application.properties

表7-40 MonitorServer 角色客户端所需修改的参数列表

参数名称	参数值填写规则	参数样例
ssl_need_kspasswd_decrypt_key	是否开启自定义密钥加解密功能（基于安全要求，建议用户启用此功能）。 • true 表示启用。 • false 表示不启用。	true
ssl_client_enable	是否启用 SSL 认证（基于安全要求，建议用户启用此功能）。 • true 表示启用。 • false 表示不启用。	true
ssl_client_key_store	根据具体的存放位置进行修改。	${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/conf/ms_cChat.jks
ssl_client_trust_key_store	根据具体的存放位置进行修改。	${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-

参数名称	参数值填写规则	参数样例
		1.9.0/flume/conf/ms_cChatt.jks
ssl_client_key_store_password	keystore 密码，根据具体制作证书的实际情况修改（生成证书的明文密钥）。 输入步骤 6.3 中获取的“password”值。	-
ssl_client_trust_key_store_password	trustkeystore 密码，根据具体制作证书的实际情况修改（生成信任列表的明文密钥）。 输入步骤 6.3 中获取的“password”值。	-
ssl_need_client_auth	是否启用客户端认证（基于安全要求，建议用户启用此功能）。 • true 表示启用。 • false 表示不启用。	true

----结束

7.11.2 典型场景：从本地采集静态日志保存到 HDFS

操作场景

该任务指导用户使用 Flume 从本地采集静态日志保存到 HDFS 上如下目录“/flume/test”。

本章节适用于 MRS 3.x 及之后版本。

前提条件

- 已成功安装集群、HDFS 及 Flume 服务、Flume 客户端。
- 已创建用户 **flume_hdfs** 并授权验证日志时操作的 HDFS 目录和数据。

操作步骤

步骤 1 分别生成 Flume 角色服务端和客户端的证书和信任列表。

1. 以 **omm** 用户登录 Flume 服务端所在节点。进入“${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/bin”目录。

   cd ${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/bin

2. 执行以下命令，生成并导出 Flume 角色服务端、客户端证书。

   sh geneJKS.sh -f *密码* **-g** *密码*

   生成的证书在“${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/conf”路径下。其中：

   - “flume_sChat.jks”是 Flume 角色服务端的证书库，“flume_sChat.crt”是“flume_sChat.jks”证书的导出文件，“-f”配置项是证书和证书库的密码；
   - “flume_cChat.jks”是 Flume 角色客户端的证书库，“flume_cChat.crt”是“flume_cChat.jks”证书的导出文件，“-g”配置项是证书和证书库的密码；
   - “flume_sChatt.jks”和“flume_cChatt.jks”分别为 Flume 服务端、客户端 SSL 证书信任列表。

   说明

   本章节涉及到所有的用户自定义密码，需满足以下复杂度要求：

   - 至少包含大写字母、小写字母、数字、特殊符号 4 种类型字符
   - 至少 8 位，最多 64 位
   - 出于安全考虑，建议用户定期更换自定义密码（例如三个月更换一次），并重新生成各项证书和信任列表。

步骤 2 在 FusionInsight Manager 管理界面，选择“系统 > 权限 > 用户”，选择“更多 > 下载认证凭据”下载用户 **flume_hdfs** 的 kerberos 证书文件并保存在本地。

步骤 3 配置 Flume 角色的服务端参数，并将配置文件上传到集群。

1. 以 **omm** 用户登录任意一个 Flume 角色所在的节点。执行以下命令进入“${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/bin”。

   cd ${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/bin

2. 执行以下命令，生成并得到 Flume 服务端密钥库密码、信任列表密码和 keystore-password 加密的私钥信息。连续输入两次密码并确认，该密码是 *flume_sChat.jks* 证书库的密码。

   ./genPwFile.sh

   cat password.property

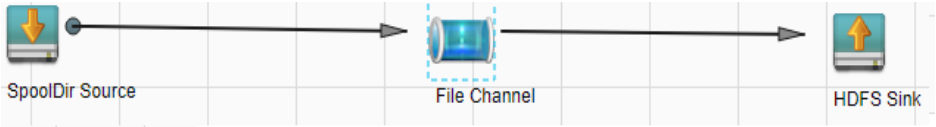
3. 使用 FusionInsight Manager 界面中的 Flume 配置工具来配置服务端参数并生成配置文件。

   a. 登录 FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > Flume > 配置工具”。

   b. “Agent 名”选择“server”，然后选择要使用的 source、channel 以及 sink，将其拖到右侧的操作界面中并将其连接。

      采用 SpoolDir Source、File Channel 和 HDFS Sink，如图 7-17 所示。

图7-17 Flume 配置工具示例



c. 双击对应的 source、channel 以及 sink，根据实际环境并参考表 7-41 设置对应的配置参数。

说明

- 如果对应的 Flume 角色之前已经配置过服务端参数，为保证与之前的配置保持一致，在 FusionInsight Manager 界面选择“集群 > *待操作集群的名称* > 服务 > Flume > 实例”，选择相应的 Flume 角色实例，单击“实例配置”页面“flume.config.file”参数后的“下载文件”按钮，可获取已有的服务端参数配置文件。然后选择“集群 > *待操作集群的名称* > 服务 > Flume > 配置工具 > 导入”，将该文件导入后再修改加密传输的相关配置项即可。
- 导入配置文件时，建议配置 source/channel/sink 的各自的个数都不要超过 40 个，否则可能导致界面响应时间过长。
- 不同的 File Channel 均需要配置一个不同的 checkpoint 目录。

d. 单击“导出”，将配置文件“properties.properties”保存到本地。

表7-41 Flume 角色服务端所需修改的参数列表

参数名称	参数值填写规则	参数样例
名称	不能为空，必须唯一。	test
bind	avro source 绑定的 ip 地址，此参数不能为空。须配置为服务端配置文件即将要上传的主机 IP。	192.168.108.11
port	avro source 监听的端口,此参数不能为空。须配置为未被使用的端口。	21154
ssl	是否启用 SSL 认证（基于安全要求，建议用户启用此功能）。 只有“Avro”类型的 Source 才有此配置项。 • true 表示启用。 • false 表示不启用。	true
keystore	服务端证书。	${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/conf/flume_sChat.jks
keystore-password	密钥库密码，获取 keystore 信息所需	-

参数名称	参数值填写规则	参数样例
	密码。 输入步骤 3.2 中获取的“password”值。	
truststore	服务端的 SSL 证书信任列表。	${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/conf/flume_sChatt.jks
truststore-password	信任列表密码，获取 truststore 信息所需密码。 输入步骤 3.2 中获取的“password”值。	-
dataDirs	缓冲区数据保存目录，默认为运行目录。配置多个盘上的目录可以提升传输效率，多个目录使用逗号分隔。如果为集群内，则可以指定在如下目录/srv/BigData/hadoop/dataX/flume/data，dataX 为 data1~dataN。如果为集群外，则需要单独规划。	/srv/BigData/hadoop/data1/flumeserver/data
checkpointDir	checkpoint 信息保存目录，默认在运行目录下。如果为集群内，则可以指定在如下目录/srv/BigData/hadoop/dataX/flume/checkpoint，dataX 为 data1~dataN。如果为集群外，则需要单独规划。	/srv/BigData/hadoop/data1/flumeserver/checkpoint
transactionCapacity	事务大小：即当前 channel 支持事务处理的事件个数。建议和 Source 的 batchSize 设置为同样大小，不能小于 batchSize。	61200
hdfs.path	写入 HDFS 的目录，此参数不能为空。	hdfs://hacluster/flume/test
hdfs.inUsePrefix	正在写入 HDFS 的文件的前缀。	TMP_
hdfs.batchSize	一次写入 HDFS 的最大事件数目。	61200
hdfs.kerberosPrincipal	kerberos 认证时用户，在安全版本下必须填写。安全集群需要配置此项，普通模式集群无需配置。	flume_hdfs
hdfs.kerberosKeytab	kerberos 认证时 keytab 文件路径，在安全版本下必须填写。安全集群需要配置此项，普通模式集群无需配置。	/opt/test/conf/user.keytab 说明 user.keytab 文件从下载用户 **flume_hdfs** 的 kerberos

参数名称	参数值填写规则	参数样例
		证书文件中获取，另外，确保用于安装和运行 Flume 客户端的用户对 user.keytab 文件有读写权限。
hdfs.useLocalTimeStamp	是否使用本地时间，取值为"true"或者"false"。	true

4. 登录 FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > Flume”，在“角色”下单击“Flume”角色。
5. 选择准备上传配置文件的节点行的“Flume”角色，单击“实例配置”页面“flume.config.file”参数后的“上传文件”，选择“properties.properties”文件完成操作。

说明

- 每个 Flume 实例均可以上传单独的服务端配置文件。
- 更新配置文件需要按照此步骤操作，后台修改配置文件是不规范操作，同步配置时后台做的修改将会被覆盖。

6. 单击“保存”，单击“确定”。
7. 单击“完成”完成操作。

步骤 4 配置 Flume 角色客户端参数。

1. 执行以下命令将生成的客户端证书（flume_cChat.jks）和客户端信任列表（flume_cChatt.jks）复制到客户端目录下，如“/opt/flume-client/fusionInsight-flume-1.9.0/conf/”（要求已安装 Flume 客户端），其中 **10.196.26.1** 为客户端所在节点业务平面的 IP 地址。

   scp ${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/conf/flume_cChat.jks user@10.196.26.1:/opt/flume-client/fusionInsight-flume-1.9.0/conf/

   scp ${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/conf/flume_cChatt.jks user@10.196.26.1:/opt/flume-client/fusionInsight-flume-1.9.0/conf/

   说明

   复制过程中需要输入客户端所在主机（如 **10.196.26.1**）**user** 用户的密码。

2. 以 **user** 用户登录解压 Flume 客户端的节点。执行以下命令进入客户端目录“/opt/flume-client/fusionInsight-flume-1.9.0/bin”。

   cd opt/flume-client/fusionInsight-flume-1.9.0/bin

3. 执行以下命令，生成并得到 Flume 客户端密钥库密码、信任列表密码和 keystore-password 加密的私钥信息。连续输入两次密码并确认，该密码是别名为 *flumechatclient* 的证书和 *flume_cChat.jks* 证书库的密码。

   ./genPwFile.sh

cat password.property

说明

若产生以下错误提示，可执行命令 **export JAVA_HOME**=*JDK 路径*进行处理。

```
JAVA_HOME is null in current user,please install the JDK and set the JAVA_HOME
```

4. 执行 **echo $SCC_PROFILE_DIR** 检查 SCC_PROFILE_DIR 环境变量是否为空。
   - 是，执行 **source .sccfile**。
   - 否，执行步骤 4.5。
5. 使用 FusionInsight Manager 界面中的 Flume 配置工具来配置 Flume 角色客户端参数并生成配置文件。
   a. 登录 FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > Flume > 配置工具”。
   b. “Agent 名”选择“client”，然后选择要使用的 source、channel 以及 sink，将其拖到右侧的操作界面中并将其连接。

      采用 SpoolDir Source、File Channel 和 HDFS Sink，如图 7-18 所示。

      图7-18 Flume 配置工具示例

   c. 双击对应的 source、channel 以及 sink，根据实际环境并参考表 7-42 设置对应的配置参数。

说明

- 如果对应的 Flume 角色之前已经配置过客户端参数，为保证与之前的配置保持一致，可以到“*客户端安装目录*/fusioninsight-flume-1.9.0/conf/properties.properties”获取已有的客户端参数配置文件。然后登录 FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > Flume > 配置工具 > 导入”，将该文件导入后再修改加密传输的相关配置项即可。
- 导入配置文件时，建议配置中 source/channel/sink 的各自的个数都不要超过 40 个，否则可能导致界面响应时间过长。

   d. 单击“导出”，将配置文件“properties.properties”保存到本地。

表7-42 Flume 角色客户端所需修改的参数列表

参数名称	参数值填写规则	参数样例
名称	不能为空，必须唯一。	test
spoolDir	待采集的文件所在的目录路径，此参数不能为空。该路径需存在，且对 flume 运行用户有读写执行权限。	/srv/BigData/hadoop/data1/zb

参数名称	参数值填写规则	参数样例
trackerDir	flume 采集文件信息元数据保存路径。	/srv/BigData/hadoop/data1/tracker
batch-size	Flume 一次发送数据的最大事件数。	61200
dataDirs	缓冲区数据保存目录，默认为运行目录。配置多个盘上的目录可以提升传输效率，多个目录使用逗号分隔。如果为集群内，则可以指定在如下目录/srv/BigData/hadoop/dataX/flume/data，dataX 为 data1~dataN。如果为集群外，则需要单独规划。	/srv/BigData/hadoop/data1/flume/data
checkpointDir	checkpoint 信息保存目录，默认在运行目录下。如果为集群内，则可以指定在如下目录/srv/BigData/hadoop/dataX/flume/checkpoint，dataX 为 data1~dataN。如果为集群外，则需要单独规划。	/srv/BigData/hadoop/data1/flume/checkpoint
transactionCapacity	事务大小：即当前 channel 支持事务处理的事件个数，建议和 Source 的 batchSize 设置为同样大小，不能小于 batchSize。	61200
hostname	要发送数据的主机名或者 IP，此参数不能为空。须配置为与之相连的 avro source 所在的主机名或 IP。	192.168.108.11
port	avro sink 监听的端口,此参数不能为空。须配置为与之相连的 avro source 监听的端口。	21154
ssl	是否启用 SSL 认证（基于安全要求，建议用户启用此功能）。 只有“Avro”类型的 Source 才有此配置项。 • true 表示启用。	true

参数名称	参数值填写规则	参数样例
	• false 表示不启用。	
keystore	服务端生成的 flume_cChat.jks 证书。	/opt/flume-client/fusionInsight-flume-1.9.0/conf/flume_cChat.jks
keystore-password	密钥库密码，获取 keystore 信息所需密码。 输入步骤 4.3 中获取的“password”值。	-
truststore	服务端的 SSL 证书信任列表。	/opt/flume-client/fusionInsight-flume-1.9.0/conf/flume_cChatt.jks
truststore-password	信任列表密码，获取 truststore 信息所需密码。 输入步骤 4.3 中获取的“password”值。	-

6. 将“properties.properties”文件上传到 Flume 客户端安装目录下的“flume/conf/”下。

步骤 5 分别生成 MonitorServer 角色服务端和客户端的证书和信任列表。

1. 以 **omm** 用户登录 MonitorServer 角色所在主机。

   进入“${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/bin”目录。

   cd ${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/bin

2. 执行以下命令，生成并导出 MonitorServer 角色服务端、客户端证书。

   sh geneJKS.sh -m *密码* **-n** *密码*

   生成的证书在“${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/conf”路径下，其中：

   - “ms_sChat.jks”是 MonitorServer 角色服务端的证书库，“ms_sChat.crt”是“ms_sChat.jks”证书的导出文件，“-m”配置项是证书和证书库的密码；
   - “ms_cChat.jks”是 MonitorServer 角色客户端的证书库，“ms_cChat.crt”是“ms_cChat.jks”证书的导出文件，“-n”配置项是证书和证书库的密码；
   - “ms_sChatt.jks”、“ms_cChatt.jks”分别为 MonitorServer 服务端、客户端 SSL 证书信任列表。

步骤 6 配置 MonitorServer 角色服务端参数。

1. 执行以下命令，生成并得到 MonitorServer 服务端密钥库密码、信任列表密码和 keystore-password 加密的私钥信息。连续输入两次密码并确认，该密码是别名为 *mschatserver* 的证书和 *ms_sChat.jks* 证书库的密码。

./genPwFile.sh

cat password.property

2. 使用以下命令打开“${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/conf/service/application.properties”文件。根据表 7-43 中的说明，修改相关参数，并保存退出。

   vi ${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1**/install/FusionInsight-Flume-1.9.0/flume/conf/service/application.properties**

表7-43 MonitorServer 角色服务端所需修改的参数列表

参数名称	参数值填写规则	参数样例
ssl_need_kspasswd_decrypt_key	是否开启自定义密钥加解密功能（基于安全要求，建议用户启用此功能）。 • true 表示启用。 • false 表示不启用。	true
ssl_server_enable	是否启用 SSL 认证（基于安全要求，建议用户启用此功能）。 • true 表示启用。 • false 表示不启用。	true
ssl_server_key_store	根据具体的存放位置进行修改。	${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/conf/ms_sChat.jks
ssl_server_trust_key_store	根据具体的存放位置进行修改。	${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/conf/ms_sChatt.jks
ssl_server_key_store_password	keystore 密码，根据具体制作证书的实际情况修改（生成证书的明文密钥）。 输入步骤 6.1 中获取的“password”值。	-
ssl_server_trust_key_store_password	krustkeystore 密码，根据具体制作证书的实际情况修改（生成信任列表的明文密钥）。 输入步骤 6.1 中获取的“password”值。	-
ssl_need_client_auth	是否启用客户端认证（基于安全要求，建议用户启用此功能）。	true

参数名称	参数值填写规则	参数样例
	• true 表示启用。 • false 表示不启用。	

3. 重启 MonitorServer 实例。选择“集群 > *待操作集群的名称* > 服务 > Flume > 实例 > MonitorServer”，勾选配置的“MonitorServer”实例，选择“更多 > 重启实例”。输入管理员密码，单击“确定”，重启完成后单击“完成”完成操作。

步骤 7 配置 MonitorServer 角色客户端参数。

1. 执行以下命令将生成的客户端证书（ms_cChat.jks）和客户端信任列表（ms_cChatt.jks）复制到客户端的“/opt/flume-client/fusionInsight-flume-1.9.0/conf/”目录下，其中 **10.196.26.1** 为客户端所在节点业务平面的 IP 地址。

   scp ${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/conf/ms_cChat.jks user@10.196.26.1:/opt/flume-client/fusionInsight-flume-1.9.0/conf/

   scp ${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/conf/ms_cChatt.jks user@10.196.26.1:/opt/flume-client/fusionInsight-flume-1.9.0/conf/

2. 以 **user** 用户登录 Flume 客户端所在的节点。执行以下命令进入客户端目录“/opt/flume-client/fusionInsight-flume-1.9.0/bin”。

   cd /opt/flume-client/fusionInsight-flume-1.9.0/bin

3. 执行以下命令，生成并得到 MonitorServer 客户端密钥库密码、信任列表密码和 keystore-password 加密的私钥信息。连续输入两次密码并确认，该密码是别名为 *mschatclient* 的证书和 *ms_cChat.jks* 证书库的密码。

   ./genPwFile.sh

   cat password.property

4. 使用以下命令打开“/opt/flume-client/fusionInsight-flume-1.9.0/conf/service/application.properties”文件（“/opt/flume-client/fusionInsight-flume-1.9.0”为客户端安装后的目录）。根据表 7-44 中的说明，修改相关参数，并保存退出。

   vi /opt/flume-client/fusionInsight-flume-1.9.0/conf/service/application.properties

表7-44 MonitorServer 角色客户端所需修改的参数列表

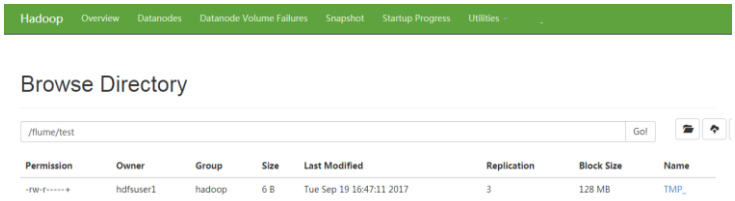
参数名称	参数值填写规则	参数样例
ssl_need_kspassword_decrypt_key	是否开启自定义密钥加解密功能（基于安全要求，建议用户启用此功能）。 • true 表示启用。 • false 表示不启用。	true
ssl_client_enable	是否启用 SSL 认证（基于安全要求，建议用户启用此功能）。	true

参数名称	参数值填写规则	参数样例
	• true 表示启用。 • false 表示不启用。	
ssl_client_key_store	根据具体的存放位置进行修改。	${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/conf/ms_cChat.jks
ssl_client_trust_key_store	根据具体的存放位置进行修改。	${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Flume-1.9.0/flume/conf/ms_cChatt.jks
ssl_client_key_store_password	keystore 密码，根据具体制作证书的实际情况修改（生成证书的明文密钥）。 输入步骤 7.3 中获取的“password”值。	-
ssl_client_trust_key_store_password	trustkeystore 密码，根据具体制作证书的实际情况修改（生成信任列表的明文密钥）。 输入步骤 7.3 中获取的“password”值。	-
ssl_need_client_auth	是否启用客户端认证（基于安全要求，建议用户启用此功能）。 • true 表示启用。 • false 表示不启用。	true

步骤 8 验证日志是否传输成功。

1. 以具有 HDFS 组件管理权限的用户登录 FusionInsight Manager，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本）。在 FusionInsight Manager 界面选择“集群 > *待操作集群的名称* > 服务 > HDFS”，单击“NameNode(*节点名称*，主)”对应的链接，打开 HDFS WebUI，然后选择“Utilities > Browse the file system”
2. 观察 HDFS 上“/flume/test”目录下是否有产生数据。

图7-19 查看 HDFS 目录和文件



----结束

7.12 查看 Flume 客户端监控信息

操作场景

集群外的 Flume 客户端也是端到端数据采集的一环，与集群内 Flume 服务端一起都需要监控，用户通过 FusionInsight Manager 可以对 Flume 客户端进行监控，可以查看客户端的 Source、Sink、Channel 的监控指标以及客户端的进程状态。

本章节适用于 MRS 3.x 及之后版本。

操作步骤

步骤 1 登录 FusionInsight Manager。

步骤 2 选择“集群 > *待操作集群的名称* > 服务 > Flume > Flume 管理”，即可查看当前 Flume 客户端列表及进程状态。

步骤 3 选择“实例 ID”，进入客户端监控列表，在“实时”区域框中，可查看客户端的各监控指标。

步骤 4 选择“历史”进入历史监控数据查询界面。筛选时间段，单击“查看”可显示该时间段内的监控数据。

----结束

7.13 Flume 对接安全 Kafka 指导

操作场景

使用 Flume 客户端对接安全 kafka。

本章节适用于 MRS 3.x 及之后版本。

操作步骤

步骤 1 新增 jaas.conf 文件，并保存到“${Flume 客户端安装目录}/conf”下，jaas.conf 文件内容如下：

```
KafkaClient {
com.sun.security.auth.module.Krb5LoginModule required
useKeyTab=true
keyTab="/opt/test/conf/user.keytab"
principal="flume_hdfs@<系统域名>"
useTicketCache=false
storeKey=true
debug=true;
 };
```

其中 **keyTab** 和 **principal** 的值请按照实际情况配置，所配置的 **principal** 需要有相应的 kafka 的权限。

步骤 2 配置业务，其中 **kafka.bootstrap.servers** 的端口号使用 21007，**kafka.security.protocol** 使用 **SASL_PLAINTEXT**。

步骤 3 如果 Kafka 所在集群的域名发生了更改，需要对${Flume 客户端安装目录}/conf/flume-env.sh 文件中的-**Dkerberos.domain.name** 项的值做修改，具体请根据实际域名进行配置。

步骤 4 上传所配置的 **properties.properties** 文件到${Flume 客户端安装目录}/conf 目录下。

----结束

7.14 Flume 对接安全 Hive 指导

操作场景

使用 Flume 对接集群中的 Hive（3.1.0 版本）。

本章节适用于 MRS 3.x 及之后版本。

前置条件

集群正确安装了 Flume 服务和 Hive 服务，且服务正常无告警异常。

操作步骤

步骤 1 使用 **omm** 用户将如下 jar 包导入到需要测试的 Flume 实例的 lib 目录下（客户端/服务端），列表如下：

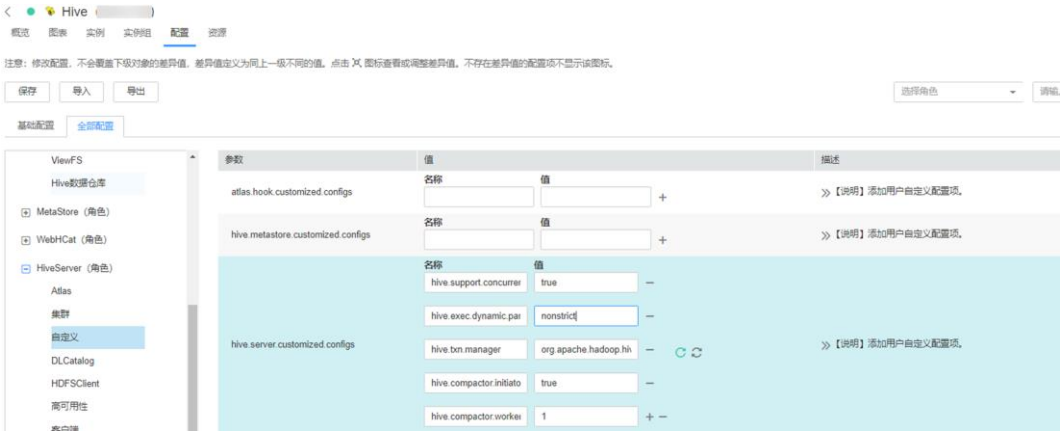
- antlr-2.7.7.jar
- antlr-runtime-3.4.jar
- calcite-core-1.16.0.jar
- hadoop-mapreduce-client-core-3.1.1.jar
- hive-beeline-3.1.0.jar

- hive-cli-3.1.0.jar
- hive-common-3.1.0.jar
- hive-exec-3.1.0.jar
- hive-hcatalog-core-3.1.0.jar
- hive-hcatalog-pig-adapter-3.1.0.jar
- hive-hcatalog-server-extensions-3.1.0.jar
- hive-hcatalog-streaming-3.1.0.jar
- hive-metastore-3.1.0.jar
- hive-service-3.1.0.jar
- libfb303-0.9.3.jar
- hadoop-plugins-1.0.jar

相关 jar 包可从 Hive 安装目录中获取，重启对应的 Flume 进程，保证 jar 包加载到运行环境中。

步骤 2 配置 Hive 配置项。

在 FusionInsight Manager 界面，选择“集群 > 服务 > Hive > 配置 > 全部配置 > HiveServer > 自定义 > hive.server.customized.configs”，具体配置如图所示：



配置项如下：

名称	值
hive.support.concurrency	true
hive.exec.dynamic.partition.mode	nonstrict
hive.txn.manager	org.apache.hadoop.hive.ql.lockmgr.DbTxnManager
hive.compactor.initiator.on	true
hive.compactor.worker.threads	1

步骤 3 准备具备 supergroup 和 Hive 权限的系统用户 flume_hive，安装客户端并创建所需的 Hive 表。

示例如下：

1. 正确安装集群客户端，例如安装目录为“/opt/client”。
2. 执行以下命令完成用户认证。

   cd /opt/client

   source bigdata_env

   kinit flume_hive
3. 执行 **beeline** 命令，然后执行以下建表语句。

```
create table flume_multi_type_part(id string, msg string)
partitioned by (country string, year_month string, day string)
clustered by (id) into 5 buckets
stored as orc TBLPROPERTIES('transactional'='true');
```

4. 执行 **select * from** *表名*命令**;**，查询表中数据。

   此时表中数据量为 0 行。

步骤 4 准备相关配置文件，假设下载的客户端安装包在“/opt/FusionInsight_Cluster_1_Services_ClientConfig”。

1. 从“${*客户端解压目录*}/Hive/config”目录获取以下文件：
   - hivemetastore-site.xml
   - hive-site.xml
2. 从“${*客户端解压目录*}/HDFS/config”目录下获取以下文件：

   core-site.xml
3. 在 Flume 实例启动的机器上创建目录，将准备好的上述文件放置在创建的目录下。

   例如：“/opt/hivesink-conf/hive-site.xml”。
4. 将“hivemetastore-site.xml”文件中的所有 property 配置，拷贝至“hive-site.xml”，并保证处于原有配置之前。

   因为 hive 内部加载有顺序。

说明

保证配置文件所在的目录对于 Flume 运行用户 **omm** 有读写权限。

步骤 5 结果观察。

在 Hive 的客户端执行，**select * from** *表名***;**查看对应的数据是否已经写入到 Hive 表中。

----结束

参考实例

Flume 配置参考示例（SpoolDir--Mem--Hive）：

```
server.sources = spool_source
server.channels = mem_channel
server.sinks = Hive_Sink
```

```
#config the source
server.sources.spool_source.type = spooldir
server.sources.spool_source.spoolDir = /tmp/testflume
server.sources.spool_source.montime =
server.sources.spool_source.fileSuffix =.COMPLETED
server.sources.spool_source.deletePolicy = never
server.sources.spool_source.trackerDir =.flumespool
server.sources.spool_source.ignorePattern = ^$
server.sources.spool_source.batchSize = 20
server.sources.spool_source.inputCharset =UTF-8
server.sources.spool_source.selector.type = replicating
server.sources.spool_source.fileHeader = false
server.sources.spool_source.fileHeaderKey = file
server.sources.spool_source.basenameHeaderKey= basename
server.sources.spool_source.deserializer = LINE
server.sources.spool_source.deserializer.maxBatchLine= 1
server.sources.spool_source.deserializer.maxLineLength= 2048
server.sources.spool_source.channels = mem_channel

#config the channel
server.channels.mem_channel.type = memory
server.channels.mem_channel.capacity =10000
server.channels.mem_channel.transactionCapacity= 2000
server.channels.mem_channel.channelfullcount= 10
server.channels.mem_channel.keep-alive = 3
server.channels.mem_channel.byteCapacity =
server.channels.mem_channel.byteCapacityBufferPercentage= 20

#config the sink
server.sinks.Hive_Sink.type = hive
server.sinks.Hive_Sink.channel = mem_channel
server.sinks.Hive_Sink.hive.metastore = thrift://${任意metastore业务IP}:21088
server.sinks.Hive_Sink.hive.hiveSite = /opt/hivesink-conf/hive-site.xml
server.sinks.Hive_Sink.hive.coreSite = /opt/hivesink-conf/core-site.xml
server.sinks.Hive_Sink.hive.metastoreSite = /opt/hivesink-conf/hivemeatastore-
site.xml
server.sinks.Hive_Sink.hive.database = default
server.sinks.Hive_Sink.hive.table = flume_multi_type_part
server.sinks.Hive_Sink.hive.partition = Tag,%Y-%m,%d
server.sinks.Hive_Sink.hive.txnsPerBatchAsk= 100
server.sinks.Hive_Sink.hive.autoCreatePartitions= true
server.sinks.Hive_Sink.useLocalTimeStamp = true
server.sinks.Hive_Sink.batchSize = 1000
server.sinks.Hive_Sink.hive.kerberosPrincipal= super1
server.sinks.Hive_Sink.hive.kerberosKeytab= /opt/mykeytab/user.keytab
server.sinks.Hive_Sink.round = true
server.sinks.Hive_Sink.roundValue = 10
server.sinks.Hive_Sink.roundUnit = minute
server.sinks.Hive_Sink.serializer = DELIMITED
server.sinks.Hive_Sink.serializer.delimiter= ";"
server.sinks.Hive_Sink.serializer.serdeSeparator= ';'
server.sinks.Hive_Sink.serializer.fieldnames= id,msg
```

7.15 Flume 业务模型配置指导

7.15.1 概述

本章节适用于 MRS 3.x 及之后版本。

本任务旨在提供 Flume 常用模块的性能差异，用于指导用户进行合理的 Flume 业务配置，避免出现前端 Source 和后端 Sink 性能不匹配进而导致整体业务性能不达标的场景。

本任务只针对于单通道的场景进行比较说明。

7.15.2 业务模型配置指导

本章节适用于 MRS 3.x 及之后版本。

Flume 业务配置及模块选择过程中，一般要求 Sink 的极限吞吐量需要大于 Source 的极限吞吐量，否则在极限负载的场景下，Source 往 Channel 的写入速度大于 Sink 从 Channel 取出的速度，从而导致 Channel 频繁被写满，进而影响性能表现。

Avro Source 和 Avro Sink 一般都是成对出现，用于多个 Flume Agent 间进行数据中转，因此一般场景下 Avro Source 和 Avro Sink 都不会成为性能瓶颈。

模块间性能

根据模块间极限性能对比，可以看到对于前端是 SpoolDir Source 的场景下，Kafka Sink 和 HDFS Sink 都能满足吞吐量要求，但是 HBase Sink 由于自身写入性能较低的原因，会成为性能瓶颈，会导致数据都积压在 Channel 中。但是如果有必须使用 HBase Sink 或者其他性能容易成为瓶颈的 Sink 的场景时，可以选择使用 **Channel Selector** 或者 **Sink Group** 来满足性能要求。

Channel Selector

Channel Selector 可以允许一个 Source 对接多个 Channel，通过选择不同的 Selector 类型来将 Source 的数据进行分流或者复制，目前 Flume 提供的 Channel Selector 有两种：Replicating 和 Multiplexing。

Replicating：表示 Source 的数据同步发送给所有 Channel。

Multiplexing：表示根据 Event 中的 Header 的指定字段的值来进行判断，从而选择相应的 Channel 进行发送，从而起到根据业务类型进行分流的目的。

- Replicating 配置样例：

```
client.sources = kafkasource
client.channels = channel1 channel2
client.sources.kafkasource.type = org.apache.flume.source.kafka.KafkaSource
client.sources.kafkasource.kafka.topics = topic1,topic2
client.sources.kafkasource.kafka.consumer.group.id = flume
client.sources.kafkasource.kafka.bootstrap.servers = 10.69.112.108:21007
client.sources.kafkasource.kafka.security.protocol = SASL_PLAINTEXT
client.sources.kafkasource.batchDurationMillis = 1000
client.sources.kafkasource.batchSize = 800
```

```
client.sources.kafkasource.channels = channel1 c el2

client.sources.kafkasource.selector.type = replicating
client.sources.kafkasource.selector.optional = channel2
```

表7-45 Replicating 配置样例参数说明

选项名称	默认值	描述
Selector.type	replicating	Selector 类型，应配置为 replicating
Selector.optional	-	可选 Channel，可以配置为列表

- Multiplexing 配置样例：

```
client.sources = kafkasource
client.channels = channel1 channel2
client.sources.kafkasource.type = org.apache.flume.source.kafka.KafkaSource
client.sources.kafkasource.kafka.topics = topic1,topic2
client.sources.kafkasource.kafka.consumer.group.id = flume
client.sources.kafkasource.kafka.bootstrap.servers = 10.69.112.108:21007
client.sources.kafkasource.kafka.security.protocol = SASL_PLAINTEXT
client.sources.kafkasource.batchDurationMillis = 1000
client.sources.kafkasource.batchSize = 800
client.sources.kafkasource.channels = channel1 channel2

client.sources.kafkasource.selector.type = multiplexing
client.sources.kafkasource.selector.header = myheader
client.sources.kafkasource.selector.mapping.topic1 = channel1
client.sources.kafkasource.selector.mapping.topic2 = channel2
client.sources.kafkasource.selector.default = channel1
```

表7-46 Multiplexing 配置样例参数说明

选项名称	默认值	描述
Selector.type	replicating	Selector 类型，应配置为 multiplexing
Selector.header	Flume.selector.header	-
Selector.default	-	-
Selector.mapping.*	-	-

Multiplexing 类型的 Selector 的样例中，选择 Event 中 Header 名称为 topic 的字段来进行判断，当 Header 中 topic 字段的值为 topic1 时，向 channel1 发送该 Event，当 Header 中 topic 字段的值为 topic2 时，向 channel2 发送该 Event。

这种 Selector 需要借助 Source 中 Event 的特定 Header 来进行 Channel 的选择，需要根据业务场景选择合理的 Header 来进行数据分流。

SinkGroup

当后端单 Sink 性能不足、需要高可靠性保证或者异构输出时可以使用 Sink Group 来将指定的 Channel 和多个 Sink 对接，从而满足相应的使用场景。目前 Flume 提供了两种 Sink Processor 用于对 Sink Group 中的 Sink 进行管理：Load Balancing 和 Failover。

Failover：表示在 Sink Group 中同一时间只有一个 Sink 处于活跃状态，其他 Sink 作为备份处于非活跃状态，当活跃状态的 Sink 故障时，根据优先级从非活跃状态的 Sink 中选择一个来接管业务，保证数据不会丢失，多用于高可靠性场景。

Load Balancing：表示在 Sink Group 中所有 Sink 都处于活跃状态，每个 Sink 都会从 Channel 中去获取数据并进行处理，并且保证在运行过程中该 Sink Group 的所有 Sink 的负载是均衡的，多用于性能提升场景。

- Load Balancing 配置样例：

```
client.sources = source1
client.sinks = sink1 sink2
client.channels = channel1

client.sinkgroups = g1
client.sinkgroups.g1.sinks = sink1 sink2
client.sinkgroups.g1.processor.type = load_balance
client.sinkgroups.g1.processor.backoff = true
client.sinkgroups.g1.processor.selector = random

client.sinks.sink1.type = logger
client.sinks.sink1.channel = channel1

client.sinks.sink2.type = logger
client.sinks.sink2.channel = channel1
```

表7-47 Load Balancing 配置样例参数说明

选项名称	默认值	描述
sinks	-	Sink Group 的 sink 列表，多个以空格分隔
processor.type	default	Processor 的类型，应配置为 load_balance
processor.backoff	false	是否以指数的形式退避失败的 Sinks
processor.selector	round_robin	选择机制。必须是 round_robin,random 或者自定义的类，且该类继承了 AbstractSinkSelector
processor.selector.maxTimeOut	30000	屏蔽故障 sink 的时间，默认是 30000 毫秒

- Failover 配置样例：

```
client.sources = source1
client.sinks = sink1 sink2
client.channels = channel1

client.sinkgroups = g1
client.sinkgroups.g1.sinks = sink1 sink2
client.sinkgroups.g1.processor.type = failover
client.sinkgroups.g1.processor.priority.sink1 = 10
client.sinkgroups.g1.processor.priority.sink2 = 5
client.sinkgroups.g1.processor.maxpenalty = 10000

client.sinks.sink1.type = logger
client.sinks.sink1.channel = channel1

client.sinks.sink2.type = logger
client.sinks.sink2.channel = channel1
```

表7-48 **Failover** 配置样例参数说明

选项名称	默认值	描述
sinks	-	Sink Group 的 sink 列表，多个以空格分隔
processor.type	default	Processor 的类型，应配置为 failover
processor.priority.<sinkName>	-	优先级值。<sinkName> 必须是 sinks 中有定义的。优先级值高 Sink 会更早被激活。值越大，优先级越高。**注**：多个 sinks 的话，优先级的值不要相同，如果优先级相同的话，只会有一个生效。
processor.maxpenalty	30000	失败的 Sink 最大的退避时间(单位：毫秒)

Interceptors

Flume 的拦截器（Interceptor）支持在数据传输过程中修改或丢弃传输的基本单元 Event。用户可以通过在配置中指定 Flume 内建拦截器的类名列表，也可以开发自定义的拦截器来实现 Event 的修改或丢弃。Flume 内建支持的拦截器如下表所示，本章节会选取一个较为复杂的作为示例。其余的用户可以根据需要自行配置使用。官网参考：

http://flume.apache.org/releases/content/1.9.0/FlumeUserGuide.html

说明

1. 拦截器用在 Flume 的 Source、Channel 之间，大部分的 Source 都带有 Interceptor 参数。用户可以依据需要配置。
2. Flume 支持一个 Source 配置多个拦截器，各拦截器名称用空格分开。
3. 指定拦截器的顺序就是它们被调用的顺序。
4. 使用拦截器在 Header 中插入的内容，都可以在 Sink 中读取并使用。

表7-49 Flume 内建支持的拦截器类型

拦截器类型	简要描述
Timestamp Interceptor	该拦截器会在 Event 的 Header 中插入一个时间戳。
Host Interceptor	该拦截器会在 Event 的 Header 中插入当前 Agent 所在节点的 IP 或主机名。
Remove Header Interceptor	该拦截器会依据 Header 中包含的符合正则匹配的字符串，丢弃掉对应的 Event。
UUID Interceptor	该拦截器会为每个 Event 的 Header 生成一个 UUID 字符串。
Search and Replace Interceptor	该拦截器基于 Java 正则表达式提供简单的基于字符串的搜索和替换功能。与 Java Matcher.replaceAll() 的规则相同。
Regex Filtering Interceptor	该拦截器通过将 Event 的 Body 体解释为文本文件，与配置的正则表达式进行匹配来选择性的过滤 Event。提供的正则表达式可用于排除或包含事件。
Regex Extractor Interceptor	该拦截器使用正则表达式抽取原始 events 中的内容，并将该内容加入 events 的 header 中。

下面以 Regex Filtering Interceptor 为例说明 Interceptor 使用(其余的可参考官网配置)：

表7-50 Regex Filtering Interceptor 配置参数说明

选项名称	默认值	描述
type	-	组件类型名称，必须写为 regex_filter。
regex	-	用于匹配事件的正则表达式。
excludeEvents	false	默认收集匹配到的 Event。设置为 true，则会删除匹配的 Event，保留不匹配的。

配置示例(为了方便观察，此模型使用了 netcat tcp 作为 Source 源，logger 作为 Sink)。配置好如下参数后，在 Linux 的配置的主机节点上执行 Linux 命令“telnet *主机名或 IP* 44444”，并任意敲入符合正则和不符合正则的字符串。会在日志中观察到，只有匹配到的字符串被传输了。

```
#define the source、channel、sink
server.sources = r1

server.channels = c1
server.sinks = k1

#config the source
server.sources.r1.type = netcat
server.sources.r1.bind = ${主机 IP}
server.sources.r1.port = 44444
server.sources.r1.interceptors= i1
server.sources.r1.interceptors.i1.type= regex_filter
server.sources.r1.interceptors.i1.regex= (flume)|(myflume)
server.sources.r1.interceptors.i1.excludeEvents= false
server.sources.r1.channels = c1

#config the channel
server.channels.c1.type = memory
server.channels.c1.capacity = 1000
server.channels.c1.transactionCapacity = 100
#config the sink
server.sinks.k1.type = logger
server.sinks.k1.channel = c1
```

7.16 Flume 日志介绍

日志描述

日志路径：Flume 相关日志的默认存储路径为“/var/log/Bigdata/角色名”。

- FlumeServer：“/var/log/Bigdata/flume/flume”
- FlumeClient：“/var/log/Bigdata/flume-client-n/flume”
- MonitorServer：“/var/log/Bigdata/flume/monitor”

日志归档规则：Flume 日志启动了自动压缩归档功能，缺省情况下，当日志大小超过 50MB 的时候，会自动压缩，压缩后的日志文件名规则为：“<原有日志名>-<yyyy-mm-dd_hh-mm-ss>.[编号].log.zip”。最多保留最近的 20 个压缩文件，压缩文件保留个数可以在 Manager 界面中配置。

表7-51 Flume 日志列表

日志类型	日志文件名	描述
运行日志	/flume/flumeServer.log	FlumeServer 运行环境信息日志。

日志类型	日志文件名	描述
	/flume/install.log	FlumeServer 安装日志。
	/flume/flumeServer-gc.log.<编号>	FlumeServer 进程的 GC 日志。
	/flume/prestartDvietail.log	Flume 启动前的工作日志。
	/flume/startDetail.log	Flume 进程启动工作日志。
	/flume/stopDetail.log	Flume 进程停止日志。
	/monitor/monitorServer.log	MonitorServer 运行环境信息日志。
	/monitor/startDetail.log	MonitorServer 进程启动工作日志。
	/monitor/stopDetail.log	MonitorServer 进程停止日志。
	function.log	外部函数调用日志。

日志级别

Flume 提供了如表 7-52 所示的日志级别。

运行日志的级别优先级从高到低分别是 FATAL、ERROR、WARN、INFO、DEBUG，程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。

表7-52 日志级别

日志类型	级别	描述
运行日志	FATAL	FATAL 表示系统运行的致命错误信息。
	ERROR	ERROR 表示系统运行的错误信息。
	WARN	WARN 表示当前事件处理存在异常信息。
	INFO	INFO 表示记录系统及各事件正常运行状态信息。
	DEBUG	DEBUG 表示记录系统及系统的调试信息。

如果您需要修改日志级别，请执行如下操作：

步骤 1 请参考修改集群服务配置参数，进入 Flume 的“全部配置”页面。

步骤 2 左边菜单栏中选择所需修改的角色所对应的日志菜单。

步骤 3 选择所需修改的日志级别。

步骤 4 保存配置，在弹出窗口中单击“确定”使配置生效。

----结束

说明

配置完成后即生效，不需要重启服务。

日志格式

Flume 的日志格式如下所示：

表7-53 日志格式

日志类型	格式	示例
运行日志	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level><产生该日志的线程名字>|<log 中的 message>|<日志事件的发生位置>	2014-12-12 11:54:57,316	INFO	[main]	log4j dynamic load is start.	org.apache.flume.tools.LogDynamicLoad.start(LogDynamicLoad.java:59)
	<yyyy-MM-dd HH:mm:ss,SSS><User Name><User IP><Time><Operation><Resource><Result><Detail>	2014-12-12 23:04:16,572	INFO	[SinkRunner-PollingRunner-DefaultSinkProcessor]	SRCIP=null OPERATION=close

7.17 Flume 客户端 Cgroup 使用指导

操作场景

该操作指导用户加入、退出 Cgroup，查询 Cgroup 状态以及更改 Cgroup cpu 阈值。

本章节适用于 MRS 3.x 及之后版本。

操作步骤

- **加入 Cgroup**

执行以下命令，加入 Cgroup，假设 Flume 客户端安装路径为“/opt/FlumeClient”，Cgroup cpu 阈值设置为 50%：

cd /opt/FlumeClient/fusioninsight-flume-1.9.0/bin

./flume-manage.sh cgroup join 50

说明

- 该命令不仅可以加入 Cgroup，同时也可以更改 Cgroup cpu 阈值。
- Cgroup cpu 阈值取值范围为 1~100*N 之间的整数，N 表示机器 cpu 核数。

- **查询 Cgroup 状态**

  执行以下命令，查询 Cgroup 状态，假设 Flume 客户端安装路径为“/opt/FlumeClient”：

  cd /opt/FlumeClient/fusioninsight-flume-1.9.0/bin

  ./flume-manage.sh cgroup status

- **退出 Cgroup**

  执行以下命令，退出 Cgroup，假设 Flume 客户端安装路径为“/opt/FlumeClient”：

  cd /opt/FlumeClient/fusioninsight-flume-1.9.0/bin

  ./flume-manage.sh cgroup exit

  说明

  - 客户端安装完成后，会自动创建默认 Cgroup。若安装客户端时未配置“-s”参数，则默认值为“-1”，表示 agent 进程不受 cpu 使用率限制。
  - 加入、退出 Cgroup 时，agent 进程不受影响。若 agent 进程未启动，加入、退出 Cgroup 仍然可以成功执行，待下一次 agent 启动时生效。
  - 客户端卸载完成后，安装时期创建的 Cgroup 会自动删除。

7.18 Flume 第三方插件二次开发指导

操作场景

该操作指导用户进行第三方插件二次开发。

本章节适用于 MRS 3.x 及之后版本。

前提条件

- 第三方 jar 包。
- 已成功安装 Flume 服务端或者客户端。

操作步骤

步骤 1 将自主研发的代码打成 jar 包。

步骤 2 建立插件目录布局。

1. 进入$FLUME_HOME/plugins.d 路径下，使用以下命令建立目录：

   mkdir thirdPlugin

   cd thirdPlugin

   mkdir lib libext native

   显示结果如下：

   ```
   [root@187-7-44-162 plugins.d]#mkdir thirdPlugin
   [root@187-7-44-162 plugins.d]#ll
   total 8
   drwxr-x--- 3 root root 4096 Aug 26 18:54 native
   drwxr-xr-x 2 root root 4096 Sep 17 18:44 thirdPlugin
   [root@187-7-44-162 plugins.d]#cd thirdPlugin/
   [root@187-7-44-162 thirdPlugin]#mkdir lib libext native
   [root@187-7-44-162 thirdPlugin]#ll
   total 12
   drwxr-xr-x 2 root root 4096 Sep 17 18:45 lib
   drwxr-xr-x 2 root root 4096 Sep 17 18:45 libext
   drwxr-xr-x 2 root root 4096 Sep 17 18:45 native
   [root@187-7-44-162 thirdPlugin]#
   ```

2. 将第三方 jar 包放入**$FLUME_HOME/plugins.d/thirdPlugin/lib** 路径下，若该 jar 包依赖其他 jar 包，则将所依赖的 jar 包放入 **$FLUME_HOME/plugins.d/thirdPlugin/libext** 文件夹中，**$FLUME_HOME/plugins.d/thirdPlugin/native** 放置本地库文件。

步骤 3 配置**$FLUME_HOME/conf/properties.properties** 文件。

具体 **properties.properties** 参数配置方法，参考非加密传输和加密传输对应典型场景中 **properties.properties** 文件参数列表的说明。

说明

- **$FLUME_HOME** 表示 Flume 安装路径，配置第三方插件时，根据实际情况（服务端/客户端）指定。
- **thirdPlugin** 根据实际业务进行命名，无固定名称。

----结束

7.19 配置 Flume 定制脚本

操作场景

Flume 支持定制脚本，支持在传输前或者传输后执行指定的脚本，用于执行准备工作。

本章节适用于 MRS 3.x 及之后版本。

操作步骤

- **场景一：未安装 Flume 客户端**

步骤 1 获取软件包。

登录 FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > Flume”进入 Flume 服务界面，在右上角选择“更多 > 下载客户端”，选择“选择客户端类型”为“完整客户端”，下载 Flume 服务客户端文件。

客户端文件名称为“FusionInsight_Cluster_<*集群 ID*>_Flume_Client.tar”，本章节以“FusionInsight_Cluster_1_Flume_Client.tar”为例进行描述。

步骤 2 上传软件包。以 **user** 用户将软件包上传到将要安装 Flume 服务客户端的节点目录上，例如“/opt/client”

说明

user 用户为安装和运行 Flume 客户端的用户。

步骤 3 解压软件包。

以 **user** 用户登录将要安装 Flume 服务客户端的节点。进入安装包所在目录，例如“/opt/client”，执行如下命令解压安装包到当前目录。

cd /opt/client

tar -xvf FusionInsight_Cluster_1_Flume_Client.tar

步骤 4 校验软件包。

执行 **sha256sum -c** 命令校验解压得到的文件，返回“OK”表示校验通过。例如：

sha256sum -c FusionInsight_Cluster_1_Flume_ClientConfig.tar.sha256

```
FusionInsight_Cluster_1_Flume_ClientConfig.tar: OK
```

步骤 5 解压文件。

tar -xvf FusionInsight_Cluster_1_Flume_ClientConfig.tar

步骤 6 在客户端 /opt/client/FusionInsight_Cluster_1_Flume_ClientConfig/Flume/FlumeClient/flume/conf/flume-check.properties 文件中配置 client.per-check.shell，指向 plugin.sh 的绝对路径。

配置如下：

client.per-check.shell=/opt/client/FusionInsight_Cluster_1_Flume_ClientConfig/Flume/FlumeClient/flume/plugins.s/plugin.sh

plugins = com.xxx.flume.services.FlumePreTransmitService

flume.check.default.interval = 15

步骤 7 配置 /opt/client/FusionInsight_Cluster_1_Flume_ClientConfig/Flume/FlumeClient/flume/conf/plugin.conf 文件，定义具体调用的脚本、相关参数。

配置如下：

RUN_PLUGIN="PLUGIN_LIST_1"

LOG_TO_HDFS_PATH="/yxs"

LOG_TO_HDFS_ENCODE_PATH="${LOG_TO_HDFS_PATH}/Flume_Encoded/"

PLUGIN_LINK_DIR="/tmp/yxs1"

PLUGIN_MV_TARGET_DIR="/tmp/yxs2"

PLUGIN_SUFFIX="COMPLETED"

PLUGIN_LIST_1="mv_complete.sh --linkdir ${PLUGIN_LINK_DIR} --mvtargetdir ${PLUGIN_MV_TARGET_DIR} --suffix ${PLUGIN_SUFFIX}"

步骤 8 安装并启动 Flume 客户端。安装客户端详细操作请参考安装 Flume 客户端。

----结束

- **场景二：已安装 Flume 客户端**

步骤 1 在客户端 flume-check.properties 文件中配置 client.per-check.shell，指向 plugin.sh 的绝对路径。

例如 Flume 客户端安装路径为“/opt/FlumeClient”，则 flume-check.properties 文件所在目录为/opt/FlumeClient/fusioninsight-flume-1.9.0/conf，

配置如下：

client.per-check.shell=/opt/FlumeClient/fusioninsight-flume-1.9.0/plugins.s/plugin.sh

plugins = com.xxx.flume.services.FlumePreTransmitService

flume.check.default.interval = 15

步骤 2 配置 plugin.conf，定义具体调用的脚本、相关参数。

例如 Flume 客户端安装路径为“/opt/FlumeClient”，则 plugin.conf 配置文件所在目录为/opt/FlumeClient/fusioninsight-flume-1.9.0/conf，

配置如下：

RUN_PLUGIN="PLUGIN_LIST_1"

LOG_TO_HDFS_PATH="/yxs"

LOG_TO_HDFS_ENCODE_PATH="${LOG_TO_HDFS_PATH}/Flume_Encoded/"

PLUGIN_LINK_DIR="/tmp/yxs1"

PLUGIN_MV_TARGET_DIR="/tmp/yxs2"

PLUGIN_SUFFIX="COMPLETED"

PLUGIN_LIST_1="mv_complete.sh --linkdir ${PLUGIN_LINK_DIR} --mvtargetdir ${PLUGIN_MV_TARGET_DIR} --suffix ${PLUGIN_SUFFIX}"

步骤 3 在客户端安装路径 bin 目录执行以下命令，重启 Flume 客户端，例如“/opt/FlumeClient/fusioninsight-flume-1.9.0/bin”。

./flume-manage.sh restart

----结束

7.20 Flume 常见问题

Flume 日志保存在/var/log/Bigdata/flume/flume/flumeServer.log 里。绝大多数数据传输异常、数据传输不成功，在日志里都可以看到提示。可以直接输入以下命令查看：

tailf /var/log/Bigdata/flume/flume/flumeServer.log

- 问题：当配置文件上传后，发现异常，重新上传配置文件，发现仍然没有满足场景要求，但日志上没有任何异常。

  解决方法：重启此 flume 进程，**kill -9** *进程代码*，再看日志。

- 问题：连接 HDFS 出现 java.lang.IllegalArgumentException: Keytab is not a readable file: /opt/test/conf/user.keytab。

  解决方法：添加 Flume 运行用户读写权限。

- 问题：执行 Flume 客户端连接 Kafka 报如下错误：

  ```
  Caused by: java.io.IOException: /opt/FlumeClient/fusioninsight-flume-
  1.9.0/cof//jaas.conf (No such file or directory)
  ```

  解决方法：新增 jaas.conf 配置文件并保存到 flume client 的 conf 路径下。

  vi jaas.conf

  ```
  KafkaClient {
  com.sun.security.auth.module.Krb5LoginModule required
  useKeyTab=true
  keyTab="/opt/test/conf/user.keytab"
  principal="flume_hdfs@<系统域名>"
  useTicketCache=false
  storeKey=true
  debug=true;
  };
  ```

  参数 keyTab 和 principal 根据实际情况修改。

- 问题：执行 Flume 客户端连接 HBase 报如下错误：

  ```
  Caused by: java.io.IOException: /opt/FlumeClient/fusioninsight-flume-
  1.9.0/cof//jaas.conf (No such file or directory)
  ```

  解决方法：新增 jaas.conf 配置文件并保存到 flume client 的 conf 路径下。

  vi jaas.conf

  ```
  Client {
  com.sun.security.auth.module.Krb5LoginModule required
  useKeyTab=true
  keyTab="/opt/test/conf/user.keytab"
  principal="flume_hbase@<系统域名>"
  useTicketCache=false
  storeKey=true
  debug=true;
  };
  ```

  参数 keyTab 和 principal 根据实际情况修改。

- 问题：一旦提交配置文件后，flume agent 即在占用资源运行，如何恢复到没有上传配置文件的状态？

  解决方法：提交一个内容为空的 properties.properties 文件。

8 使用 HBase

8.1 从零开始使用 HBase

HBase 是一个高可靠性、高性能、面向列、可伸缩的分布式存储系统。本章节提供从零开始使用 HBase 的操作指导，在集群 Master 节点中更新客户端，通过客户端实现创建表，往表中插入数据，修改表，读取表数据，删除表中数据以及删除表的功能。

背景信息

假定用户开发一个应用程序，用于管理企业中的使用 A 业务的用户信息，使用 HBase 客户端实现 A 业务操作流程如下：

- 创建用户信息表 user_info。
- 在用户信息中新增用户的学历、职称信息。
- 根据用户编号查询用户姓名和地址。
- 根据用户姓名进行查询。
- 用户销户，删除用户信息表中该用户的数据。
- A 业务结束后，删除用户信息表。

表8-1 用户信息

编号	姓名	性别	年龄	地址
12005000201	A	男	19	A 城市
12005000202	B	女	23	B 城市
12005000203	C	男	26	C 城市
12005000204	D	男	18	D 城市
12005000205	E	女	21	E 城市
12005000206	F	男	32	F 城市
12005000207	G	女	29	G 城市

编号	姓名	性别	年龄	地址
12005000208	H	女	30	H 城市
12005000209	I	男	26	I 城市
12005000210	J	男	25	J 城市

前提条件

已安装客户端，例如安装目录为“/opt/client”。以下操作的客户端目录只是举例，请根据实际安装目录修改。在使用客户端前，需要先下载并更新客户端配置文件，确认Manager 的主管理节点后才能使用客户端。

操作步骤

MRS 3.x 以前版本集群执行以下操作：

步骤 1 下载客户端配置文件。

1. 登录 MRS Manager 页面，具体请参见访问集群 Manager，然后选择“服务管理”。
2. 单击“下载客户端”。

   “客户端类型”选择“仅配置文件”，“下载路径”选择“服务器端”，单击“确定”开始生成客户端配置文件，文件生成后默认保存在主管理节点“/tmp/MRS-client”。文件保存路径支持自定义。

步骤 2 登录 MRS Manager 的主管理节点。

1. 在集群详情的“节点信息”页签中查看节点名称，名称中包含“master1”的节点为 Master1 节点，名称中包含“master2”的节点为 Master2 节点。

   MRS Manager 的主备管理节点默认安装在集群 Master 节点上。在主备模式下，由于 Master1 和 Master2 之间会切换，Master1 节点不一定是 MRS Manager 的主管理节点，需要在 Master1 节点中执行命令，确认 MRS Manager 的主管理节点。命令请参考步骤 2.4。
2. 以 **root** 用户使用密码方式登录 Master1 节点。操作方法，请参见“用户指南 > 连接集群 > 登录集群 > 登录集群节点”章节。
3. 切换至 **omm** 用户。

   sudo su - root

   su - omm
4. 执行以下命令确认 MRS Manager 的主管理节点。

   sh ${BIGDATA_HOME}/om-0.0.1/sbin/status-oms.sh

   回显信息中“HAActive”参数值为“active”的节点为主管理节点（如下例中“mgtomsdat-sh-3-01-1”为主管理节点），参数值为“standby”的节点为备管理节点（如下例中“mgtomsdat-sh-3-01-2”为备管理节点）。

```
Ha mode
double
```

```
NodeName            HostName                    HAVersion         StartTime
HAActive            HAAllResOK          HARunPhase
192-168-0-30         mgtomsdat-sh-3-01-1          V100R001C01       2019-11-18
23:43:02     active             normal              Actived
192-168-0-24         mgtomsdat-sh-3-01-2          V100R001C01       2019-11-21
07:14:02     standby            normal              Deactived
```

5. 使用 **root** 用户登录 MRS Manager 的主管理节点，例如“192-168-0-30”节点，并执行以下命令切换到 **omm** 用户。

   sudo su - omm

步骤 3 执行以下命令切换到客户端安装目录，例如“/opt/client”。

cd /opt/client

步骤 4 执行以下命令，更新主管理节点的客户端配置。

sh refreshConfig.sh /opt/client *客户端配置文件压缩包完整路径*

例如，执行命令：

sh refreshConfig.sh /opt/client /tmp/MRS-client/MRS_Services_Client.tar

界面显示以下信息表示配置刷新更新成功：

```
ReFresh components client config is complete.
Succeed to refresh components client config.
```

说明

步骤步骤 1~步骤 4 的操作也可以参考“用户指南 > Manager 操作指导 > 集群管理 > 配置客户端 > 更新已安装客户端的配置”页面的方法二操作。

步骤 5 在 Master 节点使用客户端。

1. 在已更新客户端的主管理节点，例如“192-168-0-30”节点，执行以下命令切换到客户端目录。

   cd /opt/client

2. 执行以下命令配置环境变量。

   source bigdata_env

3. 如果当前集群已启用 Kerberos 认证，执行以下命令认证当前用户，当前用户需要具有创建 HBase 表的权限，具体请参见“用户指南 > 管理现有集群 > MRS 多用户权限管理 > 创建角色”配置拥有对应权限的角色，参考“用户指南 > 管理现有集群 > MRS 多用户权限管理 > 创建用户”为用户绑定对应角色。如果当前集群未启用 Kerberos 认证，则无需执行此命令。

   kinit MRS 集群用户

   例如，**kinit hbaseuser**。

4. 直接执行 HBase 组件的客户端命令。

   hbase shell

步骤 6 运行 HBase 客户端命令，实现 A 业务。

1. 根据表 8-1 创建用户信息表 user_info 并添加相关数据。

   create '*user_info*',{**NAME** => 'i'}

以增加编号 12005000201 的用户信息为例，其他用户信息参照如下命令依次添加：

put '*user_info*','*12005000201*','***i:name***','*A*'

put '*user_info*','*12005000201*','***i:gender***','*Male*'

put '*user_info*','*12005000201*','***i:age***','*19*'

put '*user_info*','*12005000201*','***i:address***','*City A*'

2. 在用户信息表 user_info 中新增用户的学历、职称信息。

   以增加编号为 12005000201 的用户的学历、职称信息为例，其他用户类似。

   put '*user_info*','*12005000201*','**i:degree**','*master*'

   put '*user_info*','*12005000201*','**i:pose**','*manager*'

3. 根据用户编号查询用户姓名和地址。

   以查询编号为 12005000201 的用户姓名和地址为例，其他用户类似。

   scan'*user_info*',{**STARTROW**=>'*12005000201*',**STOPROW**=>'*12005000201*',**COLUMNS**=>['**i:name**','**i:address**']}

4. 根据用户姓名进行查询。

   以查询 A 用户信息为例，其他用户类似。

   scan'*user_info*',{**FILTER**=>"SingleColumnValueFilter('i','name',=,'binary:*A*')"}

5. 删除用户信息表中该用户的数据。

   所有用户的数据都需要删除，以删除编号为 12005000201 的用户数据为例，其他用户类似。

   delete'*user_info*','*12005000201*','i'

6. 删除用户信息表。

   disable'*user_info*'

   drop '*user_info*'

----结束

MRS 3.x 及之后版本集群执行以下操作：

步骤 1 在主管理节点使用客户端。

1. 以客户端安装用户登录客户端安装节点，执行以下命令切换到客户端目录。

   cd /opt/client

2. 执行以下命令配置环境变量。

   source bigdata_env

3. 如果当前集群已启用 Kerberos 认证，执行以下命令认证当前用户，当前用户需要具有创建 HBase 表的权限，具体请参见“Manager 操作指导 > 系统设置 > 权限设置 > 角色管理”配置拥有对应权限的角色，参考“Manager 操作指导 > 系统设置 > 权限设置 > 用户管理 > 创建用户”为用户绑定对应角色。如果当前集群未启用 Kerberos 认证，则无需执行此命令。

   kinit *MRS 集群用户*

   例如，**kinit hbaseuser**。

4. 直接执行 HBase 组件的客户端命令。

hbase shell

步骤 2 运行 HBase 客户端命令，实现 A 业务。

1. 根据表 8-1 创建用户信息表 user_info 并添加相关数据。

   create '*user_info*',{**NAME** => 'i'}

   以增加编号 12005000201 的用户信息为例，其他用户信息参照如下命令依次添加：

   put '*user_info*','*12005000201*','***i:name***','*A*'

   put '*user_info*','*12005000201*','***i:gender***','*Male*'

   put '*user_info*','*12005000201*','***i:age***','*19*'

   put '*user_info*','*12005000201*','***i:address***','*City A*'

2. 在用户信息表 user_info 中新增用户的学历、职称信息。

   以增加编号为 12005000201 的用户的学历、职称信息为例，其他用户类似。

   put '*user_info*','*12005000201*','**i:degree**','*master*'

   put '*user_info*','*12005000201*','**i:pose**','*manager*'

3. 根据用户编号查询用户姓名和地址。

   以查询编号为 12005000201 的用户姓名和地址为例，其他用户类似。

   scan'*user_info*',{**STARTROW**=>'*12005000201*',**STOPROW**=>'*12005000201*',**COLUMNS**=>[**'i:name'**,**'i:address'**]}

4. 根据用户姓名进行查询。

   以查询 A 用户信息为例，其他用户类似。

   scan'*user_info*',{**FILTER**=>"SingleColumnValueFilter('i','name',=,'binary:*A*')"}

5. 删除用户信息表中该用户的数据。

   所有用户的数据都需要删除，以删除编号为 12005000201 的用户数据为例，其他用户类似。

   delete'*user_info*','*12005000201*','i'

6. 删除用户信息表。

   disable'*user_info*'

   drop '*user_info*'

----结束

8.2 使用 HBase 客户端

操作场景

该任务指导用户在运维场景或业务场景中使用 HBase 客户端。

前提条件

- 已安装客户端。例如安装目录为“/opt/hadoopclient”，以下操作的客户端目录只是举例，请根据实际安装目录修改。

- 各组件业务用户由系统管理员根据业务需要创建。

  “机机”用户需要下载 keytab 文件，“人机”用户第一次登录时需修改密码。
- 非 **root** 用户使用 HBase 客户端，请确保该 HBase 客户端目录的属主为该用户，否则请参考如下命令修改属主。

  chown user:group -R *客户端安装目录***/HBase**

使用 Hbase 客户端（MRS 3.x 之前版本）

步骤 1 以客户端安装用户，登录安装客户端的节点。

步骤 2 执行以下命令切换到客户端目录。

cd /opt/hadoopclient

步骤 3 执行以下命令配置环境变量。

source bigdata_env

步骤 4 如果当前集群已启用 Kerberos 认证，执行以下命令认证当前用户，当前用户需要具有创建 HBase 表的权限，具体请参见“用户指南 > 管理现有集群 > MRS 多用户权限管理 > 创建角色”配置拥有对应权限的角色，参考“用户指南 > 管理现有集群 > MRS 多用户权限管理 > 创建用户”章节，为用户绑定对应角色。如果当前集群未启用 Kerberos 认证，则无需执行此命令。

kinit *组件业务用户*

例如，**kinit hbaseuser**。

步骤 5 直接执行 HBase 组件的客户端命令。

hbase shell

----结束

使用 HBase 客户端（MRS 3.x 及之后版本）

步骤 1 以客户端安装用户，登录安装客户端的节点。

步骤 2 执行以下命令切换到客户端目录。

cd /opt/hadoopclient

步骤 3 执行以下命令配置环境变量。

source bigdata_env

步骤 4 若安装了 HBase 多实例，在使用客户端连接具体 HBase 实例时，请执行以下命令加载具体实例的环境变量，否则请跳过此步骤。例如，加载 HBase2 实例变量：

source HBase2/component_env

步骤 5 如果当前集群已启用 Kerberos 认证，执行以下命令认证当前用户，当前用户需要具有创建 HBase 表的权限，具体请参见“用户指南 > 管理现有集群 > MRS 多用户权限管理 > 创建角色”配置拥有对应权限的角色，参考“用户指南 > 管理现有集群 > MRS

多用户权限管理 > 创建用户”章节，为用户绑定对应角色。如果当前集群未启用Kerberos 认证，则无需执行此命令。

kinit *组件业务用户*

例如，**kinit hbaseuser**。

步骤 6 直接执行 HBase 组件的客户端命令。

hbase shell

----结束

HBase 客户端常用命令

常用的 HBase 客户端命令如下表所示。更多命令可参考 http://hbase.apache.org/2.2/book.html

表8-2 HBase 客户端命令

命令	说明
create	创建一张表，例如 **create 'test', 'f1', 'f2', 'f3'**。
disable	停止指定的表，例如 **disable 'test'**。
enable	启动指定的表，例如 **enable 'test'**。
alter	更改表结构。可以通过 alter 命令增加、修改、删除列族信息以及表相关的参数值，例如 **alter 'test', {NAME => 'f3', METHOD => 'delete'}**。
describe	获取表的描述信息，例如 **describe 'test'**。
drop	删除指定表。删除前表必须已经是停止状态，例如 **drop 'test'**。
put	写入指定 cell 的 value。Cell 的定位由表、rowk、列组合起来唯一决定，例如 **put 'test','r1','f1:c1','myvalue1'**。
get	获取行的值或者行的指定 cell 的值。例如 **get 'test','r1'**。
scan	查询表数据。参数中指定表名和 scanner，例如 **scan 'test'**。

8.3 创建 HBase 角色

操作场景

该任务指导系统管理员在 Manager 创建并设置 HBase 的角色。HBase 角色可设置 HBase 管理员权限以及 HBase 表和列族的读（R）、写（W）、创建（C）、执行（X）或管理（A）权限。

用户需要在 HBase 中对指定的数据库或表设置权限，才能够创建表、查询数据、删除数据、插入数据、更新数据以及授权他人访问 HBase 表。

说明

- 本章节适用于 MRS 3.x 及之后版本。
- 安全模式支持创建 HBase 角色，普通模式不支持创建 HBase 角色。
- 如果当前组件使用了 Ranger 进行权限控制，须基于 Ranger 配置相关策略进行权限管理，具体操作可参考添加 HBase 的 Ranger 访问权限策略。

前提条件

- 系统管理员已明确业务需求。
- 已登录 Manager。

操作步骤

步骤 1 在 Manager 界面，选择“系统 > 权限 > 角色”。

步骤 2 单击“添加角色”，然后在“角色名称”和“描述”输入角色名字与描述。

步骤 3 设置角色“配置资源权限”请参见表 8-3。

HBase 权限：

- HBase Scope：对 HBase 表授权，最小支持设置列的读（R）和写（W）权限。
- HBase 管理员权限：HBase 管理员权限。

说明

用户对自己创建的表具有读（R）、写（W）、创建（C）、执行（X）或管理（A）权限。

表8-3 设置角色

任务场景	角色授权操作
设置 HBase 管理员权限	在“配置资源权限”的表格中选择“*待操作集群的名称* > HBase”，勾选“HBase 管理员权限”。
设置用户创建表的权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > HBase > HBase Scope”。 2. 单击“global”。 3. 在指定命名空间的“权限”列，勾选“创建”和“执行”。例如勾选默认命名空间“default”的“创建”和“执行”。
设置用户写入数据的权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > HBase > HBase Scope > global”。 2. 在指定命名空间的“权限”列，勾选“写”。例如勾选默认命名空间“default”的“写”。HBase 子对象默认可从父对象继承权限，此时已授予向命名空间中的表写入数据的权限。

任务场景	角色授权操作
设置用户读取数据的权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > HBase > HBase Scope > global”。 2. 在指定命名空间的“权限”列，勾选“读”。例如勾选默认命名空间“default”的“读”。HBase 子对象默认可从父对象继承权限，此时已授予从命名空间中的表读取数据的权限。
设置用户管理命名空间或表的权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > HBase > HBase Scope > global”。 2. 在指定命名空间的“权限”列，勾选“管理”。例如勾选默认命名空间“default”的“管理”。
设置列的读取或写入权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > HBase > HBase Scope > global”，单击指定命名空间显示命名空间的表。 2. 单击指定的表。 3. 单击指定的列族。 4. 确认是否是新建角色？ • 是，在“资源名称”的输入框输入列名称，多个列用英文逗号分隔，勾选“读”或“写”。如果 HBase 表中不存在同名的列，则创建同名的列后角色将拥有该列的权限。列权限设置完成。 • 否，修改已有 HBase 角色的列权限，表格将显示已单独设置权限的列，执行步骤 3.5。 5. 角色新增列权限，在“资源名称”的输入框输入列名称并设置列的权限。角色修改列权限，可以在“资源名称”的输入框输入列名称并设置列权限，也可以在表格中直接修改列的权限。若在表格中修改了列权限，又同时增加了同名的列权限，则无法保存。角色修改列权限，建议直接修改列的权限。支持搜索功能。

步骤 4 单击“确定”完成，返回“角色”。

----结束

8.4 配置 HBase 备份

操作场景

HBase 集群备份作为提高 HBase 集群系统高可用性的一个关键特性，为 HBase 提供了实时的异地数据备份功能。它对外提供了基础的运维工具，包含主备关系维护、重建，数据校验，数据同步进展查看等功能。为了实现数据的实时备份，可以把本 HBase 集群中的数据备份到另一个集群。

前提条件

- 主备集群都已经安装并启动成功（在 Console 页面“现有集群”页签，查看集群状态为“运行中”），且获取集群的管理员权限。
- 必须保证主备集群间的网络畅通和端口的使用。
- 主备集群必须已配置跨集群互信，请参见“用户指南 > 管理现有集群 > MRS 多用户权限管理 > 配置跨集群互信”。
- 如果主集群上有历史数据，需要同步到备集群上，那么主备集群必须配置跨集群拷贝，请参见启用集群间拷贝功能。
- 主备集群上的时间必须一致，而且主备集群上的 NTP 服务必须使用同一个时间源。
- 必须在主备集群中的“/etc/hosts”文件中，配置**主备集群所有机器**的机器名与业务 IP 地址的对应关系。配置方式为在 hosts 文件中追加"192.***.***.*** host1"。
- 主备集群间的网络带宽需要根据业务流量而定，不应少于最大的可能业务流量。

使用约束

- 尽管备份提供了实时的数据复制功能，但实际的数据同步进展，由多方面的因素决定的，例如，当前主集群业务的繁忙程度，备集群进程的健康状态等。因此，在正常情形下，备集群不应该接管业务。极端情形下是否可以接管业务，可由系统维护人员以及决策人员根据当前的数据同步指标来决定。
- 备份功能当前仅支持一主一备。
- 通常情况下，不允许对备集群的同步表进行表级别的操作，例如修改表属性、删除表等，一旦误操作备集群后会造成主集群数据同步失败、备集群对应表的数据丢失。
- 主集群的 HBase 表已启用备份功能同步数据，用户每次修改表的结构时，需要手动修改备集群的同步表结构，保持与主集群表结构一致。

操作步骤

启用主集群的备份功能来同步 put 方式写入的数据

步骤 1 登录 MRS 控制台，单击集群名称，选择“组件管理”。

步骤 2 进入 HBase 服务参数“全部配置”界面，具体操作请参考修改集群服务配置参数。

说明

若集群详情页面没有“组件管理”页签，请先完成 IAM 用户同步（在集群详情页的“概览”页签，单击“IAM 用户同步”右侧的“同步”进行 IAM 用户同步）。

步骤 3 （可选）如表 8-4 所示，为 HBase 备份操作过程中的可选配置项，您可以根据描述来进行参数配置，或者使用缺省提供的值。

表8-4 可选配置项

配置入口	配置项	默认值	描述
“HMaster >	hbase.master.logcleane	600000	HLog 文件生存时间。如果配置

配置入口	配置项	默认值	描述
性能”	r.ttl		值为“604800000”（单位：毫秒），表示 HLog 的保存期限为 7 天。
	hbase.master.cleaner.interval	60000	HMaster 清理过去 HLog 文件的周期，即超过设置的时间的 HLog 会被自动删除。建议尽可能配置大的值来保留更多的 HLog。
“RegionServer > Replication”	replication.source.size.capacity	16777216	edits 最大大小。单位为 byte。如果 edit 大小超过这个值 Hlog edits 将会发送到备集群。
	replication.source.nb.capacity	25000	edits 最大数目，是另一个触发 Hlog edits 到备集群的条件。当主集群同步数据到备集群中时，主集群会从 HLog 中读取数据，此时会根据本参数配置的个数读取并发送。与“replication.source.size.capacity”一起配置使用。
	replication.source.maxretriesmultiplier	10	replication 出现异常时的最大重试次数。
	replication.source.sleepforretries	1000	每次重试的 sleep 时间。（单位：毫秒）
	hbase.regionserver.replication.handler.count	6	RegionServer 上的 replication RPC 服务器实例数。

启用主集群备份功能来同步 Bulkload 方式写入的数据

步骤 4 是否启用 Bulkload 写数据备份功能？

说明

当使用了 HBase 的 Bulkload 导入数据的特性时且需要同步这些数据时，需要开启批量写数据备份功能。

是，执行步骤 5。

否，执行步骤 9。

步骤 5 参考修改集群服务配置参数进入 HBase 服务参数“全部配置”界面。

步骤 6 在主、备集群的 HBase 配置界面，搜索并修改“hbase.replication.cluster.id”参数，表示主、备集群 HBase 的 id，例如主集群 HBase 的 id 配置为“replication1”，备集群

HBase 的 id 配置为“replication2”，用于主备集群的连接。为了节省数据开销建议参数值长度不超过 30。

步骤 7 在备集群的 HBase 配置界面，搜索并修改“hbase.replication.conf.dir”参数，表示备集群中所使用主集群客户端的 HBase 配置，用于启用 bulkload 数据备份功能时的数据备份。参数值为路径名，例如“/home”。

说明

- MRS 3.x 之前的版本无需配置此参数，可跳过步骤步骤 7。
- 当启用 bulkload 数据备份功能时，需在备集群的所有 RegionServer 节点上手动放置主集群中 HBase 相应客户端配置文件(core-site.xml, hdfs-site.xml, hbase-site.xml)，放置配置文件的实际路径为“${hbase.replication.conf.dir}/${hbase.replication.cluster.id}”。例如备集群的 hbase.replication.conf.dir 配置为“/home”，主集群的 hbase.replication.cluster.id 配置为“replication1”，则配置文件放置在备集群中实际的路径为“/home/replication1”。并修改对应目录及文件相应权限，可执行如下命令 **chown -R omm:wheel /home/replication1**。
- 客户端配置文件可从主集群中的的客户端中获取，例如，路径为“/opt/client/HBase/hbase/conf”。更新配置文件的方法请参见“连接集群 > 使用 MRS 客户端 > 更新客户端”。

步骤 8 在主集群的 HBase 配置界面，搜索并修改“hbase.replication.bulkload.enabled”参数，将配置项的值修改为“true”，启用 Bulkload 写数据备份功能。

重启 HBase 服务并安装客户端。

步骤 9 保存配置，并重启 HBase 服务。

步骤 10 在主备集群，请参见“Manager 操作指导 > 集群管理 > 配置客户端 > 更新已安装客户端的配置”，更新客户端配置文件。

同步主集群表数据。（主集群无数据可不执行）

步骤 11 以“hbase”用户进入集群的 HBase shell 界面。

1. 在已更新客户端的主管理节点，执行以下命令切换到客户端目录。

   cd /opt/client

2. 执行以下命令配置环境变量。

   source bigdata_env

3. 如果当前集群已启用 Kerberos 认证，执行以下命令认证当前用户。如果当前集群未启用 Kerberos 认证，则无需执行此命令。

   kinit hbase

   说明

   执行 **kinit hbase** 后系统提示输入密码，hbase 用户默认密码是 Hbase@123。

4. 直接执行 HBase 组件的客户端命令。

   hbase shell

步骤 12 检查备集群上是否已有历史数据。如果有历史数据且需要保持主备集群上的数据完全一致，需要先清理备集群上的数据。

1. 在备集群的 hbase shell 界面中，执行 **list** 命令查看备集群中已经存在的表。
2. 根据输出列表删除备集群上的数据表。

   disable '*tableName*'

   drop '*tableName*'

步骤 13 检查配置 HBase 备份并启用数据同步后，主集群是否已存在表及数据，且历史数据需要同步到备集群。

执行 **list** 命令查看主集群中已经存在的表，使用 **scan** '*tableName*'命令查看表中是否已经有历史数据。

- 是，存在表且需要同步数据，执行步骤 14。
- 否，不需要同步数据，任务结束。

步骤 14 配置 HBase 备份时不支持自动同步表中的历史数据，需要对主集群的历史数据进行备份，然后再手动同步历史数据到备集群中。

手动同步即单表的同步，单表手动同步通过 Export、distcp、Import 来完成。

单表手动同步操作步骤：

1. 从主集群导出表中数据。

   hbase org.apache.hadoop.hbase.mapreduce.Export -Dhbase.mapreduce.include.deleted.rows=true *表名 保存源数据的目录*

   例如，**hbase org.apache.hadoop.hbase.mapreduce.Export -Dhbase.mapreduce.include.deleted.rows=true t1 /user/hbase/t1**
2. 把导出的数据复制到备集群。

   hadoop distcp *主集群保存源数据的目录 hdfs://ActiveNameNodeIP:9820/备集群保存源数据的目录*

   其中，ActiveNameNodeIP 是备集群中主 NameNode 节点的 IP 地址。

   例如，**hadoop distcp /user/hbase/t1 hdfs://192.168.40.2:9820/user/hbase/t1**
3. 使用备集群 HBase 表用户，在备集群中导入数据。

   hbase org.apache.hadoop.hbase.mapreduce.Import *-Dimport.bulk.output=备集群保存输出的目录 表名 备集群保存源数据的目录*

   hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles *备集群保存输出的目录 表名*

   例如，**hbase org.apache.hadoop.hbase.mapreduce.Import -Dimport.bulk.output=/user/hbase/output_t1 t1 /user/hbase/t1**

   hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles /user/hbase/output_t1 t1

添加主备集群备份关系。

步骤 15 在 HBase shell 中执行如下命令，创建主集群 HBase 与备集群 HBase 之间的备份同步关系。

add_peer '*备集群 ID*', *CLUSTER_KEY => '备集群 ZooKeeper 地址信息*',**{HDFS_CONFS => true}**

- 备集群 ID 表示主集群识别备集群使用的 id，建议使用字母与数字。

- 备集群 ZooKeeper 地址信息包含 ZooKeeper 业务 IP 地址、侦听客户端连接的端口和备集群的 HBase 在 ZooKeeper 上的根目录。
- **{HDFS_CONFS => true}**表示将主集群的默认 HDFS 配置信息同步到对应集群，用于备集群的 HBase 访问主集群的 HDFS。如果不启用 Bulkload 批量写数据备份，可以不使用此参数。

  例如，添加包含 BulkLoad 数据的主备集群备份关系，若备集群 ID 为“replication2”，备集群 ZooKeeper 地址信息为“**192.168.40.2,192.168.40.3,192.168.40.4:2181:/hbase**”。

  - 安全集群请执行：**add_peer 'replication2',CLUSTER_KEY => '192.168.40.2,192.168.40.3,192.168.40.4:2181:/hbase',CONFIG => { "hbase.regionserver.kerberos.principal" => "<val>", "hbase.master.kerberos.principal" => "<val2>" }**，普通集群请执行 **add_peer 'replication2',CLUSTER_KEY => '192.168.40.2,192.168.40.3,192.168.40.4:2181:/hbase'**

    其中参数“hbase.master.kerberos.principal”和“hbase.regionserver.kerberos.principal”为安全集群中 hbase 的 kerberos 用户，可搜索客户端中 hbase-site.xml 文件得到参数值。例如，客户端安装在 master 节点的“/opt/client”下，则可使用命令 **grep "kerberos.principal" /opt/client/HBase/hbase/conf/hbase-site.xml -A1** 获取，如下图所示。

    图8-1 获取 hbase 的 principal

    ```
    [root@xxxxx-xxxxx6 opt]# grep "kerberos.principal" /opt/client/HBase/hbase/conf/hbase-site.xml -A1
    <name>hbase.regionserver.kerberos.principal</name>
    <value>hbase/hadoop.hadoop.com@HADOOP.COM</value>
    --
    <name>hbase.master.kerberos.principal</name>
    <value>hbase/hadoop.hadoop.com@HADOOP.COM</value>
    --
    ```

说明

1. 获取 ZooKeeper 业务 IP 地址。

   登录 MRS 控制台，单击集群名称，选择“组件管理 > ZooKeeper > 实例”，获取 ZooKeeper 业务 IP 地址。

2. 在 ZooKeeper 服务参数“全部配置”界面，搜索获取 clientPort，即为客户端连接服务器的端口。

3. 执行 **list_peers** 命令判断主备备份关系添加结果，当界面提示以下信息表示成功。

   ```
   hbase(main):003:0> list_peers
   PEER_ID CLUSTER_KEY ENDPOINT_CLASSNAME STATE REPLICATE_ALL NAMESPACES
   TABLE_CFS BANDWIDTH SERIAL
   replication2 192.168.0.13,192.168.0.177,192.168.0.25:2181:/hbase ENABLED
   true   0 false
   ```

指定主备集群写数据状态。

步骤 16 在主集群 HBase shell 界面，执行以下命令保持写数据状态。

set_clusterState_active

界面提示以下信息表示执行成功：

```
hbase(main):001:0> set_clusterState_active
=> true
```

步骤 17 在备集群 HBase shell 界面，执行以下命令保持只读数据状态。

set_clusterState_standby

界面提示以下信息表示执行成功：

```
hbase(main):001:0> set_clusterState_standby
=> true
```

启用 HBase 备份功能同步数据。

步骤 18 检查备集群的 HBase 服务实例中，是否已存在一个命名空间，与待启用备份功能的 HBase 表所属的命名空间名称相同？

在备集群的 HBase shell 中，执行 **list_namespace** 命令，查询命名空间。

- 是，存在同名的命令空间，执行步骤 19。
- 否，不存在同名的命令空间，需先在备集群的 HBase shell 中，执行 **create_namespace 'ns1'**创建同名的命名空间，然后执行步骤 19。

步骤 19 在主集群的 HBase shell 中，执行以下命令，启用主集群表的数据实时备份功能，确保后续主集群中修改的数据能够实时同步到备集群中。

一次只能针对一个 HTable 进行数据同步。

enable_table_replication '*表名***'**

说明

- 若备集群中不存与要开启实时同步的表同名的表，则该表会自动创建。
- 若备集群中存在与要开启实时同步的表同名的表，则两个表的结构必须一致。
- 若'表名'设置了加密算法 SMS4 或 AES，则不支持对此 HBase 表启用将数据从主集群实时同步到备集群的功能。
- 若备集群不在线，或备集群中已存在同名但结构不同的表，启用备份功能将失败。

  若备集群不在线，请启动备集群。

  若备集群中已存在同名但结构不同的表，请修改备集群的表结构为相同的表结构。在备集群的 HBase shell 中，执行 ***alter*** 命令，参考示例修改。

步骤 20 在主集群的 HBase shell 中，执行以下命令，启用主集群的实时备份功能，同步 HBase 的权限表。

enable_table_replication 'hbase:acl'

说明

主集群 HBase 源数据表修改权限时，如果备集群需要正常读取数据，请修改备集群角色的权限。

检验主备集群数据同步状态。

步骤 21 在 HBase 客户端执行以下命令，校验主备集群同步的数据。启用备份同步功能后，也可以执行该命令检验新的同步数据是否一致。

hbase org.apache.hadoop.hbase.mapreduce.replication.VerifyReplication --starttime=*开始时间* **--endtime**=*结束时间 列族名称 备集群 ID 表名*

说明

- 开始时间必须早于结束时间。
- 开始时间和结束时间需要填写时间戳的格式，例如执行 **date -d "2015-09-30 00:00:00" +%s** 将普通时间转化为时间戳格式。因此命令返回的为 10 位数字（精确到秒），而 HBase 识别的为 13 位（精确到毫秒），所以需要在 date 命令返回的结果后补上 3 个 0。

主备集群发生倒换

说明

1. 当备集群需要被倒换为主集群时，请参见步骤 2~步骤 10 和步骤 15~步骤 20 重新配置主备关系。
2. 勿需执行“同步集群表数据”操作，即步骤 11~步骤 14。

----结束

相关命令

表8-5 HBase 备份

操作	命令	描述
建立主备关系	**add_peer** *'备集群 ID', '备集群地址信息'* 示例： **add_peer '1', 'zk1,zk2,zk3:2181:/hbase'** **add_peer '1', 'zk1,zk2,zk3:2181:/hbase1'**	建立主集群与备集群的关系，让其互相对应。如果启用 Bulkload 批量写数据备份，则命令为 **add_peer** *'备集群 ID'*,***CLUSTER_KEY*** => *'备集群地址信息'*，并配置参数 *"hbase.replication.conf.dir"*，同时手动拷贝主集群的 hbase 相应客户端配置文件到备集群的所有 RegionServer 节点，详情请参考步骤 4~11。
移除主备关系	**remove_peer** *'备集群 ID'* 示例： **remove_peer '1'**	在主集群中移除备集群的信息。
查询主备关系	**list_peers**	在主集群中查询已经设置的备集群的信息，主要为 Zookeeper 信息。
启用用户表实时同	**enable_table_replication** *'表名'* 示例：	在主集群中，设置已存在的表同步到备集群。

操作	命令	描述
步	**enable_table_replication 't1'**	
禁用用户表实时同步	**disable_table_replication** *'表名'* 示例： **disable_table_replication 't1'**	在主集群中，设置已存在的表不同步到备集群。
主备集群数据校验	**bin/hbase org.apache.hadoop.hbase.mapreduce.replication.VerifyReplication --starttime**=*开始时间* **--endtime=***结束时间 列族名称 备集群 ID 表名*	检查指定的表在主备集群间的数据是否一致。 命令行中参数说明如下： • 开始时间：如果未设置，则取默认的开始时间为 0。 • 结束时间：如果未设置，则取默认的结束时间为当前操作提交的时间。 • 表名：如果未输入表名，则默认校验所有的启用了实时同步的用户表。
切换数据写入状态	**set_clusterState_active** **set_clusterState_standby**	设置集群 HBase 表是否可写入数据。
新增或更新已经在对端集群保存的主集群中 HDFS 配置	**set_replication_hdfs_confs 'PeerId', {'key1' => 'value1', 'key2' => 'value2'}**	启用包含 Bulkload 数据的备份，在主集群修改 HDFS 参数时，新的参数值默认不会从主集群自动同步到备集群，需要手动执行命令同步。受影响的参数如下： • “fs.defaultFS” • “dfs.client.failover.proxy.provider.hacluster” • “dfs.client.failover.connection.retries.on.timeouts” • “dfs.client.failover.connection.retries” 例如，“fs.defaultFS”修改为“hdfs://hacluster_sale”，同步 HDFS 配置到 id 为 1 的备集群时执行：**set_replication_hdfs_confs '1', {'fs.defaultFS' => 'hdfs://hacluster_sale'}**

8.5 配置 HBase 参数

说明

该章节操作仅适用于 MRS 3.x 之前版本集群。

当 MRS 服务中默认的参数配置不足以满足用户需要时，用户可以自定义修改参数配置来适应自身需求。

步骤 1 登录集群详情页面，选择“组件管理”。

说明

若集群详情页面没有“组件管理”页签，请先完成 IAM 用户同步（在集群详情页的“概览”页签，单击“IAM 用户同步”右侧的“同步”进行 IAM 用户同步）。

步骤 2 选择“HBase > 服务配置”，将“基础配置”切换为“全部配置”，进入 HBase 配置界面修改参数配置。

表8-6 HBase 参数说明

参数	参数说明	参数值
hbase.regionserver.hfile.durable.sync	设置是否启用 Hfile 耐久性以将数据持久化到磁盘。若将该参数设置为 true，由于每个 Hfile 写入 HBase 时都会被 hadoop fsync 同步到磁盘上，则 HBase 性能将受到影响。 该参数仅在 MRS 1.9.2 及之前版本存在。	取值范围： • true • false 默认值为 true
hbase.regionserver.wal.durable.sync	设置是否启用 WAL 文件耐久性以将 WAL 数据持久化到磁盘。若将该参数设置为 true，由于每个 WAL 的编辑都会被 hadoop fsync 同步到磁盘上，则 HBase 性能将受到影响。 该参数仅在 MRS 1.9.2 及之前版本存在。	取值范围： • true • false 默认值为 true

----结束

8.6 启用集群间拷贝功能

操作场景

当用户需要将保存在 HDFS 中的数据从当前集群备份到另外一个集群时，需要使用 DistCp 工具。DistCp 工具依赖于集群间拷贝功能，该功能默认未启用。两个集群都需要配置。

该任务指导系统管理员在 MRS 修改参数以启用集群间拷贝功能。

对系统的影响

启用集群间复制功能需要重启 Yarn，服务重启期间无法访问。

前提条件

两个集群 HDFS 的参数“hadoop.rpc.protection”需使用相同的数据传输方式。设置为“privacy”表示加密，“authentication”表示不加密。

说明

参考修改集群服务配置参数，进入 HDFS 服务参数“全部配置”界面“，搜索 hadoop.rpc.protection 查看。

针对 MRS 3.x 之前版本，在集群详情页选择“组件管理 > HDFS > 服务配置”，将“基础配置”切换为“全部配置”，搜索 hadoop.rpc.protection 查看。

操作步骤

步骤 1 进入 Yarn 服务参数“全部配置”界面，具体操作请参考修改集群服务配置参数。

说明

若集群详情页面没有“组件管理”页签，请先完成 IAM 用户同步（在集群详情页的“概览”页签，单击“IAM 用户同步”右侧的“同步”进行 IAM 用户同步）。

步骤 2 左边菜单栏中选择“Yarn > 集群间拷贝”。

步骤 3 设置“dfs.namenode.rpc-address”参数的“haclusterX.remotenn1”值为对端集群其中一个 NameNode 实例的业务 IP 和 RPC 端口，设置“haclusterX.remotenn2”值为对端集群另外一个 NameNode 实例的业务 IP 和 RPC 端口。按照“*IP:port*”格式填写。

说明

针对 MRS 3.x 版本集群，登录 FusionInsight Manager 页面，选择“集群 > *待操作集群的名称* > 服务 > HDFS > 实例”，获取 NameNode 实例的业务 IP。

针对 MRS 3.x 之前版本，在集群详情页选择“组件管理 > HDFS > 实例”，获取 NameNode 实例的业务 IP。

“dfs.namenode.rpc-address.haclusterX.remotenn1”和“dfs.namenode.rpc-address.haclusterX.remotenn2”不区分主备 NameNode。NameNode RPC 端口默认为“9820”，不支持通过 Manager 修改。

修改后参数值例如：“10.1.1.1:9820”和“10.1.1.2:9820”。

步骤 4 保存配置并在概览页面选择“更多 > 重启服务”，重启 Yarn 服务。

界面提示“操作成功。”，单击“完成”，Yarn 服务启动成功。

步骤 5 登录另外一个集群，重复以上操作。

----结束

8.7 使用 ReplicationSyncUp 工具

前提条件

1. 主备集群已经安装并且启动。
2. 主备集群上的时间必须一致，而且主备集群上的 NTP 服务必须使用同一个时间源。
3. 当主集群 HBase 服务关闭时，Zookeeper 和 HDFS 服务应该启动并运行。
4. 该工具应该由启动 HBase 进程的系统用户运行。
5. 如果处于安全模式，请确保备用集群的 HBase 系统用户具有主集群 HDFS 的读取权限。因为它将更新 HBase 系统 Zookeeper 节点和 HDFS 文件。
6. 主集群 HBase 故障后，主集群的 Zookeeper，文件系统和网络依然可用。

场景介绍

Replication 机制可以使用 WAL 将一个集群的状态与另一个集群的状态保持同步。启用 HBase 备份后，若主集群出现故障，ReplicationSyncUp 工具会使用来自 zookeeper 的信息将主集群中的启用 HBase 备份功能的数据增量同步到备集群中。数据同步完成后，备集群可以作为主集群使用。

参数配置

参数	描述	默认值
hbase.replication.bulkload.enabled	是否开启批量加载数据复制功能。参数值类型为 Boolean。开启批量加载数据复制功能后该参数须在主集群中设置为 true。	false
hbase.replication.cluster.id	源 HBase 集群 ID。开启批量加载数据复制功能是必须设置该参数，在源集群定义。参数值类型为 String。	-

工具使用

在主集群 client 上输入如下命令使用：

hbase org.apache.hadoop.hbase.replication.regionserver.ReplicationSyncUp -Dreplication.sleep.before.failover=1

说明

replication.sleep.before.failover 是指在 RegionServer 启动失败时备份其剩余数据前需要的休眠时间。由于 30 秒（默认值）的睡眠时间没有任何意义，因此将其设置为 1（s），使备份过程更快触发。

注意事项

1. 当主集群关闭时，此工具将从 ZooKeeper 节点（RS znode）获得 WAL 的处理进度以及 WAL 的处理队列，并将未复制的队列复制到备集群中。
2. 每个主集群的 RegionServer 在备集群 ZooKeeper 上的 replication 节点下都有自己的 znode。它包含每个对等集群的一个 znode。
3. 当 Regionserver 故障时，主集群的每个 RegionServer 都会通过 watcher 收到通知，并尝试锁定故障 RegionServer 的 znode，包含它的队列。成功创建的 RegionServer 会将所有队列转移到自己队列的 znode 下。队列传输后，它们将从旧位置删除。
4. 在主集群关闭期间，ReplicationSyncUp 工具将使用来自 ZooKeeper 节点的信息同步主备集群的数据，并且 RegionServer znode 的 wals 将被移动到备集群下。

限制和约束

如果备集群处于关闭状态或关闭了对等关系，该工具正常运行，只有该对等关系复制不会发生。

8.8 GeoMesa 命令行简介

说明

该章节内容仅适用于 MRS 3.1.0 及之后版本。

本节介绍常用的 GeoMesa 命令。更多的 GeoMesa 命令，请参见 https://www.geomesa.org/documentation/user/accumulo/commandline.html。

安装 hbase 客户端后，加载环境变量后，可使用 geomesa-hbase 命令行。

- 查看 classpath

  执行“classpath”命令，将会返回当前命令行工具的所有 classpath 信息。

  bin/geomesa-hbase classpath

- 创建表

  执行“create-schema”命令创建表，创建表时至少要指定目录名称与表名称，以及表规格。

bin/geomesa-hbase create-schema -c geomesa -f test -s Who:String,What:java.lang.Long,When:Date,*Where:Point:srid=4326,Why:String

- 描述表

  执行“describe-schema”命令获取表描述信息，描述表信息时必须要指定目录名称与表名称。

  bin/geomesa-hbase describe-schema -c geomesa -f test

- 批量导入数据

  执行“ingest”命令批量导入数据，导入时需要指定目录名称，表名称，表规格，以及相应的数据转换器等。

  数据(车牌号，车辆颜色，经度，纬度，时间)：data.csv，并将数据表放在 data 文件夹中。

```
AAA,red,113.918417,22.505892,2017-04-09 18:03:46
BBB,white,113.960719,22.556511,2017-04-24 07:38:47
CCC,blue,114.088333,22.637222,2017-04-23 15:07:54
DDD,yellow,114.195456,22.596103,2017-04-21 21:27:06
EEE,black,113.897614,22.551331,2017-04-09 09:34:48
```

  表结构定义：myschema.sft，并将 myschema.sft 放在 geomesa 命令行工具的 conf 文件夹中。

```
geomesa.sfts.cars = {
  attributes = [
      { name = "carid", type = "String", index = true }
      { name = "color", type = "String", index = false }
      { name = "time", type = "Date",   index = false }
      { name = "geom", type = "Point",  index = true,srid = 4326,default =
true }
  ]
}
```

  转换器定义：myconvertor.convert，并将 myconvertor.convert 放在 geomesa 命令行工具的 conf 文件夹中。

```
geomesa.converters.cars= {
    type   = "delimited-text",
    format = "CSV",
    id-field = "$fid",
    fields = [
      { name = "fid",     transform = "concat($1,$5)" }
      { name = "carid",   transform = "$1::string" }
      { name = "color",   transform = "$2::string" }
      { name = "lon",     transform = "$3::double" }
      { name = "lat",     transform = "$4::double" }
      { name = "geom",    transform = "point($lon,$lat)" }
      { name = "time",    transform = "date('YYYY-MM-dd HH:mm:ss',$5)" }
    ]
}
```

  执行命令导入数据：

  bin/geomesa-hbase ingest -c geomesa -C conf/myconvertor.convert -s conf/myschema.sft data/data.csv

  数据导入其他参数具体说明请参见：
  https://www.geomesa.org/documentation/user/accumulo/examples.html#ingesting-data

- 解释查询

  执行“explain”命令获取指定查询语句执行计划的解释说明，解释语句时必须指定目录名称和表名称，以及给定查询语句。

  bin/geomesa-hbase explain -c geomesa -f cars -q "carid = 'BBB'"

- 统计分析

  执行“stats-analyze”命令对数据表进行统计分析，同时还可以进一步执行“stats-bounds”，“stats-count”，“stats-histogram”，“stats-top-k”命令对数据表做更详细的统计。

  bin/geomesa-hbase stats-analyze -c geomesa -f cars

  bin/geomesa-hbase stats-bounds -c geomesa -f cars

  bin/geomesa-hbase stats-count -c geomesa -f cars

  bin/geomesa-hbase stats-histogram -c geomesa -f cars

  bin/geomesa-hbase stats-top-k -c geomesa -f cars

- 导出 feature

  执行“export”命令导出 feature，导出时必须指定目录名称和表名称，同时还可以根据指定的查询语句进行导出。

  bin/geomesa-hbase export -c geomesa -f cars -q "carid = 'BBB'"

- 删除 feature

  执行“delete-features”命令删除 feature，删除时必须指定目录名称和表名称，同时还可以根据指定的查询语句进行删除。

  bin/geomesa-hbase delete-features -c geomesa -f cars -q "carid = 'BBB'"

- 获取目录中的全部表的名称

  执行“get-type-names”命令获取指定目录中的表名称。

  bin/geomesa-hbase get-type-names -c geomesa

- 删除表

  执行“remove-schema”命令删除表，删除表示至少要指定表所在的目录与表名称。

  bin/geomesa-hbase remove-schema -c geomesa -f test

  bin/geomesa-hbase remove-schema -c geomesa -f cars

- 删除目录

  执行“delete-catalog”命令删除指定的目录。

  bin/geomesa-hbase delete-catalog -c geomesa

8.9 使用 HIndex

8.9.1 HIndex 介绍

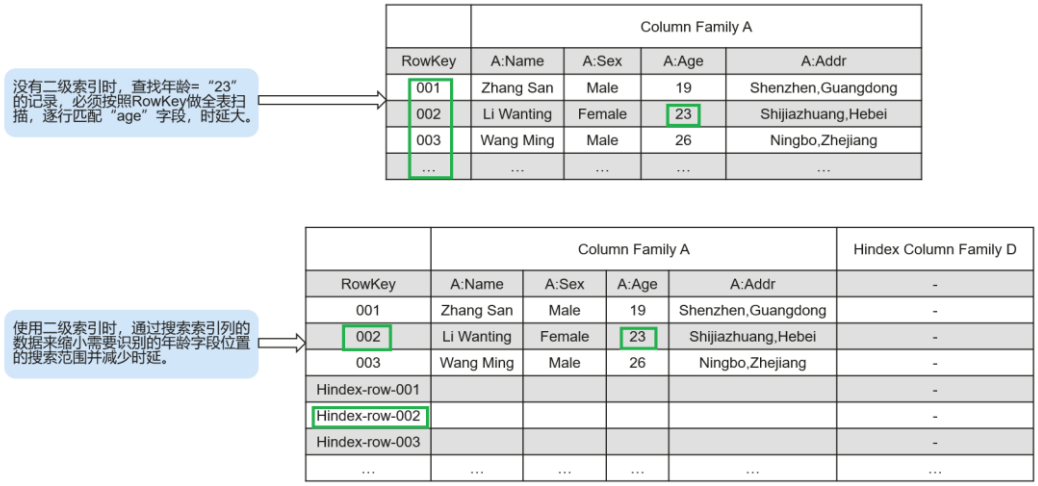
场景介绍

HBase 是基于 Key-Value 的分布式存储数据库，基于 rowkeys 对表中的数据按照字典进行排序。如果您根据指定的 rowkey 查询数据，或者扫描指定 rowkey 范围内的数据，

HBase 可以快速查找到需要读取的数据，从而提高效率。在大多数实际情况下，会需要查询列值为 XXX 的数据。HBase 提供了 Filter 功能来查询具有特定列值的数据：所有数据按 RowKey 的顺序进行扫描，然后将数据与特定的列值进行匹配，直到找到所需的数据。过滤器功能会 scan 一些不必要的数据以获取所需的数据。基于前面的描述，Filter 功能不能满足高性能标准频繁查询的要求。

这就是 HBase HIndex 产生的背景。如图 8-2 所示，HBase HIndex 为 HBase 提供了能够根据特定的列值进行索引的能力，使得查询会变得更快。

图8-2 HBase HIndex



说明

- 索引数据不支持滚动升级。
- 复合索引：用户必须将所有参与复合索引的列全部放入/删除，否则会导致数据不一致。
- 用户不应将任何 split policy 显式地配置到已建立索引的数据表中。
- 不支持 mutation 操作，如 increment,append。
- 不支持列索引的版本 maxVersions> 1。
- 添加索引的列值不应超过 32KB。
- 当用户数据由于列族级 TTL 失效而被删除时，相应的索引数据不会立即删除。索引数据将在 major compaction 期间被删除。
- 创建索引后，不应更改用户列族的 TTL。
- 如果在创建索引后将列族 TTL 更改为更高值，则应删除并重新创建索引，否则某些已生成的索引数据可能比用户数据先删除。
- 如果在创建索引后将列族 TTL 更改为较低值，则索引可能会晚于用户数据被删除。
- HBase 表启动容灾之后，主集群新建二级索引，索引表变更不会自动同步到备集群。要实现该容灾场景，必须执行以下操作：
- 在主表创建二级索引之后，需要在备集群使用相同方法创建结构、名称完全相同的二级索引。

1. 在主集群手动将索引列族（默认是 d）的 REPLICATION_SCOPE 设置为 1。

参数配置

1. 登录 MRS 控制台，单击集群名称，选择“组件管理”。
2. 进入 HBase 服务参数“全部配置”界面，具体操作请参考修改集群服务配置参数。
3. 在 HBase 全部配置界面查看参数。

配置入口	配置项	默认值	描述
“HMaster > 系统”	hbase.coprocessor.master.classes	org.apache.hadoop.hbase.hindex.server.master.HIndexMasterCoprocessor,com.xxx.hadoop.hbase.backup.services.RecoveryCoprocessor,org.apache.ranger.authorization.hbase.RangerAuthorizationCoprocessor,org.apache.hadoop.hbase.security.access.ReadOnlyClusterEnabler,org.apache.hadoop.hbase.rsgroup.RSGroupAdminEndpoint	该协处理器用于在启用 Hindex 功能后处理 Master 级的操作，比如创建索引 meta 表，添加索引，删除索引，删除表删除索引元数据。
“RegionServer > RegionServer”	hbase.coprocessor.regionserver.classes	org.apache.hadoop.hbase.hindex.server.regionserver.HIndexRegionServerCoprocessor,org.apache.hadoop.hbase.JMXListener,org.apache.ranger.authorization.hbase.RangerAuthorizationCoprocessor	该协处理器用于在启用 Hindex 功能后实际上处理 master 下发到 Regionserver 上的操作。
	hbase.coprocessor.region.classes	org.apache.hadoop.hbase.hindex.server.regionserver.HIndexRegionCoprocessor,org.apache.hadoop.hbase.s	该协处理器用于在启用 Hindex 功能后实际上操作 Region 上的数据。

配置入口	配置项	默认值	描述
		ecurity.token.TokenProvider,com.xxx.hadoop.hbase.backup.services.RecoveryCoprocessor,org.apache.ranger.authorization.hbase.RangerAuthorizationCoprocessor,org.apache.hadoop.hbase.security.access.SecureBulkLoadEndpoint,org.apache.hadoop.hbase.security.access.ReadOnlyClusterEnabler,org.apache.hadoop.hbase.coprocessor.MetaTableMetrics	
	hbase.coprocessor.wal.classes	org.apache.hadoop.hbase.hindex.server.regionserver.HIndexRegionServerCoprocessor,org.apache.hadoop.hbase.JMXListener,org.apache.ranger.authorization.hbase.RangerAuthorizationCoprocessor	该协处理器用于 Replication，其会过滤掉索引数据以避免索引数据发送到对等集群中，对等集群中的数据索引数据将会自己生成。 该参数仅 MRS 3.x 之前版本支持。

说明

1.上述默认值为启用 HBase HIndex 功能后需额外配置的值，当前支持 HBase HIndex 功能的 MRS 集群默认已配置。

2.必须确保 master 参数配置在 hmster 上，region/regionserver 参数配置在 regonserver 上。

相关接口

使用 HIndex 的 API 都在类 org.apache.hadoop.hbase.hindex.client.HIndexAdmin 中，相关接口介绍如下：

操作	接口	描述	注意事项

操作	接口	描述	注意事项
添加索引	addIndices()	将索引添加到没有数据的表中。调用此接口会将用户指定的索引添加到表中，但会跳过生成索引数据。因此，在此操作之后，索引不能用于scan/filter 操作。它的使用场景为用户想要在具有大量预先存在用户数据的表上批量添加索引，其具体操作为使用诸如TableIndexer 工具之类的外部工具来构建索引数据。	• 索引一旦添加则不能修改。若要修改，则需先删除旧的索引然后重新创建。 • 用户应注意不要在具有不同索引名称的相同列上创建两个索引。如果这样做，将会导致存储和处理的浪费。 • 索引不能添加到系统表中。 • 向索引列 put 数据时不支持 append 和 increment 操作。 • 如果客户端出现任何故障，除非发生DoNotRetryIOException，否则用户应该重试。 • 索引列族根据可用性按顺序从以下条件中选择： – 默认索引列族“d”或者如果设置了属性“hindex.default.family.name”的值，则以该值为准。 – 符号＃，@，$或％ – ＃0，@ 0，$ 0，％0，＃1，@ 1 ...上至＃255，@ 255，$ 255，％255 – throw Exception • 可以通过 HIndex TableIndexer 工具添加索引而无需建立索引数据。
	addIndicesWithData()	将索引添加到有数据的表中。此方法将用户指定的索引添加到表中，并会对已经存在的用户数据创建对应的索引数据，也可先调用该方法生成索引再在存入用户数据的同时生成索引数据。在此操作之后，这些索引立即可用于 scan/filter 操作。	

操作	接口	描述	注意事项
删除索引	dropIndices()	仅删除索引。该 API 从表中删除用户指定的索引，但跳过相应的索引数据。在此操作之后，索引不能用于 scan/filter 操作。集群在 major compaction 期间会自动删除旧的索引数据。 此 API 使用场景为表中包含大量索引数据且 dropIndicesWithData()不可行。另外，用户也可以通过 TableIndexer 工具删除索引以及索引数据。	• 在索引的状态为 ACTIVE，INACTIVE 和 DROPPING 时，允许禁用索引的操作。 • 对于使用 dropIndices()删除索引的操作，用户必须确保在将索引添加到具有相同索引名的表之前，相应的索引数据已被删除（即 major compaction 已完成）。 • 用户删除相应的索引会删除： – 一个带有索引的列族。 – 组合索引所有列族中的任一个列族。 • 索引可以通过 HIndex TableIndexer 工具与索引数据一起删除。
	dropIndicesWithData()	删除索引数据。此 API 删除用户指定的索引，并删除用户表中与这些索引对应的所有索引数据。在此操作之后，删除的索引完全从表中删除，不再可用于 scan/filter 操作。	
启用/禁用索引	disableIndices()	该 API 禁用所有用户指定的索引，使其不再可用于 scan/filter 操作。	• 在索引的状态为 ACTIVE，INACTIVE 和 BUILDING 时允许启用索引的操作。 • 在索引的状态为 ACTIVE 和 INACTIVE 时允许禁用索引操作。 • 在禁用索引之前，用户必须确保索引数据与用户数据一致。如果在索引处于禁用状态期间没有在表中添加新的数据，索引数据与用户数据将保持一
	enableIndices()	该 API 启用所有用户指定的索引，使其可用于 scan/filter 操作。	

操作	接口	描述	注意事项
			致。 • 启用索引时，可以通过使用 TableIndexer 工具构建索引来保证数据一致性。
查看已创建的索引	listIndices()	该 API 可用于列出给定表中的所有索引。	无

基于索引查询数据

在具有索引的用户表中，可以使用 Filter 来查询数据。对于创建单索引和组合索引的用户表，使用过滤器查询的结果与没有使用索引的表相同，但数据查询性能高于没有使用索引的表。

索引的使用规则如下：

- 对于为一个或多个列创建单个索引的情况：
  - 当将此列用于 AND 或 OR 查询筛选时，使用索引可以提高查询性能。

    例如，Filter_Condition（IndexCol1）AND / OR Filter_Condition（IndexCol2）。
  - 当在查询中使用“索引列和非索引列”进行过滤时，此索引可以提高查询性能。

    例如，Filter_Condition（IndexCol1）AND Filter_Condition（IndexCol2）AND Filter_Condition（NonIndexCol1）。
  - 当在查询中使用“索引列或非索引列”进行筛选时，但不使用索引，查询性能不会提高。

    例如，Filter_Condition（IndexCol1）AND / OR Filter_Condition（IndexCol2）OR Filter_Condition（NonIndexCol1）。
- 对于为多个列创建组合索引的情况：
  - 当用于查询的列是组合索引的全部或部分列并且与组合索引具有相同的顺序时，使用索引会提高查询性能。

    例如，为 C1，C2 和 C3 创建组合索引。
    - 该索引在以下情况下生效：

      Filter_Condition（IndexCol1）AND Filter_Condition（IndexCol2）AND Filter_Condition（IndexCol3）

      Filter_Condition（IndexCol1）AND Filter_Condition（IndexCol2）

      FILTER_CONDITION（IndexCol1）
    - 该索引在下列情况下不生效：

      Filter_Condition（IndexCol2）AND Filter_Condition（IndexCol3）

      Filter_Condition（IndexCol1）AND Filter_Condition（IndexCol3）

FILTER_CONDITION（IndexCol2）

FILTER_CONDITION（IndexCol3）

- 当在查询中使用“索引列和非索引列”进行过滤时，使用索引可提高查询性能。

  例如：

  Filter_Condition（IndexCol1）AND Filter_Condition（NonIndexCol1）

  Filter_Condition（IndexCol1）AND Filter_Condition（IndexCol2）AND Filter_Condition（NonIndexCol1）

- 当在查询中使用“索引列或非索引列”进行筛选时，但不使用索引，查询性能不会提高。

  例如：

  Filter_Condition（IndexCol1）OR Filter_Condition（NonIndexCol1）

  （Filter_Condition（IndexCol1）AND Filter_Condition（IndexCol2））OR（Filter_Condition（NonIndexCol1））

- 当多个列用于查询时，只能为组合索引中的最后一列指定值范围，而其他列只能设置为指定值。

  例如，为 C1，C2 和 C3 创建组合索引。在范围查询中，只能为 C3 设置数值范围，过滤条件为“C1 = XXX，C2 = XXX，C3 = 数值范围”。

查询策略选择

使用 SingleColumnValueFilter 或 SingleColumnRangeFilter，它会在一个在过滤条件中提供确定值 column_family:qualifierpair（我们称该列为 col1）。

若 col1 作为表上的第一个索引列，那么该表上的任何索引都可以成为查询期间使用的候选索引。例如：

如果有 col1 上的索引，可以将此索引作为候选索引，因为 col1 是此索引的第一列也是唯一的列；如果在 col1 和 col2 上有另一个索引，可以将此索引视为候选索引，因为 col1 是索引列列表中的第一列。另一方面，如果在 col2 和 col1 上有一个索引，则不能将此索引作为候选索引，因为索引列列表中的第一列不是 col1。

现在最适合使用索引的方法是，当有多个候选索引时，需要从可能的候选索引中选择最适合 scan 数据的索引。

可借助以下方案来了解如何选择索引策略：

- 最好可以完全匹配。

  场景：有两个索引可用，一个用于 col1＆col2，另一个单独用于 col1。

  在上面的场景中，第二个索引会比第一个索引更好，因为它会使我们 scan 的较少索引数据。

- 如果有多个候选多列索引，则选择具有较少索引列的索引。

  场景：有两个索引可用，一个用于 col1＆col2，另一个用于 col1＆col2＆col3。

  在这种情况下，最好使用 col1 和 col2 上的索引，因为它会使我们 scan 的较少索引数据。

说明

- 基于索引查询时索引的状态必须为 ACTIVE（可通过调用 listIndices() API 查看索引的状态）。
- 为了保证基于索引查询数据的正确性，用户应该确保索引数据与用户数据的一致性。
- 使用以下命令可通过 HBase shell 客户端执行复杂查询（假定此时 已为指定列建立索引）。

  scan 'tablename', {FILTER => "SingleColumnValueFilter(family, qualifier, compareOp, comparator, filterIfMissing, latestVersionOnly)"}

  例如：**scan 'test', {FILTER => "SingleColumnValueFilter('info', 'age', =, 'binary:26', true, true)"}**

  （在以上场景中，用户希望在结果中保存没有查询到的列所在行时，不应该在任何这样的列上创建任何索引，因为如果查询的列不存在于其中时，使用 SCVF 扫描索引列会过滤出一行。而使用 filterIfMissingset 为 false（这是默认值）的 SCVF 扫描非索引列时，也将会在结果中返回没有查询列的行。因此，为避免查询结果不一致，建议在为索引列创建 SCVF 后将 filterIfMissing 设置为 true。）

- 在 hbase shell 中可以通过以下命令查看为用户数据建立的索引数据。

  scan 'tablename', {ATTRIBUTES => {'FETCH_INDEX_DATA' => 'true'}}

8.9.2 批量加载索引数据

场景介绍

HBase 本身提供了 ImportTsv＆LoadIncremental 工具来批量加载用户数据。当前提供了 HIndexImportTsv 来支持加载用户数据的同时可以完成对索引数据的批量加载。HIndexImportTsv 继承了 HBase 批量加载数据工具 ImportTsv 的所有功能。此外，若在执行 HIndexImportTsv 工具之前未建表，直接运行该工具，将会在创建表时创建索引，并在生成用户数据的同时生成索引数据。

操作步骤

1. 将数据导入到 HDFS 中。

   hdfs dfs -mkdir <*inputdir*>

   hdfs dfs -put <local_data_file> <*inputdir*>

   例如定义数据文件“data.txt”，内容如下：

```
12005000201,Zhang San,Male,19,Shenzhen City, Guangdong Province
12005000202,Li Wanting,Female,23,Hangzhou City, Zhejiang Province
12005000203,Wang Ming,Male,26,Ningbo City, Zhejiang Province
12005000204,Li Gang,Male,18,Xiangyang City, Hubei Province
12005000205,Zhao Enru,Female,21,Shangrao City, Jiangxi Province
12005000206,Chen Long,Male,32,Zhuzhou City, Hunan Province
12005000207,Zhou Wei,Female,29,Nanyang City, Henan Province
12005000208,Yang Yiwen,Female,30,Wenzhou City, Zhejiang Province
12005000209,Xu Bing,Male,26,Weinan City, Shanxi Province
12005000210,Xiao Kai,Male,25,Dalian City, Liaoning Province
```

   执行以下命令：

   hdfs dfs -mkdir /datadirImport

hdfs dfs -put data.txt /datadirImport

2. 建表 bulkTable，进入 hbase shell，执行命令建表，例如：

   create 'bulkTable', {NAME => 'info',COMPRESSION => 'SNAPPY', DATA_BLOCK_ENCODING => 'FAST_DIFF'},{NAME=>'address'}

   执行完成后退出 hbase shell。

3. 执行如下命令，生成 HFile 文件（StoreFiles）：

   hbase org.apache.hadoop.hbase.hindex.mapreduce.HIndexImportTsv -Dimporttsv.separator=<*separator*>

   -Dimporttsv.bulk.output=<*/path/for/output*> **-Dindexspecs.to.add=**<indexspecs> **-Dimporttsv.columns=**<columns> tableName <*inputdir*>

   - -Dimport.separator：分隔符，例如，-Dimport.separator=','。
   - -Dimport.bulk.output=</path/for/output>：指的是执行结果输出路径，需指定一个不存在的路径。
   - <columns>：指的是导入数据在表中的对应关系，例如，-Dimporttsv.columns=HBASE_ROW_KEY,info:name,info:gender,info:age,address:city,address:province。
   - <tablename>：指的是要操作的表名。
   - <inputdir>：指的是要批量导入的数据目录。
   - -Dindexspecs.to.add=<indexspecs>：指的是索引名与列的映射，例如-Dindexspecs.to.add='index_bulk=>info:[age->String]'。 其构成可以表示如下：

     indexNameN=>familyN :[columnQualifierN-> columnQualifierDataType], [columnQualifierM-> columnQualifierDataType];familyM: [columnQualifierO-> columnQualifierDataType]# indexNameN=> familyM: [columnQualifierO-> columnQualifierDataType]

     其中，列限定符用逗号（,）分隔

     例如：“index1 => f1：[c1-> String]，[c2-> String]”

     列族由分号（;）分隔

     例如：“index1 => f1：[c1-> String]，[c2-> String]; f2：[c3-> Long]”

     多个索引由#号键（＃）分隔

     例如：“index1 => f1：[c1-> String]，[c2-> String]; f2：[c3-> Long]＃index2 => f2：[c3-> Long]”

     列限定的数据类型：

     可用的数据类型有：STRING，INTEGER，FLOAT，LONG，DOUBLE，SHORT，BYTE，CHAR

说明

数据类型也可以用小写传递。

例如执行以下命令：

hbase org.apache.hadoop.hbase.hindex.mapreduce.HIndexImportTsv -Dimporttsv.separator=',' -Dimporttsv.bulk.output=/dataOutput -Dindexspecs.to.add='index_bulk=>info:[age->String]' -Dimporttsv.columns=HBASE_ROW_KEY,info:name,info:gender,info:age,address:city,address:province bulkTable /datadirImport/data.txt

输出：

```
[root@shap000000406 opt]# hbase
org.apache.hadoop.hbase.hindex.mapreduce.HIndexImportTsv -
Dimporttsv.separator=',' -Dimporttsv.bulk.output=/dataOutput -
Dindexspecs.to.add='index_bulk=>info:[age->String]' -
Dimporttsv.columns=HBASE_ROW_KEY,info:name,info:gender,info:age,address:city,ad
dress:province bulkTable /datadirImport/data.txt
2018-05-08 21:29:16,059 INFO  [main] mapreduce.HFileOutputFormat2: Incremental
table bulkTable output configured.
2018-05-08 21:29:16,069 INFO  [main]
client.ConnectionManager$HConnectionImplementation: Closing master protocol:
MasterService
2018-05-08 21:29:16,069 INFO  [main]
client.ConnectionManager$HConnectionImplementation: Closing zookeeper
sessionid=0x80007c2cb4fd5b4d
2018-05-08 21:29:16,072 INFO  [main] zookeeper.ZooKeeper: Session:
0x80007c2cb4fd5b4d closed
2018-05-08 21:29:16,072 INFO  [main-EventThread] zookeeper.ClientCnxn:
EventThread shut down for session: 0x80007c2cb4fd5b4d
2018-05-08 21:29:16,379 INFO  [main] client.ConfiguredRMFailoverProxyProvider:
Failing over to 147
2018-05-08 21:29:17,328 INFO  [main] input.FileInputFormat: Total input files
to process : 1
2018-05-08 21:29:17,413 INFO  [main] mapreduce.JobSubmitter: number of splits:1
2018-05-08 21:29:17,430 INFO  [main] Configuration.deprecation:
io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2018-05-08 21:29:17,687 INFO  [main] mapreduce.JobSubmitter: Submitting tokens
for job: job_1525338489458_0002
2018-05-08 21:29:18,100 INFO  [main] impl.YarnClientImpl: Submitted application
application_1525338489458_0002
2018-05-08 21:29:18,136 INFO  [main] mapreduce.Job: The url to track the job:
http://shap000000407:8088/proxy/application_1525338489458_0002/
2018-05-08 21:29:18,136 INFO  [main] mapreduce.Job: Running job:
job_1525338489458_0002
2018-05-08 21:29:28,248 INFO  [main] mapreduce.Job: Job job_1525338489458_0002
running in uber mode : false
2018-05-08 21:29:28,249 INFO  [main] mapreduce.Job:  map 0% reduce 0%
2018-05-08 21:29:38,344 INFO  [main] mapreduce.Job:  map 100% reduce 0%
2018-05-08 21:29:51,421 INFO  [main] mapreduce.Job:  map 100% reduce 100%
2018-05-08 21:29:51,428 INFO  [main] mapreduce.Job: Job job_1525338489458_0002
completed successfully
2018-05-08 21:29:51,523 INFO  [main] mapreduce.Job: Counters: 50
```

4. 执行如下命令将生成的 HFile 导入 HBase 中：

   hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles *</path/for/output>* *<tablename>*

   例如执行以下命令：

   hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles /dataOutput bulkTable

   输出：

```
[root@shap000000406 opt]# hbase
org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles /dataOutput bulkTable
2018-05-08 21:30:01,398 WARN  [main] mapreduce.LoadIncrementalHFiles: Skipping
non-directory hdfs://hacluster/dataOutput/_SUCCESS
2018-05-08 21:30:02,006 INFO  [LoadIncrementalHFiles-0] hfile.CacheConfig:
```

```
Created cacheConfig: CacheConfig:disabled
2018-05-08 21:30:02,006 INFO  [LoadIncrementalHFiles-2] hfile.CacheConfig:
Created cacheConfig: CacheConfig:disabled
2018-05-08 21:30:02,006 INFO  [LoadIncrementalHFiles-1] hfile.CacheConfig:
Created cacheConfig: CacheConfig:disabled
2018-05-08 21:30:02,085 INFO  [LoadIncrementalHFiles-2] compress.CodecPool: Got
brand-new decompressor [.snappy]
2018-05-08 21:30:02,120 INFO  [LoadIncrementalHFiles-0]
mapreduce.LoadIncrementalHFiles: Trying to load
hfile=hdfs://hacluster/dataOutput/address/042426c252f74e859858c7877b95e510
first=12005000201 last=12005000210
2018-05-08 21:30:02,120 INFO  [LoadIncrementalHFiles-2]
mapreduce.LoadIncrementalHFiles: Trying to load
hfile=hdfs://hacluster/dataOutput/info/f3995920ae0247a88182f637aa031c49
first=12005000201 last=12005000210
2018-05-08 21:30:02,128 INFO  [LoadIncrementalHFiles-1]
mapreduce.LoadIncrementalHFiles: Trying to load
hfile=hdfs://hacluster/dataOutput/d/c53b252248af42779f29442ab84f86b8
first=\x00index_bulk\x00\x00\x00\x00\x00\x00\x00\x0018\x00\x0012005000204
last=\x00index_bulk\x00\x00\x00\x00\x00\x00\x00\x0032\x00\x0012005000206
2018-05-08 21:30:02,231 INFO  [main]
client.ConnectionManager$HConnectionImplementation: Closing master protocol:
MasterService
2018-05-08 21:30:02,231 INFO  [main]
client.ConnectionManager$HConnectionImplementation: Closing zookeeper
sessionid=0x81007c2cf0f55cc5
2018-05-08 21:30:02,235 INFO  [main] zookeeper.ZooKeeper: Session:
0x81007c2cf0f55cc5 closed
2018-05-08 21:30:02,235 INFO  [main-EventThread] zookeeper.ClientCnxn:
EventThread shut down for session: 0x81007c2cf0f55cc5
```

8.9.3 使用索引生成工具

场景介绍

为了快速对用户数据创建索引，HBase 提供了可通过 MapReduce 功能创建索引的 TableIndexer 工具，该工具可实现添加，构建和删除索引。具体使用场景如下：

- 在用户的表中预先存在大量数据的情况下，可能希望在某个列上添加索引。但是，使用 addIndicesWithData（）API 添加索引会生成与相关用户数据对应的索引数据，这将花费大量时间。另一方面，使用 addIndices（）创建的索引不会构建与用户数据对应的索引数据。因此，为了为这样的用户数据建立索引数据，用户可以使用 TableIndexer 工具来完成索引的构建。
- 如果索引数据与用户数据不一致，该工具可用于重新构建索引数据。

  如果用户暂时禁用索引并且在此期间，向禁用的索引列执行新的 put 操作，然后直接将索引从禁用状态启用可能会导致索引数据与用户数据不一致。因此，用户必须注意在再次使用之前重新构建所有索引数据。
- 对于大量现有的索引数据，用户可以使用 TableIndexer 工具将索引数据从用户表中完全删除。
- 对于未建立索引的用户表，该工具允许用户同时添加和构建索引。

使用方法

- **添加新的索引到用户表**

  命令如下所示：

  hbase org.apache.hadoop.hbase.hindex.mapreduce.TableIndexer -Dtablename.to.index=tablename -Dindexspecs.to.add='idx_0=>cf_0:[q_0->string],[q_1];cf_1:[q_2],[q_3]#*idx_1*=>cf_1:[q_4]'

  它需要以下参数:

  - **tablename.to.index**：表示创建索引的表的名称
  - **indexspecs.to.add**：表示与索引名与对应用户表的列的映射
  - **scan.caching**（可选）：包含一个整数值，表示在扫描数据表时将传递给扫描器的缓存行数

  上述命令中的参数描述如下：

  - **idx_1**：表示索引名称
  - **cf_0**：表示列族名称
  - **q_0**：表示列名称
  - **string**：表示数据类型。它可以是 STRING，INTEGER，FLOAT，LONG，DOUBLE，SHORT，BYTE 或 CHAR

  说明

  - '#'用于分隔索引，';' 用于分隔列族，','用于分隔列限定符。
  - 列名及其数据类型应包含在'[]'中。
  - 列名及其数据类型通过' - >'分隔。
  - 如果未指定具体列的数据类型，则使用默认数据类型（string）。
  - 如果未设置可选参数 scan.caching，则将采用默认值 1000。
  - 用户表必须存在。
  - 表中指定的索引不能存在.。
  - 如果用户表中已经存在名称为'd'的 ColumnFamily，则用户必须使用 TableIndexer 工具构建索引数据。

  在执行以上的命令之后，指定的索引将被添加到表中并且将处于 INACTIVE 状态。该行为与 addIndices() API 类似。

- **为用户表中的现有索引构建索引数据**

  该命令如下：

  hbase org.apache.hadoop.hbase.hindex.mapreduce.TableIndexer -Dtablename.to.index=tablename -Dindexnames.to.build ='idx_0＃idx_1'

  它采用以下参数：

  - **tablename.to.index**：表示创建索引的表的名称
  - **indexspecs.to.build**：表示与索引名称
  - **scan.caching**（可选）：包含一个整数值，表示在扫描数据表时将传递给扫描器的缓存行数

  上述命令中的参数描述如下：

- **idx_1**：表示索引名称

说明

- '#'用于分隔索引名称。
- 如果未设置可选参数 scan.caching，则将采用默认值 1000。
- 用户表必须存在。

在执行以上的命令之后，指定的索引将被设置为 ACTIVE 状态。用户扫描数据时可以使用它们。

- **从用户表中删除现有索引及其数据**

  该命令如下：

  hbase org.apache.hadoop.hbase.hindex.mapreduce.TableIndexer -Dtablename.to.index=tablename -Dindexnames.to.drop='idx_0＃idx_1'

  它需要以下参数：

  - **tablename.to.index**：表示创建索引的表的名称
  - **indexnames.to.drop**：表示应该和其数据一起删除的索引的名称（必须存在于表中）
  - **scan.caching**（可选）：其中包含一个整数值，指示在扫描数据表时将传递给扫描器的缓存行数

  上述命令中的参数描述如下：

  - **idx_1**：表示索引名称

  说明

  - '#'用于分隔索引名称。
  - 如果未设置可选参数 scan.caching，则将采用默认值 1000。
  - 用户表必须存在。

  在执行前面的命令之后，指定的索引将从表中删除。

- **为用户表添加新的索引以及基于现有数据的数据构建**

  该命令如下：

  hbase org.apache.hadoop.hbase.hindex.mapreduce.TableIndexer -Dtablename.to.index=tablename -Dindexspecs.to.add='idx_0 => cf_0：[q_0-> string],[q_1];cf_1:[q_2]，[q_3]＃idx_1 => cf_1:[q_4]' -Dindexnames.to.build='idx_0'

  说明

  - 参数与之前的情况相同。
  - 用户表必须存在。
  - indexspecs.to.add 中指定的索引不得存在于表中。
  - indexnames.to.build 中指定的索引名称必须已经存在于表中，或者应该是 indexspecs.to.add 的一部分。

  在执行前面的命令之后，indexspecs.to.add 中指定的所有索引都将添加到该表中，并且将为通过 indexnames.to.build 为指定的所有索引构建索引数据。

8.9.4 索引数据迁移

操作场景

MRS 1.7 及其以后版本中使用的索引与以前 MRS 版本中 HBase 使用的二级索引都不兼容。因此，为了将索引数据从以前的版本（MRS 1.5 及其以前版本）迁移到 MRS 1.7 及其以后版本，需要遵循以下步骤。

前提条件

1. 迁移数据时旧版本集群应为 MRS1.5 及其以前的版本，新版本集群应为 MRS1.7 及其以后的版本。
2. 迁移数据前用户应该有旧的索引数据。
3. 安全集群需配置跨集群互信和启用集群间拷贝功能，普通集群仅需启用集群间拷贝功能。详情请参见“用户指南 > 管理现有集群 > MRS 多用户权限管理 > 配置跨集群互信”和启用集群间拷贝功能。

操作步骤

把旧集群中的用户数据迁移至新集群中。迁移数据需单表手动同步新旧集群的数据，通过 Export、distcp、Import 来完成。

例如，当前旧集群有用户表（t1，索引名为 idx_t1）及其对应的索引表（t1_idx）。迁移数据的操作步骤如下：

1. 从旧集群导出表中数据。

   ```
   hbase org.apache.hadoop.hbase.mapreduce.Export -
   Dhbase.mapreduce.include.deleted.rows=true <tableName> <path/for/data>
   ```

   - <*tableName*>：指的是表名。例如，t1。
   - <*path/for/data*>：指的是保存源数据的路径，例如“/user/hbase/t1”。

   例如，**hbase org.apache.hadoop.hbase.mapreduce.Export -Dhbase.mapreduce.include.deleted.rows=true t1 /user/hbase/t1**

2. 把导出的数据按如下步骤复制到新集群中。

   ```
   hadoop distcp <path/for/data> hdfs://ActiveNameNodeIP:9820/<path/for/newData>
   ```

   - <*path/for/data*>：指的是旧集群保存源数据的路径。例如，/user/hbase/t1。
   - <path/for/newData>：指的是新集群保存源数据的路径。例如，/user/hbase/t1。

   其中，ActiveNameNodeIP 是新集群中主 NameNode 节点的 IP 地址。

   例如，**hadoop distcp /user/hbase/t1 hdfs://192.168.40.2:9820/user/hbase/t1**

   说明

   - 可手动把导出的数据复制到新集群 HDFS 中，如上路径：“/user/hbase/t1”。

3. 使用新集群 HBase 表用户，在新集群中生成 HFiles。

   ```
   hbase org.apache.hadoop.hbase.mapreduce.Import -
   Dimport.bulk.output=<path/for/hfiles> <tableName><path/for/newData>
   ```

   - <*path/for/hfiles*>：指的是新集群生成 HFiles 的路径。例如，/user/hbase/output_t1。

- <tableName>：指的是表名。例如，t1。
- <path/for/newData>：指的是新集群保存源数据的路径。例如，/user/hbase/t1。

例如，

hbase org.apache.hadoop.hbase.mapreduce.Import -Dimport.bulk.output=/user/hbase/output_t1 t1 /user/hbase/t1

4. 把生成的 HFiles 导入新集群相应表中。

   命令如下：

   ```
   hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles <path/for/hfiles> <tableName>
   ```

   - *<path/for/hfiles>*：指的是新集群生成 HFiles 的路径。例如，/user/hbase/output_t1。
   - <tableName>：指的是表名。例如，t1。

   例如，

   hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles /user/hbase/output_t1 t1

   说明

   1. 以上为迁移用户数据的过程，旧集群的索引数据迁移只需按照前三步操作，并更改相应表名为索引表名（如，t1_idx）。
   2. 迁移索引数据时无需执行 4。

5. 向新集群表中导入索引数据。

   a. 在新集群的用户表中添加与之前版本用户表相同的索引（名称为'd'的列族不应该已经存在于用户表中）。

      命令如下所示：

      ```
      hbase org.apache.hadoop.hbase.hindex.mapreduce.TableIndexer -Dtablename.to.index=<tableName> -Dindexspecs.to.add=<indexspecs>
      ```

      - -Dtablename.to.index=<tableName>：指的是表名。例如，-Dtablename.to.index=t1。
      - -Dindexspecs.to.add=<indexspecs>：指的是索引名与列的映射，例如-Dindexspecs.to.add='idx_t1=>info:[name->String]'。

      例如，

      hbase org.apache.hadoop.hbase.hindex.mapreduce.TableIndexer -Dtablename.to.index=t1 -Dindexspecs.to.add='idx_t1=>info:[name->String]'

   说明

   如果用户表中已经存在名称为'd'的 ColumnFamily，则用户必须使用 TableIndexer 工具构建索引数据。

   b. 运行 LoadIncrementalHFiles 工具加载索引数据，将旧集群索引数据加载到新集群表中。

      命令如下：

      ```
      hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles </path/for/hfiles> <tableName>
      ```

- </path/for/hfiles>：指的是索引数据在 HDFS 上的路径（其为-Dimport.bulk.output 中指定的索引生成路径）。例如，/user/hbase/output_t1_idx。
- <tableName>：指的是新集群中表名，例如，t1。

例如，

hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles /user/hbase/output_t1_idx t1

8.10 配置 RSGroup

操作场景

HBase 服务的数据节点较多，需要根据不同的业务规模将数据节点资源分配给特定的业务，从而达到资源独占使用的目的。当 AZ 容灾特性被开启时，为了保证 AZ 容灾生效，保障业务可靠性，在为 RSGroup 分配 RegionServer 时，需遵循分配结果能使该 RSGroup 在每个 AZ 下都存在 RegionServer 实例的规则。

说明

本章节内容仅适用于 MRS 3.1.2 及之后版本。

前提条件

- 已登录 Manager。
- 登录角色拥有 Manager 管理员权限。
- 将 RSGroup 最小节点数设置为下述三种情况的最大值。
  - 为了保证服务的可靠性，RSGroup 内的 RegionServer 节点数量需要配置一定的冗余量，确保冗余节点数>（RSGroup 内业务表 region 总数/2000)*50%。
  - 如果系统表在单独的 RSGroup，需要确保该 RSGroup 的节点数量>2。
  - 为了不影响滚动重启功能，如果 RegionServer 节点总数在 300 以内，那么单个 RSGroup 的节点数量不应小于 3。如果 RegionServer 节点总数大于等于 300，那么单个 RSGroup 的节点数量不应小于(节点数*1%)+1。

可能的影响

- 由于 RSGroup 约束了 region 转移可用的 RegionServer 节点，如果 RSGroup 内部分节点故障或者滚动重启，可能会触发 region 超过阈值的告警，也可能会导致业务性能下降。
- 当提交修改 RSGroup 请求产生大量 region 转移任务时，如果进行相关 RSGroup 操作会面临失败。需先观察原生页面的 region 转移情况，等待转移任务结束后再进行后续操作。

操作步骤

创建 RSGroup

步骤 1 在 FusionInsight Manager 界面，选择“集群 > 服务 > HBase > RSGroup 管理”。

步骤 2 单击“添加 RSGroup”按钮，在弹出的添加 RSGroup 页面填写新增的 RSGroup 名称，RSGroup 名称包括数字、字母或下划线（_），长度为 1-120 个字符。然后单击“确定”。

查看 RSGroup

步骤 3 选择待操作的 RSGroup，在操作列单击“查看”，即可在弹出框中查看该 RSGroup 的 RegionServers 详情和 Tables 详情。

说明

default RSGroup 是 HBase 的默认 RSGroup，所有已启动并且未手动添加到其他 RSGroup 的 RegionServer 节点都会添加到 default RSGroup。

修改 RSGroup 名称

步骤 4 选择待操作的 RSGroup，在操作列单击“修改名称”。在修改 RSGroup 名称弹出框中填写 RSGroup 新名称，新名称不能与已存在名称相同，单击“确定”。

修改 RSGroup

步骤 5 单击待操作的 RSGroup 名称，跳转到修改 RSGroup 页面。

步骤 6 勾选欲分配的 RegionServer 实例，单击“下一步”。

说明

- 一次分配操作仅允许勾选来自同一 RSGroup 的一个到多个 RegionServer 实例，且 default 组中的 RegionServer 的运行状态不为良好时不允许被勾选分配。若想要分配来自不同 RSGroup 的 RegionServer 实例，请分多次修改操作进行分配。
- 开启跨 AZ 特性时，分配操作需要保证分配结果能使每个 AZ 中均存在该 RSGroup 的 RegionServer 实例而且无法对开启前已分配的 RSGroup 进行 AZ 约束校验。

步骤 7 勾选欲分配的表，单击“下一步”。

说明

- 一次分配操作仅允许勾选来自同一 RSGroup 的一个或多个表。若想要分配来自不同 RSGroup 的 RegionServer 实例，请分多次修改来进行分配。
- 当修改 RSGroup 操作中同时勾选了分配 RegionServer 和表时，RegionServer 和表需来自同一 RSGroup。
- 当修改 RSGroup 操作中只勾选了分配表，且分配前该 RSGroup 下不存在 RegionServer，则将修改失败。

步骤 8 单击“提交”。修改成功后，提示修改结果，页面将跳转至 RSGroup 列表展示界面。

当提示“任务入队”相关信息时，页面将跳转至 RSGroup 列表展示界面。此次提交的修改 RSGroup 请求，已进入任务队列中，请按照界面提示，观察原生界面 region 转移完成，确认入队任务执行成功，再进行后续操作。

删除 RSGroup

步骤 9 在 RSGroup 管理页面，勾选需要删除的 RSGroup，然后选择“删除 RSGroup > 确定”。

说明

RSGroup 删除失败可能原因及解决方法：

1. “default”组不允许被删除。
2. 该 RSGroup 中仍包含 RegionServer 或 Table，请将该 RSGroup 中 RegionServer 或 Table 分配给别的 RSGroup 组后，再进行删除。

----结束

8.11 配置 HBase 容灾

操作场景

HBase 集群容灾作为提高 HBase 集群系统高可用性的一个关键特性，为 HBase 提供了实时的异地数据容灾功能。它对外提供了基础的运维工具，包含灾备关系维护，重建，数据校验，数据同步进展查看等功能。为了实现数据的实时容灾，可以把本 HBase 集群中的数据备份到另一个集群。支持 HBase 表普通写数据与 Bulkload 批量写数据场景下的容灾。

说明

本章节适用于 MRS 3.x 及之后版本。

前提条件

- 主备集群都已经安装并启动成功，且获取集群的管理员权限。
- 必须保证主备集群间的网络畅通和端口的使用。
- 如果主集群部署为安全模式且不由一个 FusionInsight Manager 管理，主备集群必须已配置跨集群互信。如果主集群部署为普通模式，不需要配置跨集群互信。
- 主备集群必须已配置跨集群拷贝。
- 主备集群上的时间必须一致，而且主备集群上的 NTP 服务必须使用同一个时间源。
- 必须在主备集群的所有节点以及主集群客户端所在节点的 hosts 文件中，配置主备集群所有机器的机器名与业务 IP 地址的对应关系。
- 主备集群间的网络带宽需要根据业务流量而定，不应少于最大的可能业务流量。
- 主备集群安装的 MRS 版本需要保持一致。
- 备集群规模不小于主集群规模。

使用约束

- 尽管容灾提供了实时的数据复制功能，但实际的数据同步进展，由多方面的因素决定的，例如，当前主集群业务的繁忙程度，备集群进程的健康状态等。因此，

在正常情形下，备集群不应该接管业务。极端情形下是否可以接管业务，可由系统维护人员以及决策人员根据当前的数据同步指标来决定。

- 容灾功能当前仅支持一主一备。
- 通常情况下，不允许对备集群的灾备表进行表级别的操作，例如修改表属性、删除表等，一旦误操作备集群后会造成主集群数据同步失败、备集群对应表的数据丢失。
- 主集群的 HBase 表已启用容灾功能同步数据，用户每次修改表的结构时，需要手动修改备集群的灾备表结构，保持与主集群表结构一致。

操作步骤

配置主集群普通写数据容灾参数。

步骤 1 登录主集群的 Manager。

步骤 2 选择“集群 > *待操作集群的名称* > 服务 > HBase > 配置”，单击“全部配置”，进入 HBase 配置界面。

步骤 3 （可选）如表 8-7 所示，为 HBase 容灾操作过程中的可选配置项，您可以根据描述来进行参数配置，或者使用缺省提供的值。

表8-7 可选配置项

配置入口	配置项	缺省值	描述
“HMaster > 性能”	hbase.master.logcleaner.ttl	600000	指定 HLog 的保存期限。如果配置值为“604800000”（单位：毫秒），表示 HLog 的保存期限为 7 天。
	hbase.master.cleaner.interval	60000	HMaster 清理过去 HLog 文件的周期，即超过设置的时间的 HLog 会被自动删除。建议尽可能配置大的值来保留更多的 HLog。
“RegionServer > Replication”	replication.source.size.capacity	16777216	edits 最大大小。单位为 byte。如果 edit 大小超过这个值 Hlog edits 将会发送到备集群。
	replication.source.nb.capacity	25000	edits 最大数目，这是另一个触发 Hlog edits 到备集群的条件。当主集群同步数据到备集群中时，主集群会从 HLog 中读取数据，此时会根据本参数配置的个数读取并发送。与“replication.source.size.capacity”一起配置使用。
	replication.source.maxretriesmultiplier	10	replication 出现异常时的最大重试次数。

配置入口	配置项	缺省值	描述
	replication.source.sleepforretries	1000	每次重试的 sleep 时间。（单位：毫秒）
	hbase.regionserver.replication.handler.count	6	RegionServer 上的 replication RPC 服务器实例数。

配置主集群 Bulkload 批量写数据容灾参数。

步骤 4 是否启用 Bulkload 批量写数据容灾功能？

是，执行步骤 5。

否，执行步骤 8。

步骤 5 选择“集群 > *待操作集群的名称* > 服务 > HBase > 配置”，单击“全部配置”，进入 HBase 配置界面。

步骤 6 搜索并修改“hbase.replication.bulkload.enabled”参数，将配置项的值修改为“true”，启用 Bulkload 批量写数据容灾功能。

步骤 7 搜索并修改“hbase.replication.cluster.id”参数，表示标识主集群 HBase 的 ID，用于备集群连接主集群。参数值支持大小写字母、数字和下划线（_），长度不超过 30。

重启 HBase 服务并安装客户端。

步骤 8 单击“保存”，保存配置。在弹出的窗口中单击“确定”。重启 HBase 服务。

步骤 9 在主备集群，选择“集群 > *待操作集群的名称* > 服务 > HBase > 更多 > 下载客户端”，下载客户端并安装。

添加主备集群容灾关系。

步骤 10 以“hbase”用户进入主集群的 HBase shell 界面。

步骤 11 在 HBase shell 中执行如下命令，创建主集群 HBase 与备集群 HBase 之间的容灾同步关系。

add_peer '*备集群 ID*'**, CLUSTER_KEY** => "*备集群 ZooKeeper 业务 ip 地址*"**, CONFIG** => **{"hbase.regionserver.kerberos.principal"** => "*备集群 RegionServer principal*"**,** **"hbase.master.kerberos.principal"** => "*备集群 HMaster principal*"**}**

- 备集群 ID 表示主集群识别备集群使用的 id，请重新指定 id 值。可以任意指定，建议使用数字。
- 备集群 ZooKeeper 地址信息包含 ZooKeeper 业务 IP 地址、侦听客户端连接的端口和备集群的 HBase 在 ZooKeeper 上的根目录。
- hbase.master.kerberos.principal 、hbase.regionserver.kerberos.principal 在备集群 HBase hbase-site.xml 配置文件中查找。

例如，添加主备集群容灾关系，执行：**add_peer** '*备集群 ID*'**, CLUSTER_KEY** => "***192.168.40.2,192.168.40.3,192.168.40.4:24002:/hbase***"**, CONFIG** => **{"hbase.regionserver.kerberos.principal"** =>

"*hbase/hadoop.hadoop.com@HADOOP.COM*", "hbase.master.kerberos.principal" => "*hbase/hadoop.hadoop.com@HADOOP.COM*"}

步骤 12 （可选）如果启用 Bulkload 批量写数据容灾功能，主集群 HBase 客户端配置必须拷贝到备集群。

- 在备集群 HDFS 创建目录/hbase/replicationConf/*主集群 hbase.replication.cluster.id*
- 主机群 HBase 客户端配置文件，拷贝到备集群 HDFS 目录/hbase/replicationConf/*主集群 hbase.replication.cluster.id*

  例如：**hdfs dfs -put HBase/hbase/conf/core-site.xml HBase/hbase/conf/hdfs-site.xml HBase/hbase/conf/yarn-site.xml hdfs://***NameNode IP***:*25000*/hbase/replicationConf/*source_cluster***

启用 HBase 容灾功能同步数据。

步骤 13 检查备集群的 HBase 服务实例中，是否已存在一个命名空间，与待启用容灾功能的 HBase 表所属的命名空间名称相同？

- 是，存在同名的命令空间，执行步骤 14。
- 否，不存在同名的命令空间，需先在备集群的 HBase shell 中，创建同名的命名空间，然后执行步骤 14。

步骤 14 在主集群的 HBase shell 中，以“hbase”用户执行以下命令，启用将主集群表的数据实时容灾功能，确保后续主集群中修改的数据能够实时同步到备集群中。

一次只能针对一个 HTable 进行数据同步。

enable_table_replication '*表名*'

说明

- 若备集群中不存在与要开启实时同步的表同名的表，则该表会自动创建。
- 若备集群中存在与要开启实时同步的表同名的表，则两个表的结构必须一致。
- 若'表名'设置了加密算法 SMS4 或 AES，则不支持对此 HBase 表启用将数据从主集群实时同步到备集群的功能。
- 若备集群不在线，或备集群中已存在同名但结构不同的表，启用容灾功能将失败。
- 若主集群中部分 Phoenix 表启用容灾功能同步数据，则备集群中不能存在与主集群 Phoenix 表同名的普通 HBase 表，否则启用容灾功能失败或影响备集群的同名表正常使用。
- 若主集群中 Phoenix 表启用容灾功能同步数据，还需要对 Phoenix 表的元数据表启用容灾功能同步数据。需配置的元数据表包含 SYSTEM.CATALOG、SYSTEM.FUNCTION、SYSTEM.SEQUENCE 和 SYSTEM.STATS。
- 若主集群的 HBase 表启用容灾功能同步数据，用户每次为 HBase 表增加新的索引，需要手动在备集群的灾备表增加二级索引，保持与主集群二级索引结构一致。
- HBase 多实例也支持容灾功能，需要修改主集群对应的 HBase 服务实例参数，且在备集群对应多实例的客户端执行命令。添加容灾关系时需要选择备集群 ZooKeeper 保存 HBase 多实例数据的目录，例如“hbase1”。

步骤 15 （可选）如果 HBase 没有使用 Ranger，在主集群的 HBase shell 中，以“hbase”用户执行以下命令，启用主集群的 HBase 表权限控制信息数据实时容灾功能。

enable_table_replication 'hbase:acl'

创建用户

步骤 16 登录备集群的 FusionInsight Manager，选择“系统 > 权限 > 角色 > 添加角色”创建一个角色，并根据主集群 HBase 源数据表的权限，为角色添加备数据表的相同权限。

步骤 17 选择“系统 > 权限 > 用户 > 添加用户”创建一个用户，根据业务需要选择用户类型为“人机”或“机机”，并将用户加入创建的角色。使用新创建的用户，访问备集群的 HBase 容灾数据。

说明

- 主集群 HBase 源数据表修改权限时，如果备集群需要正常读取数据，请修改备集群角色的权限。
- 如果当前组件使用了 Ranger 进行权限控制，须基于 Ranger 配置相关策略进行权限管理，具体操作可参考添加 HBase 的 Ranger 访问权限策略。

同步主集群表数据。

步骤 18 检查配置 HBase 容灾并启用数据同步后，主集群是否已存在表及数据，且历史数据需要同步到备集群？

- 是，存在表且需要同步数据，以 HBase 表用户登录安装主集群 HBase 客户端的节点，并执行 kinit 用户名认证身份。该用户需要拥有表的读写权限，以及“hbase:meta”表的执行权限。然后执行步骤 19。
- 否，不需要同步数据，任务结束。

步骤 19 配置 HBase 容灾时不支持自动同步表中的历史数据，需要对主集群的历史数据进行备份，然后再手动恢复历史数据到备集群中。

手动恢复即单表的恢复，单表手动恢复通过 Export、distcp、Import 来完成。

单表手动恢复操作步骤：

1. 从主集群导出表中数据。

   hbase org.apache.hadoop.hbase.mapreduce.Export -Dhbase.mapreduce.include.deleted.rows=true *表名 保存源数据的目录*

   例如，**hbase org.apache.hadoop.hbase.mapreduce.Export -Dhbase.mapreduce.include.deleted.rows=true t1 /user/hbase/t1**

2. 把导出的数据复制到备集群。

   hadoop distcp *主集群保存源数据的目录 hdfs://ActiveNameNodeIP:8020/备集群保存源数据的目录*

   其中，ActiveNameNodeIP 是备集群中主 NameNode 节点的 IP 地址。

   例如，**hadoop distcp /user/hbase/t1 hdfs://192.168.40.2:8020/user/hbase/t1**

3. 使用备集群 HBase 表用户，在备集群中导入数据。

   在备集群 HBase shell 界面，使用“hbase”用户执行以下命令保持写数据状态：

   set_clusterState_active

   界面提示以下信息表示执行成功：

```
hbase(main):001:0> set_clusterState_active
=> true
```

hbase org.apache.hadoop.hbase.mapreduce.Import *-Dimport.bulk.output=备集群保存输出的目录 表名 备集群保存源数据的目录*

hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles *备集群保存输出的目录 表名*

例如：

```
hbase(main):001:0> set_clusterState_active
=> true
```

hbase org.apache.hadoop.hbase.mapreduce.Import -Dimport.bulk.output=/user/hbase/output_t1 t1 /user/hbase/t1

hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles /user/hbase/output_t1 t1

步骤 20 在 HBase 客户端执行以下命令，校验主备集群同步的数据。启用容灾功能同步功能后，也可以执行该命令检验新的同步数据是否一致。

hbase org.apache.hadoop.hbase.mapreduce.replication.VerifyReplication --starttime=*开始时间* **--endtime**=*结束时间 列族名称 备集群 ID 表名*

说明

- 开始时间必须早于结束时间
- 开始时间和结束时间需要填写时间戳的格式，例如执行 date -d "2015-09-30 00:00:00" +%s 将普通时间转化为时间戳格式。

指定主备集群写数据状态。

步骤 21 在主集群 HBase shell 界面，使用“hbase”用户执行以下命令保持写数据状态。

set_clusterState_active

界面提示以下信息表示执行成功：

```
hbase(main):001:0> set_clusterState_active
=> true
```

步骤 22 在备集群 HBase shell 界面，使用“hbase”用户执行以下命令保持只读数据状态。

set_clusterState_standby

界面提示以下信息表示执行成功：

```
hbase(main):001:0> set_clusterState_standby
=> true
```

----结束

相关命令

表8-8 HBase 容灾

操作	命令	描述

操作	命令	描述
建立灾备关系	add_peer '*备集群 ID*', CLUSTER_KEY => "*备集群 ZooKeeper 业务 ip 地址*", CONFIG => {"hbase.regionserver.kerberos.principal" => "*备集群 RegionServer principal*", "hbase.master.kerberos.principal" => "*备集群 HMaster principal*"} **add_peer '1','zk1,zk2,zk3:2181:/hbase1'** **2181** 表示集群中 ZooKeeper 的端口号。	建立主集群与备集群的关系，让其互相对应。 如果启用 Bulkload 批量写数据容灾： • 在备集群 HDFS 创建目录 /hbase/replicationConf/*主集群 hbase.replication.cluster.id* • 主集群 HBase 客户端配置文件，拷贝到备集群 HDFS 目录 /hbase/replicationConf/*主集群 hbase.replication.cluster.id*
移除灾备关系	**remove_peer** *'备集群 ID'* 示例： **remove_peer '1'**	在主集群中移除备集群的信息。
查询灾备关系	**list_peers**	在主集群中查询已经设置的备集群的信息，主要为 Zookeeper 信息。
启用用户表实时同步	**enable_table_replication** *'表名'* 示例： **enable_table_replication 't1'**	在主集群中，设置已存在的表同步到备集群。
禁用用户表实时同步	**disable_table_replication** *'表名'* 示例： **disable_table_replication 't1'**	在主集群中，设置已存在的表不同步到备集群。
主备集群数据校验	**bin/hbase org.apache.hadoop.hbase.mapreduce.replication.VerifyReplication --starttime**=*开始时间* **--endtime=***结束时间 列族名称 备集群 ID 表名*	检查指定的表在主备集群间的数据是否一致。 命令行中参数说明如下： • 开始时间：如果未设置，则取默认的开始时间为 0。 • 结束时间：如果未设置，则取默认的结束时间为当前操作提交的时间。 • 表名：如果未输入表名，则默认校验所有的启用了实时同步的用户表。
切换数据写入状态	**set_clusterState_active** **set_clusterState_standby**	设置集群 HBase 表是否可写入数据。
新增或更新已经在对端集群	**hdfs dfs -put -f HBase/hbase/conf/core-site.xml HBase/hbase/conf/hdfs-site.xml**	启用包含 Bulkload 数据的容灾，在主集群修改 HDFS 参数时，新的参数值默认不会从主集群自动同

操作	命令	描述
保存的主集群中HDFS 配置	**HBase/hbase/conf/yarn-site.xml hdfs://**备集群 ***NameNode IP:PORT*/hbase/replicationConf/***主集群 hbase.replication.cluster.id***	步到备集群，需要手动执行命令同步。受影响的参数如下： • “fs.defaultFS” • “dfs.client.failover.proxy.provider.hacluster” • “dfs.client.failover.connection.retries.on.timeouts” • “dfs.client.failover.connection.retries” 例如，“fs.defaultFS”修改为“hdfs://hacluster_sale”， 主集群 HBase 客户端配置文件，重新拷贝到备集群 HDFS 目录 /hbase/replicationConf/*主集群 hbase.replication.cluster.id*

8.12 配置 HBase 数据压缩和编码

操作场景

HBase 可以通过对 HFile 中的 data block 编码，减少 keyvalue 中 key 的重复部分，从而减少空间的使用。目前对 data block 的编码方式有：NONE、PREFIX、DIFF、FAST_DIFF 和 ROW_INDEX_V1，其中 NONE 表示不使用编码。另外，HBase 还支持使用压缩算法对 HFile 文件进行压缩，默认支持的压缩算法有：NONE、GZ、SNAPPY 和 ZSTD，其中 NONE 表示 HFile 不压缩。

这两种方式都是作用在 HBase 的列簇上，可以同时使用，也可以单独使用。

前提条件

- 已安装 HBase 客户端。例如，客户端安装目录为“opt/client”。
- 如果 HBase 已经开启了鉴权，操作的用户还需要具备对应的操作权限。即创建表时需要具备对应的 namespace 或更高级别的创建(C)或者管理(A)权限，修改表时需要具备已创建的表或者更高级别的创建(C)或者管理(A)权限。具体的授权操作请参考创建 HBase 角色章节。

操作步骤

创建时设置 data block encoding 和压缩算法。

- **方法一：使用 hbase shell**。
  a. 以客户端安装用户，登录安装客户端的节点。

b. 执行以下命令切换到客户端目录。

cd /opt/client

c. 执行以下命令配置环境变量。

source bigdata_env

d. 如果当前集群已启用 Kerberos 认证，执行以下命令认证当前用户。如果当前集群未启用 Kerberos 认证，则无需执行此命令。

kinit *组件业务用户*

例如，**kinit hbaseuser**。

e. 直接执行 HBase 组件的客户端命令。

hbase shell

f. 创建表。

create '*t1*'**, {NAME =>** '*f1*'**, COMPRESSION =>** '*SNAPPY*'**, DATA_BLOCK_ENCODING =>** '*FAST_DIFF*'**}**

说明

- t1：表名。
- f1：列簇名。
- SNAPPY：该列簇使用的压缩算法为 SNAPPY。
- FAST_DIFF：使用的编码方式为 FAST_DIFF。
- {}内的参数为指定列簇的参数，多个列簇可以用多个{}，然后用逗号隔开。关于建表语句的更多使用说明可以在 **hbase shell** 中执行 **help 'create'** 进行查看。

- **方法二：使用 Java API**。

以下代码片段仅展示如何在建表时设置列簇的编码和压缩方式，完整的建表代码以及如何通过代码建表请参考中“HBase 开发指南 > 修改表”章节。

```
TableDescriptorBuilder htd =
TableDescriptorBuilder.newBuilder(TableName.valueOf("t1"));// 创建 t1 表的
descriptor.
ColumnFamilyDescriptorBuilder hcd =
ColumnFamilyDescriptorBuilder.newBuilder(Bytes.toBytes("f1"));//  创建列簇 f1 的
builder.
hcd.setDataBlockEncoding(DataBlockEncoding.FAST_DIFF);// 设置列簇 f1 的编码方式为
FAST_DIFF.
hcd.setCompressionType(Compression.Algorithm.SNAPPY);// 设置列簇 f1 的压缩算法为
SNAPPY
htd.setColumnFamily(hcd.build())// 将列簇 f1 添加到 t1 表的 descriptor.
```

对已存在的表设置或修改 data block encoding 和压缩算法

- **方法一：使用 hbase shell**。

a. 以客户端安装用户，登录安装客户端的节点。

b. 执行以下命令切换到客户端目录。

cd /opt/client

c. 执行以下命令配置环境变量。

source bigdata_env

d. 如果当前集群已启用 Kerberos 认证，执行以下命令认证当前用户。如果当前集群未启用 Kerberos 认证，则无需执行此命令。

kinit *组件业务用户*

例如，**kinit hbaseuser**。

e. 直接执行 HBase 组件的客户端命令。

hbase shell

f. 执行修改表的命令。

alter '*t1***', {NAME => '***f1***', COMPRESSION => '***SNAPPY***', DATA_BLOCK_ENCODING => '***FAST_DIFF***'}**

- **方法二：使用 Java API**。

以下代码片段仅展示如何修改指定表的已有列簇的编码和压缩方式，完整的修改表代码以及如何通过代码修改表请参考 HBase 应用开发指南：

```
TableDescriptor htd = admin.getDescriptor(TableName.valueOf("t1"));// 获取表 t1 的 descriptor
ColumnFamilyDescriptor originCF = htd.getColumnFamily(Bytes.toBytes("f1"));// 获取列簇 f1 的 descriptor
builder.ColumnFamilyDescriptorBuilder hcd = ColumnFamilyDescriptorBuilder.newBuilder(originCF);// 通过已有的列簇属性构造一个 builder
hcd.setDataBlockEncoding(DataBlockEncoding.FAST_DIFF);// 重新设置列簇的编码方式为 FAST_DIFF
hcd.setCompressionType(Compression.Algorithm.SNAPPY);// 重新设置列簇的压缩算法为 SNAPPY
admin.modifyColumnFamily(TableName.valueOf("t1"), hcd.build());// 提交到服务端修改列簇 f1 的属性
```

修改后完成后，已有的 HFile 的编码和压缩方式需要在下次做完 compaction 后才会生效。

8.13 HBase 容灾业务切换

操作场景

系统管理员可配置 HBase 集群容灾功能，以提高系统可用性。容灾环境中的主集群完全故障影响 HBase 上层应用连接时，需要为 HBase 上层应用配置备集群信息，才可以使得该应用在备集群上运行。

说明

本章节适用于 MRS 3.x 及之后版本。

对系统的影响

切换业务后，写入备集群的数据默认不会同步到主集群。主集群故障修复后，备集群新增的数据需要通过备份恢复的方式同步到主集群。如果需要自动同步数据，需要切换 HBase 容灾主备集群。

操作步骤

步骤 1 登录备集群 FusionInsight Manager。

步骤 2 下载并安装 HBase 客户端。

步骤 3 在备集群 HBase 客户端，以 **hbase** 用户执行以下命令指定备集群写数据状态启用。

kinit hbase

hbase shell

set_clusterState_active

界面提示以下信息表示执行成功：

```
hbase(main):001:0> set_clusterState_active
=> true
```

步骤 4 确认 HBase 上层应用中原有的配置文件“hbase-site.xml”、“core-site.xml”和“hdfs-site.xml”是否为适配应用运行修改或新增过配置内容。

- 是，将相关内容同步更新到新的配置文件中，并替换旧的配置文件。
- 否，使用新的配置文件替换 HBase 上层应用中原有的配置文件。

步骤 5 配置 HBase 上层应用所在主机与备集群的网络连接。

说明

当客户端所在主机不是集群中的节点时，配置客户端网络连接，可避免执行客户端命令时出现错误。

1. 确保客户端所在主机能与客户端安装包文件解压目录下的“hosts”文件中所列出的集群各主机在网络上互通。
2. 当客户端所在主机不是集群中的节点时，需要在客户端所在节点的“/etc/hosts”文件中设置主机名和 IP 地址（业务平面）映射。主机名和 IP 地址请保持一一对应。

步骤 6 配置 HBase 上层应用所在主机的时间与备集群的时间保持一致，时间差要小于 5 分钟。

步骤 7 检查主集群的认证模式。

- 若为安全模式，执行步骤 8。
- 若为普通模式，任务结束。

步骤 8 获取 HBase 上层应用用户的 keytab 文件和 krb5.conf 配置文件。

1. 在备集群 FusionInsight Manager 界面，选择“系统 > 权限 > 用户”。
2. 在用户所在行的“操作”列单击“更多 > 下载认证凭据”，下载 keytab 文件到本地。
3. 解压得到“user.keytab”和“krb5.conf”。

步骤 9 使用“user.keytab”和“krb5.conf”两个文件替换 HBase 上层应用中原有的文件。

步骤 10 停止上层业务。

步骤 11 是否需要切换 HBase 主备集群，即主变成备，备变成主。如果不切换，数据将不再同步。

- 是，先执行 HBase 容灾主备集群倒换，具体请参考 HBase 容灾主备集群倒换，然后再执行步骤 12。
- 否，直接执行步骤 12。

步骤 12 启动上层业务。

----结束

8.14 HBase 容灾主备集群倒换

操作场景

当前环境 HBase 已经是容灾集群，因为某些原因，需要将主备集群互换，即备集群变成主集群，主集群变成备集群。

说明

本章节适用于 MRS 3.x 及之后版本。

对系统的影响

主备集群互换后，原先主集群将不能再写入数据，原先备集群将变成主集群，接管上层业务。

操作步骤

确保上层业务已经停止

步骤 1 确保上层业务已经停止，如果没有停止，先执行 参考 HBase 容灾业务切换。

关闭主集群写功能

步骤 2 下载并安装 HBase 客户端。

步骤 3 在备集群 HBase 客户端，以 **hbase** 用户执行以下命令指定备集群写数据状态关闭。

kinit hbase

hbase shell

set_clusterState_standby

界面提示以下信息表示执行成功：

```
hbase(main):001:0> set_clusterState_standby
=> true
```

检查当前主备同步是否完成

步骤 4 执行以下命令，确保当前数据已经同步，要求 SizeOfLogQueue=0，SizeOfLogToReplicate=0，如果不为零，等待，重复执行以下命令，直到等于 0。

status 'replication'

关闭主备集群同步

步骤 5 查询所有的同步集群，获取 PEER_ID。

list_peers

步骤 6 删除所有同步集群。

remove_peer '备集群 ID'

示例：

remove_peer *'1'*

步骤 7 查询所有同步的 table。

list_replicated_tables

步骤 8 分别 disable 上面查询到的所有同步的 table。

disable_table_replication *'表名'*

示例：

disable_table_replication *'t1'*

切换主备

步骤 9 重新配置 HBase 容灾，参考配置 HBase 容灾。

----结束

8.15 社区 BulkLoad Tool

Apache HBase 官方网站提供了批量导入数据的功能，详细操作请参见官网对“Import”和“ImportTsv”工具的描述：http://hbase.apache.org/2.2/book.html#tools。

8.16 自研增强 BulkLoad Tool

8.16.1 按自定义方式导入数据

8.16.1.1 批量导入数据

操作场景

您可以按照自定义的方式，通过命令批量导入数据到 HBase 中。

您可以在“configuration.xml”文件中定义多个方式来批量导入数据。导入数据时可不创建索引。

说明

- 列的名称不能包含特殊字符，只能由字母、数字和下划线组成。
- 大任务下 MR 任务运行失败，请参考 MapReduce 任务运行失败，ApplicationMaster 出现物理内存溢出异常进行处理。
- BulkLoad 支持的数据源格式为带分隔符的文本文件。
- 已安装客户端。例如安装目录为“/opt/hadoopclient”，以下操作的客户端目录只是举例，请根据实际安装目录修改。

操作步骤

步骤 1 以客户端安装用户，登录安装客户端的节点。

步骤 2 执行以下命令切换到客户端目录。

cd /opt/hadoopclient

步骤 3 执行以下命令配置环境变量。

source bigdata_env

步骤 4 若安装了 HBase 多实例，在使用客户端连接具体 HBase 实例时，请执行以下命令加载具体实例的环境变量，否则请跳过此步骤。例如，加载 HBase2 实例变量：

source HBase2/component_env

步骤 5 如果当前集群已启用 Kerberos 认证，执行以下命令认证当前用户，当前用户需要具有创建 HBase 表的权限和 HDFS 的操作权限：

kinit *组件业务用户*

如果当前集群未启用 Kerberos 认证，则执行以下命令设置 Hadoop 用户名：

export HADOOP_USER_NAME=hbase

步骤 6 将数据导入到 HDFS 中。

hdfs dfs -mkdir <*inputdir*>

hdfs dfs -put <local_data_file> <*inputdir*>

例如定义数据文件“data.txt”，内容如下：

```
001,Hadoop,shenzhen
002,HBaseFS,longgang
003,HBase,bantian
004,Hive,longgang
005,Streaming,shenzhen
006,MapReduce,shenzhen
007,Kerberos,bantian
008,LdapServer,longgang
```

执行以下命令：

hdfs dfs -mkdir /datadirImport

hdfs dfs -put data.txt /datadirImport

步骤 7 进入 hbase shell，建表 ImportTable 并创建“configuration.xml”文件（该文件可以参考模板文件进行编辑，模板文件获取路径为：“/opt/client/HBase/hbase/conf/import.xml.template”）。

例如执行以下命令建表：

create 'ImportTable', {NAME => 'f1',COMPRESSION => 'SNAPPY', DATA_BLOCK_ENCODING => 'FAST_DIFF'},{NAME=>'f2'}

例如自定义导入模板文件 configuration.xml：

说明

- column_num 要和数据文件中的列的数量对应。
- family 的指定要和表的列族名称对应。

```
<?xml version="1.0" encoding="UTF-8"?>

 <configuration>
       <import id="first" column_num="3">
              <columns>
                     <column index="1" type="int">SMS_ID</column>
                     <column index="2" type="string">SMS_NAME</column>
                     <column index="3" type="string">SMS_ADDRESS</column>
              </columns>

              <rowkey>
                     SMS_ID+'_'+substring(SMS_NAME,1,4)+'_'+reverse(SMS_ADDRESS)
              </rowkey>

              <qualifiers>
                     <normal family="f1">
                           <qualifier column="SMS_ID">H_ID</qualifier>
                           <qualifier column="SMS_NAME">H_NAME</qualifier>
                           <qualifier column="SMS_ADDRESS">H_ADDRESS</qualifier>
                     </normal>

                     <!-- Define composite columns -->
                     <composite family="f2">
                           <qualifier
class="com.xxx.H_COMBINE_1">H_COMBINE_1</qualifier>
                           <columns>
                                 <column>SMS_ADDRESS</column>
                                 <column>SMS_NAME</column>
                           </columns>
                     </composite>

              </qualifiers>
              <badlines>SMS_ID &lt; 7000 &amp;&amp; SMS_NAME == 'HBase'</badlines>
       </import>
 </configuration>
```

步骤 8 执行如下命令，生成 HFile 文件。

hbase com.xxx.hadoop.hbase.tools.bulkload.ImportData -Dimport.skip.bad.lines=*true* **-Dimport.separator**=<*separator*> **-Dimport.bad.lines.output**=*</path/badlines/output>* **-Dimport.hfile.output=***</path/for/output>* <*configuration xmlfile*> <*tablename*> <*inputdir*>

- -Dimport.skip.bad.lines：指定值为 false，表示遇到不适用的行则停止执行。指定值为 true，表示遇到不适用的数据行则跳过该行继续执行，如果没有在 configuration.xml 中定义不适用行，该参数不需要添加。
- -Dimport.separator：分隔符，例如，-Dimport.separator=','。
- -Dimport.bad.lines.output=</path/badlines/output>：指的是不适用的数据行输出路径，如果没有在 configuration.xml 中定义不适用行，该参数不需要添加。
- -Dimport.hfile.output=< /path/for/output>：指的是执行结果输出路径。
- <configuration xmlfile>：指向 configuration 配置文件。
- <tablename>：指的是要操作的表名。
- <inputdir>：指的是要批量上传的数据目录。

例如执行以下命令：

hbase com.xxx.hadoop.hbase.tools.bulkload.ImportData -Dimport.skip.bad.lines=true -Dimport.separator=',' -Dimport.bad.lines.output=/badline -Dimport.hfile.output=/hfile configuration.xml ImportTable /datadirImport

须知

- 当 HBase 已经配置透明加密后，在执行 bulkload 命令生成 HFile 时，“-Dimport.hfile.output”指定的 HFile 路径必须为“/*HBase 根目录*/extdata”的子目录，例如“/hbase/extdata/bulkloadTmp/hfile”。
- 当 HBase 已经配置透明加密后，执行 bulkload 命令的 HBase 用户需要添加到对应集群的 hadoop 用户组（非 FusionInsight Manager 下第一个安装的集群，用户组为“c<*集群 ID*>_hadoop”，例如“c2_hadoop”），且具有 HBase 根目录的加密 key 的读权限。
- 检查目录/tmp/hbase 的权限，需要手动添加当前用户对该目录的写权限。

步骤 9 执行如下命令将 HFile 导入 HBase。

hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles *</path/for/output> <tablename>*

例如执行以下命令：

hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles /hfile ImportTable

----结束

8.16.1.2 组合 rowkey

操作场景

支持用户自定义的组合 rowkey。BulkLoad 组合 rowkey 即通过一些规则将多个列名经过一些自定义处理，组合生成新的 rowkey。

说明

列的名称不能包含特殊字符，只能由字母、数字和下划线组成。

配置方法

关于组合 rowkey 在“configuration.xml”文件中的配置如下所示。

样例：定义组合 rowkey 为列“SMS_ID”、“SMS_NAME”的取第二个字符开始的三个字符以及“SMS_SERAIL”的反转（各部分用'_'连接）。

```
<columns>
                    <column index="1" type="int">SMS_ID</column>
                    <column index="2" type="string">SMS_NAME</column>
                    <column index="3" type="string">SMS_ADDRESS</column>
              </columns>

              <rowkey>
                    SMS_ID+'_'+substring(SMS_NAME,1,4)+'_'+reverse(SMS_ADDRESS)
              </rowkey>
```

表8-9 rowkey 字段处理函数

函数原型	描述	示例
format(data,"DataType")	格式化字符串数据。	例如，format(data,"0.000")是指将数据按照"0.000"格式输出。
converse(data,"yyyy-MM-dd","yyyyMMdd")	转化日期格式。	例如，converse(data,"yyyy-MM-dd","yyyyMMdd")是指将日期格式从"yyyy-MM-dd"转化为"yyyyMMdd"。
rand	随机一个整数，只支持 int 类型。	无
replace(data,"A","B")	数据替换。	例如，replace(data,"A","B")是指将 A 用 B 替换。
reverse(data)	将字符串反转。	例如，reverse(ABC)将"ABC"反转成"CBA"。
substring(data,Length1,Length2), or substring(data,Length3)	截取字符串。	例如，substring(data,1,5), or substring(data,3)是指将 data 字符串进行截取[1,5)或[3,data.length)。
to_number("data")	将字符串转化成数值型，支持返回 Long 类型。	例如，to_number("123")是指将"123"转化为 123，注意当前 data 必须为数值。

8.16.1.3 自定义 rowkey 实现

操作场景

支持用户自定义的组合 rowkey 实现。用户可编写 rowkey 实现代码，导入时根据该代码逻辑进行组合 rowkey 导入。

配置方法

步骤 1 用户编写自定义 rowkey 的实现类，需要继承接口：

```
[com.xxx.hadoop.hbase.tools.bulkload.RowkeyHandlerInterface],
```

实现接口中方法：

```
byte[] getRowkeyBytes(String[] colsValues, RegulationDomain regulation)
```

其中：

- 传入参数“colsValues”为原始数据中的一行数据集合，每个元素为一列。
- 传入参数“regulation”为配置导入文件信息（一般情况下并不需要使用）。

步骤 2 将该实现类与其依赖包同时打包成 jar 文件，保存到 HBase 客户端任意位置。

步骤 3 在执行导入命令时，增加两个参数配置项：

-Dimport.rowkey.jar="*第二步中 jar 包的全路径*"

-Dimport.rowkey.class="*用户实现类的全类名*"

----结束

8.16.1.4 组合字段

功能介绍

BulkLoad 支持自定义组合字段，把多个列通过追加的方式即多个列串到一块组合成一个列。

说明

列的名称不能包含特殊字符，只能由字母、数字和下划线组成。

配置方法

关于组合字段 H_COMBINE_1 的定义如下所示。

样例：H_COMBINE_1 由字段“SMS_ADDRESS”、“SMS_SNAME”构成。

```
<!-- Define composite columns -->
                    <composite family="f2">
                          <!-- 定义拼接字段的类名，且该类必须在客户应用中不存在 -->
                          <qualifier
```

```
class="com.xxx.H_COMBINE_1">H_COMBINE_1</qualifier>
                            <columns>
                                  <column>SMS_ADDRESS</column>
                                  <column>SMS_NAME</column>
                            </columns>
                      </composite>
```

8.16.1.5 指定字段数据类型

功能介绍

HBase BulkLoad 支持读取原生态数据文件，把数据文件的每个字段映射为 HBase 定义的字段，并对该字段的数据类型做定义。

您可以在“configuration.xml”文件中定义多个方式来批量导入数据。

说明

列的名称不能包含特殊字符，只能由字母、数字和下划线组成。

配置方法

指定字段数据类型的配置如下所示。

样例：对“SMS_ID”、“SMS_NAME”、“SMS_ADDRESS”列指定数据类型。

```
            <columns>
                  <column index="1" type="int">SMS_ID</column>
                  <column index="2" type="string">SMS_NAME</column>
                  <column index="3" type="string">SMS_ADDRESS</column>
             </columns>
```

说明

支持的数据类型有：short、int、long、float、double、boolean 和 string。

8.16.1.6 定义不适用的数据行

操作场景

BulkLoad 支持定义不适用数据行的功能，不适用数据行不会存储到 HBase 中，这些数据会被保存到指定的文件中。

您可以在“configuration.xml”文件中定义多个方式来批量导入数据。

说明

列的名称不能包含特殊字符，只能由字母、数字和下划线组成。

操作步骤

定义不适用的行，配置样例如下所示。

```
<!-- Define bad line filter rule -->
<badlines>SMS_ID &lt; 7000 &amp;&amp; SMS_NAME == 'HBase'</badlines>
```

说明

SMS_ID < 7000 && SMS_NAME == 'HBase'

针对“<badlines>”标签中的算符和对应的参数类型如表 8-10 所示。

表8-10 算符和对应的参数类型

算符类型	参数类型
&&	对应的参数类型应为布尔型。
&	对应的参数类型应为整数。
|	对应的参数类型应为整数。
^	对应的参数类型应为整数。
/	对应的参数类型应为数字。
==	对应的参数类型应为字符串。
>=	对应的参数类型应为数字。
>	对应的参数类型应为数字。
<<	对应的参数类型应为整数。
<=	对应的参数类型应为数字。
<	对应的参数类型应为数字。
%	对应的参数类型应为数字。
*	对应的参数类型应为数字。
!=	对应的参数类型应为字符串。
||	对应的参数类型应为布尔型。
+	对应的参数类型应为数字和字符串。
>>	对应的参数类型应为整数。
-	对应的参数类型应为字符串。
>>>	对应的参数类型应为整数。

8.16.2 按自定义方式导入带有索引的数据

8.16.2.1 批量导入数据时创建二级索引

操作场景

您可以按照自定义的方式，通过命令批量导入数据到 HBase 中。

说明

- 列的名称不能包含特殊字符，只能由字母、数字和下划线组成。
- 当将列的类型设置为 string 时，不能设置其长度。例如“<column index="1" type="string" **length="1"** >COLOUMN_1</column>”，此类型不支持。
- 当将列的类型设置为 date 时，不能设置其日期格式。例如“<column index="13" type="date" **format="yyyy-MM-dd hh:mm:ss"**>COLOUMN_13</column>”，此类型不支持。
- 不能针对组合列建立二级索引。

操作步骤

步骤 1 将数据导入到 HDFS 中。

hdfs dfs -mkdir <*inputdir*>

hdfs dfs -put <local_data_file> <*inputdir*>

例如定义数据文件“data.txt”，内容如下：

```
001,Hadoop,shenzhen
002,HBaseFS,longgang
003,HBase,bantian
004,Hive,longgang
005,Streaming,shenzheng
006,Mapreduce,shenzheng
007,Kerberos,bantian
008,LdapServer,longgang
```

执行以下命令：

hdfs dfs -mkdir /datadirIndexImport

hdfs dfs -put data.txt /datadirIndexImport

步骤 2 建表 IndexImportTable 并创建“configuration_index.xml”文件（该文件可以参考模板文件进行编辑，模板文件获取路径为：“${client path}/HBase/hbase/conf/index_import.xml.template”）。

例如执行以下命令建表：

create 'IndexImportTable', {NAME => 'f1',COMPRESSION => 'SNAPPY', DATA_BLOCK_ENCODING => 'FAST_DIFF'},{NAME=>'f2'}

例如自定义导入模板文件 configuration_index.xml：

说明

- column_num 要和数据文件中的列的数量对应。
- family 的指定要和表的列族名称对应。
- 索引类型的首字母需要大写，例如“type”=“String”。

```
<?xml version="1.0" encoding="UTF-8"?>

 <configuration>
       <import id="first" column_num="3">
```

```
            <columns>
                  <column index="1" type="int">SMS_ID</column>
                  <column index="2" type="string">SMS_NAME</column>
                  <column index="3" type="string">SMS_ADDRESS</column>
            </columns>

            <rowkey>
                  SMS_ID+'_'+substring(SMS_NAME,1,4)+'_'+reverse(SMS_ADDRESS)
            </rowkey>

            <qualifiers>
                  <normal family="f1">
                        <qualifier column="SMS_ID">H_ID</qualifier>
                        <qualifier column="SMS_NAME">H_NAME</qualifier>
                        <qualifier column="SMS_ADDRESS">H_ADDRESS</qualifier>
                  </normal>

                  <!-- Define composite columns -->
                  <composite family="f2">
                        <qualifier
class="com.xxx.H_COMBINE_1">H_COMBINE_1</qualifier>
                        <columns>
                              <column>SMS_ADDRESS</column>
                              <column>SMS_NAME</column>
                        </columns>
                  </composite>

            </qualifiers>

               <indices>
                  <index name="IDX1">
                        <index_column family="f1">
                              <qualifier type="String"
length="30">H_ID</qualifier>
                        </index_column>
                  </index>
            </indices>

            <badlines>SMS_ID &lt; 7000 &amp;&amp; SMS_NAME == 'HBase'</badlines>
      </import>
</configuration>
```

说明

上述片段信息中“length="30"”指索引列“H_ID”的列值不能超过 30 个字符。

步骤 3 执行如下命令，生成 HFile 文件。

hbase com.xxx.hadoop.hbase.tools.bulkload.IndexImportData -Dimport.skip.bad.lines=*true* **-Dimport.separator**=<*separator*> **-Dimport.bad.lines.output**=</*path/badlines/output*> **-Dimport.hfile.output=**</*path/for/output*> <*configuration xmlfile*> <*tablename*> <*inputdir*>

- -Dimport.skip.bad.lines：指定值为 false，表示遇到不适用的行则停止执行。指定值为 true，表示遇到不适用的数据行则跳过该行继续执行，如果没有在 configuration.xml 中定义不适用行，该参数不需要添加。

- -Dimport.separator：分隔符。例如，-Dimport.separator=','
- -Dimport.bad.lines.output=</path/badlines/output>：指的是不适用的数据行输出路径，如果没有在 configuration.xml 中定义不适用行，该参数不需要添加。
- -Dimport.hfile.output=< /path/for/output>：指的是执行结果输出路径。
- <configuration xmlfile>：指向 configuration 配置文件。
- <tablename>：指的是要操作的表名。
- <inputdir>：指的是要批量上传的数据目录。

例如执行以下命令：

hbase com.xxx.hadoop.hbase.tools.bulkload.IndexImportData -Dimport.skip.bad.lines=true -Dimport.separator=',' -Dimport.bad.lines.output=/badline -Dimport.hfile.output=/hfile configuration_index.xml IndexImportTable /datadirIndexImport

须知

- 当 HBase 已经配置透明加密后，在执行 bulkload 命令生成 HFile 时，“-Dimport.hfile.output”指定的 HFile 路径必须为“/*HBase 根目录*/extdata”的子目录，例如“/hbase/extdata/bulkloadTmp/hfile”。
- 当 HBase 已经配置透明加密后，执行 bulkload 命令的 HBase 用户需要添加到对应集群的 hadoop 用户组（非 FusionInsight Manager 下第一个安装的集群，用户组为“c<*集群 ID*>_hadoop”，例如“c2_hadoop”），且具有 HBase 根目录的加密 key 的读权限。

步骤 4 执行如下命令将 HFile 导入 HBase。

hbase org.apache.hadoop.hbase.hindex.mapreduce.HIndexLoadIncrementalHFiles *</path/for/output> <tablename>*

例如执行以下命令：

hbase org.apache.hadoop.hbase.hindex.mapreduce.HIndexLoadIncrementalHFiles /hfile IndexImportTable

----结束

8.16.2.2 组合 rowkey

操作场景

支持用户自定义的组合 rowkey。BulkLoad 组合 rowkey 即通过一些规则将多个列名经过一些自定义处理，组合生成新的 rowkey。

说明

列的名称不能包含特殊字符，只能由字母、数字和下划线组成。

配置方法

关于组合 rowkey 在“configuration.xml”文件中的配置如下所示。

样例：定义组合 rowkey 为列“SMS_ID”、“SMS_NAME”的取第二个字符开始的三个字符以及“SMS_SERAIL”的反转（各部分用'_'连接）。

```
<columns>
                    <column index="1" type="int">SMS ID</column>
                    <column index="2" type="string">SMS NAME</column>
                    <column index="3" type="string">SMS_ADDRESS</column>
              </columns>

              <rowkey>
                    SMS_ID+'_'+substring(SMS_NAME,1,4)+'_'+reverse(SMS_ADDRESS)
              </rowkey>
```

表8-11 rowkey 字段处理函数

函数原型	描述	示例
format(data,"DataType")	格式化字符串数据。	例如，format(data,"0.000")是指将数据按照"0.000"格式输出。
converse(data,"yyyy-MM-dd","yyyyMMdd")	转化日期格式。	例如，converse(data,"yyyy-MM-dd","yyyyMMdd")是指将日期格式从"yyyy-MM-dd"转化为"yyyyMMdd"。
rand	随机一个整数，只支持 int 类型。	无
replace(data,"A","B")	数据替换。	例如，replace(data,"A","B")是指将 A 用 B 替换。
reverse(data)	将字符串反转。	例如，reverse(ABC)将"ABC"反转成"CBA"。
substring(data,Length1,Length2), or substring(data,Length3)	截取字符串。	例如，substring(data,1,5), or substring(data,3)是指将 data 字符串进行截取[1,5)或[3,data.length)。
to_number("data")	将字符串转化成数值型，支持返回 Long 类型。	例如，to_number("123")是指将"123"转化为 123，注意当前 data 必须为数值。

8.16.2.3 自定义 rowkey 实现

操作场景

支持用户自定义的组合 rowkey 实现。用户可编写 rowkey 实现代码，导入时根据该代码逻辑进行组合 rowkey 导入。

配置方法

步骤 1 用户编写自定义 rowkey 的实现类，需要继承接口：

```
[com.xxx.hadoop.hbase.tools.bulkload.RowkeyHandlerInterface]
```

实现接口中方法：

```
byte[] getRowkeyBytes(String[] colsValues, RegulationDomain regulation)
```

其中：

- 传入参数“colsValues”为原始数据中的一行数据集合，每个元素为一列。
- 传入参数“regulation”为配置导入文件信息（一般情况下并不需要使用）。

步骤 2 将该实现类与其依赖包同时打包成 jar 文件，保存到 HBase 客户端任意位置。

步骤 3 在执行导入命令时，增加两个参数配置项：

-Dimport.rowkey.jar="*第二步中 jar 包的全路径*"

-Dimport.rowkey.class="*用户实现类的全类名*"

----结束

8.16.2.4 组合字段

功能介绍

BulkLoad 支持自定义组合字段，把多个列通过追加的方式即多个列串到一块组合成一个列。

说明

列的名称不能包含特殊字符，只能由字母、数字和下划线组成。

配置方法

关于组合字段 H_COMBINE_1 的定义如下所示。

样例：H_COMBINE_1 由字段“SMS_ADDRESS”、“SMS_SNAME”构成。

```
<!-- Define composite columns -->
                    <composite family="f2">
                          <!-- 定义拼接字段的类名，且该类必须在客户应用中不存在 -->
                          <qualifier
class="com.xxx.H_COMBINE_1">H_COMBINE_1</qualifier>
```

```
                    <columns>
                        <column>SMS_ADDRESS</column>
                        <column>SMS_NAME</column>
                    </columns>
                </composite>
```

8.16.2.5 指定字段数据类型

功能介绍

HBase BulkLoad 支持读取原生态数据文件，把数据文件的每个字段映射为 HBase 定义的字段，并对该字段的数据类型做定义。

您可以在“configuration.xml”文件中定义多个方式来批量导入数据。

说明

列的名称不能包含特殊字符，只能由字母、数字和下划线组成。

配置方法

指定字段数据类型的配置如下所示。

样例：对“SMS_ID”、“SMS_NAME”、“SMS_ADDRESS”列指定数据类型。

```
<columns>
        <column index="1" type="int">SMS_ID</column>
        <column index="2" type="string">SMS_NAME</column>
        <column index="3" type="string">SMS_ADDRESS</column>
 </columns>
```

说明

支持的数据类型有：short、int、long、float、double、boolean 和 string。

8.16.2.6 定义不适用的数据行

操作场景

BulkLoad 支持定义不适用数据行的功能，不适用数据行不会存储到 HBase 中，这些数据会被保存到指定的文件中。

您可以在“configuration.xml”文件中定义多个方式来批量导入数据。

说明

列的名称不能包含特殊字符，只能由字母、数字和下划线组成。

操作步骤

定义不适用的行，配置样例如下所示。

```
<!-- Define bad line filter rule -->
<badlines>SMS_ID &lt; 7000 &amp;&amp; SMS_NAME == 'HBase'</badlines>
```

说明

SMS_ID < 7000 && SMS_NAME == 'HBase'

针对“<badlines>”标签中的算符和对应的参数类型如表 8-12 所示。

表8-12 算符和对应的参数类型

算符类型	参数类型
&&	对应的参数类型应为布尔型。
&	对应的参数类型应为整数。
|	对应的参数类型应为整数。
^	对应的参数类型应为整数。
/	对应的参数类型应为数字。
==	对应的参数类型应为字符串。
>=	对应的参数类型应为数字。
>	对应的参数类型应为数字。
<<	对应的参数类型应为整数。
<=	对应的参数类型应为数字。
<	对应的参数类型应为数字。
%	对应的参数类型应为数字。
*	对应的参数类型应为数字。
!=	对应的参数类型应为字符串。
||	对应的参数类型应为布尔型。
+	对应的参数类型应为数字和字符串。
>>	对应的参数类型应为整数。
-	对应的参数类型应为字符串。
>>>	对应的参数类型应为整数。

8.16.3 批量更新

操作场景

支持根据 RowKey 的命名规则、RowKey 的范围、字段名以及字段值进行批量更新。

操作步骤

执行如下命令更新从“row_start”到“row_stop”的行，并且把输出结果定向到“/output/destdir/”。

```
hbase com.xxx.hadoop.hbase.tools.bulkload.UpdateData
  -Dupdate.rowkey.start="row_start"
  -Dupdate.rowkey.stop="row_stop"
  -Dupdate.hfile.output=/user/output/
  -Dupdate.qualifier=f1:c1,f2
  -Dupdate.qualifier.new.value=0,a
  'table1'
```

- -Dupdate.rowkey.start="row_start"：表示开始行号为 row_start。
- -Dupdate.rowkey.stop="row_stop"：表示结束行号为 row_stop。
- -Dupdate.hfile.output=/user/output/：表示执行结果输出路径为/user/output/。

须知

当 HBase 已经配置透明加密后，“批量更新”操作注意事项请参考步骤 8。

执行以下命令，加载 HFiles：

hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles *<path/for/output>* *<tablename>*

注意事项

1. 批量更新会把满足条件的行对应的字段值替换为要更新的值。
2. 如果要更新的字段上建有索引，批量更新是不允许的。
3. 如果不设置执行结果输出文件，默认是（/tmp/updatedata/表名）。

8.16.4 批量删除

操作场景

根据 rowkey 的取值模式、范围、字段名、字段值对 HBase 做批量删除。

操作步骤

执行如下命令删除从“row_start”到“row_stop”的行，并且把输出结果定向到“/output/destdir/”。

```
hbase com.xxx.hadoop.hbase.tools.bulkload.DeleteData
  -Ddelete.rowkey.start="row_start"
  -Ddelete.rowkey.stop="row_stop"
  -Ddelete.hfile.output="/output/destdir/"
  -Ddelete.qualifier="cf1,cf0:vch,cf0:lng:1000"
  'table1'
```

- -Ddelete.rowkey.start="row_start"：表示开始行号为 row_start。

- -Ddelete.rowkey.stop="row_stop"：表示结束行号为 row_stop。
- -Ddelete.hfile.output="/output/destdir/"：表示执行结果输出到/output/destdir/目录下。
- -Ddelete.qualifier="cf1,cf0:vch,cf0:lng:1000"：表示删除 column family cf1 中所有列，column family cf0 中列为 vch 的列，column family cf0 中列 lng 中值为 1000 的列。

须知

当 HBase 已经配置透明加密后，“批量删除”操作注意事项请参考步骤 8。

执行以下命令，加载 HFiles。

hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles *<path/for/output> <tablename>*

注意事项

1. 如果 column qualifier 上建有索引，在该字段的批量删除是会失败的，即不允许在建有索引的字段上执行批量删除。
2. 如果不设置执行结果输出数据文件（delete.hfile.output），默认是/tmp/deletedata/表名。

8.16.5 获取行统计数

操作场景

支持根据 rowkey 的命名规则、rowkey 的范围、字段名以及字段值统计符合条件的行数。

操作步骤

直接执行如下命令统计满足如下条件的行数。rowkey 在从“row_start”到“row_stop”的范围，字段“f3:age”的值为“25”，rowkey 的前两个字符为“mi”的行数。

```
hbase com.xxx.hadoop.hbase.tools.bulkload.RowCounter -
Dcounter.rowkey.start="row_start" -Dcounter.rowkey.stop="row_stop" -
Dcounter.qualifier="f3:age:25" -Dcounter.rowkey.value="substring(0,2) == 'mi'"
table1
```

- -Dcounter.rowkey.start="row_start"：表示开始的行号为"row_start"。
- -Dcounter.rowkey.stop="row_stop"：表示结束的行号为"row_stop"。
- -Dcounter.qualifier="f3:age:25"：表示列族 f3 中列为 age 的列值为 25。
- -Dcounter.rowkey.value="substring(0,2) == 'mi'"：表示 rowkey 的值中前两个为 mi。

8.17 配置 MOB

配置场景

在实际应用中，需要存储大大小小的数据，比如图像数据、文档。小于 10MB 的数据一般都可以存储在 HBase 上，对于小于 100KB 的数据，HBase 的读写性能是最优的。如果存放在 HBase 的数据大于 100KB 甚至到 10MB 大小时，插入同样个数的数据文件，但是总的数据量会很大，会导致频繁的 compaction 和 split，占用很多 CPU，磁盘 IO 频率很高，性能严重下降。

通过将 MOB（Medium-sized Objects）数据（即 100KB 到 10MB 大小的数据）直接以 HFile 的格式存储在文件系统上（例如 HDFS 文件系统），通过 expiredMobFileCleaner 和 Sweeper 工具集中管理这些文件，然后把这些文件的地址信息及大小信息作为 value 存储在普通 HBase 的 store 上。这样就可以大大降低 HBase 的 compaction 和 split 频率，提升性能。

HBase 当前默认开启 MOB 功能，相关配置项如表 8-13 所示。如果需要使用 MOB 功能，用户需要在创建表或者修改表属性时在指定的列族上指定使用 mob 方式存储数据。

说明

本章节适用于 MRS 3.x 及之后版本。

配置描述

为了开启 HBase MOB 功能，用户需要在创建表或者修改表属性时在指定的列族上指定使用 mob 方式存储数据。

使用代码声明使用 mob 存储的方式：

```
HColumnDescriptor hcd = new HColumnDescriptor("f");
hcd.setMobEnabled(true);
```

使用 shell 声明使用 mob 的方式，MOB_THRESHOLD 单位是字节：

```
hbase(main):009:0> create 't3',{NAME => 'd', MOB_THRESHOLD => '102400', IS_MOB =>
'true'}

0 row(s) in 0.3450 seconds

=> Hbase::Table - t3
hbase(main):010:0> describe 't3'
Table t3 is ENABLED


t3


COLUMN FAMILIES DESCRIPTION


{NAME => 'd', MOB_THRESHOLD => '102400', VERSIONS => '1', KEEP_DELETED_CELLS =>
'FALSE', DATA_BLOCK_ENCODING => 'NONE',
```

```
TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE => '0', BLOOMFILTER =>
'ROW',
IN_MEMORY => 'false', IS_MOB => 'true', COMPRESSION => 'NONE', BLOCKCACHE => 'true',
BLOCKSIZE => '65536'}

1 row(s) in 0.0170 seconds
```

参数入口：

在 FusionInsight Manager 系统中，选择“集群 > *待操作集群的名称* > 服务 > HBase > 配置”，单击“全部配置”。在搜索框中输入参数名称。

表8-13 参数描述

参数	描述	默认值
hbase.mob.file.cache.size	已经打开的文件句柄的缓存区大小。如果该值设置的比较大，cache 可以缓存更多的文件句柄，从而降低打开关闭文件的频率。但是如果该值设置过大会导致打开的文件句柄数过多。默认值是：“1000”。此参数在服务端 ResionServer 上配置。	1000
hbase.mob.cache.evict.period	缓存 mob 文件在 mob 缓存中的超期时间，单位为秒。	3600
hbase.mob.cache.evict.remain.ratio	mob cache 回收之后保留的文件个数占 cache 容量个数的比例。hbase.mob.cache.evict.remain.ratio 是一个算法因子，当缓存 mob 文件数达到 hbase.mob.file.cache.size*hbase.mob.cache.evict.remain.ratio 的大小后触发缓存回收。	0.5
hbase.master.mob.ttl.cleaner.period	过期文件清理任务的运行周期，以秒为单位。默认值是一天(86400 秒)。 说明 如果生存时间值过期了，即文件从创建起已经超过了 24 小时，则 MOB 文件将会被过期 mob 文件清理工具删除。	86400

8.18 配置安全的 HBase Replication

配置场景

安全模式下，在交叉域设置 Kerberos 时，配置安全的 HBase replication 的过程。

前提条件

- 在 Kerberos 配置文件中必须定义所有 FQDN 映射到它的域。
- ONE.COM 和 TWO.COM 的密码和 keytab 必须要一样。

操作步骤

步骤 1 为两个域创建 krbtgt 帐户名。

比如，有 ONE.COM 和 TWO.COM 两个域，需要添加如下帐户名：krbtgt/ONE.COM@TWO.COM 及 krbtgt/TWO.COM@ONE.COM。

在两个域中均添加这两个帐户名。

```
kadmin: addprinc -e "<enc_type_list>" krbtgt/ONE.COM@TWO.COM
kadmin: addprinc -e "<enc_type_list>" krbtgt/TWO.COM@ONE.COM
```

说明

在这两个域之间必须至少有一个共同的 keytab 模式。

步骤 2 在 Zookeeper 中，为创建短名称添加规则。

Dzookeeper.security.auth_to_local 是 Zookeeper 服务器进程的参数。以下例子说明了如何支持 ONE.COM，在帐户名中有两个成员（如 service/instance@ONE.COM）。

```
Dzookeeper.security.auth_to_local=RULE:[2:\$1@\$0](.*@\\QONE.COM\\E$)s/@\\QONE.COM\
\E$//DEFAULT
```

以上代码案例为在不同的域中支持 ONE.COM。因此在 replication 中，需要为在从属集群域的主集群域添加规则。DEFAULT 是已经添加了默认规则。

步骤 3 在 Hadoop 进程中，为创建短名称添加规则。

在从属集群的 HBase 进程中的“core-site.xml”配置文件的属性 hadoop.security.auth_to_local。比如：支持 ONE.COM：

```
<property>
<name>hadoop.security.auth_to_local</name>
<value>RULE:[2:$1@$0](.*@\QONE.COM\E$)s/@\QONE.COM\E$//DEFAULT</value>
</property>
```

说明

如果启用 bulkload replication 功能，那么在主集群 HBase 进程的配置文件“core-site.xml”中需要添加支持从属域的相同属性。

例如：

```
<property>
<name>hadoop.security.auth_to_local</name>
<value>RULE:[2:$1@$0](.*@\QTWO.COM\E$)s/@\QTWO.COM\E$//DEFAULT</value>
</property>
```

----结束

8.19 配置 Region Transition 恢复线程

配置场景

在故障环境中，由于诸如 region 服务器响应慢，网络不稳定，ZooKeeper 节点版本不匹配等各种原因，有可能导致 region 长时间处于 transition 下。在 region transition 下，由于一些 region 不能对外提供服务，客户端操作可能无法正常执行。

配置描述

在 HMaster 上设置 chore 服务，用于识别和恢复长期处于 transition 的 region。

下表是用于启用此功能的配置参数。

表8-14 参数描述

参数	描述	默认值
hbase.region.assignment.auto.recovery.enabled	配置该参数以启用或禁用 region 分配恢复线程功能。	true

8.20 使用二级索引

操作场景

HIndex 为 HBase 提供了按照某些列的值进行索引的能力，缩小搜索范围并缩短时延。

使用约束

- 列族应以“;”分隔。
- 列和数据类型应包含在“[]”中。
- 列数据类型在列名称后使用“->”指定。
- 如果未指定列数据类型，则使用默认数据类型（字符串）。
- “＃”用于在两个索引详细信息之间进行分隔。
- 以下是一个可选参数：

  -Dscan.caching：在扫描数据表时的缓存行数。

  如果不设置该参数，则默认值为 1000。
- 为单个 Region 构建索引是为了修复损坏的索引。

  此功能不应用于生成新索引。

操作步骤

步骤 1 安装 HBase 客户端，详情参见使用 HBase 客户端。

步骤 2 进入客户端安装路径，例如“/opt/client”

cd /opt/client

步骤 3 配置环境变量。

source bigdata_env

步骤 4 如果集群为安全模式，执行以下命令进行用户认证。普通模式集群无需执行用户认证。

kinit *组件业务用户*

步骤 5 执行以下命令访问 Hindex。

hbase org.apache.hadoop.hbase.hindex.mapreduce.TableIndexer

表8-15 HIndex 常用命令

说明	命令
增加索引	TableIndexer-Dtablename.to.index=table1-Dindexspecs.to.add='IDX1=>cf1:[q1->datatype],[q2],[q3];cf2:[q1->datatype],[q2->datatype]#IDX2=>cf1:[q5]'
构建索引	TableIndexer -Dtablename.to.index=table1 -Dindexnames.to.build='IDX1#IDX2'
删除索引	TableIndexer -Dtablename.to.index=table1 -Dindexnames.to.drop='IDX1#IDX2'
禁用索引	TableIndexer -Dtablename.to.index=table1 -Dindexnames.to.disable='IDX1#IDX2'
同时添加和构建索引	TableIndexer -Dtablename.to.index=table1 -Dindexspecs.to.add='IDX1=>cf1:[q1->datatype],[q2],[q3];cf2:[q1->datatype],[q2->datatype]#IDX2=>cf1:[q5]' -Dindexnames.to.build='IDX1'
为单个 Region 构建索引	TableIndexer -Dtablename.to.index=table1 -Dregion.to.index=regionEncodedName -Dindexnames.to.build='IDX1#IDX2'

说明

- **IDX1**：索引名称。
- **cf1**：列族名称。
- **q1**：列名称。
- **datatype**：数据类型，包括 String，Integer，Double，Float，Long，Short，Byte，Char。

----结束

8.21 HBase 日志介绍

日志描述

日志存储路径：HBase 相关日志的默认存储路径为“/var/log/Bigdata/hbase/*角色名*”。

- HMaster：“/var/log/Bigdata/hbase/hm”（运行日志），“/var/log/Bigdata/audit/hbase/hm”（审计日志）。
- RegionServer：“/var/log/Bigdata/hbase/rs”（运行日志），“/var/log/Bigdata/audit/hbase/rs”（审计日志）。
- ThriftServer：“/var/log/Bigdata/hbase/ts2”（运行日志，ts2 为具体实例名称），“/var/log/Bigdata/audit/hbase/ts2”（审计日志，ts2 为具体实例名称）。

日志归档规则：HBase 的日志启动了自动压缩归档功能，缺省情况下，当日志大小超过 30MB 的时候，会自动压缩，压缩后的日志文件名规则为：“<原有日志名>-<yyyy-mm-dd_hh-mm-ss>.[编号].log.zip”。最多保留最近的 20 个压缩文件，压缩文件保留个数可以在 Manager 界面中配置。

表8-16 HBase 日志列表

日志类型	日志文件名	描述
运行日志	hbase-<SSH_USER>-<process_name>-<hostname>.log	HBase 系统日志，主要包括启动时间，启动参数信息以及 HBase 系统运行时候所产生的大部分日志。
	hbase-<SSH_USER>-<process_name>-<hostname>.out	HBase 运行环境信息日志。
	<process_name>-<SSH_USER>-<DATE>-<PID>-gc.log	HBase 服务垃圾回收日志。
	checkServiceDetail.log	HBase 服务启动是否成功的检查日志。
	hbase.log	HBase 服务健康检查脚本以及部分告警检查脚本执行所产生的日志。
	sendAlarm.log	HBase 告警检查脚本上报告警信息日志。
	hbase-haCheck.log	HMaster 主备状态检测日志。
	stop.log	HBase 服务进程启停操作日志。
审计日志	hbase-audit-<process_name>.log	HBase 安全审计日志。

日志级别

HBase 中提供了如表 8-17 所示的日志级别。日志级别优先级从高到低分别是 FATAL、ERROR、WARN、INFO、DEBUG。程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。

表8-17 日志级别

级别	描述
FATAL	FATAL 表示当前事件处理出现严重错误信息，可能导致系统崩溃。
ERROR	ERROR 表示当前事件处理出现错误信息，系统运行出错。
WARN	WARN 表示当前事件处理存在异常信息，但认为是正常范围，不会导致系统出错。
INFO	INFO 表示记录系统及各事件正常运行状态信息
DEBUG	DEBUG 表示记录系统及系统的调试信息。

如果您需要修改日志级别，请执行如下操作：

步骤 1 进入 HBase 服务参数“全部配置”界面，具体操作请参考修改集群服务配置参数。

步骤 2 左边菜单栏中选择所需修改的角色所对应的日志菜单。

步骤 3 选择所需修改的日志级别。

步骤 4 保存配置，在弹出窗口中单击“确定”使配置生效。

说明

配置完成后立即生效，不需要重启服务。

----结束

日志格式

HBase 的日志格式如下所示：

表8-18 日志格式

日志类型	组件	格式	示例
运行日志	HMaster	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<产生该日志的线程名字>|<log 中的 message>|<日志事件的	2020-01-19 16:04:53,558 | INFO | main | env:HBASE_THRIFT_OPTS = | org.apache.hadoop.hbase.util.S

日志类型	组件	格式	示例
		发生位置>	erverCommandLine.logProcessInfo(ServerCommandLine.java:113)
	RegionServer	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<产生该日志的线程名字>|<log 中的message>|<日志事件的发生位置>	2020-01-19 16:05:18,589 | INFO | regionserver16020-SendThread(linux-k6da:2181) | Client will use GSSAPI as SASL mechanism. | org.apache.zookeeper.client.ZooKeeperSaslClient$1.run(ZooKeeperSaslClient.java:285)
	ThriftServer	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<产生该日志的线程名字>|<log 中的message>|<日志事件的发生位置>	2020-02-16 09:42:55,371 | INFO | main | loaded properties from hadoop-metrics2.properties | org.apache.hadoop.metrics2.impl.MetricsConfig.loadFirst(MetricsConfig.java:111)
审计日志	HMaster	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<产生该日志的线程名字>|<log 中的message>|<日志事件的发生位置>	2020-02-16 09:42:40,934 | INFO | master:linux-k6da:16000 | Master: [master:linux-k6da:16000] start operation called. | org.apache.hadoop.hbase.master.HMaster.run(HMaster.java:581)
	RegionServer	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<产生该日志的线程名字>|<log 中的message>|<日志事件的发生位置>	2020-02-16 09:42:51,063 | INFO | main | RegionServer: [regionserver16020] start operation called. | org.apache.hadoop.hbase.regionserver.HRegionServer.startRegionServer(HRegionServer.java:2396)
	ThriftServer	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<产生该日志的线程名字>|<log 中的message>|<日志事件的发生位置>	2020-02-16 09:42:55,512 | INFO | main | thrift2 server start operation called. | org.apache.hadoop.hbase.thrift2.ThriftServer.main(ThriftServer.java:421)

8.22 HBase 性能调优

8.22.1 提升 BulkLoad 效率

操作场景

批量加载功能采用了 MapReduce jobs 直接生成符合 HBase 内部数据格式的文件，然后把生成的 StoreFiles 文件加载到正在运行的集群。使用批量加载相比直接使用 HBase 的 API 会节约更多的 CPU 和网络资源。

ImportTSV 是一个 HBase 的表数据加载工具。

说明

本章节适用于 MRS 3.x 及之后版本。

前提条件

在执行批量加载时需要通过“Dimporttsv.bulk.output”参数指定文件的输出路径。

操作步骤

参数入口：执行批量加载任务时，在 BulkLoad 命令行中加入如下参数。

表8-19 增强 BulkLoad 效率的配置项

参数	描述	配置的值
-Dimporttsv.mapper.class	用户自定义 mapper 通过把键值对的构造从 mapper 移动到 reducer 以帮助提高性能。mapper 只需要把每一行的原始文本发送给 reducer，reducer 解析每一行的每一条记录并创建键值对。 说明 当该值配置为“org.apache.hadoop.hbase.mapreduce.TsvImporterByteMapper”时，只在执行没有 *HBASE_CELL_VISIBILITY OR HBASE_CELL_TTL* 选项的批量加载命令时使用。使用“org.apache.hadoop.hbase.mapreduce.TsvImporterByteMapper”时可以得到更好的性能。	org.apache.hadoop.hbase.mapreduce.TsvImporterByteMapper 和 org.apache.hadoop.hbase.mapreduce.TsvImporterTextMapper

8.22.2 提升连续 put 场景性能

操作场景

对大批量、连续 put 的场景，配置下面的两个参数为“false”时能大量提升性能。

- “hbase.regionserver.wal.durable.sync”
- “hbase.regionserver.hfile.durable.sync”

当提升性能时，缺点是对于 DataNode（默认是 3 个）同时故障时，存在小概率数据丢失的现象。对数据可靠性要求高的场景请慎重配置。

说明

本章节适用于 MRS 3.x 及之后版本。

操作步骤

参数入口：

在 FusionInsight Manager 系统中，选择“集群 > *待操作集群的名称* > 服务 > HBase > 配置”，单击“全部配置”。在搜索框中输入参数名称，并进行修改。

表8-20 提升连续 put 场景性能的参数

参数	描述	配置值
hbase.wal.hsync	设置是否启用 WAL 文件耐久性以将 WAL 数据持久化到磁盘。若将该参数设置为 true，则性能将受到影响，原因是每个 WAL 的编辑都会被 hadoop fsync 同步到磁盘上。	false
hbase.hfile.hsync	设置是否启用 Hfile 耐久性以将数据持久化到磁盘。若将该参数设置为 true，则性能将受到影响，原因是每个 Hfile 写入时都会被 hadoop fsync 同步到磁盘上。	false

8.22.3 Put 和 Scan 性能综合调优

操作场景

HBase 有很多与读写性能相关的配置参数。读写请求负载不同的情况下，配置参数需要进行相应的调整，本章节旨在指导用户通过修改 RegionServer 配置参数进行读写性能调优。

说明

本章节适用于 MRS 3.x 及之后版本。

操作步骤

- JVM GC 参数

RegionServer GC_OPTS 参数设置建议：

- -Xms 与-Xmx 设置相同的值，需要根据实际情况设置，增大内存可以提高读写性能，可以参考参数“hfile.block.cache.size”（见表 8-22）和参数“hbase.regionserver.global.memstore.size”（见表 8-21）的介绍进行设置。
- -XX:NewSize 与-XX:MaxNewSize 设置相同值，建议低负载场景下设置为“512M”，高负载场景下设置为“2048M”。
- -XX:CMSInitiatingOccupancyFraction 建议设置为“100 * (hfile.block.cache.size + hbase.regionserver.global.memstore.size + 0.05)”，最大值不超过 90。
- -XX:MaxDirectMemorySize 表示 JVM 使用的堆外内存，建议低负载情况下设置为“512M”，高负载情况下设置为“2048M”。

说明

GC_OPTS 参数中-XX:MaxDirectMemorySize 默认没有配置，如需配置，用户可在 GC_OPTS 参数中自定义添加。

- Put 相关参数

  RegionServer 处理 put 请求的数据，会将数据写入 memstore 和 hlog，

  - 当 memstore 大小达到设置的“hbase.hregion.memstore.flush.size”参数值大小时，memstore 就会刷新到 HDFS 生成 HFile。
  - 当当前 region 的列簇的 HFile 数量达到“hbase.hstore.compaction.min”参数值时会触发 compaction。
  - 当当前 region 的列簇 HFile 数达到“hbase.hstore.blockingStoreFiles”参数值时会阻塞 memstore 刷新生成 HFile 的操作，导致 put 请求阻塞。

表8-21 Put 相关参数

参数	描述	默认值
hbase.wal.hsync	每一条 wal 是否持久化到硬盘。 参考提升连续 put 场景性能。	true
hbase.hfile.hsync	hfile 写是否立即持久化到硬盘。 参考提升连续 put 场景性能。	true
hbase.hregion.memstore.flush.size	若 MemStore 的大小（单位：Byte）超过指定值，MemStore 将被冲洗至磁盘。该参数值将被运行每个 hbase.server.thread.wakefrequency 的线程所检验。建议设置为 HDFS 块大小的整数倍，在内存足够 put 负载大情况下可以调整增大。	134217728
hbase.regionserver.global.memstore.size	更新被锁定以及强制冲洗发生之前一个 RegionServer 上支持的所有 MemStore 的大小。建议设置为	0.4

参数	描述	默认值
	“hbase.hregion.memstore.flush.size * 写活跃 region 数 / RegionServer GC -Xmx”。默认值为“0.4”，表示使用 RegionServer GC -Xmx 的 40%。	
hbase.hstore.flusher.count	memstore 的 flush 线程数，在 put 高负载场景下可以适当调大。	2
hbase.regionserver.thread.compaction.small	小压缩线程数，在 put 高负载情况下可以适当调大。	10
hbase.hstore.blockingStoreFiles	若一个 Store 内的 HStoreFile 文件数量超过指定值，则针对此 HRegion 的更新将被锁定直到一个压缩完成或者 base.hstore.blockingWaitTime 被超过。每冲洗一次 MemStore 一个 StoreFile 文件被写入。在 put 高负载场景下可以适当调大。	15

- Scan 相关参数

表8-22 Scan 相关参数

参数	描述	默认值
hbase.client.scanner.timeout.period	客户端和 RegionServer 端参数，表示客户端执行 scan 的租约超时时间。建议设置为 60000ms 的整数倍，在读高负载情况下可以适当调大。单位：毫秒。	60000
hfile.block.cache.size	数据缓存所占的 RegionServer GC -Xmx 百分比，在读高负载情况下可以适当调大以增大缓存命中率以提高性能。表示分配给 HFile/StoreFile 所使用的块缓存的最大 heap（-Xmx setting）的百分比。	当 offheap 关闭时，默认值为 0.25，当 offheap 开启时，默认值是 0.1。

- Handler 相关参数

表8-23 Handler 相关参数

参数	描述	默认值

参数	描述	默认值
hbase.regionserver.handler.count	RegionServer 上的 RPC 侦听器实例数，建议设置为 200 ~ 400 之间。	200
hbase.regionserver.metahandler.count	RegionServer 中处理优先请求的程序实例的数量，建议设置为 200 ~ 400 之间。	200

8.22.4 提升实时写数据效率

操作场景

需要把数据实时写入到 HBase 中或者对于大批量、连续 put 的场景。

说明

本章节适用于 MRS 3.x 及之后版本。

前提条件

调用 HBase 的 put 或 delete 接口，把数据保存到 HBase 中。

操作步骤

- **写数据服务端调优**

  参数入口：

  进入 HBase 服务参数“全部配置”界面，具体操作请参考修改集群服务配置参数章节。

表8-24 影响实时写数据配置项

配置参数	描述	默认值
hbase.wal.hsync	控制 HLog 文件在写入到 HDFS 时的同步程度。如果为 true，HDFS 在把数据写入到硬盘后才返回；如果为 false，HDFS 在把数据写入 OS 的缓存后就返回。 把该值设置为 false 比 true 在写入性能上会更优。	true
hbase.hfile.hsync	控制 HFile 文件在写入到 HDFS 时的同步程度。如果为 true，HDFS 在把数据写入到硬盘后才返回；如果为 false，HDFS 在把数据写入 OS 的缓存后就返回。	true

配置参数	描述	默认值
	把该值设置为 false 比 true 在写入性能上会更优。	
GC_OPTS	HBase 利用内存完成读写操作。提高 HBase 内存可以有效提高 HBase 性能。GC_OPTS 主要需要调整 HeapSize 的大小和 NewSize 的大小。调整 HeapSize 大小的时候，建议将 Xms 和 Xmx 设置成相同的值，这样可以避免 JVM 动态调整 HeapSize 大小的时候影响性能。调整 NewSize 大小的时候，建议把其设置为 HeapSize 大小的 1/8。 • HMaster：当 HBase 集群规模越大、Region 数量越多时，可以适当调大 HMaster 的 GC_OPTS 参数。 • RegionServer：RegionServer 需要的内存一般比 HMaster 要大。在内存充足的情况下，HeapSize 可以相对设置大一些。 说明 主 HMaster 的 HeapSize 为 4G 的时候，HBase 集群可以支持 100000 region 数的规模。根据经验值，集群每增加 35000 个 region，HeapSize 增加 2G，主 HMaster 的 HeapSize 不建议超过 32GB。	• HMaster -server -Xms4G -Xmx4G -XX:NewSize=512M -XX:MaxNewSize=512M -XX:MetaspaceSize=128M -XX:MaxMetaspaceSize=512M -XX:+UseConcMarkSweepGC -XX:+CMSParallelRemarkEnabled -XX:CMSInitiatingOccupancyFraction=65 -XX:+PrintGCDetails -Dsun.rmi.dgc.client.gcInterval=0x7FFFFFFFFFFFFE -Dsun.rmi.dgc.server.gcInterval=0x7FFFFFFFFFFFFE -XX:-OmitStackTraceInFastThrow -XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=1M • Region Server -server -Xms6G -Xmx6G -XX:NewSize=1024M -XX:MaxNewSiz

配置参数	描述	默认值
		e=1024M -XX:MetaspaceSize=128M -XX:MaxMetaspaceSize=512M -XX:+UseConcMarkSweepGC -XX:+CMSParallelRemarkEnabled -XX:CMSInitiatingOccupancyFraction=65 -XX:+PrintGCDetails -Dsun.rmi.dgc.client.gcInterval=0x7FFFFFFFFFFFFFE -Dsun.rmi.dgc.server.gcInterval=0x7FFFFFFFFFFFFFE -XX:-OmitStackTraceInFastThrow -XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=1M
hbase.regionserver.handler.count	表示在 RegionServer 上启动的 RPC 侦听器实例数。如果设置过高会导致激烈线程竞争，如果设置过小，请求将会在 RegionServer 长时间等待，降低处理能力。根据资源情况，适当增加处理线程数。 建议根据 CPU 的使用情况，可以选择设置为 100 至 300 之间的值。	200
hbase.hregion.max.filesize	HStoreFile 的最大大小（单位：Byte）。若任何一个列族 HStoreFile 超过此参数值，则托管 Hregion 将会一分为二。	10737418240

配置参数	描述	默认值
hbase.hregion.memstore.flush.size	在 RegionServer 中，当写操作内存中存在超过 memstore.flush.size 大小的 memstore，则 MemStoreFlusher 就启动 flush 操作将该 memstore 以 hfile 的形式写入对应的 store 中。 如果 RegionServer 的内存充足，而且活跃 Region 数量也不是很多的时候，可以适当增大该值，可以减少 compaction 的次数，有助于提升系统性能。 同时，这种 flush 产生的时候，并不是紧急的 flush，flush 操作可能会有一定延迟，在延迟期间，写操作还可以进行，Memstore 还会继续增大，最大值为“memstore.flush.size” * “hbase.hregion.memstore.block.multiplier”。当超过最大值时，将会阻塞操作。适当增大“hbase.hregion.memstore.block.multiplier”可以减少阻塞，减少性能波动。单位：字节。	134217728
hbase.regionserver.global.memstore.size	更新被锁定以及强制冲洗发生之前一个 RegionServer 上支持的所有 MemStore 的大小。RegionServer 中，负责 flush 操作的是 MemStoreFlusher 线程。该线程定期检查写操作内存，当写操作占用内存总量达到阈值，MemStoreFlusher 将启动 flush 操作，按照从大到小的顺序，flush 若干相对较大的 memstore，直到所占用内存小于阈值。 阈值 = “hbase.regionserver.global.memstore.size” * “hbase.regionserver.global.memstore.size.lower.limit” * “HBase_HEAPSIZE” 说明 该配置与“hfile.block.cache.size”的和不能超过 0.8，也就是写和读操作的内存不能超过 HeapSize 的 80%，这样可以保证除读和写外其它操作的正常运行。	0.4
hbase.hstore.blockingStoreFiles	在 region flush 前首先判断 file 文件个数，是否大于 hbase.hstore.blockingStoreFiles。 如果大于需要先 compaction 并且让	15

配置参数	描述	默认值
	flush 延时 90s（这个值可以通过 hbase.hstore.blockingWaitTime 进行配置），在延时过程中，将会继续写从而使得 Memstore 还会继续增大超过最大值 “memstore.flush.size” * “hbase.hregion.memstore.block.multiplier”，导致写操作阻塞。当完成 compaction 后，可能就会产生大量写入。这样就导致性能激烈震荡。 增加 hbase.hstore.blockingStoreFiles，可以减低 BLOCK 几率。	
hbase.regionserver.thread.compaction.throttle	大于此参数值的压缩将被大线程池执行，单位：Byte。控制一次 Minor Compaction 时，进行 compaction 的文件总大小的阈值。Compaction 时的文件总大小会影响这一次 compaction 的执行时间，如果太大，可能会阻塞其它的 compaction 或 flush 操作。	1610612736
hbase.hstore.compaction.min	每次执行 minor compaction 的 HStoreFile 的最小数量。当一个 Store 中文件超过该值时，会进行 compact，适当增大该值，可以减少文件被重复执行 compaction。但是如果过大，会导致 Store 中文件数过多而影响读取的性能。	6
hbase.hstore.compaction.max	每次执行 minor compaction 的 HStoreFile 的最大数量。与“hbase.hstore.compaction.max.size”的作用基本相同，主要是控制一次 compaction 操作的时间不要太长。	10
hbase.hstore.compaction.max.size	如果一个 HFile 文件的大小大于该值，那么在 Minor Compaction 操作中不会选择这个文件进行 compaction 操作，除非进行 Major Compaction 操作。 这个值可以防止较大的 HFile 参与 compaction 操作。在禁止 Major Compaction 后，一个 Store 中可能存在几个 HFile，而不会合并成为一个 HFile，这样不会对数据读取造成太大的性能影响。单位：字节。	9223372036854775807
hbase.hregion.majorcompaction	单个区域内所有 HStoreFile 文件主压缩的时间间隔，单位：毫秒。由于执行 Major Compaction 会占用较多的系统资	604800000

配置参数	描述	默认值
	源，如果正在处于系统繁忙时期，会影响系统的性能。 如果业务没有较多的更新、删除、回收过期数据空间时，可以把该值设置为0，以禁止 Major Compaction。 如果必须要执行 Major Compaction，以回收更多的空间，可以适当增加该值，同时配置参数“hbase.offpeak.end.hour”和“hbase.offpeak.start.hour”以控制 Major Compaction 发生在业务空闲的时期。单位：毫秒。	
• hbase.regionserver.maxlogs • hbase.regionserver.hlog.blocksize	• 表示一个 RegionServer 上未进行 Flush 的 Hlog 的文件数量的阈值，如果大于该值，RegionServer 会强制进行 flush 操作。 • 表示每个 HLog 文件的最大大小。如果 HLog 文件大小大于该值，就会滚动出一个新的 HLog 文件，旧的将被禁用并归档。 这两个参数共同决定了 RegionServer 中可以存在的未进行 Flush 的 hlog 数量。当这个数据量小于 MemStore 的总大小的时候，会出现由于 HLog 文件过多而触发的强制 flush 操作。这个时候可以适当调整这两个参数的大小，以避免出现这种强制 flush 的情况。单位：字节。	• 32 • 134217728

- **写数据客户端调优**

  写数据时，在场景允许的情况下，最好使用 Put List 的方式，可以极大的提升写性能。每一次 Put 的 List 的长度，需要结合单条 Put 的大小，以及实际环境的一些参数进行设定。建议在选定之前先做一些基础的测试。

- **写数据表设计调优**

表8-25 影响实时写数据相关参数

配置参数	描述	默认值
COMPRESSION	配置数据的压缩算法，这里的压缩是 HFile 中 block 级别的压缩。对于可以压缩的数据，配置压缩算法可以有效减少磁盘的 IO，从而达到提高性能的目的。	NONE

配置参数	描述	默认值
	说明 并非所有数据都可以进行有效压缩。例如一张图片的数据，因为图片一般已经是压缩后的数据，所以压缩效果有限。常用的压缩算法是 SNAPPY，因为它有较好的 Encoding/Decoding 速度和可以接受的压缩率。	
BLOCKSIZE	配置 HFile 中 block 块的大小，不同的 block 块大小，可以影响 HBase 读写数据的效率。越大的 block 块，配合压缩算法，压缩的效率就越好；但是由于 HBase 的读取数据是以 block 块为单位的，所以越大的 block 块，对于随机读的情况，性能可能会比较差。 如果要提升写入的性能，一般扩大到 128KB 或者 256KB，可以提升写数据的效率，也不会影响太大的随机读性能。单位：字节	65536
IN_MEMORY	配置这个表的数据优先缓存在内存中，这样可以有效提升读取的性能。对于一些小表，而且需要频繁进行读取操作的，可以设置此配置项。	false

8.22.5 提升实时读数据效率

操作场景

需要读取 HBase 数据场景。

前提条件

调用 HBase 的 get 或 scan 接口，从 HBase 中实时读取数据。

操作步骤

- **读数据服务端调优**

  参数入口：

  进入 HBase 服务参数“全部配置”界面，具体操作请参考修改集群服务配置参数章节。

表8-26 影响实时读数据配置项

配置参数	描述	默认值
GC_OPTS	HBase 利用内存完成读写操作。提高 HBase 内存可以有效提高 HBase 性	MRS 3.x 之前版本：

配置参数	描述	默认值
	能。 GC_OPTS 主要需要调整 HeapSize 的大小和 NewSize 的大小。调整 HeapSize 大小的时候，建议将 Xms 和 Xmx 设置成相同的值，这样可以避免 JVM 动态调整 HeapSize 大小的时候影响性能。调整 NewSize 大小的时候，建议把其设置为 HeapSize 大小的 1/8。 • HMaster：当 HBase 集群规模越大、Region 数量越多时，可以适当调大 HMaster 的 GC_OPTS 参数。 • RegionServer：RegionServer 需要的内存一般比 HMaster 要大。在内存充足的情况下，HeapSize 可以相对设置大一些。 说明 主 HMaster 的 HeapSize 为 4G 的时候，HBase 集群可以支持 100000 region 数的规模。根据经验值，集群每增加 35000 个 region，HeapSize 增加 2G，主 HMaster 的 HeapSize 不建议超过 32GB。	• HMaster： -server -Xms2G -Xmx2G -XX:NewSize=256M -XX:MaxNewSize=256M -XX:MetaspaceSize=128M -XX:MaxMetaspaceSize=512M -XX:MaxDirectMemorySize=512M -XX:+UseConcMarkSweepGC -XX:+CMSParallelRemarkEnabled -XX:CMSInitiatingOccupancyFraction=65 -XX:+PrintGCDetails -Dsun.rmi.dgc.client.gcInterval=0x7FFFFFFFFFFFFFE -Dsun.rmi.dgc.server.gcInterval=0x7FFFFFFFFFFFFFE -XX:-OmitStackTraceInFastThrow -XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=1M • RegionServer

配置参数	描述	默认值
		: -server -Xms4G -Xmx4G -XX:NewSize=512M -XX:MaxNewSize=512M -XX:MetaspaceSize=128M -XX:MaxMetaspaceSize=512M -XX:MaxDirectMemorySize=512M -XX:+UseConcMarkSweepGC -XX:+CMSParallelRemarkEnabled -XX:CMSInitiatingOccupancyFraction=65 -XX:+PrintGCDetails -Dsun.rmi.dgc.client.gcInterval=0x7FFFFFFFFFFFFFE -Dsun.rmi.dgc.server.gcInterval=0x7FFFFFFFFFFFFFE -XX:-OmitStackTraceInFastThrow -XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=1M MRS 3.x 及之后

配置参数	描述	默认值
		版本： • HMaster -server -Xms4G -Xmx4G -XX:NewSize=512M -XX:MaxNewSize=512M -XX:MetaspaceSize=128M -XX:MaxMetaspaceSize=512M -XX:+UseConcMarkSweepGC -XX:+CMSParallelRemarkEnabled -XX:CMSInitiatingOccupancyFraction=65 -XX:+PrintGCDetails -Dsun.rmi.dgc.client.gcInterval=0x7FFFFFFFFFFFFFFE -Dsun.rmi.dgc.server.gcInterval=0x7FFFFFFFFFFFFFFE -XX:-OmitStackTraceInFastThrow -XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=1M • Region Server -server -Xms6G -

配置参数	描述	默认值
		-Xmx6G -XX:NewSize=1024M -XX:MaxNewSize=1024M -XX:MetaspaceSize=128M -XX:MaxMetaspaceSize=512M -XX:+UseConcMarkSweepGC -XX:+CMSParallelRemarkEnabled -XX:CMSInitiatingOccupancyFraction=65 -XX:+PrintGCDetails -Dsun.rmi.dgc.client.gcInterval=0x7FFFFFFFFFFFFFE -Dsun.rmi.dgc.server.gcInterval=0x7FFFFFFFFFFFFFE -XX:-OmitStackTraceInFastThrow -XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=1M
hbase.regionserver.handler.count	表示 RegionServer 在同一时刻能够并发处理多少请求。如果设置过高会导致激烈线程竞争，如果设置过小，请求将会在 RegionServer 长时间等待，降低处理能力。根据资源情况，适当增加处理线程数。	200

配置参数	描述	默认值
	建议根据 CPU 的使用情况，可以选择设置为 100 至 300 之间的值。	
hfile.block.cache.size	HBase 缓存区大小，主要影响查询性能。根据查询模式以及查询记录分布情况来决定缓存区的大小。如果采用随机查询使得缓存区的命中率较低，可以适当降低缓存区大小。	当 offheap 关闭时，默认值为 0.25。当 offheap 开启时，默认值是 0.1。

说明

如果同时存在读和写的操作，这两种操作的性能会互相影响。如果写入导致的 flush 和 Compaction 操作频繁发生，会占用大量的磁盘 IO 操作，从而影响读取的性能。如果写入导致阻塞较多的 Compaction 操作，就会出现 Region 中存在多个 HFile 的情况，从而影响读取的性能。所以如果读取的性能不理想的时候，也要考虑写入的配置是否合理。

- **读数据客户端调优**

  Scan 数据时需要设置 caching（一次从服务端读取的记录条数，默认是 1），若使用默认值读性能会降到极低。

  当不需要读一条数据所有的列时，需要指定读取的列，以减少网络 IO。

  只读取 RowKey 时，可以为 Scan 添加一个只读取 RowKey 的 filter（FirstKeyOnlyFilter 或 KeyOnlyFilter）。

- **读数据表设计调优**

表8-27 影响实时读数据相关参数

配置参数	描述	默认值
COMPRESSION	配置数据的压缩算法，这里的压缩是 HFile 中 block 级别的压缩。对于可以压缩的数据，配置压缩算法可以有效减少磁盘的 IO，从而达到提高性能的目的。 说明 并非所有数据都可以进行有效压缩。例如一张图片的数据，因为图片一般已经是压缩后的数据，所以压缩效果有限。常用的压缩算法是 SNAPPY，因为它有较好的 Encoding/Decoding 速度和可以接受的压缩率。	NONE
BLOCKSIZE	配置 HFile 中 block 块的大小，不同的 block 块大小，可以影响 HBase 读写数据的效率。越大的 block 块，配合压缩算法，压缩的效率就越好；但是由于 HBase 的读取数据是以 block 块为单位的，所以越大的 block 块，对于随机读的情况，性能可能会比较差。	65536

配置参数	描述	默认值
	如果要提升写入的性能，一般扩大到 128KB 或者 256KB，可以提升写数据的效率，也不会影响太大的随机读性能。单位：字节。	
DATA_BLOCK_ENCODING	配置 HFile 中 block 块的编码方法。当一行数据中存在多列时，一般可以配置为“FAST_DIFF”，可以有效的节省数据存储的空间，从而提供性能。	NONE

8.22.6 JVM 参数优化

操作场景

当集群数据量达到一定规模后，JVM 的默认配置将无法满足集群的业务需求，轻则集群变慢，重则集群服务不可用。所以需要根据实际的业务情况进行合理的 JVM 参数配置，提高集群性能。

操作步骤

参数入口：

HBase 角色相关的 JVM 参数需要配置在安装有 HBase 服务的节点的“${BIGDATA_HOME}/FusionInsight_HD_*/install/FusionInsight-HBase-2.2.3/hbase/conf/”目录下的“hbase-env.sh”文件中。

每个角色都有各自的 JVM 参数配置变量，如表 8-28。

表8-28 HBase 相关 JVM 参数配置变量

变量名	变量影响的角色
HBASE_OPTS	该变量中设置的参数，将影响 HBase 的所有角色。
SERVER_GC_OPTS	该变量中设置的参数，将影响 HBase Server 端的所有角色，例如：Master、RegionServer 等。
CLIENT_GC_OPTS	该变量中设置的参数，将影响 HBase 的 Client 进程。
HBASE_MASTER_OPTS	该变量中设置的参数，将影响 HBase 的 Master。
HBASE_REGIONSERVER_OPTS	该变量中设置的参数，将影响 HBase 的 RegionServer。
HBASE_THRIFT_OPTS	该变量中设置的参数，将影响 HBase 的 Thrift。

配置方式举例：

```
export HADOOP_NAMENODE_OPTS="-Dhadoop.security.logger=${HADOOP_SECURITY_LOGGER:-
INFO,RFAS} -Dhdfs.audit.logger=${HDFS_AUDIT_LOGGER:-INFO,NullAppender}
$HADOOP_NAMENODE_OPTS"
```

8.23 HBase 常见问题

8.23.1 客户端连接服务端时，长时间无法连接成功

问题

在 HBase 服务端出现问题，无法提供服务，此时 HBase 客户端进行表操作，会出现该操作挂起，长时间无任何反应。

回答

问题分析

当 HBase 服务端出现问题，HBase 客户端进行表操作的时候，会进行重试，并等待超时。该超时默认值为 Integer.MAX_VALUE (2147483647 ms)，所以 HBase 客户端会在这么长的时间内一直重试，造成挂起表象。

解决方法

HBase 客户端提供两个配置项来控制客户端的重试超时方式，如表 8-29。

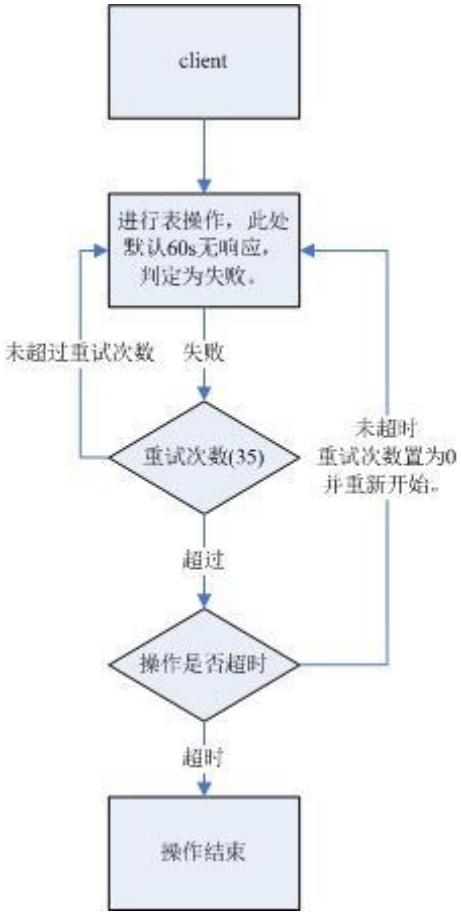
在“*客户端安装路径*/HBase/hbase/conf/hbase-site.xml”配置文件中配置如下参数。

表8-29 HBase 客户端操作重试超时相关配置

配置参数	描述	默认值
hbase.client.operation.timeout	客户端操作超时时间。需在配置文件中手动添加。	2147483647 ms
hbase.client.retries.number	最大重试次数。用于表示所有可重试操作所支持的最大重试次数。	35

这两个参数的重试超时的配合方式如图 8-3 所示。

图8-3 HBase 客户端操作重试超时流程



从该流程可以看出，如果未对这两个配置参数根据具体使用场景进行配置，会造成挂起迹象。建议根据使用场景，配置合适的超时时间，如果是长时间操作，则把超时时间设置长一点；如果是短时间操作，则把超时时间设置短一点。而重试次数可以设置为：“(hbase.client.retries.number)*60*1000(ms)”。刚好大于“hbase.client.operation.timeout”设置的超时时间。

8.23.2 结束 BulkLoad 客户端程序，导致作业执行失败

问题

执行 BulkLoad 程序导入数据时，如果结束客户端程序，为什么有时会导致已提交的作业执行失败？

回答

BulkLoad 程序在客户端启动时会生成一个 partitioner 文件，用于划分 Map 任务数据输入的范围。此文件在 BulkLoad 客户端退出时会被自动删除。一般来说当所有 Map 任务都启动运行以后，退出 BulkLoad 客户端也不会导致已提交的作业失败。但由于 Map 任务存在重试机制和推测执行机制；Reduce 任务下载一个已运行完成的 Map 任务的数据失败次数过多时，Map 任务也会被重新执行。如果此时 BulkLoad 客户端已经退出，

则重试的 Map 任务会因为找不到 partitioner 文件而执行失败，导致作业执行失败。因此，强烈建议 BulkLoad 程序在数据导入期间不要结束客户端程序。

8.23.3 在 HBase 连续对同一个表名做删除创建操作时，可能出现创建表异常

问题

在 HBase 连续对同一个表名做删除创建操作时，可能出现创建表异常。

回答

执行过程：Disable Table > Drop Table > Create Table > Disable Table > Drop Table >...

1. 在 Disable 表时，HMaster 会发送 RPC 请求到 RegionServer，RegionServer 会将相关 Region 下线。当 RegionServer 上的 Region 关闭所需的时间超过 HBase 的 HMaster 等待 Region 处于 RIT 状态的超时时间，HMaster 会默认该 Region 下线，实际上该 Region 可能还处在 flush memstore 阶段。
2. 发送 RPC 请求关闭 Region 之后，HMaster 会判断该表的所有 Region 是否下线，上述 1 的情况下关闭超时也会认为是下线，然后 HMaster 返回关闭成功。
3. 关闭成功之后，删除表，HBase 表对应的数据目录被删掉。
4. 在删除表之后，该数据目录会被还处于 flush memstore 阶段的 Region 重新创建。
5. 再创建该表时，将 temp 目录拷贝到 HBase 数据目录时，由于 HBase 数据目录不为空，导致调用 HDFS rename 接口时，数据目录变为 temp 目录最后一层追加到 HBase 的数据目录下，如$rootDir/data/$nameSpace/$tableName/$tableName，那样创建表就会失败。

解决办法：

出现该问题时，请检查该表对应的 HBase 数据目录是否存在，如果存在请将该目录重命名。

HBase 数据目录由$rootDir/data/$nameSpace/$tableName 组成，例如“hdfs://hacluster/hbase/data/default/TestTable”，其中$rootDir 是 HBase 的根目录，该值通过在“hbase-site.xml”中配置 hbase.rootdir.perms 得到，data 目录是 HBase 的固定目录，$nameSpace 是 nameSpace 名字，$tableName 是表名。

8.23.4 HBase 占用网络端口，连接数过大会导致其他服务不稳定

问题

HBase 占用网络端口，连接数过大会导致其他服务不稳定。

回答

使用操作系统命令 ***lsof*** 或者 ***netstat*** 发现大量 TCP 连接处于 CLOSE_WAIT 状态，且连接持有者为 HBase RegionServer，可能导致网络端口耗尽或 HDFS 连接超限，那样可能会导致其他服务不稳定。HBase CLOSE_WAIT 现象为 HBase 机制。

HBase CLOSE_WAIT 产生原因：HBase 数据以 HFile 形式存储在 HDFS 上，这里可以叫 StoreFiles，HBase 作为 HDFS 的客户端，HBase 在创建 StoreFile 或启动加载 StoreFile 时创建了 HDFS 连接，当创建 StoreFile 或加载 StoreFile 完成时，HDFS 方面认为任务已完成，将连接关闭权交给 HBase，但 HBase 为了保证实时响应，有请求时就可以连接对应数据文件，需要保持连接，选择不关闭连接，所以连接状态为 CLOSE_WAIT（需客户端关闭）。

什么时候会创建 StoreFile：当 HBase 执行 Flush 时。

什么时候执行 Flush：HBase 写入数据首先会存在内存 memstore，只有内存使用达到阈值或手动执行 ***flush*** 命令时会触发 flush 操作，将数据写入 HDFS。

解决方法：

由于 HBase 连接机制，若想减小 HBase 端口占用，则需控制 StoreFile 数量，具体可以通过触发 HBase 的 compaction 动作完成，即触发 HBase 文件合并，方法如下：

方法 1：使用 HBase shell 客户端，在客户端手动执行 ***major_compact*** 操作。

方法 2：编写 HBase 客户端代码，调用 HBaseAdmin 类中的 compact 方法触发 HBase 的 compaction 动作。

如果 compact 无法解决 HBase 端口占用现象，说明 HBase 使用情况已经达到瓶颈，需考虑如下几点：

- table 的 Region 数初始设置是否合适。
- 是否存在无用数据。

若存在无用数据，可删除对应数据以减小 HBase 存储文件数量，若以上情况都不满足，则需考虑扩容。

8.23.5 HBase bulkload 任务（单个表有 26T 数据）有 210000 个 map 和 10000 个 reduce，任务失败

问题

MRS 3.x 及之后版本 HBase bulkLoad 任务（单个表有 26T 数据）有 210000 个 map 和 10000 个 reduce，任务失败。

回答

ZooKeeper IO 瓶颈观测手段：

1. 通过 Manager 的监控页面查看单个节点上 ZooKeeper 请求监控，判断是否严重超出规格限制。
2. 通过观测 ZooKeeper 的日志以及 HBase 的日志，查看是否有大量的 IO Exception Timeout 或者 SocketTimeout Exception 异常。

调优建议：

1. 将 ZooKeeper 实例个数调整为 5 个及以上，最好是通过设置 peerType=observer 来增加 observer 的数目。

2. 通过控制单个任务并发的 map 数或减少每个节点下运行 task 的内存，降低节点负载。
3. 升级 ZooKeeper 数据磁盘，如 SSD 等。

8.23.6 如何修复长时间处于 RIT 状态的 Region

问题

在 HBase WEBUI 界面看到有长时间处于 RIT 状态的 Region，如何修复？

回答

登录 HMaster WebUI，在导航栏选择“Procedure & Locks”，查看是否有处于 Waiting 状态的 process id。如果有，需要执行以下命令将 procedure lock 释放：

hbase hbck -j *客户端安装目录*/**HBase/hbase/tools/hbase-hbck2-*.jar bypass -o** *pid*

查看 State 是否处于 Bypass 状态，如果界面上的 procedures 一直处于 RUNNABLE(Bypass)状态，需要进行主备切换。执行 **assigns** 命令使 region 重新上线。

hbase hbck -j *客户端安装目录*/**HBase/hbase/tools/hbase-hbck2-*.jar assigns -o** *regionName*

8.23.7 HMaster 等待 namespace 表上线时超时退出

问题

为什么在等待 namespace 表上线时超时 HMaster 退出？

回答

在 HMaster 主备倒换或启动期间，HMaster 为先前失败/停用的 RegionServer 执行 WAL splitting 及 region 恢复。

在后台运行有多个监控 HMaster 启动进程的线程：

- TableNamespaceManager

  这是一个帮助类，用于在 HMaster 主备倒换或启动期间，管理 namespace 表及监控表 region 的分配。如果 namespace 表在规定时间（hbase.master.namespace.init.timeout，默认为 3600000ms）内没有上线，那么它就会异常中断 HMaster 进程。
- InitializationMonitor

  这是一个主 HMaster 初始化线程监控类，用于监控主 Master 的初始化。如果在规定时间（hbase.master.initializationmonitor.timeout，默认为 3600000ms）内初始化线程失败，该线程会异常终止 HMaster（如果该 hbase.master.initializationmonitor.haltontimeout 被启动，默认为 false）。

在 HMaster 主备倒换或启动期间，如果 WAL hlog 文件存在，它会初始化 WAL splitting 任务。如果 WAL hlog splitting 任务完成，它将初始化表 region 分配任务。

HMaster 通过 ZooKeeper 协调 log splitting 任务和有效的 RegionServer，并追踪任务的发展。如果主 HMaster 在 log splitting 任务期间退出，新的主 HMaster 会尝试重发没有完成的任务，RegionServer 从头启动 log splitting 任务。

HMaster 初始化工作完成情况会由于很多原因被延迟：

- 间歇性的网络故障。
- 磁盘瓶颈。
- log split 任务工作负荷较大，RegionServer 运行缓慢。
- RegionServer（region openning）响应缓慢。

在以上场景中，为使 HMaster 更早完成恢复任务，建议增加以下配置参数，否则 Master 将退出导致整个恢复进程被更大程度地延迟。

- 增加 namespace 表在线等待超时周期，保证 Master 有足够的时间协调 RegionServer workers split 任务，避免一次次重复相同的任务。

  “hbase.master.namespace.init.timeout”（默认为 3600000ms）
- 通过 RegionServer worker 增加并行 split 任务执行数，保证 RegionServer worker 能并行处理 split work（RegionServer 需要有更多的核心）。在“*客户端安装路径*/HBase/hbase/conf/hbase-site.xml”中添加参数：

  “hbase.regionserver.wal.max.splitters”（默认为 2）
- 如果所有的恢复过程都需要时间，增加初始化监控线程超时时间。

  “hbase.master.initializationmonitor.timeout”（默认为 3600000ms）

8.23.8 客户端查询 HBase 出现 SocketTimeoutException 异常

问题

使用 HBase 客户端操作表数据的时候客户端出现类似如下异常：

```
2015-12-15 02:41:14,054 | WARN  | [task-result-getter-2] | Lost task 2.0 in stage
58.0 (TID 3288, linux-175):
org.apache.hadoop.hbase.client.RetriesExhaustedException: Failed after attempts=36,
exceptions:
Tue Dec 15 02:41:14 CST 2015, null, java.net.SocketTimeoutException:
callTimeout=60000, callDuration=60303:
row 'xxxxxx' on table 'xxxxxx' at
region=xxxxxx,\x05\x1E\x80\x00\x00\x00\x80\x00\x00\x00\x00\x00\x00\x00\x80\x00\x00\
x00\x00\x00\x00\x00\x000\x00\x80\x00\x00\x00\x80\x00\x00\x00\x80\x00\x00,
1449912620868.6a6b7d0c272803d8186930a3bfdb10a9.,
hostname=xxxxxx,16020,1449941841479, seqNum=5
at
org.apache.hadoop.hbase.client.RpcRetryingCallerWithReadReplicas.throwEnrichedExcep
tion(RpcRetryingCallerWithReadReplicas.java:275)
at
org.apache.hadoop.hbase.client.ScannerCallableWithReplicas.call(ScannerCallableWith
Replicas.java:223)
at
org.apache.hadoop.hbase.client.ScannerCallableWithReplicas.call(ScannerCallableWith
Replicas.java:61)
at
```

```
org.apache.hadoop.hbase.client.RpcRetryingCaller.callWithoutRetries(RpcRetryingCall
er.java:200)
at org.apache.hadoop.hbase.client.ClientScanner.call(ClientScanner.java:323)
```

同时，在 RegionServer 上出现类似如下日志：

```
2015-12-15 02:45:44,551 | WARN  | PriorityRpcServer.handler=7,queue=1,port=16020 |
(responseTooSlow):
{"call":"Scan(org.apache.hadoop.hbase.protobuf.generated.ClientProtos$ScanRequest)
","starttimems":1450118730780,"responsesize":416,"method":"Scan","processingtimems"
:13770,"client":"10.91.8.175:41182","queuetimems":0,"class":"HRegionServer"} |
org.apache.hadoop.hbase.ipc.RpcServer.logResponse(RpcServer.java:2221)
2015-12-15 02:45:57,722 | WARN  | PriorityRpcServer.handler=3,queue=1,port=16020 |
(responseTooSlow):
{"call":"Scan(org.apache.hadoop.hbase.protobuf.generated.ClientProtos$ScanRequest)"
,"starttimems":1450118746297,"responsesize":416,
"method":"Scan","processingtimems":11425,"client":"10.91.8.175:41182","queuetimems"
:1746,"class":"HRegionServer"} |
org.apache.hadoop.hbase.ipc.RpcServer.logResponse(RpcServer.java:2221)
2015-12-15 02:47:21,668 | INFO  | LruBlockCacheStatsExecutor | totalSize=7.54 GB,
freeSize=369.52 MB, max=7.90 GB, blockCount=406107,
accesses=35400006, hits=16803205, hitRatio=47.47%, , cachingAccesses=31864266,
cachingHits=14806045, cachingHitsRatio=46.47%,
evictions=17654, evicted=16642283, evictedPerRun=942.69189453125 |
org.apache.hadoop.hbase.io.hfile.LruBlockCache.logStats(LruBlockCache.java:858)
2015-12-15 02:52:21,668 | INFO  | LruBlockCacheStatsExecutor | totalSize=7.51 GB,
freeSize=395.34 MB, max=7.90 GB, blockCount=403080,
accesses=35685793, hits=16933684, hitRatio=47.45%, , cachingAccesses=32150053,
cachingHits=14936524, cachingHitsRatio=46.46%,
evictions=17684, evicted=16800617, evictedPerRun=950.046142578125 |
org.apache.hadoop.hbase.io.hfile.LruBlockCache.logStats(LruBlockCache.java:858)
```

回答

出现该问题的主要原因为 RegionServer 分配的内存过小、Region 数量过大导致在运行过程中内存不足，服务端对客户端的响应过慢。在 RegionServer 的配置文件“hbase-site.xml”中需要调整如下对应的内存分配参数。

表8-30 RegionServer 内存调整参数

参数	描述	默认值
GC_OPTS	在启动参数中给 RegionServer 分配的初始内存和最大内存。	-Xms8G -Xmx8G
hfile.block.cache.size	分配给 HFile/StoreFile 所使用的块缓存的最大 heap（-Xmx setting）的百分比。	当 offheap 关闭时，默认值为 0.25。当 offheap 开启时，默认值是 0.1。

8.23.9 使用 scan 命令仍然可以查询到已修改和已删除的数据

问题

为什么使用如下 **scan** 命令仍然可以查询到已修改和已删除的数据？

```
scan
'<table_name>',{FILTER=>"SingleColumnValueFilter('<column_family>','column',=,'binary:<value>')"}
```

回答

由于 HBase 的可扩展性，在查询表的时候，默认情况下会匹配被查询列的所有版本的值，即使被删除或被修改的值也可以查询出来。对于命中列失败的行（即在某一行中不存在该列），HBase 会将该行查询出来。

如果用户仅需查询该表的最新值和命中列成功的行，可使用如下查询语句：

```
scan
'<table_name>',{FILTER=>"SingleColumnValueFilter('<column_family>','column',=,'binary:<value>',true,true)"}
```

使用该命令，不但可以过滤掉命中列失败的行，而且查询的是表的当前数据的最新版本的值，即不查询被修改之前的值和被删除的值。

说明

过滤器 SingleColumnValueFilter 的相关参数说明如下：

SingleColumnValueFilter(final byte[] family, final byte[] qualifier, final CompareOp compareOp, ByteArrayComparable comparator, final boolean filterIfMissing, final boolean latestVersionOnly)

参数说明：

- family：需要查询的列所在的列族；
- qualifier：需要查询的列；
- compareOp：比较符，如“=”、“>”等等；
- comparator：需要查找的目标值；
- filterIfMissing：如果某一行不存在该列，是否过滤，默认值为 false；
- latestVersionOnly：是否仅查询最新版本的值，默认值为 false。

8.23.10 在启动 HBase shell 时，为什么会抛出“java.lang.UnsatisfiedLinkError: Permission denied”异常

问题

在启动 HBase shell 时，为什么会抛出“java.lang.UnsatisfiedLinkError: Permission denied”异常？

回答

在执行 HBase shell 期间，JRuby 会在“java.io.tmpdir”路径下创建一个临时文件，该路径的默认值为“/tmp”。如果为“/tmp”目录设置 NOEXEC 权限，然后 HBase shell 会启动失败并抛出“java.lang.UnsatisfiedLinkError: Permission denied”异常。

因此，如果为“/tmp”目录设置了 NOEXEC 权限，那么“java.io.tmpdir”必须设置为 HBASE_OPTS/CLIENT_GC_OPTS 中不同的路径。

8.23.11 在 HMaster Web UI 中显示处于“Dead Region Servers”状态的 RegionServer 什么时候会被清除掉

问题

在 HMaster Web UI 中显示处于“Dead Region Servers”状态的 RegionServer 什么时候会被清除掉？

回答

当一个在线的 RegionServer 突然运行停止，会在 HMaster Web UI 中显示处于“Dead Region Servers”状态。当停止运行的 RegionServer 重启并且向 HMaster 上报成功信息，在 HMaster Web UI 中会清除掉“Dead Region Servers”信息。

当 HMaster 主备倒换操作成功执行时，在 HMaster Web UI 中也会清除掉“Dead Region Servers”信息。

以防掌控有一些 region 的主用 HMaster 突然停止响应，备用的 HMaster 将会成为新的主用 HMaster，同时显示先前主用 HMaster 变成 dead RegionServer。当 HMaster 主备倒换操作成功执行，在 HMaster Web UI 中也会清除掉“Dead Region Servers”。

8.23.12 使用 HBase bulkload 导入数据成功，执行相同的查询时却可能返回不同的结果

问题

在使用 HBase bulkload 导入数据时，如果导入的数据存在相同的 rowkey 值，数据可以导入成功，但是执行相同的查询时可能返回不同的结果。

回答

正常情况下，相同 rowkey 值的数据加载到 HBase 是有先后顺序的，HBase 以最近的时间戳的数据为最新数据，一般的默认查询中，没有指定时间戳的，就会对相同 rowkey 值的数据仅返回最新数据。

使用 bulkload 加载数据，由于数据在内存中处理生成 HFile，速度是很快的，很可能出现相同 rowkey 值的数据具有相同时间戳，从而造成查询结果混乱的情况。

建议在建表和数据加载时，设计好 rowkey 值，尽量避免在同一个数据文件中存在相同 rowkey 值的情况。

8.23.13 如何处理由于 Region 处于 FAILED_OPEN 状态而造成的建表失败异常

问题

如何处理由于 Region 处于 FAILED_OPEN 状态而造成的建表失败异常。

回答

建表过程中如果发生网络故障、HDFS 故障或者 Active HMaster 故障等情况时，可能会造成部分 Region 上线失败而处于 FAILED_OPEN 状态，导致建表失败。

由于 Region 上线失败而处于 FAILED_OPEN 状态造成的建表失败异常不能直接修复，需要删除该表后重新建表。

操作步骤如下：

1. 在集群客户端使用如下命令修复表的状态。

   hbase hbck -fixTableStates

2. 进入 HBase shell 并执行以下命令完成表的清理。

   truncate *'<table_name>'*

   disable *'<table_name>'*

   drop *'<table_name>'*

3. 使用建表命令重新创建该表。

8.23.14 如何清理由于建表失败残留在 ZooKeeper 中/hbase/table-lock 目录下的表名

问题

安全模式下，由于建表失败，在 ZooKeeper 的 table-lock 节点（默认路径/hbase/table-lock）下残留有新建的表名，请问该如何清理？

回答

操作步骤如下：

1. 在安装好客户端的环境下，使用 hbase 用户进行 kinit 认证。
2. 执行 ***hbase zkcli*** 命令进入 ZooKeeper 命令行。
3. 在 ZooKeeper 命令行中执行 ***ls /hbase/table***，查看新建的表名是否存在。
   - 是，结束。
   - 否，执行 ***ls /hbase/table-lock*** 查看新建的表名是否存在，若存在新建的表名时使用 ***delete*** 命令（***delete /hbase/table-lock/<table>***，其中<table>为残留的表名）删除该表名。

8.23.15 为什么给 HDFS 上的 HBase 使用的目录设置 quota 会造成 HBase 故障

问题

为什么给 HDFS 上的 HBase 使用的目录设置 quota 会造成 HBase 故障？

回答

表的 flush 操作是在 HDFS 中写 memstore 数据。

如果 HDFS 目录没有足够的磁盘空间 quota，flush 操作会失败，这样 region server 将会终止。

```
Caused by: org.apache.hadoop.hdfs.protocol.DSQuotaExceededException: The DiskSpace
quota of /hbase/data/<namespace>/<tableName> is exceeded: quota = 1024 B = 1 KB but
diskspace consumed = 402655638 B = 384.00 MB
?at
org.apache.hadoop.hdfs.server.namenode.DirectoryWithQuotaFeature.verifyStoragespace
Quota(DirectoryWithQuotaFeature.java:211)
?at
org.apache.hadoop.hdfs.server.namenode.DirectoryWithQuotaFeature.verifyQuota(Direct
oryWithQuotaFeature.java:239)
?at
org.apache.hadoop.hdfs.server.namenode.FSDirectory.verifyQuota(FSDirectory.java:882)
?at
org.apache.hadoop.hdfs.server.namenode.FSDirectory.updateCount(FSDirectory.java:711)
?at
org.apache.hadoop.hdfs.server.namenode.FSDirectory.updateCount(FSDirectory.java:670)
?at
org.apache.hadoop.hdfs.server.namenode.FSDirectory.addBlock(FSDirectory.java:495)
```

上述异常中，表“/hbase/data/<namespace>/<tableName>”的磁盘空间 quota 值为 1KB，但是 memstore 数据为 384.00MB，所以 flush 操作失败并且 region server 会终止。

在 region server 终止时，HMaster 对终止的 region server 的 WAL 文件进行 replay 操作以恢复数据。由于限制了磁盘空间 quota 值，导致 WAL 文件的 replay 操作失败进而导致 HMaster 进程异常退出。

```
2016-07-28 19:11:40,352 | FATAL | MASTER_SERVER_OPERATIONS-10-91-9-131:16000-0 |
Caught throwable while processing event M_SERVER_SHUTDOWN |
org.apache.hadoop.hbase.master.HMaster.abort(HMaster.java:2474)
java.io.IOException: failed log splitting for 10-91-9-131,16020,1469689987884, will
retry
?at
org.apache.hadoop.hbase.master.handler.ServerShutdownHandler.resubmit(ServerShutdow
nHandler.java:365)
?at
org.apache.hadoop.hbase.master.handler.ServerShutdownHandler.process(ServerShutdown
Handler.java:220)
?at org.apache.hadoop.hbase.executor.EventHandler.run(EventHandler.java:129)
?at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
?at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
```

```
?at java.lang.Thread.run(Thread.java:745)
Caused by: java.io.IOException: error or interrupted while splitting logs in
[hdfs://hacluster/hbase/WALs/<RS-Hostname>,<RS-Port>,<startcode>-splitting] Task =
installed = 6 done = 3 error = 3
?at
org.apache.hadoop.hbase.master.SplitLogManager.splitLogDistributed(SplitLogManager.
java:290)
?at
org.apache.hadoop.hbase.master.MasterFileSystem.splitLog(MasterFileSystem.java:402)
?at
org.apache.hadoop.hbase.master.MasterFileSystem.splitLog(MasterFileSystem.java:375)
```

因此，不支持用户对 HDFS 上的 HBase 目录进行 quota 值设置。上述问题可通过下述步骤解决：

步骤 1 在客户端命令提示符下运行 **kinit** *用户名*命令，使 HBase 用户获得安全认证。

步骤 2 运行 **hdfs dfs -count -q** */hbase/data/<namespace>/<tableName>*命令检查分配的磁盘空间 quota。

步骤 3 使用下列命令取消 quota 值限制，恢复 HBase。

hdfs dfsadmin -clrSpaceQuota */hbase/data/<namespace>/<tableName>*

----结束

8.23.16 为什么在使用 OfflineMetaRepair 工具重新构建元数据后，HMaster 启动的时候会等待 namespace 表分配超时，最后启动失败

问题

为什么在使用 OfflineMetaRepair 工具重新构建元数据后，HMaster 启动的时候会等待 namespace 表分配超时，最后启动失败？

且 HMaster 将输出下列 FATAL 消息表示中止：

```
2017-06-15 15:11:07,582 FATAL [Hostname:16000.activeMasterManager] master.HMaster:
Unhandled exception. Starting shutdown.
java.io.IOException: Timedout 120000ms waiting for namespace table to be assigned
      at
org.apache.hadoop.hbase.master.TableNamespaceManager.start(TableNamespaceManager.ja
va:98)
      at org.apache.hadoop.hbase.master.HMaster.initNamespace(HMaster.java:1054)
      at
org.apache.hadoop.hbase.master.HMaster.finishActiveMasterInitialization(HMaster.jav
a:848)
      at org.apache.hadoop.hbase.master.HMaster.access$600(HMaster.java:199)
      at org.apache.hadoop.hbase.master.HMaster$2.run(HMaster.java:1871)
      at java.lang.Thread.run(Thread.java:745)
```

回答

当通过 OfflineMetaRepair 工具重建元数据时，HMaster 在启动期间等待所有 region server 的 WAL 分割，以避免数据不一致问题。一旦 WAL 分割完成，HMaster 将进行用

户 region 的分配。所以当在集群异常的场景下，WAL 分割可能需要很长时间，这取决于多个因素，例如太多的 WALs，较慢的 I/O，region servers 不稳定等。

为确保 HMaster 能够成功完成所有 region server WAL 分割，请执行以下步骤：

1. 确保集群稳定，不存在其他问题。如有任何问题，请先修复。
2. 为“hbase.master.initializationmonitor.timeout”参数配置一个较大的值，默认值为“3600000”毫秒。
3. 重启 HBase 服务。

8.23.17 为什么 splitWAL 期间 HMaster 日志中频繁打印出 FileNotFoundException 及 no lease 信息

问题

当集群重启后会进行 split WAL 操作，在 splitWAL 期间，HMaster 出现不能 close log，日志中频繁打印出 FileNotFoundException 及 no lease 信息。

```
2017-06-10 09:50:27,586 | ERROR | split-log-closeStream-2 | Couldn't close log at
hdfs://hacluster/hbase/data/default/largeT1/2b48346d087275fe751fc049334fda93/recove
red.edits/0000000000000000000.temp |
org.apache.hadoop.hbase.wal.WALSplitter$LogRecoveredEditsOutputSink$2.call(WALSplit
ter.java:1330)
java.io.FileNotFoundException: No lease on
/hbase/data/default/largeT1/2b48346d087275fe751fc049334fda93/recovered.edits/000000
0000000000000.temp (inode 1092653): File does not exist. [Lease.  Holder:
DFSClient_NONMAPREDUCE_1202985678_1, pendingcreates: 1936]
?at
org.apache.hadoop.hdfs.server.namenode.FSNamesystem.checkLease(FSNamesystem.java:34
32)
?at
org.apache.hadoop.hdfs.server.namenode.FSNamesystem.analyzeFileState(FSNamesystem.j
ava:3223)
?at
org.apache.hadoop.hdfs.server.namenode.FSNamesystem.getNewBlockTargets(FSNamesystem
.java:3057)
?at
org.apache.hadoop.hdfs.server.namenode.FSNamesystem.getAdditionalBlock(FSNamesystem
.java:3011)
?at
org.apache.hadoop.hdfs.server.namenode.NameNodeRpcServer.addBlock(NameNodeRpcServer
.java:842)
?at
org.apache.hadoop.hdfs.protocolPB.ClientNamenodeProtocolServerSideTranslatorPB.addB
lock(ClientNamenodeProtocolServerSideTranslatorPB.java:526)
?at
org.apache.hadoop.hdfs.protocol.proto.ClientNamenodeProtocolProtos$ClientNamenodePr
otocol$2.callBlockingMethod(ClientNamenodeProtocolProtos.java)
?at
org.apache.hadoop.ipc.ProtobufRpcEngine$Server$ProtoBufRpcInvoker.call(ProtobufRpcE
ngine.java:616)
?at org.apache.hadoop.ipc.RPC$Server.call(RPC.java:973)
?at org.apache.hadoop.ipc.Server$Handler$1.run(Server.java:2260)
```

```
?at org.apache.hadoop.ipc.Server$Handler$1.run(Server.java:2256)
?at java.security.AccessController.doPrivileged(Native Method)
?at javax.security.auth.Subject.doAs(Subject.java:422)
?at
org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1769)
?at org.apache.hadoop.ipc.Server$Handler.run(Server.java:2254)

?at sun.reflect.GeneratedConstructorAccessor40.newInstance(Unknown Source)
?at
sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAcce
ssorImpl.java:45)
?at java.lang.reflect.Constructor.newInstance(Constructor.java:423)
?at
org.apache.hadoop.ipc.RemoteException.instantiateException(RemoteException.java:106)
?at
org.apache.hadoop.ipc.RemoteException.unwrapRemoteException(RemoteException.java:73)
?at  org.apache.hadoop.hdfs.DataStreamer.locateFollowingBlock(DataStreamer.java:1842)
?at
org.apache.hadoop.hdfs.DataStreamer.nextBlockOutputStream(DataStreamer.java:1639)
?at org.apache.hadoop.hdfs.DataStreamer.run(DataStreamer.java:665)
```

回答

在 splitWAL 的过程中，参数“hbase.splitlog.manager.timeout”控制 splitWAL 的超时时间，若该时间内 splitWAL 无法完成，则会再次提交相同的任务，在一定时间内多次提交了相同的任务，当其中某次任务执行完毕时会删除这个 temp 文件，所以在后来的任务执行时无法找到这个文件，故出现 FileNotFoudException。需做如下调整：

当前“hbase.splitlog.manager.timeout”的默认时间为“600000ms”，集群规格为每个 regionserver 上有 2000~3000 个 region，在集群正常情况下(HBase 无异常，HDFS 无大量的读写操作等)，建议此参数依据集群的规格进行调整，若实际规格（实际平均每个 regonserver 上 region 的个数）大于默认规格（默认平均每个 regionserver 上 region 的个数，即 2000），则调整方案为（实际规格 / 默认规格）* 默认时间。

在服务端的“hbase-site.xml”文件中配置 splitlog 参数，如表 8-31 所示。

表8-31 splitlog 参数说明

参数	描述	默认值
hbase.splitlog.manager.timeout	分布式日志分裂管理程序接收 worker 回应的超时时间	600000

8.23.18 当使用与 Region Server 相同的 Linux 用户但不同的 kerberos 用户时，为什么 ImportTsv 工具执行失败报“Permission denied”的异常

问题

当使用与 Region Server 相同的 Linux 用户（例如 omm 用户）但不同的 kerberos 用户（例如 admin 用户）时，为什么 ImportTsv 工具执行失败报“Permission denied”的异常？

```
Exception in thread "main" org.apache.hadoop.security.AccessControlException:
Permission denied: user=admin, access=WRITE, inode="/user/omm-bulkload/hbase-
staging/partitions_cab16de5-87c2-4153-9cca-a6f4ed4278a6":hbase:hadoop:drwx--x--x
       at
org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.check(FSPermissionChecke
r.java:342)
       at
org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.check(FSPermissionChecke
r.java:315)
       at
org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.checkPermission(FSPermis
sionChecker.java:231)
       at
com.xxx.hadoop.adapter.hdfs.plugin.XXAccessControlEnforce.checkPermission(XXAccessC
ontrolEnforce.java:69)
       at
org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.checkPermission(FSPermis
sionChecker.java:190)
       at
org.apache.hadoop.hdfs.server.namenode.FSDirectory.checkPermission(FSDirectory.java
:1789)
       at
org.apache.hadoop.hdfs.server.namenode.FSDirectory.checkPermission(FSDirectory.java
:1773)
       at
org.apache.hadoop.hdfs.server.namenode.FSDirectory.checkAncestorAccess(FSDirectory.
java:1756)
       at
org.apache.hadoop.hdfs.server.namenode.FSNamesystem.startFileInternal(FSNamesystem.
java:2490)
       at
org.apache.hadoop.hdfs.server.namenode.FSNamesystem.startFileInt(FSNamesystem.java:
2425)
       at
org.apache.hadoop.hdfs.server.namenode.FSNamesystem.startFile(FSNamesystem.java:230
8)
       at
org.apache.hadoop.hdfs.server.namenode.NameNodeRpcServer.create(NameNodeRpcServer.j
ava:745)
       at
org.apache.hadoop.hdfs.protocolPB.ClientNamenodeProtocolServerSideTranslatorPB.crea
te(ClientNamenodeProtocolServerSideTranslatorPB.java:434)
       at
```

```
org.apache.hadoop.hdfs.protocol.proto.ClientNamenodeProtocolProtos$ClientNamenodePr
otocol$2.callBlockingMethod(ClientNamenodeProtocolProtos.java)
       at
org.apache.hadoop.ipc.ProtobufRpcEngine$Server$ProtoBufRpcInvoker.call(ProtobufRpcE
ngine.java:616)
       at org.apache.hadoop.ipc.RPC$Server.call(RPC.java:973)
       at org.apache.hadoop.ipc.Server$Handler$1.run(Server.java:2260)
       at org.apache.hadoop.ipc.Server$Handler$1.run(Server.java:2256)
       at java.security.AccessController.doPrivileged(Native Method)
       at javax.security.auth.Subject.doAs(Subject.java:422)
       at
org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1781)
       at org.apache.hadoop.ipc.Server$Handler.run(Server.java:2254)
```

回答

ImportTsv 工具在“*客户端安装路径*/HBase/hbase/conf/hbase-site.xml”文件中“hbase.fs.tmp.dir”参数所配置的 HBase 临时目录中创建 partition 文件。因此客户端（kerberos 用户）应该在指定的临时目录上具有 rwx 的权限来执行 ImportTsv 操作。“hbase.fs.tmp.dir”参数的默认值为“/user/${user.name}/hbase-staging”（例如“/user/omm/hbase-staging”），此处“$ {user.name}”是操作系统用户名（即 omm 用户），客户端（kerberos 用户，例如 admin 用户）不具备该目录的 rwx 权限。

上述问题可通过执行以下步骤解决：

1. 在客户端将“hbase.fs.tmp.dir”参数设置为当前 kerberos 用户的目录（如“/user/admin/hbase-staging”），或者为客户端（kerberos 用户）提供已配置的目录所必需的 rwx 权限。
2. 重试 ImportTsv 操作。

8.23.19 租户访问 Phoenix 提示权限不足

问题

使用租户访问 Phoenix 提示权限不足。

回答

创建租户的时候需要关联 HBase 服务和 Yarn 队列。

租户要操作 Phoenix 还需要额外操作的权限，即 Phoenix 系统表的 RWX 权限。

例如：

创建好的租户为 **hbase**，使用 **admin** 用户登录 hbase shell，执行 **scan 'hbase:acl'**命令查询租户对应的角色为 **hbase_1450761169920**（格式为：租户名_时间戳）。

执行以下命令进行授权（如果还没有生成 Phoenix 系统表，请用 **admin** 用户登录 Phoenix 客户端后再回到 hbase shell 里授权）：

grant '@hbase_1450761169920','RWX','*SYSTEM.CATALOG*'

grant '@hbase_1450761169920','RWX','*SYSTEM.FUNCTION*'

grant '@hbase_1450761169920','RWX','*SYSTEM.SEQUENCE*'

grant '@hbase_1450761169920','RWX','*SYSTEM.STATS*'

新建用户 **phoenix** 并绑定租户 **hbase**，该用户 phoenix 就可以用来访问 Phoenix 客户端。

8.23.20 租户使用 HBase bulkload 功能提示权限不足

问题

租户使用 HBase bulkload 功能提示权限不足。

回答

创建租户的时候需要关联 HBase 服务和 Yarn 队列。

例如：

新建用户 **user** 并绑定租户同名的角色。

用户 **user** 需要使用 bulkload 功能还需要额外权限。

以下以用户 user 为例：

参见“批量导入数据”章节举例，以下是一些差异点。

1. 将数据文件目录建在“/tmp”目录下，执行以下命令：

   hdfs dfs -mkdir /tmp/datadirImport

   hdfs dfs -put data.txt /tmp/datadirImport

2. 生成 HFile 的时候使用 HDFS 的“/tmp”目录：

   hbase com.xxx.hadoop.hbase.tools.bulkload.ImportData -Dimport.skip.bad.lines=true -Dimport.separator=',' -Dimport.bad.lines.output=/tmp/badline -Dimport.hfile.output=/tmp/hfile configuration.xml ImportTable /tmp/datadirImport

3. 导入 HFile 的时候使用 HDFS 的“/tmp”目录：

   hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles /tmp/hfile ImportTable

8.23.21 如何解决 HBase 恢复数据任务失败后错误详情中提示：Rollback recovery failed 的回滚失败问题

问题

HBase 恢复任务执行失败后系统自动回滚数据，若页面详情中提示“Rollback recovery failed”信息，表示回滚失败。由于回滚失败后就不会处理数据，所以有可能产生垃圾数据，需要如何解决？

回答

在下次执行备份或恢复任务前，需要手动清除这些垃圾数据。

步骤 1 安装集群客户端，例如安装目录为“/opt/client”。

步骤 2 使用客户端安装用户，执行 **source /opt/client/bigdata_env** 命令配置环境变量。

步骤 3 执行 **kinit admin** 认证管理员身份。

步骤 4 执行 **zkCli.sh -server *ZooKeeper 节点业务 IP 地址*:2181** 连接 ZooKeeper。

步骤 5 执行 **deleteall /recovering** 删除垃圾数据。然后执行 **quit** 退出 ZooKeeper 连接。

说明

执行该命令会导致数据丢失，请谨慎操作。

步骤 6 执行 **hdfs dfs -rm -f -r /user/hbase/backup** 删除临时数据。

步骤 7 登录 FusionInsight Manager 界面，选择“运维 > 备份恢复 > 恢复管理”，在任务列表中对应任务的“操作”列，单击“查询历史”，在弹出的窗口中，在指定一次执行记录前单击，即可查看相关的快照名称信息：

```
Snapshot [ snapshot name ] is created successfully before recovery.
```

步骤 8 切换到客户端，执行 **hbase shell**，然后运行 **delete_all_snapshot** '*snapshot name*.*'删除临时快照。

----结束

8.23.22 如何修复 Region Overlap

问题

MRS 3.x 及之后版本，使用 HBck 工具检查 Region 状态，若日志中存在“ERROR: (regions region1 and region2) There is an overlap in the region chain.”或者“ERROR: (region region1) Multiple regions have the same startkey: xxx”信息，表示某些 Region 存在 Overlap 的问题，需要如何解决？

回答

修复步骤如下：

步骤 1 执行 **hbase hbck -repair** *tableName* 命令修复存在 overlap 的表。

步骤 2 执行 **hbase hbck** *tableName* 命令检查修复的表是否还存在 overlap。

- 如果不存在 overlap，执行步骤 3。
- 如果存在 overlap，执行步骤 1。

步骤 3 登录 FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > HBase > 更多 > 执行 HMaster 倒换”，完成 HMaster 主备倒换。

步骤 4 执行 **hbase hbck** *tableName* 命令检查修复的表是否还存在 overlap。

- 如果不存在 overlap，修复完成。
- 如果存在 overlap，从步骤 1 开始重新执行修复步骤。

----结束

8.23.23 HBase RegionServer GC 参数 Xms，Xmx 配置 31G，导致 RegionServer 启动失败

问题

MRS 3.x 及之后版本，查看 RegionServer 启动失败节点的 hbase-omm-*.out 日志，发现日志中存在“An error report file with more information is saved as: /tmp/hs_err_pid*.log”，查看/tmp/hs_err_pid*.log 发现日志存在“#Internal Error (vtableStubs_aarch64.cpp:213), pid=9456, tid=0x0000ffff97fdd200”和“#guarantee(__ pc() <= s->code_end()) failed: overflowed buffer”，表示此问题是由 JDK 导致，需要如何解决？

回答

修复步骤如下：

步骤 1 在 RegionServer 启动失败的某个节点执行 **su - omm**，切换到 **omm** 用户。

步骤 2 在 omm 用户下执行 **java -XX:+PrintFlagsFinal -version |grep HeapBase**，出现如下类似结果。

```
uintx HeapBaseMinAddress = 2147483648 {pd product}
```

步骤 3 修改“GC_OPTS”中“-Xms”和“-Xmx”的值使其不在 32G-HeapBaseMinAddress 和 32G 的值之间，不包括 32G 和 32G-HeapBaseMinAddress 的值。

步骤 4 登录 FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > HBase > 实例”，选择失败实例，选择“更多 > 重启实例”来重启失败实例。

----结束

8.23.24 使用集群内节点执行批量导入，为什么 LoadIncrementalHFiles 工具执行失败报“Permission denied”的异常

问题

在普通集群中手动创建 Linux 用户，并使用集群内 DataNode 节点执行批量导入时，为什么 LoadIncrementalHFiles 工具执行失败报“Permission denied”的异常？

```
2020-09-20 14:53:53,808 WARN  [main] shortcircuit.DomainSocketFactory: error
creating DomainSocket
java.net.ConnectException: connect(2) error: Permission denied when trying to
connect to '/var/run/FusionInsight-HDFS/dn_socket'
    at org.apache.hadoop.net.unix.DomainSocket.connect0(Native Method)
    at org.apache.hadoop.net.unix.DomainSocket.connect(DomainSocket.java:256)
    at
```

```
org.apache.hadoop.hdfs.shortcircuit.DomainSocketFactory.createSocket(DomainSocketFa
ctory.java:168)
    at
org.apache.hadoop.hdfs.client.impl.BlockReaderFactory.nextDomainPeer(BlockReaderFac
tory.java:804)
    at
org.apache.hadoop.hdfs.client.impl.BlockReaderFactory.createShortCircuitReplicaInfo
(BlockReaderFactory.java:526)
    at
org.apache.hadoop.hdfs.shortcircuit.ShortCircuitCache.create(ShortCircuitCache.java
:785)
    at
org.apache.hadoop.hdfs.shortcircuit.ShortCircuitCache.fetchOrCreate(ShortCircuitCac
he.java:722)
    at
org.apache.hadoop.hdfs.client.impl.BlockReaderFactory.getBlockReaderLocal(BlockRead
erFactory.java:483)
    at
org.apache.hadoop.hdfs.client.impl.BlockReaderFactory.build(BlockReaderFactory.java
:360)
   at  org.apache.hadoop.hdfs.DFSInputStream.getBlockReader(DFSInputStream.java:663)
    at org.apache.hadoop.hdfs.DFSInputStream.blockSeekTo(DFSInputStream.java:594)
    at
org.apache.hadoop.hdfs.DFSInputStream.readWithStrategy(DFSInputStream.java:776)
    at org.apache.hadoop.hdfs.DFSInputStream.read(DFSInputStream.java:845)
    at java.io.DataInputStream.readFully(DataInputStream.java:195)
    at
org.apache.hadoop.hbase.io.hfile.FixedFileTrailer.readFromStream(FixedFileTrailer.j
ava:401)
    at org.apache.hadoop.hbase.io.hfile.HFile.isHFileFormat(HFile.java:651)
    at org.apache.hadoop.hbase.io.hfile.HFile.isHFileFormat(HFile.java:634)
    at
org.apache.hadoop.hbase.tool.LoadIncrementalHFiles.visitBulkHFiles(LoadIncrementalH
Files.java:1090)
    at
org.apache.hadoop.hbase.tool.LoadIncrementalHFiles.discoverLoadQueue(LoadIncrementa
lHFiles.java:1006)
    at
org.apache.hadoop.hbase.tool.LoadIncrementalHFiles.prepareHFileQueue(LoadIncrementa
lHFiles.java:257)
    at
org.apache.hadoop.hbase.tool.LoadIncrementalHFiles.doBulkLoad(LoadIncrementalHFiles
.java:364)
    at
org.apache.hadoop.hbase.tool.LoadIncrementalHFiles.run(LoadIncrementalHFiles.java:1
263)
    at
org.apache.hadoop.hbase.tool.LoadIncrementalHFiles.run(LoadIncrementalHFiles.java:1
276)
    at
org.apache.hadoop.hbase.tool.LoadIncrementalHFiles.run(LoadIncrementalHFiles.java:1
311)
    at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:76)
    at
```

```
org.apache.hadoop.hbase.tool.LoadIncrementalHFiles.main(LoadIncrementalHFiles.java:1333)
```

回答

如果 LoadIncrementalHFiles 工具依赖的 Client 在集群内安装，且和 DataNode 在相同的节点上，在工具执行过程中 HDFS 会创建短路读提高性能。短路读依赖“/var/run/FusionInsight-HDFS”目录(“dfs.domain.socket.path”)，该目录默认权限是750。而当前 Linux 用户没有权限操作该目录。

上述问题可通过执行以下方法解决：

方法一：创建新用户(推荐使用)。

步骤 1 通过 Manager 页面创建新的用户，该用户属组中默认包含 ficommon 组。

```
[root@xxx-xxx-xxx-xxx ~]# id test
uid=20038(test) gid=9998(ficommon) groups=9998(ficommon)
```

步骤 2 重新执行 ImportData。

----结束

方法二：修改当前用户的属组。

步骤 1 将该用户添加到 ficommon 组中。

```
[root@xxx-xxx-xxx-xxx ~]# usermod -a -G ficommon test
[root@xxx-xxx-xxx-xxx ~]# id test
uid=2102(test) gid=2102(test) groups=2102(test),9998(ficommon)
```

步骤 2 重新执行 ImportData。

----结束

8.23.25 Phoenix sqlline 脚本使用，报 import argparse 错误

问题

在客户端使用 sqlline 脚本时，报 import argparse 错误。

回答

步骤 1 以 **root** 用户登录安装 HBase 客户端的节点，使用 **hbase** 用户进行安全认证。

步骤 2 进入 HBase 客户端 sqlline 脚本所在目录执行 **python3 sqlline.py** 命令。

----结束

8.23.26 Phoenix BulkLoad Tool 限制

问题

当更新索引字段数据时，若用户表已经存在一批数据，则 BulkLoad 工具不能更新全局和局部可变索引。

回答

问题分析

1. 创建表。

```
CREATE TABLE TEST_TABLE(
DATE varchar not null,
NUM integer not null,
SEQ_NUM integer not null,
ACCOUNT1 varchar not null,
ACCOUNTDES varchar,
FLAG varchar,
SALL double,
CONSTRAINT PK PRIMARY KEY (DATE,NUM,SEQ_NUM,ACCOUNT1)
);
```

2. 创建全局索引

   CREATE INDEX TEST_TABLE_INDEX ON TEST_TABLE(ACCOUNT1,DATE,NUM,ACCOUNTDES,SEQ_NUM);

3. 插入数据

   UPSERT INTO TEST_TABLE (DATE,NUM,SEQ_NUM,ACCOUNT1,ACCOUNTDES,FLAG,SALL) values ('20201001',30201001,13,'367392332','sffa1','','');

4. 执行 BulkLoad 任务更新数据

   hbase org.apache.phoenix.mapreduce.CsvBulkLoadTool -t TEST_TABLE -i /tmp/test.csv，test.csv 内容如下：

20201001	30201001	13	367392332	sffa888	1231243	23

5. 问题现象：无法直接更新之前存在的索引数据，导致存在两条索引数据。

```
+------------+-----------+-----------+---------------+----------------+
| :ACCOUNT1  |   :DATE   |   :NUM    | 0:ACCOUNTDES  | :SEQ_NUM  |
+------------+-----------+-----------+---------------+----------------+
| 367392332  | 20201001  | 30201001  | sffa1         |  13               |
| 367392332  | 20201001  | 30201001  | sffa888      | 13               |
+------------+-----------+-----------+---------------+----------------+
```

解决方法

步骤 1 删除旧的索引表。

DROP INDEX TEST_TABLE_INDEX ON TEST_TABLE;

步骤 2 异步方式创建新的索引表。

CREATE INDEX TEST_TABLE_INDEX ON TEST_TABLE(ACCOUNT1,DATE,NUM,ACCOUNTDES,SEQ_NUM) ASYNC;

步骤 3 索引重建。

hbase org.apache.phoenix.mapreduce.index.IndexTool --data-table TEST_TABLE --index-table TEST_TABLE_INDEX --output-path /user/test_table

----结束

8.23.27 CTBase 对接 Ranger 权限插件，提示权限不足

问题

CTBase 访问启用 Ranger 插件的 HBase 服务时，如果创建聚簇表，提示权限不足。

```
ERROR: Create ClusterTable failed. Error:
org.apache.hadoop.hbase.security.AccessDeniedException: Insufficient permissions
for user 'ctbase2@HADOOP.COM' (action=create)
at
org.apache.ranger.authorization.hbase.AuthorizationSession.publishResults(Authoriza
tionSession.java:278)
at
org.apache.ranger.authorization.hbase.RangerAuthorizationCoprocessor.authorizeAcces
s(RangerAuthorizationCoprocessor.java:654)
at
org.apache.ranger.authorization.hbase.RangerAuthorizationCoprocessor.requirePermiss
ion(RangerAuthorizationCoprocessor.java:772)
at
org.apache.ranger.authorization.hbase.RangerAuthorizationCoprocessor.preCreateTable
(RangerAuthorizationCoprocessor.java:943)
at
org.apache.ranger.authorization.hbase.RangerAuthorizationCoprocessor.preCreateTable
(RangerAuthorizationCoprocessor.java:428)
at
org.apache.hadoop.hbase.master.MasterCoprocessorHost$12.call(MasterCoprocessorHost.
java:351)
at
org.apache.hadoop.hbase.master.MasterCoprocessorHost$12.call(MasterCoprocessorHost.
java:348)
at
org.apache.hadoop.hbase.coprocessor.CoprocessorHost$ObserverOperationWithoutResult.
callObserver(CoprocessorHost.java:581)
at
org.apache.hadoop.hbase.coprocessor.CoprocessorHost.execOperation(CoprocessorHost.j
ava:655)
at
org.apache.hadoop.hbase.master.MasterCoprocessorHost.preCreateTable(MasterCoprocess
orHost.java:348)
at org.apache.hadoop.hbase.master.HMaster$5.run(HMaster.java:2192)
at
org.apache.hadoop.hbase.master.procedure.MasterProcedureUtil.submitProcedure(Master
ProcedureUtil.java:134)
```

```
at org.apache.hadoop.hbase.master.HMaster.createTable(HMaster.java:2189)
at
org.apache.hadoop.hbase.master.MasterRpcServices.createTable(MasterRpcServices.java
:711)
at
org.apache.hadoop.hbase.shaded.protobuf.generated.MasterProtos$MasterService$2.call
BlockingMethod(MasterProtos.java)
at org.apache.hadoop.hbase.ipc.RpcServer.call(RpcServer.java:458)
at org.apache.hadoop.hbase.ipc.CallRunner.run(CallRunner.java:133)
at org.apache.hadoop.hbase.ipc.RpcExecutor$Handler.run(RpcExecutor.java:338)
at org.apache.hadoop.hbase.ipc.RpcExecutor$Handler.run(RpcExecutor.java:318)
```

回答

CTBase 用户在 Ranger 界面配置权限策略，赋予 CTBase 元数据表_ctmeta_、聚簇表和索引表 RWCAE（READ，WRITE，EXEC，CREATE，ADMIN）权限。

9 使用 HDFS

9.1 从零开始使用 Hadoop

本章节提供从零开始使用 Hadoop 提交 wordcount 作业的操作指导，wordcount 是最经典的 Hadoop 作业，它用来统计海量文本的单词数量。

操作步骤

步骤 1 准备 wordcount 程序。

开源的 Hadoop 的样例程序包含多个例子，其中包含 wordcount。可以从 https://dist.apache.org/repos/dist/release/hadoop/common/中下载 Hadoop 的样例程序。

例如，选择 hadoop-2.10.x 版本，下载“hadoop-2.10.x.tar.gz”，解压后在“hadoop-2.10.x\share\hadoop\mapreduce”路径下获取“hadoop-mapreduce-examples-2.10.x.jar”，即为 Hadoop 的样例程序。“hadoop-mapreduce-examples-2.10.x.jar”样例程序包含了 wordcount 程序。

说明

hadoop-2.10.x 表示 Hadoop 的版本号。

步骤 2 准备数据文件。

数据文件无格式要求，准备一个或多个 txt 文件即可，如下内容为 txt 文件样例：

```
qwsdfhoedfrffrofhuncckgktpmhutopmma
jjpsffjfjorgjgtyiuyjmhombmbogohoyhm
jhheyeombdhuaqqiquyebchdhmamdhdemmj
doeyhjwedcrfvtgbmojiyhhqssddddddfkf
kjhhjkehdeiyrudjhfhfhffooqweopuyyyy
```

步骤 3 上传数据至 OBS。

1. 登录 OBS 控制台。
2. 单击“并行文件系统 > 创建并行文件系统”，创建一个名称为 wordcount01 的文件系统。

   wordcount01 仅为示例，文件系统名称必须全局唯一，否则会创建并行文件系统失败。

3. 在 OBS 文件系统列表中单击文件系统名称 wordcount01，选择“文件 > 新建文件夹”，分别创建 program、input 文件夹。
   - program：存放用户程序
   - input：存放用户数据文件
4. 进入 program 文件夹，选择“上传文件 > 添加文件”，从本地选择步骤 1 中下载的程序包，然后单击“上传”。
5. 进入 input 文件夹，将步骤 2 中准备的数据文件上传到 input 文件夹。

步骤 4 登录 MRS 控制台，在左侧导航栏选择“集群列表 > 现有集群”，单击集群名称，该集群需要包含 Hadoop 组件。

步骤 5 提交 wordcount 作业。

在 MRS 控制台选择“作业管理”页签，单击“添加”，进入“添加作业”页面，具体请参见“用户指南 > 管理现有集群 > 作业管理 > 运行 MapReduce 作业”。

- 作业类型选择“MapReduce”。
- 作业名称为“mr_01”。
- 执行程序路径配置为 OBS 上存放程序的地址。例如：obs://wordcount01/program/hadoop-mapreduce-examples-2.10.x.jar。
- 执行程序参数中填写的参数为：wordcount obs://wordcount01/input/ obs://wordcount01/output/。

  说明

  - 参数“**obs://wordcount01/input/**”中的 OBS 文件系统名需要替换为实际环境创建的文件系统名。
  - 参数“**obs://wordcount01/output/**”中的 OBS 文件系统名需要替换为实际环境创建的文件系统名，目录 output 请手动输入一个不存在的目录。
- 服务配置参数无需填写。

只有集群处于“运行中”状态时才能提交作业。

作业提交成功后默认为“已接受”状态，不需要用户手动执行作业。

步骤 6 查看作业执行结果。

1. 进入“作业管理”页面，查看作业是否执行完成。

   作业运行需要时间，作业运行结束后，刷新作业列表。

   作业执行成功或失败后都不能再次执行，只能新增或者复制作业，配置作业参数后重新提交作业。
2. 登录 OBS 控制台，进入 OBS 路径，查看作业输出信息。

   进入到步骤 5 中创建的 output 路径查看相关的 output 文件，需要下载到本地以文本方式打开进行查看。

----结束

9.2 配置内存管理

配置场景

在 HDFS 中，每个文件对象都需要在 NameNode 中注册相应的信息，并占用一定的存储空间。随着文件数的增加，当原有的内存空间无法存储相应的信息时，需要修改内存大小的设置。

配置描述

参数入口：

请参考修改集群服务配置参数，进入 HDFS“全部配置”页面。

表9-1 参数说明

配置参数	说明	默认值
GC_PROFILE	NameNode 所占内存主要由 FsImage 大小决定。FsImage Size = 文件数 * 900 Bytes，根据计算结果可估算 hdfs 的 NameNode 应设内存大小。 该参数项的内存大小取值如下： • high：4G • medium：2G • low：256M • custom：根据实际数据量大小在 GC_OPTS 中设置内存大小。	custom
GC_OPTS	JVM 用于 gc 的参数。仅当 GC_PROFILE 设置为 custom 时该配置才会生效。需确保 GC_OPT 参数设置正确，否则进程启动会失败。 须知 请谨慎修改该项。如果配置不当，将造成服务不可用。	-Xms2G -Xmx4G -XX:NewSize=128M -XX:MaxNewSize=256M -XX:MetaspaceSize=128M -XX:MaxMetaspaceSize=128M -XX:+UseConcMarkSweepGC -XX:+CMSParallelRemarkEnabled -XX:CMSInitiatingOccupancyFraction=65 -XX:+PrintGCDetails -Dsun.rmi.dgc.client.gcInterval=0x7FFFFFFFFFFFFE -Dsun.rmi.dgc.server.gcInterval=0x7FFFFFFFFFFFFE -XX:-OmitStackTraceInFastThrow -XX:+PrintGCDateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=1M -

配置参数	说明	默认值
		Djdk.tls.ephemeralDHKeySize=2048

9.3 创建 HDFS 角色

操作场景

该任务指导系统管理员在 FusionInsight Manager 创建并设置 HDFS 的角色。HDFS 角色可设置 HDFS 目录或文件的读、写和执行权限。

用户在 HDFS 中对自己创建的目录或文件拥有完整权限，可直接读取、写入以及授权他人访问此 HDFS 目录与文件。

说明

- 本章节适用于 MRS 3.x 及后续版本。
- 安全模式支持创建 HDFS 角色，普通模式不支持创建 HDFS 角色。
- 如果当前组件使用了 Ranger 进行权限控制，须基于 Ranger 配置 HDFS 相关策略进行权限管理，具体操作可参考添加 HDFS 的 Ranger 访问权限策略。

前提条件

系统管理员已明确业务需求。

操作步骤

步骤 1 登录 FusionInsight Manager，选择“系统 > 权限 > 角色”。

步骤 2 单击“添加角色”，然后在“角色名称”和“描述”中输入角色名字与描述。

步骤 3 配置资源权限，请参见表 9-2。

“文件系统”：HDFS 中的目录和文件授权。

HDFS 常见目录如下：

- “flume”：Flume 数据存储目录。
- “hbase”：HBase 数据存储目录。
- “mr-history”：MapReduce 任务信息存储目录。
- “tmp”：临时数据存储目录。
- “user”：用户数据存储目录。

表9-2 设置角色

任务场景	角色授权操作

任务场景	角色授权操作
设置 HDFS 管理员权限	在“配置资源权限”的表格中选择“*待操作集群的名称* > HDFS”，勾选“集群管理操作权限 ”。 说明 设置 HDFS 管理员权限需要重启 HDFS 服务才可生效。
设置用户执行 HDFS 检查和 HDFS 修复的权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > HDFS > 文件系统”。 2. 定位到指定目录或文件在 HDFS 中保存的位置。 3. 在指定目录或文件的“权限”列，勾选“读”和“执行”。
设置用户读取其他用户的目录或文件的权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > HDFS > 文件系统”。 2. 定位到指定目录或文件在 HDFS 中保存的位置。 3. 在指定目录或文件的“权限”列，勾选“读”和“执行”。
设置用户在其他用户的文件写入数据的权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > HDFS > 文件系统”。 2. 定位到指定文件在 HDFS 中保存的位置。 3. 在指定文件的“权限”列，勾选“写”和“执行”。
设置用户在其他用户的目录新建或删除子文件、子目录的权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > HDFS > 文件系统”。 2. 定位到指定目录在 HDFS 中保存的位置。 3. 在指定目录的“权限”列，勾选“写”和“执行”。
设置用户在其他用户的目录或文件执行的权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > HDFS > 文件系统”。 2. 定位到指定目录或文件在 HDFS 中保存的位置。 3. 在指定目录或文件的“权限”列，勾选“执行”。
设置子目录继承上级目录权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > HDFS > 文件系统”。 2. 定位到指定目录或文件在 HDFS 中保存的位置。 3. 在指定目录或文件的“权限”列，勾选“递归”。

步骤 4 单击“确定”完成，返回“角色”。

----结束

9.4 使用 HDFS 客户端

操作场景

该任务指导用户在运维场景或业务场景中使用 HDFS 客户端。

前提条件

- 已安装客户端。

  例如安装目录为“/opt/hadoopclient”，以下操作的客户端目录只是举例，请根据实际安装目录修改。
- 各组件业务用户由系统管理员根据业务需要创建。安全模式下，“机机”用户需要下载 keytab 文件。“人机”用户第一次登录时需修改密码。（普通模式不涉及）

使用 HDFS 客户端

步骤 1 以客户端安装用户，登录安装客户端的节点。

步骤 2 执行以下命令，切换到客户端安装目录。

cd /opt/hadoopclient

步骤 3 执行以下命令配置环境变量。

source bigdata_env

步骤 4 如果集群为安全模式，执行以下命令进行用户认证。普通模式集群无需执行用户认证。

kinit *组件业务用户*

步骤 5 直接执行 HDFS Shell 命令。例如：

hdfs dfs -ls /

----结束

HDFS 客户端常用命令

常用的 HDFS 客户端命令如下表所示。

更多命令可参考 https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/CommandsManual.html#User_Commands

表9-3 HDFS 客户端常用命令

命令	说明	样例
hdfs dfs -mkdir *文件夹名称*	创建文件夹	**hdfs dfs -mkdir /tmp/mydir**
hdfs dfs -ls *文件夹名称*	查看文件夹	**hdfs dfs -ls /tmp**

命令	说明	样例
hdfs dfs -put *客户端节点上本地文件 HDFS 指定路径*	上传本地文件到 HDFS 指定路径	**hdfs dfs -put /opt/test.txt /tmp** 上传客户端节点“/opt/test.txt”文件到 HDFS 的“/tmp”路径下
hdfs dfs -get *hdfs 指定文件 客户端节点上指定路径*	下载 HDFS 文件到本地指定路径	**hdfs dfs -get /tmp/test.txt /opt/** 下载 HDFS 的“/tmp/test.txt”文件到客户端节点的“/opt”路径下
hdfs dfs -rm -r -f *hdfs 指定文件夹*	删除文件夹	**hdfs dfs -rm -r -f /tmp/mydir**
hdfs dfs -chmod *权限参数 文件目录*	为用户设置 HDFS 目录权限	**hdfs dfs -chmod 700 /tmp/test**

客户端常见使用问题

1. 当执行 HDFS 客户端命令时，客户端程序异常退出，报“java.lang.OutOfMemoryError”的错误。

   这个问题是由于 HDFS 客户端运行时的所需的内存超过了 HDFS 客户端设置的内存上限（默认为 128MB）。可以通过修改“<*客户端安装路径*>/HDFS/component_env”中的“CLIENT_GC_OPTS”来修改 HDFS 客户端的内存上限。例如，需要设置该内存上限为 1GB，则设置：

   ```
   CLIENT_GC_OPTS="-Xmx1G"
   ```

   在修改完后，使用如下命令刷新客户端配置，使之生效：

   source <*客户端安装路径*>**/bigdata_env**

2. 如何设置 HDFS 客户端运行时的日志级别？

   HDFS 客户端运行时的日志是默认输出到 Console 控制台的，其级别默认是 INFO 级别。有的时候为了定位问题，需要开启 DEBUG 级别日志，可以通过导出一个环境变量来设置，命令如下：

   export HADOOP_ROOT_LOGGER=DEBUG,console

   在执行完上面命令后，再执行 HDFS Shell 命令时，即可打印出 DEBUG 级别日志。

   如果想恢复 INFO 级别日志，可执行如下命令：

   export HADOOP_ROOT_LOGGER=INFO,console

3. 如何彻底删除 HDFS 文件？

   由于 HDFS 的回收站机制，一般删除 HDFS 文件后，文件会移动到 HDFS 的回收站中。如果确认文件不再需要并且需要立马释放存储空间，可以继续清理对应的回收站目录（例如：hdfs://hacluster/user/*xxx*/.Trash/Current/*xxx*）。

9.5 使用 distcp 命令

操作场景

distcp 是一种在集群间或集群内部拷贝大量数据的工具。它利用 MapReduce 任务实现大量数据的分布式拷贝。

前提条件

- 已安装 Yarn 客户端或者包括 Yarn 的客户端。例如安装目录为“/opt/client”。
- 各组件业务用户由系统管理员根据业务需要创建。安全模式下，“机机”用户需要下载 keytab 文件。“人机”用户第一次登录时需修改密码。（普通模式不涉及）
- 如需在集群间拷贝数据，拷贝数据的集群双方都需要启用集群间拷贝数据功能。

操作步骤

步骤 1 登录安装客户端的节点。

步骤 2 执行以下命令，切换到客户端安装目录。

cd /opt/client

步骤 3 执行以下命令配置环境变量。

source bigdata_env

步骤 4 如果集群为安全模式，执行 distcp 命令的用户所属的用户组必须为 **supergroup** 组，且执行以下命令进行用户认证。普通模式集群无需执行用户认证。

kinit *组件业务用户*

步骤 5 直接执行 distcp 命令。例如：

hadoop distcp hdfs://hacluster/source hdfs://hacluster/target

----结束

distcp 常见用法

1. 最常见的 distcp 用法，示例如下：

```
hadoop distcp -numListstatusThreads 40 -update -delete -prbugpaxtq
hdfs://cluster1/source hdfs://cluster2/target
```

说明

在上述命令中：

- -numListstatusThreads 指定了 40 个构建被拷贝文件的列表的线程数；
- -update -delete 表示将源位置和目标位置的文件同步，删除掉目标位置多余的文件，注意如果需要增量拷贝文件，请将-delete 删掉；
- -prbugpaxtq 与-update 配合，表示被拷贝文件的状态信息也会被更新；
- hdfs://cluster1/source、hdfs://cluster2/target 分别表示源位置和目标位置。

2. 集群间的数据拷贝，示例如下：

```
hadoop distcp hdfs://cluster1/foo/bar hdfs://cluster2/bar/foo
```

说明

集群 cluster1 和集群 cluster2 之间的网络必须保持互通，且两个集群需要使用相同或兼容的 HDFS 版本。

3. 多个源目录的数据拷贝，示例如下：

```
hadoop distcp hdfs://cluster1/foo/a \
hdfs://cluster1/foo/b \
hdfs://cluster2/bar/foo
```

上面的命令的效果是将集群 cluster1 的文件夹 a、b 拷贝到集群 cluster2 的“/bar/foo”目录下，它的效果等效于下面的命令：

```
hadoop distcp -f hdfs://cluster1/srclist \
hdfs://cluster2/bar/foo
```

其中 srclist 里面的内容如下。注意运行 distcp 命令前，需要将 srclist 文件上传到 HDFS 上。

```
hdfs://cluster1/foo/a
hdfs://cluster1/foo/b
```

4. update 和 overwrite 选项的用法，-update 用于被拷贝的文件在目标位置中不存在，或者更新目标位置中被拷贝文件的内容；-overwrite 用于覆盖在目标位置中已经存在的文件。

不加选项和加两个选项中任一个选项的区别，示例如下：

假设，源位置的文件结构如下：

```
hdfs://cluster1/source/first/1
hdfs://cluster1/source/first/2
hdfs://cluster1/source/second/10
hdfs://cluster1/source/second/20
```

不加选项的命令：

```
hadoop distcp hdfs://cluster1/source/first  hdfs://cluster1/source/second
hdfs://cluster2/target
```

上述命令默认会在目标位置创建文件夹 first、second，所以拷贝结果如下：

```
hdfs://cluster2/target/first/1
hdfs://cluster2/target/first/2
hdfs://cluster2/target/second/10
hdfs://cluster2/target/second/20
```

加两个选项中任一个选项的命令，例如加 update 选项：

```
hadoop distcp -update hdfs://cluster1/source/first
hdfs://cluster1/source/second  hdfs://cluster2/target
```

上述命令只会将源位置的内容拷贝到目标位置，所以拷贝结果如下：

```
hdfs://cluster2/target/1
hdfs://cluster2/target/2
hdfs://cluster2/target/10
hdfs://cluster2/target/20
```

说明

- 如果多个源位置有相同名称的文件，则 distcp 命令会失败。

- 在不使用 update 和 overwrite 选项的情况下，如果被拷贝文件在目标位置中已经存在，则该文件会跳过。
- 在使用 update 选项的情况下，如果被拷贝文件在目标位置中已经存在，但文件内容不同，则目标位置的文件内容会被更新。
- 在使用 overwrite 选项的情况下，如果被拷贝文件在目标位置中已经存在，目标位置的文件依然会被覆盖。

5. 其它命令选项：

表9-4 其他命令选项

选项	描述
-p[rbugpcaxtq]	当同时使用-update 选项时，即使被拷贝文件的内容没有被更新，它的状态信息也会被更新 r：副本数，b：块大小，u：所属用户，g：所属用户组，p：许可，c：校验和类型，a：访问控制，t：时间戳，q：Quota 信息
-i	拷贝过程中忽略失败
-log <logdir>	指定日志路径
-v	指定日志中的额外信息
-m <num_maps>	最大的同时运行的执行拷贝的任务数
-numListstatusThreads	构建被拷贝文件的文件列表时所用的线程数，该选项会提高 distcp 的运行速度
-overwrite	覆盖目标位置的文件
-update	如果源位置和目标位置的文件的大小，校验和不同，则更新目标位置的文件
-append	当同时使用-update 选项时，追加源位置的文件内容到目标位置的文件
-f <urilist_uri>	将<urilist_uri>文件的内容作为需要拷贝的文件列表
-filters	指定一个本地文件，其文件内容是多条正则表达式。当被拷贝的文件与某条正则表达式匹配时，则该文件不会被拷贝
-async	异步运行 distcp 命令
-atomic {-tmp <tmp_dir>}	指定一次原子性的拷贝，可以添加一个临时目录的选项，作为拷贝过程中的暂存目录
-bandwidth	指定每个拷贝任务的传输带宽，单位 MB/s
-delete	删除掉目标位置中存在，但源位置不存在的文件。该选项通常会和-update 配合使用，表示将源位置和目标位置的文

选项	描述
	件同步，删除掉目标位置多余的文件
-diff <oldSnapshot> <newSnapshot>	将新旧版本之间的差异内容，拷贝到目标位置的旧版本文件中
-skipcrccheck	是否跳过源文件和目标文件之间的 CRC 校验
-strategy {dynamic|uniformsize}	指定拷贝任务的拷贝策略，默认策略是 uniformsize，即每个拷贝任务复制相同的字节数

distcp 常见使用问题

1. 当使用 distcp 命令时，如果某些被拷贝的文件内容较大时，建议修改执行拷贝任务的 mapreduce 的超时时间。可以通过在 distcp 命令中指定 **mapreduce.task.timeout** 选项实现。例如，修改超时时间为 30 分钟，则命令如下：

   ```
   hadoop distcp -Dmapreduce.task.timeout=1800000 hdfs://cluster1/source
   hdfs://cluster2/target
   ```

   您也可以使用选项 filters，不对这种大文件进行拷贝，命令示例如下：

   ```
   hadoop distcp -filters /opt/client/filterfile hdfs://cluster1/source
   hdfs://cluster2/target
   ```

   其中 filterfile 是本地文件，它的内容是多条用于匹配不拷贝文件路径的正则表达式，它的内容示例如下：

   ```
   .*excludeFile1.*
   .*excludeFile2.*
   ```

2. 当使用 distcp 命令时，命令异常退出，报“java.lang.OutOfMemoryError”的错误。

   这个问题的原因是拷贝任务运行时所需的内存超过了客户端设置的内存上限（默认为 128MB）。可以通过修改“*<客户端安装路径>*/HDFS/component_env”中的“CLIENT_GC_OPTS”来修改客户端的内存上限。例如，需要设置该内存上限为 1GB，则设置：

   ```
   CLIENT_GC_OPTS="-Xmx1G"
   ```

   在修改完后，使用如下命令刷新客户端配置，使之生效：

   source *<客户端安装路径>*/**bigdata_env**

3. 使用 dynamic 策略执行 distcp 命令时，命令异常退出，报“Too many chunks created with splitRatio”的错误。

   这个问题的原因是“distcp.dynamic.max.chunks.tolerable”的值（默认值为 20000）小于“distcp.dynamic.split.ratio”的值（默认为 2）乘以 Map 数。即一般出现在 Map 数超过 10000 的情况。可以通过-m 参数降低 Map 数小于 10000：

   ```
   hadoop distcp -strategy dynamic -m 9500 hdfs://cluster1/source
   hdfs://cluster2/target
   ```

   或通过-D 参数指定更大的“distcp.dynamic.max.chunks.tolerable”的值：

   ```
   hadoop distcp -Ddistcp.dynamic.max.chunks.tolerable=30000 -strategy dynamic
   hdfs://cluster1/source hdfs://cluster2/target
   ```

9.6 HDFS 文件系统目录简介

HDFS 文件系统中目录结构如下表所示。

表9-5 HDFS 文件系统目录结构（适用于 MRS 3.x 之前版本）

路径	类型	简略功能	是否可以删除	删除的后果
/tmp/spark/sparkhive-scratch	固定目录	存放 Spark JDBCServer 中 metastore session 临时文件	否	任务运行失败
/tmp/sparkhive-scratch	固定目录	存放 Spark cli 方式运行 metastore session 临时文件	否	任务运行失败
/tmp/carbon/	固定目录	数据导入过程中，如果存在异常 CarbonData 数据，则将异常数据放在此目录下	是	错误数据丢失
/tmp/Loader-${作业名}_${MR 作业 id}	临时目录	存放 Loader Hbase bulkload 作业的 region 信息，作业完成后自动删除	否	Loader Hbase Bulkload 作业失败
/tmp/logs	固定目录	MR 任务日志在 HDFS 上的聚合路径	是	MR 任务日志丢失
/tmp/archived	固定目录	MR 任务日志在 HDFS 上的归档路径	是	MR 任务日志丢失
/tmp/hadoop-yarn/staging	固定目录	保存 AM 运行作业运行日志、作业概要信息和作业配置属性	否	任务运行异常
/tmp/hadoop-yarn/staging/history/done_intermediate	固定目录	所有任务运行完成后，临时存放/tmp/hadoop-yarn/staging 目录下文件	否	MR 任务日志丢失
/tmp/hadoop-yarn/staging/history/d	固定	周期性扫描线程定期将 done_intermediate 的日志文件	否	MR 任务日志丢失

路径	类型	简略功能	是否可以删除	删除的后果
one	目录	转移到 done 目录		
/tmp/mr-history	固定目录	存储预加载历史记录文件的路径	否	MR 历史任务日志数据丢失
/tmp/hive	固定目录	存放 Hive 的临时文件	否	导致 Hive 任务失败
/tmp/hive-scratch	固定目录	Hive 运行时生成的临时数据，如会话信息等	否	当前执行的任务会失败
/user/{user}/.sparkStaging	固定目录	存储 SparkJDBCServer 应用临时文件	否	executor 启动失败
/user/spark/jars	固定目录	存放 Spark executor 运行依赖包	否	executor 启动失败
/user/loader	固定目录	存放 loader 的作业脏数据以及 HBase 作业数据的临时存储目录	否	HBase 作业失败或者脏数据丢失
/user/loader/etl_dirty_data_dir				
/user/loader/etl_hbase_putlist_tmp				
/user/loader/etl_hbase_tmp				
/user/mapred	固定目录	存放 Hadoop 相关的文件	否	导致 Yarn 启动失败
/user/hive	固定目录	Hive 相关数据存储的默认路径，包含依赖的 spark lib 包和用户默认表数据存储位置等	否	用户数据丢失
/user/omm-bulkload	临	HBase 批量导入工具临时目录	否	HBase 批量导

路径	类型	简略功能	是否可以删除	删除的后果
	时目录			入任务失败
/user/hbase	临时目录	HBase 批量导入工具临时目录	否	HBase 批量导入任务失败
/sparkJobHistory	固定目录	Spark eventlog 数据存储目录	否	HistoryServer 服务不可用，任务运行失败
/flume	固定目录	Flume 采集到 HDFS 文件系统中的数据存储目录	否	Flume 工作异常
/mr-history/tmp	固定目录	MapReduce 作业产生的日志存放位置	是	日志信息丢失
/mr-history/done	固定目录	MR JobHistory Server 管理的日志的存放位置	是	日志信息丢失
/tenant	添加租户时创建	配置租户在 HDFS 中的存储目录，系统默认将自动在“/tenant”目录中以租户名称创建文件夹。例如租户“ta1”，默认 HDFS 存储目录为“tenant/ta1”。第一次创建租户时，系统自动在 HDFS 根目录创建“/tenant”目录。支持自定义存储路径。	否	租户不可用
/apps{1~5}/	固定目录	WebHCat 使用到 Hive 的包的路径	否	执行 WebHCat 任务会失败
/hbase	固定目录	HBase 数据存储目录	否	HBase 用户数据丢失

路径	类型	简略功能	是否可以删除	删除的后果
/hbaseFileStream	固定目录	HFS 文件存储目录	否	HFS 文件丢失，且无法恢复
/ats/active	固定目录	HDFS 路径，用于存储活动的应用程序的 timeline 数据	否	删除后会导致 tez 任务运行失败
/ats/done	固定目录	HDFS 路径，用于存储完成的应用程序的 timeline 数据	否	删除后会自动创建
/flink	固定目录	存放 checkpoint 任务数据	否	删除会导致运行任务失败

表9-6 HDFS 文件系统目录结构（适用于 MRS 3.x 及之后版本）

路径	类型	简略功能	是否可以删除	删除的后果
/tmp/spark2x/sparkhive-scratch	固定目录	存放 Spark2x JDBCServer 中 metastore session 临时文件	否	任务运行失败
/tmp/sparkhive-scratch	固定目录	存放 Spark2x cli 方式运行 metastore session 临时文件	否	任务运行失败
/tmp/logs/	固定目录	存放 container 日志文件	是	container 日志不可查看
/tmp/carbon/	固定目录	数据导入过程中，如果存在异常 CarbonData 数据，则将异常数据放在此目录下	是	错误数据丢失

路径	类型	简略功能	是否可以删除	删除的后果
/tmp/Loader-${作业名}_${MR 作业 id}	临时目录	存放 Loader Hbase bulkload 作业的 region 信息，作业完成后自动删除	否	Loader Hbase Bulkload 作业失败
/tmp/hadoop-omm/yarn/system/rmstore	固定目录	ResourceManager 运行状态信息	是	ResourceManager 重启后状态信息丢失
/tmp/archived	固定目录	MR 任务日志在 HDFS 上的归档路径	是	MR 任务日志丢失
/tmp/hadoop-yarn/staging	固定目录	保存 AM 运行作业运行日志、作业概要信息和作业配置属性	否	任务运行异常
/tmp/hadoop-yarn/staging/history/done_intermediate	固定目录	所有任务运行完成后，临时存放/tmp/hadoop-yarn/staging 目录下文件	否	MR 任务日志丢失
/tmp/hadoop-yarn/staging/history/done	固定目录	周期性扫描线程定期将 done_intermediate 的日志文件转移到 done 目录	否	MR 任务日志丢失
/tmp/mr-history	固定目录	存储预加载历史记录文件的路径	否	MR 历史任务日志数据丢失
/tmp/hive-scratch	固定目录	Hive 运行时生成的临时数据，如会话信息等	否	当前执行的任务会失败
/user/{user}/.sparkStaging	固定目录	存储 SparkJDBCServer 应用临时文件	否	executor 启动失败
/user/spark2x/jars	固定目	存放 Spark2x executor 运行依赖包	否	executor 启动失败

路径	类型	简略功能	是否可以删除	删除的后果
	录			
/user/loader	固定目录	存放 loader 的作业脏数据以及 HBase 作业数据的临时存储目录	否	HBase 作业失败或者脏数据丢失
/user/loader/etl_dirty_data_dir				
/user/loader/etl_hbase_putlist_tmp				
/user/loader/etl_hbase_tmp				
/user/oozie	固定目录	存放 oozie 运行时需要的依赖库，需用户手动上传	否	oozie 调度失败
/user/mapred/hadoop-mapreduce-*3.1.1*.tar.gz	固定文件	MR 分布式缓存功能使用的各 jar 包	否	MR 分布式缓存功能无法使用
/user/hive	固定目录	Hive 相关数据存储的默认路径，包含依赖的 spark lib 包和用户默认表数据存储位置等	否	用户数据丢失
/user/omm-bulkload	临时目录	HBase 批量导入工具临时目录	否	HBase 批量导入任务失败
/user/hbase	临时目录	HBase 批量导入工具临时目录	否	HBase 批量导入任务失败
/spark2xJobHistory2x	固定目录	Spark2x eventlog 数据存储目录	否	HistoryServer 服务不可用，任务运行失败
/flume	固定目录	Flume 采集到 HDFS 文件系统中的数据存储目录	否	Flume 工作异常
/mr-history/tmp	固定	MapReduce 作业产生的日志存放位置	是	日志信息丢失

路径	类型	简略功能	是否可以删除	删除的后果
	目录			
/mr-history/done	固定目录	MR JobHistory Server 管理的日志的存放位置	是	日志信息丢失
/tenant	添加租户时创建	配置租户在 HDFS 中的存储目录，系统默认将自动在“/tenant”目录中以租户名称创建文件夹。例如租户“ta1”，默认 HDFS 存储目录为“tenant/ta1”。第一次创建租户时，系统自动在 HDFS 根目录创建“/tenant”目录。支持自定义存储路径。	否	租户不可用
/apps{1~5}/	固定目录	WebHCat 使用到 Hive 的包的路径	否	执行 WebHCat 任务会失败
/hbase	固定目录	HBase 数据存储目录	否	HBase 用户数据丢失
/hbaseFileStream	固定目录	HFS 文件存储目录	否	HFS 文件丢失，且无法恢复

9.7 更改 DataNode 的存储目录

操作场景

说明

本章节适用于 MRS 3.x 及后续版本。

HDFS DataNode 定义的存储目录不正确或 HDFS 的存储规划变化时，系统管理员需要在 FusionInsight Manager 中修改 DataNode 的存储目录，以保证 HDFS 正常工作。适用于以下场景：

- 更改 DataNode 角色的存储目录，所有 DataNode 实例的存储目录将同步修改。
- 更改 DataNode 单个实例的存储目录，只对单个实例生效，其他节点 DataNode 实例存储目录不变。

对系统的影响

- 更改 DataNode 角色的存储目录需要停止并重新启动 HDFS 服务，集群未完全启动前无法提供服务。
- 更改 DataNode 单个实例的存储目录需要停止并重新启动实例，该节点 DataNode 实例未启动前无法提供服务。
- 服务参数配置如果使用旧的存储目录，需要更新为新目录。

前提条件

- 在各个数据节点准备并安装好新磁盘，并格式化磁盘。
- 规划好新的目录路径，用于保存旧目录中的数据。
- 已安装好 HDFS 客户端。
- 准备好系统管理员用户 **hdfs**。
- 更改 DataNode 单个实例的存储目录时，保持活动的 DataNode 实例数必须大于“dfs.replication”的值。

操作步骤

检查环境

步骤 1 以 **root** 用户登录安装 HDFS 客户端的服务器，执行以下命令配置环境变量。

source *HDFS 客户端安装目录*/**bigdata_env**

步骤 2 如果集群为安全模式，执行以下命令认证用户身份。

kinit hdfs

步骤 3 在 HDFS 客户端执行以下命令，检查 HDFS 根目录下全部目录和文件是否状态正常。

hdfs fsck /

检查 fsck 显示结果：

- 显示如下信息，表示无文件丢失或损坏，执行步骤 4。

```
The filesystem under path '/' is HEALTHY
```

- 显示其他信息，表示有文件丢失或损坏，执行步骤 5。

步骤 4 登录 FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务”查看 HDFS 的状态“运行状态”是否为“良好”。

- 是，执行步骤 6。
- 否，HDFS 状态不健康，执行步骤 5。

步骤 5 修复 HDFS 异常的具体操作，任务结束。

步骤 6 确定修改 DataNode 的存储目录场景。

- 更改 DataNode 角色的存储目录，执行步骤 7。
- 更改 DataNode 单个实例的存储目录，执行步骤 12。

更改 DataNode 角色的存储目录

步骤 7 选择“集群 > *待操作集群的名称* > 服务 > HDFS > 停止服务”，停止 HDFS 服务。

步骤 8 以 **root** 用户登录到安装 HDFS 服务的各个数据节点中，执行如下操作：

1. 创建目标目录（data1,data2 为集群原有目录）。

   例如目标目录为“${BIGDATA_DATA_HOME}/hadoop/data3/dn”：

   执行 **mkdir -p ${BIGDATA_DATA_HOME}/hadoop/data3/dn**。
2. 挂载目标目录到新磁盘。例如挂载“${BIGDATA_DATA_HOME}/hadoop/data3”到新磁盘。
3. 修改新目录的权限。

   例如新目录路径为“${BIGDATA_DATA_HOME}/hadoop/data3/dn”：

   执行 **chmod 700 ${BIGDATA_DATA_HOME}/hadoop/data3/dn -R** 和 **chown omm:wheel ${BIGDATA_DATA_HOME}/hadoop/data3/dn -R**。
4. 将数据复制到目标目录。

   例如旧目录为“${BIGDATA_DATA_HOME}/hadoop/data1/dn”，目标目录为“${BIGDATA_DATA_HOME}/hadoop/data3/dn”：

   执行 **cp -af ${BIGDATA_DATA_HOME}/hadoop/data1/dn/* ${BIGDATA_DATA_HOME}/hadoop/data3/dn**。

步骤 9 在 FusionInsight Manager 管理界面，选择“集群 > *待操作集群的名称* > 服务 > HDFS > 配置 > 全部配置”，打开 HDFS 服务配置页面。

将配置项“dfs.datanode.data.dir”从默认值“%{@auto.detect.datapart.dn}”修改为新的目标目录，例如“${BIGDATA_DATA_HOME}/hadoop/data3/dn”。

例如：原有的数据存储目录为“/srv/BigData/hadoop/data1”，“/srv/BigData/hadoop/data2”，如需将 data1 目录的数据迁移至新建的“/srv/BigData/hadoop/data3”目录，则将服务级别的此参数替换为现有的数据存储目录，如果有多个存储目录，用“,”隔开。则本示例中，为“/srv/BigData/hadoop/data2,/srv/BigData/hadoop/data3”。

步骤 10 单击“保存”。然后在“集群 > *待操作集群的名称* > 服务”界面启动集群中各个停止的服务。

步骤 11 启动 HDFS 成功以后，在 HDFS 客户端执行以下命令，检查 HDFS 根目录下全部目录和文件是否复制正确。

hdfs fsck /

检查 fsck 显示结果：

- 显示如下信息，表示无文件丢失或损坏，数据复制成功，操作结束。

  ```
  The filesystem under path '/' is HEALTHY
  ```
- 显示其他信息，表示有文件丢失或损坏，则检查 8.4 是否正确，并执行 **hdfs fsck** *损坏的文件名称* **-delete**。

更改 DataNode 单个实例的存储目录

步骤 12 选择“集群 > *待操作集群的名称* > 服务 > HDFS > 实例”，勾选需要修改存储目录的 DataNode 单个实例，选择“更多 > 停止实例”。

步骤 13 以 **root** 用户登录到这个 DataNode 节点，执行如下操作。

1. 创建目标目录。

   例如目标目录为“${BIGDATA_DATA_HOME}/hadoop/data3/dn”：

   执行 **mkdir -p ${BIGDATA_DATA_HOME}/hadoop/data3/dn**。

2. 挂载目标目录到新磁盘。

   例如挂载“${BIGDATA_DATA_HOME}/hadoop/data3”到新磁盘。

3. 修改新目录的权限。

   例如新目录路径为“${BIGDATA_DATA_HOME}/hadoop/data3/dn”：

   执行 **chmod 700 ${BIGDATA_DATA_HOME}/hadoop/data3/dn -R 和 chown omm:wheel ${BIGDATA_DATA_HOME}/hadoop/data3/dn -R**。

4. 将数据复制到目标目录。

   例如旧目录为“${BIGDATA_DATA_HOME}/hadoop/data1/dn”，目标目录为“${BIGDATA_DATA_HOME}/hadoop/data3/dn”：

   执行 **cp -af ${BIGDATA_DATA_HOME}/hadoop/data1/dn/* ${BIGDATA_DATA_HOME}/hadoop/data3/dn**。

步骤 14 在 FusionInsight Manager 管理界面，选择“集群 > *待操作集群的名称* > 服务 > HDFS > 实例”，单击指定的 DataNode 实例并切换到“实例配置”页签。

将配置项“dfs.datanode.data.dir”从默认值“%{@auto.detect.datapart.dn}”修改为新的目标目录，例如“${BIGDATA_DATA_HOME}/hadoop/data3/dn”。

示例：原有的数据存储目录为“/srv/BigData/hadoop/data1,/srv/BigData/hadoop/data2”，此处如需将 data1 目录的数据迁移至新建的/srv/BigData/hadoop/data3 目录，则将该参数修改为“/srv/BigData/hadoop/data2,/srv/BigData/hadoop/data3”。

步骤 15 单击“保存”，单击“确定”。

界面提示“操作成功。”，单击“完成”。

步骤 16 选择“更多 > 重启实例”，重启 DataNode 实例。

----结束

9.8 配置 HDFS 目录权限

操作场景

默认情况下，某些 HDFS 的文件目录权限为 777 或者 750，存在安全风险。建议您在安装完成后修改该 HDFS 目录的权限，增加用户的安全性。

操作步骤

在 HDFS 客户端中，使用具有 HDFS 管理员权限的用户，执行如下命令，将“/user”的目录权限进行修改。

此处将权限修改为“1777”，即在权限处增加“1”，表示增加目录的粘性，即只有创建的用户才可以删除此目录。

hdfs dfs -chmod 1777 */user*

为了系统文件的安全，建议用户将非临时目录进行安全加固，例如：

- /user:777
- /mr-history:777
- /mr-history/tmp:777
- /mr-history/done:777
- /user/mapred:755

9.9 配置 NFS

操作场景

说明

本章节适用于 MRS 3.x 及后续版本。

用户在部署集群前，可根据需要规划 Network File System（简称 NFS）服务器，用于存储 NameNode 元数据，以提高数据可靠性。

如果您已经部署 NFS 服务器，并已配置 NFS 服务，本操作提供集群侧的配置指导，为可选任务。

操作步骤

步骤 1 在 NFS 服务器上检查 NFS 的共享目录权限，确认服务器可以访问 MRS 集群的 NameNode。

步骤 2 以 **root** 用户登录 NameNode 主节点。

步骤 3 执行如下命令，创建目录并赋予目录写权限。

mkdir ${BIGDATA_DATA_HOME}/namenode-nfs

chown omm:wheel ${BIGDATA_DATA_HOME}/namenode-nfs

chmod 750 ${BIGDATA_DATA_HOME}/namenode-nfs

步骤 4 执行如下命令，挂载 NFS 到 NameNode 主节点。

mount -t nfs -o rsize=8192,wsize=8192,soft,nolock,timeo=3,intr *NFS 服务器 IP 地址:共享目录* **${BIGDATA_DATA_HOME}/namenode-nfs**

例如，NFS 服务器的 IP 为“192.168.0.11”，共享目录为“/opt/Hadoop/NameNode”，则执行命令：

mount -t nfs -o rsize=8192,wsize=8192,soft,nolock,timeo=3,intr 192.168.0.11:/opt/Hadoop/NameNode ${BIGDATA_DATA_HOME}/namenode-nfs

步骤 5 在 NameNode 备节点上执行步骤 2～步骤 4。

说明

主备 NameNode 节点在 NFS 服务器上创建的共享目录名称（如“/opt/Hadoop/NameNode”）不能相同。

步骤 6 登录 FusionInsight Manager 系统，选择“集群 > *待操作集群的名称* > 服务 > HDFS > 配置 > 全部配置”。

步骤 7 在界面右侧的“搜索”框中输入“dfs.namenode.name.dir”搜索，在其值中增加“${BIGDATA_DATA_HOME}/namenode-nfs”路径，多个路径间使用“,”隔开，然后单击“保存”。

步骤 8 单击“确定”。在概览页面选择“更多 > 重启服务”，重启服务。

----结束

9.10 规划 HDFS 容量

HDFS DataNode 以 Block 的形式，保存用户的文件和目录，同时在 NameNode 中生成一个文件对象，对应 DataNode 中每个文件、目录和 Block。

NameNode 中文件对象需要占用一定的内存，消耗内存大小随文件对象的生成而线性递增。DataNode 实际保存的文件和目录越多，NameNode 文件对象总量增加，需要消耗更多的内存，使集群现有硬件可能会难以满足业务需求，且导致集群难以扩展。

规划存储大量文件的 HDFS 系统容量，就是规划 NameNode 的容量规格和 DataNode 的容量规格，并根据容量设置参数。

容量规格

- NameNode 容量规格

  在 NameNode 中，每个文件对象对应 DataNode 中的一个文件、目录或 Block。

  一个文件至少占用一个 Block，默认每个 Block 大小为“134217728”即 128MB，对应参数为“dfs.blocksize”。默认情况下一个文件小于 128MB 时，只占用一个 Block；文件大于 128MB 时，占用 Block 数为：文件大小/128MB。目录不占用 Block。

  根据“dfs.blocksize”，NameNode 的文件对象数计算方法如下：

表9-7 NameNode 文件对象数计算

单个文件大小	文件对象数
小于 128MB	1（对应文件）+1（对应 Block）=2

单个文件大小	文件对象数
大于 128MB（例如 128G）	1（对应文件）+1,024（对应 128GB/128MB=1024 Block）=1,025

主备 NameNode 支持最大文件对象的数量为 300,000,000（最多对应 150,000,000 个小文件）。“dfs.namenode.max.objects”规定当前系统可生成的文件对象数，默认值为“0”表示不限制。

- DataNode 容量规格

  在 HDFS 中，Block 以副本的形式存储在 DataNode 中，默认副本数为“3”，对应参数为“dfs.replication”。

  集群中所有 DataNode 角色实例保存的 Block 总数为：HDFS Block * 3。集群中每个 DataNode 实例平均保存的 Blocks= HDFS Block * 3/DataNode 节点数。

表9-8 DataNode 支持规格

项目	规格
单个 DataNode 实例支持最大 Block 数	5,000,000
单个 DataNode 实例上单个磁盘支持最大 Block 数	500,000
单个 DataNode 实例支持最大 Block 数需要的最小磁盘数	10

表9-9 DataNode 节点数规划

HDFS Block 数	最少 DataNode 角色实例数
10,000,000	10,000,000 *3/5,000,000 = 6
50,000,000	50,000,000 *3/5,000,000 = 30
100,000,000	100,000,000 *3/5,000,000 = 60

内存参数设置

- NameNode JVM 参数配置规则

  NameNode JVM 参数“GC_OPTS”默认值为：

  -Xms2G -Xmx4G -XX:NewSize=128M -XX:MaxNewSize=256M -XX:MetaspaceSize=128M -XX:MaxMetaspaceSize=128M -XX:+UseConcMarkSweepGC -XX:+CMSParallelRemarkEnabled -XX:CMSInitiatingOccupancyFraction=65 -XX:+PrintGCDetails -Dsun.rmi.dgc.client.gcInterval=0x7FFFFFFFFFFFFE -Dsun.rmi.dgc.server.gcInterval=0x7FFFFFFFFFFFFE -XX:-

OmitStackTraceInFastThrow -XX:+PrintGCDateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=1M -Djdk.tls.ephemeralDHKeySize=3072 -Djdk.tls.rejectClientInitiatedRenegotiation=true -Djava.io.tmpdir=${Bigdata_tmp_dir}

NameNode 文件数量和 NameNode 使用的内存大小成比例关系，文件对象变化时请修改默认值中的“-Xms2G -Xmx4G -XX:NewSize=128M -XX:MaxNewSize=256M”。参考值如下表所示。

表9-10 NameNode JVM 配置

文件对象数量	参考值
10,000,000	“-Xms6G -Xmx6G -XX:NewSize=512M -XX:MaxNewSize=512M”
20,000,000	“-Xms12G -Xmx12G -XX:NewSize=1G -XX:MaxNewSize=1G”
50,000,000	“-Xms32G -Xmx32G -XX:NewSize=3G -XX:MaxNewSize=3G”
100,000,000	“-Xms64G -Xmx64G -XX:NewSize=6G -XX:MaxNewSize=6G”
200,000,000	“-Xms96G -Xmx96G -XX:NewSize=9G -XX:MaxNewSize=9G”
300,000,000	“-Xms164G -Xmx164G -XX:NewSize=12G -XX:MaxNewSize=12G”

- DataNode JVM 参数配置规则

  DataNode JVM 参数“GC_OPTS”默认值为：

  -Xms2G -Xmx4G -XX:NewSize=128M -XX:MaxNewSize=256M -XX:MetaspaceSize=128M -XX:MaxMetaspaceSize=128M -XX:+UseConcMarkSweepGC -XX:+CMSParallelRemarkEnabled -XX:CMSInitiatingOccupancyFraction=65 -XX:+PrintGCDetails -Dsun.rmi.dgc.client.gcInterval=0x7FFFFFFFFFFFFE -Dsun.rmi.dgc.server.gcInterval=0x7FFFFFFFFFFFFE -XX:-OmitStackTraceInFastThrow -XX:+PrintGCDateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=1M -Djdk.tls.ephemeralDHKeySize=3072 -Djdk.tls.rejectClientInitiatedRenegotiation=true -Djava.io.tmpdir=${Bigdata_tmp_dir}

  集群中每个 DataNode 实例平均保存的 Blocks= HDFS Block * 3/DataNode 节点数，单个 DataNode 实例平均 Block 数量变化时请修改默认值中的“-Xms2G -Xmx4G -XX:NewSize=128M -XX:MaxNewSize=256M”。参考值如下表所示。

表9-11 DataNode JVM 配置

单个 DataNode 实例平均 Block 数量	参考值

单个 DataNode 实例平均 Block 数量	参考值
2,000,000	“-Xms6G -Xmx6G -XX:NewSize=512M -XX:MaxNewSize=512M”
5,000,000	“-Xms12G -Xmx12G -XX:NewSize=1G -XX:MaxNewSize=1G”

Xmx 内存值对应 DataNode 节点块数阈值，每 GB 对应 500000 块数，用户可根据需要调整内存值。

查看 HDFS 容量状态

- NameNode 信息

  MRS 3.x 之前版本：登录 MRS 控制台，选择“组件管理 > HDFS > NameNode(主)”，单击“Overview”，查看“Summary”显示的当前 HDFS 中文件对象、文件数量、目录数量和 Block 数量信息。

  MRS 3.x 及后续版本：登录 FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > HDFS > NameNode(主)”，单击“Overview”，查看“Summary”显示的当前 HDFS 中文件对象、文件数量、目录数量和 Block 数量信息。

- DataNode 信息

  MRS 3.x 之前版本：登录 MRS 控制台，选择“组件管理 > HDFS > NameNode(主)”，单击“Datanodes”，查看所有告警 DataNode 节点的 Block 数量信息。

  MRS 3.x 及后续版本：登录 FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > HDFS > NameNode(主)”，单击“DataNodes”，查看所有告警 DataNode 节点的 Block 数量信息。

- 告警信息

  监控 ID 为 14007、14008、14009 的告警是否产生，根据业务需要修改告警阈值。

9.11 设置 HBase 和 HDFS 的 ulimit

现象描述

当打开一个 HDFS 文件时，句柄数限制导出，出现如下错误：

```
IOException (Too many open files)
```

处理步骤

您可以联系管理员增加各用户的句柄数。该配置为操作系统的配置，并非 HBase 或者 HDFS 的配置。建议管理员根据 HBase 和 HDFS 的业务量及各操作系统用户的权限进行句柄数设置。如果某一个用户需对业务量很大的 HDFS 进行很频繁且很多的操作，则为此用户设置较大的句柄数，避免出现以上错误。

步骤 1 使用 **root** 用户登录集群所有节点机器或者客户端机器的操作系统，并进入“/etc/security”目录。

步骤 2 执行如下命令编辑“limits.conf”文件。

vi limits.conf

新增如下内容：

```
hdfs  -      nofile  32768
hbase -      nofile  32768
```

其中“hdfs”和“hbase”表示业务中用到的操作系统用户名称。

说明

- 只有 **root** 用户有权限编辑“limits.conf”文件。
- 如果修改的配置不生效，请确认“/etc/security/limits.d”目录下是否有针对操作系统用户的其他 nofile 值。这样的值可能会覆盖“/etc/security/limits.conf”中配置的值。
- 如果用户需要对 HBase 进行操作，建议将该用户的句柄数设置为“10000”以上。如果用户需要对 HDFS 进行操作，建议根据业务量大小设置对应的句柄数，建议不要给太小的值。如果用户需要对 HBase 和 HDFS 操作，建议设置较大的值，例如“32768”。

步骤 3 您可以使用如下命令查看某一用户的句柄数限制。

su - *user_name*

ulimit -n

界面会返回此用户的句柄数限制值。如下所示：

```
8194
```

----结束

9.12 配置 DataNode 容量均衡

操作场景

说明

本章节适用于 MRS 3.x 及后续版本。

HDFS 集群可能出现 DataNode 节点间磁盘利用率不平衡的情况，比如集群中添加新数据节点的场景。如果 HDFS 出现数据不平衡的状况，可能导致多种问题，比如 MapReduce 应用程序无法很好地利用本地计算的优势、数据节点之间无法达到更好的网络带宽使用率或节点磁盘无法利用等等。所以系统管理员需要定期检查并保持 DataNode 数据平衡。

HDFS 提供了一个容量均衡程序 Balancer。通过运行这个程序，可以使得 HDFS 集群达到一个平衡的状态，使各 DataNode 磁盘使用率与 HDFS 集群磁盘使用率的偏差不超过阈值。图 9-1 和图 9-2 分别是 Balance 前后 DataNode 的磁盘使用率变化。

图9-1 执行均衡操作前 DataNode 的磁盘使用率

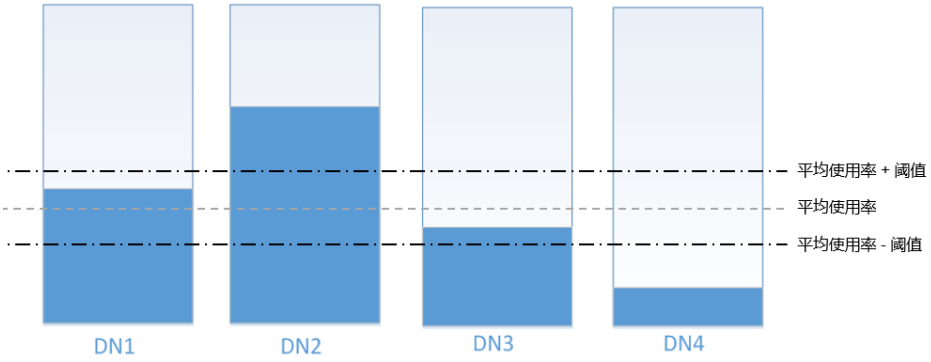
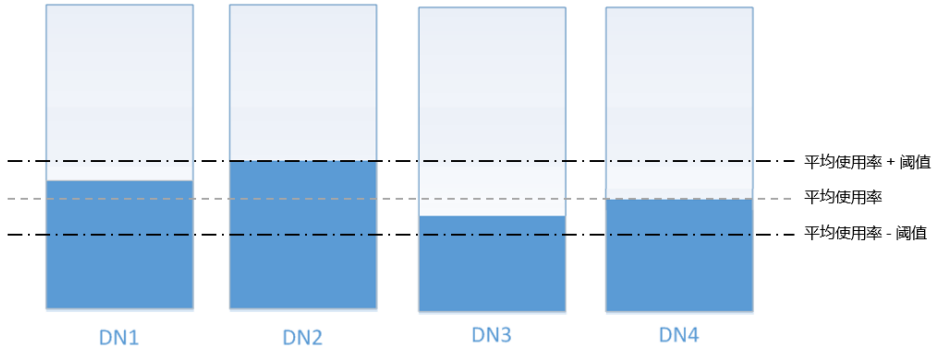


图9-2 执行均衡操作后 DataNode 的磁盘使用率



均衡操作时间估算受两个因素影响：

1. 需要迁移的总数据量：

   每个 DataNode 节点的数据量应大于（平均使用率-阈值）*平均数据量，小于（平均使用率+阈值）*平均数据量。若实际数据量小于最小值或大于最大值即存在不平衡，系统选择所有 DataNode 节点中偏差最多的数据量作为迁移的总数据量。

2. Balancer 的迁移是按迭代（iteration）方式串行顺序处理的，每个 iteration 迁移数据量不超过 10GB，每个 iteration 重新计算使用率的情况。

因此针对集群情况，可以大概估算每个 iteration 耗费的时间（可以通过执行 Balancer 的日志观察到每次 iteration 的时间），并用总数据量除以 10GB 估算任务执行时间。

由于按 iteration 处理，Balancer 可以随时启动或者停止。

对系统的影响

- 执行 Balance 操作时会占用 DataNode 的网络带宽资源，请根据业务需求在维护期间执行任务。
- 默认使用带宽控制为 20MB/s，如果重新设置带宽流量或加大数据量，Balance 操作可能会对正在运行的业务产生影响。

前提条件

已安装 HDFS 客户端。

操作步骤

步骤 1 使用客户端安装用户登录客户端所在节点。执行命令切换到客户端安装目录，例如“/opt/client”。

cd /opt/client

说明

如果集群为普通模式，需先执行 **su - omm** 切换为 **omm** 用户。

步骤 2 执行以下命令配置环境变量。

source bigdata_env

步骤 3 如果集群为安全模式，执行以下命令认证 **hdfs** 身份。

kinit hdfs

步骤 4 是否调整带宽控制？

- 是，执行步骤 5。
- 否，执行步骤 6。

步骤 5 执行以下命令，修改 Balance 的最大带宽，然后执行步骤 6。

hdfs dfsadmin -setBalancerBandwidth <*bandwidth in bytes per second*>

<*bandwidth in bytes per second*>表示带宽控制的数值，单位为字节。例如要设置带宽控制为 20MB/s，对应值为 20971520，完整命令为：

hdfs dfsadmin -setBalancerBandwidth 20971520

说明

- 默认为 20MB/s，适用于当前集群使用万兆网络，且有业务正在执行的场景。若没有足够的业务空闲时间窗用于 Balance 维护，可适当增加该值以缩短 Balance 时间，如增大到 209715200（即 200MB/s）。
- 这个参数的调整要看组网情况，如果集群负载较高，可以改为 209715200(200MB/s)；如果集群空闲，可以改为 1073741824 (1GB/s)。
- 如果 DataNode 节点的带宽无法达到指定的最大带宽，可以在 FusionInsight Manager 修改 HDFS 的参数“dfs.datanode.balance.max.concurrent.moves”，将每个 DataNode 节点执行均衡的线程数修改为“32”，并重启 HDFS 服务。

步骤 6 执行以下命令，启动 Balance 任务。

bash /opt/client/HDFS/hadoop/sbin/start-balancer.sh -threshold <*threshold of* balancer>

-threshold 表示 HDFS 数据达到平衡状态时 DataNode 磁盘使用率偏差值，各个 DataNode 节点磁盘的使用率和整体 HDFS 集群的磁盘空间平均使用率偏差小于此阈值时，系统认为 HDFS 集群已经达到了平衡的状态并结束 Balance 任务。

例如，需要设置偏差率为 5%，则执行：

bash /opt/client/HDFS/hadoop/sbin/start-balancer.sh -threshold 5

说明

- 上述命令会在后台执行该任务，相关日志可以通过客户端安装目录“/opt/client/HDFS/hadoop/logs”下的 hadoop-root-balancer-*主机名*.out 查看。
- 如果需要停止 Balance 任务，请执行以下命令：

  bash /opt/client/HDFS/hadoop/sbin/stop-balancer.sh
- 如果只需要对部分节点进行数据均衡，可以在脚本上加上-include 参数指定要移动的节点。具体参数使用方法，可通过命令行查看。
- “/opt/client”为客户端安装目录，如果不一致，替换即可。
- 如果该命令执行失败，在日志中看到的错误信息为“Failed to APPEND_FILE /system/balancer.id”，则需要执行如下命令强行删除“/system/balancer.id”，再次执行 **start-balancer.sh** 脚本即可。

  hdfs dfs -rm -f /system/balancer.id

步骤 7 用户在执行了步骤 6 的脚本后，会在客户端安装目录“/opt/client/HDFS/hadoop/logs”目录下生成名为 hadoop-root-balancer-*主机名*.out 日志。打开该日志可以看到如下字段信息：

- Time Stamp：时间戳
- Bytes Already Moved：已经移动的字节数
- Bytes Left To Move：待移动的字节数
- Bytes Being Moved：正在移动的字节数

日志出现“Balancing took *xxx* seconds”信息表示均衡操作已完成。

----结束

相关任务

设置自动执行 Balance 任务

步骤 1 登录 FusionInsight Manager。

步骤 2 选择“集群 > *待操作集群的名称* > 服务 > HDFS > 配置”，选择“全部配置”，搜索以下参数名并修改参数值。

- “dfs.balancer.auto.enable”表示是否启用自动执行 Balance 任务，默认值为“false”表示不启用，修改为“true”表示启用。
- “dfs.balancer.auto.cron.expression”表示任务执行的时间，默认值“0 1 * * 6”表示在每周六的 1 点执行任务。仅在启用自动执行 Balance 功能时有效。

  修改此参数时，表达式介绍如表 9-12 所示。支持“*”表示连续的时间段。

表9-12 执行表达式参数解释

列	说明

列	说明
第 1 列	分钟，参数值为 0～59。
第 2 列	小时，参数值为 0～23。
第 3 列	日期，参数值为 1～31。
第 4 列	月份，参数值为 1～12。
第 5 列	星期，参数值为 0～6，0 表示星期日。

- “dfs.balancer.auto.stop.cron.expression”表示任务自动停止的时间，默认值为空，表示不自动停止正在运行的 Balancer 任务。以“0 5 * * 6”为例，则表示在每周六的 5 点停止正在运行的 Balancer 任务。仅在启用自动执行 Balance 功能时有效。

  修改此参数时，表达式介绍如表 9-12 所示。支持“*”表示连续的时间段。

步骤 3 修改自动 Balancer 的运行参数，如表 9-13 所示：

表9-13 自动 Balancer 运行参数

参数名	参数介绍	默认值
dfs.balancer.auto.threshold	表示磁盘容量百分比的均衡阈值。仅当 dfs.balancer.auto.enable 设置为 true 时才有效。	10
dfs.balancer.auto.exclude.datanodes	不需要执行磁盘自动均衡的 DataNode 列表，用逗号分隔。仅当 dfs.balancer.auto.enable 设置为 true 时才有效。	默认为空
dfs.balancer.auto.bandwidthPerSec	每个 DataNode 可用于负载均衡的最大带宽量（单位：MB/s）。	20
dfs.balancer.auto.maxIdleIterations	Balancer 的最大连续空闲迭代次数。一次空闲迭代为没有 Block 块被移动的迭代，当连续空闲迭代次数达到最大连续空闲迭代次数时，本次 Balancer 结束。当取值为-1 时，代表无穷大。	5
dfs.balancer.auto.maxDataNodesNum	该参数用来控制进行自动 Balancer 的 DataNode 数量。假设该参数值为 N，当 N 大于 0， 则选择剩余空间比例最高的 N 个 DataNode 和最低的 N 个 DataNode 之间进行数据均衡；当 N 等于 0， 则对集群中所有 DataNode 进行数据均衡。	5

步骤 4 单击“保存”使配置生效。无需重启 HDFS 服务。

任务执行日志保存在主 NameNode 节点中，请查看“/var/log/Bigdata/hdfs/nn/hadoop-omm-balancer-*主机名*.log”。

----结束

9.13 配置 DataNode 节点间容量异构时的副本放置策略

操作场景

默认情况下，NameNode 会随机选择 DataNode 节点写文件。当集群内某些数据节点的磁盘容量不一致（某些节点的磁盘总容量大，某些总容量小），会导致磁盘总容量小的节点先写满。通过修改集群默认的 DataNode 写数据时的磁盘选择策略为“节点磁盘可用空间块放置策略”，可提高将块数据写到磁盘可用空间较大节点的概率，解决因为数据节点磁盘容量不一致导致的节点使用率不均衡的情况。

对系统的影响

修改磁盘选择策略为“节点磁盘可用空间块放置策略（org.apache.hadoop.hdfs.server.blockmanagement.AvailableSpaceBlockPlacementPolicy）”，经过测试验证，在该测试结果中，修改前后，HDFS 写文件性能影响范围在 3%以内。

说明

NameNode 默认的副本存储策略为：

1. 第一副本：存放到客户端所在节点。
2. 第二副本：远端机架的数据节点。
3. 第三副本：存放到客户端所在节点的相同机架的不同节点。

如还有更多副本，则随机选择其它 DataNode。

“节点磁盘可用空间块放置策略”的副本选择机制为：

1. 第一个副本：存放在客户端所在 DataNode（和默认的存放策略一样）。
2. 第二个副本：

- 选择存储节点的时候，先挑选 2 个满足要求的数据节点。
- 比较这 2 个节点磁盘空间使用比例，如果磁盘空间使用率的相差小于 5%，随机存放到第一个节点。
- 如果磁盘空间使用率相差超过 5%，即有 60%（由 dfs.namenode.available-space-block-placement-policy.balanced-space-preference-fraction 指定，默认值 0.6）的概率写到磁盘空间使用率低的节点。

3. 第三副本等其他后续副本的存储情况，也参考第二个副本的选择方式。

前提条件

集群里 DataNode 节点的磁盘总容量偏差不能超过 100%。

操作步骤

步骤 1 请参考修改集群服务配置参数，进入 HDFS 的“全部配置”页面。

步骤 2 调整 HDFS 写数据时的依据的磁盘选择策略参数。搜索“dfs.block.replicator.classname”参数，并将参数的值改为“org.apache.hadoop.hdfs.server.blockmanagement.AvailableSpaceBlockPlacementPolicy”。

步骤 3 保存修改的配置。保存完成后请重新启动配置过期的服务或实例以使配置生效。

----结束

9.14 配置 HDFS 单目录文件数量

操作场景

通常一个集群上部署了多个服务，且大部分服务的存储都依赖于 HDFS 文件系统。当集群运行时，不同组件（例如 Spark、Yarn）或客户端可能会向同一个 HDFS 目录不断写入文件。但 HDFS 系统支持的单目录文件数目是有上限的，因此用户需要提前做好规划，防止单个目录下的文件数目超过阈值，导致任务出错。

HDFS 提供了“dfs.namenode.fs-limits.max-directory-items”参数设置单个目录下可以存储的文件数目。

操作步骤

步骤 1 请参考修改集群服务配置参数，进入 HDFS 的“全部配置”页面。

步骤 2 搜索配置项“dfs.namenode.fs-limits.max-directory-items”。

表9-14 参数说明

参数名称	描述	默认值
dfs.namenode.fs-limits.max-directory-items	定义目录中包含的最大条目数。 取值范围：1～6400000	1048576

步骤 3 设置单个 HDFS 目录下最大可容纳的文件数目。保存修改的配置。保存完成后请重新启动配置过期的服务或实例以使配置生效。

说明

用户尽量将数据做好存储规划，可以按时间、业务类型等分类，不要单个目录下直属的文件过多，建议使用默认值，单个目录下约 100 万条。

----结束

9.15 配置回收站机制

配置场景

在 HDFS 中，删除的文件将被移动到回收站（trash）中，以便在误操作的情况下恢复被删除的数据。

您可以设置文件保留在回收站中的时间阈值，一旦文件保存时间超过此阈值，将从回收站中永久地删除。如果回收站被清空，回收站中的所有文件将被永久删除。

配置描述

在 HDFS 中，如果删除 HDFS 的文件，文件会被保存到 trash 空间中，不会被立即清除。被删除的文件在超过老化时间后将变为老化文件，会基于系统机制清除或用户手动清除。

参数入口：

请参考修改集群服务配置参数，进入 HDFS 的“全部配置”页面，在搜索框中输入参数名称。

表9-15 参数说明

参数	描述	默认值
fs.trash.interval	以分钟为单位的垃圾回收时间，垃圾站中数据超过此时间，会被删除。取值范围：1440～259200。	1440
fs.trash.checkpoint.interval	垃圾检查点间的间隔。单位：分钟。应小于等于 fs.trash.interval 的值。检查点程序每次运行时都会创建一个新的检查点并会移除 fs.trash.interval 分钟前创建的检查点。例如，系统每 10 分钟检测是否存在老化文件，如果发现有老化文件，则删除。对于未老化文件，则会存储在 checkpoint 列表中，等待下一次检查。 如果此参数的值设置为 0，则表示系统不会检查老化文件，所有老化文件会被保存在系统中。 取值范围：0～*fs.trash.interval*。 说明 不推荐将此参数值设置为 0，这样系统的老化文件会一	60

参数	描述	默认值
	直存储下去，导致集群的磁盘空间不足。	

9.16 配置文件和目录的权限

配置场景

HDFS 支持用户进行文件和目录默认权限的修改。HDFS 默认用户创建文件和目录的权限的掩码为“022”，如果默认权限满足不了用户的需求，可以通过配置项进行默认权限的修改。

配置描述

参数入口：

请参考修改集群服务配置参数，进入 HDFS 的“全部配置”页面，在搜索框中输入参数名称。

表9-16 参数说明

参数	描述	默认值
fs.permissions.umask-mode	当客户端在 HDFS 上创建文件和目录时使用此 umask 值（用户掩码）。类似于 linux 上的文件权限掩码。 可以使用八进制数字也可以使用符号，例如：“022” (八进制，等同于以符号表示的 u=rwx,g=r-x,o=r-x)，或者“u=rwx,g=rwx,o=” (符号法，等同于八进制的“007”)。 说明 8 进制的掩码，和实际权限设置值正好相反，建议使用符号表示法，描述更清晰。	022

9.17 配置 token 的最大存活时间和时间间隔

配置场景

安全模式下，HDFS 中用户可以对 token 的最大存活时间和 token renew 的时间间隔进行灵活地设置，根据集群的具体需求合理地配置。

配置描述

参数入口：

请参考修改集群服务配置参数，进入 HDFS 的“全部配置”页面，在搜索框中输入参数名称。

表9-17 参数说明

参数	描述	默认值
dfs.namenode.delegation.token.max-lifetime	该参数为服务器端参数，设置 token 的最大存活时间，单位为毫秒。取值范围：10000~10000000000000。	604800000
dfs.namenode.delegation.token.renew-interval	该参数为服务器端参数，设置 token renew 的时间间隔，单位为毫秒。取值范围：10000~10000000000000。	86400000

9.18 配置磁盘坏卷

配置场景

在开源版本中，如果为 DataNode 配置多个数据存放卷，默认情况下其中一个卷损坏，则 DataNode 将不再提供服务。用户可以通过修改配置项“dfs.datanode.failed.volumes.tolerated”的值，指定失败的个数，小于该个数，DataNode 可以继续提供服务。

配置描述

参数入口：

请参考修改集群服务配置参数，进入 HDFS 的“全部配置”页面，在搜索框中输入参数名称。

表9-18 参数说明

参数	描述	默认值
dfs.datanode.failed.volumes.tolerated	DataNode 停止提供服务前允许失败的卷数。默认情况下，必须至少有一个有效卷。值-1 表示有效卷的最小值是 1。大于等于 0 的值表示允许失败的卷数。	MRS 3.x 之前版本：0 MRS 3.x 及之后版本：-1

9.19 使用安全加密通道

配置场景

安全加密通道是 HDFS 中 RPC 通信的一种加密协议，当用户调用 RPC 时，用户的 login name 会通过 RPC 头部传递给 RPC，之后 RPC 使用 Simple Authentication and Security Layer（SASL）确定一个权限协议（支持 Kerberos 和 DIGEST-MD5 两种），完成 RPC 授权。用户在部署安全集群时，需要使用安全加密通道，配置如下参数。安全 Hadoop RPC 相关信息请参考：https://hadoop.apache.org/docs/r3.1.1/hadoop-project-dist/hadoop-common/SecureMode.html#Data_Encryption_on_RPC

配置描述

请参考修改集群服务配置参数，进入 HDFS 的“全部配置”页面，在搜索框中输入参数名称。

表9-19 参数说明

参数	描述	默认值
hadoop.rpc.protection	须知 • 设置后需要重启服务生效，且不支持滚动重启。 • 设置后需要重新下载客户端配置，否则 HDFS 无法提供读写服务。 设置 Hadoop 中各模块的 RPC 通道是否加密。通道包括： • 客户端访问 HDFS 的 RPC 通道。 • HDFS 中各模块间的 RPC 通道，如 DataNode 与 NameNode 间的 RPC 通道。 • 客户端访问 Yarn 的 RPC 通道。 • NodeManager 和 ResourceManager 间的 RPC 通道。 • Spark 访问 Yarn，Spark 访问 HDFS 的 RPC 通道。 • Mapreduce 访问 Yarn，Mapreduce 访问 HDFS 的 RPC 通道。 • HBase 访问 HDFS 的 RPC 通道。 说明 用户可在 HDFS 组件的配置界面中设置该参数的值，设置后全局生效，即 Hadoop 中各模块的 RPC 通道的加密属性全部生效。 对 RPC 的加密方式，有如下三种取值： • “authentication”：普通模式默认值，指数据在鉴权后直接传输，不加密。这种方式能保证性能但	• 安全模式：privacy • 普通模式：authentication

参数	描述	默认值
	存在安全风险。 • “integrity”：指数据直接传输，即不加密也不鉴权。 为保证数据安全，请谨慎使用这种方式。 • “privacy”：安全模式默认值，指数据在鉴权及加密后再传输。这种方式会降低性能。	

9.20 在网络不稳定的情况下，降低客户端运行异常概率

配置场景

在网络不稳定的情况下，调整如下参数，降低客户端应用运行异常概率。

配置描述

请参考修改集群服务配置参数，进入 HDFS 的“全部配置”页面，在搜索框中输入参数名称。

表9-20 参数说明

参数	描述	默认值
ha.health-monitor.rpc-timeout.ms	zkfc 对 namenode 健康状态检查的超时时间。增大该参数值，可以防止出现双 Active NameNode，降低客户端应用运行异常的概率。 单位：毫秒。取值范围：30000~3600000	180000
ipc.client.connect.max.retries.on.timeouts	客户端与服务端建立 Socket 连接超时时，客户端的重试次数。 取值范围：1~256	45
ipc.client.connect.timeout	客户端与服务端建立 socket 连接的超时时间。增大该参数值，可以增加建立连接的超时时间。 单位：毫秒。取值范围：1~3600000	20000

9.21 配置 NameNode blacklist

配置场景

说明

本章节适用于 MRS 3.x 及后续版本。

在现有的缺省 DFSclient failover proxy provider 中，一旦某进程中的一个 NameNode 发生故障，在同一进程中的所有 HDFS client 实例都会尝试再次连接 NameNode，导致应用长时间等待超时。

当位于同一 JVM 进程中的客户端对无法访问的 NameNode 进行连接时，会对系统造成负担。为了避免这种负担，MRS 集群搭载了 NameNode blacklist 功能。

在新的 Blacklisting DFSClient failover provider 中，故障的 NameNode 将被记录至一个列表中。DFSClient 会利用这些信息，防止客户端再次连接这些 NameNode。该功能被称为 NameNode blacklisting。

例如，如下集群配置：

namenode：nn1、nn2

dfs.client.failover.connection.retries：20

单 JVM 中的进程：10 个客户端

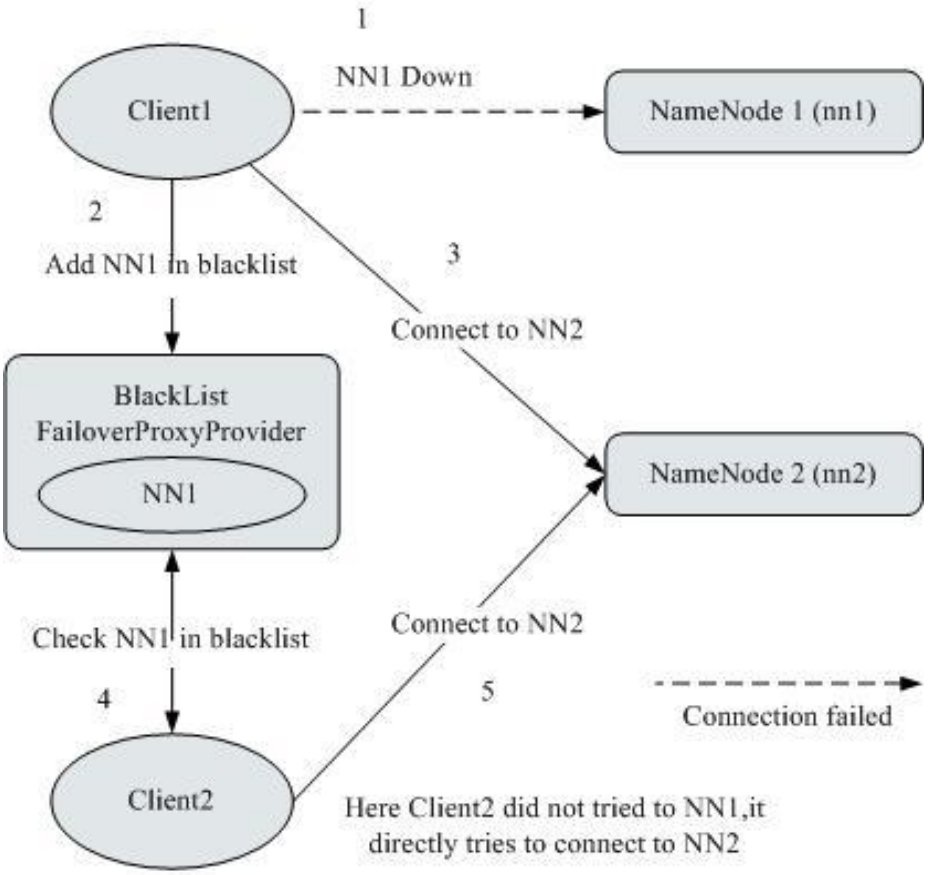
在上述集群中，如果当前处于 active 状态的 nn1 无法访问，client1 将会对 nn1 进行 20 次重新连接，之后发生故障转移，client1 将会连接至 nn2。与此相同，client2 至 client10 也会在对 nn1 进行 20 次重新连接后连接至 nn2。这样会延长 NameNode 的整体故障恢复时间。

针对该情况，当 client1 试图连接当前处于 active 状态的 nn1，但其已经发生故障时，nn1 将会被添加至 blacklist。这样其余 client 就不会连接已被添加至 blacklist 的 nn1，而是会选择连接 nn2。

说明

若在任一时刻，所有 NameNode 都被添加至 blacklist，则其内容会被清空，client 会按照初始的 NameNode list 重新尝试连接。若再次出现任何故障，NameNode 仍会被添加至 blacklist。

图9-3 NameNode blacklisting 状态图



配置描述

请参考修改集群服务配置参数，进入 HDFS 的“全部配置”页面，在搜索框中输入参数名称。

表9-21 NameNode blacklisting 的相关参数

参数	描述	默认值
dfs.client.failover.proxy.provider.*[nameservice ID]*	利用已通过的协议创建 namenode 代理的 Client Failover proxy provider 类。 将参数值设置为“org.apache.hadoop.hdfs.server.namenode.ha.BlackListingFailoverProxyProvider”， 可使用从 NameNode 支持读的特性。	org.apache.hadoop.hdfs.server.namenode.ha.AdaptiveFailoverProxyProvider

9.22 优化 HDFS NameNode RPC 的服务质量

配置场景

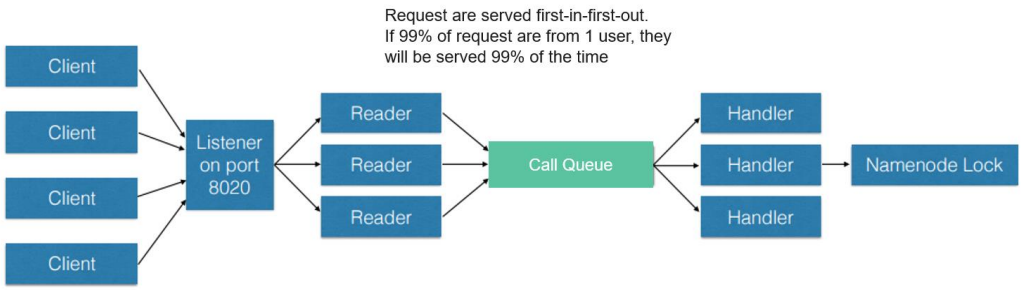
说明

本章节适用于 MRS 3.x 及后续版本。

数个成品 Hadoop 集群由于 NameNode 超负荷运行并失去响应而发生故障。

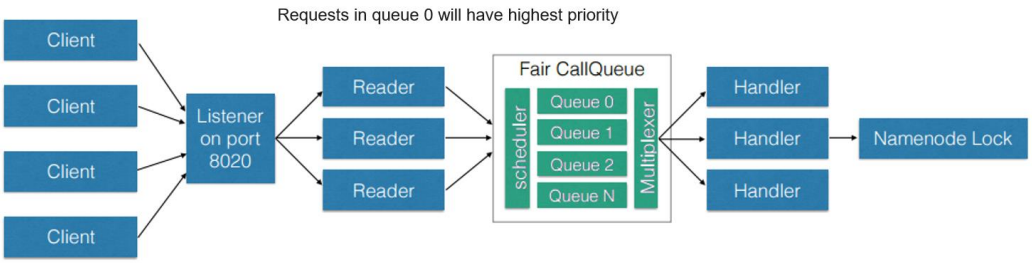
这种阻塞现象是由于 Hadoop 的初始设计造成的。在 Hadoop 中，NameNode 作为单独的机器，在其 namespace 内协调 HDFS 的各种操作。这些操作包括获取数据块位置，列出目录及创建文件。NameNode 接受 HDFS 的操作，将其视作 RPC 调用并置入 FIFO 调用队列，供读取线程处理。虽然 FIFO 在先到先服务的情况下足够公平，但如果用户执行的 I/O 操作较多，相比 I/O 操作较少的用户，将获得更多的服务。在这种情况下，FIFO 有失公平并且会导致延迟增加。

图9-4 基于 FIFO 调用队列的 NameNode 请求处理



如果将 FIFO 队列替换为一种被称作 FairCallQueue 的新型队列，这种情况就能够得到改善。按照这种方法，FAIR 队列会根据调用者的调用规模将传入的 RPC 调用分配至多个队列中。调度模块会跟踪最新的调用，并为调用量较小的用户分配更高的优先级。

图9-5 基于 FAIRCallQueue 的 NameNode 请求处理



配置描述

- FairCallQueue 通过在内部调整 RPC 调用的顺序确保服务质量。

该队列由以下三部分组成：

a. 调度模块（DecayRpcScheduler）用于提供从 0 至 N 的优先值数字（0 的优先级最高）。

b. 多级队列（位于 FairCallQueue 内部）保持调用在内部按优先级排列。

c. 多路转换器（提供有 WeightedRoundRobinMultiplexer）为队列选择提供逻辑控制。

在对 FairCallQueue 进行配置后，由控制模块决定将收到的调用分配至哪个子队列。当前调度模块为 DecayRpcScheduler。该模块仅持续对各类调用的优先级数字进行追踪，并周期性地对这些数字进行减小处理。

请参考修改集群服务配置参数，进入 HDFS 的“全部配置”页面，在搜索框中输入参数名称。

表9-22 Fair 调用队列参数

参数	描述	默认值
ipc.<*port*>.callqueue.impl	队列的实现类。用户需要通过“org.apache.hadoop.ipc.FairCallQueue”启用 QoS 特性。	java.util.concurrent.LinkedBlockingQueue

- RPC BackOff

  Backoff 是 FairCallQueue 的功能之一，要求客户端在一段时间后重试操作（如创建，删除，打开文件等）。当 Backoff 发生时，RCP 服务器将抛出 RetriableException 异常。FairCallQueue 在以下两种情况时进行 Backoff。

  - 当队列已满，即队列中有许多客户端调用时。
  - 当队列的响应时间大于配置的阈值（由参数“ipc.<port>.decay-scheduler.backoff.responsetime.thresholds”决定）时。

表9-23 RPC BackOff 配置

参数	描述	默认值
ipc.<*port*>.backoff.enable	启用 Backoff 配置参数。当前，如果应用程序中包含较多的用户调用，假设没有达到操作系统的连接限制，则 RPC 请求将处于阻塞状态。或者，当 RPC 或 NameNode 在重负载时，可以基于某些策略将一些明确定义的异常抛回给客户端，客户端将理解这种异常并进行指数回退，以此作为类 RetryInvocationHandler 的另一个实现。	false

参数	描述	默认值
ipc.<*port*>.decay-scheduler.backoff.responsetime.enable	根据队列平均响应时间启用 Backoff。	false
ipc.<*port*>.decay-scheduler.backoff.responsetime.thresholds	配置每个队列的响应时间阈值。ResponseTime 阈值必须与优先级数目（ipc.<port> .faircallqueue.priority-levels）相匹配。单位：毫秒。	10000,20000,30000,40000

说明

- <port>表示在 NameNode 上配置的 RPC 端口。
- 只有在“ipc.<port> .backoff.enable”为“true”时，响应时间 backoff 功能才会起作用。

9.23 优化 HDFS DataNode RPC 的服务质量

配置场景

当客户端写入 HDFS 的速度大于 DataNode 的硬盘带宽时，硬盘带宽会被占满，导致 DataNode 失去响应。客户端只能通过取消或恢复通道进行规避，这会导致写入失败及不必要的通道恢复操作。

说明

本章节适用于 MRS 3.x 及后续版本。

配置步骤

引入了新的配置参数“dfs.pipeline.ecn”。当该配置启用时，DataNode 会在写入通道超出负荷时从其中发出信号。客户端可以基于该阻塞信号进行退避，从而防止系统超出负荷。引入该配置参数的目的是为了使通道更加稳定，并减少不必要的取消或恢复操作。收到信号后，客户端会退避一定的时间（5000ms），然后根据相关过滤器调整退避时间（单次退避最长时间为 50000ms）。

请参考修改集群服务配置参数，进入 HDFS 的“全部配置”页面，在搜索框中输入参数名称。

表9-24 DN ECN 配置

参数	描述	缺省值
dfs.pipeline.ecn	进行该配置后，DataNode 能够向客户端发送阻塞通	false

参数	描述	缺省值
	知。	

9.24 配置 LZC 压缩

配置场景

文件压缩带来了两个主要好处：减少了储存文件的空间，并且提高数据从磁盘读取和网络传输的速度。HDFS 有 gzip 和 Snappy 这两种默认压缩格式。本章节为 HDFS 新增加的压缩格式 LZC(Lempel-Ziv Compression)提供配置方法。这种压缩格式增强了 Hadoop 压缩能力。有关 Snappy 的详细信息，请参阅 http://code.google.com/p/snappy/。

说明

本章节适用于 MRS 3.x 及后续版本。

配置描述

为了使 LZC 压缩生效，需要在 client 客户端的配置文件“core-site.xml”中（例如“客户端安装路径/HDFS/hadoop/etc/hadoop/”）配置下面的参数。

表9-25 参数描述

参数	描述	默认值
io.compression.codecs	为了使 LZC 压缩格式生效，在现有的压缩格式列表中增加了下面的值： “com.xxx.hadoop.datasight.io.compress.lzc.ZCodec” 说明 若配置了多于一种的压缩格式需要使用逗号分隔。	org.apache.hadoop.io.compress.BZip2Codec,org.apache.hadoop.io.compress.DefaultCodec,org.apache.hadoop.io.compress.DeflateCodec,org.apache.hadoop.io.compress.Lz4Codec,org.apache.hadoop.io.compress.SnappyCodec,org.apache.hadoop.io.compress.GzipCodec,org.apache.hadoop.io.compress.ZStandardCodec,com.xxx.hadoop.datasight.io.compress.lzc.ZCodec
io.compression.codec.lzc.class	为了使 LZC 压缩格式生效，配置参数值为“com.xxx.hadoop.datasight.io.compress.lzc.ZCodec”。	com.xxx.hadoop.datasight.io.compress.lzc.ZCodec

说明

1. LZC 压缩格式不支持 FSImage 和 SequenceFile 压缩。
2. 当前 HDFS 提供了多种压缩算法，包括 Gzip、LZ4、Snappy、Bzip2 等。这几种压缩算法的压缩比和解压速度可参考如下：

   压缩比排序：Bzip2>Gzip>LZ4>Snappy

   解压速度排序：LZ4>Snappy>Gzip>Bzip2
3. 使用场景建议：

- 追求速度的场景（如 Mapreduce 任务中间数据的存储等）——建议使用 LZ4 和 Snappy（高可靠场景，建议使用 Snappy）。
- 追求压缩比，而对压缩速度要求不高的场景（如冷数据的保存）—— 建议使用 Bzip2 或 Gzip。

4. 上述压缩算法除 LZC 外，皆支持 Native（基于 C 语言实现）实现，压缩和解压缩效率较高。建议根据业务场景优先选用具备 Native 实现的压缩算法。

9.25 配置 DataNode 预留磁盘百分比

配置场景

当 YARN 本地目录和 DataNode 目录配置在同一个磁盘时，具有较大容量的磁盘可以运行更多的任务，因此将有更多的中间数据存储在 YARN 本地目录。

目前 DataNode 支持通过配置“dfs.datanode.du.reserved”来配置预留磁盘空间大小的绝对值。配置较小的数值不能满足更大的磁盘要求。但对于更小的磁盘配置更大的数值将浪费大量的空间。

为了避免这种情况，添加一个新的参数“dfs.datanode.du.reserved.percentage”来配置预留磁盘空间占总磁盘空间大小的百分比，那样可以基于总的磁盘空间来预留磁盘百分比。

说明

- 如果用户同时配置“dfs.datanode.du.reserved.percentage”和“dfs.datanode.du.reserved”，则采用这两个参数较大的数值作为 DataNode 的预留空间大小。
- 建议基于磁盘空间设置“dfs.datanode.du.reserved”或者“dfs.datanode.du.reserved.percentage”。

配置描述

请参考修改集群服务配置参数，进入 HDFS 的“全部配置”页面，在搜索框中输入参数名称。

表9-26 参数描述

参数	描述	默认值
dfs.datanode.du.reserved.percentage	DataNode 预留空间占总磁盘空间大小的百分比。DataNode 会永久预留由此百分比计算得出的磁盘空间大小。 整数值，取值范围是 0～100。	10

9.26 配置 HDFS NodeLabel

配置场景

用户需要通过数据特征灵活配置 HDFS 文件数据块的存储节点。通过设置 HDFS 目录/文件对应一个标签表达式，同时设置每个 Datanode 对应一个或多个标签，从而给文件的数据块存储指定了特定范围的 Datanode。

当使用基于标签的数据块摆放策略，为指定的文件选择 DataNode 节点进行存放时，会根据文件的标签表达式选择出 Datanode 节点范围，然后在这些 Datanode 节点范围内，选择出合适的存放节点。

说明

本章节适用于 MRS 3.x 及后续版本。

开启单集群跨 AZ 高可用后，不支持配置 HDFS NodeLabel 功能。

- 场景 1 DataNodes 分区场景。

  场景说明：

  用户需要让不同的应用数据运行在不同的节点，分开管理，就可以通过标签表达式，来实现不同业务的分离，指定业务存放到对应的节点上。

  通过配置 NodeLabel 特性使得：

  - /HBase 下的数据存储在 DN1、DN2、DN3、DN4 节点上。
  - /Spark 下的数据存储在 DN5、DN6、DN7、DN8 节点上。

图9-6 DataNode 分区场景

说明

- 通过 **hdfs nodelabel -setLabelExpression -expression 'LabelA[fallback=NONE]' -path /Hbase** 命令，给 Hbase 目录设置表达式。从图 9-6 中可知，“/Hbase”文件的数据块副本会被放置在有 LabelA 标签的节点上，即 DN1、DN2、DN3、DN4。同理，通过 **hdfs nodelabel -setLabelExpression -expression 'LabelB[fallback=NONE]' -path /Spark** 命令，给 Spark 目录设置表达式。在“/Spark”目录下文件对应的数据块副本只能放置到 LabelB 标签上的节点，如 DN5、DN6、DN7、DN8。
- 设置数据节点的标签参考配置描述。
- 如果同一个集群上存在多个机架，每个标签下最好有多个机架的 datanodes，以确保数据块摆放的可靠性。

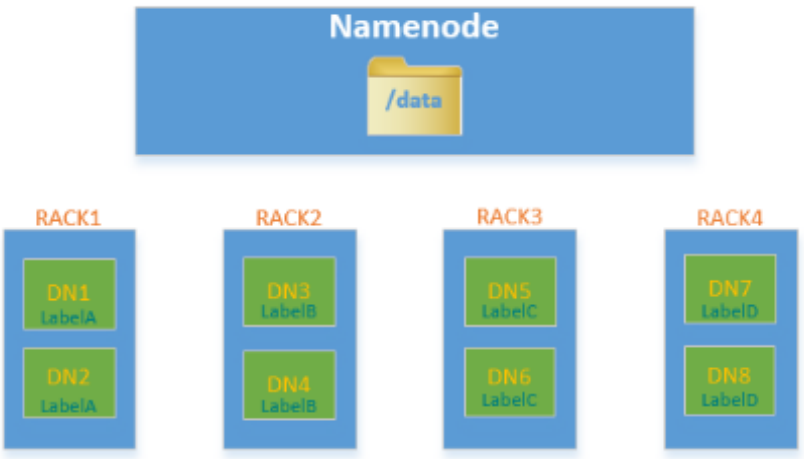
- 场景 2 多机架下指定副本位置场景

  场景说明：

  在异构集群中，客户需要分配一些特定的具有高可靠性的节点用以存放重要的商业数据，可以通过标签表达式指定副本位置，指定文件数据块的其中一个副本存放到高可靠性的节点上。

  “/data”目录下的数据块，默认三副本情况下，其中至少有一个副本会被存放到 RACK1 或 RACK2 机架的节点上（RACK1 和 RACK2 机架的节点为高可靠性节点），另外两个副本会被分别存放到 RACK3 和 RACK4 机架的节点上。

图9-7 场景样例



说明

通过 **hdfs nodelabel -setLabelExpression -expression 'LabelA||LabelB[fallback=NONE],LabelC,LabelD' -path /data** 命令给“/data”目录设置表达式。

当向“/data”目录下写数据时，至少有一个数据块副本存放在 LabelA 或者 LabelB 标签的节点中，剩余的两个数据块副本会被存放在有 LabelC 和 LabelD 标签的节点上。

配置描述

- Datanode 节点标签配置

  请参考修改集群服务配置参数，进入 HDFS 的“全部配置”页面，在搜索框中输入参数名称。

表9-27 参数说明

参数	描述	默认值
dfs.block.replicator.classname	配置 HDFS 的 DataNode 原则策略。 如果需要开启 NodeLabeNodel 功能，需要将该值设置为 org.apache.hadoop.hdfs.server.blockmanagement.BlockPlacementPolicyWithNodeLabel。	org.apache.hadoop.hdfs.server.blockmanagement.AvailableSpaceBlockPlacementPolicy
host2tags	配置 DataNode 主机与标签的对应关系。 主机名称支持配置 IP 扩展表达式（如 192.168.1.[1-128]或者 192.168.[2-3].[1-128]，且 IP 必须为业务 IP），或者为前后加上 / 的主机名的正则表达式（如 /datanode-[123]/或者/datanode-\d{2}/）。标签配置名称不允许包含 =/\ 字符。【注意】配置 IP 时必须是业务 IP。	-

说明

- host2tags 配置项内容详细说明：

  假如有一套集群，有 20 个 Datanode：dn-1 到 dn-20，对应的 IP 地址为 10.1.120.1 到 10.1.120.20，host2tags 配置文件内容可以使用如下的表示方式。

  主机名正则表达式

  “/dn-\d/ = label-1”表示 dn-1 到 dn-9 对应的标签为 label-1，即 dn-1 = label-1，dn-2 = label-1，...dn-9 = label-1。

  “/dn-((1[0-9]$)|(20$))/ = label-2”表示 dn-10 到 dn-20 对应的标签为 label-2，即 dn-10 = label-2，dn-11 = label-2，...dn-20 = label-2。

  IP 地址范围表示方式

  “10.1.120.[1-9] = label-1”表示 10.1.120.1 到 10.1.120.9 对应的标签为 label-1，即 10.1.120.1 = label-1，10.1.120.2 = label-1，...10.1.120.9 = label-1。

  “10.1.120.[10-20] = label-2”表示 10.1.120.10 到 10.1.120.20 对应的标签为 label-2，即 10.1.120.10 = label-2，10.1.120.11 = label-2，...10.1.120.20 = label-2。

- 基于标签的数据块摆放策略支持扩容减容场景：

  当集群中新增加 DataNode 节点时，如果该 DataNode 对应的 IP 匹配 **host2tags** 配置项中的 IP 地址范围，或者该 DataNode 的主机名匹配 **host2tags** 配置项中的主机名正则表达式，则该 DataNode 节点会被设置成对应的标签。

  例如“host2tags”配置值为 10.1.120.[1-9] = label-1，而当前集群只有 10.1.120.1 到 10.1.120.3 三个数据节点。进行扩容后，又添加了 10.1.120.4 这个数据节点，则该数据节点会被设置成 label-1 的标签；如果 10.1.120.3 这个数据节点被删除或者退出服务后，数据块不会再被分配到该节点上。

- 设置目录/文件的标签表达式
  - 在 HDFS 参数配置页面配置“path2expression”，配置 HDFS 目录与标签的对应关系。当配置的 HDFS 目录不存在时，也可以配置成功，新建不存在的同名目录，已设置的标签对应关系将在 30 分钟之内被继承。设置了标签的目录被删除后，新增一个同名目录，原有的对应关系也将在 30 分钟之内被继承。
  - 命令行设置方式请参考 **hdfs nodelabel -setLabelExpression** 命令。
  - Java API 设置方式通过 NodeLabelFileSystem 实例化对象调用 setLabelExpression(String src, String labelExpression)方法。*src* 为 HDFS 上的目录或文件路径，“labelExpression”为标签表达式。
- 开启 NodeLabel 特性后，可以通过命令 **hdfs nodelabel -listNodeLabels** 查看每个 Datanode 的标签信息。

块副本位置选择

Nodelabel 支持对各个副本的摆放采用不同的策略，如表达式“label-1,label-2,label-3”，表示 3 个副本分别放到含有 label-1、label-2、label-3 的 DataNode 中，不同的副本策略用逗号分隔。

如果 label-1，希望放 2 个副本，可以这样设置表达式：“label-1[replica=2],label-2,label-3”。这种情况下，如果默认副本数是 3，则会选择 2 个带有 label-1 和一个 label-2 的节点；如果默认副本数是 4，会选择 2 个带有 label-1、一个 label-2 以及一个 label-3 的节点。可以注意到，副本数是从左到右依次满足各个副本策略的，但也有副本数超过表达式表述的情况，当默认副本数为 5 时，多出来的一个副本会放到最后一个节点中，也就是 label-3 的节点里。

当启用 ACLs 功能并且用户无权访问表达式中使用的标签时，将不会为副本选择属于该标签的 DataNode。

多余块副本删除选择

如果块副本数超过参数“dfs.replication”值（即用户指定的文件副本数），hdfs 会删除多余块副本来保证集群资源利用率。

删除规则如下：

- 优先删除不满足任何表达式的副本。

  示例：文件默认副本数为 3

  /test 标签表达式为“LA[replica=1],LB[replica=1],LC[replica=1]”，

  /test 文件副本分布的四个节点（D1～D4）以及对应标签（LA～LD）：

  ```
  D1:LA
  D2:LB
  D3:LC
  D4:LD
  ```

  则选择删除 D4 节点上的副本块。
- 如果所有副本都满足表达式，删除多于表达式指定的数量的副本。

  示例：文件默认副本数为 3

  /test 标签表达式为“LA[replica=1],LB[replica=1],LC[replica=1]”，

  /test 文件副本分布的四个节点以及对应标签：

  ```
  D1:LA
  D2:LA
  D3:LB
  D4:LC
  ```

  则选择删除 D1 或者 D2 上的副本块。
- 如果文件所有者或文件所有者的组不能访问某个标签，则优先删除映射到该标签的 DataNode 中的副本。

基于标签的数据块摆放策略样例

假如有一套集群，有六个 DataNode：dn-1，dn-2，dn-3，dn-4，dn-5 以及 dn-6，对应的 IP 为 10.1.120.[1-6]。有六个目录需要配置标签表达式，Block 默认备份数为 3。

- 下面给出 3 种 DataNode 标签信息在“host2labels”文件中的表示方式，其作用是一样的。
  - 主机名正则表达式

    ```
    /dn-[1456]/ = label-1,label-2
    /dn-[26]/ = label-1,label-3
    ```

```
/dn-[3456]/ = label-1,label-4
/dn-5/ = label-5
```

- IP 地址范围表示方式

```
10.1.120.[1-6] = label-1
10.1.120.1 = label-2
10.1.120.2 = label-3
10.1.120.[3-6] = label-4
10.1.120.[4-6] = label-2
10.1.120.5 = label-5
10.1.120.6 = label-3
```

- 普通的主机名表达式

```
/dn-1/ = label-1, label-2
/dn-2/ = label-1, label-3
/dn-3/ = label-1, label-4
/dn-4/ = label-1, label-2, label-4
/dn-5/ = label-1, label-2, label-4, label-5
/dn-6/ = label-1, label-2, label-3, label-4
```

- 目录的标签表达式设置结果如下：

```
/dir1 = label-1
/dir2 = label-1 && label-3
/dir3 = label-2 || label-4[replica=2]
/dir4 = (label-2 || label-3) && label-4
/dir5 = !label-1
/sdir2.txt = label-1 && label-3[replica=3,fallback=NONE]
/dir6 = label-4[replica=2],label-2
```

说明

标签表达式设置方式请参考 hdfs nodelabel -setLabelExpression 命令。

文件的数据块存放结果如下：

- “/dir1”目录下文件的数据块可存放在 dn-1，dn-2，dn-3，dn-4，dn-5 和 dn-6 六个节点中的任意一个。
- “/dir2”目录下文件的数据块可存放在 dn-2 和 dn-6 节点上。Block 默认备份数为 3，表达式只匹配了两个 DataNode 节点，第三个副本会在集群上剩余的节点中选择一个 DataNode 节点存放。
- “/dir3”目录下文件的数据块可存放在 dn-1，dn-3，dn-4，dn-5 和 dn-6 中的任意三个节点上。
- “/dir4”目录下文件的数据块可存放在 dn-4，dn-5 和 dn-6。
- “/dir5”目录下文件的数据块没有匹配到任何一个 DataNode，会从整个集群中任意选择三个节点存放（和默认选块策略行为一致）。
- “/sdir2.txt”文件的数据块，两个副本存放在 dn-2 和 dn-6 节点上，虽然还缺失一个备份节点，但由于使用了 fallback=NONE 参数，所以只存放两个备份。
- “/dir6”目录下文件的数据块在具备 label-4 的节点中选择 2 个节点(dn-3 -- dn-6)，然后在 label-2 中选择一个节点，如果用户指定“/dir6”下文件副本数大于 3，则多出来的副本均在 label-2。

使用限制

配置文件中，“key”、“value”是以“=”、“：”及空白字符作为分隔的。因此，“key”对应的主机名中间请勿包含以上字符，否则会被误认为分隔符。

9.27 配置 HDFS Mover

配置场景

Mover 是一个新的数据迁移工具，工作方式与 HDFS 的 Balancer 接口工作方式类似。Mover 能够基于设置的数据存储策略，将集群中的数据重新分布。

通过运行 Mover，周期性地检测 HDFS 文件系统中用户指定的 HDFS 文件或目录，判断该文件或目录是否满足设置的存储策略，如果不满足，则进行数据迁移，使目标目录或文件满足设定的存储策略。

说明

本章节适用于 MRS 3.x 及后续版本。

配置描述

请参考修改集群服务配置参数，进入 HDFS 的“全部配置”页面，在搜索框中输入参数名称。

表9-28 参数说明

参数	描述	默认值
dfs.mover.auto.enable	是否开启数据副本迁移功能，该功能支持多种。默认值为“false”，表示关闭该特性。	false
dfs.mover.auto.cron.expression	HDFS 执行自动数据迁移的 CRON 表达式，用于控制数据迁移操作的开始时间。仅当 dfs.mover.auto.enable 设置为 true 时才有效。默认值“0 * * * *”表示在每个整点执行任务。表达式的具体含义可参见表 9-29。	0 * * * *
dfs.mover.auto.hdfsfiles_or_dirs	指定集群执行自动副本迁移的 HDFS 文件或目录列表，以空格分隔。仅当 dfs.mover.auto.enable 设置为 true 时才有效。	-

表9-29 Cron 表达式解释

列	说明
第 1 列	分钟，参数值为 0~59。
第 2 列	小时，参数值为 0~23。

列	说明
第 3 列	日期，参数值为 1~31。
第 4 列	月份，参数值为 1~12。
第 5 列	星期，参数值为 0~6，0 表示星期日。

使用限制

若要在 HDFS 的客户端通过命令行执行 mover 功能，其命令格式如下：

hdfs mover -p <*HDFS 文件全路径或目录路径*>

说明

在客户端执行此命令时，用户需要具备 supergroup 权限。可以使用 HDFS 服务的系统用户 hdfs。或者在集群上创建一个具有 supergroup 权限的用户，再在客户端中执行此命令。

9.28 使用 HDFS AZ Mover

操作场景

AZ Mover 是一个副本迁移工具，用来移动副本以满足目录上设置的新 AZ 策略。它可以用来从一个 AZ 策略迁移到另一个 AZ 策略，AZ Mover 通过指示 NameNode 按照新的 AZ 策略来移动副本，如果 NameNode 拒绝删除旧副本就不能保证一定能满足新的策略，例如副本被标记为过时等原因。

使用限制

- 将策略更改为 LOCAL_AZ 与更改为 ONE_AZ 相同，因为上传文件写入时无法确定写入期间的客户端位置。
- Mover 无法确定 AZ 的状态，因此可能会导致将副本移动到异常的 AZ，并依赖 NameNode 来进一步处理。
- Mover 依赖于每个 AZ 的 DataNode 节点数达到最小要求，如果在一个 DataNode 节点数很少的 AZ 执行，可能会导致与预期不同的结果。
- Mover 只满足 AZ 级别的策略，并不保证满足基本 BPP。
- Mover 不支持更改复制因子，新旧 AZ 策略之间的副本计数差异会导致异常结果。

操作步骤

步骤 1 执行以下命令，切换到客户端安装目录。

cd /opt/client

步骤 2 执行以下命令配置环境变量。

source bigdata_env

步骤 3 如果集群为安全模式，执行的用户需要源目录或文件读权限，目的目录有写权限，且执行以下命令进行用户认证。普通模式集群无需执行用户认证。

kinit *组件业务用户*

步骤 4 创建目录并设置 AZ 策略。

执行以下命令创建目录：

hdfs dfs -mkdir <*path*>

执行以下命令设置 AZ 策略，azexpression 代表 AZ 策略：

hdfs dfsadmin -setAZExpression <*path*> <*azexpression*>

执行以下命令查看 AZ 策略：

hdfs dfsadmin -getAZExpression <*path*>

步骤 5 在目录中上传文件。

hdfs dfs -put <*localfile*> <*hdfs-path*>

步骤 6 删除目录上的旧策略，再设置一个新的策略。

执行以下命令清楚旧策略：

hdfs dfsadmin -clearAZExpression <*path*>

执行以下命令设置新策略：

hdfs dfsadmin -setAZExpression <*path*> <*azexpression*>

步骤 7 执行 **azmover** 命令，使副本分布满足新的 AZ 策略。

hdfs azmover -p /targetDirecotry

----结束

9.29 配置 HDFS DiskBalancer

配置场景

DiskBalancer 是一个在线磁盘均衡器，旨在根据各种指标重新平衡正在运行的 DataNode 上的磁盘数据。工作方式与 HDFS 的 Balancer 工具类似。不同的是，HDFS Balancer 工具用于 DataNode 节点间的数据均衡，而 HDFS DiskBalancer 用于单个 DataNode 节点上各磁盘之间的数据均衡。

长时间运行的集群会因为曾经删除过大量的文件，或者集群中的节点做磁盘扩容等操作导致节点上出现磁盘间数据不均衡的现象。磁盘间数据不均衡会引起 HDFS 整体并发读写性能的下降或者因为不恰当的 HDFS 写策略导致业务故障。此时需要平衡节点磁盘间的数据密度，防止异构的小磁盘成为该节点的性能瓶颈。

说明

本章节适用于 MRS 3.x 及后续版本。

配置描述

请参考修改集群服务配置参数，进入 HDFS 的“全部配置”页面，在搜索框中输入参数名称。

表9-30 参数说明

参数	描述	默认值
dfs.disk.balancer.auto.enabled	是否开启自动执行 HDFS diskbalancer 特性。默认值为“false”，表示关闭该特性。	false
dfs.disk.balancer.auto.cron.expression	HDFS 磁盘均衡操作的 CRON 表达式，用于控制均衡操作的开始时间。仅当 dfs.disk.balancer.auto.enabled 设置为 true 时才有效。默认值“0 1 * * 6”表示在每周六的 1 点执行任务。表达式的具体含义可参见表 9-31。默认值表示每周六一点执行。	0 1 * * 6
dfs.disk.balancer.max.disk.throughputInMBperSec	执行磁盘数据均衡时可使用的最大磁盘带宽。单位为 MB/s，默认值为 10，可依据集群的实际磁盘条件设置。	10
dfs.disk.balancer.max.disk.errors	设置我们能够容忍的在指定的移动过程中出现的最大错误次数，超过此阈值则移动失败。	5
dfs.disk.balancer.block.tolerance.percent	设置磁盘之间进行数据均衡操作时，各个磁盘的数据存储量与完美状态之间的差异阈值。例如，各个磁盘的理想数据存储量为 1TB，此参数设置为 10。那么，当目标磁盘的数据存储量达到 900GB 时，就认为该磁盘的存储状态就已经足够好了。取值范围[1-100]。	10
dfs.disk.balancer.plan.threshold.percent	设置在磁盘数据均衡中可容忍的两磁盘之间的数据密度域值差。如果任意两个磁盘数据密度差值的绝对值超过了此阈值，意味着对应的磁盘应该进行数据均衡。取值范围[1-100]。	10
dfs.disk.balancer.top.nodes.number	该参数用来指定集群中需要执行磁盘数据均衡的 Top N 节点。	5

使用此功能时，需要先将参数 dfs.disk.balancer.auto.enabled 设置为 true，并配置合理的 CRON 表达式。其它参数依据集群状况设置。

表9-31 CRON 表达式解释

列	说明
第 1 列	分钟，参数值为 0~59。

列	说明
第 2 列	小时，参数值为 0~23。
第 3 列	日期，参数值为 1~31。
第 4 列	月份，参数值为 1~12。
第 5 列	星期，参数值为 0~6，0 表示星期日。

使用限制

1. 只支持同类型磁盘之间的数据移动，例如 SSD->SSD，DISK->DISK 等。
2. 执行该特性会占用涉及节点的磁盘 IO 资源、网络带宽资源，请尽量在业务不繁忙的时候使用。
3. 参数 dfs.disk.balancer.top.nodes.number 指定 Top N 节点返回的 DataNode 列表是不断重新计算的，因此不必设置的过大。
4. 如果要在 HDFS 客户端通过命令行使用 DiskBalancer 功能，其接口如下：

表9-32 DiskBalancer 功能的接口说明

命令格式	说明
hdfs diskbalancer -report -top <N>	N 可以指定为大于 0 的整数，先利用此条命令查询集群中最需要执行磁盘数据均衡的 Top N 节点。
hdfs diskbalancer -plan <Hostname| IP Address>	此条命令可以根据传入的 DN 生成一个 Json 文件，该文件包含了数据移动的源磁盘、目标磁盘、待移动的块等信息。同时，该命令还支持指定一些其他网络带宽参数等。
hdfs diskbalancer -query <Hostname:$dfs.datanode.ipc.port>	集群默认的 port 值为 9867。此条命令可以查询当前节点上运行的 DiskBalancer 任务的运行状态。
hdfs diskbalancer -execute <planfile>	此命令中的 planfile 指的是第二条命令中生成的 Json 文件，请使用绝对路径。
hdfs diskbalancer -cancel <planfile>	取消正在运行的 planfile，同样需要使用绝对路径。

说明

- 在客户端执行此命令时，用户需要具备 supergroup 权限。可以使用 HDFS 服务的系统用户 hdfs。或者在集群上创建一个具有 supergroup 权限的用户，再在客户端中执行此命令。

- 表 9-32 只说明了命令接口的含义及使用方法，实际每个接口提供了更多的配置参数。具体信息可通过"hdfs diskbalancer -help <command>"命令查看。
- 在集群运维过程中，排查性能类问题时。可查看集群的事件信息中是否有 HDFS 磁盘均衡任务事件发生，如果有的话。可以排查集群中是否开启了 DiskBalancer。
- 自动执行磁盘均衡的特性开启以后，会在此次数据均衡执行完成之后才会退出。无法在执行均衡中途取消本次执行任务。
- 如果想要灵活选择某些指定节点进行数据均衡，可以在客户端手动指定执行。

9.30 配置从 NameNode 支持读

配置场景

在配置了 HA 的 HDFS 集群中，存在一个主 NameNode 和一个备 NameNode。主 NameNode 处理所有的客户端请求，备 NameNode 保持最新的元数据信息和块位置信息。但是在这种架构存在一个缺点：主 NameNode 会成为客户端请求处理的瓶颈，在请求繁忙的集群中表现更为明显。

为了解决主 NameNode 的瓶颈问题，引入了一个新状态的 NameNode：从 NameNode。从 NameNode 类似于备 NameNode，也保持着最新的元数据信息和块位置信息。除此之外，从 NameNode 也可以像主 NameNode 一样处理客户端的读请求。由于在典型的 HDFS 集群中，读请求占大多数，因此从 NameNode 支持读可以降低主 NameNode 的负载，提高集群处理能力。

说明

本章节适用于 MRS 3.x 及后续版本。

对系统的影响

- 配置从 NameNode 支持读可以降低主 NameNode 的负载，提高 HDFS 集群的处理能力，尤其是在大集群下效果明显。
- 配置从 NameNode 支持读需要更新客户端应用配置。

前提条件

- 已安装 HDFS 集群，主备 NameNode 正常，HDFS 服务正常。
- 规划安装从 NameNode 的节点已经创建“${BIGDATA_DATA_HOME}/namenode”分区。

操作步骤

以配置 hacluster 的从 NameNode 支持读为例来说明，如果集群中有多对 NameService，且都在使用，可参考如下步骤为每对 NameService 配置从 NameNode 支持读。

步骤 1 登录 FusionInsight Manager 页面。

步骤 2 选择“集群 > *待操作集群的名称* > 服务 > HDFS > 管理 NameService”。

步骤 3 单击 hacluster 后的“添加”按钮。

步骤 4 在添加 NameNode 页面，“NameNode 类型”选择“从”，单击“下一步”。

步骤 5 在分配角色页面，选择已规划的主机，添加从 NameNode，单击“下一步”。

说明

每对 NameService 最多可添加 5 个从 NameNode。

步骤 6 在配置页面，按照规划配置 NameNode 的存储目录、端口等信息，单击“下一步”。

步骤 7 确认信息无误，单击“提交”，等待从 NameNode 安装完成。

步骤 8 重启依赖 HDFS 的上层组件，更新客户端应用配置，重启客户端应用。

----结束

9.31 使用 HDFS 文件并发操作命令

操作场景

集群内并发修改文件和目录的权限及访问控制的工具。

说明

本章节适用于 MRS 3.x 及后续版本。

对系统的影响

因为集群内使用文件并发修改命令会对集群性能造成较大负担，所以在集群空闲时使用文件并发操作命令。

前提条件

- 已安装 HDFS 客户端或者包括 HDFS 的客户端。例如安装目录为“/opt/client”。
- 各组件业务用户由系统管理员根据业务需要创建。安全模式下，“机机”用户需要下载 keytab 文件。“人机”用户第一次登录时需修改密码（普通模式不涉及）。

操作步骤

步骤 1 以客户端安装用户，登录安装客户端的节点。

步骤 2 执行以下命令，切换到客户端安装目录。

cd /opt/client

步骤 3 执行以下命令配置环境变量。

source bigdata_env

步骤 4 如果集群为安全模式，执行的用户所属的用户组必须为 **supergroup** 组，且执行以下命令进行用户认证。普通模式集群无需执行用户认证。

kinit *组件业务用户*

步骤 5 增大客户端的 JVM 大小，防止 OOM，方法如下。（1 亿文件建议 **32G**）

说明

若执行 HDFS 客户端命令时，客户端程序异常退出，并且报“java.lang.OutOfMomeryError”错误。

这个问题是由于 HDFS 客户端运行时的所需的内存超过了 HDFS 客户端设置的内存上限（默认 128M）。可通过修改“<*客户端安装路径*>/HDFS/component_env”中的“CLIENT_GC_OPTS”来修改 HDFS 客户端的内存上限。例如，需要设置内存上限为 1GB，则设置：

CLIENT_GC_OPTS="-Xmx1G"

在修改完后，使用如下命令刷新客户端配置，使之生效;

source <*客户端安装路径*>**/bigdata_env**

步骤 6 直接执行并发命令，命令详情如下表。

命令	参数及说明	命令作用
hdfs quickcmds [-t threadsNumber] [-p principal] [-k keytab] -setrep <rep> <path> ...	threadsNumber：并发线程数，默认为本机 CPU 核数 principal：Kerberos 用户 keytab：Keytab 文件 rep：副本数 path：HDFS 目录	多并发设置目录中所有文件的副本数
hdfs quickcmds [-t threadsNumber] [-p principal] [-k keytab] -chown [owner][:[group]] <path> ...	threadsNumber：并发线程数，默认为本机 CPU 核数 principal：Kerberos 用户 keytab：Keytab 文件 owner：所属用户 group：所属组 path：HDFS 目录	多并发设置目录中所有文件的属组
hdfs quickcmds [-t threadsNumber] [-p principal] [-k keytab] -chmod <mode> <path> ...	threadsNumber：并发线程数，默认为本机 CPU 核数 principal：Kerberos 用户 keytab：Keytab 文件 mode：权限（如 754） path：HDFS 目录	多并发设置目录中所有文件的权限
hdfs quickcmds [-t threadsNumber] [-p principal] [-k keytab] -setfacl [{-b|-k} {-m|-x <acl_spec>} <path> ...]|[--set <acl_spec> <path> ...]	threadsNumber：并发线程数，默认为本机 CPU 核数 principal：Kerberos 用户 keytab：Keytab 文件	多并发设置目录中所有文件的 ACL 信息

命令	参数及说明	命令作用
	acl_spec：逗号分隔的 ACL 列表 path：HDFS 目录	

----结束

9.32 HDFS 日志介绍

日志描述

日志存储路径：HDFS 相关日志的默认存储路径为“/var/log/Bigdata/hdfs/角色名”

- NameNode：“/var/log/Bigdata/hdfs/nn”（运行日志），“/var/log/Bigdata/audit/hdfs/nn”（审计日志）。
- DataNode：“/var/log/Bigdata/hdfs/dn”（运行日志），“/var/log/Bigdata/audit/hdfs/dn”（审计日志）。
- ZKFC：“/var/log/Bigdata/hdfs/zkfc”（运行日志），“/var/log/Bigdata/audit/hdfs/zkfc”（审计日志）。
- JournalNode：“/var/log/Bigdata/hdfs/jn”（运行日志），“/var/log/Bigdata/audit/hdfs/jn”（审计日志）。
- Router：“/var/log/Bigdata/hdfs/router”（运行日志），“/var/log/Bigdata/audit/hdfs/router”（审计日志）。
- HttpFS：“/var/log/Bigdata/hdfs/httpfs”（运行日志），“/var/log/Bigdata/audit/hdfs/httpfs”（审计日志）。

日志归档规则：HDFS 的日志启动了自动压缩归档功能，默认情况下，当日志大小超过 100MB 的时候，会自动压缩，压缩后的日志文件名规则为：“<原有日志名>-<yyyy-mm-dd_hh-mm-ss>.[编号].log.zip”。最多保留最近的 100 个压缩文件，压缩文件保留个数可以在 Manager 界面中配置。

表9-33 HDFS 日志列表

日志类型	日志文件名	描述
运行日志	hadoop-<SSH_USER>-<process_name>-<hostname>.log	HDFS 系统日志，记录 HDFS 系统运行时候所产生的大部分日志。
	hadoop-<SSH_USER>-<process_name>-<hostname>.out	HDFS 运行环境信息日志。
	hadoop.log	Hadoop 客户端操作日志。
	hdfs-period-check.log	周期运行的脚本的日志记录。包括：自动均衡、数据迁移、journalnode 数据

日志类型	日志文件名	描述
		同步检测等。
	<process_name>-<SSH_USER>-<DATE>-<PID>-gc.log	垃圾回收日志。
	postinstallDetail.log	HDFS 服务安装后启动前工作日志。
	hdfs-service-check.log	HDFS 服务启动是否成功的检查日志。
	hdfs-set-storage-policy.log	HDFS 数据存储策略日志。
	cleanupDetail.log	HDFS 服务卸载时候的清理日志。
	prestartDetail.log	HDFS 服务启动前集群操作的记录日志。
	hdfs-recover-fsimage.log	NameNode 元数据恢复日志。
	datanode-disk-check.log	集群安装过程和使用过程中磁盘状态检测的记录日志。
	hdfs-availability-check.log	HDFS 服务是否可用日志。
	hdfs-backup-fsimage.log	NameNode 元数据备份日志。
	startDetail.log	hdfs 服务启动的详细日志。
	hdfs-blockplacement.log	HDFS 块放置策略记录日志。
	upgradeDetail.log	升级日志。
	hdfs-clean-acls-java.log	HDFS 清除已删除角色的 ACL 信息的日志。
	hdfs-haCheck.log	NameNode 主备状态获取脚本运行日志。
	<process_name>-jvmpause.log	进程运行中，记录 JVM 停顿的日志。
	hadoop-<SSH_USER>-balancer-<hostname>.log	HDFS 自动均衡的运行日志。
	hadoop-<SSH_USER>-balancer-	HDFS 运行自动均衡的环

日志类型	日志文件名	描述
	<hostname>.out	境信息日志。
	hdfs-switch-namenode.log	HDFS 主备倒换运行日志
	hdfs-router-admin.log	管理挂载表操作的运行日志
Tomcat 日志	hadoop-omm-host1.out，httpfs-catalina.<DATE>.log，httpfs-host-manager.<DATE>.log，httpfs-localhost.<DATE>.log，httpfs-manager.<DATE>.log，localhost_access_web_log.log	tomcat 运行日志
审计日志	hdfs-audit-<process_name>.log ranger-plugin-audit.log	HDFS 操作审计日志（例如：文件增删改查）。
	SecurityAuth.audit	HDFS 安全审计日志。

日志级别

HDFS 中提供了如表 9-34 所示的日志级别，日志级别优先级从高到低分别是 FATAL、ERROR、WARN、INFO、DEBUG，程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。

表9-34 日志级别

级别	描述
FATAL	FATAL 表示系统运行的致命错误信息。
ERROR	ERROR 表示系统运行的错误信息。
WARN	WARN 表示当前事件处理存在异常信息。
INFO	INFO 表示系统及各事件正常运行状态信息。
DEBUG	DEBUG 表示系统及系统调试信息。

如果您需要修改日志级别，请执行如下操作：

步骤 1 请参考修改集群服务配置参数，进入 HDFS 的“全部配置”页面。

步骤 2 左边菜单栏中选择所需修改的角色所对应的日志菜单。

步骤 3 选择所需修改的日志级别。

步骤 4 保存配置，在弹出窗口中单击“确定”使配置生效。

说明

配置完成后立即生效，不需要重启服务。

----结束

日志格式

HDFS 的日志格式如下所示：

表9-35 日志格式

日志类型	格式	示例
运行日志	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<产生该日志的线程名字>|<log 中的 message>|<日志事件的发生位置>	2015-01-26 18:43:42,840 | INFO | IPC Server handler 40 on 8020 | Rolling edit logs | org.apache.hadoop.hdfs.server.namenode.FSEditLog.rollEditLog(FSEditLog.java:1096)
审计日志	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<产生该日志的线程名字>|<log 中的 message>|<日志事件的发生位置>	2015-01-26 18:44:42,607 | INFO | IPC Server handler 32 on 8020 | allowed=true ugi=hbase (auth:SIMPLE) ip=/10.177.112.145 cmd=getfileinfo src=/hbase/WALs/hghoulaslx410,16020,1421743096083/hghoulaslx410%2C16020%2C1421743096083.1422268722795 dst=null perm=null | org.apache.hadoop.hdfs.server.namenode.FSNamesystem$DefaultAuditLogger.logAuditMessage(FSNamesystem.java:7950)

9.33 HDFS 性能调优

9.33.1 提升写性能

操作场景

在 HDFS 中，通过调整属性的值，使得 HDFS 集群更适应自身的业务情况，从而提升 HDFS 的写性能。

说明

本章节适用于 MRS 3.x 及后续版本。

操作步骤

参数入口：

在 FusionInsight Manager 系统中，选择“集群 > *待操作集群的名称* > 服务 > HDFS > 配置”，选择“全部配置”。在搜索框中输入参数名称。

表9-36 HDFS 写性能优化配置

参数	描述	默认值
dfs.datanode.drop.cache.behind.reads	表示是否让 DataNode 将在缓冲区中的数据传递给客户端后自动清除缓冲区中的所有数据。 • true：表示丢弃缓存的数据（需要在 DataNode 中配置）。 当同一份数据，重复读取的次数较少时，建议设置为 true，使得缓存能够被其他操作使用。 • false：重复读取的次数较多时，设置为 false 能够提升重复读取的速度。 说明 在提升写性能操作中，该参数为可选参数，请根据实际需要进行修改。	false
dfs.client-write-packet-size	客户端写包的大小。当 HDFS Client 往 DataNode 写数据时，将数据生成一个包。然后将这个包在网络上传出。此参数指定传输数据包的大小，可以通过各 Job 来指定。单位：字节。 在万兆网部署下，可适当增大该参数值，来提升传输的吞吐量。	262144

9.33.2 使用客户端元数据缓存提高读取性能

操作场景

通过使用客户端缓存元数据块的位置来提高 HDFS 读取性能。

说明

此功能仅用于读取不经常修改的文件。因为在服务器端由某些其他客户端完成的数据修改，对于高速缓存的客户端将是不可见的，这可能导致从缓存中拿到的元数据是过期的。

本章节适用于 MRS 3.x 及后续版本。

操作步骤

设置参数的路径：

在 FusionInsight Manager 页面中，选择“集群 > *待操作集群的名称* > 服务 > HDFS > 配置”，选择“全部配置”，并在搜索框中输入参数名称。

表9-37 参数配置

参数	描述	默认值
dfs.client.metadata.cache.enabled	启用/禁用块位置元数据的客户端缓存。将此参数设置为“true”，搭配“dfs.client.metadata.cache.pattern”参数以启用缓存。	false
dfs.client.metadata.cache.pattern	需要缓存的文件路径的正则表达式模式。只有这些文件的块位置元数据被缓存，直到这些元数据过期。此配置仅在参数“dfs.client.metadata.cache.enabled”设置为“true”时有效。 示例：“/test.*”表示读取其路径是以“/test”开头的所有文件。 说明 • 为确保一致性，配置特定模式以仅缓存其他客户端不经常修改的文件。 • 正则表达式模式将仅验证 URI 的 path 部分，而不验证在 Fully Qualified 路径情况下的 schema 和 authority。	-
dfs.client.metadata.cache.expiry.sec	缓存元数据的持续时间。缓存条目在该持续时间过期后失效。即使在缓存过程中经常使用的元数据也会发生失效。 配置值可采用时间后缀 s/m/h 表示，分别表示秒，分钟和小时。 说明 若将该参数配置为“0s”，将禁用缓存功能。	60s
dfs.client.metadata.cache.max.entries	缓存一次最多可保存的非过期数据条目。	65536

说明

要在过期前完全清除客户端缓存，可调用 *DFSClient#clearLocatedBlockCache()*。

用法如下所示。

```
FileSystem fs = FileSystem.get(conf);
DistributedFileSystem dfs = (DistributedFileSystem) fs;
DFSClient dfsClient = dfs.getClient();
dfsClient.clearLocatedBlockCache();
```

9.33.3 使用当前活动缓存提升客户端与 NameNode 的连接性能

操作场景

HDFS 部署在具有多个 NameNode 实例的 HA（High Availability）模式中，HDFS 客户端需要依次连接到每个 NameNode，以确定当前活动的 NameNode 是什么，并将其用于客户端操作。

一旦识别出来，当前活动的 NameNode 的详细信息就可以被缓存并共享给在客户端机器中运行的所有客户端。这样，每个新客户端可以首先尝试从缓存加载活动的 NameNode 的详细信息，并将 RPC 调用保存到备用的 NameNode。在异常情况下有很多优势，例如当备用的 NameNode 连接长时间不响应时。

当发生故障，将另一个 NameNode 切换为活动状态时，缓存的详细信息将被更新为当前活动的 NameNode 的信息。

说明

本章节适用于 MRS 3.x 及后续版本。

操作步骤

设置参数的路径如下：

在 FusionInsight Manager 页面中，选择“集群 > *待操作集群的名称* > 服务 > HDFS > 配置”，选择“全部配置”，并在搜索框中输入参数名称。

表9-38 配置参数

参数	描述	默认值
dfs.client.failover.proxy.provider.[nameservice ID]	用已通过的协议创建 namenode 代理的 Client Failover proxy provider 类。配置成 org.apache.hadoop.hdfs.server.namenode.ha.BlackListingFailoverProxyProvider，可在 HDFS 客户端使用 NameNode 黑名单特性。配置成 org.apache.hadoop.hdfs.server.namenode.ha.ObserverReadProxyProvider，可使用从 NameNode 支持读的特性。	org.apache.hadoop.hdfs.server.namenode.ha.AdaptiveFailoverProxyProvider
dfs.client.failover.activeinfo.share.flag	启用缓存并将当前活动的 NameNode 的详细信息共享给其他客户端。若要启用缓存，需将其设置为“true”。	false
dfs.client.failover.activeinfo.share.path	指定将在机器中的所有客户端创建的共享文件的本地目录。如果要为不同用户共享	/tmp

参数	描述	默认值
	缓存，该文件夹应具有必需的权限（如在给定目录中创建，读写缓存文件）。	
dfs.client.failover.activeinfo.share.io.timeout.sec	控制超时的可选配置。用于在读取或写入缓存文件时获取锁定。如果在该时间内无法获取缓存文件上的锁定，则放弃尝试读取或更新缓存。单位为秒。	5

说明

由 HDFS 客户端创建的缓存文件必须由其他客户端重新使用。因此，这些文件永远不会从本地系统中删除。若禁用该功能，可能需要进行手动清理。

9.34 HDFS 常见问题

9.34.1 NameNode 启动慢

问题

删除大量文件之后立刻重启 NameNode（例如删除 100 万个文件），NameNode 启动慢。

回答

由于在删除了大量文件之后，DataNode 需要时间去删除对应的 Block。当立刻重启 NameNode 时，NameNode 会去检查所有 DataNode 上报的 Block 信息，发现已删除的 Block 时，会输出对应的 INFO 日志信息，如下所示：

```
2015-06-10 19:25:50,215 | INFO  | IPC Server handler 36 on 25000 | BLOCK*
processReport:
blk_1075861877_2121067 on node 10.91.8.218:9866 size 10249 does not belong to any
file |
org.apache.hadoop.hdfs.server.blockmanagement.BlockManager.processReport(BlockManag
er.java:1854)
```

每一个被删除的 Block 会产生一条日志信息，一个文件可能会存在一个或多个 Block。当删除的文件数过多时，NameNode 会花大量的时间打印日志，然后导致 NameNode 启动慢。

当出现这种现象时，您可以通过如下方式提升 NameNode 的启动速度。

1. 删除大量文件时，不要立刻重启 NameNode，待 DataNode 删除了对应的 Block 后重启 NameNode，即不会存在这种情况。

   您可以通过 ***hdfs dfsadmin -report*** 命令来查看磁盘空间，检查文件是否删除完毕。
2. 如已大量出现以上日志，您可以将 NameNode 的日志级别修改为 ERROR，NameNode 不会再打印此日志信息。

等待 NameNode 启动完毕后，再将此日志级别修改为 INFO。修改日志级别后无需重启服务。

9.34.2 DataNode 状态正常，但无法正常上报数据块

问题

DataNode 正常，但无法正常上报数据块，导致存在的数据块无法使用。

回答

当某个数据目录中的数据块数量超过 4 倍的数据块限定值(1M)时，可能会出现该错误。DataNode 会产生相应的错误日志记录，如下所示：

```
2015-11-05 10:26:32,936 | ERROR |
DataNode:[[[DISK]file:/srv/BigData/hadoop/data1/dn/]] heartbeating to
vm-210/10.91.8.210:8020 | Exception in BPOfferService for Block pool BP-805114975-
10.91.8.210-1446519981645
(Datanode Uuid bcada350-0231-413b-bac0-8c65e906c1bb) service to vm-
210/10.91.8.210:8020 | BPServiceActor.java:824
java.lang.IllegalStateException:com.google.protobuf.InvalidProtocolBufferException:
Protocol message was too large.May
be malicious.Use CodedInputStream.setSizeLimit() to increase the size limit. at
org.apache.hadoop.hdfs.protocol.BlockListAsLongs$BufferDecoder$1.next(BlockListAsLo
ngs.java:369)
at
org.apache.hadoop.hdfs.protocol.BlockListAsLongs$BufferDecoder$1.next(BlockListAsLo
ngs.java:347) at org.apache.hadoop.hdfs.
protocol.BlockListAsLongs$BufferDecoder.getBlockListAsLongs(BlockListAsLongs.java:3
25) at org.apache.hadoop.hdfs.protocolPB.DatanodeProtocolClientSideTranslatorPB.
blockReport(DatanodeProtocolClientSideTranslatorPB.java:190) at
org.apache.hadoop.hdfs.server.datanode.BPServiceActor.blockReport(BPServiceActor.ja
va:473)
at
org.apache.hadoop.hdfs.server.datanode.BPServiceActor.offerService(BPServiceActor.j
ava:685) at
org.apache.hadoop.hdfs.server.datanode.BPServiceActor.run(BPServiceActor.java:822)
at java.lang.Thread.run(Thread.java:745) Caused
by:com.google.protobuf.InvalidProtocolBufferException:Protocol message was too
large.May be malicious.Use CodedInputStream.setSizeLimit()
to increase the size limit. at
com.google.protobuf.InvalidProtocolBufferException.sizeLimitExceeded(InvalidProtoco
lBufferException.java:110) at
com.google.protobuf.CodedInputStream.refillBuffer(CodedInputStream.java:755)
at com.google.protobuf.CodedInputStream.readRawByte(CodedInputStream.java:769) at
com.google.protobuf.CodedInputStream.readRawVarint64(CodedInputStream.java:462) at
com.google.protobuf.
CodedInputStream.readSInt64(CodedInputStream.java:363) at
org.apache.hadoop.hdfs.protocol.BlockListAsLongs$BufferDecoder$1.next(BlockListAsLo
ngs.java:363)
```

如今，数据目录中数据块的数量会显示为 Metric。用户可以通过以下 URL 对该值进行监视 http://<datanode-ip>:<http-port>/jmx，如果该值超过 4 倍的限定值(4*1M)，建议用户配置多个驱动器并重新启动 HDFS。

恢复步骤：

1. 在 DataNode 上配置多个数据目录。

   示例：在原先只配置了/data1/datadir 的位置

   ```
   <property> <name>dfs.datanode.data.dir</name> <value>/data1/datadir</value>
   </property>
   ```

   按照如下内容进行配置。

   ```
   <property> <name>dfs.datanode.data.dir</name>
   <value>/data1/datadir/,/data2/datadir,/data3/datadir</value> </property>
   ```

   说明

   建议多个数据目录应该配置到多个磁盘中，否则所有的数据都将写入同一个磁盘，对性能有很大的影响。

2. 重新启动 HDFS。
3. 按照如下方法将数据移动至新的数据目录。

   mv */data1/datadir/current/finalized/subdir1　/data2/datadir/current/finalized/subdir1*

4. 重新启动 HDFS。

9.34.3 HDFS Web UI 无法正常刷新损坏数据的信息

问题

1. 当 DataNode 的“dfs.datanode.data.dir”所配置的目录因权限或者磁盘损坏发生错误时，HDFS Web UI 没有显示损坏数据的信息。
2. 当此错误被修复后，HDFS Web UI 没有及时移除损坏数据的相关信息。

回答

1. DataNode 只有在执行文件操作发生错误时，才会去检查磁盘是否正常，若发现数据损坏，则将此错误上报至 NameNode，此时 NameNode 才会在 HDFS Web UI 显示数据损坏信息。
2. 当错误修复后，需要重启 DataNode。当重启 DataNode 时，会检查所有数据状态并上传损坏数据信息至 NameNode。所以当此错误被修复后，只有重启 DataNode 后，才会不显示损坏数据信息。

9.34.4 distcp 命令在安全集群上失败并抛出异常

问题

为何 distcp 命令在安全集群上失败并抛出异常？

客户端出现异常：

```
Invalid arguments:Unexpected end of file from server
```

服务器端出现异常：

```
javax.net.ssl.SSLException:Unrecognized SSL message, plaintext connection?
```

回答

当用户在 distcp 命令中使用 **webhdfs://**时，会抛出上述异常，是由于集群所使用的 HTTP 政策为 HTTPS，即配置在“core-site.xml”的“dfs.http.policy”值为“HTTPS_ONLY”。所以要避免出现此异常，应使用 **swebhdfs://**替代 **webhdfs://**。

例如：

./hadoop distcp swebhdfs://IP:PORT/testfile hdfs://IP:PORT/testfile1

9.34.5 当 dfs.datanode.data.dir 中定义的磁盘数量等于 dfs.datanode.failed.volumes.tolerated 的值时，DataNode 启动失败

问题

当“dfs.datanode.data.dir”中定义的磁盘数量等于“dfs.datanode.failed.volumes.tolerated”的值时，DataNode 启动失败。

回答

默认情况下，单个磁盘的故障将会引起 HDFS DataNode 进程关闭，导致 NameNode 为每一个存在 DataNode 上的 block 调度额外的副本，在没有故障的磁盘中引起不必要的块复制。

为了防止此情况，用户可以通过配置 DataNodes 来承受 dfs.data.dir 目录的故障。在“hdfs-site.xml”中配置参数“dfs.datanode.failed.volumes.tolerated”。例如：如果该参数值为 3，DataNode 只有在 4 个或者更多个目录故障之后才会出现故障。该值会影响到 DataNode 的启动。

如果想要 DataNode 不出现故障，配置的“dfs.datanode.failed.volumes.tolerated”一定要小于所配置的卷数，也可以将“dfs.datanode.failed.volumes.tolerated”设置成-1，相当于设置该值为 n-1（n 为卷数），那样 DataNode 就不会出现启动失败。

9.34.6 当多个 data.dir 被配置在一个磁盘分区内，DataNode 的容量计算将会出错

问题

当多个 data.dir 被配置在一个磁盘分区内，DataNode 的容量计算将会出错。

回答

目前容量计算是基于磁盘的，类似于 Linux 里面的 ***df*** 命令。理想状态下，用户不会在同一个磁盘内配置多个 data.dir，否则所有的数据都将写入一个磁盘，在性能上会有很大的影响。

因此最好配置如下：

例如，如果机器有如下磁盘：

```
host-4:~ # df -h
Filesystem     Size    Used    Avail    Use%   Mounted on
/dev/sda1      352G   11G     324G    4%     /
udev            190G    252K   190G    1%     /dev
tmpfs           190G   72K     190G    1%     /dev/shm
/dev/sdb1      2.7T    74G     2.5T     3%    /data1
/dev/sdc1      2.7T    75G     2.5T     3%    /data2
/dev/sdd1      2.7T    73G     2.5T     3%    /da
```

建议的配置方式：

```
<property>
<name>dfs.datanode.data.dir</name>
<value>/data1/datadir/,/data2/datadir,/data3/datadir</value>
</property>
```

不建议的配置方式：

```
<property>
<name>dfs.datanode.data.dir</name>
<value>/data1/datadir1/,/data2/datadir1,/data3/datadir1,/data1/datadir2,data1/datadir3,/data2/datadir2,/data2/datadir3,/data3/datadir2,/data3/datadir3</value>
</property>
```

9.34.7 当 Standby NameNode 存储元数据（命名空间）时，出现断电的情况，Standby NameNode 启动失败

问题

当 Standby NameNode 存储元数据（命名空间）时，出现断电的情况，Standby NameNode 启动失败并抛出如下错误信息。

```
2015-12-04 11:49:12,121 | ERROR | main | Failed to load image from FS
ImageFile(file=/srv/BigData/namenode/current/fsimage_0000000000000096
080,
cpktTxId=0000000000000096080) | FSImage.java:685
java.io.IOException: Invalid MD5 file /srv/BigData/namenode/current/f
simage_0000000000000096080.md5:
the content "锟斤拷锟斤拷锟斤拷锟斤拷锟斤拷锟斤拷[1m^A!锟 does not match the expecte
d pattern.
at org.apache.hadoop.hdfs.util.MD5FileUtils.readStoredMd5(MD5FileUtil
s.java:92)
at org.apache.hadoop.hdfs.util.MD5FileUtils.readStoredMd5ForFile(MD5F
ileUtils.java:109)
at org.apache.hadoop.hdfs.server.namenode.FSImage.loadFSImage(FSImage
.java:975)
at org.apache.hadoop.hdfs.server.namenode.FSImage.loadFSImageFile(FSI
mage.java:744)
at org.apache.hadoop.hdfs.server.namenode.FSImage.loadFSImage(FSImage
.java:682)
at org.apache.hadoop.hdfs.server.namenode.FSImage.recoverTransitionRe
ad(FSImage.java:300)
at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.loadFSImage(FS
Namesystem.java:968)
at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.loadFromDisk(F
SNamesystem.java:675)
at org.apache.hadoop.hdfs.server.namenode.NameNode.loadNamesystem(Nam
eNode.java:625)
at org.apache.hadoop.hdfs.server.namenode.NameNode.initialize(NameNod
e.java:685)
at org.apache.hadoop.hdfs.server.namenode.NameNode.<init>(NameNode.ja
va:889)
at org.apache.hadoop.hdfs.server.namenode.NameNode.<init>(NameNode.ja
va:872)
at org.apache.hadoop.hdfs.server.namenode.NameNode.createNameNode(Nam
eNode.java:1580)
at org.apache.hadoop.hdfs.server.namenode.NameNode.main(NameNode.java
:1654)
```

回答

当 Standby NameNode 存储元数据（命名空间）时，出现断电的情况，Standby NameNode 启动失败，MD5 文件会损坏。通过移除损坏的 fsimage，然后启动 Standby NameNode，可以修复此问题。Standby NameNode 会加载先前的 fsimage 并重现所有的 edits。

修复步骤：

1. 移除损坏的 fsimage。

   rm -rf ${BIGDATA_DATA_HOME}/namenode/current/fsimage_0000000000000096

2. 启动 Standby NameNode。

9.34.8 在存储小文件过程中，系统断电，缓存中的数据丢失

问题

在存储小文件过程中，系统断电，缓存中的数据丢失。

回答

由于断电，当写操作完成之后，缓存中的 block 不会立即被写入磁盘，如果要同步地将缓存的 block 写入磁盘，用户需要将“hdfs-site.xml”中的“dfs.datanode.synconclose”设置为“true”。

默认情况下，“dfs.datanode.synconclose”为“false”，虽然性能很高，但是断电之后，存储在缓存中的数据会丢失。将“dfs.datanode.synconclose”设置为“true”，可以解决此问题，但对性能有很大影响。请根据具体的应用场景决定是否开启该参数。

9.34.9 FileInputFormat split 的时候出现数组越界

问题

HDFS 调用 FileInputFormat 的 getSplit 方法的时候，出现 ArrayIndexOutOfBoundsException: 0，日志如下：

```
java.lang.ArrayIndexOutOfBoundsException: 0
at org.apache.hadoop.mapred.FileInputFormat.identifyHosts(FileInputFormat.java:708)
at
org.apache.hadoop.mapred.FileInputFormat.getSplitHostsAndCachedHosts(FileInputForma
t.java:675)
at org.apache.hadoop.mapred.FileInputFormat.getSplits(FileInputFormat.java:359)
at org.apache.spark.rdd.HadoopRDD.getPartitions(HadoopRDD.scala:210)
at org.apache.spark.rdd.RDD$$anonfun$partitions$2.apply(RDD.scala:239)
at org.apache.spark.rdd.RDD$$anonfun$partitions$2.apply(RDD.scala:237)
at scala.Option.getOrElse(Option.scala:120)
at org.apache.spark.rdd.RDD.partitions(RDD.scala:237)
at org.apache.spark.rdd.MapPartitionsRDD.getPartitions(MapPartitionsRDD.scala:35)
```

回答

每个 block 对应的机架信息组成为：/default/rack0/:,/default/rack0/datanodeip:port。

该问题是由于某个 block 块损坏或者丢失，导致该 block 对应的机器 ip 和 port 为空引起的，出现该问题的时候使用 **hdfs fsck** 检查对应文件块的健康状态，删除损坏或者恢复丢失的块，重新进行任务计算即可。

9.34.10 当分级存储策略为 LAZY_PERSIST 时，为什么文件的副本的存储类型都是 DISK

问题

当文件的存储策略为 LAZY_PERSIST 时，文件的第一副本的存储类型应为 RAM_DISK，其余副本为 DISK。

为什么文件的所有副本的存储类型都是 DISK？

回答

当用户写入存储策略为 LAZY_PERSIST 的文件时，文件的三个副本会逐一写入。第一副本会优先选择客户端所在的 DataNode 节点，在以下情况下，当文件的存储策略为 LAZY_PERSIST 时，文件的所有副本的存储类型都是 DISK：

- 当客户端所在的 DataNode 节点没有 RAM_DISK 时，则会写入客户端所在的 DataNode 节点的 DISK 磁盘，其余副本会写入其他节点的 DISK 磁盘。
- 当客户端所在的 DataNode 节点有 RAM_DISK，但"dfs.datanode.max.locked.memory"参数值未设置或设置过小（小于“dfs.blocksize”参数值）时，则会写入客户端所在的 DataNode 节点的 DISK 磁盘，其余副本会写入其他节点的 DISK 磁盘。

9.34.11 NameNode 节点长时间满负载，HDFS 客户端无响应

问题

当 NameNode 节点处于满负载、NameNode 所在节点的 CPU 100%耗尽时，导致 NameNode 无法响应，对于新连接到该 NameNode 的 HDFS 客户端，能够主备切换连接到另一个 NameNode，进行正常的操作，而对于已经连接到该 NameNode 节点的 HDFS 客户端可能会卡住，无法进行下一步操作。

回答

目前出现上述问题时使用的是默认配置，如表 9-39 所示，HDFS 客户端到 NameNode 的 RPC 连接存在 keep alive 机制，保持连接不会超时，尽力等待服务器的响应，因此导致已经连接的 HDFS 客户端的操作会卡住。

对于已经卡住的 HDFS 客户端，可以进行如下操作：

- 等待 NameNode 响应，一旦 NameNode 所在节点的 CPU 利用率回落，NameNode 可以重新获得 CPU 资源时，HDFS 客户端即可得到响应。
- 如果无法等待更长时间，需要重启 HDFS 客户端所在的应用程序进程，使得 HDFS 客户端重新连接空闲的 NameNode。

解决措施：

为了避免该问题出现，可以在客户端的配置文件“core-site.xml”中做如下配置。

表9-39 参数说明

参数	描述	默认值
ipc.client.ping	当配置为 true 时，客户端会尽力等待服务端响应，定期发送 ping 消息，使得连接不会因为 tcp timeout 而断开。 当配置为 false 时，客户端会使用配置项“ipc.ping.interval”对应的值，作为 timeout 时间，在该时间内没有得到响应，即会超时。 在上述问题场景下，建议配置为 false。	true

参数	描述	默认值
ipc.ping.interval	当“ipc.client.ping”配置为 true 时，表示发送 ping 消息的周期。 当“ipc.client.ping”设置为 false 时，表示连接的超时时间。 在上述问题场景下，建议配置一个较大的超时时间，避免服务繁忙时的超时，建议配置为 900000，单位为 ms。	60000

9.34.12 DataNode 禁止手动删除或修改数据存储目录

问题

- 数据块在 DataNode 上的存储目录由“dfs.datanode.data.dir”配置项指定，是否可以修改该配置项来修改数据存储目录？
- 是否可以手动拷贝数据存储目录下的文件？

回答

“dfs.datanode.data.dir”配置项用于指定数据块在 DataNode 上的存储目录，在系统安装时需要指定根目录，并且可以指定多个根目录。

- 请谨慎修改该配置项，可以添加新的数据根目录。
- 禁止删除原有存储目录，否则会造成数据块丢失，导致文件无法正常读写。
- 禁止手动删除或修改存储目录下的数据块，否则可能会造成数据块丢失。

说明

NameNode 和 JournalNode 存在类似的配置项，也同样禁止删除原有存储目录，禁止手动删除或修改存储目录下的数据块。

- dfs.namenode.edits.dir
- dfs.namenode.name.dir
- dfs.journalnode.edits.dir

9.34.13 成功回滚后，为什么 NameNode UI 上显示有一些块缺失

问题

回滚成功后，为什么 NameNode UI 上显示有一些块缺失？

回答

原因：具有新 id/genstamps 的块可能存在于 DataNode 上。DataNode 中的块文件可能具有与 NameNode 的回滚 image 中不同的生成标记和长度，所以 NameNode 会拒绝 DataNode 中的这些块，并将文件标记为已损坏。

场景如下：

1. 升级前

   客户端 A - >将一些数据写入文件 X（假设已写入“A”字节）

2. 升级开始了

   客户端 A - >仍然将数据写入文件 X（现在文件中的数据是“A + B”字节）

3. 升级完成

   客户端 A - >完成写入文件。最终数据为“A + B”字节。

4. 回滚开始

   将回滚到步骤 1（升级前）的状态。因此，NameNode 中的文件 X 将具有“A”字节，但 DataNode 中的块文件将具有“A + B”字节。

恢复步骤：

1. 从 NameNode Web UI 中获取已损坏的文件列表，或者通过下面的命令获取。

   hdfs fsck <filepath> -list-corruptfileblocks

2. 对于不需要的文件，请使用以下命令删除文件。

   hdfs fsck <corrupt file path> - delete

   说明

   删除文件为高危操作，在执行操作前请务必确认对应文件是否不再需要。

3. 对于所需的文件，执行 fsck 命令来获取块列表和块的顺序。
   - 在 fsck 中给出的块序列表中，使用块 id 搜索 DataNode 中的数据目录，并从 DataNode 下载相应的块。
   - 按照序列以追加的方式写入所有这样的块文件，并构造成原始文件。

     例如：

     File 1--> blk_1, blk_2, blk_3

     通过组合来自同一序列的所有三个块文件的内容来创建文件。
   - 从 HDFS 中删除旧文件并重写新构建的文件。

9.34.14 为什么在往 HDFS 写数据时报"java.net.SocketException: No buffer space available"异常

问题

为什么在往 HDFS 写数据时报"java.net.SocketException: No buffer space available"异常？

这个问题发生在往 HDFS 写文件时。查看客户端和 DataNode 的错误日志。

客户端日志如下：

图9-8 客户端日志

DataNode 日志如下：

```
2017-07-24 20:43:39,269 | ERROR | DataXceiver for client
DFSClient NONMAPREDUCE 996005058 86
 at /192.168.164.155:40214 [Receiving block BP-1287143557-192.168.199.6-
1500707719940:blk_1074269754_528941 with io weight 10] |
DataNode{data=FSDataset{dirpath='[/srv/BigData/hadoop/data1/dn/current,
/srv/BigData/hadoop/data2/dn/current, /srv/BigData/hadoop/data3/dn/current,
/srv/BigData/hadoop/data4/dn/current, /srv/BigData/hadoop/data5/dn/current,
/srv/BigData/hadoop/data6/dn/current, /srv/BigData/hadoop/data7/dn/current]'},
localName='192-168-164-155:9866', datanodeUuid='a013e29c-4e72-400c-bc7b-
bbbf0799604c', xmitsInProgress=0}:Exception transfering block BP-1287143557-
192.168.199.6-1500707719940:blk_1074269754_528941 to mirror 192.168.202.99:9866:
java.net.SocketException: No buffer space available | DataXceiver.java:870
2017-07-24 20:43:39,269 | INFO | DataXceiver for client
DFSClient_NONMAPREDUCE_996005058_86
 at /192.168.164.155:40214 [Receiving block BP-1287143557-192.168.199.6-
1500707719940:blk_1074269754_528941 with io weight 10] | opWriteBlock BP-
1287143557-192.168.199.6-1500707719940:blk_1074269754_528941 received exception
java.net.SocketException: No buffer space available | DataXceiver.java:933
2017-07-24 20:43:39,270 | ERROR | DataXceiver for client
DFSClient_NONMAPREDUCE_996005058_86
 at /192.168.164.155:40214 [Receiving block BP-1287143557-192.168.199.6-
1500707719940:blk_1074269754_528941 with io weight 10] | 192-168-164-
155:9866:DataXceiver error processing WRITE_BLOCK operation src:
/192.168.164.155:40214 dst: /192.168.164.155:9866 | DataXceiver.java:304
java.net.SocketException: No buffer space available
 at sun.nio.ch.Net.connect0(Native Method)
 at sun.nio.ch.Net.connect(Net.java:454)
 at sun.nio.ch.Net.connect(Net.java:446)
 at sun.nio.ch.SocketChannelImpl.connect(SocketChannelImpl.java:648)
 at org.apache.hadoop.net.SocketIOWithTimeout.connect(SocketIOWithTimeout.java:192)
 at org.apache.hadoop.net.NetUtils.connect(NetUtils.java:531)
 at org.apache.hadoop.net.NetUtils.connect(NetUtils.java:495)
 at
```

```
org.apache.hadoop.hdfs.server.datanode.DataXceiver.writeBlock(DataXceiver.java:800)
 at
org.apache.hadoop.hdfs.protocol.datatransfer.Receiver.opWriteBlock(Receiver.java:13
8)
 at
org.apache.hadoop.hdfs.protocol.datatransfer.Receiver.processOp(Receiver.java:74)
 at org.apache.hadoop.hdfs.server.datanode.DataXceiver.run(DataXceiver.java:265)
 at java.lang.Thread.run(Thread.java:748)
```

回答

上述问题可能是因为网络内存枯竭而导致的。

问题的解决方案是根据实际场景适当增大网络设备的阈值级别。

例如：

```
[root@xxxxx ~]# cat /proc/sys/net/ipv4/neigh/default/gc_thresh*
128
512
1024
[root@xxxxx ~]# echo 512 > /proc/sys/net/ipv4/neigh/default/gc_thresh1
[root@xxxxx ~]# echo 2048 > /proc/sys/net/ipv4/neigh/default/gc_thresh2
[root@xxxxx ~]# echo 4096 > /proc/sys/net/ipv4/neigh/default/gc_thresh3
[root@xxxxx ~]# cat /proc/sys/net/ipv4/neigh/default/gc_thresh*
512
2048
4096
```

还可以将以下参数添加到“/etc/sysctl.conf”中，即使主机重启，配置依然能生效。

```
net.ipv4.neigh.default.gc_thresh1 = 512
net.ipv4.neigh.default.gc_thresh2 = 2048
net.ipv4.neigh.default.gc_thresh3 = 4096
```

9.34.15 为什么主 NameNode 重启后系统出现双备现象

问题

为什么主 NameNode 重启后系统出现双备现象？

出现该问题时，查看 Zookeeper 和 ZKFC 的日志，发现 Zookeeper 服务端与客户端（ZKFC）通信时所使用的 session 不一致，Zookeeper 服务端的 sessionId 为 0x164cb2b3e4b36ae4，ZKFC 的 sessionId 为 0x144cb2b3e4b36ae4。这意味着 Zookeeper 服务端与客户端（ZKFC）之间数据交互失败。

Zookeeper 日志，如下所示：

```
2015-04-15 21:24:54,257 | INFO | CommitProcessor:22 | Established session
0x164cb2b3e4b36ae4 with negotiated timeout 45000 for client /192.168.0.117:44586 |
org.apache.zookeeper.server.ZooKeeperServer.finishSessionInit(ZooKeeperServer.java:
623)
2015-04-15 21:24:54,261 | INFO | NIOServerCxn.Factory:192-168-0-
114/192.168.0.114:2181 | Successfully authenticated client:
authenticationID=hdfs/hadoop@<系统域名>; authorizationID=hdfs/hadoop@<系统域名>. |
org.apache.zookeeper.server.auth.SaslServerCallbackHandler.handleAuthorizeCallback(
```

```
SaslServerCallbackHandler.java:118)
2015-04-15 21:24:54,261 | INFO | NIOServerCxn.Factory:192-168-0-
114/192.168.0.114:2181 | Setting authorizedID: hdfs/hadoop@<系统域名> |
org.apache.zookeeper.server.auth.SaslServerCallbackHandler.handleAuthorizeCallback(
SaslServerCallbackHandler.java:134)
2015-04-15 21:24:54,261 | INFO | NIOServerCxn.Factory:192-168-0-
114/192.168.0.114:2181 | adding SASL authorization for authorizationID:
hdfs/hadoop@<系统域名> |
org.apache.zookeeper.server.ZooKeeperServer.processSasl(ZooKeeperServer.java:1009)
2015-04-15 21:24:54,262 | INFO | ProcessThread(sid:22 cport:-1): | Got user-level
KeeperException when processing sessionid:0x164cb2b3e4b36ae4 type:create cxid:0x3
zxid:0x20009fafc txntype:-1 reqpath:n/a Error Path:/hadoop-
ha/hacluster/ActiveStandbyElectorLock Error:KeeperErrorCode = NodeExists for
/hadoop-ha/hacluster/ActiveStandbyElectorLock |
org.apache.zookeeper.server.PrepRequestProcessor.pRequest(PrepRequestProcessor.java
:648)
```

ZKFC 日志，如下所示：

```
2015-04-15 21:24:54,237 | INFO | main-SendThread(192-168-0-114:2181) | Socket
connection established to 192-168-0-114/192.168.0.114:2181, initiating session |
org.apache.zookeeper.ClientCnxn$SendThread.primeConnection(ClientCnxn.java:854)
2015-04-15 21:24:54,257 | INFO | main-SendThread(192-168-0-114:2181) | Session
establishment complete on server 192-168-0-114/192.168.0.114:2181, sessionid =
0x144cb2b3e4b36ae4 , negotiated timeout = 45000 |
org.apache.zookeeper.ClientCnxn$SendThread.onConnected(ClientCnxn.java:1259)
2015-04-15 21:24:54,260 | INFO | main-EventThread | EventThread shut down |
org.apache.zookeeper.ClientCnxn$EventThread.run(ClientCnxn.java:512)
2015-04-15 21:24:54,262 | INFO | main-EventThread | Session connected. |
org.apache.hadoop.ha.ActiveStandbyElector.processWatchEvent(ActiveStandbyElector.ja
va:547)
2015-04-15 21:24:54,264 | INFO | main-EventThread | Successfully authenticated to
ZooKeeper using SASL. |
org.apache.hadoop.ha.ActiveStandbyElector.processWatchEvent(ActiveStandbyElector.ja
va:573)
```

回答

- 原因分析

  NameNode 的主节点重启后，它原先在 Zookeeper 上建立的临时节点（/hadoop-ha/hacluster/ActiveStandbyElectorLock）就会被清理。同时，NameNode 备节点发现这个信息后进行抢占希望升主，所以它重新在 Zookeeper 上建立了 active 的节点/hadoop-ha/hacluster/ActiveStandbyElectorLock。但是 NameNode 备节点通过客户端（ZKFC）与 Zookeeper 建立连接时，由于网络问题、CPU 使用率高、集群压力大等原因，出现了客户端（ZKFC）的 session（0x144cb2b3e4b36ae4）与 Zookeeper 服务端的 session（0x164cb2b3e4b36ae4）不一致的问题，这就导致了 NameNode 备节点的 watcher 没有感知到自己已经成功建立临时节点，依然认为自己还是备。而 NameNode 主节点启动后，发现/hadoop-ha/hacluster 目录下已经有 active 的节点，所以也无法升主，导致两个节点都为备。

- 解决方法

  建议通过在 FusionInsight Manager 界面上重启 HDFS 的两个 ZKFC 加以解决。

9.34.16 HDFS 执行 Balance 时被异常停止，再次执行 Balance 会失败

问题

在 HDFS 客户端启动一个 Balance 进程，该进程被异常停止后，再次执行 Balance 操作，操作会失败。

回答

通常，HDFS 执行 Balance 操作结束后，会自动释放“/system/balancer.id”文件，可再次正常执行 Balance。

但在上述场景中，由于第一次的 Balance 操作是被异常停止的，所以第二次进行 Balance 操作时，“/system/balancer.id”文件仍然存在，则会触发 ***append /system/balancer.id*** 操作，进而导致 Balance 操作失败。

- 如果“/system/balancer.id”文件的释放时间超过了软租期 60s，则第二次执行 Balance 操作的客户端的 append 操作会抢占租约，此时最后一个 block 处于 under construction 或者 under recovery 状态，会触发 block 的恢复操作，那么“/system/balancer.id”文件必须等待 block 恢复完成才能关闭，所以此次 append 操作失败。

  append /system/balancer.id 操作失败后，会向客户端抛出 RecoveryInProgressException 异常：

  ```
  org.apache.hadoop.ipc.RemoteException(org.apache.hadoop.protocol.RecoveryInProgressException): Failed to APPEND_FILE /system/balancer.id for DFSClient because lease recovery is in progress. Try again later.
  ```

- 如果该文件的释放时间没有超过默认设置 60s，原有客户端会继续持有该租约，则会抛出 AlreadyBeingCreatedException 异常，实际上向客户端返回的是 null，导致客户端出现如下异常：

  ```
  java.io.IOException: Cannot create any NameNode Connectors.. Exiting...
  ```

可通过以下方法避免上述问题：

- 方案 1：等待硬租期超过 1 小时后，原有客户端释放租约，再执行第二次 Balance 操作。
- 方案 2：执行第二次 Balance 操作之前删除“/system/balancer.id”文件。

9.34.17 IE 浏览器访问 HDFS 原生 UI 界面失败，显示无法显示此页

问题

通过 IE 9、IE 10 和 IE 11 浏览器访问 HDFS 的原生 UI 界面，偶尔出现访问失败情况。

现象

访问页面失败，浏览器无法显示此页，如下图所示：

原因

IE 9、IE 10、IE 11 浏览器的某些版本在处理 SSL 握手有问题导致访问失败。

解决方法

重新刷新页面即可。

9.34.18 EditLog 不连续导致 NameNode 启动失败

问题

在 JournalNode 节点有断电，数据目录磁盘占满，网络异常时，会导致 JournalNode 上的 EditLog 不连续。此时如果重启 NameNode，很可能会失败。

现象

重启 NameNode 会失败。在 NameNode 运行日志中会报如下的错误：

```
2019-11-08 16:30:28,399 | ERROR | main | Failed to start namenode. | NameNode.java:1732
java.io.IOException: There appears to be a gap in the edit log.  We expected txid 13698019, but got txid 13698088.
        at org.apache.hadoop.hdfs.server.namenode.MetaRecoveryContext.editLogLoaderPrompt(MetaRecoveryContext.java:94)
        at org.apache.hadoop.hdfs.server.namenode.FSEditLogLoader.loadEditRecords(FSEditLogLoader.java:278)
        at org.apache.hadoop.hdfs.server.namenode.FSEditLogLoader.loadFSEdits(FSEditLogLoader.java:188)
        at org.apache.hadoop.hdfs.server.namenode.FSImage.loadEdits(FSImage.java:924)
        at org.apache.hadoop.hdfs.server.namenode.FSImage.loadFSImage(FSImage.java:771)
        at org.apache.hadoop.hdfs.server.namenode.FSImage.recoverTransitionRead(FSImage.java:331)
        at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.loadFSImage(FSNamesystem.java:1108)
        at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.loadFromDisk(FSNamesystem.java:727)
        at org.apache.hadoop.hdfs.server.namenode.NameNode.loadNamesystem(NameNode.java:638)
        at org.apache.hadoop.hdfs.server.namenode.NameNode.initialize(NameNode.java:700)
        at org.apache.hadoop.hdfs.server.namenode.NameNode.<init>(NameNode.java:943)
        at org.apache.hadoop.hdfs.server.namenode.NameNode.<init>(NameNode.java:916)
        at org.apache.hadoop.hdfs.server.namenode.NameNode.createNameNode(NameNode.java:1655)
        at org.apache.hadoop.hdfs.server.namenode.NameNode.main(NameNode.java:1725)
```

解决方法

1. 找到重启前的主 NameNode，进入其数据目录（查看配置项“dfs.namenode.name.dir”可获取，例如/srv/BigData/namenode/current），得到最新的 FSImage 文件的序号。一般如下：

```
-rw-------. 1 omm wheel     574 Oct  2 01:12 edits_0000000000013259401-0000000000013
-rw-------. 1 omm wheel     575 Oct  2 01:13 edits_0000000000013259409-0000000000013
-rw-------. 1 omm wheel      42 Oct  2 01:13 edits_0000000000013259417-0000000000013
-rw-------. 1 omm wheel 1048576 Nov  8 16:01 edits_inprogress_0000000000013698088
-rw-------. 1 omm wheel  314803 Nov  8 15:53 fsimage_0000000000013698018
-rw-------. 1 omm wheel      62 Nov  8 15:53 fsimage_0000000000013698018.md5
-rw-------. 1 omm wheel  314803 Nov  8 15:56 fsimage_0000000000013698050
-rw-------. 1 omm wheel      62 Nov  8 15:56 fsimage_0000000000013698050.md5
-rw-------. 1 omm wheel  314803 Nov  8 15:59 fsimage_0000000000013698066
-rw-------. 1 omm wheel      62 Nov  8 15:59 fsimage_0000000000013698066.md5
-rw-------. 1 omm wheel       9 Oct  2 01:13 seen_txid
-rw-------. 1 omm wheel     187 Nov  8 15:59 VERSION
```

2. 查看各 JournalNode 的数据目录（查看配置项“dfs.journalnode.edits.dir”可获取，例如/srv/BigData/journalnode/hacluster/current），查看序号从第一部获取到的序号开始的 edits 文件，看是否有不连续的情况（即前一个 edits 文件的最后一个序号和 后一个 edits 文件的第一个序号 不是连续的，如下图中的 edits_0000000000013259231-0000000000013259237 就和后一个 edits_0000000000013259239-0000000000013259246 就是不连续的）。

```
-rw-------. 1 omm wheel     576 Oct  2 00:41 edits_0000000000013259151-0000000000013259158
-rw-------. 1 omm wheel     575 Oct  2 00:43 edits_0000000000013259159-0000000000013259166
-rw-------. 1 omm wheel     576 Oct  2 00:43 edits_0000000000013259167-0000000000013259174
-rw-------. 1 omm wheel     575 Oct  2 00:45 edits_0000000000013259175-0000000000013259182
-rw-------. 1 omm wheel     575 Oct  2 00:45 edits_0000000000013259183-0000000000013259190
-rw-------. 1 omm wheel     576 Oct  2 00:47 edits_0000000000013259191-0000000000013259198
-rw-------. 1 omm wheel     575 Oct  2 00:48 edits_0000000000013259199-0000000000013259206
-rw-------. 1 omm wheel     575 Oct  2 00:49 edits_0000000000013259207-0000000000013259214
-rw-------. 1 omm wheel     575 Oct  2 00:50 edits_0000000000013259215-0000000000013259222
-rw-------. 1 omm wheel     573 Oct  2 00:51 edits_0000000000013259223-0000000000013259230
-rw-------. 1 omm wheel     571 Oct  2 00:52 edits_0000000000013259231-0000000000013259237
-rw-------. 1 omm wheel     576 Oct  2 00:53 edits_0000000000013259239-0000000000013259246
-rw-------. 1 omm wheel     575 Oct  2 00:54 edits_0000000000013259247-0000000000013259254
-rw-------. 1 omm wheel     576 Oct  2 00:55 edits_0000000000013259255-0000000000013259262
-rw-------. 1 omm wheel      42 Oct  2 00:56 edits_0000000000013259263-0000000000013259264
-rw-------. 1 omm wheel    1107 Oct  2 00:57 edits_0000000000013259265-0000000000013259278
-rw-------. 1 omm wheel      42 Oct  2 00:58 edits_0000000000013259279-0000000000013259280
-rw-------. 1 omm wheel    1109 Oct  2 00:59 edits_0000000000013259281-0000000000013259294
-rw-------. 1 omm wheel      42 Oct  2 01:00 edits_0000000000013259295-0000000000013259296
-rw-------. 1 omm wheel    1299 Oct  2 01:01 edits_0000000000013259297-0000000000013259312
-rw-------. 1 omm wheel     260 Oct  2 01:02 edits_0000000000013259313-0000000000013259316
-rw-------. 1 omm wheel     984 Oct  2 01:03 edits_0000000000013259317-0000000000013259328
-rw-------. 1 omm wheel     572 Oct  2 01:04 edits_0000000000013259329-0000000000013259336
-rw-------. 1 omm wheel     575 Oct  2 01:05 edits_0000000000013259337-0000000000013259344
-rw-------. 1 omm wheel     983 Oct  2 01:06 edits_0000000000013259345-0000000000013259356
```

3. 如果有这种不连续的 edits 文件，则需要查看其它的 JournalNode 的数据目录或 NameNode 数据目录中，有没有连续的该序号相关的连续的 edits 文件。如果可以找到，复制一个连续的片段到该 JournalNode。
4. 如此把所有的不连续的 edits 文件全部都修复。
5. 重启 NameNode，观察是否成功。如还是失败，请联系技术支持。

10 使用 Hive

10.1 从零开始使用 Hive

Hive 是基于 Hadoop 的一个数据仓库工具，可将结构化的数据文件映射成一张数据库表，并提供类 SQL 的功能对数据进行分析处理，通过类 SQL 语句快速实现简单的 MapReduce 统计，不必开发专门的 MapReduce 应用，十分适合数据仓库的统计分析。

背景信息

假定用户开发一个应用程序，用于管理企业中的使用 A 业务的用户信息，使用 Hive 客户端实现 A 业务操作流程如下：

普通表的操作：

- 创建用户信息表 user_info。
- 在用户信息中新增用户的学历、职称信息。
- 根据用户编号查询用户姓名和地址。
- A 业务结束后，删除用户信息表。

表10-1 用户信息

编号	姓名	性别	年龄	地址
12005000201	A	男	19	A 城市
12005000202	B	女	23	B 城市
12005000203	C	男	26	C 城市
12005000204	D	男	18	D 城市
12005000205	E	女	21	E 城市
12005000206	F	男	32	F 城市
12005000207	G	女	29	G 城市
12005000208	H	女	30	H 城市

编号	姓名	性别	年龄	地址
12005000209	I	男	26	I 城市
12005000210	J	女	25	J 城市

操作步骤

步骤 1 下载客户端配置文件。

- MRS 3.x 之前版本，操作如下：
  a. 登录 MRS Manager 页面，具体请参见访问集群 Manager，然后选择“服务管理”。
  b. 单击“下载客户端”。
     “客户端类型”选择“仅配置文件”，“下载路径”选择“服务器端”，单击“确定”开始生成客户端配置文件，文件生成后默认保存在主管理节点“/tmp/MRS-client”。
- MRS 3.x 及后续版本，操作如下：
  a. 登录 FusionInsight Manager 页面，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本）。
  b. 选择“集群 > *待操作集群的名称* > 概览 > 更多 > 下载客户端”。
  c. 下载集群客户端。
     “选择客户端类型”选择“仅配置文件”，选择平台类型，单击“确定”开始生成客户端配置文件，文件生成后默认保存在主管理节点“/tmp/FusionInsight-Client/”。

步骤 2 登录 Manager 的主管理节点。

- MRS 3.x 之前版本，操作如下：
  a. 在 MRS 控制台，选择“集群列表 > 现有集群”，单击集群名称，在“节点管理”页签中查看节点名称，名称中包含“master1”的节点为 Master1 节点，名称中包含“master2”的节点为 Master2 节点。
     MRS Manager 的主备管理节点默认安装在集群 Master 节点上。在主备模式下，由于 Master1 和 Master2 之间会切换，Master1 节点不一定是 MRS Manager 的主管理节点，需要在 Master1 节点中执行命令，确认 MRS Manager 的主管理节点。命令请参考步骤 2.d。
  b. 以 **root** 用户使用密码方式登录 Master1 节点。操作方法，请参见“用户指南 > 连接集群 > 登录集群 > 登录集群节点”章节。
  c. 切换至 **omm** 用户。
     sudo su - root
     su - omm
  d. 执行以下命令确认 MRS Manager 的主管理节点。
     sh ${BIGDATA_HOME}/om-0.0.1/sbin/status-oms.sh

回显信息中“HAActive”参数值为“active”的节点为主管理节点（如下例中“mgtomsdat-sh-3-01-1”为主管理节点），参数值为“standby”的节点为备管理节点（如下例中“mgtomsdat-sh-3-01-2”为备管理节点）。

```
Ha mode
double
NodeName            HostName                    HAVersion         StartTime
HAActive            HAAllResOK          HARunPhase
192-168-0-30         mgtomsdat-sh-3-01-1          V100R001C01       2014-11-
18 23:43:02      active              normal              Actived
192-168-0-24         mgtomsdat-sh-3-01-2          V100R001C01       2014-11-
21 07:14:02      standby             normal              Deactived
```

e. 使用 **root** 用户登录 Manager 的主管理节点，例如“192-168-0-30”节点。

- MRS 3.x 及后续版本，操作如下：

a. 以 **root** 用户登录任意部署 Manager 的节点。

b. 执行以下命令确认主备管理节点。

sh ${BIGDATA_HOME}/om-server/om/sbin/status-oms.sh

界面打印信息中“HAActive”参数值为“active”的节点为主管理节点（如下例中“node-master1”为主管理节点），参数值为“standby”的节点为备管理节点（如下例中“node-master2”为备管理节点）。

```
HAMode
double
NodeName            HostName       HAVersion         StartTime
HAActive            HAAllResOK          HARunPhase
192-168-0-30         node-master1    V100R001C01       2020-05-01 23:43:02
active              normal              Actived
192-168-0-24         node-master2    V100R001C01       2020-05-01 07:14:02
standby             normal              Deactived
```

c. 以 **root** 用户登录主管理节点，并执行以下命令切换到 **omm** 用户。

sudo su - omm

步骤 3 执行以下命令切换到客户端安装目录。

提前已安装集群客户端，以下客户端安装目录为举例，请根据实际情况修改。

cd /opt/client

步骤 4 执行以下命令，更新主管理节点的客户端配置。

sh refreshConfig.sh /opt/client *客户端配置文件压缩包完整路径*

例如，执行命令：

sh refreshConfig.sh /opt/client /tmp/FusionInsight-Client/FusionInsight_Cluster_1_Services_Client.tar

界面显示以下信息表示配置刷新更新成功：

```
ReFresh components client config is complete.
Succeed to refresh components client config.
```

说明

步骤 1~步骤 4 的操作也可以参考“用户指南 > 使用 MRS 客户端 > 更新客户端”页面的方法二操作。

步骤 5 在 Master 节点使用客户端。

1. 在已更新客户端的主管理节点，例如“192-168-0-30”节点，执行以下命令切换到客户端目录，客户端安装目录如：/opt/client。

   cd /opt/client

2. 执行以下命令配置环境变量。

   source bigdata_env

3. 如果当前集群已启用 Kerberos 认证，执行以下命令认证当前用户，当前用户需要具有创建 Hive 表的权限，具体请参见“用户指南 > 管理现有集群 > MRS 多用户权限管理 > 创建角色”配置拥有对应权限的角色，参考“用户指南 > 管理现有集群 > MRS 多用户权限管理 > 创建用户”为用户绑定对应角色。如果当前集群未启用 Kerberos 认证，则无需执行此命令。

   kinit *MRS 集群用户*

   例如，**kinit hiveuser**。

4. 直接执行 Hive 组件的客户端命令。

   beeline

步骤 6 运行 Hive 客户端命令，实现 A 业务。

内部表的操作：

1. 根据表 10-1 创建用户信息表 user_info 并添加相关数据，例如：

   create table user_info(id string,name string,gender string,age int,addr string);

   MRS 1.x 和 MRS 3.x 及后续版本，操作如下：

   insert into table user_info(id,name,gender,age,addr) values("12005000201","A","男",19,"A 城市");

   MRS 2.x 版本，操作如下：

   insert into table user_info values("12005000201","A","男",19,"A 城市");

2. 在用户信息表 user_info 中新增用户的学历、职称信息。

   以增加编号为 12005000201 的用户的学历、职称信息为例，其他用户类似。

   alter table user_info add columns(education string,technical string);

3. 根据用户编号查询用户姓名和地址。

   以查询编号为 12005000201 的用户姓名和地址为例，其他用户类似。

   select name,addr from user_info where id='12005000201';

4. 删除用户信息表。

   drop table user_info;

外部分区表的操作：

创建外部分区表并导入数据：

1. 创建外部表数据存储路径：

hdfs dfs -mkdir /hive/

hdfs dfs -mkdir /hive/user_info

2. 建表：

   create external table user_info(id string,name string,gender string,age int,addr string) partitioned by(year string) row format delimited fields terminated by ' ' lines terminated by '\n' stored as textfile location '/hive/user_info';

   说明

   fields terminated 指明分隔的字符，如按空格分隔，' '。

   lines terminated 指明分行的字符，如按换行分隔，'\n'。

   /hive/user_info 为数据文件的路径。

3. 导入数据。

   a. 使用 insert 语句插入数据。

      insert into user_info partition(year="2018") values ("12005000201","A","男",19,"A 城市");

   b. 使用 load data 命令导入文件数据。

      i. 根据表 10-1 数据创建文件。如，文件名为 txt.log，以空格拆分字段，以换行符作为行分隔符。

      ii. 上传文件至 hdfs。

      hdfs dfs -put txt.log /tmp

      iii. 加载数据到表中。

      load data inpath '/tmp/txt.log' into table user_info partition (year='2011');

4. 查询导入数据。

   select * from user_info;

5. 删除用户信息表。

   drop table user_info;

6. 执行以下命令退出客户端。

   !q

----结束

10.2 配置 Hive 常用参数

参数入口

请参考修改集群服务配置参数进入 Hive 服务配置页面。

参数说明

表10-2 Hive 参数说明

参数	参数说明	默认值

参数	参数说明	默认值
hive.auto.convert.join	Hive 基于输入文件大小将普通 join 转为 mapjoin 的开关。 说明 在使用 Hive 进行联表查询，且关联的表无大小表的分别（小表数据<24M）时，建议将此参数值改为 false，如果此时将此参数设置为 true，执行联表查询时无法生成新的 mapjoin。	取值范围： • true • false 默认值为 true
hive.default.fileformat	Hive 使用的默认文件格式。	MRS 3.x 之前版本：TextFile MRS 3.x 及后续版本：RCFile
hive.exec.reducers.max	Hive 提交的 MR 任务中 reducer 的最大个数。	999
hive.server2.thrift.max.worker.threads	HiveServer 内部线程池，最大能启动的线程数量。	1000
hive.server2.thrift.min.worker.threads	HiveServer 内部线程池，初始化时启动的线程数量。	5
hive.hbase.delete.mode.enabled	从 Hive 删除 HBase 记录的功能开关。如果启用，用户可以使用“remove table xx where xxx”从 Hive 中删除 HBase 记录。 说明 本参数适用于 MRS 3.x 及后续版本。	true
hive.metastore.server.min.threads	MetaStore 启动的用于处理连接的线程数，如果超过设置的值之后，MetaStore 就会一直维护不低于设定值的线程数，即常驻 MetaStore 线程池的线程会维护在指定值之上。	200
hive.server2.enable.doAs	HiveServer2 在与其他服务（如 YARN、HDFS 等）会话时是否模拟客户端用户。如果将此配置项从 false 改成 true，会导致只有列权限的用户访问相应表权限缺失。 说明 本参数适用于 MRS 3.x 及后续版本。	true

10.3 Hive SQL

Hive SQL 支持 Hive-3.1.0 版本中的所有特性，详情请参见 https://cwiki.apache.org/confluence/display/hive/languagemanual。

系统提供的扩展 Hive 语句如表 10-3 所示。

表10-3 扩展 Hive 语句

扩展语法	语法说明	语法示例	示例说明
CREATE [TEMPORARY] [EXTERNAL] TABLE [IF NOT EXISTS] [db_name.]table_name (col_name data_type [COMMENT col_comment], ...) [ROW FORMAT row_format] [STORED AS file_format] | STORED BY 'storage.handler.class.name' [WITH SERDEPROPERTIES (...)] ..... [TBLPROPERTIES ("groupId"=" group1","locatorId"="locator1")] ...;	创建一个 hive 表，并指定表数据文件分布的 locator 信息。详细说明请参见使用 HDFS Colocation 存储 Hive 表。	CREATE TABLE tab1 (id INT, name STRING) row format delimited fields terminated by '\t' stored as RCFILE TBLPROPERTIES("groupId"=" group1","locatorId"="locator1");	创建表 tab1，并指定 tab1 的表数据分布在 locator1 节点上。
CREATE [TEMPORARY] [EXTERNAL] TABLE [IF NOT EXISTS] [db_name.]table_name (col_name data_type [COMMENT col_comment], ...) [ROW FORMAT row_format] [STORED AS file_format] | STORED BY 'storage.handler.class.name' [WITH SERDEPROPERTIES (...)] ... [TBLPROPERTIES	创建一个 hive 表，并指定表的加密列和加密算法。详细说明请参见使用 Hive 列加密功能。	create table encode_test(id INT, name STRING, phone STRING, address STRING) ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe' WITH SERDEPROPERTIES ('column.encode.indices'='2,3', 'column.encode.classname'='org.apache.hadoop.hive.serde2.SMS4Rewriter') STORED AS	创建表 encode_test，并指定插入数据时对第 2、3 列加密，加密算法类为 org.apache.hadoop.hive.serde2.SMS4Rewriter。

扩展语法	语法说明	语法示例	示例说明
('**column.encode.columns**'='col_name1, col_name2'| '**column.encode.indices**'='col_id1,col_id2', '**column.encode.classname**'='encode_classname')]...;		TEXTFILE;	
REMOVE TABLE *hbase_tablename* [WHERE where_condition];	删除 hive on hbase 表中符合条件的数据。详细说明请参见删除 Hive on HBase 表中的单行记录。	remove table hbase_table1 where id = 1;	删除表中符合条件“id =1”的数据。
CREATE [TEMPORARY] [EXTERNAL] TABLE [IF NOT EXISTS] [db_name.]table_name (col_name data_type [COMMENT col_comment], ...) [ROW FORMAT row_format] **STORED AS inputformat 'org.apache.hadoop.hive.contrib.fileformat.SpecifiedDelimiterInputFormat'** outputformat 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat';	创建 hive 表，并设定表可以指定自定义行分隔符。详细说明请参见自定义行分隔符。	create table blu(time string, num string, msg string) row format delimited fields terminated by ',' **stored as inputformat 'org.apache.hadoop.hive.contrib.fileformat.SpecifiedDelimiterInputFormat'** outputformat 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat';	创建表 blu，指定 inputformat 为 SpecifiedDelimiterInputFormat，以便查询时可以指定表的查询行分隔符。

10.4 权限管理

10.4.1 Hive 权限介绍

Hive 是建立在 Hadoop 上的数据仓库框架，提供类似 SQL 的 HQL 操作结构化数据。

MRS 提供用户、用户组和角色，集群中的各类权限需要先授予角色，然后将用户或者用户组与角色绑定。用户只有绑定角色或者加入绑定角色的用户组，才能获得权限。Hive 授权相关信息请参考：https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Authorization。

说明

- Hive 在安全模式下需要进行权限管理，在普通模式下无需进行权限管理。
- MRS 3.x 及后续版本支持 Ranger，如果当前组件使用了 Ranger 进行权限控制，须基于 Ranger 配置相关策略进行权限管理，具体操作可参考添加 Hive 的 Ranger 访问权限策略。

Hive 权限模型

使用 Hive 组件，必须对 Hive 数据库和表（含外表和视图）拥有相应的权限。在 MRS 中，完整的 Hive 权限模型由 Hive 元数据权限与 HDFS 文件权限组成。使用数据库或表时所需要的各种权限都是 Hive 权限模型中的一种。

- Hive 元数据权限。

  与传统关系型数据库类似，MRS 的 Hive 数据库包含“建表”和“查询”权限，Hive 表和列包含“查询”、“插入”和“删除”权限。Hive 中还包含拥有者权限“OWNERSHIP”和“Hive 管理员权限”。

  说明

  Hive 表和列的“更新”和“删除”操作只有在开启“ACID”的情况下支持。

- Hive 数据文件权限，即 HDFS 文件权限。

  Hive 的数据库、表对应的文件保存在 HDFS 中。默认创建的数据库或表保存在 HDFS 目录“/user/hive/warehouse”。系统自动以数据库名称和数据库中表的名称创建子目录。访问数据库或者表，需要在 HDFS 中拥有对应文件的权限，包含“读”、“写”和“执行”权限。

  说明

  MRS 3.X 支持 Hive 多实例，多实例场景下，目录为“/user/hiven（n=1 ~4）/warehouse”。

用户对 Hive 数据库或表执行不同操作时，需要关联不同的元数据权限与 HDFS 文件权限。例如，对 Hive 数据表执行查询操作，需要关联元数据权限“查询”，以及 HDFS 文件权限“读”和“写”。

使用 Manager 界面图形化的角色管理功能来管理 Hive 数据库和表的权限，只需要设置元数据权限，系统会自动关联 HDFS 文件权限，减少界面操作，提高效率。

Hive 用户对象

MRS 提供了用户和角色来使用 Hive，比如创建表、在表中插入数据或者查询表。Hive 中定义了“USER”类，对应用户实例；定义了“GROUP”类，对应角色实例。

使用 Manager 设置 Hive 用户对象的权限，只支持在角色中设置，用户或用户组需要绑定角色才能获得权限。支持授予管理员权限、访问数据库、表和列的权限。

Hive 使用场景及对应权限

用户使用 Hive 并创建数据库需要加入 hive 组，不需要角色授权。用户在 Hive 和 HDFS 中对自己创建的数据库或表拥有完整权限，可直接创建表、查询数据、删除数据、插入数据、更新数据以及授权他人访问表与对应 HDFS 目录与文件。

如果用户访问别人创建的表或数据库，需要授予权限。所以根据 Hive 使用场景的不同，用户需要的权限可能也不相同。

表10-4 Hive 使用场景

主要场景	用户需要的权限
使用 Hive 表、列或数据库	使用其他用户创建的 Hive 表、列或数据库，不同的场景需要不同的 Hive 权限，例如： • 创建表，需要“建表”。 • 查询数据，需要“查询”。 • 插入数据，需要“插入”。 • 删除数据，需要“删除”。
关联使用其他组件	部分场景除了 Hive 权限，还可能需要组件的权限，例如： • 执行部分 HQL 命令，例如 **insert**，**count**，**distinct**，**group by**，**order by**，**sort by** 或 **join** 等语句时，需要设置 YARN 权限。建议为每个 Hive 用户的角色添加此权限。 • 使用 Hive over HBase，例如在 Hive 中查询 HBase 表数据，需要设置 HBase 权限。

在一些特殊 Hive 使用场景下，需要单独设置其他权限。

表10-5 Hive 授权注意事项

可能场景	用户需要的权限
创建 Hive 数据库、表、外表，或者为已经创建的 Hive 表或外表添加分区，且 Hive 用户指定数据文件保存在“/user/hive/warehouse”以外的 HDFS 目录。	需要此目录已经存在，Hive 用户是目录的属主，且用户对目录拥有“读”、“写”和“执行”权限。同时用户对此目录上层的每一级目录都拥有“读”和“写”权限。然后管理员通过角色管理功能授予角色使用 Hive 的权限，会自动关联 HDFS 权限。
Hive 用户使用 **load** 将指定目录下所有文件或者指定文件，导入数据到 Hive 表。	• 数据源为 Linux 本地磁盘，指定目录时需要此目录已经存在，系统用户

可能场景	用户需要的权限
	“omm”对此目录以及此目录上层的每一级目录拥有“r”和“x”的权限。指定文件时需要此文件已经存在，“omm”对此文件拥有“r”的权限，同时对此文件上层的每一级目录拥有“r”和“x”的权限。 • 数据源为 HDFS，指定目录时需要此目录已经存在，Hive 用户是目录属主，且用户对此目录及其子目录拥有“读”、“写”和“执行”权限，并且其上层的每一级目录拥有“读”和“写”权限。指定文件时需要此文件已经存在，Hive 用户是文件属主，且用户对文件拥有“读”、“写”和“执行”权限，同时对此文件上层的每一级目录拥有“读”和“执行”权限。 说明 使用 **load** 从 Linux 本地磁盘导入数据时，文件需上传到执行命令的 HiveServer 并修改权限。建议使用客户端执行命令，可查看客户端连接的 HiveServer。例如，Hive 客户端显示“0: jdbc:hive2://10.172.0.43:21066/>”，表示当前连接的 HiveServer 节点 IP 地址为“10.172.0.43”。
创建函数、删除函数或者修改任意数据库。	需要授予“Hive 管理员权限”。
操作 Hive 中所有的数据库和表。	需加入到 supergroup 用户组，并且授予“Hive 管理员权限”。

10.4.2 创建 Hive 角色

操作场景

该任务指导系统管理员在 Manager 创建并设置 Hive 的角色。Hive 角色可设置 Hive 管理员权限以及 Hive 数据表的数据操作权限。

用户使用 Hive 并创建数据库需要加入 hive 组，不需要角色授权。用户在 Hive 和 HDFS 中对自己创建的数据库或表拥有完整权限，可直接创建表、查询数据、删除数据、插入数据、更新数据以及授权他人访问表与对应 HDFS 目录与文件。默认创建的数据库或表保存在 HDFS 目录“/user/hive/warehouse”。

说明

- 安全模式支持创建 Hive 角色，普通模式不支持创建 Hive 角色。
- MRS 3.x 及后续版本支持 Ranger，如果当前组件使用了 Ranger 进行权限控制，须基于 Ranger 配置相关策略进行权限管理，具体操作可参考添加 Hive 的 Ranger 访问权限策略。

前提条件

- 系统管理员已明确业务需求。
- 已登录 Manager。
- 已安装好 Hive 客户端。

操作步骤

MRS 3.x 之前版本，创建 Hive 角色的操作如下：

步骤 1 登录 MRS Manager。

步骤 2 选择“系统设置 > 权限配置 > 角色管理”。

步骤 3 单击“添加角色”，输入“角色名称”和“描述”。

步骤 4 设置角色“权限”请参见表 10-6。

- “Hive Admin Privilege”：Hive 管理员权限。如需使用该权限，在执行 SQL 语句时需要先执行 **set role admin** 来设置权限。
- “Hive Read Write Privileges”：Hive 数据表管理权限，可设置与管理已创建的表的数据操作权限。根据需要勾选相应 database 的权限，如果要精确到表，可以单击 database 名称，勾选相应表的权限。

说明

- Hive 角色管理支持授予管理员权限、访问表和视图的权限，不支持数据库的授权。
- Hive 管理员权限不支持管理 HDFS 的权限。
- 如果数据库中的表或者表中的文件数量比较多，在授权时可能需要等待一段时间。例如表的文件数量为 1 万时，可能需要等待 2 分钟。

表10-6 设置角色

任务场景	角色授权操作
设置 Hive 管理员权限	在“权限”的表格中单击“Hive”，勾选“Hive Admin Privilege”。 说明 用户绑定 Hive 管理员角色后，在每个维护操作会话中，还需要执行以下操作： 1. 请根据客户端所在位置，参考“用户指南 > 使用 MRS 客户端 > 安装客户端”章节，登录安装客户端的节点。

任务场景	角色授权操作
	2. 执行以下命令配置环境变量。 例如，Hive 客户端安装目录为“/opt/hiveclient”，执行 **source /opt/hiveclient/bigdata_env** 3. 执行以下命令认证用户。 **kinit** *Hive 业务用户* 4. 执行以下命令登录客户端工具。 **beeline** 5. 执行以下命令更新用户的管理员权限。 **set role admin;**
设置在默认数据库中，查询其他用户表的权限	1. 在“权限”的表格中选择“Hive > Hive Read Write Privileges”。 2. 在指定表的“权限”列，勾选“Select”。
设置在默认数据库中，插入其他用户表的权限	1. 在“权限”的表格中选择“Hive > Hive Read Write Privileges”。 2. 在指定表的“权限”列，勾选“Insert”。
设置在默认数据库中，导入数据到其他用户表的权限	1. 在“权限”的表格中选择“Hive > Hive Read Write Privileges”。 2. 在指定表的“权限”列，勾选“Delete”和“Insert”。
设置提交 Hql 命令到 Yarn 执行的权限	部分业务需求使用的 Hql 命令将转化为 MapReduce 任务并提交到 Yarn 中执行，需要设置 Yarn 权限。例如运行的 HQL 使用了 **insert**，**count**，**distinct**，**group by**，**order by**，**sort by** 或 **join** 等语句的相关场景。 1. 在“权限”的表格中选择“Yarn > Scheduler Queue > root”。 2. 在“default”队列的“权限”列，勾选“Submit”。

步骤 5 单击“确定”，返回“角色”。

步骤 6 选择“系统设置 > 用户管理 > 添加用户”。

步骤 7 输入用户名，在“用户类型”选择“人机”类型，设置用户密码，在用户组添加一个绑定了 Hive 管理员角色的用户组，并绑定新创建的 Hive 角色，单击“确定”完成 Hive 用户创建。

步骤 8 待用户生成后，即可使用该用户执行相应 SQL 语句。

----结束

MRS 3.x 及后续版本，创建 Hive 角色的操作如下：

步骤 1 登录 FusionInsight Manager，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本）。

步骤 2 选择“系统 > 权限 > 角色”。

步骤 3 单击“添加角色”，输入“角色名称”和“描述”。

步骤 4 设置角色“配置资源权限”请参见表 10-7。

- 设置 HDFS 目录的读和执行权限。
  - 选择“*待操作集群的名称* > HDFS > 文件系统 > hdfs://hacluster/ > user”，在“hive”的“权限”列，勾选“读”和“执行”。
  - 选择“*待操作集群的名称* > HDFS > 文件系统 > hdfs://hacluster/ > user > hive”，在“warehouse”的“权限”列，勾选“读”和“执行”。
  - 选择“*待操作集群的名称* > HDFS > 文件系统 > hdfs://hacluster/ > tmp”,在“hive-scratch”的“权限”列，勾选“读”和“执行”。
- “Hive 管理员权限”：Hive 管理员权限。
- “Hive 读写权限”：Hive 数据表管理权限，可设置与管理已创建的表的数据操作权限。

说明

- MRS 3.1.0 版本，Hive 角色管理支持授予管理员权限、访问表和视图的权限，不支持数据库的授权。
- Hive 管理员权限不支持管理 HDFS 的权限。
- 如果数据库中的表或者表中的文件数量比较多，在授权时可能需要等待一段时间。例如表的文件数量为 1 万时，可能需要等待 2 分钟。

表10-7 设置角色

任务场景	角色授权操作
设置 Hive 管理员权限	在“配置资源权限”的表格中选择“*待操作集群的名称* > Hive”，勾选“Hive 管理员权限”。 说明 用户绑定 Hive 管理员角色后，在每个维护操作会话中，还需要执行以下操作： 1. 以客户端安装用户，登录安装 Hive 客户端的节点。 2. 执行以下命令配置环境变量。 例如，Hive 客户端安装目录为“/opt/hiveclient”，执行 **source**

任务场景	角色授权操作
	/opt/hiveclient/bigdata_env 3. 执行以下命令认证用户。 **kinit** *Hive 业务用户* 4. 执行以下命令登录客户端工具。 **beeline** 5. 执行以下命令更新用户的管理员权限。 **set role admin;**
设置在默认数据库中，查询其他用户表的权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > Hive > Hive 读写权限”。 2. 在数据库列表中单击指定的数据库名称，显示数据库中的表。 3. 在指定表的“权限”列，勾选“查询”。
设置在默认数据库中，插入其他用户表的权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > Hive > Hive 读写权限”。 2. 在数据库列表中单击指定的数据库名称，显示数据库中的表。 3. 在指定表的“权限”列，勾选“插入”。
设置在默认数据库中，导入数据到其他用户表的权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > Hive > Hive 读写权限”。 2. 在数据库列表中单击指定的数据库名称，显示数据库中的表。 3. 在指定表的“权限”列，勾选“删除”和“插入”。
设置提交 Hql 命令到 Yarn 执行的权限	部分业务需求使用的 Hql 命令将转化为 MapReduce 任务并提交到 Yarn 中执行，需要设置 Yarn 权限。例如运行的 HQL 使用了 **insert**，**count**，**distinct**，**group by**，**order by**，**sort by** 或 **join** 等语句的相关场景。 1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > Yarn > 调度队列 > root”。 2. 在“default”队列的“权限”列，勾选“提交”。

步骤 5 单击“确定”，返回“角色”。

----结束

10.4.3 配置 Hive 表、列或数据库的权限

操作场景

使用 Hive 表或者数据库时，如果用户访问别人创建的表或数据库，需要授予对应的权限。为了实现更严格权限控制，Hive 也支持列级别的权限控制。如果要访问别人创建的表上某些列，需要授予列权限。以下介绍使用 Manager 角色管理功能在表授权、列授权和数据库授权三个场景下的操作。

说明

- 安全模式支持配置 Hive 表、列或数据库的权限，普通模式不支持配置 Hive 表、列或数据库的权限。
- MRS 3.x 及后续版本支持 Ranger，如果当前组件使用了 Ranger 进行权限控制，须基于 Ranger 配置相关策略进行权限管理，具体操作可参考添加 Hive 的 Ranger 访问权限策略。

前提条件

- 获取一个拥有管理员权限的用户，例如“admin”。
- 请参考创建 Hive 角色，在 Manager 界面创建一个角色，例如“hrole”，不需要设置 Hive 权限，设置提交 Hql 命令到 Yarn 执行的权限。
- 在 Manager 界面创建两个使用 Hive 的“人机”用户并加入“hive”组，例如“huser1”和“huser2”。“huser2”需绑定“hrole”。使用“huser1”创建一个数据库“hdb”，并在此数据库中创建表“htable”。

操作步骤

- 表授权

  用户在 Hive 和 HDFS 中对自己创建的表拥有完整权限，用户访问别人创建的表，需要授予权限。授予权限时只需要授予 Hive 元数据权限，HDFS 文件权限将自动关联。以授予用户对应角色在表“htable”中查询、插入和删除数据的权限为例，操作步骤如下：

  MRS 3.x 之前版本，表授权的操作如下：

  a. 在 MRS Manager 界面，选择“系统设置 > 权限配置 > 角色管理”。
  b. 在角色“hrole”所在行，单击“修改”。
  c. 选择“Hive > Hive Read Write Privileges”。
  d. 在数据库列表中单击指定的数据库名称“hdb”，显示数据库中的表“htable”。
  e. 在表“htable”的“权限”列，勾选“Select”、“Insert”和“Delete”。
  f. 单击“确定”完成。

  MRS 3.x 及后续版本，表授权的操作如下：

  a. 在 FusionInsight Manager 界面，选择“系统 > 权限 > 角色”。
  b. 在角色“hrole”所在行，单击“修改”。

c. 选择“*待操作的集群* > Hive > Hive 读写权限”。

d. 在数据库列表中单击指定的数据库名称“hdb”，显示数据库中的表“htable”。

e. 在表“htable”的“权限”列，勾选“查询”、“插入”和“删除”。

f. 单击“确定”完成。

说明

在角色管理中，授予角色在 Hive 外表中查询、插入和删除数据的操作与 Hive 表相同，授予元数据权限将自动关联 HDFS 文件权限。

- 列授权

  用户在 Hive 和 HDFS 中对自己创建的表拥有完整权限，用户没有权限访问别人创建的表。如果要访问别人创建的表上某些列，需要授予列权限。授予权限时只需要授予 Hive 元数据权限，HDFS 文件权限将自动关联。以授予用户对应角色在表“htable”的列“hcol”中查询、插入数据的权限为例，操作步骤如下：

  MRS 3.x 之前版本，列授权的操作如下：

  a. 在 MRS Manager 界面，选择“系统设置 > 权限配置 > 角色管理”。

  b. 在角色“hrole”所在行，单击“修改”。

  c. 选择“Hive > Hive Read Write Privileges”。

  d. 在数据库列表中单击指定的数据库名称“hdb”，显示数据库中的表“htable”，单击表“htable”，显示表下的列“hcol”。

  e. 在列“hcol”的“权限”列，勾选“Select”和“Insert”。

  f. 单击“确定”完成。

  MRS 3.x 及后续版本，列授权的操作如下：

  a. 在 FusionInsight Manager 界面，选择“系统 > 权限 > 角色”。

  b. 在角色“hrole”所在行，单击“修改”。

  c. 选择“*待操作的集群* > Hive > Hive 读写权限”。

  d. 在数据库列表中单击指定的数据库名称“hdb”，显示数据库中的表“htable”，单击表“htable”，显示表下的列“hcol”。

  e. 在列“hcol”的“权限”列，勾选“查询”和“插入”。

  f. 单击“确定”完成。

说明

在权限管理中，授予元数据权限将自动关联 HDFS 文件权限，所以列授权后会增加表对应所有文件的 HDFS ACL 权限。

- 数据库授权

  用户在 Hive 和 HDFS 中对自己创建的数据库拥有完整权限，用户访问别人创建的数据库，需要授予权限。授予权限时只需要授予 Hive 元数据权限，HDFS 文件权限将自动关联。以授予用户对应角色在数据库“hdb”中查询和创建表的权限为例，操作步骤如下，不支持对角色授予数据库其他的操作权限：

  MRS 3.x 之前版本，数据库授权的操作如下：

  a. 在 MRS Manager 界面，选择“系统设置 > 权限配置 > 角色管理”。

b. 在角色“hrole”所在行，单击“修改”。
c. 选择“Hive > Hive Read Write Privileges”。
d. 在数据库“hdb”的“权限”列，勾选“Select”和“Create”。
e. 单击“确定”完成。

MRS 3.x 及后续版本，数据库授权的操作如下：

a. 在 FusionInsight Manager 界面，选择“系统 > 权限 > 角色”。
b. 在角色“hrole”所在行，单击“修改”。
c. 选择“*待操作的集群* > Hive > Hive 读写权限”。
d. 在数据库“hdb”的“权限”列，勾选“查询”和“建表”。
e. 单击“确定”完成。

说明

- 在权限管理中，为了方便用户使用，授予数据库下表的任意权限将自动关联该数据库目录的 HDFS 权限。为了避免产生性能问题，取消表的任意权限，系统不会自动取消数据库目录的 HDFS 权限，但对应的用户只能登录数据库和查看表名。
- 若为角色添加或删除数据库的查询权限，数据库中的表也将自动添加或删除查询权限。

相关概念

表10-8 使用 Hive 表、列或数据库场景权限一览

操作场景	用户需要的权限
DESCRIBE TABLE	查询（Select）
SHOW PARTITIONS	查询（Select）
ANALYZE TABLE	查询（Select）、插入（Insert）
SHOW COLUMNS	查询（Select）
SHOW TABLE STATUS	查询（Select）
SHOW TABLE PROPERTIES	查询（Select）
SELECT	查询（Select）
EXPLAIN	查询（Select）
CREATE VIEW	查询（Select）、Select 授权（Grant Of Select）、建表（Create）
SHOW CREATE TABLE	查询（Select）、Select 授权（Grant Of Select）
CREATE TABLE	建表（Create）
ALTER TABLE ADD PARTITION	插入（Insert）
INSERT	插入（Insert）

操作场景	用户需要的权限
INSERT OVERWRITE	插入（Insert）、删除（Delete）
LOAD	插入（Insert）、删除（Delete）
ALTER TABLE DROP PARTITION	删除（Delete）
CREATE FUNCTION	Hive 管理员权限（Hive Admin Privilege）
DROP FUNCTION	Hive 管理员权限（Hive Admin Privilege）
ALTER DATABASE	Hive 管理员权限（Hive Admin Privilege）

10.4.4 配置 Hive 业务使用其他组件的权限

操作场景

Hive 业务还可能需要关联使用其他组件，例如 HQL 语句触发 MapReduce 任务需要设置 Yarn 权限，或者 Hive over HBase 的场景需要 HBase 权限。以下介绍 Hive 关联 Yarn 和 Hive over HBase 两个场景下的操作。

说明

- 安全模式下 Yarn 和 HBase 的权限管理默认是开启的，因此在安全模式下默认需要配置 Yarn 和 HBase 权限。
- 在普通模式下，Yarn 和 HBase 的权限管理默认是关闭的，即任何用户都有权限，因此普通模式下默认不需要配置 Yarn 和 HBase 权限。如果用户修改了 YARN 或者 HBase 的配置来开启权限管理，则修改后也需要配置 Yarn 和 HBase 权限。
- MRS 3.x 及后续版本支持 Ranger，如果当前组件使用了 Ranger 进行权限控制，须基于 Ranger 配置相关策略进行权限管理，具体操作可参考添加 Hive 的 Ranger 访问权限策略。

前提条件

- 完成 Hive 客户端的安装。例如安装目录为“/opt/client”。
- 获取一个拥有管理员权限的用户，例如“admin”。

操作步骤

MRS 3.x 之前版本，Hive 关联 Yarn

用户如果执行 **insert**，**count**，**distinct**，**group by**，**order by**，**sort by** 或 **join** 等语句时，将触发 MapReduce 任务，需要设置 Yarn 权限。以授予角色在表“thc”执行 **count** 语句的权限为例，操作步骤如下：

步骤 1 在 MRS Manager 角色界面创建一个角色。

步骤 2 在“权限”的表格中选择“Yarn > Scheduler Queue > root”。

步骤 3 在“default”队列的“权限”列，勾选“Submit”，单击“确定”保存。

步骤 4 在“权限”的表格中选择“Hive > Hive Read Write Privileges > default”，勾选表“thc”的“Select”，单击“确定”保存。

----结束

MRS 3.x 及后续版本，Hive 关联 Yarn

用户如果执行 **insert**，**count**，**distinct**，**group by**，**order by**，**sort by** 或 **join** 等语句时，将触发 MapReduce 任务，需要设置 Yarn 权限。以授予角色在表“thc”执行 **count** 语句的权限为例，操作步骤如下：

步骤 1 在 FusionInsight Manager 角色界面创建一个角色。

步骤 2 在“配置资源权限”的表格中选择“*待操作集群的名称* > Yarn > 调度队列 > root”。

步骤 3 在“default”队列的“权限”列，勾选“提交”，单击“确定”保存。

步骤 4 在“配置资源权限”的表格中选择“*待操作集群的名称* > Hive > Hive 读写权限 > default”，勾选表“thc”的“查询”，单击“确定”保存。

----结束

MRS 3.x 之前版本，Hive over HBase 授权

用户如果需要使用类似 SQL 语句的方式来操作 HBase 表，授予权限后可以在 Hive 中使用 HQL 命令访问 HBase 表。以授予用户在 Hive 中查询 HBase 表的权限为例，操作步骤如下

步骤 1 在 MRS Manager 角色管理界面创建一个 HBase 角色，例如“hive_hbase_create”，并授予创建 HBase 表的权限。

在“权限”的表格中选择“HBase > HBase Scope > global”，勾选命名空间“default”的“Create”，单击“确定”保存。

步骤 2 在 MRS Manager 用户管理界面创建一个“人机”用户，例如“hbase_creates_user”，加入“hive”组，绑定角色“hive_hbase_create”，用于创建 Hive 表和 HBase 表。

步骤 3 请根据客户端所在位置，参考“用户指南 > 使用 MRS 客户端 > 安装客户端”章节，登录安装客户端的节点。

步骤 4 执行以下命令，配置环境变量。

source /opt/client/bigdata_env

步骤 5 执行以下命令，认证用户。

kinit hbase_creates_user

步骤 6 执行以下命令，进入 Hive 客户端 shell 环境：

beeline

步骤 7 执行以下命令，同时在 Hive 和 HBase 中创建表。例如创建表“thh”。

CREATE TABLE thh(id int, name string, country string) STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler' WITH SERDEPROPERTIES("hbase.columns.mapping" = "cf1:id,cf1:name,:key") TBLPROPERTIES ("hbase.table.name" = "thh");

创建好的 Hive 表和 HBase 表分别保存在 Hive 的数据库“default”和 HBase 的命名空间“default”。

步骤 8 在 MRS Manager 角色管理界面创建一个角色，例如“hive_hbase_select”，并授予查询 Hive 表“thh”和 HBase 表“thh”的权限。

1. 在“权限”的表格中选择“HBase > HBase Scope > global > default”，勾选表“thh”的“read”，单击“确定”保存，授予 HBase 角色查询表的权限。
2. 编辑角色，在“权限”的表格中选择“HBase > HBase Scope > global > hbase”，勾选表“hbase:meta”的“Execute”，单击“确定”保存。
3. 编辑角色，在“权限”的表格中选择“Hive > Hive Read Write Privileges > default”，勾选表“thh”的“Select”，单击“确定”保存。

步骤 9 在 MRS Manager 用户管理界面创建一个“人机”用户，例如“hbase_select_user”，加入“hive”组，绑定角色“hive_hbase_select”，用于查询 Hive 表和 HBase 表。

步骤 10 执行以下命令，配置环境变量。

source /opt/client/bigdata_env

步骤 11 执行以下命令，认证用户。

kinit hbase_select_user

步骤 12 执行以下命令，进入 Hive 客户端 shell 环境。

beeline

步骤 13 执行以下命令，使用 Hive 的 HQL 语句查询 HBase 表的数据。

select * from thh;

----结束

MRS 3.x 及后续版本，Hive over HBase 授权

用户如果需要使用类似 SQL 语句的方式来操作 HBase 表，授予权限后可以在 Hive 中使用 HQL 命令访问 HBase 表。以授予用户在 Hive 中查询 HBase 表的权限为例，操作步骤如下

步骤 1 在 FusionInsight Manager 角色管理界面创建一个 HBase 角色，例如“hive_hbase_create”，并授予创建 HBase 表的权限。

在“配置资源权限”的表格中选择“*待操作集群的名称* > HBase > HBase Scope > global”，勾选命名空间“default”的“创建”，单击“确定”保存。

步骤 2 在 FusionInsight Manager 用户管理界面创建一个“人机”用户，例如“hbase_creates_user”，加入“hive”组，绑定角色“hive_hbase_create”，用于创建 Hive 表和 HBase 表。

步骤 3 如果当前组件使用了 Ranger 进行权限控制，需给“hive_hbase_create”或“hbase_creates_user”配置“Create”权限，具体操作可参考添加 Hive 的 Ranger 访问权限策略。

步骤 4 以客户端安装用户，登录安装客户端的节点。

步骤 5 执行以下命令，配置环境变量。

source /opt/client/bigdata_env

步骤 6 执行以下命令，认证用户。

kinit hbase_creates_user

步骤 7 执行以下命令，进入 Hive 客户端 shell 环境：

beeline

步骤 8 执行以下命令，同时在 Hive 和 HBase 中创建表。例如创建表“thh”。

CREATE TABLE thh(id int, name string, country string) STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler' WITH SERDEPROPERTIES("hbase.columns.mapping" = "cf1:id,cf1:name,:key") TBLPROPERTIES ("hbase.table.name" = "thh");

创建好的 Hive 表和 HBase 表分别保存在 Hive 的数据库“default”和 HBase 的命名空间“default”。

步骤 9 在 FusionInsight Manager 角色管理界面创建一个角色，例如“hive_hbase_select”，并授予查询 Hive 表“thh”和 HBase 表“thh”的权限。

1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > HBase > HBase Scope > global > default”，勾选表“thh”的“读”，单击“确定”保存，授予 HBase 角色查询表的权限。
2. 编辑角色，在“配置资源权限”的表格中选择“*待操作集群的名称* > HBase > HBase Scope > global > hbase”，勾选表“hbase:meta”的“执行”，单击“确定”保存。
3. 编辑角色，在“配置资源权限”的表格中选择“*待操作集群的名称* > Hive > Hive 读写权限 > default”，勾选表“thh”的“查询”，单击“确定”保存。

步骤 10 在 FusionInsight Manager 用户管理界面创建一个“人机”用户，例如“hbase_select_user”，加入“hive”组，绑定角色“hive_hbase_select”，用于查询 Hive 表和 HBase 表。

步骤 11 执行以下命令，配置环境变量。

source /opt/client/bigdata_env

步骤 12 执行以下命令，认证用户。

kinit hbase_select_user

步骤 13 执行以下命令，进入 Hive 客户端 shell 环境。

beeline

步骤 14 执行以下命令，使用 Hive 的 HQL 语句查询 HBase 表的数据。

select * from thh;

----结束

10.5 使用 Hive 客户端

操作场景

该任务指导用户在运维场景或业务场景中使用 Hive 客户端。

前提条件

- 已安装客户端，例如安装目录为“/opt/hadoopclient”，以下操作的客户端目录只是举例，请根据实际安装目录修改。
- 各组件业务用户由系统管理员根据业务需要创建。安全模式下，“机机”用户需要下载 keytab 文件。“人机”用户第一次登录时需修改密码。

使用 Hive 客户端（MRS 3.x 之前版本）

步骤 1 以客户端安装用户，登录安装客户端的节点。

步骤 2 执行以下命令，切换到客户端安装目录。

cd /opt/hadoopclient

步骤 3 执行以下命令配置环境变量。

source bigdata_env

步骤 4 根据集群认证模式，完成 Hive 客户端登录。

- 安全模式，则执行以下命令，完成用户认证并登录 Hive 客户端。

  kinit *组件业务用户*

  beeline

- 普通模式，则执行以下命令，登录 Hive 客户端，如果不指定组件业务用户，则会以当前操作系统用户登录。

  beeline -n *组件业务用户*

  说明

  进行 beeline 连接后，可以编写并提交 HQL 语句执行相关任务。如需执行 Catalog 客户端命令，需要先执行**!q** 命令退出 beeline 环境。

步骤 5 使用以下命令，执行 HCatalog 的客户端命令。

hcat -e *"cmd"*

其中*"cmd"*必须为 Hive DDL 语句，如 **hcat -e "show tables"**。

说明

- 若要使用 HCatalog 客户端，必须从“组件管理”页面单击“下载客户端”，下载全部服务的客户端。Beeline 客户端不受此限制。
- 由于权限模型不兼容，使用 HCatalog 客户端创建的表，在 HiveServer 客户端中不能访问，但可以使用 WebHCat 客户端访问。
- 在普通模式下使用 HCatalog 客户端，系统将以当前登录操作系统用户来执行 DDL 命令。
- 退出 beeline 客户端时请使用**!q** 命令，不要使用“Ctrl + c”。否则会导致连接生成的临时文件无法删除，长期会累积产生大量的垃圾文件。
- 在使用 beeline 客户端时，如果需要在一行中输入多条语句，语句之间以“;”分隔，需要将“entireLineAsCommand”的值设置为“false”。

  设置方法：如果未启动 beeline，则执行 **beeline --entireLineAsCommand=false** 命令；如果已启动 beeline，则在 beeline 中执行**!set entireLineAsCommand false** 命令。

  设置完成后，如果语句中含有不是表示语句结束的“;”，需要进行转义，例如 **select concat_ws('\;', collect_set(col1)) from tbl**。

----结束

使用 Hive 客户端（MRS 3.x 及之后版本）

步骤 1 以客户端安装用户，登录安装客户端的节点。

步骤 2 执行以下命令，切换到客户端安装目录。

cd /opt/hadoopclient

步骤 3 执行以下命令配置环境变量。

source bigdata_env

步骤 4 MRS 3.X 支持 Hive 多实例，若安装了 Hive 多实例，在使用客户端连接具体 Hive 实例时，请执行以下命令加载具体实例的环境变量，否则请跳过此步骤。例如，加载 Hive2 实例变量：

source Hive2/component_env

步骤 5 根据集群认证模式，完成 Hive 客户端登录。

- 安全模式，则执行以下命令，完成用户认证并登录 Hive 客户端。

  kinit *组件业务用户*

  beeline
- 普通模式，则执行以下命令，登录 Hive 客户端，如果不指定组件业务用户，则会以当前操作系统用户登录。

  beeline -n *组件业务用户*

步骤 6 使用以下命令，执行 HCatalog 的客户端命令。

hcat -e *"cmd"*

其中*"cmd"*必须为 Hive DDL 语句，如 **hcat -e "show tables"**。

说明

- 若要使用 HCatalog 客户端，必须从服务页面选择“更多 > 下载客户端”，下载全部服务的客户端。Beeline 客户端不受此限制。
- 由于权限模型不兼容，使用 HCatalog 客户端创建的表，在 HiveServer 客户端中不能访问，但可以使用 WebHCat 客户端访问。
- 在普通模式下使用 HCatalog 客户端，系统将以当前登录操作系统用户来执行 DDL 命令。
- 退出 beeline 客户端时请使用**!q** 命令，不要使用“Ctrl + C”。否则会导致连接生成的临时文件无法删除，长期会累积产生大量的垃圾文件。
- 在使用 beeline 客户端时，如果需要在一行中输入多条语句，语句之间以“;”分隔，需要将“entireLineAsCommand”的值设置为“false”。

  设置方法：如果未启动 beeline，则执行 **beeline --entireLineAsCommand=false** 命令；如果已启动 beeline，则在 beeline 中执行**!set entireLineAsCommand false** 命令。

  设置完成后，如果语句中含有不是表示语句结束的“;”，需要进行转义，例如 **select concat_ws('\;', collect_set(col1)) from tbl。**

----结束

Hive 客户端常用命令

常用的 Hive Beeline 客户端命令如下表所示。

更多命令可参考 https://cwiki.apache.org/confluence/display/Hive/HiveServer2+Clients#HiveServer2Clients-BeelineCommands。

表10-9 Hive Beeline 客户端常用命令

命令	说明
set <key>=<value>	设置特定配置变量（键）的值。 说明 若变量名拼错，Beeline 不会显示错误。
set	打印由用户或 Hive 覆盖的配置变量列表。
set -v	打印 Hadoop 和 Hive 的所有配置变量。
add FILE[S] <filepath> <filepath>*add JAR[S] <filepath> <filepath>*add ARCHIVE[S] <filepath> <filepath>*	将一个或多个文件、JAR 文件或 ARCHIVE 文件添加至分布式缓存的资源列表中。
add FILE[S] <ivyurl> <ivyurl>* add JAR[S] <ivyurl> <ivyurl>*	使用“ivy://goup:module:version?query_string”格式的 lvy URL，将一个或多个文件、JAR 文件或 ARCHIVE 文件添加至分布式缓存的资源列表中。

命令	说明
add ARCHIVE[S] <ivyurl> <ivyurl>*	
list FILE[S]list JAR[S]list ARCHIVE[S]	列出已添加至分布式缓存中的资源。
list FILE[S] <filepath>*list JAR[S] <filepath>*list ARCHIVE[S] <filepath>*	检查给定的资源是否已添加至分布式缓存中。
delete FILE[S] <filepath>*delete JAR[S] <filepath>*delete ARCHIVE[S] <filepath>*	从分布式缓存中删除资源。
delete FILE[S] <ivyurl> <ivyurl>* delete JAR[S] <ivyurl> <ivyurl>* delete ARCHIVE[S] <ivyurl> <ivyurl>*	从分布式缓存中删除使用<ivyurl>添加的资源。
reload	使 HiveServer2 发现配置参数指定路径下 JAR 文件的变更“hive.reloadable.aux.jars.path”（无需重启 HiveServer2）。更改操作包括添加、删除或更新 JAR 文件。
dfs <dfs command>	执行 dfs 命令。
<query string>	执行 Hive 查询，并将结果打印到标准输出。

10.6 使用 HDFS Colocation 存储 Hive 表

操作场景

HDFS Colocation（同分布）是 HDFS 提供的数据分布控制功能，利用 HDFS Colocation 接口，可以将存在关联关系或者可能进行关联操作的数据存放在相同的存储节点上。Hive 支持 HDFS 的 Colocation 功能，即在创建 Hive 表时，设置表文件分布的 locator 信息，当使用 insert 语句向该表中插入数据时会将该表的数据文件存放在相同的存储节点上（不支持其他数据导入方式），从而使后续的多表关联的数据计算更加方便和高效。表格式只支持 TextFile 和 RCFile。

说明

本章节适用于 MRS 3.x 及后续版本。

操作步骤

步骤 1 使用客户端安装用户登录客户端所在节点。

步骤 2 执行以下命令，切换到客户端安装目录，如：opt/client。

cd /opt/client

步骤 3 执行以下命令配置环境变量。

source bigdata_env

步骤 4 若集群为安全模式，执行以下命令认证用户。

kinit *MRS 用户名*

步骤 5 通过 HDFS 接口创建<*groupid*>

hdfs colocationadmin -createGroup -groupId <*groupid*> -locatorIds <*locatorid1*>,<*locatorid2*>,<*locatorid3*>

说明

其中<*groupid*>为创建的 group 名称，该示例语句创建的 group 包含三个 locator，用户可以根据需要定义 locator 的数量。

关于 hdfs 创建 groupid，以及 HDFS Colocation 的详细介绍请参考 hdfs 的相关说明，这里不做赘述。

步骤 6 执行以下命令进入 Hive 客户端：

beeline

步骤 7 Hive 使用 colocation。

假设 table_name1 和 table_name2 是相关联的两张表，创建两表的语句如下：

CREATE TABLE <*[db_name.]table_name1*>*[(col_name data_type , ...)]* [ROW FORMAT <*row_format*>] [STORED AS <*file_format*>] TBLPROPERTIES("groupId"=" <*group*> ","locatorId"="<*locator1*>");

CREATE TABLE <*[db_name.]table_name2*> *[(col_name data_type , ...)]* [ROW FORMAT <*row_format*>] [STORED AS <*file_format*>] TBLPROPERTIES("groupId"=" <*group*> ","locatorId"="<*locator1*>");

当使用 insert 语句分别向 table_name1 和 table_name2 插入数据后，table_name1 和 table_name2 的数据文件就会分布在 hdfs 的相同存储位置上，从而方便两表进行关联操作。

----结束

10.7 使用 Hive 列加密功能

操作场景

Hive 支持对表的某一列或者多列进行加密；在创建 Hive 表时，可以指定要加密的列和加密算法。当使用 insert 语句向表中插入数据时，即可实现将对应列加密。列加密只支持存储在 HDFS 上的 TextFile 和 SequenceFile 文件格式的表。Hive 列加密不支持视图以及 Hive over HBase 场景。

Hive 列加密机制目前支持的加密算法有两种，在建表时指定：

- AES(对应加密类名称为：org.apache.hadoop.hive.serde2.AESRewriter)
- SMS4(对应加密类名称为：org.apache.hadoop.hive.serde2.SMS4Rewriter)

说明

- 国密集群场景下，Hive 列加密只支持创建 SMS4 算法的表，不支持创建 AES 算法类型的表。
- 将原始数据从普通 Hive 表导入到 Hive 列加密表后，在不影响其他业务情况下，建议删除普通 Hive 表上原始数据，因为保留一张未加密的表存在安全风险。

操作步骤

步骤 1 在创建表时指定相应的加密列和加密算法：

create table<*[db_name.]table_name*> (<*col_name1*> <*data_type*> ,<*col_name2*> <*data_type*>,<*col_name3*> <*data_type*>,<*col_name4*> <*data_type*>) ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe' WITH SERDEPROPERTIES ('column.encode.columns'='<*col_name2*>,<*col_name3*>', 'column.encode.classname'='org.apache.hadoop.hive.serde2.AESRewriter')STORED AS TEXTFILE;

或者使用如下语句：

create table <*[db_name.]table_name*> (<*col_name1*> <*data_type*> ,<*col_name2*> <*data_type*>,<*col_name3*> <*data_type*>,<*col_name4*> <*data_type*>) ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe' WITH SERDEPROPERTIES ('column.encode.indices'='1,2', 'column.encode.classname'='org.apache.hadoop.hive.serde2.SMS4Rewriter') STORED AS TEXTFILE;

说明

- 使用序号指定加密列时，序号从 0 开始。0 代表第 1 列，1 代表第 2 列，依次类推。
- 创建列加密表时，表所在的目录必须是空目录。

步骤 2 使用 insert 语法向设置列加密的表中导入数据。

假设 test 表已存在且有数据：

insert into table <*table_name*> select <*col_list*> from test;

----结束

10.8 自定义行分隔符

操作场景

通常情况下，Hive 以文本文件存储的表会以回车作为其行分隔符，即在查询过程中，以回车符作为一行表数据的结束符。但某些数据文件并不是以回车分隔的规则文本格式，而是以某些特殊符号分割其规则文本。

MRS Hive 支持指定不同的字符或字符组合作为 Hive 文本数据的行分隔符，即在创建表的时候，指定 inputformat 为 SpecifiedDelimiterInputFormat，然后在每次查询前，都设置如下参数来指定分隔符，就可以以指定的分隔符查询表数据。

set hive.textinput.record.delimiter='';

说明

- 当前版本的 Hue 组件，不支持导入文件到 Hive 表时设置多个分割符。
- 本章节适用于 MRS 3.x 及后续版本。

操作步骤

步骤 1 创建表时指定 inputFormat 和 outputFormat：

CREATE [TEMPORARY] [EXTERNAL] TABLE [IF NOT EXISTS] *[db_name.]table_name* **[(***col_name data_type* **[COMMENT** *col_comment***], ...)] [ROW FORMAT** *row_format***] STORED AS inputformat 'org.apache.hadoop.hive.contrib.fileformat.SpecifiedDelimiterInputFormat' outputformat 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'**

步骤 2 查询之前指定配置项：

set hive.textinput.record.delimiter='!@!'

Hive 会以‘!@!’为行分隔符查询数据。

----结束

10.9 配置跨集群互信下 Hive on HBase

两个开启 Kerberos 认证的互信集群中，使用 Hive 集群操作 HBase 集群，将目的端 HBase 集群的 HBase 关键配置项配置到源端 Hive 集群的 HiveServer 中。

前提条件

两个开启 Kerberos 认证的安全集群已完成跨集群互信配置。

跨集群配置 Hive on HBase

步骤 1 下载 HBase 配置文件到本地，并解压。

1. 登录目的端 HBase 集群的 FusionInsight Manager，选择“集群 > 服务 > HBase”。
2. 选择“更多 > 下载客户端”。
3. 下载 HBase 配置文件，客户端类型选择仅配置文件。

步骤 2 登录源端 Hive 集群的 FusionInsight Manager。

步骤 3 选择“集群 > 服务 > Hive > 配置 > 全部配置”进入 Hive 服务配置页面，修改 HiveServer 角色的 hive-site.xml 自定义配置文件，增加 HBase 配置文件的如下配置项。

从已下载的 HBase 客户端配置文件的 hbase-site.xml 中，搜索并添加如下配置项及其取值到 HiveServer 中。

- hbase.security.authentication
- hbase.security.authorization
- hbase.zookeeper.property.clientPort
- hbase.zookeeper.quorum（域名需要转换为 IP）
- hbase.regionserver.kerberos.principal
- hbase.master.kerberos.principal

步骤 4 保存配置并重启 Hive 服务。

----结束

10.10 删除 Hive on HBase 表中的单行记录

操作场景

由于底层存储系统的原因，Hive 并不能支持对单条表数据进行删除操作，但在 Hive on HBase 功能中，MRS Hive 提供了对 HBase 表的单条数据的删除功能，通过特定的语法，Hive 可以将自己的 HBase 表中符合条件的一条或者多条数据清除。

表10-10 删除 Hive on HBase 表中的单行记录所需权限

集群认证模式	用户所需权限
安全模式	“SELECT”、“INSERT”和“DELETE”
普通模式	无

操作步骤

步骤 1 如果要删除某张 HBase 表中的某些数据，可以执行 HQL 语句：

remove table <*table_name*> where <*expression*>;

其中<*expression*>规定要删除数据的筛选条件；<*table_name*>为要删除数据的 Hive on HBase 表。

----结束

10.11 配置基于 HTTPS/HTTP 协议的 REST 接口

操作场景

WebHCat 为 Hive 提供了对外可用的 REST 接口，开源社区版本默认使用 HTTP 协议。

MRS Hive 支持使用更安全的 HTTPS 协议，并且可以在两种协议间自由切换。

说明

安全模式支持 HTTPS 和 HTTP 协议，普通模式只支持 HTTP 协议。

操作步骤

步骤 1 进入 Hive 服务配置页面：

- MRS 3.x 之前版本，单击集群名称，登录集群详情页面，选择“组件管理 > Hive > 服务配置”，单击“基础配置”下拉菜单，选择“全部配置”。

  说明

  若集群详情页面没有“组件管理”页签，请先完成 IAM 用户同步（在集群详情页的“概览”页签，单击“IAM 用户同步”右侧的“同步”进行 IAM 用户同步）。

- MRS 3.x 及后续版本，登录 FusionInsight Manager，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本）。然后选择“集群 > *待操作的集群名称* > 服务 > Hive > 配置 > 全部配置”。

步骤 2 修改 Hive 配置：

- MRS 3.x 之前版本：在搜索框中输入参数名称，搜索“templeton.protocol.type”，修改参数值为 HTTPS 或者 HTTP，修改后重启 Hive 服务即可使用对应的协议。
- MRS 3.x 及后续版本：选择“WebHCat > 安全”，在该界面选择 HTTPS 或者 HTTP，修改后重启 Hive 服务即可使用对应的协议。

----结束

10.12 配置是否禁用 Transform 功能

操作场景

Hive 开源社区版本禁用 Transform 功能。

MRS Hive 提供配置开关，默认为禁用 Transform 功能，与开源社区版本保持一致。

用户可修改配置开关，开启 Transform 功能，当开启 Transform 功能时，存在一定的安全风险。

说明

只有安全模式支持禁用 Transform 功能，普通模式不支持该功能。

操作步骤

步骤 1 进入 Hive 服务配置页面：

- MRS 3.x 之前版本，单击集群名称，登录集群详情页面，选择“组件管理 > Hive > 服务配置”，单击“基础配置”下拉菜单，选择“全部配置”。

  说明

  若集群详情页面没有“组件管理”页签，请先完成 IAM 用户同步（在集群详情页的“概览”页签，单击“IAM 用户同步”右侧的“同步”进行 IAM 用户同步）。

- MRS 3.x 及后续版本，登录 FusionInsight Manager，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本）。然后选择“集群 > *待操作的集群名*称 > 服务 > Hive > 配置 > 全部配置”。

步骤 2 在搜索框中输入参数名称，搜索“hive.security.transform.disallow”，修改参数值为“true”或“false”，修改后重启所有 HiveServer 实例。

说明

- 选择“true”时，禁用 Transform 功能，与开源社区版本保持一致。
- 选择“false”时，开启 Transform 功能，存在一定的安全风险。

----结束

10.13 Hive 支持创建单表动态视图授权访问控制

操作场景

MRS 中安全模式下 Hive 可以创建一个视图并控制用户访问权限，支持授权给不同的用户访问，又可以限定不同用户只能访问的不同数据。

在视图中，Hive 可以通过获取当前客户端提交任务的用户的内置函数“current_user()”来进行过滤，这样被授权的用户，在访问视图时，即可被限定访问对应的数据。

说明

- 在普通模式下“current_user()”函数无法区别客户端提交任务的用户，因此，当前访问控制仅对安全模式下的 Hive 有效。
- 如果已经在实际业务逻辑中使用了“current_user()”函数，那么，在安全模式与普通模式互转时，需要充分评估可能的风险。

操作示例

- 不采用“current_user”函数，我们要实现不同的用户，访问不同数据，需要创建不同的视图：
  - 将视图 v1 授权给用户 hiveuser1，hiveuser1 用户可以访问表 table1 中“type='hiveuser1'”的数据：

    create view v1 as select * from table1 where type='hiveuser1'
  - 将视图 v2 授权给用户 hiveuser2，hiveuser2 用户可以访问表 table1 中“type='hiveuser2'”的数据：

    create view v2 as select * from table1 where type='hiveuser2'
- 采用“current_user”函数，则只需要创建一个视图：

  将视图 v 分别赋给用户 hiveuser1、hiveuser2，当 hiveuser1 查询视图 v 时，“current_user()”被自动转化为 hiveuser1，当 hiveuser2 查询视图 v 时，“current_user()”被自动转化为 hiveuser2：

  create view v as select * from table1 where type=current_user()

10.14 配置创建临时函数是否需要 ADMIN 权限

操作场景

Hive 开源社区版本创建临时函数需要用户具备 ADMIN 权限。

MRS Hive 提供配置开关，默认为创建临时函数需要 ADMIN 权限，与开源社区版本保持一致。

用户可修改配置开关，实现创建临时函数不需要 ADMIN 权限。当该选项配置成 false 时，存在一定的安全风险。

说明

安全模式支持配置创建临时函数是否需要 ADMIN 权限功能，而普通模式不支持该功能。

操作步骤

步骤 1 进入 Hive 服务配置页面：

- MRS 3.x 之前版本，单击集群名称，登录集群详情页面，选择“组件管理 > Hive > 服务配置”，单击“基础配置”下拉菜单，选择“全部配置”。

  说明

  若集群详情页面没有“组件管理”页签，请先完成 IAM 用户同步（在集群详情页的“概览”页签，单击“IAM 用户同步”右侧的“同步”进行 IAM 用户同步）。
- MRS 3.x 及后续版本，登录 FusionInsight Manager，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本）。然后选择“集群 > *待操作的集群名称* > 服务 > Hive > 配置 > 全部配置”。

步骤 2 在搜索框中输入参数名称，搜索“hive.security.temporary.function.need.admin”，修改参数值为“true”或“false”，修改后重启所有 HiveServer 实例。

说明

- 选择“true”时，创建临时函数需要 ADMIN 权限，与开源社区版本保持一致。
- 选择“false”时，创建临时函数不需要 ADMIN 权限。

----结束

10.15 使用 Hive 读取关系型数据库数据

操作场景

Hive 支持创建与其他关系型数据库关联的外表。该外表可以从关联到的关系型数据库中读取数据，并与 Hive 的其他表进行 Join 操作。

目前支持使用 Hive 读取数据的关系型数据库如下：

- DB2
- Oracle

说明

本章节适用于 MRS 3.x 及后续版本。

前提条件

已安装 Hive 客户端。

操作步骤

步骤 1 以 Hive 客户端安装用户登录安装客户端的节点。

步骤 2 执行以下命令，切换到客户端安装目录。

cd *客户端安装目录*

例如安装目录为“/opt/client”，则执行以下命令：

cd /opt/client

步骤 3 执行以下命令配置环境变量。

source bigdata_env

步骤 4 集群认证模式是否为安全模式。

- 是，执行以下命令进行用户认证：

  kinit *Hive 业务用户*

- 否，执行步骤 5。

步骤 5 执行以下命令，将需要关联的关系型数据库驱动 Jar 包上传到 HDFS 目录下。

hdfs dfs -put *Jar 包所在目录 保存 Jar 包的 HDFS 目录*

例如将“/opt”目录下 ORACLE 驱动 Jar 包上传到 HDFS 的“/tmp”目录下，则执行如下命令。

hdfs dfs -put /opt/ojdbc6.jar /tmp

步骤 6 按照如下示例，在 Hive 客户端创建关联关系型数据库的外表。

说明

如果是安全模式，建表的用户需要“ADMIN”权限，ADD JAR 的路径请以实际路径为准。

```
-- 关联 oracle linux6 版本示例
-- 如果是安全模式，设置 admin 权限
set role admin;
-- 添加连接关系型数据库的驱动 jar 包,不同数据库有不同的驱动 JAR
ADD JAR hdfs:///tmp/ojdbc6.jar;

CREATE EXTERNAL TABLE ora_test
-- hive 表的列需比数据库返回结果多一列用于分页查询
(id STRING,rownum string)
STORED BY 'com.qubitproducts.hive.storage.jdbc.JdbcStorageHandler'
TBLPROPERTIES (
-- 关系型数据库类型
"qubit.sql.database.type" = "ORACLE",
-- 通过 JDBC 连接关系型数据库的 url（不同数据库有不同的 url 格式）
"qubit.sql.jdbc.url" = "jdbc:oracle:thin:@//10.163.0.1:1521/mydb",
-- 关系型数据库驱动类名
"qubit.sql.jdbc.driver" = "oracle.jdbc.OracleDriver",
-- 在关系型数据库查询的 sql 语句,结果将返回 hive 表
"qubit.sql.query" = "select name from aaa",
-- hive 表的列与关系型数据库表的列进行匹配（可忽略）
"qubit.sql.column.mapping" = "id=name",
-- 关系型数据库用户
"qubit.sql.dbcp.username" = "test",
-- 关系型数据库密码
"qubit.sql.dbcp.password" = "xxx");
```

----结束

10.16 Hive 支持的传统关系型数据库语法

概述

Hive 支持如下传统关系型数据库语法：

- Grouping
- EXCEPT、INTERSECT

Grouping

语法简介：

- 当 Group by 语句带 with rollup/cube 选项时，Grouping 才有意义。

- CUBE 生成的结果集显示了所选列中值的所有组合的聚合。
- ROLLUP 生成的结果集显示了所选列中值的某一层次结构的聚合。
- Grouping：当用 CUBE 或 ROLLUP 运算符添加行时，附加的列输出值为 1；当所添加的行不是由 CUBE 或 ROLLUP 产生时，附加列值为 0。

例如，Hive 中有一张表“table_test”，表结构如下所示：

```
+----------------+-------------------+--+
| table_test.id  | table_test.value  |
+----------------+-------------------+--+
| 1              | 10                |
| 1              | 15                |
| 2              | 20                |
| 2              | 5                 |
| 2              | 13                |
+----------------+-------------------+--+
```

执行如下语句：

select id,grouping(id),sum(value) from table_test group by id with rollup;

得到如下结果：

```
+-------+-----------------+------+--+
|  id   | groupingresult  | sum  |
+-------+-----------------+------+--+
| 1     | 0               | 25   |
| NULL  | 1               | 63   |
| 2     | 0               | 38   |
+-------+-----------------+------+--+
```

EXCEPT、INTERSECT

语法简介

- EXCEPT 返回两个结果集的差（即从左查询中返回右查询没有找到的所有非重复值）。
- INTERSECT 返回两个结果集的交集（即两个查询都返回的所有非重复值）。

例如，Hive 中有两张表“test_table1”、“test_table2”。

“test_table1”表结构如下所示：

```
+------------------+--+
| test_table1.id   |
+------------------+--+
| 1                |
| 2                |
| 3                |
| 4                |
+------------------+--+
```

“test_table2”表结构如下所示：

```
+------------------+--+
| test_table2.id   |
```

```
+------------------+--+
| 2                |
| 3                |
| 4                |
| 5                |
+------------------+--+
```

- 执行如下的 EXCEPT 语句：

  select id from test_table1 except select id from test_table2;

  显示如下结果：

  ```
  +---------------+--+
  | _alias_0.id  |
  +---------------+--+
  | 1           |
  +---------------+--+
  ```

- 执行 INTERSECT 语句：

  select id from test_table1 intersect select id from test_table2;

  显示如下结果：

  ```
  +---------------+--+
  | _alias_0.id  |
  +---------------+--+
  | 2           |
  | 3           |
  | 4           |
  +---------------+--+
  ```

10.17 创建 Hive 用户自定义函数

当 Hive 的内置函数不能满足需要时，可以通过编写用户自定义函数 UDF（User-Defined Functions）插入自己的处理代码并在查询中使用它们。

按实现方式，UDF 分如下分类：

- 普通的 UDF，用于操作单个数据行，且产生一个数据行作为输出。
- 用户定义聚集函数 UDAF（User-Defined Aggregating Functions），用于接受多个输入数据行，并产生一个输出数据行。
- 用户定义表生成函数 UDTF（User-Defined Table-Generating Functions），用于操作单个输入行，产生多个输出行。

按使用方法，UDF 有如下分类：

- 临时函数，只能在当前会话使用，重启会话后需要重新创建。
- 永久函数，可以在多个会话中使用，不需要每次创建。

说明

用户自定义函数需要用户控制函数中变量的内存、线程等资源的占用，如果控制不当可能会导致内存溢出、CPU 使用高等问题。

下面以编写一个 AddDoublesUDF 为例，说明 UDF 的编写和使用方法。

功能介绍

AddDoublesUDF 主要用来对两个及多个浮点数进行相加，在该样例中可以掌握如何编写和使用 UDF。

说明

- 一个普通 UDF 必须继承自“org.apache.hadoop.hive.ql.exec.UDF”。
- 一个普通 UDF 必须至少实现一个 evaluate()方法，evaluate 函数支持重载。
- 开发自定义函数需要在工程中添加“hive-exec-3.1.0.jar”依赖包，可从 Hive 服务的安装目录下获取。

样例代码

以下为 UDF 示例代码：

```
package com.xxx.bigdata.hive.example.udf;
import org.apache.hadoop.hive.ql.exec.UDF;

public class AddDoublesUDF extends UDF {
 public Double evaluate(Double... a) {
   Double total = 0.0;
   // 处理逻辑部分.
   for (int i = 0; i < a.length; i++)
     if (a[i] != null)
       total += a[i];
   return total;
  }
}
```

如何使用

步骤 1 在客户端安装节点，把以上程序打包成 AddDoublesUDF.jar，并上传到 HDFS 指定目录下（例如“/user/hive_examples_jars”）。

创建函数的用户与使用函数的用户都需要具有该文件的可读权限。

示例语句：

hdfs dfs -put ./hive_examples_jars /user/hive_examples_jars

hdfs dfs -chmod 777 /user/hive_examples_jars

步骤 2 判断集群的认证模式。

- 安全模式，需要使用一个具有 Hive 管理权限的用户登录 beeline 客户端，执行如下命令：

  kinit *Hive 业务用户*

  beeline

  set role admin;
- 普通模式，执行如下命令：

  beeline -n *Hive 业务用户*

步骤 3 在 Hive Server 中定义该函数，以下语句用于创建永久函数：

CREATE FUNCTION *addDoubles* **AS**
'com.xxx.bigdata.hive.example.udf.AddDoublesUDF' using jar
'hdfs://hacluster/user/hive_examples_jars/AddDoublesUDF.jar';

其中 *addDoubles* 是该函数的别名，用于 SELECT 查询中使用。

以下语句用于创建临时函数：

CREATE TEMPORARY FUNCTION addDoubles AS
'com.xxx.bigdata.hive.example.udf.AddDoublesUDF' using jar
'hdfs://hacluster/user/hive_examples_jars/AddDoublesUDF.jar';

- *addDoubles* 是该函数的别名，用于 SELECT 查询中使用。
- 关键字 TEMPORARY 说明该函数只在当前这个 Hive Server 的会话过程中定义使用。

步骤 4 在 Hive Server 中使用该函数，执行 SQL 语句：

SELECT addDoubles(1,2,3);

说明

若重新连接客户端再使用函数出现[Error 10011]的错误，可执行 **reload function;**命令后再使用该函数。

步骤 5 在 Hive Server 中删除该函数，执行 SQL 语句：

DROP FUNCTION addDoubles;

----结束

扩展应用

无

10.18 beeline 可靠性增强特性介绍

操作场景

- 在批处理任务运行过程中，beeline 客户端由于网络异常等问题断线时，Hive 能支持 beeline 在断线前已经提交的任务继续运行。当再次运行该批处理任务时，已经提交过的任务不再重新执行，直接从下一个任务开始执行。
- 在批处理任务运行过程中，HiveServer 服务由于某些原因导致宕机时，Hive 能支持当再次运行该批处理任务时，已经成功执行完成的任务不再重新执行，直接从 HiveServer2 宕机时正在运行的任务开始运行。

说明

本章节适用于 MRS 3.x 及后续版本。

操作示例

1. beeline 启动断线重连功能。

   示例：

   beeline -e "${*SQL*}" --hivevar batchid=*xxxxx*

2. beeline kill 正在运行的任务。

   示例：

   beeline -e "" --hivevar batchid=*xxxxx* --hivevar kill=true

3. 登录 beeline 客户端，启动断线重连机制。

   登录 beeline 客户端后，执行“set hivevar:batchid=*xxxx*”

   说明

   使用说明：

   - 其中“*xxxx*”表示每一次通过 beeline 提交任务的批次号，通过该批次号，可以识别出先提交的任务。如果提交任务时不带批次号，该特性功能不会启用。“*xxxx*”的值是执行任务时指定的，如下所示，“*xxxx*”值为“012345678901”：

     beeline -f hdfs://hacluster/user/hive/table.sql --hivevar batchid=012345678901

   - 如果运行的 SQL 脚本依赖数据的失效性，建议不启用断点重连机制，或者每次运行时使用新的 batchid。因为重复执行时，可能由于某些 SQL 语句已经执行过了不再重新执行，导致获取到过期的数据。
   - 如果 SQL 脚本中使用了一些内置时间函数，建议不启用断点重连机制，或者每次运行时使用新的 batchid，理由同上。
   - 一个 SQL 脚本里面会包含一个或多个子任务。如果 SQL 脚本中存在先创建再删除临时表的逻辑，建议将删除临时表的逻辑放到脚本的最后。假定删除临时表子任务的后续子任务执行失败，并且删除临时表的子任务之前的子任务用到了该临时表；当下一次以相同 batchid 执行该 SQL 脚本时，因为临时表在上一次执行时已被删除，则会导致删除临时表的子任务之前用到该临时表的子任务（不包括创建该临时表的子任务，因为上一次已经执行成功，本次不会再执行，仅可编译）编译失败。这种情况下，建议使用新的 batchid 执行脚本。

   参数说明：

   - zk.cleanup.finished.job.interval：执行清理任务的间隔时间，默认隔 60s 执行一次。
   - zk.cleanup.finished.job.outdated.threshold：节点的过期时间，每个批次的任务都会生成对应节点，从当前批次任务的结束时间开始算，如果超过 60 分钟，则表示已经过期了，那么就清除节点。
   - batch.job.max.retry.count：单批次任务的最大重试次数，当单批次的任务失败重试次数超过这个值，就会删除该任务记录，下次运行时将从头开始运行，默认是 10 次。
   - beeline.reconnect.zk.path：存储任务执行进度的根节点，Hive 服务默认是/beeline。

10.19 具备表 select 权限可用 show create table 查看表结构

操作场景

此功能在 MRS 3.x 及后续版本适用于 Hive，Spark2x。

开启此功能后，使用 Hive 建表时，其他用户被授予 select 权限后，可通过 **show create table** 查看表结构。

操作步骤

步骤 1 进入 Hive 服务配置页面：

- MRS 3.x 之前版本，单击集群名称，登录集群详情页面，选择“组件管理 > Hive > 服务配置”，单击“基础配置”下拉菜单，选择“全部配置”。

  说明

  若集群详情页面没有“组件管理”页签，请先完成 IAM 用户同步（在集群详情页的“概览”页签，单击“IAM 用户同步”右侧的“同步”进行 IAM 用户同步）。

- MRS 3.x 及后续版本，登录 FusionInsight Manager，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本）。然后选择“集群 > *待操作的集群名称* > 服务 > Hive > 配置 > 全部配置”。

步骤 2 选择“HiveServer（角色） > 自定义”，对参数文件“hive-site.xml”添加自定义参数，设置“名称”为“hive.allow.show.create.table.in.select.nogrant”，“值”为“true ”，修改后重启所有 Hive 实例。

步骤 3 是否需要在 Spark/Spark2x 客户端中启用此功能？

- 是，重新下载并安装 Spark/Spark2x 客户端。
- 否，操作结束。

----结束

10.20 Hive 写目录旧数据进回收站

操作场景

此功能适用于 Hive 组件。

开启此功能后，执行写目录：**insert overwrite directory** "*/path1*" **...**，写成功之后，会将旧数据移除到回收站，并且同时限制该目录不能为 Hive 元数据库中已经存在的数据库路径。

步骤 1 进入 Hive 服务配置页面：

- MRS 3.x 之前版本，单击集群名称，登录集群详情页面，选择“组件管理 > Hive > 服务配置”，单击“基础配置”下拉菜单，选择“全部配置”。

说明

若集群详情页面没有“组件管理”页签，请先完成 IAM 用户同步（在集群详情页的“概览”页签，单击“IAM 用户同步”右侧的“同步”进行 IAM 用户同步）。

- MRS 3.x 及后续版本，登录 FusionInsight Manager，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本）。然后选择“集群 > *待操作的集群名称* > 服务 > Hive > 配置 > 全部配置”。

步骤 2 选择“HiveServer（角色） > 自定义”，对参数文件“hive-site.xml”添加自定义参数，设置“名称”为“hive.overwrite.directory.move.trash”，“值”为“true”，修改后重启所有 Hive 实例。

----结束

10.21 Hive 能给一个不存在的目录插入数据

操作场景

此功能适用于 Hive 组件。

开启此功能后，在执行写目录：**insert overwrite directory** *“/path1/path2/path3”* ... 时，其中“*/path1/path2*”目录权限为 700 且属主为当前用户，“*path3*”目录不存在，会自动创建“*path3*”目录，并写数据成功。

上述功能，在 Hive 参数“hive.server2.enable.doAs”为“true”时已经支持，本次增加当“hive.server2.enable.doAs”为“false”时的功能支持。

说明

本功能参数调整与 Hive 写目录旧数据进回收站添加的自定义参数相同。

操作步骤

步骤 1 进入 Hive 服务配置页面：

- MRS 3.x 之前版本，单击集群名称，登录集群详情页面，选择“组件管理 > Hive > 服务配置”，单击“基础配置”下拉菜单，选择“全部配置”。

说明

若集群详情页面没有“组件管理”页签，请先完成 IAM 用户同步（在集群详情页的“概览”页签，单击“IAM 用户同步”右侧的“同步”进行 IAM 用户同步）。

- MRS 3.x 及后续版本，登录 FusionInsight Manager，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本）。然后选择“集群 > *待操作的集群名称* > 服务 > Hive > 配置 > 全部配置”。

步骤 2 选择“HiveServer（角色） > 自定义”，对参数文件“hive-site.xml”添加自定义参数，设置“名称”为“hive.overwrite.directory.move.trash”，“值”为“true”，修改后重启所有 Hive 实例。

----结束

10.22 限定仅 admin 用户能创建库和在 default 库建表

操作场景

此功能在 MRS 3.x 之前版本适用于 Hive，Spark。在 MRS 3.x 及后续版本适用于 Hive，Spark2x。

开启此功能后，仅有 Hive 管理员可以创建库和在 default 库中建表，其他用户需通过 Hive 管理员授权才可使用库。

说明

- 开启本功能之后，会限制普通用户新建库和在 default 库新建表。请充分考虑实际应用场景，再决定是否作出调整。
- 因为对执行用户做了限制，使用非管理员用户执行建库、表脚本迁移、重建元数据操作时需要特别注意，防止错误。

操作步骤

步骤 1 进入 Hive 服务配置页面：

- MRS 3.x 之前版本，单击集群名称，登录集群详情页面，选择“组件管理 > Hive > 服务配置”，单击“基础配置”下拉菜单，选择“全部配置”。

  说明

  若集群详情页面没有“组件管理”页签，请先完成 IAM 用户同步（在集群详情页的“概览”页签，单击“IAM 用户同步”右侧的“同步”进行 IAM 用户同步）。

- MRS 3.x 及后续版本，登录 FusionInsight Manager，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本）。然后选择“集群 > *待操作的集群名称* > 服务 > Hive > 配置 > 全部配置”。

步骤 2 选择“HiveServer（角色） > 自定义”，对参数文件“hive-site.xml”添加自定义参数，设置“名称”为“hive.allow.only.admin.create”，“值”为“true”，修改后重启所有 Hive 实例。

步骤 3 是否需要在 Spark/Spark2x 客户端中启用此功能？

- 是，执行步骤 4。
- 否，操作结束。

步骤 4 选择“SparkResource2x > 自定义”和“JDBCServer2x > 自定义”，对参数文件“hive-site.xml”添加自定义参数，设置“名称”为“hive.allow.only.admin.create”，“值”为“true”，修改后重启所有 Spark2x 实例。

步骤 5 重新下载并安装 Spark/Spark2x 客户端。

----结束

10.23 限定创建 Hive 内部表不能指定 location

操作场景

此功能在 MRS 3.x 之前版本适用于 Hive，Spark。在 MRS 3.x 及后续版本适用于 Hive，Spark2x。

开启此功能后，在创建 Hive 内部表时，不能指定 location。即表创建成功之后，表的 location 路径会被创建在当前默认 warehouse 目录下，不能被指定到其他目录。如果创建内部表时指定 location，则创建失败。

说明

开启本功能之后，创建 Hive 内部表不能执行 location。因为对建表语句做了限制，如果数据库中已存在建表时指向非当前默认 warehouse 目录的表，在执行建库、表脚本迁移、重建元数据操作时需要特别注意，防止错误。

操作步骤

步骤 1 进入 Hive 服务配置页面：

- MRS 3.x 之前版本，单击集群名称，登录集群详情页面，选择“组件管理 > Hive > 服务配置”，单击“基础配置”下拉菜单，选择“全部配置”。

  说明

  若集群详情页面没有“组件管理”页签，请先完成 IAM 用户同步（在集群详情页的“概览”页签，单击“IAM 用户同步”右侧的“同步”进行 IAM 用户同步）。

- MRS 3.x 及后续版本，登录 FusionInsight Manager，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本）。然后选择“集群 > *待操作的集群名称* > 服务 > Hive > 配置 > 全部配置”。

步骤 2 选择“HiveServer（角色） > 自定义”，对参数文件“hive-site.xml”添加自定义参数，设置“名称”为“hive.internaltable.notallowlocation”，“值”为“true”，修改后重启所有 Hive 实例。

步骤 3 是否需要在 Spark/Spark2x 客户端中启用此功能？

- 是，重新下载并安装 Spark/Spark2x 客户端。
- 否，操作结束。

----结束

10.24 允许在只读权限的目录建外表

操作场景

此功能在 MRS 3.x 之前版本适用于 Hive，Spark。在 MRS 3.x 及后续版本适用于 Hive，Spark2x。

开启此功能后，允许有目录读权限和执行权限的用户和用户组创建外部表，而不必检查用户是否为该目录的属主，并且禁止外表的 location 目录在当前默认 warehouse 目录下。同时在外表授权时，禁止更改其 location 目录对应的权限。

说明

开启本功能之后，外表功能变化大。请充分考虑实际应用场景，再决定是否作出调整。

操作步骤

步骤 1 进入 Hive 服务配置页面：

- MRS 3.x 之前版本，单击集群名称，登录集群详情页面，选择“组件管理 > Hive > 服务配置”，单击“基础配置”下拉菜单，选择“全部配置”。

  说明

  若集群详情页面没有“组件管理”页签，请先完成 IAM 用户同步（在集群详情页的“概览”页签，单击“IAM 用户同步”右侧的“同步”进行 IAM 用户同步）。

- MRS 3.x 及后续版本，登录 FusionInsight Manager，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本）。然后选择“集群 > *待操作的集群名*称 > 服务 > Hive > 配置 > 全部配置”。

步骤 2 选择“HiveServer（角色） > 自定义”，对参数文件“hive-site.xml”添加自定义参数，设置“名称”为“hive.restrict.create.grant.external.table”，“值”为“true”。

步骤 3 选择“MetaStore（角色） > 自定义”，对参数文件“hivemetastore-site.xml”添加自定义参数，设置“名称”为“hive.restrict.create.grant.external.table”，“值”为“true”，修改后重启所有 Hive 实例。

步骤 4 是否需要在 Spark/Spark2x 客户端中启用此功能？

- 是，重新下载并安装 Spark/Spark2x 客户端。
- 否，操作结束。

----结束

10.25 Hive 支持授权超过 32 个角色

操作场景

此功能适用于 Hive。

因为操作系统用户组个数限制，导致 Hive 不能创建超过 32 个角色，开启此功能后，Hive 将支持创建超过 32 个角色。

说明

- 开启本功能并对表库等授权后，对表库目录具有相同权限的角色将会用“|”合并。查询 acl 权限时，将显示合并后的结果，与开启该功能前的显示会有区别。此操作不可逆，请充分考虑实际应用场景，再决定是否作出调整。

- MRS 3.x 及后续版本支持 Ranger，如果当前组件使用了 Ranger 进行权限控制，需基于 Ranger 配置相关策略进行权限管理，具体操作可参考添加 Hive 的 Ranger 访问权限策略。
- 开启此功能后，包括 **owner** 在内默认最大可支持 512 个角色，由 MetaStore 自定义参数“hive.supports.roles.max”控制，可考虑实际应用场景进行修改。

操作步骤

步骤 1 进入 Hive 服务配置页面：

- MRS 3.x 之前版本，单击集群名称，登录集群详情页面，选择“组件管理 > Hive > 服务配置”，单击“基础配置”下拉菜单，选择“全部配置”。

  说明

  若集群详情页面没有“组件管理”页签，请先完成 IAM 用户同步（在集群详情页的“概览”页签，单击“IAM 用户同步”右侧的“同步”进行 IAM 用户同步）。

- MRS 3.x 及后续版本，登录 FusionInsight Manager，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本）。然后选择“集群 > *待操作的集群名*称 > 服务 > Hive > 配置 > 全部配置”。

步骤 2 选择“MetaStore（角色） > 自定义”，对参数文件“hivemetastore-site.xml”添加自定义参数，设置“名称”为“hive.supports.over.32.roles”，“值”为“true”。

步骤 3 选择“HiveServer（角色） > 自定义”，对参数文件“hive-site.xml”添加自定义参数，设置“名称”为“hive.supports.over.32.roles”，“值”为“true”，修改后重启所有 Hive 实例。

----结束

10.26 Hive 任务支持限定最大 map 数

操作场景

- 此功能适用于 Hive。
- 此功能用于从服务端限定 Hive 任务的最大 map 数，避免 HiveSever 服务过载而引发的性能问题。

操作步骤

步骤 1 进入 Hive 服务配置页面：

- MRS 3.x 之前版本，单击集群名称，登录集群详情页面，选择“组件管理 > Hive > 服务配置”，单击“基础配置”下拉菜单，选择“全部配置”。

  说明

  若集群详情页面没有“组件管理”页签，请先完成 IAM 用户同步（在集群详情页的“概览”页签，单击“IAM 用户同步”右侧的“同步”进行 IAM 用户同步）。

- MRS 3.x 及后续版本，登录 FusionInsight Manager，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本）。然后选择“集群 > *待操作的集群名称* > 服务 > Hive > 配置 > 全部配置”。

步骤 2 选择“MetaStore（角色） > 自定义”，对参数文件“hivemetastore-site.xml”添加自定义参数，设置“名称”为“hive.mapreduce.per.task.max.splits”，“值”为具体设定值，一般尽量设置大，修改后重启所有 Hive 实例。

----结束

10.27 HiveServer 租约隔离使用

操作场景

- 此功能适用于 Hive。
- 开启此功能可以限定指定用户访问指定节点上的 HiveServer 服务，实现对用户访问 HiveServer 服务的资源隔离。

说明

本章节适用于 MRS 3.x 及后续版本。

操作步骤

以对用户 hiveuser 设置租约隔离为例，选取 Hive 当前已有的或者新添加一个或者多个实例，此处选择已有的 HiveServer 实例：

步骤 1 登录 FusionInsight Manager，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本）。

步骤 2 选择“集群 > *待操作集群的名称* > 服务 > Hive > HiveServer”。

步骤 3 在 HiveServer 列表里选择设置租约隔离的 HiveServer，选择“HiveServer > 实例配置 > 全部配置”。

步骤 4 在“全部配置”界面的右上角搜索“hive.server2.zookeeper.namespace”，“值”为具体设定值，比如为 hiveserver2_zk。

步骤 5 单击“保存”，在弹出对话框单击“确定”。

步骤 6 选择“集群 > *待操作集群的名称* > 服务 > Hive”，选择“更多 > 重启服务”，输入密码开始重启服务。

步骤 7 使用 beeline -u 的方式登录客户端，执行以下命令：

beeline -u "jdbc:hive2://*10.5.159.13*:2181/;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=*hiveserver2_zk*;sasl.qop=auth-conf;auth=KERBEROS;principal=hive/hadoop.<*系统域名*>@<*系统域名*>"

执行命令时将“10.5.159.13”替换为任意一个 ZooKeeper 实例的 IP 地址，查找方式为“集群 > *待操作集群的名称* > 服务 > ZooKeeper > 实例”。

“zooKeeperNamespace=”后面的“hiveserver2_zk”为步骤 4 中参数“hive.server2.zookeeper.namespace”设置的具体设定值。

结果将只会登录到被设置租约隔离的 HiveServer。

说明

- 开启本功能后，必须在登录时使用以上命令才可以访问这个被设置租约隔离的 HiveServer。如果直接使用 **beeline** 命令登录客户端，将只会访问其他没有被设置租约隔离的 HiveServer。
- 用户可登录 FusionInsight Manager，选择“系统 > 权限 > 域和互信”，查看“本端域”参数，即为当前系统域名。“hive/hadoop.<系统域名>”为用户名，用户名所包含的系统域名所有字母为小写。

----结束

10.28 Hive 支持事务

操作场景

Hive 在表以及分区级别支持事务，开启事务模式下，能够增量更新、删除、读取事务表，实现了对事务表操作的原子性、隔离性、一致性和永久性。

说明

本章节适用于 MRS 3.x 及后续版本。

事务特性介绍

事务（transaction）是一组单元化操作，这些操作要么都执行，要么都不执行，是一个不可分割的工作单位。事务的四个基本要素通常被称为 ACID 特性，分别为：

- 原子性（Atomicity）：一个事务是一个不可再分割的工作单位，事务中的所有操作要么都发生，要么都不发生。
- 一致性（Consistency）：事务开始之前和事务结束以后，数据库的完整性约束没有被破坏。
- 隔离性（Isolation）：多个事务并发访问，事务之间是隔离的，一个事务不影响其它事务运行效果。事务之间的影响有：脏读、不可重复读、幻读、丢失更新。
- 持久性（Durability）：在事务完成以后，该事务锁对数据库所做的更改将永久保存在数据库中。

事务执行特点：

- 一条语句可以写入多个分区或多个表。如果操作失败，则用户看不到部分写入或插入。即使频繁更改数据，仍然能够快速执行操作。
- Hive 能够自动压缩 ACID 事务文件，而不会影响并发查询。当查询许多小分区文件时，自动压缩可提高查询性能和元数据占用量。
- 读取语义包括快照隔离。当读取操作开始时，Hive 在逻辑上处于锁定仓库的状态。读操作不受操作期间发生的任何更改的影响。

锁机制

事务通过以下两点实现 ACID 特性：

- 预写日志（Write-ahead logging）保证原子性和持久性。
- 锁（locking）保证隔离性。

操作	持有锁类型
Insert overwrite	hive.txn.xlock.iow=true 时持有排他锁， hive.txn.xlock.iow=false 时持有半共享锁。
Insert	共享锁。执行该操作时能够对当前表或分区执行读写操作。
Update/delete	半共享锁。执行该操作时能够执行持有共享锁的操作，不能执行持有排他锁或半共享锁的操作。
Drop	排他锁。执行该操作时无法对当前表或分区执行其他任何操作。

说明

如果写操作中存在锁机制引发的冲突，优先持有锁的操作将成功，其他操作将失败。

操作步骤

开启事务

步骤 1 登录 FusionInsight Manager 界面，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本），选择“集群 > *待操作的集群* > 服务 > Hive > 配置 > 全部配置 > MetaStore（角色） > 事务”。

步骤 2 将“metastore.compactor.initiator.on”设置为 true。

步骤 3 将“metastore.compactor.worker.threads”设置为大于 0 的正整数。

说明

“metastore.compactor.worker.threads”：在 MetaStore 上运行压缩程序工作线程个数。请根据实际业务设置合适的值，该值过小会引起事务压缩任务执行慢，过大会导致 MetaStore 执行性能变低。

步骤 4 登录 Hive 客户端，执行命令开启以下参数，具体操作请参考使用 Hive 客户端。

set hive.support.concurrency=true;

set hive.exec.dynamic.partition.mode=nonstrict;

set hive.txn.manager=org.apache.hadoop.hive.ql.lockmgr.DbTxnManager;

创建事务表

步骤 5 执行以下命令创建事务表。

CREATE TABLE [IF NOT EXISTS] *[db_name.]table_name* **(***col_name data_type [COMMENT col_comment], ...***) [ROW FORMAT row_format] STORED AS orc ..... TBLPROPERTIES ('transactional'='true'[,'groupId'='group1' ...]);**

例如：

CREATE TABLE acidTbl (a int, b int) STORED AS ORC TBLPROPERTIES ('transactional'='true');

说明

- 当前事务仅支持 orc 格式。
- 不支持外表。
- 不支持 sorted table。
- 创建事务表必须增加表属性**'transactional'='true'**。
- 只能在事务模式下读写事务表。

使用事务表

步骤 6 执行命令使用事务表。以 **acidTbl** 表为例：

- 向已有事务表中插入数据。

  INSERT INTO acidTbl VALUES(1,1);
- 更新已有事务表

  UPDATE acidTbl SET b = 10 where a = 1;

  acidTbl 内容变更为：

```
+------------+------------+
| acidtbl.a  | acidtbl.b  |
+------------+------------+
| 1          | 10         |
+------------+------------+
1 row selected (0.775 seconds)
```

- 合并新旧事务表：

  acidTbl_update 表中已有数据：

```
+--------------------+--------------------+
| acidtbl_update.a   | acidtbl_update.b   |
+--------------------+--------------------+
| 1                  | 20                 |
| 2                  | 10                 |
+--------------------+--------------------+
2 rows selected (0.537 seconds)
```

  MERGE INTO acidTbl AS a

  USING acidTbl_update AS b ON a.a = b.a

  WHEN MATCHED THEN UPDATE SET b = b. b

  WHEN NOT MATCHED THEN INSERT VALUES (b.a, b.b);

  acidTbl 内容变更为：

```
+------------+------------+
| acidtbl.a  | acidtbl.b  |
+------------+------------+
| 1          | 20         |
| 2          | 10         |
+------------+------------+
2 rows selected (0.666 seconds)
```

说明

执行 merge 命令时，如果出现“Error evaluating cardinality_violation”异常。请检查连接键是否有重复，或者执行 **set hive.merge.cardinality.check=false;**命令用以规避。

- 删除事务表记录：

  DELETE FROM acidTbl where a = 2;

```
+------------+------------+
| acidtbl.a  | acidtbl.b  |
+------------+------------+
| 1          | 20         |
+------------+------------+
1 row selected (1.253 seconds)
```

查看事务执行状态

步骤 7 执行以下命令查看事务执行状态。

- 查看锁：

  show locks;

- 查看压缩任务：

  show compactions;

- 查看事务执行状态：

  show transactions;

- 中断事务：

  abort transactions *TransactionId*;

  其中“TransactionId”即是执行•查看事务执行状态：命令后，结果中“Transaction ID”所在列的参数值。

----结束

配置压缩功能

HDFS 不支持文件的就地更改，对于新增内容，它也不为用户提供读取的一致性。为了在 HDFS 上提供这些特性，我们遵循了在其他数据仓库工具中使用的标准方法：表或分区的数据存储在一组基本文件中，新增、更新和删除的记录存储在增量文件中。每个事务都创建一组新的增量文件以更改表或分区。在读取时，合并基础文件和增量文件并应用更新或删除的变化。

写事务表将在 HDFS 上产生部分小文件，Hive 提供合并这些小文件的 Major 压缩和 Minor 压缩策略。

自动执行压缩操作步骤

步骤 1 登录 FusionInsight Manager 界面，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本），选择“集群 > *待操作的集群* > 服务 > Hive > 配置 > 全部配置 > MetaStore（角色） > 事务”。

步骤 2 根据实际要求配置以下参数：

表10-11 参数配置

参数	描述
hive.compactor.check.interval	压缩线程的执行间隔时间。单位：秒。默认值：300。
hive.compactor.cleaner.run.interval	清理线程的执行间隔时间。单位：毫秒。默认值：5000。
hive.compactor.delta.num.threshold	触发 Minor 压缩的增量文件个数阈值。默认值：10。
hive.compactor.delta.pct.threshold	触发 Major 压缩的增量文件（delta）大小总和占 base 文件大小比例阈值，0.1 表示 delta 文件大小之和与 base 文件大小之比为 10%时触发 Major 压缩。 默认值：0.1。
hive.compactor.max.num.delta	压缩器将在单个作业中尝试处理的最大增量文件数。默认值：500。
metastore.compactor.initiator.on	是否在此 MetaStore 实例上运行启动程序线程和清理程序线程。开启事务值必须为 true。默认值：false。
metastore.compactor.worker.threads	在 MetaStore 上运行多少个压缩程序工作线程。设置为 0 表示不执行压缩，使用事务必须在 MetaStore 服务的一个或多个实例上将此值设置为正数。单位：秒。默认值：0。

步骤 3 登录 Hive 客户端，执行压缩，具体操作请参考使用 Hive 客户端。

```
CREATE TABLE table_name (
 id int, name string
)
CLUSTERED BY (id) INTO 2 BUCKETS STORED AS ORC
TBLPROPERTIES ("transactional"="true",
  "compactor.mapreduce.map.memory.mb"="2048",                 -- 指定紧缩 map 作业的属性
  "compactorthreshold.hive.compactor.delta.num.threshold"="4",  -- 如果有超过 4 个增量目录，则触发轻度紧缩
  "compactorthreshold.hive.compactor.delta.pct.threshold"="0.5" -- 如果增量文件的大小与基础文件的大小的比率大于 50%，则触发深度紧缩
);
```

或

```
ALTER TABLE table_name COMPACT 'minor' WITH OVERWRITE TBLPROPERTIES
("compactor.mapreduce.map.memory.mb"="3072");  -- 指定紧缩 map 作业的属性
ALTER TABLE table_name COMPACT 'major' WITH OVERWRITE TBLPROPERTIES
("tblprops.orc.compress.size"="8192");         -- 更改任何其他 Hive 表属性
```

说明

执行压缩后小文件不会被立即删除，cleaner 线程完成清理后文件被批量删除。

----结束

10.29 切换 Hive 执行引擎为 Tez

操作场景

Hive 支持使用 Tez 引擎处理数据计算任务，用户在执行任务前可手动切换执行引擎为 Tez。

前提条件

集群已安装 Yarn 服务的 TimelineServer 角色，且角色运行正常。

客户端切换执行引擎为 Tez

步骤 1 安装并登录 Hive 客户端，具体操作请参考使用 Hive 客户端。

步骤 2 执行以下命令切换引擎并开启“yarn.timeline-service.enabled”参数：

set hive.execution.engine=tez;

set yarn.timeline-service.enabled=true;

说明

- “yarn.timeline-service.enabled”参数开启后可以在 Tez 服务中通过 TezUI 查看 Tez 引擎执行任务的详细情况。开启后任务信息将上报 TimelineServer，如果 TimelineServer 实例故障，会导致任务失败。
- 由于 Tez 使用 ApplicationMaster 缓冲池，“yarn.timeline-service.enabled”必须在提交 Tez 任务前开启，否则会导致此参数无法生效，需要重新登录客户端进行配置。
- 当执行引擎需要切换为其它引擎时，需要通过客户端执行 **set yarn.timeline-service.enabled=false** 命令关闭“yarn.timeline-service.enabled”参数。
- 如果需要指定 Yarn 运行队列，可以在客户端执行 **set tez.queue.name=default** 命令指定运行队列。

步骤 3 提交并执行 Tez 任务。

步骤 4 登录 FusionInsight Manager 界面，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本），选择“集群 > *待操作的集群* > 服务 > Tez > TezUI（*主机名称*）”，在 TezUI 界面查看任务执行情况。

针对 MRS 3.x 之前版本，请登录 MRS Manager 界面，选择“服务管理 > Tez > Tez WebUI”，在 TezUI 界面查看任务执行情况。

----结束

切换 Hive 服务默认执行引擎为 Tez

步骤 1 登录 FusionInsight Manager 界面，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本），选择“集群 > *待操作的集群* > 服务 > Hive > 配置 > 全部配置 > HiveServer（角色）”，搜索“hive.execution.engine”参数。

针对 MRS 3.x 之前版本，请登录 MRS Manager 界面，选择“服务管理 > Hive > 服务配置 > 全部配置 > HiveServer”，搜索“hive.execution.engine”参数。

步骤 2 将“hive.execution.engine”参数设置为“tez”。

步骤 3 选择“Hive（服务） > 自定义”，搜索“yarn.site.customized.configs”。

步骤 4 在“yarn.site.customized.configs”参数后添加自定义参数，名称为“yarn.timeline-service.enabled”，值为“true”。

说明

- “yarn.timeline-service.enabled”开启后可以在 Tez 服务中通过 TezUI 查看 Tez 引擎执行任务详细情况。开启后任务信息将上报 TimelineServer，如果 TimelineServer 实例故障，会导致任务失败。
- 由于 Tez 使用 ApplicationMaster 缓冲池，“yarn.timeline-service.enabled”必须在提交 Tez 任务前开启，否则会导致此参数无法生效，需要重新登录客户端配置。
- 当执行引擎需要切换为其它引擎时，需要将自定义参数“yarn.timeline-service.enabled”的值设置为“false”。

步骤 5 单击“保存”在弹出窗口单击“确定”。

针对 MRS 3.x 之前版本，请单击“保存配置”在弹出窗口单击“是”。

步骤 6 选择“概览 > 更多 > 重启服务”，重启 Hive 服务，输入密码开始重启服务。

针对 MRS 3.x 之前版本，请在“服务状态”页签选择“更多 > 重启服务”，重启 Hive 服务。

步骤 7 安装并登录 Hive 客户端，具体操作请参考使用 Hive 客户端。

步骤 8 提交并执行 Tez 任务。

步骤 9 登录 FusionInsight Manager 界面，选择“集群 > *待操作的集群* > 服务 > Tez > TezUI（*主机名称*）”，跳转 TezUI 界面查看任务执行情况。

针对 MRS 3.x 之前版本，请登录 MRS Manager 界面，选择“服务管理 > Tez > Tez WebUI”，在 TezUI 界面查看任务执行情况。

----结束

10.30 Hive 物化视图

Hive 物化视图介绍

Hive 物化视图是基于 Hive 内部表的查询结果得到的特殊表，物化视图可以看做一张中间表，存储实际的数据，占用物理空间。物化视图赖以建立的这些表称为物化视图的基表。

物化视图主要用于预先计算并保存表连接或聚合等耗时较多的操作的结果。在执行查询时，可以将原本基于基表查询的查询语句重写成基于物化视图查询，这样就可以避免进行 join、group by 等耗时的操作，从而快速的得到结果。

说明

- 物化视图是特殊的表，存储实际的数据，占用物理空间。
- 删除基表之前必须先删除基于该基表所建立的物化视图。
- 物化视图创建语句是原子的，这意味着在填充所有查询结果之前，其他用户看不到物化视图。
- 不能基于物化视图的查询结果建立物化视图。
- 不能基于无表查询得到的查询结果建立物化视图。
- 不能对物化视图做增删改操作（即 insert、update、delete、load、merge）。
- 能对物化视图做复杂查询操作，因其本质就是一张特殊的表。
- 当基表数据更新，需要手动对物化视图进行更新，否则物化视图将保留旧数据，即过期。
- 可通过 describe 语法查看基于 acid 表创建的物化视图是否过期。
- 基于非 acid 表创建的物化视图，无法通过 describe 语句查询物化视图是否过期。
- 创建物化视图只支持文件存储格式是“ORC”，并且支持事务（即“TBLPROPERTIES ('transactional'='true')”）的 Hive 内部表。

创建物化视图

语法

```
CREATE MATERIALIZED VIEW [IF NOT EXISTS] [db name.]materialized view name
  [COMMENT materialized view comment]
  DISABLE REWRITE
    [ROW FORMAT row format]
    [STORED AS file format]
      | STORED BY 'storage.handler.class.name' [WITH SERDEPROPERTIES (...)]
  ]
  [LOCATION hdfs_path]
  [TBLPROPERTIES (property_name=property_value, ...)]
AS
<query>;
```

说明

- 目前，物化视图文件格式支持：“PARQUET”、“TextFile”、“SequenceFile”、“RCfile”、“ORC”。如未在创建语句中使用“STORED AS”指定，则默认文件格式是 ORC。
- 在同一 Database 下不可创建同名的物化视图，否则在新物化视图无法正常创建的同时，原物化视图的数据文件也会被新物化视图基于基表查询得到的数据文件覆盖，造成数据篡改（篡改后可通过重建物化视图进行恢复）。

案例

步骤 1 登录 Hive 客户端，执行命令开启以下参数，具体操作请参考使用 Hive 客户端。

set hive.support.concurrency=true;

set hive.exec.dynamic.partition.mode=nonstrict;

set hive.txn.manager=org.apache.hadoop.hive.ql.lockmgr.DbTxnManager;

步骤 2 创建基表，插入数据。

```
create table tb_emp(
empno int,ename string,job string,mgr int,hiredate TIMESTAMP,sal float,comm
float,deptno int
)stored as orc
tblproperties('transactional'='true');

insert into tb_emp values(7369, 'SMITH', 'CLERK',7902, '1980-12-17
08:30:09',800.00,NULL,20),
(7499, 'ALLEN', 'SALESMAN',7698, '1981-02-20 17:12:00',1600.00,300.00,30),
(7521, 'WARD', 'SALESMAN',7698, '1981-02-22 09:05:34',1250.00,500.00,30),
(7566, 'JONES', 'MANAGER', 7839, '1981-04-02 10:14:13',2975.00,NULL,20),
(7654, 'MARTIN', 'SALESMAN',7698, '1981-09-28 08:36:17',1250.00,1400.00,30),
(7698, 'BLAKE', 'MANAGER',7839, '1981-05-01 11:12:55',2850.00,NULL,30),
(7782, 'CLARK', 'MANAGER',7839, '1981-06-09 15:45:28',2450.00,NULL,10),
(7788, 'SCOTT', 'ANALYST',7566, '1987-04-19 14:05:34',3000.00,NULL,20),
(7839, 'KING', 'PRESIDENT',NULL, '1981-11-17 10:18:25',5000.00,NULL,10),
(7844, 'TURNER', 'SALESMAN',7698, '1981-09-08 09:05:34',1500.00,0.00,30),
(7876, 'ADAMS', 'CLERK',7788, '1987-05-23 15:07:44',1100.00,NULL,20),
(7900, 'JAMES', 'CLERK',7698, '1981-12-03 16:23:56',950.00,NULL,30),
(7902, 'FORD', 'ANALYST',7566, '1981-12-03 08:48:17',3000.00,NULL,20),
(7934, 'MILLER', 'CLERK',7782, '1982-01-23 11:45:29',1300.00,NULL,10);
```

步骤 3 基于 tb_emp 的查询，创建物化视图。

```
create materialized view group mv disable rewrite
row format serde 'org.apache.hadoop.hive.serde2.JsonSerDe'
stored as textfile
tblproperties('mv_content'='Total compensation of each department')
as select deptno,sum(sal) sum_sal from tb_emp group by deptno;
```

----结束

应用物化视图

将原本基于基表查询的查询语句重写成基于物化视图查询，从而达到提升查询效率的效果。

案例

现有查询语句如下：

select deptno,sum(sal) from tb_emp group by deptno having sum(sal)>10000;

基于所创建的物化视图，可将查询语句改写成：

select deptno, sum_sal from group_mv where sum_sal>10000;

查看物化视图

语法

SHOW MATERIALIZED VIEWS [IN database_name] ['identifier_with_wildcards’];

DESCRIBE [EXTENDED | FORMATTED] [db_name.]materialized_view_name;

案例

show materialized views;

describe formatted group_mv;

删除物化视图

语法

DROP MATERIALIZED VIEW [db_name.]materialized_view_name;

案例

drop materialized view group_mv;

重建物化视图

创建物化视图的时候，基表数据会填充到物化视图中，但是后续增删改基表的数据，这部分数据是不会自动的同步到物化视图中的。因此，我们在更新数据后，需要手动对视图进行重建。

语法

ALTER MATERIALIZED VIEW [db_name.]materialized_view_name REBUILD;

案例

alter materialized view group_mv rebuild;

说明

当基表数据更新，而物化视图的数据未更新，则默认物化视图的状态为过期。

基于事务表创建的物化视图，可以通过 descirbe 语句查看物化视图是否过期。其中“Outdated for Rewriting”值为“Yes”，表示过期，值为“No”,表示未过期。

10.31 Hive 支持分区元数据冷热存储

分区元数据冷热存储介绍

- 为了减轻元数据库压力，将长时间未使用过的指定范围的分区相关元数据移动到备份表，这一过程称为分区数据冻结，移动的分区数据称为冷分区，未冻结的分区称为热分区，存在冷分区的表称为冻结表。将被冻结的数据重新移回原元数据表，这一过程称为分区数据解冻。
- 一个分区从热分区变成冷分区，仅仅是在元数据中进行标识，其 HDFS 业务侧分区路径、数据文件内容并未发生变化。

说明

本特性仅适用于 MRS 3.1.2 及之后版本。

冻结分区

支持创建表的用户按照条件过滤的方式对一个或多个分区进行冻结，格式为：freeze partitions *数据库名称.表名称* where *分区过滤条件*

例如：

```
freeze partitions testdb.test where year <= 2021;
freeze partitions testdb.test where year<=2021 and month <= 5;
freeze partitions testdb.test where year<=2021 and month <= 5 and day <= 27;
```

解冻分区

支持创建表的用户按照条件过滤的方式对一个或多个分区进行解冻，格式为 unfreeze partitions 数据库名称.表名称 where 分区过滤条件，如：

```
unfreeze partitions testdb.test where year <= 2021;
unfreeze partitions testdb.test where year<=2021 and month <= 5;
unfreeze partitions testdb.test where year<=2021 and month <= 5 and day <= 27;
```

查询含有冻结数据的表

- 查询当前数据库下的所有冻结表：

  show frozen tables;
- 查询 dbname 数据库下的所有冻结表：

  show frozen tables in dbname;

查询冻结表的冻结分区

查询冷冻分区：

show frozen partitions table;

说明

- 默认元数据库冻结分区类型只支持 int、string、varchar、date、timestamp 类型。
- 外置元数据库只支持 Postgres 数据库，且冻结分区类型只支持 int、string、varchar、timestamp 类型。
- 对冻结后的表进行 Msck 元数据修复时，需要先解冻数据。如果对冻结表进行过备份后恢复操作，则可以直接执行 Msck 元数据修复操作，且解冻只能通过 **msck repair** 命令进行操作。
- 对冻结后的分区进行 rename 时，需要先解冻数据，否则会提示分区不存在。
- 删除存在冻结数据的表时，被冻结的数据会同步删除。
- 删除存在冻结数据的分区时，被冻结的分区信息不会被删除，HDFS 业务数据也不会被删除。
- select 查询数据时，会自动添加排查冷分区数据的过滤条件，查询结果将不包含冷分区的数据。
- show partitions table 查询表下的分区数据时，查询结果将不包含冷分区，可通过 show frozen partitions table 进行冷冻分区查询。

10.32 Hive 支持 ZSTD 压缩格式

ZSTD（全称为 Zstandard）是一种开源的无损数据压缩算法，其压缩性能和压缩比均优于当前 Hadoop 支持的其他压缩格式，本特性使得 Hive 支持 ZSTD 压缩格式的表。Hive 支持基于 ZSTD 压缩的存储格式有常见的 ORC，RCFile，TextFile，JsonFile，Parquet，Squence，CSV。

说明

本特性仅适用于 MRS 3.1.2 及之后版本。

ZSTD 压缩格式的建表方式如下：

- ORC 存储格式建表时可指定 TBLPROPERTIES("orc.compress"="zstd")：

  create table tab_1(...) stored as orc TBLPROPERTIES("orc.compress"="zstd");

- Parquet 存储格式建表可指定 TBLPROPERTIES("parquet.compression"="zstd")：

  create table tab_2(...) stored as parquet TBLPROPERTIES("parquet.compression"="zstd");

- 其他格式或通用格式建表可执行设置参数指定 compress,codec 为“org.apache.hadoop.io.compress.ZStandardCode”：

  set hive.exec.compress.output=true;

  set mapreduce.map.output.compress=true;

  set mapreduce.map.output.compress.codec=org.apache.hadoop.io.compress.ZStandardCodec;

  set mapreduce.output.fileoutputformat.compress=true;

  set mapreduce.output.fileoutputformat.compress.codec=org.apache.hadoop.io.compress.ZStandardCodec;

  set hive.exec.compress.intermediate=true;

  create table tab_3(...) stored as textfile;

说明

ZSTD 压缩格式的表和其他普通压缩表的 SQL 操作没有区别，可支持正常的增删查及聚合类 SQL 操作。

10.33 Hive 日志介绍

日志描述

日志路径：Hive 相关日志的默认存储路径为“/var/log/Bigdata/hive/角色名”，Hive1 相关日志的默认存储路径为“/var/log/Bigdata/hive1/角色名”，以此类推。

- HiveServer：“/var/log/Bigdata/hive/hiveserver”（运行日志），“/var/log/Bigdata/audit/hive/hiveserver”（审计日志）。

- MetaStore：“/var/log/Bigdata/hive/metastore”（运行日志），“/var/log/Bigdata/audit/hive/metastore”（审计日志）。
- WebHCat：“/var/log/Bigdata/hive/webhcat”（运行日志），“/var/log/Bigdata/audit/hive/webhcat”（审计日志）。

日志归档规则：Hive 的日志启动了自动压缩归档功能，缺省情况下，当日志大小超过 20MB 的时候（此日志文件大小可进行配置），会自动压缩，压缩后的日志文件名规则为：“<原有日志名>-<yyyy-mm-dd_hh-mm-ss>.[编号].log.zip”。最多保留最近的 20 个压缩文件，压缩文件保留个数和压缩文件阈值可以配置。

表10-12 Hive 日志列表

日志类型	日志文件名	描述
运行日志	/hiveserver/hiveserver.out	HiveServer 运行环境信息日志
	/hiveserver/hive.log	HiveServer 进程的运行日志
	/hiveserver/hive-omm-<日期>-<PID>-gc.log.<编号>	HiveServer 进程的 GC 日志
	/hiveserver/prestartDetail.log	HiveServer 启动前的工作日志
	/hiveserver/check-serviceDetail.log	Hive 服务启动是否成功的检查日志
	/hiveserver/cleanupDetail.log	HiveServer 卸载的清理日志
	/hiveserver/startDetail.log	HiveServer 进程启动日志
	/hiveserver/stopDetail.log	HiveServer 进程停止日志
	/hiveserver/localtasklog/omm_<日期>_<任务 ID>.log	Hive 本地任务的运行日志
	/hiveserver/localtasklog/omm_<日期>_<任务 ID>-gc.log.<编号>	Hive 本地任务的 GC 日志
	/metastore/metastore.log	MetaStore 进程的运行日志
	/metastore/hive-omm-<日期>-<PID>-gc.log.<编号>	MetaStore 进程的 GC 日志
	/metastore/postinstallDetail.log	MetaStore 安装后的工作日志
	/metastore/prestartDetail.log	MetaStore 启动前的工作日志
	/metastore/cleanupDetail.log	MetaStore 卸载的清理日志
	/metastore/startDetail.log	MetaStore 进程启动日志

日志类型	日志文件名	描述
	/metastore/stopDetail.log	MetaStore 进程停止日志
	/metastore/metastore.out	MetaStore 运行环境信息日志
	/webhcat/webhcat-console.out	Webhcat 进程启停正常日志
	/webhcat/webhcat-console-error.out	Webhcat 进程启停异常日志
	/webhcat/prestartDetail.log	WebHCat 启动前的工作日志
	/webhcat/cleanupDetail.log	Webhcat 卸载时或安装前的清理日志
	/webhcat/hive-omm-<日期>-<PID>-gc.log.<编号>	WebHCat 进程的 GC 日志
	/webhcat/webhcat.log	WebHCat 进程的运行日志
审计日志	hive-audit.log hive-rangeraudit.log	HiveServer 审计日志
	metastore-audit.log	MetaStore 审计日志
	webhcat-audit.log	WebHCat 审计日志
	jetty-<日期>.request.log	Jetty 服务的请求日志

日志级别

Hive 提供了如表 10-13 所示的日志级别。

运行日志的级别优先级从高到低分别是 ERROR、WARN、INFO、DEBUG，程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。

表10-13 日志级别

级别	描述
ERROR	ERROR 表示系统运行的错误信息。
WARN	WARN 表示当前事件处理存在异常信息。
INFO	INFO 表示记录系统及各事件正常运行状态信息。
DEBUG	DEBUG 表示记录系统及系统的调试信息。

如果您需要修改日志级别，请执行如下操作：

步骤 1 参考修改集群服务配置参数，进入 Hive 服务“全部配置”页面。

步骤 2 左边菜单栏中选择所需修改的角色所对应的日志菜单。

步骤 3 选择所需修改的日志级别并保存。

说明

配置 Hive 日志级别后可立即生效，无需重启服务。

----结束

日志格式

Hive 的日志格式如下所示：

表10-14 日志格式

日志类型	格式	示例
运行日志	<yyyy-MM-dd HH:mm:ss,SSS>|<LogLevel>|<产生该日志的线程名字>|<log 中的 message>|<日志事件的发生位置>	2014-11-05 09:45:01,242 | INFO | main | Starting hive metastore on port 21088 | org.apache.hadoop.hive.metastore.HiveMetaStore.main(HiveMetaStore.java:5198)
审计日志	<yyyy-MM-dd HH:mm:ss,SSS>|<LogLevel>|<产生该日志的线程名字>|<User Name><User IP><Time><Operation><Resource><Result><Detail >|<日志事件的发生位置>	2018-12-24 12:16:25,319 | INFO | HiveServer2-Handler-Pool: Thread-185 | UserName=hive UserIP=10.153.2.204 Time=2018/12/24 12:16:25 Operation=CloseSession Result=SUCCESS Detail= | org.apache.hive.service.cli.thrift.ThriftCLIService.logAuditEvent(ThriftCLIService.java:434)

10.34 Hive 性能调优

10.34.1 建立表分区

操作场景

Hive 在做 Select 查询时，一般会扫描整个表内容，会消耗较多时间去扫描不关注的数据。此时，可根据业务需求及其查询维度，建立合理的表分区，从而提高查询效率。

操作步骤

步骤 1 MRS 3.x 之前版本：

登录 MRS 控制台，在左侧导航栏选择“集群列表 > 现有集群”，单击集群名称。选择“节点管理 > 节点名称” ，进入弹性云主机界面。单击“远程登录”按钮，完成 Hive 节点的登录。

MRS 3.x 及后续版本：

以 **root** 用户登录已安装 Hive 客户端的节点。

步骤 2 执行以下命令，进入客户端安装目录，例如“/opt/client”。

cd /opt/client

步骤 3 执行 **source bigdata_env** 命令，配置客户端环境变量。

步骤 4 在客户端中执行如下命令，执行登录操作。

kinit *用户名*

步骤 5 执行以下命令登录客户端工具。

beeline

步骤 6 指定静态分区或者动态分区。

- 静态分区：

  静态分区是手动输入分区名称，在创建表时使用关键字 **PARTITIONED BY** 指定分区列名及数据类型。应用开发时，使用 **ALTER TABLE ADD PARTITION** 语句增加分区，以及使用 **LOAD DATA INTO PARTITON** 语句将数据加载到分区时，只能静态分区。

- 动态分区：通过查询命令，将结果插入到某个表的分区时，可以使用动态分区。

  动态分区通过在客户端工具执行如下命令来开启：

  set hive.exec.dynamic.partition=true;

  动态分区默认模式是 strict，也就是必须至少指定一列为静态分区，在静态分区下建立动态子分区，可以通过如下设置来开启完全的动态分区：

  set hive.exec.dynamic.partition.mode=nonstrict;

说明

- 动态分区可能导致一个 DML 语句创建大量的分区，对应的创建大量新文件夹，对系统性能可能带来影响。
- 在文件数量大的情况下，执行一个 SQL 语句启动时间较长，可以在执行 SQL 语句之前执行“set mapreduce.input.fileinputformat.list-status.num-threads = 100;”命令来缩短启动时间。“mapreduce.input.fileinputformat.list-status.num-threads”参数需要先添加到 Hive 的白名单才可设置。

----结束

10.34.2 Join 优化

操作场景

使用 Join 语句时，如果数据量大，可能造成命令执行速度和查询速度慢，此时可进行 Join 优化。

Join 优化可分为以下方式：

- Map Join
- Sort Merge Bucket Map Join
- Join 顺序优化

Map Join

Hive 的 Map Join 适用于能够在内存中存放下的小表（指表大小小于 25MB），通过“hive.mapjoin.smalltable.filesize”定义小表的大小，默认为 25MB。

Map Join 的方法有两种：

- 使用/*+ MAPJOIN(join_table) */。
- 执行语句前设置如下参数，当前版本中该值默认为 true。

  set hive.auto.convert.join=true;

使用 Map Join 时没有 Reduce 任务，而是在 Map 任务前起了一个 MapReduce Local Task，这个 Task 通过 TableScan 读取小表内容到本机，在本机以 HashTable 的形式保存并写入硬盘上传到 DFS，并在 distributed cache 中保存，在 Map Task 中从本地磁盘或者 distributed cache 中读取小表内容直接与大表 join 得到结果并输出。

使用 Map Join 时需要注意小表不能过大，如果小表将内存基本用尽，会使整个系统性能下降甚至出现内存溢出的异常。

Sort Merge Bucket Map Join

使用 Sort Merge Bucket Map Join 必须满足以下 2 个条件：

- join 的两张表都很大，内存中无法存放。
- 两张表都按照 join key 进行分桶（clustered by (column)）和排序（sorted by(column)），且两张表的分桶数正好是倍数关系。

通过如下设置，启用 Sort Merge Bucket Map Join：

set hive.optimize.bucketmapjoin=true;

set hive.optimize.bucketmapjoin.sortedmerge=true;

这种 Map Join 也没有 Reduce 任务，是在 Map 任务前启动 MapReduce Local Task，将小表内容按桶读取到本地，在本机保存多个桶的 HashTable 备份并写入 HDFS，并保存在 Distributed Cache 中，在 Map Task 中从本地磁盘或者 Distributed Cache 中按桶一个一个读取小表内容，然后与大表做匹配直接得到结果并输出。

Join 顺序优化

当有 3 张及以上的表进行 Join 时，选择不同的 Join 顺序，执行时间存在较大差异。使用恰当的 Join 顺序可以有效缩短任务执行时间。

Join 顺序原则：

- Join 出来结果较小的组合，例如表数据量小或两张表 Join 后产生结果较少，优先执行。
- Join 出来结果大的组合，例如表数据量大或两张表 Join 后产生结果较多，在后面执行。

例如，customer 表的数据量最多，orders 表和 lineitem 表优先 Join 可获得较少的中间结果。

原有的 Join 语句如下：

```
select
  l_orderkey,
  sum(l_extendedprice * (1 - l_discount)) as revenue,
  o_orderdate,
  o_shippriority
from
  customer,
  orders,
  lineitem
where
  c mktsegment = 'BUILDING'
  and c custkey = o custkey
  and l orderkey = o orderkey
  and o orderdate < '1995-03-22'
  and l shipdate > '1995-03-22'
limit 10;
```

Join 顺序优化后如下：

```
select
  l_orderkey,
  sum(l_extendedprice * (1 - l_discount)) as revenue,
  o_orderdate,
  o_shippriority
from
  orders,
  lineitem,
  customer
where
  c_mktsegment = 'BUILDING'
  and c_custkey = o_custkey
  and l_orderkey = o_orderkey
  and o_orderdate < '1995-03-22'
  and l_shipdate > '1995-03-22'
limit 10;
```

注意事项

Join 数据倾斜问题

执行任务的时候，任务进度长时间维持在 99%，这种现象叫数据倾斜。

数据倾斜是经常存在的，因为有少量的 Reduce 任务分配到的数据量和其他 Reduce 差异过大，导致大部分 Reduce 都已完成任务，但少量 Reduce 任务还没完成的情况。

解决数据倾斜的问题，可通过设置“set hive.optimize.skewjoin=true”并调整 hive.skewjoin.key 的大小。hive.skewjoin.key 是指 Reduce 端接收到多少个 key 即认为数据是倾斜的，并自动分发到多个 Reduce。

10.34.3 Group By 优化

操作场景

优化 Group by 语句，可提升命令执行速度和查询速度。

Group by 的时候， Map 端会先进行分组， 分组完后分发到 Reduce 端， Reduce 端再进行分组。可采用 Map 端聚合的方式来进行 Group by 优化，开启 Map 端初步聚合，减少 Map 的输出数据量。

操作步骤

在 Hive 客户端进行如下设置：

set hive.map.aggr=true;

注意事项

Group By 数据倾斜

Group By 也同样存在数据倾斜的问题，设置 hive.groupby.skewindata 为 true，生成的查询计划会有两个 MapReduce Job，第一个 Job 的 Map 输出结果会随机的分布到 Reduce 中，每个 Reduce 做聚合操作，并输出结果，这样的处理会使相同的 Group By Key 可能被分发到不同的 Reduce 中，从而达到负载均衡，第二个 Job 再根据预处理的结果按照 Group By Key 分发到 Reduce 中完成最终的聚合操作。

Count Distinct 聚合问题

当使用聚合函数 count distinct 完成去重计数时，处理值为空的情况会使 Reduce 产生很严重的数据倾斜，可以将空值单独处理，如果是计算 count distinct，可以通过 where 字句将该值排除掉，并在最后的 count distinct 结果中加 1。如果还有其他计算，可以先将值为空的记录单独处理，再和其他计算结果合并。

10.34.4 数据存储优化

操作场景

“ORC”是一种高效的列存储格式，在压缩比和读取效率上优于其他文件格式。

建议使用“ORC”作为 Hive 表默认的存储格式。

前提条件

已登录 Hive 客户端，具体操作请参见使用 Hive 客户端。

操作步骤

- 推荐：使用“SNAPPY”压缩，适用于压缩比和读取效率要求均衡场景。

  Create table *xx (col_name data_type)* stored as orc tblproperties ("orc.compress"="SNAPPY");

- 可用：使用“ZLIB”压缩，适用于压缩比要求较高场景。

  Create table *xx (col_name data_type)* stored as orc tblproperties ("orc.compress"="ZLIB");

说明

xx 为具体使用的 Hive 表名。

10.34.5 SQL 优化

操作场景

在 Hive 上执行 SQL 语句查询时，如果语句中存在“(a&b) or (a&c)”逻辑时，建议将逻辑改为“a & (b or c)”。

样例

假设条件 a 为“p_partkey = l_partkey”，优化前样例如下所示：

```
select
      sum(l_extendedprice* (1 - l_discount)) as revenue
from
      lineitem,
      part
where
      (
            p_partkey = l_partkey
            and p_brand = 'Brand#32'
            and p_container in ('SM CASE', 'SM BOX', 'SM PACK', 'SM PKG')
            and l_quantity >= 7 and l_quantity <= 7 + 10
            and p_size between 1 and 5
            and l_shipmode in ('AIR', 'AIR REG')
            and l_shipinstruct = 'DELIVER IN PERSON'
      )
      or
      (     p_partkey = l_partkey
            and p brand = 'Brand#35'
            and p container in ('MED BAG', 'MED BOX', 'MED PKG', 'MED PACK')
            and l quantity >= 15 and l quantity <= 15 + 10
            and p size between 1 and 10
            and l shipmode in ('AIR', 'AIR REG')
            and l_shipinstruct = 'DELIVER IN PERSON'
      )
      or
```

```
        (       p_partkey = l_partkey
                and p_brand = 'Brand#24'
                and p_container in ('LG CASE', 'LG BOX', 'LG PACK', 'LG PKG')
                and l_quantity >= 26 and l_quantity <= 26 + 10
                and p_size between 1 and 15
                and l_shipmode in ('AIR', 'AIR REG')
                and l_shipinstruct = 'DELIVER IN PERSON'
        )
```

优化后样例如下所示：

```
select
        sum(l_extendedprice* (1 - l_discount)) as revenue
from
        lineitem,
        part
where   p_partkey = l_partkey and
        ((
                p_brand = 'Brand#32'
                and p_container in ('SM CASE', 'SM BOX', 'SM PACK', 'SM PKG')
                and l_quantity >= 7 and l_quantity <= 7 + 10
                and p_size between 1 and 5
                and l_shipmode in ('AIR', 'AIR REG')
                and l_shipinstruct = 'DELIVER IN PERSON'
        )
        or
        (
                p_brand = 'Brand#35'
                and p_container in ('MED BAG', 'MED BOX', 'MED PKG', 'MED PACK')
                and l_quantity >= 15 and l_quantity <= 15 + 10
                and p_size between 1 and 10
                and l_shipmode in ('AIR', 'AIR REG')
                and l_shipinstruct = 'DELIVER IN PERSON'
        )
        or
        (
                p_brand = 'Brand#24'
                and p_container in ('LG CASE', 'LG BOX', 'LG PACK', 'LG PKG')
                and l_quantity >= 26 and l_quantity <= 26 + 10
                and p_size between 1 and 15
                and l_shipmode in ('AIR', 'AIR REG')
                and l_shipinstruct = 'DELIVER IN PERSON'
        ))
```

10.34.6 使用 Hive CBO 优化查询

操作场景

在 Hive 中执行多表 Join 时，Hive 支持开启 CBO（Cost Based Optimization），系统会自动根据表的统计信息，例如数据量、文件数等，选出最优计划提高多表 Join 的效率。Hive 需要先收集表的统计信息后才能使 CBO 正确的优化。

说明

- CBO 优化器会基于统计信息和查询条件，尽可能地使 join 顺序达到最优。但是也可能存在特殊情况导致 join 顺序调整不准确。例如数据存在倾斜，以及查询条件值在表中不存在等场景，可能调整出非优化的 join 顺序。
- 开启列统计信息自动收集时，需要在 reduce 侧做聚合统计。对于没有 reduce 阶段的 insert 任务，将会多出 reduce 阶段，用于收集统计信息。
- 本章节适用于 MRS 3.x 及后续版本。

前提条件

已登录 Hive 客户端，具体操作请参见使用 Hive 客户端。

操作步骤

步骤 1 在 Manager 界面 Hive 组件的配置中搜索“hive.cbo.enable”参数，选中“true”永久开启功能。

步骤 2 手动收集 Hive 表已有数据的统计信息。

执行以下命令，可以手动收集统计信息。仅支持统计一张表，如果需要统计不同的表需重复执行。

ANALYZE TABLE [db_name.]tablename [PARTITION(partcol1[=val1], partcol2[=val2], ...)]

COMPUTE STATISTICS

[FOR COLUMNS]

[NOSCAN];

说明

- 指定 FOR COLUMNS 时，收集列级别的统计信息。
- 指定 NOSCAN 时，将只统计文件大小和个数，不扫描具体文件。

例如：

analyze table table_name compute statistics;

analyze table table_name compute statistics for columns;

步骤 3 配置 Hive 自动收集统计信息。开启配置后，执行 **insert overwrite/into** 命令插入数据时才自动统计新数据的信息。

- 在 Hive 客户端执行以下命令临时开启收集：

  set hive.stats.autogather = true;开启表/分区级别的统计信息自动收集。

  set hive.stats.column.autogather = true; 开启列级别的统计信息自动收集。

说明

- 列级别统计信息的收集不支持复杂的数据类型，例如 Map，Struct 等。

- 表级别统计信息的自动收集不支持 Hive on HBase 表。
- 在 Manager 界面 Hive 的服务配置中，搜索参数“hive.stats.autogather”和“hive.stats.column.autogather”，选中“true”永久开启收集功能。

步骤 4 执行以下命令可以查看统计信息。

DESCRIBE FORMATTED table_name[.column_name] PARTITION partition_spec;

例如：

desc formatted table_name;

desc formatted table_name id;

desc formatted table_name partition(time='2016-05-27');

说明

分区表仅支持分区级别的统计信息收集，因此分区表需要指定分区来查询统计信息。

----结束

10.35 Hive 常见问题

10.35.1 如何在多个 HiveServer 之间同步删除 UDF

问题

如果需要删除永久函数（Permanent UDF），如何在多个 HiveServer 之间同步删除？

回答

因为多个 HiveServer 之间共用一个 MetaStore 存储数据库，所以 MetaStore 存储数据库和 HiveServer 的内存之间数据同步有延迟。如果在单个 HiveServer 上删除永久函数，操作结果将无法同步到其他 HiveServer 上。

遇到如上情况，需要登录 Hive 客户端，连接到每个 HiveServer，并分别删除永久函数。具体操作如下：

步骤 1 以 Hive 客户端安装用户登录安装客户端的节点。

步骤 2 执行以下命令，切换到客户端安装目录。

cd *客户端安装目录*

例如安装目录为“/opt/client”，则执行以下命令：

cd /opt/client

步骤 3 执行以下命令配置环境变量。

source bigdata_env

步骤 4 执行以下命令进行用户认证。

kinit *Hive 业务用户*

说明

登录的用户需具备 Hive admin 权限。

步骤 5 执行如下命令，连接指定的 HiveServer。

beeline -u "jdbc:hive2://*10.39.151.74***:***21066***/default;sasl.qop=auth-conf;auth=KERBEROS;principal=***hive/hadoop.<系统域名>@<系统域名>**"

说明

- *10.39.151.74* 为 HiveServer 所在节点的 IP 地址。
- *21066* 为 HiveServer 端口。HiveServer 端口默认范围为 21066～21070，用户需根据实际配置端口进行修改。
- *hive* 为用户名。例如，使用 Hive1 实例时，则使用 hive1。
- 用户可登录 FusionInsight Manager，选择“系统 > 权限 > 域和互信”，查看“本端域”参数，即为当前系统域名。
- “hive/hadoop.<*系统域名*>”为用户名，用户的用户名所包含的系统域名所有字母为小写。

步骤 6 执行如下命令，启用 Hive admin 权限。

set role admin;

步骤 7 执行如下命令，删除永久函数。

drop function *function_name***;**

说明

- *function_name* 为永久函数的函数名。
- 如果永久函数是在 Spark 中创建的，在 Spark 中删除该函数后需要在 HiveServer 中删除，即执行上述删除命令。

步骤 8 确定是否已连接所有 HiveServer 并删除永久函数。

- 是，操作完毕。
- 否，执行步骤 5。

----结束

10.35.2 已备份的 Hive 表无法执行 drop 操作

问题

为什么已备份的 Hive 表执行 drop 操作会失败？

回答

由于已备份 Hive 表对应的 HDFS 目录创建了快照，导致 HDFS 目录无法删除，造成 Hive 表删除失败。

Hive 表在执行备份操作时，会创建表对应的 HDFS 数据目录快照。而 HDFS 的快照机制有一个约束：如果一个 HDFS 目录已创建快照，则在快照完全删除之前，该目录无法删除或修改名称。Hive 表（除 EXTERNAL 表外）执行 drop 操作时，会尝试删除该表对应的 HDFS 数据目录，如果目录删除失败，系统会提示表删除失败。

如果确实需要删除该表，可手动删除涉及到该表的所有备份任务。

10.35.3 如何在 Hive 自定义函数中操作本地文件

问题

在 Hive 自定义函数中需要操作本地文件，例如读取文件的内容，需要如何操作？

回答

默认情况下，可以在 UDF 中用文件的相对路径来操作文件，如下示例代码：

```
public String evaluate(String text) {
  // some logic
  File file = new File("foo.txt");
  // some logic
  // do return here
}
```

在 Hive 中使用时，将 UDF 中用到的文件“foo.txt”上传到 HDFS 上，如上传到“hdfs://hacluster/tmp/foo.txt”，使用以下语句创建 UDF，在 UDF 中就可以直接操作“foo.txt”文件了：

create function testFunc as 'some.class' using jar 'hdfs://hacluster/somejar.jar', file 'hdfs://hacluster/tmp/foo.txt';

例外情况下，如果“hive.fetch.task.conversion”参数的值为“more”，在 UDF 中不能再使用相对路径来操作文件，而要使用绝对路径，并且保证所有的 HiveServer 节点和 NodeManager 节点上该文件是存在的且 **omm** 用户对该文件有相应的权限，才能正常在 UDF 中操作本地文件。

10.35.4 如何强制停止 Hive 执行的 MapReduce 任务

问题

在 Hive 执行 MapReduce 任务长时间卡住的情况下想手动停止任务，需要如何操作？

回答

步骤 1 登录 FusionInsight Manager。

步骤 2 选择“集群 > *待操作的集群名称* > 服务 > Yarn”。

步骤 3 单击左侧页面的“ResourceManager(*主机名称*，主)”按钮，登录 Yarn 界面。

步骤 4 单击对应任务 ID 的按钮进入任务页面，单击界面左上角的“Kill Application”按钮，在弹框中单击“确认”停止任务。

----结束

10.35.5 Hive 复杂类型字段名称中包含特殊字符导致建表失败

问题

Hive 复杂类型字段名称中包含特殊字符，导致建表失败。

回答

Hive 不支持复杂类型字段名称中包含特殊字符，特殊字符是指英文大小写字母、阿拉伯数字、中文字符、葡萄牙文字符以外的其他字符。

10.35.6 如何对 Hive 表大小数据进行监控

问题

如何对 Hive 中的表大小数据进行监控？

回答

当用户要对 Hive 表大小数据进行监控时，可以通过 HDFS 的精细化监控对指定表目录进行监控，从而到达监控指定表大小数据的目的。

前提条件

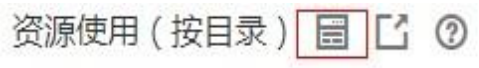
- Hive、HDFS 组件功能正常
- HDFS 精细化监控功能正常

操作步骤

步骤 1 登录 FusionInsight Manager。

步骤 2 通过“集群 > *待操作集群的名称* > 服务 > HDFS > 资源 ”，进入 HDFS 精细化页面。

步骤 3 找到“资源使用（按目录）”监控项，单击该监控项左上角第一个图标。



步骤 4 进入配置空间监控子页面，单击“添加”。

步骤 5 在名称空格中填写监控的表名称（或其他用户自定义的别名），在路径中填写需要监控表的路径。单击“确定”。该监控的横坐标为时间，纵坐标为监控目录的大小。

----结束

10.35.7 如何对重点目录进行保护，防止“insert overwrite”语句误操作导致数据丢失

问题

如何对重点目录进行保护，防止“insert overwrite”语句误操作导致数据丢失？

回答

当用户要对 Hive 重点数据库、表或目录进行监控，防止“insert overwrite”语句误操作导致数据丢失时，可以利用 Hive 配置中的“hive.local.dir.confblacklist”进行目录保护。

该配置项已对“/opt/”，“/user/hive/warehouse”等目录进行了默认配置。

前提条件

Hive、HDFS 组件功能正常。

操作步骤

步骤 1 登录 FusionInsight Manager。

步骤 2 选择“集群 > *待操作集群的名称* > 服务 > Hive > 配置 > 全部配置”，搜索“hive.local.dir.confblacklist”配置项。

步骤 3 在该配置项中添加用户要重点保护的数据库、表或目录路径。

步骤 4 输入完成后，单击“保存”，保存配置项。

----结束

10.35.8 未安装 HBase 时 Hive on Spark 任务卡顿处理

操作场景

此功能适用于 Hive 组件。

按如下操作步骤设置参数后，在未安装 HBase 的环境执行 Hive on Spark 任务时，可避免任务卡顿。

说明

Hive on Spark 任务的 Spark 内核版本已经升级到 Spark2x，可以支持在不安装 Spark2x 的情况下，执行 Hive on Spark 任务。如果没有安装 HBase，默认在执行 Spark 任务时，会尝试去连接 Zookeeper 访问 HBase，直到超时，这样会造成任务卡顿。

在未安装 HBase 的环境，要执行 Hive on Spark 任务，可以按如下操作处理。如果是从已有 HBase 低版本环境升级上来的，升级完成之后可不进行设置。

操作步骤

步骤 1 登录 FusionInsight Manager 。

步骤 2 选择“集群 > *待操作集群的名称* > 服务 > Hive > 配置 > 全部配置”。

步骤 3 选择“HiveServer（角色） > 自定义”，对参数文件“spark-defaults.conf”添加自定义参数，设置“名称”为“spark.security.credentials.hbase.enabled”，“值”为“false” 。

步骤 4 单击“保存”，在弹出对话框单击“确定”。

步骤 5 选择“集群 > *待操作集群的名称* > 服务 > Hive > 实例”，勾选所有 Hive 实例，选择“更多 > 重启实例”，输入密码，单击“确定”。

----结束

10.35.9 FusionInsight Hive 使用 WHERE 条件查询超过 3.2 万分区的表报错

问题

Hive 创建超过 3.2 万分区的表，执行带有 WHERE 分区的条件查询时出现异常，且“metastore.log”中打印的异常信息包含以下信息：

```
Caused by: java.io.IOException: Tried to send an out-of-range integer as a 2-byte
value: 32970
      at org.postgresql.core.PGStream.SendInteger2(PGStream.java:199)
      at
org.postgresql.core.v3.QueryExecutorImpl.sendParse(QueryExecutorImpl.java:1330)
      at
org.postgresql.core.v3.QueryExecutorImpl.sendOneQuery(QueryExecutorImpl.java:1601)
      at
org.postgresql.core.v3.QueryExecutorImpl.sendParse(QueryExecutorImpl.java:1191)
      at
org.postgresql.core.v3.QueryExecutorImpl.execute(QueryExecutorImpl.java:346)
```

回答

带有分区条件的查询，Hiveserver 会对分区进行优化，避免全表扫描，需要查询元数据符合条件的所有分区，而 gaussDB 中提供的接口 sendOneQuery，调用的 sendParse 方法中对参数的限制为 32767，如果分区条件数超过 32767 就异常。

10.35.10 使用 IBM 的 jdk 访问 Beeline 客户端出现连接 hiveserver 失败

操作场景

查看客户端使用的 jdk 版本，如果是 IBM JDK，则需要对 Beeline 客户端进行改造，否则会造成连接 hiveserver 失败。

操作步骤

步骤 1 登录 FusionInsight Manager 页面，选择“系统 > 权限 > 用户”，在待操作用户的“操作”栏下选择“更多 > 下载认证凭据”，选择集群信息后单击“确定”，下载 keytab 文件。

步骤 2 解压 keytab 文件，使用 WinSCP 工具将解压得到的“user.keytab”文件上传到待操作节点的 Hive 客户端安装目录下，例如：“/opt/client”。

步骤 3 使用以下命令打开 hive 客户端目录下面的配置文件 Hive/component_env：

vi *Hive 客户端安装目录*/**Hive/component_env**

在变量“export CLIENT_HIVE_URI”所在行后面添加如下内容：

```
\;user.principal=用户名@HADOOP.COM\;user.keytab=user.keytab 文件所在路径/user.keytab
```

----结束

10.35.11 关于 Hive 表的 location 支持跨 OBS 和 HDFS 路径的说明

问题

Hive 表的 location 支持跨 OBS 和 HDFS 路径吗？

回答

1. Hive 存储在 OBS 上的普通表，支持表 location 配置为 hdfs 路径。
2. 同一个 Hive 服务中可以分别创建存储在 OBS 上的表和存储在 HDFS 上的表。
3. Hive 存储在 OBS 上的分区表，不支持将分区 location 配置为 hdfs 路径（存储在 HDFS 上的分区表也不支持修改分区 location 为 OBS）。

10.35.12 通过 Tez 引擎执行 union 相关语句写入的数据，切换 MR 引擎后查询不出来。

问题

Hive 通过 Tez 引擎执行 union 相关语句写入的数据，切换到 Mapreduce 引擎后进行查询，发现数据没有查询出来。

回答

由于 Hive 使用 Tez 引擎在执行 union 语句时，生成的输出文件会存在 HIVE_UNION_SUBDIR 目录，切回 Mapreduce 引擎后默认不读取目录下的文件，所以没有读取到 HIVE_UNION_SUBDIR 目录下的数据。

此时可以设置参数 **set mapreduce.input.fileinputformat.input.dir.recursive=true**，开启 union 优化，决定是否读取目录下的数据。

10.35.13 Hive 不支持对同一张表或分区进行并发写数据

问题

为什么通过接口并发对 Hive 表进行写数据会导致数据不一致？

回答

Hive 不支持对同一张表或同一个分区进行并发数据插入，这样会导致多个任务操作同一个数据临时目录，一个任务将另一个任务的数据移走，导致任务数据异常。解决方法是修改业务逻辑，单线程插入数据到同一张表或同一个分区。

10.35.14 Hive 不支持向量化查询

问题

当设置向量化参数 hive.vectorized.execution.enabled=true 时，为什么执行 hive on Tez/Mapreduce/Spark 时会偶现一些空指针或类型转化异常？

回答

当前 Hive 不支持向量化执行，向量化执行有很多社区问题引入目前没有稳定修复，默认 hive.vectorized.execution.enabled=false，不建议将次参数打开。

10.35.15 Hive 表 HDFS 数据目录被误删，但是元数据仍然存在，导致执行任务报错处理

问题

Hive 表 HDFS 数据目录被误删，但是元数据仍然存在，导致执行任务报错。

回答

这是一种误操作的异常情况，需要手动删除对应表的元数据后重试。

例如：

执行以下命令进入控制台：

source ${BIGDATA_HOME}/FusionInsight_BASE_8.1.0.1/install/FusionInsight-dbservice-2.7.0/.dbservice_profile

gsql -p 20051 -U hive -d hivemeta -W HiveUser@

执行 **delete from tbls where tbl_id='xxx';**

10.35.16 如何关闭 Hive 客户端日志

问题

如何关闭 Hive 客户端的运行日志？

回答

步骤 1 使用 **root** 用户登录安装客户端的节点。

步骤 2 执行以下命令，切换到客户端安装目录，例如“/opt/Bigdata/client”。

cd /opt/Bigdata/client

步骤 3 执行以下命令配置环境变量。

source bigdata_env

步骤 4 根据集群认证模式，完成 Hive 客户端登录。

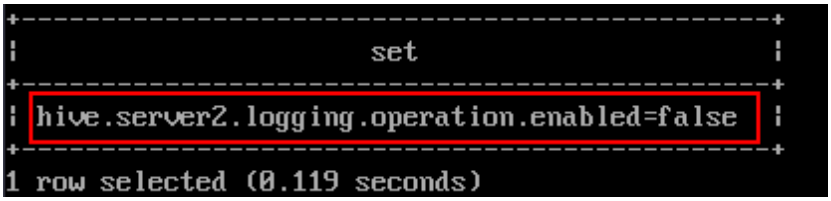
- 安全模式，则执行以下命令，完成用户认证并登录 Hive 客户端。
  kinit *组件业务用户*
  beeline
- 普通模式，则执行以下命令，登录 Hive 客户端。
  - 使用指定组件业务用户登录 Hive 客户端。
    beeline -n *组件业务用户*
  - 不指定组件业务用户登录 Hive 客户端，则会以当前操作系统用户登录。
    beeline

步骤 5 执行以下命令关闭客户端日志：

set hive.server2.logging.operation.enabled=false;

步骤 6 执行以下命令查看客户端日志是否已关闭，如下图所示即为关闭成功。

set hive.server2.logging.operation.enabled;



----结束

10.35.17 Hive 快删目录配置类问题

问题

在配置 MRS 多用户访问 OBS 细粒度权限的场景中，在 Hive 自定义配置中添加 OBS 快删目录的配置后，删除 Hive 表，执行结果为成功，但是 OBS 目录没有删掉。

回答

由于没有给用户配置快删目录的权限，导致数据不能被删除。需要修改用户对应的委托的 IAM 自定义策略，在策略内容上，配置 Hive 快删目录的权限。

10.35.18 Hive 配置类问题

- Hive SQL 执行报错：java.lang.OutOfMemoryError: Java heap space.

  解决方案：

  - 对于 MapReduce 任务，增大下列参数：

    set mapreduce.map.memory.mb=8192;

    set mapreduce.map.java.opts=-Xmx6554M;

    set mapreduce.reduce.memory.mb=8192;

    set mapreduce.reduce.java.opts=-Xmx6554M;

  - 对于 Tez 任务，增大下列参数：

    set hive.tez.container.size=8192;

- Hive SQL 对列名 as 为新列名后，使用原列名编译报错：Invalid table alias or column reference 'xxx'.

  解决方案：**set hive.cbo.enable=true;**

- Hive SQL 子查询编译报错：Unsupported SubQuery Expression 'xxx': Only SubQuery expressions that are top level conjuncts are allowed.

  解决方案：**set hive.cbo.enable=true;**

- Hive SQL 子查询编译报错：CalciteSubquerySemanticException [Error 10249]: Unsupported SubQuery Expression Currently SubQuery expressions are only allowed as Where and Having Clause predicates.

  解决方案：**set hive.cbo.enable=true;**

- Hive SQL 编译报错：Error running query: java.lang.AssertionError: Cannot add expression of different type to set.

  解决方案：**set hive.cbo.enable=false;**

- Hive SQL 执行报错：java.lang.NullPointerException at org.apache.hadoop.hive.ql.udf.generic.GenericUDAFComputeStats$GenericUDAFNumericStatsEvaluator.init.

  解决方案：**set hive.map.aggr=false;**

- Hive SQL 设置 hive.auto.convert.join = true（默认开启）和 hive.optimize.skewjoin=true 执行报错：ClassCastException org.apache.hadoop.hive.ql.plan.ConditionalWork cannot be cast to org.apache.hadoop.hive.ql.plan.MapredWork.

解决方案：**set hive.optimize.skewjoin=false;**

- Hive SQL 设置 hive.auto.convert.join=true（默认开启）、hive.optimize.skewjoin=true 和 hive.exec.parallel=true 执行报错：java.io.FileNotFoundException: File does not exist:xxx/reduce.xml.

  解决方案：

  - 方法一：切换执行引擎为 Tez，详情请参考切换 Hive 执行引擎为 Tez。
  - 方法二：**set hive.exec.parallel=false;**
  - 方法三：**set hive.auto.convert.join=false;**

- Hive on Tez 执行 Bucket 表 Join 报错：NullPointerException at org.apache.hadoop.hive.ql.exec.CommonMergeJoinOperator.mergeJoinComputeKeys

  解决方案：**set tez.am.container.reuse.enabled=false;**

11 使用 Hudi

11.1 快速入门

操作场景

本指南通过使用 spark-shell 简要介绍了 Hudi 功能。使用 Spark 数据源，我们将通过代码段展示如何插入和更新 Hudi 的默认存储类型数据集： COW 表。每次写操作之后，我们还将展示如何读取快照和增量数据。

前提条件

- 在 Manager 界面创建用户并添加 hadoop 和 hive 用户组，主组加入 hadoop。

操作步骤

步骤 1 下载并安装 Hudi 客户端，具体请参考安装客户端（3.x 及之后版本）章节。

说明

目前 Hudi 集成在 Spark2x 中，用户从 Manager 页面下载 Spark2x 客户端即可，例如客户端安装目录为：“/opt/client”。

步骤 2 使用 root 登录客户端安装节点，执行如下命令：

cd /opt/client

步骤 3 执行命令加载环境变量：

source bigdata_env

source Hudi/component_env

kinit *创建的用户*

说明

- 新创建的用户需要修改密码，更改密码后重新 kinit 登录。
- 普通模式（未开启 kerberos 认证）无需执行 **kinit** 命令。

步骤 4 使用 **spark-shell --master yarn-client**，引入 Hudi 包生成测试数据：

```
//引入需要的包
import org.apache.hudi.QuickstartUtils._
import scala.collection.JavaConversions._
import org.apache.spark.sql.SaveMode._
import org.apache.hudi.DataSourceReadOptions._
import org.apache.hudi.DataSourceWriteOptions._
import org.apache.hudi.config.HoodieWriteConfig._
//定义表名，存储路径，生成测试数据
val tableName = "hudi_cow_table"
val basePath = "hdfs://hacluster/tmp/hudi_cow_table"
val dataGen = new DataGenerator
val inserts = convertToStringList(dataGen.generateInserts(10))
val df = spark.read.json(spark.sparkContext.parallelize(inserts, 2))
```

步骤 5 写入 Hudi 表，模式为 OVERWRITE。

```
df.write.format("org.apache.hudi").
options(getQuickstartWriteConfigs).
option(PRECOMBINE_FIELD_OPT_KEY, "ts").
option(RECORDKEY_FIELD_OPT_KEY, "uuid").
option(PARTITIONPATH_FIELD_OPT_KEY, "partitionpath").
option(TABLE_NAME, tableName).
mode(Overwrite).
save(basePath)
```

步骤 6 查询 Hudi 表。

注册临时表并查询：

```
val roViewDF = spark.
read.
format("org.apache.hudi").
load(basePath + "/*/*/*/*")
roViewDF.createOrReplaceTempView("hudi_ro_table")
spark.sql("select fare, begin_lon, begin_lat, ts from  hudi_ro_table where fare >
20.0").show()
```

步骤 7 生成更新数据并更新 Hudi 表，模式为 APPEND。

```
val updates = convertToStringList(dataGen.generateUpdates(10))
val df = spark.read.json(spark.sparkContext.parallelize(updates, 1))
df.write.format("org.apache.hudi").
options(getQuickstartWriteConfigs).
option(PRECOMBINE FIELD OPT KEY, "ts").
option(RECORDKEY FIELD OPT KEY, "uuid").
option(PARTITIONPATH FIELD OPT KEY, "partitionpath").
option(TABLE NAME, tableName).
mode(Append).
save(basePath)
```

步骤 8 查询 Hudi 表增量数据。

- 重新加载：

  ```
  spark.
  read.
  format("org.apache.hudi").
  ```

```
load(basePath + "/*/*/*/*").
createOrReplaceTempView("hudi_ro_table")
```

- 进行增量查询：

```
val commits = spark.sql("select distinct(_hoodie_commit_time) as commitTime
from  hudi_ro_table order by commitTime").map(k => k.getString(0)).take(50)
val beginTime = commits(commits.length - 2)
val incViewDF = spark.
read.
format("org.apache.hudi").
option(VIEW_TYPE_OPT_KEY, VIEW_TYPE_INCREMENTAL_OPT_VAL).
option(BEGIN_INSTANTTIME_OPT_KEY, beginTime).
load(basePath);
incViewDF.registerTempTable("hudi_incr_table")
spark.sql("select `_hoodie_commit_time`, fare, begin_lon, begin_lat, ts from
hudi_incr_table where fare > 20.0").show()
```

步骤 9 进行指定时间点提交的查询。

```
val beginTime = "000"
val endTime = commits(commits.length - 2)
val incViewDF = spark.read.format("org.apache.hudi").
option(VIEW_TYPE_OPT_KEY, VIEW_TYPE_INCREMENTAL_OPT_VAL).
option(BEGIN_INSTANTTIME_OPT_KEY, beginTime).
option(END INSTANTTIME OPT KEY, endTime).
load(basePath);
incViewDF.registerTempTable("hudi incr table")
spark.sql("select ` hoodie commit time`, fare, begin lon, begin lat, ts from
hudi_incr_table where fare > 20.0").show()
```

步骤 10 删除数据。

- 准备删除的数据

```
val df = spark.sql("select uuid, partitionpath from hudi_ro_table limit 2")
val deletes = dataGen.generateDeletes(df.collectAsList())
```

- 执行删除操作

```
val df = spark.read.json(spark.sparkContext.parallelize(deletes, 2));
df.write.format("org.apache.hudi").
options(getQuickstartWriteConfigs).
option(OPERATION_OPT_KEY,"delete").
option(PRECOMBINE_FIELD_OPT_KEY, "ts").
option(RECORDKEY_FIELD_OPT_KEY, "uuid").
option(PARTITIONPATH_FIELD_OPT_KEY, "partitionpath").
option(TABLE_NAME, tableName).
mode(Append).
save(basePath);
```

- 重新查询数据

```
val roViewDFAfterDelete = spark.
read.
format("org.apache.hudi").
load(basePath + "/*/*/*/*")
roViewDFAfterDelete.createOrReplaceTempView("hudi_ro_table")
spark.sql("select uuid, partitionPath from hudi_ro_table").show()
```

----结束

11.2 基本操作

11.2.1 Hudi 表结构

Hudi 在写入数据时会根据设置的存储路径、表名、分区结构等属性生成 Hudi 表。

Hudi 表的数据文件，可以使用操作系统的文件系统存储，也可以使用 HDFS 这种分布式的文件系统存储。为了后续分析性能和数据的可靠性，一般使用 HDFS 进行存储。以 HDFS 存储来看，一个 Hudi 表的存储文件分为两类。

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	testcz	hadoop	0 B	Apr 25 15:32	0	0 B	.hoodie
drwxr-xr-x	testcz	hadoop	0 B	Apr 25 15:30	0	0 B	americas
drwxr-xr-x	testcz	hadoop	0 B	Apr 25 15:30	0	0 B	asia

- “.hoodie”文件夹中存放了对应的文件合并操作相关的日志文件。

drwxr-xr-x	admintest	hadoop	0 B	Mar 30 09:44	0	0 B	.aux
drwxr-xr-x	admintest	hadoop	0 B	Mar 30 11:45	0	0 B	.temp
-rw-r--r--	admintest	hadoop	4.58 KB	Mar 30 09:44	3	128 MB	20210330094435.deltacommit
-rw-r--r--	admintest	hadoop	0 B	Mar 30 09:44	3	128 MB	20210330094435.deltacommit.inflight
-rw-r--r--	admintest	hadoop	0 B	Mar 30 09:44	3	128 MB	20210330094435.deltacommit.requested

- 包含_partition_key 相关的路径是实际的数据文件和 metadata，按分区存储。

  Hudi 的数据文件使用 Parquet 文件格式的 base file 和 Avro 格式的 log file 存储。

-rw-r--r--	admintest	hadoop	93 B	Mar 30 09:44	3	128 MB	.hoodie_partition_metadata
-rw-r--r--	admintest	hadoop	441.77 KB	Mar 30 09:46	3	128 MB	2b4d098e-4dc8-4633-a22a-dc22f87c57d9-1_0-13-22_20210330094613.parquet
-rw-r--r--	admintest	hadoop	445.28 KB	Mar 30 09:44	3	128 MB	4010e8a8-1b20-4be7-8442-4e30af401e84-0_1-4-8_20210330094435.parquet

11.2.2 写操作指导

Hudi 当前支持用 Spark 和 Flink 作为写入引擎，当前版本 Flink 写入能力较弱不建议使用，后续版本社区会逐步增强。

11.2.2.1 批量写入

操作场景

Hudi 提供多种写入方式，具体见 hoodie.datasource.write.operation 配置项，这里主要介绍 UPSERT、INSERT 和 BULK_INSERT。

- INSERT（插入）： 该操作流程和 UPSERT 基本一致，但是不需要通过索引去查询具体更新的文件分区，因此它的速度比 UPSERT 快。当数据源不包含更新数据时建议使用该操作，若数据源中存在更新数据，则在数据湖中会出现重复数据。
- BULK_INSERT（批量插入）：用于初始数据集加载， 该操作会对主键进行排序后直接以写普通 parquet 表的方式插入 Hudi 表，该操作性能是最高的，但是无法控制小文件，而 UPSERT 和 INSERT 操作使用启发式方法可以很好的控制小文件。

- UPSERT（插入更新）： 默认操作类型。Hudi 会根据主键进行判断，如果历史数据存在则 update 如果不存在则 insert。因此在对于 CDC 之类几乎肯定包括更新的数据源，建议使用该操作。

说明

- 由于 INSERT 时不会对主键进行排序，所以初始化数据集不建议使用 INSERT。
- 在确定数据都为新数据时建议使用 INSERT，当存在更新数据时建议使用 UPSERT，当初始化数据集时建议使用 BULK_INSERT。

批量写入 Hudi 表

1. 引入 Hudi 包生成测试数据，参考快速入门章节的步骤 2 到步骤 4。
2. 写入 Hudi 表，写入命令中加入参数：option("hoodie.datasource.write.operation", "bulk_insert")，指定写入方式为 bulk_insert，如下所示：

```
df.write.format("org.apache.hudi").
options(getQuickstartWriteConfigs).
option("hoodie.datasource.write.precombine.field", "ts").
option("hoodie.datasource.write.recordkey.field", "uuid").
option("hoodie.datasource.write.partitionpath.field", "").
option("hoodie.datasource.write.operation", "bulk_insert").
option("hoodie.table.name", tableName).
option("hoodie.datasource.write.keygenerator.class",
"org.apache.hudi.keygen.NonpartitionedKeyGenerator").
option("hoodie.datasource.hive_sync.enable", "true").
option("hoodie.datasource.hive_sync.partition_fields", "").
option("hoodie.datasource.hive_sync.partition_extractor_class",
"org.apache.hudi.hive.NonPartitionedExtractor").
option("hoodie.datasource.hive_sync.table", tableName).
option("hoodie.datasource.hive_sync.use_jdbc", "false").
option("hoodie.bulkinsert.shuffle.parallelism", 4).
mode(Overwrite).
save(basePath)
```

说明

- 示例中各参数介绍请参考表 11-4。
- 使用 spark datasource 接口更新 Mor 表，Upsert 写入小数据量时可能触发更新数据的小文件合并，使在 Mor 表的读优化视图中能查到部分更新数据。
- 当 update 的数据对应的 base 文件是小文件时，insert 中的数据和 update 中的数据会被合在一起和 base 文件直接做合并产生新的 base 文件，而不是写 log。

分区设置操作

Hudi 支持多种分区方式，如多级分区、无分区、单分区、时间日期分区。用户可以根据实际需求选择合适的分区方式，接下来将详细介绍 Hudi 如何配置各种分区类型。

- 多级分区

  多级分区即指定多个字段为分区键，需要注意的配置项：

配置项	说明

配置项	说明
hoodie.datasource.write.partitionpath.field	配置为多个分区字段，例如：p1，p2，p3
hoodie.datasource.hive_sync.partition_fields	配置为 p1，p2，p3 和 hoodie.datasource.write.partitionpath.field 的分区字段保持一致
hoodie.datasource.write.keygenerator.class	配置为 org.apache.hudi.keygen.ComplexKeyGenerator
hoodie.datasource.hive_sync.partition_extractor_class	配置为 org.apache.hudi.hive.MultiPartKeysValueExtractor

- 无分区

  hudi 支持无分区表，需要注意的配置项：

配置项	说明
hoodie.datasource.write.partitionpath.field	配置为空
hoodie.datasource.hive_sync.partition_fields	配置为空
hoodie.datasource.write.keygenerator.class	配置为 org.apache.hudi.keygen.NonpartitionedKeyGenerator
hoodie.datasource.hive_sync.partition_extractor_class	配置为 org.apache.hudi.hive.NonPartitionedExtractor

- 单分区

  和多级分区类似，需要配置项：

配置项	说明
hoodie.datasource.write.partitionpath.field	配置为一个字段，例如：p
hoodie.datasource.hive_sync.partition_fields	配置为 p，和 hoodie.datasource.write.partitionpath.field 分区字段保持一致
hoodie.datasource.write.keygenerator.class	默认配置为 org.apache.hudi.keygen.SimpleKeyGenerator ，也可以不配置
hoodie.datasource.hive_sync.partition_extra	配置为

配置项	说明
ctor_class	org.apache.hudi.hive.MultiPartKeysValueExtractor

- 时间日期分区

  即指定 date 类型字段作为分区字段，需要注意的配置项：

配置项	说明
hoodie.datasource.write.partitionpath.field	配置为 date 类型字段比如 operationTime
hoodie.datasource.hive_sync.partition_fields	配置为 operationTime，和上面分区字段保持一致
hoodie.datasource.write.keygenerator.class	默认配置为 org.apache.hudi.keygen.SimpleKeyGenerator ，也可以不配置
hoodie.datasource.hive_sync.partition_extractor_class	配置 org.apache.hudi.hive.SlashEncodedDayPartitionValueExtractor

说明

SlashEncodedDayPartitionValueExtractor 存在以下约束：要求写入的日期格式为 yyyy/mm/dd。

- 分区排序：

配置项	说明
hoodie.bulkinsert.user.defined.partitioner.class	指定分区排序类，可自行定义排序方法，具体参考样例代码

说明

bulk_insert 默认字符排序，仅适用于 StringType 的主键。

11.2.2.2 流式写入

HoodieDeltaStreamer 流式写入

Hudi 自带 HoodieDeltaStreamer 工具支持流式写入，也可以使用 SparkStreaming 以微批的方式写入。HoodieDeltaStreamer 提供以下功能：

- 支持 Kafka，DFS 多种数据源接入 。
- 支持管理检查点、回滚和恢复，保证 exactly once 语义。
- 支持自定义转换操作。

示例：

准备配置文件 kafka-source.properties

```
#hudi 配置
hoodie.datasource.write.recordkey.field=id
hoodie.datasource.write.partitionpath.field=age
hoodie.upsert.shuffle.parallelism=100
#hive config
hoodie.datasource.hive_sync.table=hudimor_deltastreamer_partition
hoodie.datasource.hive_sync.partition_fields=age
hoodie.datasource.hive_sync.partition_extractor_class=org.apache.hudi.hive.MultiPar
tKeysValueExtractor
hoodie.datasource.hive_sync.use_jdbc=false
hoodie.datasource.hive_sync.support_timestamp=true
# Kafka Source topic
hoodie.deltastreamer.source.kafka.topic=hudimor_deltastreamer_partition
#checkpoint
hoodie.deltastreamer.checkpoint.provider.path=hdfs://hacluster/tmp/huditest/hudimor
_deltastreamer_partition
# Kafka props
# The kafka cluster we want to ingest from
bootstrap.servers= xx.xx.xx.xx:xx
auto.offset.reset=earliest
#auto.offset.reset=latest
group.id=hoodie-delta-streamer
offset.rang.limit=10000
```

指定 HoodieDeltaStreamer 执行参数（具体参数配置，请查看官网 https://hudi.apache.org/ ）执行如下命令：

spark-submit --master yarn

--jars /opt/hudi-java-examples-1.0.jar // 指定 spark 运行时需要的 hudi jars 路径

--driver-memory 1g

--executor-memory 1g --executor-cores 1 --num-executors 2 --conf spark.kryoserializer.buffer.max=128m

--driver-class-path /opt/client/Hudi/hudi/conf:/opt/client/Hudi/hudi/lib/*:/opt/client/Spark2x/spark/jars/*:/opt/hudi-examples-0.6.1-SNAPSHOT.jar:/opt/hudi-examples-0.6.1-SNAPSHOT-tests.jar // 指定 spark driver 需要的 hudi jars 路径

--class org.apache.hudi.utilities.deltastreamer.HoodieDeltaStreamer spark-internal

--props file:///opt/kafka-source.properties // 指定配置文件，注意：使用 yarn-cluster 模式提交任务时，请指定配置文件路径为 HDFS 路径。

--target-base-path /tmp/huditest/hudimor1_deltastreamer_partition // 指定 hudi 表路径

--table-type MERGE_ON_READ // 指定要写入的 hudi 表类型

--target-table hudimor_deltastreamer_partition // 指定 hudi 表名

--source-ordering-field name // 指定 hudi 表预合并列

--source-class org.apache.hudi.utilities.sources.JsonKafkaSource // 指定消费的数据源 为 JsonKafkaSource， 该参数根据不同数据源指定不同的 source 类

--schemaprovider-class com.xxx.bigdata.hudi.examples.DataSchemaProviderExample // 指定 hudi 表所需要的 schema

--transformer-class com.xxx.bigdata.hudi.examples.TransformerExample // 指定如何处理数据源拉取来的数据，可根据自身业务需求做定制

--enable-hive-sync // 开启 hive 同步，同步 hudi 表到 hive

--continuous // 指定流处理模式为连续模式

11.2.2.3 将 Hudi 表数据同步到 Hive

通过执行 run_hive_sync_tool.sh 可以将 Hudi 表数据同步到 Hive 中。

例如：需要将 HDFS 上目录为 hdfs://hacluster/tmp/huditest/hudimor1_deltastreamer_partition 的 Hudi 表同步为 Hive 表，表名为 table hive_sync_test3，使用 unite、country 和 state 为分区键，命令示例如下：

run_hive_sync_tool.sh --partitioned-by unite,country,state --base-path hdfs://hacluster/tmp/huditest/hudimor1_deltastreamer_partition --table hive_sync_test3 --partition-value-extractor org.apache.hudi.hive.MultiPartKeysValueExtractor --support-timestamp

表11-1 参数说明

命令	描述	必填	默认值
--database	Hive database 名称	N	default
--table	Hive 表名	Y	-
--base-file-format	文件格式 (PARQUET 或 HFILE)	N	PARQUET
--user	Hive 用户名	N	-
--pass	Hive 密码	N	-
--jdbc-url	Hive jdbc connect url	N	-
--base-path	待同步的 Hudi 表存储路径	Y	-
--partitioned-by	分区键-	N	-
--partition-value-extractor	分区类，需实现 PartitionValueExtractor ，可以从 HDFS 路径中提取分区值	N	SlashEncodedDayPartitionValueExtractor
--assume-date-partitioning	以 yyyy/mm/dd 进行分区从而支持向后兼容。	N	false
--use-pre-apache-input-	使用 com.uber.hoodie 包下的 InputFormat 替换	N	false

命令	描述	必填	默认值
format	org.apache.hudi 包下的。除了从 com.uber.hoodie 迁移项目至 org.apache.hudi 外请勿使用。		
--use-jdbc	使用 Hive jdbc 连接	N	true
--auto-create-database	自动创建 Hive database	N	true
--skip-ro-suffix	注册时跳过读取_ro 后缀的读优化视图	N	false
--use-file-listing-from-metadata	从 Hudi 的元数据中获取文件列表	N	false
--verify-metadata-file-listing	根据文件系统验证 Hudi 元数据中的文件列表	N	false
--help、-h	查看帮助	N	false
--support-timestamp	将原始类型中'INT64'的 TIMESTAMP_MICROS 转换为 Hive 的 timestamp	N	false
--decode-partition	如果分区在写入过程中已编码，则解码分区值	N	false
--batch-sync-num	指定每批次同步 hive 的分区数	N	1000

说明

Hive Sync 时会判断表不存在时建外表并添加分区，表存在时对比表的 schema 是否存在差异，存在则替换，对比分区是否有新增，有则添加分区。

因此使用 hive sync 时有以下约束：

- 写入数据 Schema 只允许增加字段，不允许修改、删除字段。
- 分区目录只能新增，不会删除。
- Overwrite 覆写 Hudi 表不支持同步覆盖 Hive 表。
- Hudi 同步 Hive 表时，不支持使用 timestamp 类型作为分区列。
- 使用此脚本同步 Hive 时基于安全考虑必须使用 jdbc 方式同步，即--use-jdbc 必须为 true。

11.2.3 读操作指导

Hudi 的读操作，作用于 Hudi 的三种视图之上，可以根据需求差异选择合适的视图进行查询。

Hudi 支持多种查询引擎 Spark 和 Hive，具体支持矩阵见表 11-2 和表 11-3。

表11-2 cow 表

查询引擎	实时视图/读优化视图	增量视图
Hive	Y	Y
Spark（SparkSQL）	Y	Y
Spark（SparkDataSource API）	Y	Y

表11-3 mor 表

查询引擎	实时视图	增量视图	读优化视图
Hive	Y	Y	Y
Spark（SparkSQL）	Y	Y	Y
Spark（SparkDataSource API）	Y	Y	Y

注意

- 当前 Hudi 使用 Spark datasource 接口读取时，不支持分区推断能力。比如 bootstrap 表使用 datasource 接口查询时，可能出现分区字段不显示，或者显示为 null 的情况。
- 增量视图，需设置 **set hoodie.hudicow.consume.mode = INCREMENTAL;**，但该参数仅限于增量视图查询，不能用于 Hudi 表的其他类型查询，和其他表的查询。 恢复配置可设置 **set hoodie.hudicow.consume.mode = SNAPSHOT;**或任意值。

11.2.3.1 cow 表视图读取

- 实时视图读取（Hive，SparkSQL 为例）：直接读取 Hive 里面存储的 Hudi 表即可。

```
select count(*) from test;
```

- 实时视图读取（Spark dataSource API 为例）：和读普通的 dataSource 表类似。必须指定查询类型 QUERY_TYPE_OPT_KEY 为 QUERY_TYPE_SNAPSHOT_OPT_VAL。

```
spark.read.format("hudi")
.option(QUERY_TYPE_OPT_KEY, QUERY_TYPE_SNAPSHOT_OPT_VAL) // 指定查询类型为实时视图模式
```

```
.load("/tmp/default/cow_bugx/*/*/*/*") // 指定读取的 hudi 表路径，当前表有 3 级分区
.createTempView("mycall")
spark.sql("select * from mycall").show(100)
```

- 增量视图读取（Hive 为例）：

```
set hoodie.test.consume.mode=INCREMENTAL;  //设置增量读取模式
set hoodie.test.consume.max.commits=3;  // 指定最大消费的 commits 数量
set hoodie.test.consume.start.timestamp=20201227153030;  // 指定初始增量拉取
commit
select count(*) from default.test where `_hoodie_commit_time`>'20201227153030';
// 这个过滤条件必须加且值为初始增量拉取的 commit。
```

- 增量视图读取（SparkSQL 为例）：

```
set hoodie.test.consume.mode=INCREMENTAL;  //设置增量读取模式
set hoodie.test.consume.start.timestamp=20201227153030;  // 指定初始增量拉取
commit
set hoodie.test.consume.end.timestamp=20210308212318;  // 指定增量拉取结束 commit，
如果不指定的话采用最新的 commit
select count(*) from default.test where `_hoodie_commit_time`>'20201227153030';
// 这个过滤条件必须加且值为初始增量拉取的 commit。
```

- 增量视图读取（Spark dataSource API 为例）：

  必须指定查询类型 QUERY_TYPE_OPT_KEY 为增量模式 QUERY_TYPE_INCREMENTAL_OPT_VAL

```
spark.read.format("hudi")
.option(QUERY_TYPE_OPT_KEY, QUERY_TYPE_INCREMENTAL_OPT_VAL) // 指定查询类型为增量
模式
.option(BEGIN_INSTANTTIME_OPT_KEY, "20210308212004")  // 指定初始增量拉取 commit
.option(END_INSTANTTIME_OPT_KEY, "20210308212318")  // 指定增量拉取结束 commit
.load("/tmp/default/cow_bugx/*/*/*/*")  // 指定读取的 hudi 表路径，当前表有 3 级分区
.createTempView("mycall")  // 注册为 spark 临时表
spark.sql("select * from mycall where `_hoodie_commit_time`>'20210308211131'")
// 开始查询，和 hive 增量查询语句一样
.show(100, false)
```

- 读优化视图：cow 表读优化视图等同于实时视图。

11.2.3.2 mor 表视图读取

mor 表同步给 Hive 后，会在 Hive 表中同步出：“表名+后缀_rt”和“表名+后缀_ro”两张表。其中后缀为 rt 表代表实时视图，后缀为 ro 的表代表读优化视图。例如：同步给 Hive 的 hudi 表名为 test， 同步 Hive 后 hive 表中多出两张表分别为 test_rt,和 test_ro。

- 实时视图读取（Hive，SparkSQL 为例）：直接读取 Hive 里面存储的后缀为_rt 的 hudi 表即可。

```
select count(*) from test_rt;
```

- 实时视图读取（Spark dataSource API 为例）：和 cow 表一样，请参考 cow 表相关操作。
- 增量视图读取（hive 为例）：

```
set hive.input.format=org.apache.hudi.hadoop.hive.HoodieCombineHiveInputFormat;
// sparksql 不需要指定
set hoodie.test.consume.mode=INCREMENTAL;
set hoodie.test.consume.max.commits=3;
```

```
set hoodie.test.consume.start.timestamp=20201227153030;
select count(*) from default.test_rt where
`_hoodie_commit_time`>'20201227153030';
```

- 增量视图读取（SparkSQL 为例）：

```
set hoodie.test.consume.mode=INCREMENTAL;
set hoodie.test.consume.start.timestamp=20201227153030;  // 指定初始增量拉取
commit
set hoodie.test.consume.end.timestamp=20210308212318;  // 指定增量拉取结束 commit，
如果不指定的话采用最新的 commit
select count(*) from default.test_rt where
`_hoodie_commit_time`>'20201227153030';
```

- 增量视图（Spark dataSource API 为例）：和 cow 表一样，请参考 cow 表相关操作。
- 读优化视图读取（Hive，SparkSQL 为例）：直接读取 Hive 里面存储的后缀为_ro 的 hudi 表即可。

```
select count(*) from test_ro;
```

- 读优化视图读取（Spark dataSource API 为例）：和读普通的 dataSource 表类似。

  必须指定查询类型 QUERY_TYPE_OPT_KEY 为 QUERY_TYPE_READ_OPTIMIZED_OPT_VAL

```
spark.read.format("hudi")
.option(QUERY_TYPE_OPT_KEY, QUERY_TYPE_READ_OPTIMIZED_OPT_VAL) // 指定查询类型为
读优化视图
.load("/tmp/default/mor_bugx/*/*/*/*") // 指定读取的 hudi 表路径，当前表有 3 级分区
.createTempView("mycall")
spark.sql("select * from mycall").show(100)
```

11.2.4 数据管理维护

11.2.4.1 Clustering

什么是 Clustering

即数据布局，该服务可重新组织数据以提高查询性能，也不会影响摄取速度。

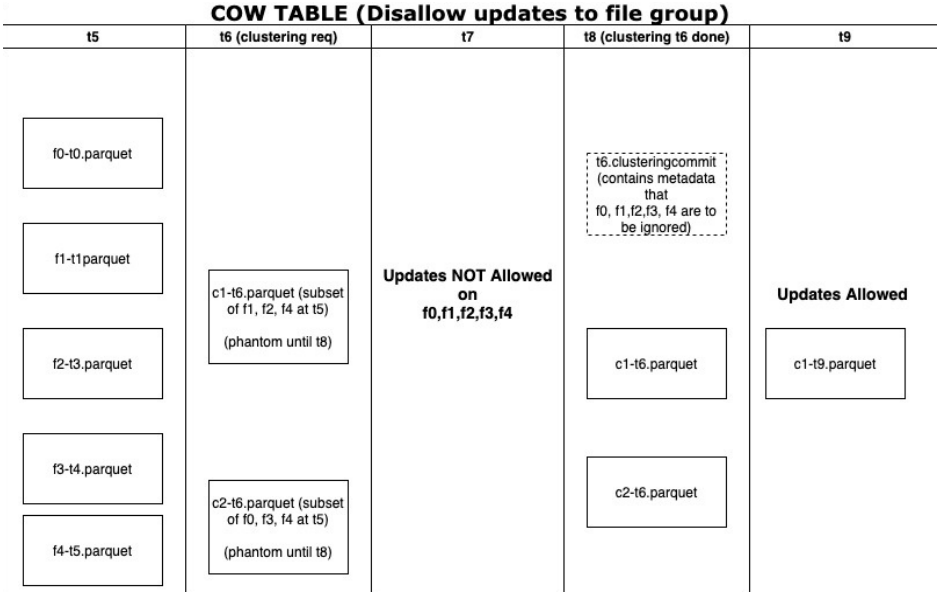
Clustering 架构

Hudi 通过其写入客户端 API 提供了不同的操作，如 insert/upsert/bulk_insert 来将数据写入 Hudi 表。为了能够在文件大小和入湖速度之间进行权衡，Hudi 提供了一个 hoodie.parquet.small.file.limit 配置来设置最小文件大小。用户可以将该配置设置为“0”，以强制新数据写入新的文件组，或设置为更高的值以确保新数据被“填充”到现有小的文件组中，直到达到指定大小为止，但其会增加摄取延迟。

为能够支持快速摄取的同时不影响查询性能，我们引入了 Clustering 服务来重写数据以优化 Hudi 数据湖文件的布局。

Clustering 服务可以异步或同步运行，Clustering 会添加了一种新的 REPLACE 操作类型，该操作类型将在 Hudi 元数据时间轴中标记 Clustering 操作。

Clustering 服务基于 Hudi 的 MVCC 设计，允许继续插入新数据，而 Clustering 操作在后台运行以重新格式化数据布局，从而确保并发读写者之间的快照隔离。



总体而言 Clustering 分为两个部分：

- 调度 Clustering：使用可插拔的 Clustering 策略创建 Clustering 计划。
  a. 识别符合 Clustering 条件的文件：根据所选的 Clustering 策略，调度逻辑将识别符合 Clustering 条件的文件。
  b. 根据特定条件对符合 Clustering 条件的文件进行分组。每个组的数据大小应为 targetFileSize 的倍数。分组是计划中定义的"策略"的一部分。此外还有一个选项可以限制组大小，以改善并行性并避免混排大量数据。
  c. 将 Clustering 计划以 avro 元数据格式保存到时间线。
- 执行 Clustering：使用执行策略处理计划以创建新文件并替换旧文件。
  a. 读取 Clustering 计划，并获得 ClusteringGroups，其标记了需要进行 Clustering 的文件组。
  b. 对于每个组使用 strategyParams 实例化适当的策略类（例如：sortColumns），然后应用该策略重写数据。
  c. 创建一个 REPLACE 提交，并更新 HoodieReplaceCommitMetadata 中的元数据。

如何执行 Clustering

1. 同步执行 Clustering 配置。

   在写入时加上配置参数：

   option("hoodie.clustering.inline", "true").

   option("hoodie.clustering.inline.max.commits", "4").

   option("hoodie.clustering.plan.strategy.target.file.max.bytes", "1073741824").

   option("hoodie.clustering.plan.strategy.small.file.limit", "629145600").

   option("hoodie.clustering.plan.strategy.sort.columns", "column1,column2").
2. 异步执行 Clustering：

spark-submit --master yarn --class org.apache.hudi.utilities.HoodieClusteringJob /opt/client/Hudi/hudi/lib/hudi-utilities*.jar --schedule --base-path <table_path> --table-name <table_name> --props /tmp/clusteringjob.properties --spark-memory 1g

spark-submit --master yarn --driver-memory 16G --executor-memory 12G --executor-cores 4 --num-executors 4 --class org.apache.hudi.utilities.HoodieClusteringJob /opt/client/Hudi/hudi/lib/hudi-utilities*.jar --base-path <table_path> --instant-time 20210605112954 --table-name <table_name> --props /tmp/clusteringjob.properties --spark-memory 12g

说明

clusteringjob.properties 中为用户自定义的 clustering 相关配置。

例如：

hoodie.clustering.plan.strategy.target.file.max.bytes=1073741824

hoodie.clustering.inline.max.commits=4

详细配置请参考配置参考。

注意

1. 分区表执行 Clustering 重组范围默认只对排序最大的 2 个分区，其他则依赖用户自定义策略。
2. Clustering 的排序列不允许值存在 null，是 spark rdd 的限制。
3. 当 target.file.max.bytes 的值较大时，启动 Clustering 执行需要提高--spark-memory，否则会导致 executor 内存溢出。
4. 当前 clean 不支持清理 Clustering 失败后的垃圾文件。
5. Clustering 后可能出现新文件大小不等引起数据倾斜的情况。
6. cluster 不支持和 upsert 并发。

11.2.4.2 Cleaning

Cleaning 用于清理不再需要的版本数据。

Hudi 使用 Cleaner 后台作业，不断清除不需要的旧得版本的数据。通过配置 hoodie.cleaner.policy 和 hoodie.cleaner.commits.retained 可以使用不同的清理策略和保存的 commit 数量。

执行 cleaning 有两种方式：

- 使用 hudi-cli

  cleans run --sparkMaster yarn --hoodieConfigs 'hoodie.cleaner.policy=KEEP_LATEST_COMMITS,hoodie.cleaner.commits.retained=1,hoodie.cleaner.incremental.mode=false,hoodie.keep.max.commits=3,hoodie.keep.min.commits=2'
- 使用 API

spark-submit --master yarn --jars /opt/client/Hudi/hudi/lib/hudi-client-common-*xxx*.jar --class org.apache.hudi.utilities.HoodieCleaner /opt/client/Hudi/hudi/lib/hudi-utilities_*xxx*.jar --target-base-path /tmp/default/tb_test_mor

更多 Cleaning 的参数配置可以参考配置参考。

11.2.4.3 Compaction

Compaction 用于合并 mor 表 Base 和 Log 文件。

对于 Merge-On-Read 表，数据使用列式 Parquet 文件和行式 Avro 文件存储，更新被记录到增量文件，然后进行同步/异步 compaction 生成新版本的列式文件。Merge-On-Read 表可减少数据摄入延迟，因而进行不阻塞摄入的异步 Compaction 很有意义。

异步 Compaction 会进行如下两个步骤

1. 调度 Compaction：由入湖作业完成，在这一步，Hudi 扫描分区并选出待进行 compaction 的 FileSlice，最后 CompactionPlan 会写入 Hudi 的 Timeline。
2. 执行 Compaction：一个单独的进程/线程将读取 CompactionPlan 并对 FileSlice 执行 Compaction 操作。

使用 Compaction 的方式分为同步和异步两种：

同步方式包括：

- 在使用 HoodieDeltaStreamer 将上游（Kafka/DFS）数据写入 hudi 数据集时参数--disable-compaction 默认为 false，自动进行 compaction 操作。
- 使用 datasource 在写入时指定参数：

  option("hoodie.compact.inline", "true").

  option("hoodie.compact.inline.max.delta.commits", "2").

异步方式包括：

- 使用 hudi-cli 进行异步 compaction

  调度 compaction：

  compaction schedule --hoodieConfigs 'hoodie.compaction.strategy=org.apache.hudi.table.action.compact.strategy.BoundedIOCompactionStrategy,hoodie.compaction.target.io=1,hoodie.compact.inline.max.delta.commits=1'

  执行 compaction：

  compaction run --parallelism 100 --sparkMemory 1g --retry 1 --compactionInstant 20210602101315 --hoodieConfigs 'hoodie.compaction.strategy=org.apache.hudi.table.action.compact.strategy.BoundedIOCompactionStrategy,hoodie.compaction.target.io=1,hoodie.compact.inline.max.delta.commits=1' --propsFilePath hdfs://hacluster/tmp/default/tb_test_mor/.hoodie/hoodie.properties --schemaFilePath /tmp/default/tb_test_mor/.hoodie/compact_tb_base.json

- 使用 API 进行异步 compaction

  调度 compaction：

  spark-submit --master yarn --jars /opt/client/Hudi/hudi/lib/hudi-client-common-*xxx*.jar --class org.apache.hudi.utilities.HoodieCompactor

/opt/client/Hudi/hudi/lib/hudi-utilities_*xxx*.jar --base-path /tmp/default/tb_test_mor --table-name tb_test_mor --parallelism 100 --spark-memory 1G --schema-file /tmp/default/tb_test_mor/.hoodie/compact_tb_base.json --instant-time 20210602141810 --schedule --strategy org.apache.hudi.table.action.compact.strategy.UnBoundedCompactionStrategy

执行 compaction：

spark-submit --master yarn --jars /opt/client/Hudi/hudi/lib/hudi-client-common-*xxx*.jar --class org.apache.hudi.utilities.HoodieCompactor /opt/client/Hudi/hudi/lib/hudi-utilities_*xxx*.jar --base-path /tmp/default/tb_test_mor --table-name tb_test_mor --parallelism 100 --spark-memory 1G --schema-file /tmp/default/tb_test_mor/.hoodie/compact_tb_base.json --instant-time 20210602141810

说明

- 使用 hudi-cli 进行调度 compaction 时，不需要指定 instant-time，系统会自动生成并在调度成功后返回，只需在执行时传入该参数即可。
- schema-file 需要用户手动编辑当前 Hudi 表的表结构 schema 文件上传到服务器上（可以使用最近一次“.commit”文件中的 schema）。

11.2.4.4 Savepoint

savepoint 用于保存并还原自定义的版本数据。

Hudi 提供的 savepoint 就可以将不同的 commit 保存起来以便清理程序不会将其删除，后续可以使用 Rollback 进行恢复。

使用 hudi-cli 管理 savepoint 主要包括：

- 创建 savepoint

  savepoint create --commit <commit_time>
- 回滚 savepoint

  savepoint rollback --savepoint <savepoint_time>
- 刷新 savepoint

  savepoints refresh
- 查看所有存在的 savepoint

  savepoints show

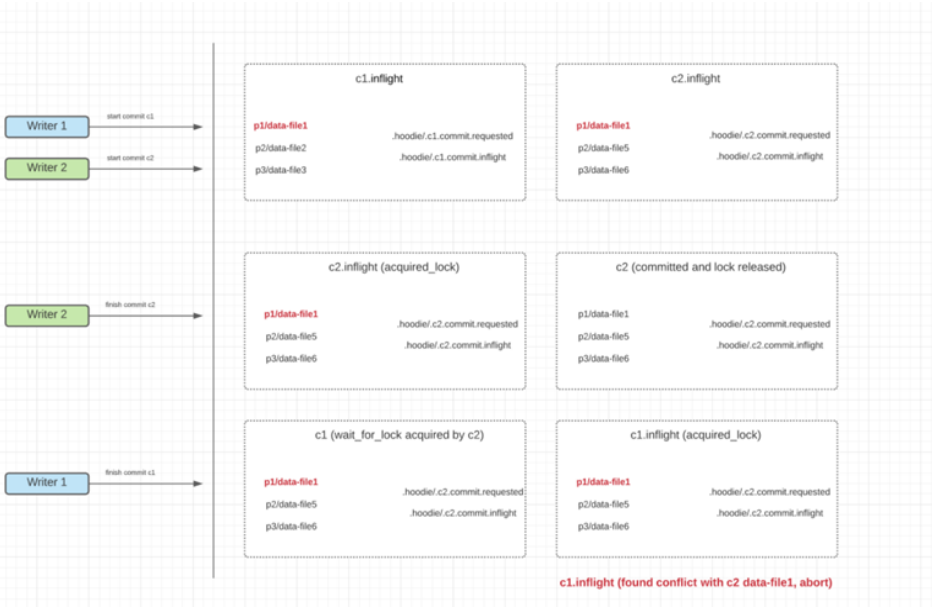
说明

MoR 表暂时不支持 savepoint。

11.2.4.5 单表并发写

Hudi 单表并发写实现方案

1. 使用外部服务（Zookeeper/Hive MetaStore）作为分布式互斥锁服务。
2. 允许并发写入文件，但是不允许并发提交 commit，提交 commit 操作封装到事务中。
3. 提交 commit 时，执行冲突检查：若本次提交的 commit 中，修改的文件列表，与本次 instanceTime 之后的 commit 存在重叠文件，则提交失败，本次写入无效。



使用并发机制需要注意问题

1. INSERT、BULK_INSERT 类型的操作，Hudi 当前并发机制无法保证写入后表主键唯一， 这个需要用户自己来保证。
2. 增量查询问题：数据消费以及 Checkpoint 可能会乱序，多个并发写操作在不同的时间点完成。
3. 并发写需要在启用并发写特性后支持并发，未开启时不支持并发写入。

如何使用并发机制

1. 启用并发写入机制。

   hoodie.write.concurrency.mode=optimistic_concurrency_control

   hoodie.cleaner.policy.failed.writes=LAZY

2. 设置并发锁方式。

   Hive MetaStore：

   hoodie.write.lock.provider=org.apache.hudi.hive.HiveMetastoreBasedLockProvider

   hoodie.write.lock.hivemetastore.database=<*database_name*>

   hoodie.write.lock.hivemetastore.table=<*table_name*>

   Zookeeper：

   hoodie.write.lock.provider=org.apache.hudi.client.transaction.lock.ZookeeperBasedLockProvider

   hoodie.write.lock.zookeeper.url=<*zookeeper_url*>

   hoodie.write.lock.zookeeper.port=<*zookeeper_port*>

   hoodie.write.lock.zookeeper.lock_key=<*table_name*>

   hoodie.write.lock.zookeeper.base_path=<*table_path*>

   更多配置参数请参考配置参考。

> **注意**
>
> 当设置 cleaner policy 为 Lazy 时，本次写入仅能关注到自己写入的文件是否过期，不能检查并清理历史写入产生的垃圾文件，即在并发场景下，无法自动清理垃圾文件。

11.2.5 Hudi 客户端使用

11.2.5.1 使用 Hudi-Cli.sh 操作 Hudi 表

前提条件

- 对于开启了 Kerberos 认证的安全模式集群，已在集群 FusionInsight Manager 界面创建一个用户并关联“hadoop”和“hive”用户组。
- 已下载并安装 Hudi 集群客户端。

基础操作

1. 使用 **root** 用户登录集群客户端节点，执行如下命令：

   cd *{客户端安装目录}*

   source bigdata_env

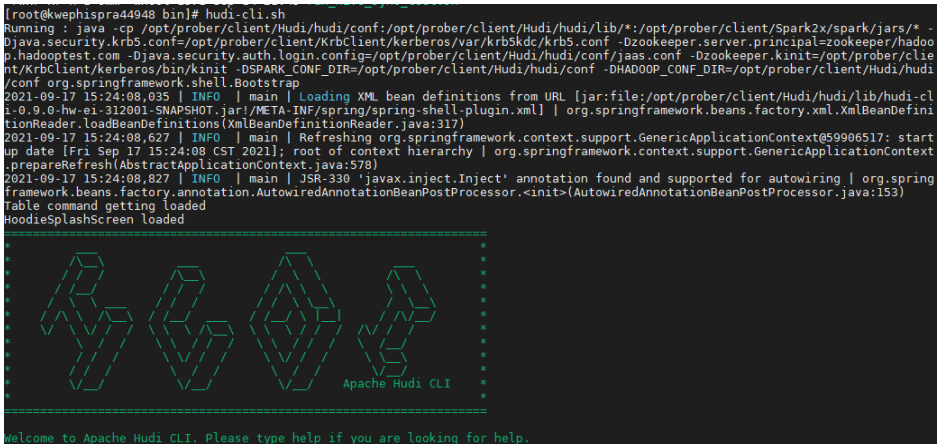
   source Hudi/component_env

   kinit *创建的用户*

2. 执行 **hudi-cli.sh** 进入 Hudi 客户端，

   cd *{客户端安装目录}*/**Hudi/hudi/bin/**

   ./hudi-cli.sh



3. 即可执行各种 Hudi 命令，执行示例（仅部分命令，全部命令请参考 Hudi 官网：https://hudi.apache.org/docs/quick-start-guide/）：

   - 查看帮助：

     help //查看 hudi-cli 的所有命令

     help '*command*' //查看某一个命令的帮助及参数列表。

   - 连接表：

     connect --path '*/tmp/huditest/test_table*'

- 查看表信息：

  desc
- 查看 compaction 计划：

  compactions show all
- 查看 clean 计划：

  cleans show
- 执行 clean：

  cleans run
- 查看 commit 信息：

  commits show
- 查看 commit 写入的分区：

  commit showpartitions --commit *20210127153356*

说明

20210127153356 表示 commit 的时间戳，下同。

- 查看指定 commit 写入的文件：

  commit showfiles --commit *20210127153356*
- 比较两个表的 commit 信息差异：

  commits compare --path */tmp/hudimor/mytest100*
- rollback 指定提交（rollback 每次只允许 rollback 最后一次 commit）：

  commit rollback --commit *20210127164905*
- compaction 调度：

  compaction schedule *--hoodieConfigs 'hoodie.compaction.strategy=org.apache.hudi.table.action.compact.strategy.BoundedIOCompactionStrategy,hoodie.compaction.target.io=1,hoodie.compact.inline.max.delta.commits=1'*
- 执行 compaction

  compaction run *--parallelism 100 --sparkMemory 1g --retry 1 --compactionInstant 20210602101315 --hoodieConfigs 'hoodie.compaction.strategy=org.apache.hudi.table.action.compact.strategy.BoundedIOCompactionStrategy,hoodie.compaction.target.io=1,hoodie.compact.inline.max.delta.commits=1' --propsFilePath hdfs://hacluster/tmp/default/tb_test_mor/.hoodie/hoodie.properties --schemaFilePath /tmp/default/tb_test_mor/.hoodie/compact_tb_base.json*
- 创建 savepoint

  savepoint create --commit *20210318155750*
- 回滚指定的 savepoint

  savepoint rollback --savepoint *20210318155750*

⚠ 注意

1. 若 commit 写入导致元数据冲突异常，执行 commit rollback、savepoint rollback 能回退数据，但不能回退 Hive 元数据，只能删除 Hive 表然后手动进行同步刷新。
2. commit rollback 只能回退当前最新的一个 commit，savepoint rollback 只能回退到最新的一个 savepoint。二者均不能随意指定进行回退。

11.2.6 配置参考

本章节介绍 Hudi 重要配置的详细信息，更多配置请参考 hudi 官网：http://hudi.apache.org/cn/docs/configurations.html。

11.2.6.1 写入操作配置

表11-4 写入操作重要配置项

参数	描述	默认值
hoodie.datasource.write.table.name	指定写入的 hudi 表名。	无
hoodie.datasource.write.operation	写 hudi 表指定的操作类型，当前支持 upsert、delete、insert、bulk_insert 等方式。 • upsert：更新插入混合操作 • delete：删除操作 • insert：插入操作 • bulk_insert： 用于初始建表导入数据， 注意初始建表禁止使用 upsert、insert 方式 • insert_overwrite：对静态分区执行 insert overwrite • insert_overwrite_table：动态分区执行 insert overwrite，该操作并不会立刻删除全表做 overwrite，会逻辑上重写 hudi 表的元数据，无用数据后续由 hudi 的 clean 机制清理。效率比 bulk_insert + overwrite 高	upsert
hoodie.datasource.write.table.type	指定 hudi 表类型，一旦这个表类型被指定，后续禁止修改该参数，可选值 MERGE_ON_READ。	COPY_ON_WRITE
hoodie.datasource.write.p	该值用于在写之前对具有相同的	ts

参数	描述	默认值
recombine.field	key 的行进行合并去重。	
hoodie.datasource.write.payload.class	在更新过程中，该类用于提供方法将要更新的记录和更新的记录做合并，该实现可插拔，如要实现自己的合并逻辑，可自行编写。	org.apache.hudi.OverwriteWithLatestAvroPayload
hoodie.datasource.write.recordkey.field	用于指定 hudi 的主键，hudi 表要求有唯一主键。	uuid
hoodie.datasource.write.partitionpath.field	用于指定分区键，该值配合 hoodie.datasource.write.keygenerator.class 使用可以满足不同的分区场景。	partitionpath
hoodie.datasource.write.hive_style_partitioning	用于指定分区方式是否和 hive 保持一致，建议该值设置为 true。	false
hoodie.datasource.write.keygenerator.class	配合 hoodie.datasource.write.partitionpath.field，hoodie.datasource.write.recordkey.field 产生主键和分区方式。	org.apache.hudi.keygen.SimpleKeyGenerator

11.2.6.2 同步 hive 表配置

参数	描述	默认值
hoodie.datasource.hive_sync.enable	是否同步 hudi 表信息到 hive metastore。 **注意** **建议该值设置为 true，统一使用 hive 管理 hudi 表。**	false
hoodie.datasource.hive_sync.database	要同步给 hive 的数据库名。	default
hoodie.datasource.hive_sync.table	要同步给 hive 的表名，建议这个值和 hoodie.datasource.write.table.name 保证一致。	无
hoodie.datasource.hive_sync.username	同步 hive 时，指定的用户名。	hive
hoodie.datasource.hive_sync.password	同步 hive 时，指定的密码。	hive

参数	描述	默认值
hoodie.datasource.hive_sync.jdbcurl	连接 hive jdbc 指定的连接。	jdbc:hive2://localhost:10000
hoodie.datasource.hive_sync.use_jdbc	是否使用 hive jdbc 方式连接 hive 同步 hudi 表信息。建议该值设置为 false，设置为 false 后 jdbc 连接相关配置无效。	true
hoodie.datasource.hive_sync.partition_fields	用于决定 hive 分区列。	``
hoodie.datasource.hive_sync.partition_extractor_class	用于提取 hudi 分区列值，将其转换成 hive 分区列。	org.apache.hudi.hive.SlashEncodedDayPartitionValueExtractor
hoodie.datasource.hive_sync.support_timestamp	当 hudi 表存在 timestamp 类型字段时，需指定此参数为 true，以实现同步 timestamp 类型到 hive 元数据中。该值默认为 false，默认将 timestamp 类型同步为 bigInt，默认情况可能导致使用 sql 查询包含 timestamp 类型字段的 hudi 表出现错误。	true

11.2.6.3 index 相关配置

参数	描述	默认值
hoodie.index.class	用户自定义索引的全路径名，索引类必须为 HoodieIndex 的子类，当指定该配置时，其会优先于 hoodie.index.type 配置。	“”
hoodie.index.type	使用的索引类型，默认为布隆过滤器。可能的选项是[BLOOM | HBASE | GLOBAL_BLOOM | SIMPLE | GLOBAL_SIMPLE] 。 布隆过滤器消除了对外部系统的依赖，并存储在 Parquet 数据文件的页脚中。	BLOOM
hoodie.index.bloom.num_entries	存储在布隆过滤器中的条目数。假设 maxParquetFileSize 为 128MB，averageRecordSize 为 1024B，因此，一个文件中的记录总数约为 130K。 默认值（60000）大约是此近似值的一	60000

参数	描述	默认值
	半。 注意 将此值设置得太低，将产生很多误报，并且索引查找将必须扫描比其所需的更多的文件；如果将其设置得非常高，将线性增加每个数据文件的大小（每 50000 个条目大约 4KB）。	
hoodie.index.bloom.fpp	根据条目数允许的错误率。 用于计算应为布隆过滤器分配多少位以及哈希函数的数量。通常将此值设置得很低（默认值：0.000000001），在磁盘空间上进行权衡以降低误报率。	0.000000001
hoodie.bloom.index.parallelism	索引查找的并行度，其中涉及 Spark Shuffle。 默认情况下，根据输入的工作负载特征自动计算的。	0
hoodie.bloom.index.prune.by.ranges	为 true 时，从文件框定信息，可以加快索引查找的速度。 如果键具有单调递增的前缀，例如时间戳，则特别有用。	true
hoodie.bloom.index.use.caching	为 true 时，将通过减少用于计算并行度或受影响分区的 IO 来缓存输入的 RDD 以加快索引查找。	true
hoodie.bloom.index.use.treebased.filter	为 true 时，启用基于间隔树的文件过滤优化。与暴力模式相比，此模式可根据键范围加快文件过滤速度。	true
hoodie.bloom.index.bucketized.checking	为 true 时，启用了桶式布隆过滤。这减少了在基于排序的布隆索引查找中看到的偏差。	true
hoodie.bloom.index.keys.per.bucket	仅在启用 bloomIndexBucketizedChecking 并且索引类型为 bloom 的情况下适用。 此配置控制“存储桶”的大小，该大小可跟踪对单个文件进行的记录键检查的次数，并且是分配给执行布隆过滤器查找的每个分	10000000

参数	描述	默认值
	区的工作单位。 较高的值将分摊将布隆过滤器读取到内存的固定成本。	
hoodie.bloom.index.update.partition.path	仅在索引类型为GLOBAL_BLOOM 时适用。 为 true 时，当对一个已有记录执行包含分区路径的更新操作时，将会导致把新记录插入到新分区，而把原有记录从旧分区里删除。为 false 时，只对旧分区的原有记录进行更新。	false
hoodie.index.hbase.zkquorum	仅在索引类型为 HBASE 时适用。要连接的 HBase ZK Quorum URL。	必填
hoodie.index.hbase.zkport	仅在索引类型为 HBASE 时适用。要连接的 HBase ZK Quorum 端口。	必填
hoodie.index.hbase.zknode.path	仅在索引类型为 HBASE 时适用。这是根 znode，它将包含 HBase 创建及使用的所有 znode。	必填
hoodie.index.hbase.table	仅在索引类型为 HBASE 时适用。HBase 表名称，用作索引。Hudi 将 row_key 和 [partition_path, fileID, commitTime]映射存储在表中。	必填

11.2.6.4 存储配置

参数	描述	默认值
hoodie.parquet.max.file.size	Hudi 写阶段生成的 parquet 文件的目标大小。对于 DFS，这需要与基础文件系统块大小保持一致，以实现最佳性能。	120 * 1024 * 1024 byte
hoodie.parquet.block.size	parquet 页面大小，页面是 parquet 文件中的读取单位，在一个块内，页面被分别压缩。	120 * 1024 * 1024 byte
hoodie.parquet.compression.ratio	当 Hudi 尝试调整新 parquet 文件的大小时，预期对 parquet 数据进行压缩的比例。 如果	0.1

参数	描述	默认值
	bulk_insert 生成的文件小于预期大小，请增加此值。	
hoodie.parquet.compression.codec	parquet 压缩编解码方式名称，默认值为 gzip。可能的选项是[gzip | snappy | uncompressed | lzo]	gzip
hoodie.logfile.max.size	LogFile 的最大值。这是在将日志文件移到下一个版本之前允许的最大值。	1GB
hoodie.logfile.data.block.max.size	LogFile 数据块的最大值。这是允许将单个数据块附加到日志文件的最大值。 这有助于确保附加到日志文件的数据被分解为可调整大小的块，以防止发生 OOM 错误。此大小应大于 JVM 内存。	256MB
hoodie.logfile.to.parquet.compression.ratio	随着记录从日志文件移动到 parquet，预期会进行额外压缩的比例。 用于 merge_on_read 存储，以将插入内容发送到日志文件中并控制压缩 parquet 文件的大小。	0.35

11.2.6.5 compaction&cleaning 配置

参数	描述	默认值
hoodie.clean.automatic	是否执行自动 clean。	true
hoodie.cleaner.policy	要使用的清理政策。Hudi 将删除旧版本的 parquet 文件以回收空间。 任何引用此版本文件的查询和计算都将失败。最好确保数据保留的时间超过最大查询执行时间。	KEEP_LATEST_COMMITS
hoodie.cleaner.commits.retained	保留的提交数。因此，数据将保留为 num_of_commits * time_between_commits（计划的），这也直接转化为逐步提取此数据集的数量。	10
hoodie.keep.min.commits, hoodie.keep.max.commit	每个提交都是“.hoodie”目录中的一个小文件。由于 DFS 通常	20

参数	描述	默认值
s	不支持大量小文件，因此 Hudi 将较早的提交归档到顺序日志中。 提交通过重命名提交文件以原子方式发布。	
hoodie.commits.archival.batch	这控制着批量读取并一起归档的提交即时的数量。	10
hoodie.parquet.small.file.limit	该值应小于 maxFileSize，如果将其设置为 0，会关闭此功能。由于批处理中分区中插入记录的数量众多，总会出现小文件。Hudi 提供了一个选项，可以通过将对该分区中的插入作为对现有小文件的更新来解决小文件的问题。此处的大小是被视为“小文件大小”的最小文件大小。	104857600 byte
hoodie.copyonwrite.insert.split.size	插入写入并行度。为单个分区的总共插入次数。写出 100MB 的文件，至少 1KB 大小的记录，意味着每个文件有 100K 记录。默认值是超额配置为 500K。 为了改善插入延迟，请对其进行调整以匹配单个文件中的记录数。将此值设置为较小的值将导致文件变小（尤其是当 compactionSmallFileSize 为 0 时）。	500000
hoodie.copyonwrite.insert.auto.split	Hudi 是否应该基于最后 24 个提交的元数据动态计算 insertSplitSize，默认关闭。	true
hoodie.copyonwrite.record.size.estimate	平均记录大小。如果指定，Hudi 将使用它，并且不会基于最后 24 个提交的元数据动态地计算。 没有默认值设置。这对于计算插入并行度以及将插入打包到小文件中至关重要。	1024
hoodie.compact.inline	当设置为 true 时，紧接在插入或插入更新或批量插入的提交或增量提交操作之后由摄取本身触发压缩。	false
hoodie.compact.inline.max.delta.commits	触发内联压缩之前要保留的最大增量提交数。	5
hoodie.compaction.lazy.b	当 CompactedLogScanner 合并所	false

参数	描述	默认值
lock.read	有日志文件时，此配置有助于选择是否应延迟读取日志块。选择 true 以使用 I/O 密集型延迟块读取（低内存使用），或者为 false 来使用内存密集型立即块读取（高内存使用）。	
hoodie.compaction.reverse.log.read	HoodieLogFormatReader 会从 pos=0 到 pos=file_length 向前读取日志文件。如果此配置设置为 true，则 Reader 会从 pos=file_length 到 pos=0 反向读取日志文件。	false
hoodie.cleaner.parallelism	如果清理变慢，请增加此值。	200
hoodie.compaction.strategy	用来决定在每次压缩运行期间选择要压缩的文件组的压缩策略。默认情况下，Hudi 选择具有累积最多未合并数据的日志文件。	org.apache.hudi.table.action.compact.strategy. LogFileSizeBasedCompactionStrategy
hoodie.compaction.target.io	LogFileSizeBasedCompactionStrategy 的压缩运行期间要花费的 MB 量。当压缩以内联模式运行时，此值有助于限制摄取延迟。	500 * 1024 MB
hoodie.compaction.daybased.target	由 org.apache.hudi.io.compact.strategy.DayBasedCompactionStrategy 使用，表示在压缩运行期间要压缩的最新分区数。	10
hoodie.compaction.payload.class	这需要与插入/插入更新过程中使用的类相同。就像写入一样，压缩也使用记录有效负载类将日志中的记录彼此合并，再次与基本文件合并，并生成压缩后要写入的最终记录。	org.apache.hudi.common.model.OverwriteWithLatestAvroPayload
hoodie.schedule.compact.only.inline	在写入操作时，是否只生成压缩计划。在 hoodie.compact.inline=true 时有效。	false
hoodie.run.compact.only.inline	通过 Sql 执行 run compaction 命令时，是否只执行压缩操作，压缩计划不存在时直接退出。	false

11.2.6.6 单表并发写配置

参数	描述	默认值
hoodie.write.lock.provider	指定 lock provider，不建议使用默认值，使用 org.apache.hudi.hive.HiveMetastoreBasedLockProvider	org.apache.hudi.client.transaction.lock.ZookeeperBasedLockProvider
hoodie.write.lock.hivemetastore.database	Hive 的 database	-
hoodie.write.lock.hivemetastore.table	Hive 的 table name	-
hoodie.write.lock.client.num_retries	重试次数	0
hoodie.write.lock.client.wait_time_ms_between_retry	重试间隔	10000
hoodie.write.lock.conflict.resolution.strategy	lock provider 类，必须是 ConflictResolutionStrategy 的子类	org.apache.hudi.client.transaction.SimpleConcurrentFileWritesConflictResolutionStrategy
hoodie.write.lock.zookeeper.base_path	存放 ZNodes 的路径，同一张表的并发写入需配置一致	-
hoodie.write.lock.zookeeper.lock_key	ZNode 的名称，建议与 Hudi 表名相同	-
hoodie.write.lock.zookeeper.connection_timeout_ms	zk 连接超时时间	15000
hoodie.write.lock.zookeeper.port	zk 端口号	-
hoodie.write.lock.zookeeper.url	zk 的 url	-
hoodie.write.lock.zookeeper.session_timeout_ms	zk 的 session 过期时间	60000

11.3 Hudi 性能调优

11.3.1 性能调优方式

当前版本 Hudi 写入操作主推 Spark，因此 Hudi 的调优和 Spark 比较类似，可参考 Spark2x 性能调优。

11.3.2 推荐资源配置

- mor 表：

  由于其本质上是写增量文件，调优可以直接根据 hudi 的数据大小（dataSize）进行调整。

  dataSize 如果只有几个 G，推荐跑单节点运行 spark，或者 yarn 模式但是只分配一个 container。

  入湖程序的并行度 p 设置：建议 p = （dataSize）/128M， 程序分配 core 的数量保持和 p 一致即可。内存设置建议内存大小和 core 的比例大于 1.5:1 即一个 core 配 1.5G 内存， 堆外内存设置建议内存大小和 core 的比例大于 0.5:1。

- cow 表：

  cow 表的原理是重写原始数据，因此这种表的调优，要兼顾 dataSize 和最后重写的文件数量。总体来说 core 数量越大越好（和最后重新的多少个文件数直接相关），并行度 p 和内存大小和 mor 设置类似。

11.4 Hudi SQL 语法参考

11.4.1 使用约束

Hudi 0.9.0 增加了对使用 Spark SQL 操作 Hudi 的 DDL/DML 的语法支持，使得所有用户（非工程师、分析师等）更容易访问和操作 Hudi。

说明

本章节内容仅使用于 MRS 3.1.2 及之后版本。

约束

- 支持在 Hudi 客户端执行 Spark SQL 操作 Hudi。
- 支持在 Spark2x 的 JDBCServer 中执行 Spark SQL 操作 Hudi。
- 不支持在 Spark2x 的客户端执行 Spark SQL 操作 Hudi。
- 不支持在 Hive、Hetu 引擎中操作 Hudi。
- 由于 SQL 的 KeyGenerator 默认是 org.apache.hudi.keygen.ComplexKeyGenerator，要求 DataSource 方式写入时 KeyGenerator 与 SQL 设置的一致。
- 首次执行 **Show Partitions** 命令，需要按照提示手动执行 **msck repair** 命令。
- DML 不支持对无主键表进行 update、delete、merge 操作。

11.4.2 DDL

11.4.2.1 CREATE TABLE

命令功能

CREATE TABLE 命令通过指定带有表属性的字段列表来创建 Hudi Table。

命令格式

CREATE TABLE *[IF NOT EXISTS] [database_name.]table_name*

[(columnTypeList)]

USING hudi

[COMMENT table_comment]

[LOCATION location_path]

[OPTIONS (options_list)]

参数描述

表11-5 CREATE TABLE 参数描述

参数	描述
database_name	Database 名称，由字母、数字和下划线（_）组成。
table_name	Database 中的表名，由字母、数字和下划线（_）组成。
columnTypeList	以逗号分隔的带数据类型的列表。列名由字母、数字和下划线（_）组成。
using	参数 hudi，定义和创建 Hudi table。
table_comment	表的描述信息。
location_path	HDFS 路径，指定该路径 Hudi 表会创建为外表。
options_list	Hudi table 属性列表。

表11-6 CREATE TABLE Options 描述

参数	描述
primaryKey	主键名，多个字段用逗号分隔。
type	表类型。'cow' 表示 COPY-ON-WRITE 表，'mor' 表示 MERGE-ON-READ 表。未指定 type 的话，默认值为 'cow'。
preCombineField	表的 Pre-Combine 字段。
payloadClass	使用 preCombineField 字段进行数据过滤的逻辑，Hudi 除了默认的 OverwriteWithLatestAvroPayload 外同时也提供了多种预置 Payload 供用户使用，如 OverwriteNonDefaultsWithLatestAvroPayload、DefaultHoodieRecordPayload 及 EmptyHoodieRecordPayload。

参数	描述
useCache	是否在 Spark 中缓存表的 relation，无需用户配置。为支持 SparkSQL 中对 COW 表增量视图查询，默认将 COW 表中该值置为 false。

示例

- **创建非分区表**

```
-- 创建一个 cow 内部表
create table if not exists hudi_table0 (
id int,
name string,
price double
)  using hudi
options (
type = 'cow',
primaryKey = 'id'
);
-- 创建一个 mor 外部表
create table if not exists hudi_table1 (
id int,
name string,
price double,
ts bigint
)  using hudi
location '/tmp/hudi/hudi_table1'
options (
type = 'mor',
primaryKey = 'id,name',
preCombineField = 'ts'
);
-- 创建一个无主键表
create table if not exists hudi_table2(
id int,
name string,
price double
)  using hudi
options (
type = 'cow'
);
```

- **创建分区表**

```
create table if not exists hudi_table_p0 (
id bigint,
name string,
ts bigint,
dt string,
hh string
)  using hudi
location '/tmp/hudi/hudi_table_p0'
options (
type = 'cow',
```

```
primaryKey = 'id',
preCombineField = 'ts'
)
partitioned by (dt, hh);
```

- **创建一个 hudi 0.9.0 版本之前通过 spark-shell or deltastreamer 创建的 hudi 表的外表**

```
create table h_p1
using hudi
options (
primaryKey = 'id',
preCombineField = 'ts'
)
partitioned by (dt)
location '/path/to/hudi';
```

- **创建表指定表属性**

```
create table if not exists h3(
id bigint,
name string,
price double
) using hudi
options (
primaryKey = 'id',
type = 'mor',
hoodie.cleaner.fileversions.retained = '20',
hoodie.keep.max.commits = '20'
);
```

注意事项

Hudi 当前不支持使用 char、varchar、tinyint、smallint 类型，建议使用 string 或 int 类型。

系统响应

Table 创建成功，创建成功的消息将被记录在系统日志中。

11.4.2.2 CREATE TABLE AS SELECT

命令功能

CREATE TABLE As SELECT 命令通过指定带有表属性的字段列表来创建 Hudi Table。

命令格式

CREATE TABLE *[IF NOT EXISTS] [database_name.]table_name*

USING hudi

[COMMENT table_comment]

[LOCATION location_path]

[OPTIONS (options_list)]

[AS query_statement]

参数描述

表11-7 CREATE TABLE As SELECT 参数描述

参数	描述
database_name	Database 名称，由字母、数字和下划线（_）组成。
table_name	Database 中的表名，由字母、数字和下划线（_）组成。
using	参数 hudi，定义和创建 Hudi table。
table_comment	表的描述信息。
location_path	HDFS 路径，指定该路径 Hudi 表会创建为外表。
options_list	Hudi table 属性列表。
query_statement	select 查询表达式

示例

- 创建分区表

```
create table h2 using hudi
options (type = 'cow', primaryKey = 'id')
partitioned by (dt)
as
select 1 as id, 'a1' as name, 10 as price, 1000 as dt;
```

- 创建非分区表

```
create table h3 using hudi
as
select 1 as id, 'a1' as name, 10 as price;

从 parquet 表加载数据到 hudi 表
# 创建 parquet 表
create table parquet_mngd using parquet
options(path='hdfs:///tmp/parquet_dataset/*.parquet');

# CTAS 创建 hudi 表
create table hudi_tbl using hudi location 'hdfs:///tmp/hudi/hudi_tbl/' options
(
type = 'cow',
primaryKey = 'id',
preCombineField = 'ts'
)
partitioned by (datestr) as select * from parquet_mngd;
```

注意事项

为了更好的加载数据性能，CTAS 使用 bulk insert 作为写入方式.

系统响应

Table 创建成功，创建成功的消息将被记录在系统日志中。

11.4.2.3 DROP TABLE

命令功能

DROP TABLE 的功能是用来删除已存在的 Table。

命令格式

DROP TABLE *[IF EXISTS] [db_name.]table_name;*

参数描述

表11-8 DROP TABLE 参数描述

参数	描述
db_name	Database 名称。如果未指定，将选择当前 database。
table_name	需要删除的 Table 名称。

注意事项

在该命令中，IF EXISTS 和 db_name 是可选配置。

示例

DROP TABLE IF EXISTS hudidb.h1;

系统响应

Table 将被删除。

11.4.2.4 SHOW TABLE

命令功能

SHOW TABLES 命令用于显示所有在当前 database 中的 table，或所有指定 database 的 table。

命令格式

SHOW TABLES *[IN db_name];*

参数描述

表11-9 SHOW TABLES 参数描述

参数	描述
IN db_name	Database 名称，仅当需要显示指定 Database 的所有 Table 时配置。

注意事项

IN db_Name 为可选配置。

示例

```
SHOW TABLES IN hudidb;
```

系统响应

显示所有 Table。

11.4.2.5 ALTER RENAME TABLE

命令功能

RENAME 命令用于重命名现有表。

命令语法

ALTER TABLE *oldTableName* **RENAME TO** *newTableName*

参数描述

表11-10 RENAME 参数描述

参数	描述
oldTableName	现有表名。
new_table_name	现有表名的新表名。

示例

```
alter table h0 rename to h0_1;
```

系统响应

可以通过运行 SHOW TABLES 显示新表名称。

11.4.2.6 ALTER ADD COLUMNS

命令功能

ADD COLUMNS 命令用于为现有表添加新列。

命令语法

ALTER TABLE *tableIdentifier* **ADD COLUMNS***(colAndType (,colAndType)*)*

参数描述

表11-11 ADD COLUMNS 参数描述

参数	描述
tableIdentifier	表名。
colAndType	带数据类型且用逗号分隔的列的名称。列名称包含字母，数字和下划线（_）。

示例

```
alter table h0_1 add columns(ext0 string);
```

系统响应

通过运行 DESCRIBE 命令，可显示新添加的列。

11.4.2.7 TRUNCATE TABLE

命令功能

该命令将会把表中的数据清空。

命令语法

TRUNCATE TABLE *tableIdentifier*

参数描述

表11-12 **TRUNCATE TABLE** 参数描述

参数	描述

参数	描述
tableIdentifier	表名。

示例

```
truncate table h0_1;
```

系统响应

通过运行 QUERY 语句查看表中数据已被删除。

11.4.3 DML

11.4.3.1 INSERT INTO

命令功能

INSERT 命令用于将 SELECT 查询结果加载到 Hudi 表中。

命令格式

INSERT INTO *tableIndentifier select query;*

参数描述

表11-13 INSERT INTO 参数

参数	描述
tableIndentifier	需要执行 INSERT 命令的 Hudi 表的名称。
select query	查询语句。

注意事项

- Insert 模式：Hudi 对于设置了主键的表支持三种 Insert 模式，用户可以设置参数来指定 Insert 模式，hoodie.sql.insert.mode，默认为 upsert。
  - strict 模式，Insert 语句将保留 COW 表的主键唯一性约束，不允许重复记录。如果在插入过程中已经存在记录，则会为 COW 表抛出 HoodieDuplicateKeyException；对于 MOR 表，该模式与 upsert 模式行为一致。
  - non-strict 模式，对主键表采用 insert 处理。
  - upsert 模式，对于主键表的重复值进行更新操作。
- 在执行 spark-sql 时，用户可以设置 hoodie.sql.bulk.insert.enable = true 和 hoodie.sql.insert.mode = non-strict 来开启 bulk insert 作为 Insert 语句的写入方式。

示例

```
insert into h0 select 1, 'a1', 20;

-- insert static partition
insert into h_p0 partition(dt = '2021-01-02') select 1, 'a1';

-- insert dynamic partition
insert into h_p0 select 1, 'a1', dt;

-- insert dynamic partition
insert into h_p1 select 1 as id, 'a1', '2021-01-03' as dt, '19' as hh;

-- insert overwrite table
insert overwrite table h0 select 1, 'a1', 20;

-- insert overwrite table with static partition
insert overwrite h_p0 partition(dt = '2021-01-02') select 1, 'a1';

-- insert overwrite table with dynamic partition
insert overwrite table h_p1 select 2 as id, 'a2', '2021-01-03' as dt, '19' as hh;
```

系统响应

可在 driver 日志中查看命令运行成功或失败。

11.4.3.2 MERGE INTO

命令功能

通过 MERGE INTO 命令，根据一张表或子查询的连接条件对另外一张表进行查询，连接条件匹配上的进行 UPDATE 或 DELETE，无法匹配的执行 INSERT。这个语法仅需要一次全表扫描就完成了全部同步工作，执行效率要高于 INSERT＋UPDATE。

命令格式

MERGE INTO *tableIdentifier AS target_alias*

USING *(sub_query | tableIdentifier) AS source_alias*

ON *<merge_condition>*

[WHEN MATCHED [AND <condition>] THEN <matched_action>]

[WHEN MATCHED [AND <condition>] THEN <matched_action>]

[WHEN NOT MATCHED [AND <condition>] THEN <not_matched_action>]

<merge_condition> =A equal bool condition

<matched_action> =

DELETE |

*UPDATE SET * |*

UPDATE SET column1 = expression1 [, column2 = expression2 ...]

<not_matched_action> =

*INSERT * |*

INSERT (column1 [, column2 ...]) VALUES (value1 [, value2 ...])

参数描述

表11-14 UPDATE 参数

参数	描述
tableIdentifier	在其中执行 MergeInto 操作的 Hudi 表的名称。
target_alias	目标表的别名。
sub_query	子查询。
source_alias	源表或源表达式的别名。
merge_condition	将源表或表达式和目标表关联起来的条件
condition	过滤条件，可选。
matched_action	当满足条件时进行 Delete 或 Update 操作
not_matched_action	当不满足条件时进行 Insert 操作

注意事项

1. merge-on condition 当前只支持主键列。
2. 当前仅支持对 COW 表进行部分字段的更新。参考：

```
merge into h0 using s0
on h0.id = s0.id
when matched then update set h0.id = s0.id,price = s0.price * 2
```

3. 当前仅支持对 COW 表进行更新时，目标表的字段出现在更新表达式的右值。参考：

```
merge into h0 using s0
on h0.id = s0.id
when matched then update set id = s0.id,
name = h0.name,
price = s0.price + h0.price
```

4. 如果针对部分字段进行更新，更新值必须包含主键列：

   如下所示，Hudi 表使用 id 和 start_dt 作为符合主键。

```
merge into hudi_test3 D using (select id,start_dt from tmp_tb2 where col2 in
('D', 'U')) N
on (N.id = D.id and N.start dt = D.start dt)
when matched then update set
D.id = N.id,
```

```
D.start_dt = N.start_dt,
D.end_dt = '2022-04-28';
```

示例

```
merge into h0 as target
using (
select id, name, price, flag from s
) source
on target.id = source.id
when matched then update set *
when not matched then insert *;

merge into h0
using (
select id, name, price, flag from s
) source
on h0.id = source.id
when matched and flag != 'delete' then update set id = source.id, name =
source.name, price = source.price * 2
when matched and flag = 'delete' then delete
when not matched then insert (id,name,price) values(source.id, source.name,
source.price);

merge into t0 as target
using s0 source
on target.id = source.id
when matched then update set *
when not matched then insert *;
```

系统响应

可在 driver 日志和客户端中查看命令运行成功或失败。

11.4.3.3 UPDATE

命令功能

UPDATE 命令根据列表达式和可选的过滤条件更新 Hudi 表。

命令格式

UPDATE *tableIdentifier SET column = EXPRESSION(,column = EXPRESSION) [WHERE boolExpression]*

参数描述

表11-15 UPDATE 参数

参数	描述
tableIdentifier	在其中执行更新操作的 Hudi 表的名称。

参数	描述
column	待更新的目标列。
EXPRESSION	需在目标表中更新的源表列值的表达式。
boolExpression	过滤条件表达式。

示例

```
update h0 set price = price + 20 where id = 1;
update h0 set price = price *2, name = 'a2' where id = 2;
```

系统响应

可在 driver 日志和客户端中查看命令运行成功或失败。

11.4.3.4 DELETE

命令功能

DELETE 命令从 Hudi 表中删除记录。

命令格式

DELETE from *tableIdentifier [WHERE boolExpression]*

参数描述

表11-16 DELETE 参数

参数	描述
tableIdentifier	在其中执行删除操作的 Hudi 表的名称。
boolExpression	删除项的过滤条件

示例

- 示例 1：

  ```
  delete from h0 where column1 = 'country';
  ```

- 示例 2：

  ```
  delete from h0 where column1 IN ('country1', 'country2');
  ```

- 示例 3：

  ```
  delete from h0 where column1 IN (select column11 from sourceTable2);
  ```

- 示例 4：

```
delete from h0 where column1 IN (select column11 from sourceTable2 where
column1 = 'USA');
```

- 示例 5：

```
delete from h0;
```

系统响应

可在 driver 日志和客户端中查看命令运行成功或失败。

11.4.3.5 COMPACTION

命令功能

压缩(compaction)用于在 MergeOnRead 表将基于行的 log 日志文件转化为 parquet 列式数据文件，用于加快记录的查找。

命令格式

SCHEDULE COMPACTION on *tableIdentifier* |tablelocation;

SHOW COMPACTION on *tableIdentifier* |tablelocation;

RUN COMPACTION on *tableIdentifier* |tablelocation [at instant-time];

参数描述

表11-17 COMPACTION 参数

参数	描述
tableIdentifier	在其中执行删除操作的 Hudi 表的名称。
tablelocation	Hudi 表的存储路径
instant-time	执行 show compaction 命令可以看到 instant-time

示例

```
schedule compaction  on h1;
show compaction on h1;
run compaction on h1 at 20210915170758;

schedule compaction  on '/tmp/hudi/h1';
run compaction on '/tmp/hudi/h1';
```

注意事项

使用 hudi-cli 或 API 方式对 SQL 创建的 Hudi 表触发 Compaction 时需要添加参数 **hoodie.payload.ordering.field** 为 **preCombineField** 的值。

系统响应

可在 driver 日志和客户端中查看命令运行成功或失败。

11.4.3.6 SET/RESET

命令功能

此命令用于动态 Add，Update，Display 或 Reset Hudi 参数，而无需重新启动 driver。

命令格式

- Add 或 Update 参数值：
  SET *parameter_name=parameter_value*
  此命令用于添加或更新“parameter_name”的值。
- Display 参数值：
  SET *parameter_name*
  此命令用于显示指定的“parameter_name”的值。
- Display 会话参数：
  SET
  此命令显示所有支持的会话参数。
- Display 会话参数以及使用细节：
  SET -v
  此命令显示所有支持的会话参数及其使用细节。
- Reset 参数值：
  RESET
  此命令清除所有会话参数。

参数描述

表11-18 SET 参数描述

参数	描述
parameter_name	其值需要被动态添加（add），更新（update）或显示（display）的参数名称。
parameter_value	将要设置的“parameter_name”的新值。

注意事项

以下为分别使用 SET 和 RESET 命令进行动态设置或清除操作的属性：

表11-19 属性描述

属性	描述
hoodie.insert.shuffle.parallelism	insert 方式写入数据时的 spark shuffle 并行度。
hoodie.upsert.shuffle.parallelism	upsert 方式写入数据时的 spark shuffle 并行度。
hoodie.delete.shuffle.parallelism	delete 方式删除数据时的 spark shuffle 并行度。
hoodie.sql.insert.mode	指定 Insert 模式，取值为 strict、non-strict 及 upsert。
hoodie.sql.bulk.insert.enable	指定是否开启 bulk insert 写入。
spark.sql.hive.convertMetastoreParquet	sparksql 把 parquet 表转化为 datasource 表进行读取。当 hudi 的 provider 为 hive 的情况下，使用 sparksql 或 sparkbeeline 进行读取，需要将该参数设置为 false。

示例

- 添加（Add）或更新（Update）：

```
set hoodie.insert.shuffle.parallelism = 100;
set hoodie.upsert.shuffle.parallelism = 100;
set hoodie.delete.shuffle.parallelism = 100;
```

- 重置（Reset）：

```
RESET
```

系统响应

- 若运行成功，将记录在 driver 日志中。
- 若出现故障，将显示在用户界面（UI）中。

11.5 Hudi 常见问题

11.5.1 数据写入

11.5.1.1 写入更新数据时报错 Parquet/Avro schema

问题

数据写入时报错：

```
org.apache.parquet.io.InvalidRecordException: Parquet/Avro schema mismatch: Avro
field 'col1' not found
```

回答

建议在使用 Hudi 时，schema 应该以向后兼容的方式演进。此错误通常发生在使用向后不兼容的演进方式删除某些列如“col1”后，更新 parquet 文件中以旧的 schema 写入的列“col1”，在这种情况下，parquet 尝试在传入记录中查找所有当前字段，当发现“col1”不存在时，抛出上述异常。

解决这个问题的办法是使用所有 schema 演进版本来创建 uber schema，并使用该 schema 作为 target schema。用户可以从 hive metastore 中获取 schema 并将其与当前 schema 合并。

11.5.1.2 写入更新数据时报错 UnsupportedOperationException

问题

数据写入时报错：

```
java.lang.UnsupportedOperationException:
org.apache.parquet.avro.AvroConverters$FieldIntegerConverter
```

回答

因为 schema 演进以非向后兼容的方式进行，此错误将再次发生。基本上，如果已经写入 Hudi 数据集 parquet 文件的记录 R 有一些更新 U。R 包含字段 F，该字段包含某类数据类型，也就是 LONG。U 具有相同的字段 F，该字段的数据类型是 INT。Parquet FS 不支持这种不兼容的数据类型转换。

对于此类错误，请从源头数据采集的位置进行有效的数据类型转换。

11.5.1.3 写入更新数据时报错 SchemaCompatabilityException

问题

数据写入时报错：

```
org.apache.hudi.exception.SchemaCompatabilityException: Unable to validate the
rewritten record <record> against schema <schema>at
org.apache.hudi.common.util.HoodieAvroUtils.rewrite(HoodieAvroUtils.java:215)
```

回答

如果 schema 包含 non-nullable 字段但是值是不存在或者 null，则可能会发生这种情况。

建议以使用向后兼容的演进 schema。本质上，这意味着要么将每个新添加的字段设置为空值，要么为每个新字段设置为默认值。从 Hudi 版本 0.5.1 起，如果依赖字段的默认值，则该故障处理对此无效。

11.5.1.4 Hudi 在 upsert 时占用了临时文件夹中大量空间

问题

Hudi 在 upsert 时占用了临时文件夹中大量空间。

回答

当 UPSERT 大量输入数据时，如果数据量达到合并的最大内存时，Hudi 将溢出部分输入数据到磁盘。

如果有足够的内存，请增加 spark executor 的内存和添加“hoodie.memory.merge.fraction”选项，如： option("hoodie.memory.merge.fraction", "0.8")

11.5.1.5 Hudi 写入小精度 Decimal 数据失败

问题

Hudi 表初始入库采用 BULK_INSERT 方式入库含有 Decimal 类型的数据，之后执行 **upsert**，数据写入时报错：

```
java.lang.UnsupportedOperationException:
org.apache.parquet.avro.AvroConverters$FieldFixedConverter
```

回答

原因：

Hudi 表数据含有 Decimal 类型数据。

初始入库 BULK_INSET 方式会使用 Spark 内部 parquet 文件的写入类进行写入，Spark 对不同精度的 Decimal 类型处理是不同的。

UPSERT 操作时，Hudi 使用 Avro 兼容的 parquet 文件写入类进行写入，这个和 Spark 的写入方式是不兼容的。

解决方案：

执行 BULK_INSERT 时指定设置“hoodie.datasource.write.row.writer.enable = false”，使 hoodie 采用 Avro 兼容的 parquet 文件写入类进行写入。

11.5.2 数据采集

11.5.2.1 使用 kafka 采集数据时报错 IllegalArgumentException

问题

线程“main” 报错 org.apache.kafka.common.KafkaException，构造 kafka 消费者失败，报错：

```
java.lang.IllegalArgumentException: Could not find a 'KafkaClient' entry in the
JAAS configuration. System property 'java.security.auth.login.config' is not set
```

回答

当试图从启用 SSL 的 kafka 数据源采集数据时，而安装程序无法读取 jars.conf 文件及其属性时，可能会发生这种情况。

要解决此问题，需要将所需的属性作为通过 Spark 提交的命令的一部分传递。如：--files jaas.conf,failed_tables.json --conf 'spark.driver.extraJavaOptions=-Djava.security.auth.login.config=jaas.conf' --conf 'spark.executor .extraJavaOptions=-Djava.security.auth.login.config=jaas.conf'

11.5.2.2 采集数据时报错 HoodieException

问题

数据采集时报错：

```
com.uber.hoodie.exception.HoodieException: created at(Part -created at) field not
found in record. Acceptable fields were :[col1, col2, col3, id, name, dob,
created_at, updated_at]
```

回答

这种情况通常当标记为 recordKey 或 partitionKey 的字段在某些传入记录中不存在时发生。请交叉验证你的传入记录。

11.5.2.3 采集数据时报错 HoodieKeyException

问题

创建 Hudi 表时，是否可以使用包含空记录的可空字段作为主键？

回答

不可以，会抛 HoodieKeyException 异常。

```
Caused by: org.apache.hudi.exception.HoodieKeyException: recordKey value: "null"
for field: "name" cannot be null or empty.
at org.apache.hudi.keygen.SimpleKeyGenerator.getKey(SimpleKeyGenerator.java:58)
at
org.apache.hudi.HoodieSparkSqlWriter$$anonfun$1.apply(HoodieSparkSqlWriter.scala:10
4)
at
org.apache.hudi.HoodieSparkSqlWriter$$anonfun$1.apply(HoodieSparkSqlWriter.scala:10
0)
```

11.5.3 Hive 同步

11.5.3.1 Hive 同步数据报错 SQLException

问题

Hive 同步数据时报错：

```
Caused by: java.sql.SQLException: Error while processing statement: FAILED:
Execution Error, return code 1 from org.apache.hadoop.hive.ql.exec.DDLTask. Unable
to alter table. The following columns have types incompatible with the existing
columns in their respective positions :
__col1,__col2
```

回答

这种情况通常会发生当您试图使用 HiveSyncTool.java 类向现有 hive 表添加新列时。数据库通常不允许将列数据类型按照从高到低的顺序修改，或者数据类型可能与表中已存储/将要存储的数据冲突。若要修复相同的问题，请尝试设置以下属性：

设置 hive.metastore.disallow.in compatible.col.type.changes 为 false。

11.5.3.2 Hive 同步数据报错 HoodieHiveSyncException

问题

Hive 同步数据时报错：

```
com.uber.hoodie.hive.HoodieHiveSyncException: Could not convert field Type from
<type1> to <type2> for field col1
```

回答

出现这种情况是因为 HiveSyncTool 目前只支持很少的兼容数据类型转换。进行任何其他不兼容的更改都会引发此异常。

请检查相关字段的数据类型演进，并验证它是否确实可以被视为根据 Hudi 代码库的有效数据类型转换。

11.5.3.3 Hive 同步数据报错 SemanticException

问题

Hive 同步数据时报错：

```
org.apache.hadoop.hive.ql.parse.SemanticException: Database does not exist: test_db
```

回答

这种情况通常在试图对 Hudi 数据集执行 Hive 同步，但配置的 hive_sync 数据库时发生。

请在您的 Hive 集群上创建对应的数据库后重试。

12 使用 Hue（MRS 3.x 之前版本）

12.1 从零开始使用 Hue

Hue 提供了文件浏览器功能，使用户可以通过界面图形化的方式查看 Hive 上文件及目录功能。

前提条件

已安装 Hive 以及 Hue 组件，且状态为运行中的 Kerberos 认证的集群。

操作步骤

步骤 1 访问 Hue WebUI，请参考访问 Hue 的 WebUI。

步骤 2 打开 Hue WebUI，然后选择“Query Editors > Hive”。

步骤 3 在“Databases”选择一个 Hive 中的数据库，默认数据库为“default”。

系统将自动显示数据库中的所有表。可以输入表名关键字，系统会自动搜索包含此关键字的全部表。

步骤 4 单击指定的表名，可以显示表中所有的列。

步骤 5 在 HiveQL 语句编辑区输入 HiveQL 语句。

create table hue_table(id int,name string,company string) row format delimited fields terminated by ',' stored as textfile;

单击 并选择“Explain”，编辑器将分析输入的 HiveQL 语句是否有语法错误以及执行计划，如果存在语法错误则显示“Error while compiling statement”。

步骤 6 单击 ，选择 HiveQL 语句执行的引擎。

步骤 7 单击 开始执行 HiveQL 语句。

步骤 8 在命令输入框内输入 **show tables;**，单击 按钮，查看结果中有步骤 5 创建的表 hue_table。

----结束

12.2 访问 Hue 的 WebUI

操作场景

MRS 集群安装 Hue 组件后，用户可以通过 Hue 的 WebUI，在图形化界面使用 Hadoop 与 Hive。

该任务指导用户在 MRS 集群中打开 Hue 的 WebUI。

说明

Internet Explorer 浏览器可能存在兼容性问题，建议更换兼容的浏览器访问 Hue WebUI，例如 Google Chrome 浏览器 50 版本。

对系统的影响

第一次访问 Manager 和 Hue WebUI，需要在浏览器中添加站点信任以继续打开 Hue WebUI。

前提条件

启用 Kerberos 认证时，MRS 集群管理员已分配用户使用 Hive 的权限。具体操作请参见“用户指南 > MRS 操作指导 > 权限管理 > 创建用户”。例如创建一个“人机”用户“hueuser”，并加入“hive”、“hadoop”、“supergroup”组和“System_administrator”角色，主组为“hive”。

该用户用于登录 Hue WebUI。

操作步骤

步骤 1 登录服务页面：单击集群名称，登录集群详情页面，选择“组件管理”。

说明

若集群详情页面没有“组件管理”页签，请先完成 IAM 用户同步（在集群详情页的“概览”页签，单击“IAM 用户同步”右侧的“同步”进行 IAM 用户同步）。

步骤 2 选择“Hue”，在“Hue WebUI”右侧，单击链接，打开 Hue 的 WebUI，以创建的“hueuser”用户登录 Hue WebUI。

Hue 的 WebUI 支持以下功能：

- 使用“Query Editors”执行 Hive 的查询语句。需要 MRS 集群已安装 Hive。
- 使用“Data Browsers”管理 Hive 中的表。需要 MRS 集群已安装 Hive。
- 使用  查看 HDFS 中的目录和文件。需要 MRS 集群已安装 HDFS。
- 使用  查看 MRS 集群中所有作业。需要 MRS 集群已安装 YARN。

说明

- 使用创建的用户第一次登录 Hue WebUI，需修改密码。

- 用户获取 Hue WebUI 的访问地址后，可以给其他无法访问 Manager 的用户用于访问 Hue WebUI。
- 在 Hue 的 WebUI 操作但不操作 Manager 页面，重新访问 Manager 时需要输入已登录的帐号密码。

----结束

12.3 Hue 常用参数

参数入口

参数入口，请参考修改集群服务配置参数。

参数说明

表12-1 Hue 常用参数

配置参数	说明	缺省值	范围
HANDLER_ACCESSLOG_LEVEL	表示 Hue 的访问日志级别。	DEBUG	• ERROR • WARN • INFO • DEBUG
HANDLER_AUDITSLOG_LEVEL	表示 Hue 的审计日志级别。	DEBUG	• ERROR • WARN • INFO • DEBUG
HANDLER_ERRORLOG_LEVEL	表示 Hue 的错误日志级别。	ERROR	• ERROR • WARN • INFO • DEBUG
HANDLER_LOGFILE_LEVEL	表示 Hue 的运行日志级别。	INFO	• ERROR • WARN • INFO • DEBUG
HANDLER_LOGFILE_MAXBACKUPINDEX	表示 Hue 日志文件最大个数。	20	1～999
HANDLER_LOGFILE_SIZE	表示 Hue 日志文件最大大小。	5MB	-

12.4 在 Hue WebUI 使用 HiveQL 编辑器

操作场景

用户需要使用图形化界面在集群中执行 HiveQL 语句时，可以通过 Hue 完成任务。

访问“Query Editors”

步骤 1 访问 Hue WebUI，请参考访问 Hue 的 WebUI。

步骤 2 选择“Query Editors > Hive”，进入“Hive”。

“Hive”支持以下功能：

- 执行和管理 HiveQL 语句。
- 在“Saved Queries”中查看当前访问用户已保存的 HiveQL 语句。
- 在“Query History”中查看当前访问用户执行过的 HiveQL 语句。
- 单击  ，在“Databases”下可以显示 Hive 中所有的数据库。

----结束

执行 HiveQL 语句

步骤 1 选择“Query Editors > Hive”，进入“Hive”。

步骤 2 单击  ，在“Databases”下选择一个数据库，默认数据库为“default”。

系统将自动显示数据库中的所有表。可以输入表名关键字，系统会自动搜索包含此关键字的全部表。

步骤 3 单击指定的表名，可以显示表中所有的列。

光标移动到表所在的行，单击 可以查看列的详细信息。

步骤 4 在 HiveQL 语句编辑区输入查询语句。

单击  并选择“Explain”，编辑器将分析输入的查询语句是否有语法错误以及执行计划，如果存在语法错误则显示“Error while compiling statement”。

步骤 5 单击 ，选择 HiveQL 语句执行的引擎。

- “mr”表示语句使用 MapReduce 计算框架执行语句。
- “spark”表示语句使用 Spark 计算框架执行语句。
- “tez”表示语句使用 Tez 计算框架执行语句。

说明

tez 适用于 MRS 1.9.x 及以后版本。

步骤 6 单击 开始执行 HiveQL 语句。

说明

- 如果希望下次继续使用已输入的 HiveQL 语句，请单击 保存。
- 格式化 HiveQL 语句，请单击 选择“Format”。
- 删除已输入的 HiveQL 语句，请单击 选择“Clear”。
- 清空已输入的语句并执行一个新的语句，请单击 选择“New query”。
- 查看历史：

  单击“Query History”，可查看 HiveQL 运行情况，支持显示所有语句或只显示保存的语句的运行情况。历史记录存在多个结果时，可以在输入框使用关键字进行搜索。
- 高级查询配置：

  单击右上角的，对文件、函数、设置等信息进行配置。
- 查看快捷键：

  单击右上角的，可查看所有快捷键信息。

----结束

查看执行结果

步骤 1 在“Hive”的执行区，默认显示“Query History”。

步骤 2 单击“Results”查看已执行语句的执行结果。

----结束

管理查询语句

步骤 1 选择“Query Editors > Hive”，进入“Hive”。

步骤 2 单击“Saved Queries”。

单击一条已保存的语句，系统会自动将其填充至编辑区中。

----结束

修改在 Hue 使用“Query Editors”的会话配置

步骤 1 在“Hive”页签，单击 。

步骤 2 在“Files”的右侧单击 ，然后单击 指定该文件的存储目录。

可以单击 新增加一个文件资源。

步骤 3 在“Functions”的右侧单击 ，输入用户自定义的名称和函数的类名称。

可以单击 新增加一个自定义函数。

步骤 4 在“Settings”的右侧单击 ，在“Key”输入 Hive 的参数名，在“Value”输入对应的参数值，则当前 Hive 会话会以用户定义的配置连接 Hive。

可以单击 新增加一个参数。

----结束

12.5 在 Hue WebUI 使用元数据浏览器

操作场景

用户需要使用图形化界面在集群中管理 Hive 的元数据，可以通过 Hue 完成任务。

Metastore 管理器使用介绍

访问 Hue WebUI，请参考访问 Hue 的 WebUI。

选择“Data Browsers > Metastore Tables”，进入“Metastore Manager”。

- 查看 Hive 表的元数据

  在左侧导航栏中，将鼠标放在某一表上，单击显示在其右侧的图标 ，界面将显示 Hive 表的元数据信息。
- 管理 Hive 表的元数据

  在 Hive 表的元数据信息界面，单击右上角的 可导入数据，单击 可浏览数据，单击 可查看表文件的位置信息。

注意

Hue 界面主要用于文件、表等数据的查看与分析，禁止通过 Hue 界面对操作对象进行删除等高危管理操作。如需操作，建议在确认对业务没有影响后通过各组件的相应操作方法进行处理，例如使用 HDFS 客户端对 HDFS 文件进行操作，使用 Hive 客户端对 Hive 表进行操作。

- 管理 Hive 元数据表

  选择右上角的 可在数据库中根据上传的文件创建一个新表，选择右上角的 可手动创建一个新表。

访问“Metastore Manager”

步骤 1 访问 Hue WebUI，请参考访问 Hue 的 WebUI。

步骤 2 选择“Data Browsers > Metastore Tables”，进入“Metastore Manager”。

“Metastore Manager”支持以下功能：

- 使用文件创建一个 Hive 表
- 手动创建一个 Hive 表
- 查看 Hive 表元数据

----结束

使用文件创建一个 Hive 表

步骤 1 访问“Metastore Manager”，在“Databases”选择一个数据库。

默认数据库为“default”。

步骤 2 单击  ，进入“Create a new table from a file”页面。

步骤 3 选择文件。

1. 在“Table Name”填写 Hive 表的名称。

   支持字母、数字、下划线，首位必须为字母或数字，且长度不能超过 128 位。
2. 根据需要，在“Description”填写 Hive 表的描述信息。
3. 在“Input File or Location”单击  ，在 HDFS 中选择一个用于创建 Hive 表文件。此文件将存储 Hive 表的新数据。

   如果文件未在 HDFS 中保存，可以单击“Upload a file”从本地选择文件并上传。支持同时上传多个文件，文件不可为空。
4. 如果需要将文件中的数据导入 Hive 表，选择“Import data”作为“Load method”。默认选择“Import data”。

   选择“Create External Table”时，创建的是 Hive 外部表。

   说明

   当选择“Create External Table”时，参数“Input File or Location”需要选择为路径。

   选择“Leave Empty”则创建空的 Hive 表。
5. 单击“Next”。

步骤 4 设置分隔符。

1. 在“Delimiter”选择一个分隔符。

   如果分隔符不在列表中，选择“Other..”，然后输入新定义的分隔符。
2. 单击“Preview”查看数据处理预览。
3. 单击“Next”。

步骤 5 定义字段列。

1. 单击“Use first row as column names”右侧的  ，则使用文件中第一行数据作为列名称。取消则不使用数据作为列名称。
2. 在“Column name”编辑每个列的名称。

支持字母、数字、下划线，首位必须为字母或数字，且长度不能超过 128 位。

说明

单击“Bulk edit column names”右侧的，可批量对列重新命名。输入所有列的名称并使用逗号分隔。

3. 在“Column Type”选择每个列的类型。

步骤 6 单击“Create Table”创建表，等待 Hue 显示 Hive 表的信息。

----结束

手工创建一个 Hive 表

步骤 1 访问“Metastore Manager”，在“Databases”选择一个数据库。

默认数据库为“default”。

步骤 2 单击，进入“Create a new table manually”页面。

步骤 3 设置表名称。

1. 在“Table Name”填写 Hive 表的名称。

   支持字母、数字、下划线，首位必须为字母或数字，且长度不能超过 128 位。
2. 根据需要，在“Description”填写 Hive 表的描述信息。
3. 单击“Next”。

步骤 4 选择一个存储数据的格式。

- 需要使用分隔符分隔数据时，选择“Delimited”，然后执行步骤 5。
- 需要使用序列化格式保存数据时，选择“SerDe”，执行步骤 6。

步骤 5 配置分隔符。

1. 在“Field terminator”设置一个列分隔符。

   如果分隔符不在列表中，选择“Other..”，然后输入新定义的分隔符。
2. 在“Collection terminator”设置一个分隔符，用于分隔 Hive 中类型为“array”的列的数据集合。例如一个列为 array 类型，其中一个值需要保存“employee”和“manager”，用户指定分隔符为“:”，则最终的值为“employee:manager”。
3. 在“Map key terminator”设置一个分隔符，用于分隔 Hive 中类型为“map”的列的数据。例如某个列为 map 类型，其中一个值需要保存描述为“aaa”的“home”，和描述为“bbb”的“company”，用户指定分隔符为“|”，则最终的值为“home|aaa:company|bbb”。
4. 单击“Next”，执行步骤 7。

步骤 6 设置序列化属性。

1. 在“SerDe Name”输入序列化格式的类名称“org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe”。

   用户可扩展 Hive 支持更多自定义的序列化类。

2. 在“Serde properties”输入序列化的样式的值：“"field.delim"="," "colelction.delim"=":" "mapkey.delim"="|"”。
3. 单击“Next”，执行执行步骤 7。

步骤 7 选择一个数据表的格式，并单击“Next”。

- “TextFile”表示使用文本类型文件存储数据。
- “SequenceFile”表示使用二进制类型文件存储数据。
- “InputFormat”表示使用自定义的输入输出格式来使用文件中的数据。

  用户可扩展 Hive 支持更多的自定义格式化类。

  a. 在“InputFormat Class”填写输入数据使用的类“org.apache.hadoop.hive.ql.io.RCFileInputFormat”。

  b. 在“OutputFormat Class”填写输出数据使用的类“org.apache.hadoop.hive.ql.io.RCFileOutputFormat”。

步骤 8 选择一个文件保存位置，并单击“Next”。

默认勾选“Use default location”。如果需要自定义存储位置，请取消选中状态并在“External location”单击 指定一个文件存储位置。

步骤 9 设置 Hive 表的字段。

1. 在“Column name”设置列的名称。

   支持字母、数字、下划线，首位必须为字母或数字，且长度不能超过 128 位。
2. 在“Column type”选择一个数据类型。

   单击“Add a column”可增加新的列。
3. 单击“Add a partition”为 Hive 表增加分区，可提高查询效率。

步骤 10 单击“Create Table”创建表，等待 Hue 显示 Hive 表的信息。

----结束

管理 Hive 表

步骤 1 访问“Metastore Manager”，在“Databases”选择一个数据库，页面显示数据库中所有的表。

默认数据库为“default”。

步骤 2 单击数据库中的表名称，打开表的详细信息。

支持导入数据、浏览数据或查看文件存储位置。查看数据库所有的表时，可以直接勾选表然后执行查看、浏览数据操作。

> **注意**
>
> Hue 界面主要用于文件、表等数据的查看与分析，禁止通过 Hue 界面对操作对象进行删除等高危管理操作。如需操作，建议在确认对业务没有影响后通过各组件的相应操作方法进行处理，例如使用 HDFS 客户端对 HDFS 文件进行操作，使用 Hive 客户端对 Hive 表进行操作。

----结束

12.6 在 Hue WebUI 使用文件浏览器

操作场景

用户需要使用图形化界面管理 HDFS 中文件时，可以通过 Hue 完成任务。

> **注意**
>
> Hue 界面主要用于文件、表等数据的查看与分析，禁止通过 Hue 界面对操作对象进行删除等高危管理操作。如需操作，建议在确认对业务没有影响后通过各组件的相应操作方法进行处理，例如使用 HDFS 客户端对 HDFS 文件进行操作，使用 Hive 客户端对 Hive 表进行操作。

访问文件浏览器（File Browser）

步骤 1 访问 Hue WebUI，请参考访问 Hue 的 WebUI。

步骤 2 单击  ，进入“File Browser”。

默认进入当前登录用户的主目录。

文件浏览器将显示目录中的子目录或文件以下信息：

表12-2 HDFS 文件属性介绍

属性名	描述
“Name”	表示目录或文件的名称。
“Size”	表示文件的大小。
“User”	表示目录或文件的属主。
“Group”	表示目录或文件的属组。
“Permissions”	表示目录或文件的权限设置。

属性名	描述
“Date”	表示目录或文件创建时间。

步骤 3 在搜索框输入关键字，系统会在当前目录自动搜索目录或文件。

步骤 4 清空搜索框的内容，系统会重新显示所有目录和文件。

----结束

执行动作

步骤 1 单击  ，选择一个或多个目录或文件。

步骤 2 单击“Actions”，在弹出菜单选择一个操作。

- “Rename”：表示重新命名一个目录或文件。
- “Move”：表示移动文件，在“移至”选择新的目录并单击“移动”完成移动。
- “Copy”：表示复制选中的文件或目录。
- “Change permissions”：表示修改选中目录或文件的访问权限。
  - 可以为属主、属组和其他用户设置“Read”、“Write”和“Excute”权限。
  - “Sticky”表示禁止 HDFS 的管理员、目录属主或文件属主以外的用户在目录中移动文件。
  - “Recursive”表示递归设置权限到子目录。
- “Storage policies”：表示设置目录或文件在 HDFS 中的存储策略。
- “Summary”：表示查看选中文件或目录的 HDFS 存储信息。

----结束

访问其他目录

步骤 1 单击目录名并输入需要访问的目录完整路径，例如“/mr-history/tmp”并按回车键进入目录。

需要当前登录 Hue WebUI 的用户拥有其他目录的访问权限。

步骤 2 单击“Home”可进入用户的主目录。

步骤 3 单击“History”可以显示最近访问目录的历史记录，并重新访问。

步骤 4 单击“Trash”可以访问当前目录的回收站空间。

单击“Empty Trash”可清空回收站。

----结束

上传用户文件

步骤 1 单击，单击 Upload。

步骤 2 选择一个操作。

- “Files”：表示上传用户文件到当前用户。
- “Zip/Tgz/Bz2 file”：表示上传了一个压缩文件，在弹出框单击“Select ZIP, TGZ or BZ2 files”选择需要上传的压缩文件。系统会自动在 HDFS 中对文件解压。支持“ZIP”、“TGZ”和“BZ2”格式的压缩文件。

----结束

创建新文件或者目录

步骤 1 单击，单击“New”。

步骤 2 选择一个操作。

- “File”：表示创建一个文件，输入文件名后单击“Create”完成。
- “Directory”：表示创建一个目录，输入目录名后单击“Create”完成

----结束

存储策略定义使用介绍

说明

若 Hue 的服务配置参数“fs_defaultFS”配置为“viewfs://ClusterX”时，不能启用存储策略定义功能。

步骤 1 登录 MRS Manager。

步骤 2 在 MRS Manager 界面，选择“系统设置 > 权限配置 > 角色管理 > 添加角色”：

1. 设置“角色名称”。
2. 选择“权限 > Hue”，勾选"Storage Policy Admin"，单击“确定”，为该角色赋予存储策略管理员的权限。

步骤 3 选择“系统设置 > 权限配置 > 用户组管理 > 添加用户组”，设置“组名”，单击“角色”后的“选择添加角色”，在弹出的界面选择刚创建的角色，单击“确定”将该角色添加到组中。

步骤 4 选择“系统设置 > 权限配置 > 用户管理 > 添加用户”：

1. 设置可以登录 Hue 的 WebUI 界面且有存储策略管理员权限的用户的“用户名”。
2. “用户类型”选择“人机”。
3. 设置登录 Hue 的 WebUI 界面的“密码”、“确认密码”。
4. 单击“用户组”后的“选择添加的用户组”，在弹出的界面选择创建的用户组、supergroup、hadoop 和 hive 用户组，单击“确定”。
5. “主组”选择“hive”。

6. 单击“分配角色权限”右侧的“选择并绑定角色”，在弹出的界面选择刚刚创建的角色和 System_administrator 角色，单击“确定”。
7. 再单击“确定”成功添加该用户。

步骤 5 访问 Hue WebUI，请参考访问 Hue 的 WebUI。

步骤 6 单击右上角的  。

步骤 7 勾选目录的复选框，单击页面上方的“Action”，选择“Storage policies”。

步骤 8 在弹出的对话框中设置新的存储策略，单击“OK”。

----结束

12.7 在 Hue WebUI 使用作业浏览器

操作场景

用户需要使用图形化界面查看集群中所有作业时，可以通过 Hue 完成任务。

访问“Job Browser”

步骤 1 访问 Hue WebUI，请参考访问 Hue 的 WebUI。

步骤 2 单击“Job Browser”。

默认显示当前集群的所有作业。

说明

“Job Browser”显示的数字表示集群中所有作业的总数。

“Job Browser”将显示作业以下信息：

表12-3 MRS 作业属性介绍

属性名	描述
“Logs”	表示作业的日志信息。如果作业有输出日志，则显示  。
“ID”	表示作业的编号，由系统自动生成。
“Name”	表示作业的名称。
“Application Type”	表示作业的类型。
“Status”	表示作业的状态，包含“RUNNING”、“SUCCEEDED”、“FAILED”和“KILLED”。
“User”	表示启动该作业的用户。
“Maps”	表示作业执行 Map 过程的进度。

属性名	描述
“Reduces”	表示作业执行 Reduce 过程的进度。
“Queue”	表示作业运行时使用的 YARN 队列。
“Priority”	表示作业运行时的优先级。
“Duration”	表示作业运行使用的时间。
“Submitted”	表示作业提交到 MRS 集群的时间。

说明

如果 MRS 集群安装了 Spark 组件，则默认会启动一个作业“Spark-JDBCServer”，用于执行任务。

----结束

搜索作业

步骤 1 在“Job Browser”的“Username”或“Text”，输入指定的字符，系统会自动搜索包含此关键字的全部作业。

步骤 2 清空搜索框的内容，系统会重新显示所有作业。

----结束

查看作业详细信息

步骤 1 在“Job Browser”的作业列表，单击作业所在的行，可以打开作业详情。

步骤 2 在“Metadata”页签，可查看作业的元数据。

说明

单击可打开作业运行时的日志。

----结束

13 使用 Hue（MRS 3.x 及之后版本）

13.1 从零开始使用 Hue

Hue 汇聚了与大多数 Apache Hadoop 组件交互的接口，致力让用户通过界面图形化的方式轻松使用 Hadoop 组件。目前 Hue 支持 HDFS、Hive、HBase、Yarn、MapReduce、Oozie 和 SparkSQL 等组件的可视化操作。

前提条件

已安装 Hue 组件，且状态为运行中的 Kerberos 认证的集群。

操作步骤

步骤 1 访问 Hue WebUI，请参考访问 Hue 的 WebUI。

步骤 2 在左侧导航栏单击编辑器，然后选择“Hive”。

步骤 3 在“Database”右侧下拉列表选择一个 Hive 中的数据库，默认数据库为“default”。

系统将自动显示数据库中的所有表。可以输入表名关键字，系统会自动搜索包含此关键字的全部表。

步骤 4 单击指定的表名，可以显示表中所有的列。

步骤 5 在 HiveQL 语句编辑区输入 HiveQL 语句。

create table hue_table(id int,name string,company string) row format delimited fields terminated by ',' stored as textfile;

步骤 6 单击 开始执行 HiveQL 语句。

步骤 7 在命令输入框内输入 **show tables;**，单击 按钮，查看“结果”中有步骤 5 创建的表 hue_table。

----结束

13.2 访问 Hue 的 WebUI

操作场景

MRS 集群安装 Hue 组件后，用户可以通过 Hue 的 WebUI，在图形化界面使用 Hadoop 生态相关组件。

该任务指导用户在 MRS 集群中打开 Hue 的 WebUI。

说明

Internet Explorer 浏览器可能存在兼容性问题，建议更换兼容的浏览器访问 Hue WebUI，例如 Google Chrome 浏览器 50 版本。

对系统的影响

第一次访问 Manager 和 Hue WebUI，需要在浏览器中添加站点信任以继续打开 Hue WebUI。

前提条件

启用 Kerberos 认证时，MRS 集群管理员已分配用户使用 Hive 的权限。具体操作请参见“用户指南 > MRS 操作指导 > 权限管理 > 创建用户”章节。例如创建一个“人机”用户“hueuser”，并加入“hive”、“hadoop”、“supergroup”组和“System_administrator”角色，主组为“hive”。

该用户用于登录 Manager。

操作步骤

步骤 1 登录服务页面：

MRS 3.x 之前版本，在 MRS 控制台单击集群名称，选择“组件管理 > Hue”。

MRS 3.x 及后续版本，登录 FusionInsight Manager，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本），选择“集群 > 服务 > Hue”。

步骤 2 在“Hue WebUI”右侧，单击链接，打开 Hue 的 WebUI。

Hue 的 WebUI 支持以下功能：

- 使用编辑器  执行 Hive、SparkSql 的查询语句以及 Notebook 代码段。需要 MRS 集群已安装 Hive、Spark2x。
- 使用计划程序  提交 Workflow 任务、计划任务、Bundle 任务。
- 使用文档  查看、导入、导出在 Hue 页面上操作的任务，例如保存的 Workflow 任务、定时任务、Bundle 任务等。
- 使用表  管理 Hive、SparkSql 中的元数据。需要 MRS 集群已安装 Hive、Spark2x。

- 使用文件  查看 HDFS 中的目录和文件。需要 MRS 集群已安装 HDFS。
- 使用作业  查看 MRS 集群中所有作业。需要 MRS 集群已安装 Yarn。
- 使用 HBase  创建/查询 HBase 表。需要 MRS 集群已安装 HBase 组件并添加 Thrift1Server 实例。
- 使用导入器  通过“.csv”，“.txt”等格式的文件导入数据。

说明

- 使用创建的用户第一次登录 Hue WebUI，需修改密码。
- 用户获取 Hue WebUI 的访问地址后，可以给其他无法访问 Manager 的用户用于访问 Hue WebUI。
- 在 Hue 的 WebUI 操作但不操作 Manager 页面，重新访问 Manager 时需要输入已登录的帐号密码。

----结束

13.3 Hue 常用参数

参数入口

参数入口，请参考修改集群服务配置参数进入 Hue 服务“全部配置”页面。

参数说明

Hue 常用参数请参见表 13-1。

表13-1 Hue 常用参数

配置参数	说明	缺省值	范围
HANDLER_ACCESSLOG_LEVEL	Hue 的访问日志级别。	DEBUG	• ERROR • WARN • INFO • DEBUG
HANDLER_AUDITSLOG_LEVEL	Hue 的审计日志级别。	DEBUG	• ERROR • WARN • INFO • DEBUG
HANDLER_ERRORLOG_LEVEL	Hue 的错误日志级别。	ERROR	• ERROR • WARN • INFO • DEBUG

配置参数	说明	缺省值	范围
HANDLER_LOGFILE_LEVEL	Hue 的运行日志级别。	INFO	• ERROR • WARN • INFO • DEBUG
HANDLER_LOGFILE_MAXBACKUPINDEX	Hue 日志文件最大个数。	20	1～999
HANDLER_LOGFILE_SIZE	Hue 日志文件最大大小。	5MB	-

13.4 在 Hue WebUI 使用 HiveQL 编辑器

操作场景

用户需要使用图形化界面在集群中执行 HiveQL 语句时，可以通过 Hue 完成任务。

访问编辑器

步骤 1 访问 Hue WebUI，请参考访问 Hue 的 WebUI。

步骤 2 在左侧导航栏单击</>，然后选择“Hive”，进入“Hive”。

“Hive”支持以下功能：

- 执行和管理 HiveQL 语句。
- 在“保存的查询”中查看当前访问用户已保存的 HiveQL 语句。
- 在“查询历史记录”中查看当前访问用户执行过的 HiveQL 语句。

----结束

执行 HiveQL 语句

步骤 1 在“Database”右侧下拉列表选择一个 Hive 中的数据库，默认数据库为“default”。

系统将自动显示数据库中的所有表。可以输入表名关键字，系统会自动搜索包含此关键字的全部表。

步骤 2 单击指定的表名，可以显示表中所有的列。

光标移动到表或列所在的行，单击 可以查看详细信息。

步骤 3 在 HiveQL 语句编辑区输入查询语句。

步骤 4 单击 开始执行 HiveQL 语句。

说明

- 如果希望下次继续使用已输入的 HiveQL 语句，请单击 保存。
- 高级查询配置：

  单击右上角的，对文件、功能、设置等信息进行配置。
- 查看快捷键：

  单击右上角的，可查看语法和键盘快捷方式信息。
- 删除已输入的 HiveQL 语句，请单击 后的三角选择“清除”。
- 查看历史：

  单击“查询历史记录”，可查看 HiveQL 运行情况，支持显示所有语句或只显示保存的语句的运行情况。历史记录存在多个结果时，可以在输入框使用关键字进行搜索。

----结束

查看执行结果

步骤 1 在“Hive”的执行区，默认显示“查询历史记录”。

步骤 2 单击结果查看已执行语句的执行结果。

----结束

管理查询语句

步骤 1 单击“保存的查询”。

步骤 2 单击一条已保存的语句，系统会自动将其填充至编辑区中。

----结束

修改在 Hue 使用编辑器的会话配置

步骤 1 在编辑器页面，单击 。

步骤 2 在“文件”的右侧单击 ，然后单击 选择文件。

可以单击“文件”后的 新增加一个文件资源。

步骤 3 在“功能” ，输入用户自定义的名称和函数的类名称。

可以单击“功能”后的 新增加一个自定义函数。

步骤 4 在“设置” ，在“设置”的“键”输入 Hive 的参数名，在“值”输入对应的参数值，则当前 Hive 会话会以用户定义的配置连接 Hive。

可以单击 新增加一个参数。

----结束

13.5 在 Hue WebUI 使用 SparkSql 编辑器

操作场景

用户需要使用图形化界面在集群中执行 SparkSql 语句时，可以通过 Hue 完成任务。

配置 Spark2x

使用 SparkSql 编辑器之前需要先修改 Spark2x 配置。

步骤 1 进入 Spark2x 的全部配置页面，具体操作请参考修改集群服务配置参数。

步骤 2 设置 Spark2x 多实例模式，搜索并修改 Spark2x 服务的以下参数：

参数名称	值
spark.thriftserver.proxy.enabled	false
spark.scheduler.allocation.file	#{conf_dir}/fairscheduler.xml

步骤 3 进入 JDBCServer2x 自定义界面，在 spark.core-site.customized.configs 参数内，添加两个自定义项：

名称为：hadoop.proxyuser.hue.groups，值为：*

名称为：hadoop.proxyuser.hue.hosts，值为：*

步骤 4 保存配置，重启 Spark2x 服务。

----结束

访问编辑器

步骤 1 访问 Hue WebUI，请参考访问 Hue 的 WebUI。

步骤 2 在左侧导航栏单击 </>，然后选择“SparkSql”，进入“SparkSql”。

“SparkSql”支持以下功能：

- 执行和管理 SparkSql 语句。
- 在“保存的查询”中查看当前访问用户已保存的 SparkSql 语句。
- 在“查询历史记录”中查看当前访问用户执行过的 SparkSql 语句。

----结束

执行 SparkSql 语句

步骤 1 在“Database”右侧下拉列表选择一个 SparkSql 中的数据库，默认数据库为“default”。

系统将自动显示数据库中的所有表。可以输入表名关键字，系统会自动搜索包含此关键字的全部表。

步骤 2 单击指定的表名，可以显示表中所有的列。

光标移动到表所在的行，单击 可以查看列的详细信息。

步骤 3 在 SparkSql 语句编辑区输入查询语句。

单击 后的三角并选择“解释”，编辑器将分析输入的查询语句是否有语法错误以及执行计划，如果存在语法错误则显示“Error while compiling statement”。

步骤 4 单击 开始执行 SparkSql 语句。

说明

- 如果希望下次继续使用已输入的 SparkSql 语句，请单击 保存。
- 高级查询配置：

  单击右上角的，对文件、功能、设置等信息进行配置。
- 查看快捷键：

  单击右上角的，可查看语法和键盘快捷方式信息。
- 格式化 SparkSql 语句，请单击 后的三角选择“格式”
- 删除已输入的 SparkSql 语句，请单击 后的三角选择“清除”
- 查看历史：

  单击“查询历史记录”，可查看 SparkSql 运行情况，支持显示所有语句或只显示保存的语句的运行情况。历史记录存在多个结果时，可以在输入框使用关键字进行搜索。

----结束

查看执行结果

步骤 1 在“SparkSql”的执行区，默认显示“查询历史记录”。

步骤 2 单击结果查看已执行语句的执行结果。

----结束

管理查询语句

步骤 1 单击“保存的查询”。

步骤 2 单击一条已保存的语句，系统会自动将其填充至编辑区中。

----结束

13.6 在 Hue WebUI 使用元数据浏览器

操作场景

用户需要使用图形化界面在集群中管理 Hive 的元数据，可以通过 Hue 完成任务。

元数据管理器使用介绍

访问 Hue WebUI，请参考访问 Hue 的 WebUI。

- 查看 Hive 表的元数据

  在左侧导航栏单击表  ，单击某一表名称，界面将显示 Hive 表的元数据信息。
- 管理 Hive 表的元数据

  在 Hive 表的元数据信息界面：
  - 单击右上角的“导入”可导入数据。
  - 单击“概述”，在“属性”域可查看表文件的位置信息。

    可查看 Hive 表各列字段的信息，并手动添加描述信息，注意此处添加的描述信息并不是 Hive 表中的字段注释信息（comment）。
  - 单击“样本”可浏览数据。
- 管理 Hive 元数据表

  单击左侧列表中的 ➕ 可在数据库中根据上传的文件创建一个新表，也可手动创建一个新表。

⚠ 注意

Hue 界面主要用于文件、表等数据的查看与分析，禁止通过 Hue 界面对操作对象进行删除等高危管理操作。如需操作，建议在确认对业务没有影响后通过各组件的相应操作方法进行处理，例如使用 HDFS 客户端对 HDFS 文件进行操作，使用 Hive 客户端对 Hive 表进行操作。

13.7 在 Hue WebUI 使用文件浏览器

操作场景

用户需要使用图形化界面管理 HDFS 中文件时，可以通过 Hue 完成任务。

> **⚠ 注意**
>
> Hue 界面主要用于文件、表等数据的查看与分析，禁止通过 Hue 界面对操作对象进行删除等高危管理操作。如需操作，建议在确认对业务没有影响后通过各组件的相应操作方法进行处理，例如使用 HDFS 客户端对 HDFS 文件进行操作，使用 Hive 客户端对 Hive 表进行操作。

访问文件浏览器

步骤 1 访问 Hue WebUI，请参考访问 Hue 的 WebUI。

步骤 2 在左侧导航栏单击文件  。进入“文件浏览器”页面。

“文件浏览器”的“主页”默认进入当前登录用户的主目录。界面将显示目录中的子目录或文件的以下信息：

表13-2 HDFS 文件属性介绍

属性名	描述
名称	表示目录或文件的名称。
大小	表示文件的大小。
用户	表示目录或文件的属主。
组	表示目录或文件的属组。
权限	表示目录或文件的权限设置。
日期	表示目录或文件创建时间。

步骤 3 在搜索框输入关键字，系统会在当前目录自动搜索目录或文件。

步骤 4 清空搜索框的内容，系统会重新显示所有目录和文件。

----结束

执行动作

步骤 1 在“文件浏览器”界面，勾选一个或多个目录或文件。

步骤 2 单击“操作”，在弹出菜单选择一个操作。

- 重命名：表示重新命名一个目录或文件。
- 移动：表示移动文件，在“移至”页面选择新的目录并单击“移动”完成移动。
- 复制：表示复制选中的文件或目录。

- 更改权限：表示修改选中目录或文件的访问权限。
  - 可以为属主、属组和其他用户设置“读取”、“写”和“执行”权限。
  - “易贴”表示禁止 HDFS 的管理员、目录属主或文件属主以外的用户在目录中移动文件。
  - “递归”表示递归设置权限到子目录。
- 存储策略：表示设置目录或文件在 HDFS 中的存储策略。
- 摘要：表示查看选中文件或目录的 HDFS 存储信息。

----结束

上传用户文件

步骤 1 在“文件浏览器”界面，单击“上传”。

步骤 2 在弹出的上传文件窗口中单击“选择文件”或将文件拖至窗口中，完成文件上传。

----结束

创建新文件或者目录

步骤 1 在“文件浏览器”界面，单击“新建”。

步骤 2 选择一个操作。

- 文件：表示创建一个文件，输入文件名后单击“创建”完成。
- 目录：表示创建一个目录，输入目录名后单击“创建”完成。

----结束

存储策略定义使用介绍

说明

若 Hue 的服务配置参数“fs_defaultFS”配置为“viewfs://ClusterX”时，不能启用存储策略定义功能。

步骤 1 登录 FusionInsight Manager。

步骤 2 在 FusionInsight Manager 界面，选择“系统 > 权限 > 角色 > 添加角色”：

1. 设置“角色名称”。
2. 在“配置资源权限”下选择“*待操作集群名称* >Hue”，勾选“存储策略管理员”，单击“确定”，为该角色赋予存储策略管理员的权限。

步骤 3 选择“系统 > 权限 > 用户组 > 添加用户组”，设置“组名”，单击“角色”后的“添加”，在弹出的界面选择步骤 2 创建的角色，单击“确定”将该角色添加到组中，单击“确定”完成用户组的创建。

步骤 4 选择“系统 > 权限 > 用户 > 添加用户”：

1. “用户名”填写待添加的用户名。

2. “用户类型”设置为“人机”。
3. 设置登录 Hue 的 WebUI 界面的“密码”、“确认密码”。
4. 单击“用户组”后的“添加”，在弹出的界面选择步骤 3 创建的用户组、supergroup、hadoop 和 hive 用户组，单击“确定”。
5. “主组”选择“hive”。
6. 单击“角色”后的“添加”，在弹出的界面选择步骤 2 创建的角色和 System_administrator 角色，单击“确定”。
7. 再单击“确定”，成功添加该用户。

步骤 5 使用创建的用户访问 Hue WebUI，具体操作请参考访问 Hue 的 WebUI。

步骤 6 左侧导航栏单击文件  。进入“文件浏览器”页面。

步骤 7 勾选目录的复选框，单击页面上方的“操作”，单击“存储策略”。

步骤 8 在弹出的对话框中设置新的存储策略，单击“保存”。

----结束

13.8 在 Hue WebUI 使用作业浏览器

操作场景

用户需要使用图形化界面查看集群中所有作业时，可以通过 Hue 完成任务。

访问作业浏览器

步骤 1 访问 Hue WebUI，请参考访问 Hue 的 WebUI。

步骤 2 单击作业  。

默认显示当前集群的所有作业。

说明

作业浏览器显示的数字表示集群中所有作业的总数。

“作业浏览器”将显示作业以下信息：

表13-3 MRS 作业属性介绍

属性名	描述
名称	表示作业的名称。
用户	表示启动该作业的用户。
类型	表示作业的类型。
状态	表示作业的状态，包含“成功”、“正在运行”、“失

属性名	描述
	败”。
进度	表示作业运行进度。
组	表示作业所属组。
开始	表示作业开始时间。
持续时间	表示作业运行使用的时间。
Id	表示作业的编号，由系统自动生成。

说明

如果 MRS 集群安装了 Spark 组件，则默认会启动一个作业“Spark-JDBCServer”，用于执行任务。

----结束

搜索作业

步骤 1 在“作业浏览器”的搜索栏，输入指定的字符，系统会按照 ID、名称、用户自动搜索包含此关键字的全部作业。

步骤 2 清空搜索框的内容，系统会重新显示所有作业。

----结束

查看作业详细信息

步骤 1 在“作业浏览器”的作业列表，单击作业所在的行，可以打开作业详情。

步骤 2 在“元数据”页签，可查看作业的元数据。

说明

单击“日志”可打开作业运行时的日志。

----结束

13.9 在 Hue WebUI 使用 HBase

操作场景

用户需要使用图形化界面在集群中创建或查询 HBase 表时，可以通过 Hue 完成任务。

需要 MRS 集群已安装 HBase 组件并添加 Thrift1Server 实例。

访问作业浏览器

步骤 1 访问 Hue WebUI，请参考访问 Hue 的 WebUI。

步骤 2 单击 HBase  ，进入“HBase Browser”页面。

----结束

新建 HBase 表

步骤 1 访问 Hue WebUI，请参考访问 Hue 的 WebUI。

步骤 2 单击 HBase  ，进入“HBase Browser”页面。

步骤 3 单击右侧“新建表”按钮，输入表名和列族参数，单击“提交”，完成 HBase 表创建。

----结束

查询 HBase 表数据

步骤 1 访问 Hue WebUI，请参考访问 Hue 的 WebUI。

步骤 2 单击 HBase  ，进入“HBase Browser”页面。

步骤 3 单击需要查询的 HBase 表。可在上方的搜索栏后单击健值，对 HBase 表进行查询。

----结束

13.10 典型场景

13.10.1 HDFS on Hue

Hue 提供了文件浏览器功能，使用户可以通过界面图形化的方式使用 HDFS。

⚠ 注意

Hue 界面主要用于文件、表等数据的查看与分析，禁止通过 Hue 界面对操作对象进行删除等高危管理操作。如需操作，建议在确认对业务没有影响后通过各组件的相应操作方法进行处理，例如使用 HDFS 客户端对 HDFS 文件进行操作，使用 Hive 客户端对 Hive 表进行操作。

文件浏览器使用介绍

访问 Hue WebUI，请参考访问 Hue 的 WebUI。

然后单击  ，进入“文件浏览器”页面。您可以进行以下操作。

- 查看文件和目录

  默认显示登录用户的目录及目录中的文件，可查看目录或文件的“名称”、“大小”、“用户”、“组”、“权限”和“日期”信息。

  单击文件名，可查看文本文件的文本信息或二进制数据。支持编辑文件内容。

  如果文件和目录数量比较多，可以在搜索框输入关键字，搜索特定的文件或目录。

- 创建文件或目录

  单击右上角的“新建”，选择“文件”创建文件，选择“目录”创建目录。

- 管理文件或目录

  勾选文件或目录的复选框，单击“操作”，选择“重命名”、“移动”、“复制”和“更改权限”等，实现文件或目录的重命名、移动、复制、更改权限等功能。

- 上传文件

  单击右上角的“上传”，单击“选择文件”或将文件拖至窗口中可进行文件上传。

存储策略定义使用介绍

说明

若 Hue 的服务配置参数“fs_defaultFS”配置为“viewfs://ClusterX”时，不能启用存储策略定义功能。

存储策略定义在 Hue 的 WebUI 界面上分为两大类：

- 静态存储策略

  当前存储策略

  根据 HDFS 的文档访问频率、重要性，为 HDFS 目录指定存储策略，例如 ONE_SSD、ALL_SSD 等，此目录下的文件可被迁移到相应存储介质上保存。

- 动态存储策略

  为 HDFS 目录设置规则，系统可以根据文件的最近访问时间、最近修改时间自动修改存储策略、修改文件副本数、移动文件目录。

  在 Hue 的 WebUI 界面设置动态存储策略之前，需先在 Manager 界面设置冷热数据迁移的 CRON 表达式，并启动自动冷热数据迁移特性。

  操作方法为：

  修改 HDFS 服务的 NameNode 的如下参数值。参数修改方法请参考修改集群服务配置参数。

参数	描述	取值示例
dfs.auto.data.mover.enable	表示是否启用自动冷热数据迁移特性。默认值是“false”。	true
dfs.auto.data.mover.cron.expression	HDFS 执行冷热数据迁移的 CRON 表达式，用于控制数据迁移操作的开始时间。仅当“dfs.auto.data.mover.enable”	0 * * * *

参数	描述	取值示例
	设置为“true”时才有效。默认值“0 * * * *”表示在每个整点执行任务。	

修改参数“dfs.auto.data.mover.cron.expression”时，表达式介绍如表 13-4 所示。支持“*”表示连续的时间段。

表13-4 执行表达式参数解释

列	说明
第 1 列	分钟，参数值为 0～59。
第 2 列	小时，参数值为 0～23。
第 3 列	日期，参数值为 1～31。
第 4 列	月份，参数值为 1～12。
第 5 列	星期，参数值为 0～6，0 表示星期日。

存储策略定义在 WebUI 界面上的操作如下：

步骤 1 登录 FusionInsight Manager，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本）。

步骤 2 在 FusionInsight Manager 界面，选择“系统 > 权限 > 角色 > 添加角色”：

1. 设置“角色名称”。
2. 在“配置资源权限”下选择“*待操作集群名称* >Hue”，勾选“存储策略管理员”，单击“确定”，为该角色赋予存储策略管理员的权限。

步骤 3 选择“系统 > 权限 > 用户组 > 添加用户组”，设置“组名”，单击“角色”后的“添加”，在弹出的界面选择步骤 2 创建的角色，单击“确定”将该角色添加到组中，单击“确定”完成用户组的创建。

步骤 4 选择“系统 > 权限 > 用户 > 添加用户”：

1. “用户名”填写待添加的用户名。
2. “用户类型”设置为“人机”。
3. 设置登录 Hue 的 WebUI 界面的“密码”、“确认密码”。
4. 单击“用户组”后的“添加”，在弹出的界面选择步骤 3 创建的用户组、supergroup、hadoop 和 hive 用户组，单击“确定”。
5. “主组”选择“hive”。
6. 单击“角色”后的“添加”，在弹出的界面选择步骤 2 创建的角色和 System_administrator 角色，单击“确定”。
7. 再单击“确定”，成功添加该用户。

步骤 5 使用创建的用户访问 Hue WebUI，请参考访问 Hue 的 WebUI。

步骤 6 左侧导航栏单击文件  。进入“文件浏览器”页面。

步骤 7 勾选目录的复选框，单击页面上方的“操作”，单击“存储策略”。

步骤 8 在弹出的对话框中设置新的存储策略，单击“确定”。

- 在“静态存储策略”页签设置静态存储策略，单击“保存”。
- 在“动态存储策略”页签可创建、删除、修改动态存储策略，详细的参数介绍如表 13-5 所示。

表13-5 动态存储策略参数介绍

分类	参数	说明
规则	文件最近访问时间	按照该文件最近一次访问时间。
	文件最近修改时间	按照该文件最近一次修改时间。
操作	修改副本数	设置文件副本数。
	修改存储策略	修改存储策略，包括 HOT、WARM、COLD、ONE_SSD、ALL_SSD。
	移动到目录	移动该文件到其他目录。

说明

- 设置规则需要用户充分考虑合理性，例如多条规则之间是否有冲突，是否会对系统造成破坏等。
- 一个目录设置多个规则和动作时，规则被先触发的放在规则/动作列表的下面，规则被后触发的放在规则/动作列表的上面，避免动作反复执行。
- 系统每个小时整点扫描动态存储策略指定的目录下的文件是否符合规则，如果满足，则触发执行动作。执行日志记录在主 NameNode 的“/var/log/Bigdata/hdfs/nn/hadoop.log”目录下。

----结束

典型场景

通过 Hue 界面对 HDFS 以文本或二进制查看和编辑文件的操作如下：

查看文件

步骤 1 访问 Hue WebUI，请参考访问 Hue 的 WebUI。

步骤 2 左侧导航栏单击文件  。进入“文件浏览器”页面。

步骤 3 单击需要查看的文件名。

步骤 4 单击“以二进制格式查看”，可以切换视图从文本到二进制；单击“以文本格式查看”，可以切换视图从二进制到文本。

编辑文件

步骤 5 单击“编辑文件”，显示文件内容可编辑。

步骤 6 单击“保存”或“另存为”保存文件。

----结束

13.10.2 配置 HDFS 冷热数据迁移

配置场景

冷热数据迁移工具根据配置的策略移动 HDFS 文件。配置策略是条件或非条件规则的集合。如果规则匹配文件集，则该工具将对该文件执行一组行为操作。

冷热数据迁移工具支持以下规则和行为。

- 迁移规则：
  - 根据文件的最后访问时间迁移数据
  - 根据年龄时间迁移数据（修改时间）
  - 无条件迁移数据

表13-6 规则条件标签

条件标签	描述
<age operator="lt">	定义年龄/修改时间的条件。
<atime operator="gt">	定义访问时间的条件。

说明

对于手动迁移规则，不需要条件。

- 行为列表：
  - 将存储策略设置为给定的数据层名称
  - 迁移到其他文件夹
  - 为文件设置新的副本数
  - 删除文件
  - 设置节点标签（NodeLabel）

表13-7 行为类型

行为类型	描述	所需参数
MARK	为确定数据的冷热度并设置	<param>

行为类型	描述	所需参数
	相应的数据存储策略。	<name>targettier</name> <value>STORAGE_POLICY</value> <param>
MOVE	为设置数据存储策略或 NodeLabel 并调用 HDFS Mover 工具。	<param> <name>targettier</name> <value>STORAGE_POLICY</value> <param> <param> <name>targetnodelabels</name> <value>SOME_EXPRESSION</value> <param> 说明 用户可以配置其中任一参数或两者都配置。
SET_REPL	为文件设置新的副本数。	<param> <name>replcount</name> <value>INTEGER</value> <param>
MOVE_TO_FOLDER	将文件移动到目标文件夹。如果“overwrite”参数为“true”，则目标路径将被覆盖。	<param> <name>target</name> <value>PATH</value> <param> <param> <name>overwrite</name> <value>true/false</value> <param> 说明 “overwrite”是可选参数，如果未配置，则默认值为“false”。
DELETE	删除文件。	NA

配置描述

必须定期调用迁移工具，并需要在客户端的“hdfs-site.xml”文件中进行以下配置。

表13-8 参数描述

参数	描述	默认值
dfs.auto-data-	用于指定默认的数据迁移策略。	com.xxx.hadoop.hdfs.datamovement.policy.

参数	描述	默认值
movement.policy.class	说明 当前只支持 DefaultDataMovementPolicy。	DefaultDataMovementPolicy
dfs.auto.data.mover.id	冷热数据迁移输出（行为状态）文件的名称。	当前系统时间（毫秒）
dfs.auto.data.mover.output.dir	冷热数据迁移输出在 HDFS 中的目录名称。迁移工具将在此处写入行为状态文件。	/system/datamovement

DefaultDataMovementPolicy 拥有配置文件“default-datamovement-policy.xml”。用户需要定义所有基于 age/accessTime 的规则和在此文件中采取的行为操作，此文件必须存储在客户端的 classpath 中。

如下为“default-datamovement-policy.xml”配置文件的示例：

```
<policies>
  <policy>
    <fileset>
      <file>
        <name>/opt/data/1.txt</name>
      </file>
      <file>
        <name>/opt/data/*/subpath/</name>
        <excludes>
          <name>/opt/data/some/subpath/sub1</name>
        </excludes>
      </file>
    </fileset>
    <rules>
      <rule>
        <age>2w</age>
        <action>
          <type>MOVE</type>
          <params>
            <param>
              <name>targettier</name>
              <value>HOT</value>
            </param>
          </params>
        </action>
      </rule>
    </rules>
  </policy>
</policies>
```

说明

在策略，规则和行为操作中使用的标签中，可以添加其他属性，例如“name”可用于管理用户界面（例如：Hue UI）和工具输入 xml 之间的映射。

示例： <policy name="Manage_File1">

标签（Tag）说明如下：

表13-9 配置标签（Tag）描述

标签（Tag）名称	描述	是否可重复使用
<policy>	定义单一策略。 • idempotent 属性：指定当策略中有多个规则时，如果满足当前规则，是否检查下一个规则。 示例：<policy name ="policy2" idempotent ="true">。 其默认值为“true”，表示其中的规则和行为操作是幂等的，可以继续检查下一个规则。如果值为“false”，则将在当前规则处停止评估。 • hours_allowed 属性：配置是否根据系统时间执行策略评估。 hours_allowed 的值是以逗号分隔的数字，范围从 0 到 23，表示系统时间。 示例：<policy name ="policy1" hours_allowed ="2-6,13-14"> 如果当前系统时间在配置的范围内，则继续评估。否则，将跳过评估。 说明 在输入 XML 中，每个文件仅支持一个策略。因此，文件中的所有规则必须由一个策略标签覆盖。	Yes
<fileset>	为每个策略定义一组文件/文件夹。	No （在 policy 标签内）
<file>	定义文件和/或文件夹在<file>标签内被配置一个或者多个<name>标签。文件/文件夹名支持 POSIX globs 配置。	Yes （在 fileset 标签内）
<excludes>	在<file>标签内定义该标签，该标签下可以包含多个<name>标签，在<file>标签中配置的文件或文件夹范围下，<name>标签所包含的文件或文件夹将会被排除。文件或文件夹名支持 POSIX globs 配置。	No （在 fileset 标签内）
<rules>	针对策略定义多个规则。	No （在 policy 标签内）
<rule>	定义单一规则。	Yes （在 rules 标签内）
<age>or<at	定义在<fileset>中定义的文件的 age/accesstime。策略	No （在 rule

标签（Tag）名称	描述	是否可重复使用
ime>	将匹配该 age。age 可以用 [num]y[num]m[num]w[num]d[num]h 的格式表示。其中 num 表示数字。 其中字母的意思如下： * y--年（一年是 365 天）。 * m--月（一个月是 30 天）。 * w--周（一周是 7 天）。 * d--天。 * h--小时。 可以单独使用年，月，周，天或小时，也可以将他们组合。比如，1y2d 表示 1 年零 2 天或者 367 天。 如果没有单位（即数字后面没有任何上述字母），默认单位为天。 说明 用户可以在<age>和<atime>标签中配置“gt”（greater）和“lt”（less），默认运算符为“gt”。 示例：<age operator="lt">	标签内）
<action>	如果规则匹配，这个标签定义了要执行的 action。	No (在 rule 标签内)
<type>	定义了 action 类型。当前支持的 action 类型是 MOVE 和 MARK。	No (在 action 标签内)
<**params**>	定义与每个 action 相关的参数。	No (在 action 标签内)
<param>	定义单个使用<name>和<value>标签的 name-value 格式参数。 对于 MARK 和 MOVE，只支持参数名“targettier”。该参数表示如果满足 age 规则，则指定数据存储策略。 如果多个 param 中具有相同 name 的参数，则采用第一个参数值。 对于 MARK，支持的“targettier”参数值为“ALL_SSD”，“ONE_SSD”，“HOT”，“WARM”，“COLD”。 对于 MOVE，支持的“targettier”参数值为“ALL_SSD”，“ONE_SSD”，“HOT”，“WARM”和“COLD”。	Yes (在 params 标签内).

对于在<file>标签下的文件/文件夹我们使用 FileSystem#globStatus API，对于其他的我们使用 GlobPattern 类（被 GlobFilter 使用）。参照支持的 API 的细节。例如，对于 globStatus，“/opt/hadoop/*”将匹配“/opt/hadoop”文件夹下的一切。“/opt/*/hadoop”将匹配/“opt”目录的子目录下的所有 hadoop 文件夹。

对于 globStatus，分别匹配每个路径组件的 glob 模式，而对于其他的，直接匹配 glob 模式。

https://hadoop.apache.org/docs/r3.1.1/api/org/apache/hadoop/fs/FileSystem.html#globStatus(org.apache.hadoop.fs.Path)

Glob	Name	Matches
*	*asterisk*	Matches zero or more characters
?	*question mark*	Matches a single character
[ab]	*character class*	Matches a single character in the set {a, b}
[^ab]	*negated character class*	Matches a single character that is not in the set {a, b}
[a-b]	*character range*	Matches a single character in the (closed) range [a, b], where a is lexicographically less than or equal to b
[^a-b]	*negated character range*	Matches a single character that is not in the (closed) range [a, b], where a is lexicographically less than or equal to b
{a,b}	*alternation*	Matches either expression a or b
\c	*escaped character*	Matches character c when it is a metacharacter

行为操作示例

- MARK

```
<action>
  <type>MARK</type>
  <params>
    <param>
      <name>targettier</name>
      <value>HOT</value>
    </param>
  </params>
</action>
```

- MOVE

```
<action>
  <type>MOVE</type>
  <params>
    <param>
      <name>targettier</name>
      <value>HOT</value>
    </param>
    <param>
      <name>targetnodelabels</name>
      <value>SOME_EXPRESSION</value>
    </param>
  </params>
</action>
```

- SET_REPL

```
<action>
  <type>SET_REPL</type>
  <params>
    <param>
      <name>replcount</name>
      <value>5</value>
    </param>
  </params>
</action>
```

- MOVE_TO_FOLDER

```
<action>
  <type>MOVE_TO_FOLDER</type>
  <params>
    <param>
      <name>target</name>
      <value>path</value>
    </param>
    <param>
      <name>overwrite</name>
      <value>true</value>
    </param>

  </params>
</action>
```

说明

MOVE_TO_FOLDER 操作只是将文件路径更改为目标文件夹，不会更改块位置。如果想要移动块，则需要配置一个独立的 move 策略。

- DELETE

```
<action>
  <type>DELETE</type>
</action>
```

说明

- 在编写 xml 文件时，用户应该注意行为操作的配置和顺序。冷热数据迁移工具按照输入 xml 中给定的顺序执行规则。
- 如果只希望运行基于 atime/age 的一个规则，则按照时间逆序排列，且将 idempotent 属性设置为 false。
- 如果为文件集配置删除操作，则在删除操作后不能再配置其他规则。
- 支持使用"-fs"选项，用于指定客户端默认的文件系统地址。

审计日志

冷热数据迁移工具支持以下操作的审计日志。

- 工具启动状态
- 行为类型及参数详细信息和状态
- 工具完成状态

对于启用审计日志工具，在“<HADOOP_CONF_DIR>/log4j.property”文件中添加以下属性。

```
autodatatool.logger=INFO, ADMTRFA
autodatatool.log.file=HDFSAutoDataMovementTool.audit
log4j.logger.com.xxx.hadoop.hdfs.datamovement.HDFSAutoDataMovementTool.audit=${auto
datatool.logger}
log4j.additivity.com.xxx.hadoop.hdfs.datamovement.HDFSAutoDataMovementTool-
audit=false
log4j.appender.ADMTRFA=org.apache.log4j.RollingFileAppender
log4j.appender.ADMTRFA.File=${hadoop.log.dir}/${autodatatool.log.file}
log4j.appender.ADMTRFA.layout=org.apache.log4j.PatternLayout
log4j.appender.ADMTRFA.layout.ConversionPattern=%d{ISO8601} %p %c: %m%n
log4j.appender.ADMTRFA.MaxBackupIndex=10
log4j.appender.ADMTRFA.MaxFileSize=64MB
```

说明

具体请参考“<HADOOP_CONF_DIR>/log4j_autodata_movment_template.properties”文件。

13.10.3 Hive on Hue

Hue 提供了 Hive 图形化管理功能，使用户可以通过界面的方式查询 Hive 的不同数据。

查询编辑器使用介绍

访问 Hue WebUI，请参考访问 Hue 的 WebUI。

在左侧导航栏单击编辑器，然后选择“Hive”，进入“Hive”。

- 执行 Hive HQL 语句

  在左侧选中目标数据库，也可通过单击右上角的 default，输入目标数据库的名称以搜索目标数据库。

  在文本编辑框输入 Hive HQL 语句，单击 或者按“Ctrl+Enter”，运行 HQL 语句，执行结果将在“结果”页签显示。

- 分析 HQL 语句

  在左侧选中目标数据库，在文本编辑框输入 Hive HQL 语句，单击 编译 HQL 语句并显示语句是否正确，执行结果将在文本编辑框下方显示。

- 保存 HQL 语句

  在文本编辑框输入 Hive HQL 语句，单击右上角的，并输入名称和描述。已保存的语句可以在“保存的查询”页签查看。

- 查看历史

  单击“查询历史记录”，可查看 HQL 运行情况，支持显示所有语句或只显示保存的语句的运行情况。历史记录存在多个结果时，可以在输入框使用关键字进行搜索。

- 高级查询配置

  单击右上角的，对文件、函数、设置等信息进行配置。

- 查看快捷键

  单击右上角的，可查看所有快捷键信息。

元数据浏览器使用介绍

访问 Hue WebUI，请参考访问 Hue 的 WebUI。

- 查看 Hive 表的元数据

  在左侧导航栏单击表，单击某一表名称，界面将显示 Hive 表的元数据信息。
- 管理 Hive 表的元数据

  在 Hive 表的元数据信息界面：
  - 单击右上角的“导入”可导入数据。
  - 单击“概述”，在“属性”域可查看表文件的位置信息。

    可查看 Hive 表各列字段的信息，并手动添加描述信息，注意此处添加的描述信息并不是 Hive 表中的字段注释信息（comment）。
  - 单击“样本”可浏览数据。
- 管理 Hive 元数据表

  单击左侧列表中的可在数据库中根据上传的文件创建一个新表，也可手动创建一个新表。

注意

Hue 界面主要用于文件、表等数据的查看与分析，禁止通过 Hue 界面对操作对象进行删除等高危管理操作。如需操作，建议在确认对业务没有影响后通过各组件的相应操作方法进行处理，例如使用 HDFS 客户端对 HDFS 文件进行操作，使用 Hive 客户端对 Hive 表进行操作。

典型场景

通过 Hue 界面对 Hive 进行创建表的操作如下：

步骤 1 单击 Hue 的 WebUI 界面左上角的，选择要操作的 Hive 实例，进入 Hive 命令的执行页面。

步骤 2 在命令输入框内输入一条 HQL 语句，例如：

create table hue_table(id int,name string,company string) row format delimited fields terminated by ',' stored as textfile;

单击 执行 HQL。

步骤 3 在命令输入框内输入：

show tables;

单击 ▶，查看“结果”中有创建的表 hue_table。

----结束

13.10.4 Oozie on Hue

Hue 提供了 Oozie 作业管理器功能，使用户可以通过界面图形化的方式使用 Oozie。

> **⚠ 注意**
>
> Hue 界面主要用于文件、表等数据的查看与分析，禁止通过 Hue 界面对操作对象进行删除等高危管理操作。如需操作，建议在确认对业务没有影响后通过各组件的相应操作方法进行处理，例如使用 HDFS 客户端对 HDFS 文件进行操作，使用 Hive 客户端对 Hive 表进行操作。

Oozie 作业设计器使用介绍

访问 Hue WebUI，请参考访问 Hue 的 WebUI。

在左侧导航栏单击  ，选择“Workflow”。

在作业设计器，支持用户创建 MapReduce、Java、Streaming、Fs、Ssh、Shell 和 DistCp 作业。

仪表板使用介绍

访问 Hue WebUI，请参考访问 Hue 的 WebUI。

选择右上角“作业”，进入“作业浏览器”。

支持查看 Workflow、Coordinator 和 Bundles 作业的运行情况。

编辑器使用介绍

访问 Hue WebUI，请参考访问 Hue 的 WebUI。

在左侧导航栏单击  ，然后选择“Workflow”。

支持创建 Workflow、计划和 Bundles 的操作。支持提交运行、共享、复制和导出已创建的应用。

- 每个 Workflow 可以包含一个或多个作业，形成完整的工作流，用于实现指定的业务。

  创建 Workflow 时，可直接在 Hue 的编辑器设计作业，并添加到 Workflow 中。
- 每个计划可定义一个时间触发器，用于定时触发执行一个指定的 Workflow。不支持多个 Workflow。
- 每个 Bundles 可定义一个集合，用于触发执行多个计划，使不同 Workflow 批量执行。

13.11 Hue 日志介绍

日志描述

日志路径：Hue 相关日志的默认存储路径为“/var/log/Bigdata/hue”（运行日志），“/var/log/Bigdata/audit/hue”（审计日志）。

日志归档规则：Hue 的日志启动了自动压缩归档功能，默认情况下，当“access.log”、“error.log”、“runcpserver.log”和“hue-audits.log”大小超过 5MB 的时候，会自动压缩。最多保留最近的 20 个压缩文件，压缩文件保留个数和压缩文件阈值可以配置。

表13-10 Hue 日志列表

日志类型	日志文件名	描述
运行日志	access.log	访问日志。
	error.log	错误日志。
	gsdb_check.log	gaussDB 检查日志。
	kt_renewer.log	Kerberos 认证日志。
	kt_renewer.out.log	Kerberos 认证日志的异常输出日志。
	runcpserver.log	操作记录日志。
	runcpserver.out.log	进程运行异常日志。
	supervisor.log	进程启动日志。
	supervisor.out.log	进程启动异常日志。
	dbDetail.log	数据库初始化日志
	initSecurityDetail.log	keytab 文件下载初始化日志。
	postinstallDetail.log	Hue 服务安装后工作日志。
	prestartDetail.log	Prestart 日志。
	statusDetail.log	Hue 服务健康状态日志。

日志类型	日志文件名	描述
	startDetail.log	启动日志。
	get-hue-ha.log	Hue HA 状态日志。
	hue-ha-status.log	Hue HA 状态监控日志。
	get-hue-health.log	Hue 健康状态日志。
	hue-health-check.log	Hue 健康检查日志。
	hue-refresh-config.log	Hue 配置刷新日志。
	hue-script-log.log	Manager 界面的 Hue 操作日志。
	hue-service-check.log	Hue 服务状态监控日志。
	db_pwd.log	Hue 连接 DBService 数据库密码修改日志
	modifyDBPwd_*日期*.log	-
	watch_config_update.log	参数更新日志。
审计日志	hue-audits.log	审计日志。

日志级别

Hue 提供了如表 13-11 所示的日志级别。

日志的级别优先级从高到低分别是 ERROR、WARN、INFO、DEBUG，程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。

表13-11 日志级别

级别	描述
ERROR	ERROR 表示系统运行的错误信息。
WARN	WARN 表示当前事件处理存在异常信息。
INFO	INFO 表示记录系统及各事件正常运行状态信息。
DEBUG	DEBUG 表示记录系统及系统的调试信息。

如果您需要修改日志级别，请执行如下操作：

步骤 1 参考修改集群服务配置参数进入 Hue 服务“全部配置”页面。

步骤 2 在左侧导航栏选择需修改的角色所对应的“日志”菜单。

步骤 3 在右侧选择所需修改的日志级别。

步骤 4 保存配置，在弹出窗口中单击“确定”使配置生效。

步骤 5 重新启动配置过期的服务或实例以使配置生效。

----结束

日志格式

Hue 的日志格式如下所示：

表13-12 日志格式

日志类型	格式	示例
运行日志	<dd-MM-yy HH:mm:ss,SSS><日志事件的发生位置><log level><log 中的 message>	[03/Nov/2014 11:57:19] middleware | INFO | Unloading MimeTypeJSFileFixStreamingMiddleware.
	<Log Level><时间格式><yyyy-MM-dd HH:mm:ss,SSS><日志事件的发生位置><log 中的 message>	INFO : CST 2014-11-06 11:22:52 hue-ha-status.sh : update 4 <= 15:myHostName=10.0.0.250 ACTIVE=10.0.0.250
审计日志	<UserName><yyyy-MM-dd HH:mm:ss,SSS><审计操作描述><资源参数><url><是否允许><审计操作><ip 地址>	{"username": "admin", "eventTime": "2014-11-06 10:28:34", "operationText": "Successful login for user: admin", "service": "accounts", "url": "/accounts/login/", "allowed": true, "operation": "USER_LOGIN", "ipAddress": "10.0.0.250"}

13.12 Hue 常见问题

13.12.1 如何解决使用 IE 浏览器在 Hue 中执行 HQL 失败的问题

问题

遇到使用 IE 浏览器在 Hue 中访问 Hive Editor 并执行所有 HQL 失败，界面提示“There was an error with your query.”，如何解决并正常执行 HQL？

回答

IE 浏览器存在功能问题，不支持在 307 重定向中处理含有 form data 的 AJAX POST 请求，建议更换兼容的浏览器，例如 Google Chrome 浏览器。

13.12.2 在使用 Hive 时，输入 use database 语句失效了

问题

使用 Hive 的时候，在输入框中输入了 **use database** 的语句切换数据库，重新在输入框内输入其他语句，为什么数据库没有切换过去？

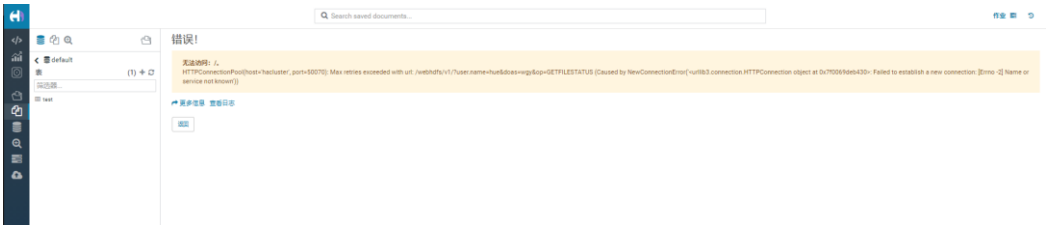
回答

在 Hue 上使用 Hive 有区别于用 Hive 客户端使用 Hive，Hue 界面上有选择数据库的按钮，当前 SQL 执行的数据库以界面上显示的数据库为准。与此相关的还有设置参数等 session 级别的一次性操作，都应该使用界面功能进行设置，不建议使用输入语句进行操作。若是必须使用输入语句进行操作，需保证所有语句在同一个输入框内。

13.12.3 如何处理使用 Hue WebUI 访问 HDFS 文件失败的问题

问题

在使用 Hue WebUI 访问 HDFS 文件时，报如下图所示无法访问的错误提示，该如何处理？



回答

1. 查看登录 Hue WebUI 的用户是否具有“hadoop”用户组权限。
2. 查看 HDFS 服务是否安装了 HttpFS 实例且运行正常。如果未安装 HttpFS 实例，需手动安装并重启 Hue 服务。

13.12.4 Hue 页面上传大文件失败如何处理

问题

通过 Hue 页面上传大文件时，上传失败。

回答

1. 不建议使用 Hue 文件浏览器上传大文件，大文件建议使用客户端通过命令上传。
2. 如果必须使用 Hue 上传，参考以下步骤修改 Httpd 的参数：

a. 以 **omm** 用户登录主管理节点。

b. 执行以下命令编辑“httpd.conf”配置文件。

vi $BIGDATA_HOME/om-server/Apache-httpd-*/conf/httpd.conf

c. 搜索 21201，在</VirtualHost>配置中加上“RequestReadTimeout handshake=0 header=0 body=0”，如下所示。

```
...
<VirtualHost *:21201>
   ServerName https://10.112.16.93:21201
   AllowEncodedSlashes On
   SSLProxyEngine On
   ProxyRequests Off
   TraceEnable off
   ProxyTimeout  1200
   RewriteEngine on
   RewriteMap proxylist dbm:${BIGDATA_ROOT_HOME}/om-server_*/Apache-httpd-
*/conf/proxylist.dbm

   RewriteRule ^(\/.*)$  ${proxylist:/Hue/Hue/21201}$1
[E=TARGET_PATH:$1,L,P]

   Header edit Location
^(?!https://10.112.16.93:20009|https://10.112.16.93:21201)http[s]?://[^/]*
(.*)$  https://10.112.16.93:21201$1

   ProxyPassReverseCookiePath / / interpolate

   SSLEngine On
   SSLProxyProtocol  All +TLSv1.2 -SSLv2 -SSLv3 -TLSv1 -TLSv1.1
   SSLProtocol ALL +TLSv1.2 -SSLv2 -SSLv3 -TLSv1 -TLSv1.1
   SSLCipherSuite ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:DHE-DSS-
AES256-GCM-SHA384:DHE-RSA-AES256-GCM-SHA384:DHE-DSS-AES128-GCM-SHA256:DHE-
RSA-AES128-GCM-SHA256
   SSLProxyCheckPeerName off
   SSLProxyCheckPeerCN off
   SSLCertificateFile "${BIGDATA_ROOT_HOME}/om-server_*/Apache-httpd-
*/conf/security/proxy_ssl.cert"
   SSLCertificateKeyFile "${BIGDATA_ROOT_HOME}/om-server_*/Apache-httpd-
*/conf/security/server.key"
   SSLProxyCACertificateFile ${BIGDATA_ROOT_HOME}/om-server_*/apache-
tomcat-*/conf/security/tomcat.crt
   SSLCertificateChainFile "${BIGDATA_ROOT_HOME}/om-server_*/Apache-httpd-
2.4.39/conf/security/proxy_chain.cert"
   RequestReadTimeout handshake=0 header=0 body=0
</VirtualHost>
...
```

d. 执行 **pkill -9 httpd** 命令结束 httpd 进程，并等待自动重启 httpd。

13.12.5 集群未安装 Hive 服务时 Hue 原生页面无法正常显示

问题

集群没有安装 Hive 服务时，Hue 服务原生页面显示空白。

回答

MRS 3.x 版本存在 Hue 依赖 Hive 组件，如果出现此情况，首先需要检查当前集群是否安装了 Hive 组件，如果没有，需要安装 Hive。

13.12.6 Hue WebUI 中 Oozie 编辑器的时区设置问题

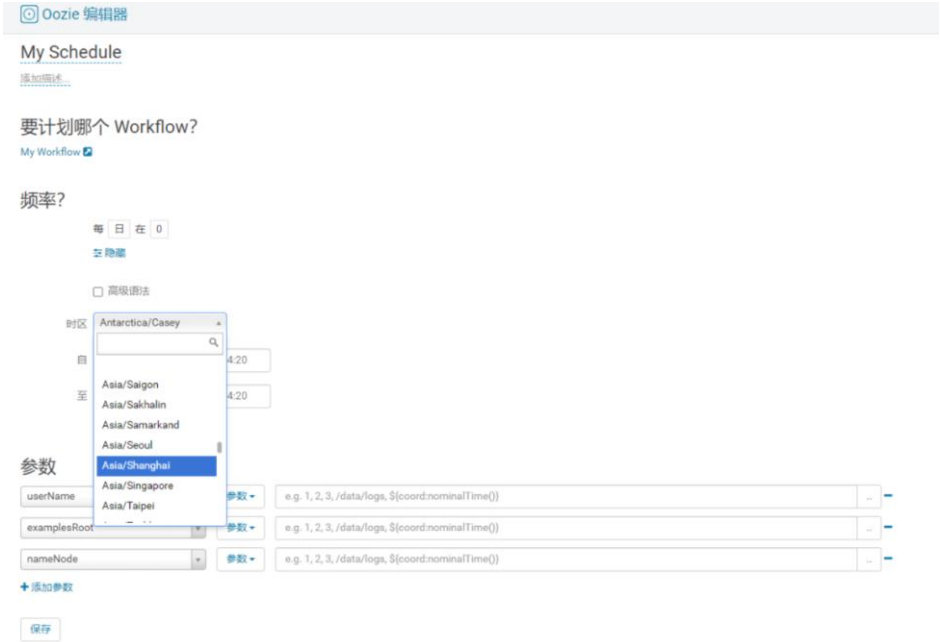
问题

在 Hue 设置 Oozie 工作流调度器的时区时，部分时区设置会导致任务提交失败。

回答

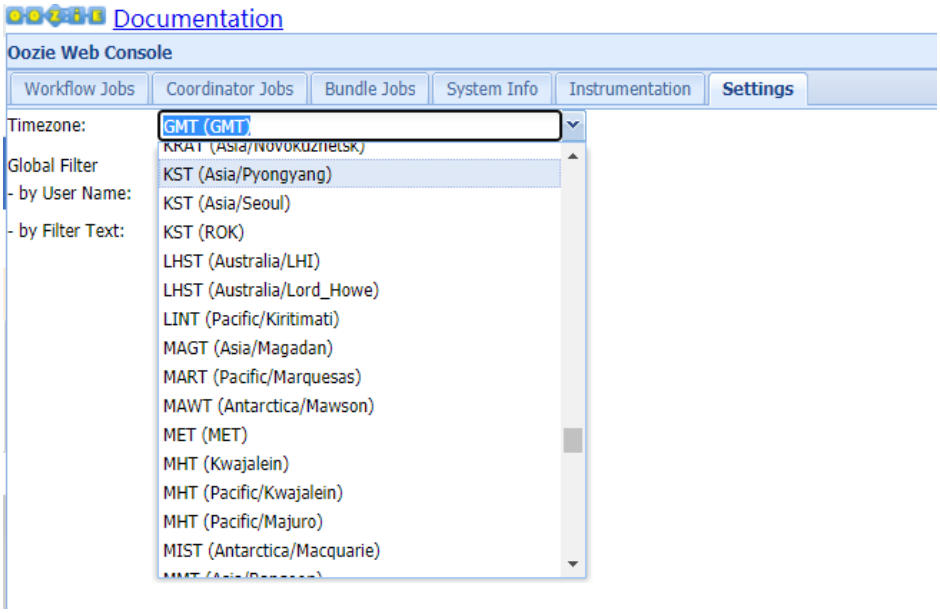
部分时区存在适配问题，建议时区选择“Asia/Shanghai”，如图 13-1 所示。

图13-1 时区选择



支持的时区可以参考 Oozie WebUI 页面“Settings”页签的“Timezone”，如图 13-2。

图13-2 时区参考


Documentation
Oozie Web Console
Workflow Jobs
Coordinator Jobs
Bundle Jobs
System Info
Instrumentation
Settings
Timezone:
GMT (GMT)
Global Filter
- by User Name:
- by Filter Text:
KST (Asia/Pyongyang)
KST (Asia/Seoul)
KST (ROK)
LHST (Australia/LHI)
LHST (Australia/Lord_Howe)
LINT (Pacific/Kiritimati)
MAGT (Asia/Magadan)
MART (Pacific/Marquesas)
MAWT (Antarctica/Mawson)
MET (MET)
MHT (Kwajalein)
MHT (Pacific/Kwajalein)
MHT (Pacific/Majuro)
MIST (Antarctica/Macquarie)

14 使用 Impala

14.1 从零开始使用 Impala

Impala 是用于处理存储在 Hadoop 集群中的大量数据的 MPP（大规模并行处理）SQL 查询引擎。 它是一个用 C++和 Java 编写的开源软件。 与其他 Hadoop 的 SQL 引擎相比，它拥有高性能和低延迟的特点。

背景信息

假定用户开发一个应用程序，用于管理企业中的使用 A 业务的用户信息，使用 Impala 客户端实现 A 业务操作流程如下：

普通表的操作：

- 创建用户信息表 user_info。
- 在用户信息中新增用户的学历、职称信息。
- 根据用户编号查询用户姓名和地址。
- A 业务结束后，删除用户信息表。

表14-1 用户信息

编号	姓名	性别	年龄	地址
12005000201	A	男	19	A 城市
12005000202	B	女	23	B 城市
12005000203	C	男	26	C 城市
12005000204	D	男	18	D 城市
12005000205	E	女	21	E 城市
12005000206	F	男	32	F 城市
12005000207	G	女	29	G 城市
12005000208	H	女	30	H 城市

编号	姓名	性别	年龄	地址
12005000209	I	男	26	I 城市
12005000210	J	女	25	J 城市

前提条件

已安装客户端，例如安装目录为“/opt/hadoopclient”，以下操作的客户端目录只是举例，请根据实际安装目录修改。

操作步骤

步骤 1 以客户端安装用户，登录安装客户端的节点。

步骤 2 执行以下命令，切换到客户端安装目录。

cd /opt/hadoopclient

步骤 3 执行以下命令配置环境变量。

source bigdata_env

步骤 4 运行 Impala 客户端命令，实现 A 业务。

直接执行 Impala 组件的客户端命令：

impala-shell

说明

默认情况下，**impala-shell** 尝试连接到 localhost 的 21000 端口上的 Impala 守护程序。如需连接到其他主机，请使用**-i <host:port>**选项，例如：impala-shell -i xxx.xxx.xxx.xxx:21000。要自动连接到特定的 Impala 数据库，请使用**-d <database>**选项。例如，如果您的所有 Kudu 表都位于数据库“impala_kudu”中，则**-d impala_kudu** 可以使用此数据库。要退出 Impala Shell，请使用 **quit** 命令。

内部表的操作：

1. 根据表 14-1 创建用户信息表 user_info 并添加相关数据。

```
create table user_info(id string,name string,gender string,age int,addr string);
insert into table user_info(id,name,gender,age,addr) values("12005000201","A","男",19,"A 城市");
```

.....（其他语句相同）

2. 在用户信息表 user_info 中新增用户的学历、职称信息。

以增加编号为 12005000201 的用户的学历、职称信息为例，其他用户类似。

```
alter table user_info add columns(education string,technical string);
```

3. 根据用户编号查询用户姓名和地址。

以查询编号为 12005000201 的用户姓名和地址为例，其他用户类似。

```
select name,addr from user_info where id='12005000201';
```

4. 删除用户信息表。

```
drop table user_info;
```

外部分区表的操作：

创建外部分区表并导入数据

1. 创建外部表数据存储路径。
   - 安全模式（集群开启了 Kerberos 认证）：

     cd /opt/hadoopclient

     source bigdata_env

     kinit *hive*

   说明

   用户 hive 需要具有 Hive 管理员权限。

     impala-shell

     hdfs dfs -mkdir /hive

     hdfs dfs -mkdir /hive/user_info
   - 普通模式（集群关闭了 Kerberos 认证）：

     su - omm

     cd /opt/hadoopclient

     source bigdata_env

     impala-shell

     hdfs dfs -mkdir /hive

     hdfs dfs -mkdir /hive/user_info
2. 建表。

```
create external table user_info(id string,name string,gender string,age
int,addr string) partitioned by(year string) row format delimited fields
terminated by ' ' lines terminated by '\n' stored as textfile location
'/hive/user_info';
```

说明

fields terminated 指明分隔的字符,如按空格分隔，''。

lines terminated 指明分行的字符，如按换行分隔，'\n'。

/hive/user_info 为数据文件的路径。

3. 导入数据。
   a. 使用 insert 语句插入数据。

```
insert into user_info partition(year="2018") values ("12005000201","A","男
",19,"A 城市");
```

   b. 使用 load data 命令导入文件数据。
      i. 根据表 14-1 数据创建文件。如，文件名为 txt.log，以空格拆分字段，以换行符作为行分隔符。
      ii. 上传文件至 hdfs。

      hdfs dfs -put txt.log /tmp

iii. 加载数据到表中。

load data inpath '/tmp/txt.log' into table user_info partition (year='2018');

4. 查询导入数据。

```
select * from user_info;
```

5. 删除用户信息表。

```
drop table user_info;
```

----结束

14.2 Impala 常用参数

本章节适用于 MRS 3.x 及后续版本。

参数入口

在 Manager 系统中，选择“集群 > 服务 > Impala > 配置”，选择“全部配置”。在搜索框中输入参数名称。

参数说明

说明

下表仅列举了部分常用参数，实际参数以 Manager 页面为准，参数详情请参见官网 https://docs.cloudera.com/documentation/enterprise/6/properties/6.3/topics/cm_props_cdh630_impala.html。

表14-2 Impala 常用参数

配置参数	说明	默认值	范围
impalad.customized.configs	impalad 进程的自定义配置项。	-	-
--enable_ldap_auth	是否开启 ldap 认证。	false	true 或 false
--ldap_bind_pattern	ldap userDNPattern 例如：cn=%s,ou=People,dc=xxx,dc=com	-	-
--ldap_passwords_in_clear_ok	如果设置为 true，将允许 ldap 密码在网络上明文发送(不含 TLS/SSL)。	false	true 或 false
--ldap_uri-ip	ldap ip	-	-
--ldap_uri-port	ldap port	389	-
--max_log_files	进程日志的最大文件个数。	10	-
--max_log_size	进程的日志文件大小最大值，单位 MB。	200	-

配置参数	说明	默认值	范围
statestored.customized.configs	Statestored 进程的自定义配置项。	-	-
catalogd.customized.configs	Catalogd 进程的自定义配置项。	-	-

14.3 访问 Impala 的 WebUI

用户可以通过 Impala 的 WebUI，在图形化界面查看 Impala 作业的相关信息。Impala 的 WebUI 根据实例不同分为如下三种：

- StateStore WebUI：用于管理节点。
- Catalog WebUI：用于查看元数据。
- Impalad WebUI：用于查看每个 SQL 执行的详细信息。

前提条件

已安装 Impala 服务的集群。

访问 StateStore WebUI

步骤 1 登录 Manager 页面，请参见访问 FusionInsight Manager（MRS 3.x 及之后版本）。

步骤 2 选择“服务管理 > Impala”。

步骤 3 在“Impala 概述”的“StateStore WebUI”中单击“StateStore(Statestore)”，打开 StateStore 的 WebUI 页面。

----结束

访问 Catalog WebUI

步骤 1 登录 Manager 页面，请参见访问 FusionInsight Manager（MRS 3.x 及之后版本）。

步骤 2 选择“服务管理 > Impala”。

步骤 3 在“Impala 概述”的“Catalog WebUI”中单击“Catalog(Catalog)”，打开 Catalog 的 WebUI 页面。

----结束

访问 Impalad WebUI

步骤 1 登录 Manager 页面，请参见访问 FusionInsight Manager（MRS 3.x 及之后版本）。

步骤 2 选择“服务管理 > Impala > 实例”。

步骤 3 移动鼠标至“角色”列的 Impalad 实例上，在页面左下角显示如下链接，获取 null 后的数值，例如本例中的 82。

https://*EIP*:9022/mrsmanager/index.jsp?locale=zh-cn#/app/services/Impala/Impalad/null/82/*EIP*/STARTED/status/detail

其中 82 为样例值，实际值请以实际环境为准。

步骤 4 参考访问 StateStore WebUI。

步骤 5 修改 StateStore WebUI 的 URL 地址中的“StateStore/xx”为“Impalad/**xx**”并访问修改后的 URL，其中 **xx** 为步骤 3 中获取的数值。

----结束

14.4 使用 Impala 操作 Kudu

您可以使用 Impala 的 SQL 语法插入、查询、更新和删除 Kudu 中的数据，作为使用 Kudu API 构建自定义 Kudu 应用程序的替代方案。

前提条件

已安装集群完整客户端。例如安装目录为“/opt/Bigdata/client”，以下操作的客户端目录只是举例，请根据实际安装目录修改。

Impala on Kudu

步骤 1 登录安装客户端的节点。

步骤 2 执行如下命令初始化环境变量。

source /opt/Bigdata/client/bigdata_env

步骤 3 若集群开启 Kerberos 认证，请执行如下步骤认证用户。若集群未开启 Kerberos 认证请跳过该步骤。

kinit *业务用户*

步骤 4 执行如下命令登录 impala 客户端。

impala-shell

说明

默认情况下，**impala-shell** 尝试连接到 localhost 的 21000 端口上的 Impala 守护程序。如需连接到其他主机，请使用**-i <host:port>**选项。要自动连接到特定的 Impala 数据库，请使用**-d <database>**选项。例如，如果您的所有 Kudu 表都位于数据库“impala_kudu”中，则**-d impala_kudu** 可以使用此数据库。要退出 Impala Shell，请使用以下命令 **quit**。

步骤 5 执行如下命令创建 Impala 表并导入已准备好的数据，例如/tmp/data10。

create table dataorigin (name string,age string,pt string, date_p date) row format delimited fields terminated by ',' stored as textfile;

load data inpath '/tmp/data10' overwrite into table dataorigin;

步骤 6 执行如下命令创建 Kudu 表，其中 kudu.master_addresses 地址为 KuduMaster 实例的 IP，请根据实际集群地址填写。

create table dataorigin2 (name string,age string,pt string, date_p date, primary key(name)) stored as kudu TBLPROPERTIES('kudu.master_addresses'='192.168.190.164:7051,192.168.204.178:7051,192.168.244.63:7051');

步骤 7 执行如下命令操作 Kudu 表。

1. 插入数据

   insert into dataorigin2 select * from dataorigin;

2. 更新数据

   UPDATE dataorigin2 SET date_p="2021-03-31" where age="73";

3. 更新或插入行

   UPSERT INTO dataorigin2 VALUES ("spjted","75","28","2021-03-32");
   UPSERT INTO dataorigin2 VALUES ("kwhakb","92","29","2021-03-33");
   UPSERT INTO dataorigin2 VALUES ("oftrkf","13","30","2021-03-34");
   UPSERT INTO dataorigin2 VALUES ("kiewti","36","31","2021-03-35");
   UPSERT INTO dataorigin2 VALUES ("rknmql","98","32","2021-03-36");
   UPSERT INTO dataorigin2 VALUES ("fwcoij","52","33","2021-03-37");
   UPSERT INTO dataorigin2 VALUES ("pgvpdo","37","34","2021-03-35");

4. 删除行

   DELETE FROM dataorigin2 WHERE date_p="2021-03-31";

----结束

14.5 Impala 对接外部 LDAP

本操作适用于 MRS 3.1.0 及之后版本。

步骤 1 登录 Manager。

步骤 2 在 Manager 界面，选择“集群 > *待操作集群的名称* > 服务 > Impala > 配置 > 全部配置 > Impalad（角色） > LDAP”。

步骤 3 配置如下参数的值。

表14-3 参数配置

参数名称	参数描述	备注
--enable_ldap_auth	是否开启 LDAP 认证	【取值范围】 true 或 false
--ldap_bind_pattern	LDAP userDNPattern	例如：cn=#UID,ou=People,dc=xxx,

参数名称	参数描述	备注
		dc=com 或 cn=%s,ou=People,dc=xxx,dc=com
--ldap_passwords_in_clear_ok	LDPA 密码是否以明文发送	如果设置为 true，将允许 LDAP 密码在网络上明文发送 【取值范围】 true 或 false 说明 当“--enable_ldap_auth”设置为“true”时，认证时默认没有开启 Ldap TLS 协议，所以需要将“--ldap_passwords_in_clear_ok”参数设置为“ture”，否则会导致 Impalad 角色启动失败。 如需开启 Ldap TLS 协议则需要在 Impalad 角色的自定义配置中添加配置项“--ldap_tls”为“true”，配置之后密码将支持用密文传输。
--ldap_uri-ip	LDAP IP	-
--ldap_uri-port	LDAP 端口	【默认值】389

步骤 4 修改完成后，单击左上方“保存”，在弹出的对话框中单击“确定”保存配置。

步骤 5 选择“集群 > *待操作集群的名称* > 服务 > Impala > 实例”，勾选配置状态为“配置过期”的实例，选择“更多 > 重启实例”重启受影响的 Impala 实例。

----结束

14.6 Impala 启用并配置动态资源池

本文介绍如何使用动态资源池控制 impala 并发。

问题背景

客户需要使用动态资源池控制 impala 并发。

Pool Config

Property	Value
Max memory (cluster wide)	1048576
Max concurrent queries	-1
Max queue size	200
Queue Timeout (ms)	60000
Min Query MEM_LIMIT range	0
Max Query MEM_LIMIT range	0
Clamp MEM_LIMIT query option	true

1. 登录到集群的 master1 节点上，然后切换到 omm 用户下，在/home/omm 目录下创建 fair-scheduler.xml、llama-site.xml 文件。

```
[omm@node-masterlIoKo impala]$ ll
total 16
-rw-------. 1 omm wheel  708 May 11 23:40 fair-scheduler.xml
-rw-------. 1 omm wheel 1062 May 11 23:53 llama-site.xml
-rw-------. 1 omm wheel 1118 May 11 23:12 llama-site.xml.bak
-rw-------. 1 omm wheel  572 May 11 23:32 update_config.sh
[omm@node-masterlIoKo impala]$
```

2. 打开 fair-scheduler.xml 文件，添加如下配置。

```
<allocations>
   <queue name="root">
      <aclSubmitApps> </aclSubmitApps>
      <queue name="default">
         <maxResources> 1mb, 0 vcores</maxResources><!--参数仅供参考-->
         <aclSubmitApps>*</aclSubmitApps>
      </queue>
      <queue name="development">
         <maxResources>2048 mb, 0 vcores</maxResources><!--参数仅供参考-->
         <aclSubmitApps>admin</aclSubmitApps>
      </queue>
      <queue name="production">
         <maxResources>7168 mb, 0 vcores</maxResources><!--参数仅供参考-->
         <aclSubmitApps>omm</aclSubmitApps>
      </queue>
   </queue>
   <queuePlacementPolicy>
      <rule name="specified" create="false"/>
      <rule name="default" />
   </queuePlacementPolicy>
</allocations>
```

3. 打开 llama-site.xml 文件，添加如下配置：

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <property>
    <name>llama.am.throttling.maximum.placed.reservations.root.default</name>
    <value>1</value>
  </property>
  <property>
    <name>llama.am.throttling.maximum.queued.reservations.root.default</name>
    <value>2</value><!--参数仅供参考-->
```

```
  </property>
  <property>
    <name>impala.admission-control.pool-default-query-
options.root.default</name>
    <value>mem_limit=128m,query_timeout_s=20,max_io_buffers=10</value>
  </property>
  <property>
    <name>impala.admission-control.pool-queue-timeout-ms.root.default</name>
    <value>30000</value><!--参数仅供参考-->
  </property>
  <property>
    <name>impala.admission-control.max-query-mem-limit.root.default</name>
    <value>1024000000</value><!--1GB--><!--参数仅供参考-->
  </property>
  <property>
    <name>impala.admission-control.min-query-mem-limit.root.default</name>
    <value>2048000000</value><!--2GB-->
  </property>
  <property>
    <name>impala.admission-control.clamp-mem-limit-query-
option.root.default.regularPool</name>
    <value>true</value>
  </property>
</configuration>
```

4. 执行如下命令分别将 fair-scheduler.xml、llama-site.xml 同步到所有的 impalad 节点的安装目录的 etc 文件夹下 。

   scp fair-scheduler.xml {impalad 实例 ip}:/opt/Bigdata/FusionInsight_Impala_*/***_Impalad/etc/**

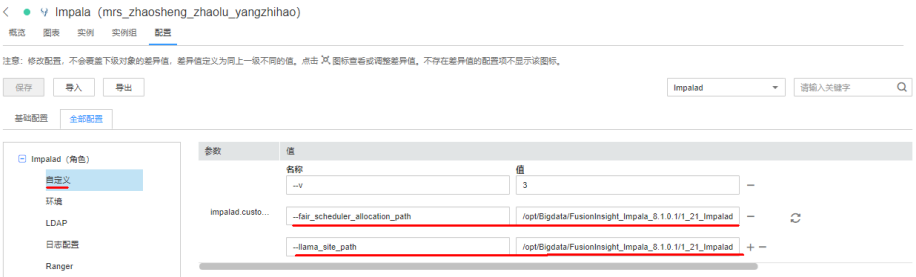
   scp llama-site.xml {impalad 实例 ip}:/opt/Bigdata/FusionInsight_Impala_*/***_Impalad/etc/**

```
+ scp fair-scheduler.xml [redacted]19:/opt/Bigdata/FusionInsight_Impala_8.1.0.1/1_21_Impalad/etc/
Warning: Permanently added '[redacted].19' (ED25519) to the list of known hosts.
```

5. 登录到 manager 管理页面上，找到 impala 组件，然后在 impalad 实例中添加如下自定义配置项及值：

   --fair_scheduler_allocation_path 值如：/opt/Bigdata/FusionInsight_Impala_*/***_Impalad/etc/fair-scheduler.xml**

   --llama_site_path 值如：/opt/Bigdata/FusionInsight_Impala_*/***_Impalad/etc/llama-site.xml**



6. 重启 impalad 实例。

7. 登录到 impala 客户端所在的节点上，source 一下环境变量，然后执行如下命令。

impala-shell -i {impalad 实例 ip:port} -Q request_pool=root.default (fair-scheduler.xml 与 llama-site.xml 文件中配置的资源池)

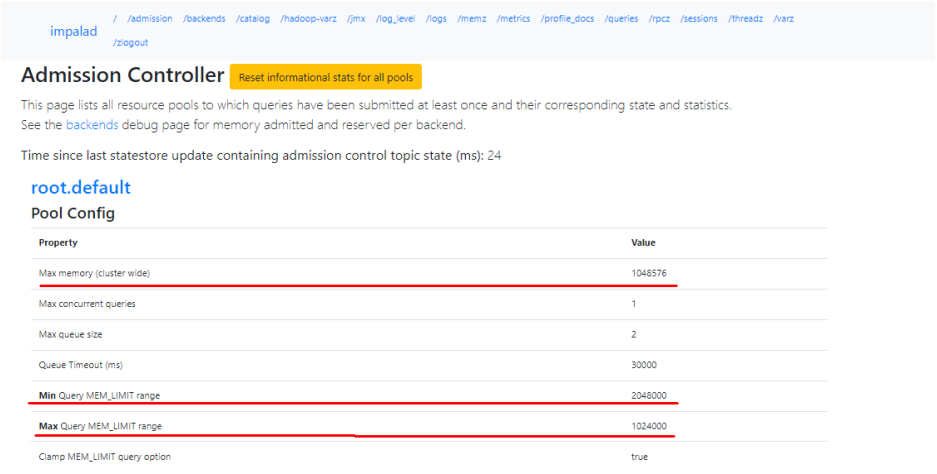
```
[root@node-master1IoKo ~]# impala-shell -i [redacted].19:21000 -Q request_pool=root.default
Starting Impala Shell without Kerberos authentication
Opened TCP connection to [redacted].19:21000
Connected to [redacted].19:21000
Server version: impalad version 3.4.0-RELEASE RELEASE (build ac0f95df4baa94fcfdc36ef370f6a432d582ac1f)
***********************************************************************************
Welcome to the Impala shell.
(Impala Shell v3.4.0-RELEASE (f68c12e) built on Sat Jun 26 17:16:01 CST 2021)

The '-B' command line flag turns off pretty-printing for query results. Use this
flag to remove formatting from results you want to save for later, or to benchmark
Impala.
***********************************************************************************
[[redacted].19:21000] default>
```

执行 SQL 查询。

```
[[redacted]19:21000] default> select * from test1;
Query: select * from test1
Query submitted at: 2022-05-12 10:01:01 (Coordinator: http://[redacted].19:25000)
Query progress can be monitored at: http://[redacted].19:25000/query_plan?query_id=97440454dbab28ea:35bd90d600000000
ERROR: Rejected query from pool root.default: Invalid pool config: the min_query_mem_limit 2048000 is greater than the max_mem_r
esources 1048576 (configured statically)
```

8. 登录到 impalad webui 上查看资源池使用情况，确认配置已生效。

https://{集群控制台地址}:9022/component/Impala/Impalad/95/



15 使用 Kafka

15.1 从零开始使用 Kafka

操作场景

用户可以在集群客户端完成 Topic 的创建、查询、删除等基本操作。

前提条件

已安装客户端，例如安装目录为“/opt/hadoopclient”，以下操作的客户端目录只是举例，请根据实际安装目录修改。

使用 Kafka 客户端（MRS 3.x 之前版本）

步骤 1 进入 ZooKeeper 实例页面：

单击集群名称，登录集群详情页面，选择“组件管理 > ZooKeeper > 实例”。

说明

若集群详情页面没有“组件管理”页签，请先完成 IAM 用户同步（在集群详情页的“概览”页签，单击“IAM 用户同步”右侧的“同步”进行 IAM 用户同步）。

步骤 2 查看 ZooKeeper 角色实例的 IP 地址。

记录 ZooKeeper 角色实例其中任意一个的 IP 地址即可。

步骤 3 登录安装客户端的节点。

步骤 4 执行以下命令，切换到客户端目录，例如“/opt/hadoopclient/Kafka/kafka/bin”。

cd /opt/hadoopclient/Kafka/kafka/bin

步骤 5 执行以下命令，配置环境变量。

source /opt/hadoopclient/bigdata_env

步骤 6 如果当前集群已启用 Kerberos 认证，执行以下命令认证当前用户。如果当前集群未启用 Kerberos 认证，则无需执行此命令。

kinit *Kafka 用户*

步骤 7 创建一个 Topic：

sh kafka-topics.sh --create --topic *主题名称* **--partitions** *主题占用的分区数* **--replication-factor** *主题的备份个数* **--zookeeper** *ZooKeeper 角色实例所在节点 IP 地址*:*clientPort*/**kafka**

例如：**sh kafka-topics.sh --create --topic TopicTest --partitions 3 --replication-factor 3 --zookeeper 10.10.10.100:2181/kafka**

步骤 8 执行以下命令，查询集群中的 Topic 信息：

sh kafka-topics.sh --list --zookeeper *ZooKeeper 角色实例所在节点 IP 地址*:*clientPort*/**kafka**

例如：**sh kafka-topics.sh --list --zookeeper 10.10.10.100:2181/kafka**

步骤 9 删除步骤 7 中创建的 Topic：

sh kafka-topics.sh --delete --topic *主题名称* **--zookeeper** *ZooKeeper 角色实例所在节点 IP 地址*:*clientPort*/**kafka**

例如：**sh kafka-topics.sh --delete --topic TopicTest --zookeeper 10.10.10.100:2181/kafka**

输入 "y"，回车。

----结束

使用 Kafka 客户端（MRS 3.x 及之后版本）

步骤 1 进入 ZooKeeper 实例页面：

登录 FusionInsight Manager，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本）。然后选择“集群 > *待操作的集群名称* > 服务 > ZooKeeper > 实例”。

步骤 2 查看 ZooKeeper 角色实例的 IP 地址。

记录 ZooKeeper 角色实例其中任意一个的 IP 地址即可。

步骤 3 登录安装客户端的节点。

步骤 4 执行以下命令，切换到客户端目录，例如“/opt/hadoopclient/Kafka/kafka/bin”。

cd /opt/hadoopclient/Kafka/kafka/bin

步骤 5 执行以下命令，配置环境变量。

source /opt/hadoopclient/bigdata_env

步骤 6 如果当前集群已启用 Kerberos 认证，执行以下命令认证当前用户。如果当前集群未启用 Kerberos 认证，则无需执行此命令。

kinit *Kafka 用户*

步骤 7 登录 FusionInsight Manager，选择“集群 > *待操作的集群名称* > 服务 > ZooKeeper > 配置 > 全部配置”，搜索参数“clientPort”,记录“clientPort”的参数值。

步骤 8 创建一个 Topic：

sh kafka-topics.sh --create --topic *主题名称* **--partitions** *主题占用的分区数* **--replication-factor** *主题的备份个数* **--zookeeper** *ZooKeeper 角色实例所在节点 IP 地址*:*clientPort*/**kafka**

例如：**sh kafka-topics.sh --create --topic TopicTest --partitions 3 --replication-factor 3 --zookeeper 10.10.10.100:2181/kafka**

步骤 9 执行以下命令，查询集群中的 Topic 信息：

sh kafka-topics.sh --list --zookeeper *ZooKeeper 角色实例所在节点 IP 地址*:*clientPort*/**kafka**

例如：**sh kafka-topics.sh --list --zookeeper 10.10.10.100:2181/kafka**

步骤 10 删除步骤 8 中创建的 Topic：

sh kafka-topics.sh --delete --topic *主题名称* **--zookeeper** *ZooKeeper 角色实例所在节点 IP 地址*:*clientPort*/**kafka**

例如：**sh kafka-topics.sh --delete --topic TopicTest --zookeeper 10.10.10.100:2181/kafka**

----结束

15.2 管理 Kafka 主题

操作场景

用户可以根据业务需要，使用集群客户端管理 Kafka 的主题。启用 Kerberos 认证的集群，需要拥有管理 Kafka 主题的权限。

前提条件

已安装客户端。

操作步骤

步骤 1 进入 ZooKeeper 实例页面：

- MRS 3.x 之前版本，单击集群名称，登录集群详情页面，选择“组件管理 > ZooKeeper > 实例”。

  说明

  若集群详情页面没有“组件管理”页签，请先完成 IAM 用户同步（在集群详情页的“概览”页签，单击“IAM 用户同步”右侧的“同步”进行 IAM 用户同步）。

- MRS 3.x 及后续版本，登录 FusionInsight Manager，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本）。然后选择“集群 > *待操作的集群名称* > 服务 > ZooKeeper > 实例”。

步骤 2 查看 ZooKeeper 角色实例的 IP 地址。

记录 ZooKeeper 角色实例其中任意一个的 IP 地址即可。

步骤 3 根据业务情况，准备好客户端，登录安装客户端的节点。

请根据客户端所在位置，参考“用户指南 > 使用 MRS 客户端 > 安装客户端”章节，登录安装客户端的节点。

步骤 4 执行以下命令，切换到客户端目录，例如“/opt/client/Kafka/kafka/bin”。

cd /opt/client/Kafka/kafka/bin

步骤 5 执行以下命令，配置环境变量。

source /opt/client/bigdata_env

步骤 6 执行以下命令，进行用户认证。（普通模式跳过此步骤）

kinit *组件业务用户*

步骤 7 MRS 3.x 之前版本：分别执行以下命令，管理 Kafka 主题。

- 创建主题

  sh kafka-topics.sh --create --topic ***主题名称*** **--partitions** ***主题占用的分区数*** **--replication-factor** ***主题的备份个数*** **--zookeeper** ***ZooKeeper 角色实例所在节点 IP 地址:clientPort*****/kafka**

- 删除主题

  sh kafka-topics.sh --delete --topic ***主题名称*** **--zookeeper** ***ZooKeeper 角色实例所在节点 IP 地址:clientPort*****/kafka**

  说明

  - 主题分区数和主题备份个数不能大于 Kafka 角色实例数量。
  - 默认情况下，ZooKeeper 的“clientPort”为“2181”。
  - ZooKeeper 角色实例所在节点 IP 地址，填写三个角色实例其中任意一个的 IP 地址即可。
  - 使用 Kafka 主题管理消息，请参见管理 Kafka 主题中的消息。

步骤 8 MRS 3.x 及后续版本：使用 **kafka-topics.sh** 管理 Kafka 主题。

- 创建主题：

  Topic 的 Partition 自动划分时，默认根据节点及磁盘上已有的 Partition 数进行均衡划分，如果期望根据磁盘容量进行 Partition 划分，那么需要修改 Kafka 服务配置“log.partition.strategy”为“capacity”。

  Kafka 创建 Topic 时，支持基于“机架感知” 和 “跨 AZ 特性” 两种选项组合生成分区及副本的分配方案且支持 “--zookeeper” 和 “--bootstrap-server”两种方式

  - 禁用机架策略 & 禁用跨 AZ 特性 (默认策略)。

    基于此策略新建的 Topic 的副本会完全随机分配到集群中任意节点上。

    ./kafka-topics.sh --create --topic *主题名称* **--partitions** *主题占用的分区数* **--replication-factor** *主题的备份数* **--zookeeper** *ZooKeeper 的任意一个节点的业务 IP:clientPort*/**kafka**

./kafka-topics.sh --create --topic *主题名称* **--partitions** *主题占用的分区数* **--replication-factor** *主题的备份数* **--bootstrap-server** *Kafka 集群 IP:21007* **--command-config ../config/client.properties**

其中，使用“--bootstrap-server”方式创建 Topic 时，需配置“rack.aware.enable=false”和“az.aware.enable=false”。

- 启用机架策略 & 禁用跨 AZ 特性。

  基于此策略新建的 Topic 的各个 Partition 的 Leader 会在集群节点上随机分配，但会确保同一 Partition 的不同 Replica 会分配在不同的机架上，所以当使用此策略时，需保证各个机架内的节点个数一致，否则会导致节点少的机架上的机器负载远高于集群平均水平。

  ./kafka-topics.sh --create --topic *主题名称* **--partitions** *主题占用的分区数* **--replication-factor** *主题的备份数* **--zookeeper** *ZooKeeper 的任意一个节点的业务 IP:clientPort*/**kafka --enable-rack-aware**

  ./kafka-topics.sh --create --topic *主题名称* **--partitions** *主题占用的分区数* **--replication-factor** *主题的备份数* **--bootstrap-server** *Kafka 集群 IP:21007* **--command-config ../config/client.properties**

  其中，使用 “--bootstrap-server”方式创建 Topic 时，需配置“rack.aware.enable=true”和“az.aware.enable=false”。

- 禁用机架策略 & 启用跨 AZ 特性。

  基于此策略新建的 Topic 的各个 Partition 的 Leader 会在集群节点上随机分配，但会确保同一 Partition 的不同 Replica 会分配在不同的 AZ 上，所以当使用此策略时，需保证各个 AZ 内的节点个数一致，否则会导致节点少的 AZ 上的机器负载远高于集群平均水平。

  ./kafka-topics.sh --create --topic *主题名称* **--partitions** *主题占用的分区数* **--replication-factor** *主题的备份数* **--zookeeper** *ZooKeeper 的任意一个节点的业务 IP:clientPort*/**kafka --enable-az-aware**

  ./kafka-topics.sh --create --topic *主题名称* **--partitions** *主题占用的分区数* **--replication-factor** *主题的备份数* **--bootstrap-server** *Kafka 集群 IP:21007* **--command-config ../config/client.properties**

  其中，使用 “--bootstrap-server”方式创建 Topic 时，需配置“rack.aware.enable=false”和“az.aware.enable=true”。

- 启用机架策略 & 启用跨 AZ 特性。

  基于此策略新建的 Topic 的各个 Partition 的 Leader 会在集群节点上随机分配，但会确保同一 Partition 的不同 Replica 会分配到不同 AZ 内的不同 RACK 上，使用此策略需保证每个 AZ 内的每个 RACK 上的节点个数一致，否则会导致集群内负载不均衡。

  ./kafka-topics.sh --create --topic *主题名称* **--partitions** *主题占用的分区数* **--replication-factor** *主题的备份数* **--zookeeper** *ZooKeeper 的任意一个节点的业务 IP:clientPort*/**kafka --enable-rack-aware --enable-az-aware**

  ./kafka-topics.sh --create --topic *主题名称* **--partitions** *主题占用的分区数* **--replication-factor** *主题的备份数* **--bootstrap-server** *Kafka 集群 IP:21007* **--command-config ../config/client.properties**

  使用 “--bootstrap-server”方式创建 Topic 时，需配置“rack.aware.enable=true”和“az.aware.enable=true”。

说明

- Kafka 创建 Topic 支持 “--zookeeper” 和 “--bootstrap-server”两种方式，区别如下：
- “--zookeeper”方式由客户端生成副本分配方案，社区从一开始就支持这种方式，为了降低对 Zookeeper 组件的依赖，社区将在后续版本中删除对这种方式的支持。基于这种方式创建 Topic 时，可以通过 "--enable-rack-aware" 和 “--enable-az-aware”这两个选项自由组合来选用副本分配策略。注意：使用“--enable-az-aware”选项的前提是服务端开启了跨 AZ 特性，即服务端启动参数“az.aware.enable”为“true”,否则会执行失败。
- “--bootstrap-server”方式由服务端生成副本分配方案，后续版本，社区将只支持这种方式来进行 Topic 管理。基于这种方式创建 Topic 时，不支持 "--enable-rack-aware" 和 "--enable-az-aware"选项来控制副本分配策略，支持 “rack.aware.enable”和“az.aware.enable”这两个服务启动参数组合来控制副本分配策略，需注意的是“az.aware.enable”参数不可修改，在创建集群时，如果开启跨 AZ 特性，会自动配置为“true”；“rack.aware.enable”参数支持用户自定义修改。

- 罗列主题：
  - **./kafka-topics.sh --list --zookeeper** *ZooKeeper 的任意一个节点的业务 IP:clientPort*/**kafka**
  - **./kafka-topics.sh --list --bootstrap-server** *Kafka 集群 IP:21007* **--command-config ../config/client.properties**
- 查看主题：
  - **./kafka-topics.sh --describe --zookeeper** *ZooKeeper 的任意一个节点的业务 IP:clientPort*/**kafka --topic** *主题名称*
  - **./kafka-topics.sh --describe --bootstrap-server** *Kafka 集群 IP:21007* **--command-config ../config/client.properties --topic** *主题名称*
- 修改主题：
  - **./kafka-topics.sh --alter --topic** *主题名称* **--config** *配置项=配置值* **--zookeeper** *ZooKeeper 的任意一个节点的业务 IP:clientPort*/**kafka**
- 扩展分区：
  - **./kafka-topics.sh --alter --topic** *主题名称* **--zookeeper** *ZooKeeper 的任意一个节点的业务 IP:clientPort*/**kafka --command-config Kafka/kafka/config/client.properties --partitions** *扩展后分区个数*
  - **./kafka-topics.sh --alter --topic** *主题名称* **--bootstrap-server** *Kafka 集群 IP:21007* **--command-config Kafka/kafka/config/client.properties --partitions** *扩展后分区个数*
- 删除主题：
  - **./kafka-topics.sh --delete --topic** *主题名称* **--zookeeper** *ZooKeeper 的任意一个节点的业务 IP:clientPort*/**kafka**
  - **./kafka-topics.sh --delete --topic** *主题名称* **--bootstrap-server** *Kafka 集群 IP:21007* **--command-config ../config/client.properties**

----结束

15.3 查看 Kafka 主题

操作场景

用户可以在 MRS 上查看 Kafka 已创建的主题信息。

操作步骤

步骤 1 进入 Kafka 服务页面：

- MRS 3.x 之前版本，单击集群名称，登录集群详情页面，选择“组件管理 > Kafka”。

  说明

  若集群详情页面没有“组件管理”页签，请先完成 IAM 用户同步（在集群详情页的“概览”页签，单击“IAM 用户同步”右侧的“同步”进行 IAM 用户同步）。

- MRS 3.x 及后续版本，登录 FusionInsight Manager，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本）。然后选择“集群 > *待操作的集群名称* > 服务 > Kafka”。

步骤 2 单击“KafkaTopic 监控”。

主题列表默认显示所有主题。可以查看主题的分区数和备份数。

步骤 3 在主题列表单击指定主题的名称，可查看详细信息。

----结束

15.4 管理 Kafka 用户权限

操作场景

在启用 Kerberos 认证的集群中，用户使用 Kafka 前需要拥有对应的权限。MRS 集群支持将 Kafka 的使用权限，授予不同用户。

Kafka 默认用户组如表 15-1 所示。

说明

在 MRS 3.x 及之后版本中，Kafka 支持两种鉴权插件：“Kafka 开源自带鉴权插件”和“Ranger 鉴权插件”。

本章节描述的是基于“Kafka 开源自带鉴权插件”的用户权限管理。若想使用 “Ranger 鉴权插件”，请参考添加 Kafka 的 Ranger 访问权限策略。

表15-1 Kafka 默认用户组

用户组名称	描述
kafkaadmin	Kafka 管理员用户组。添加入本组的用户，拥有所有主题的创

用户组名称	描述
	建，删除，授权及读写权限。
kafkasuperuser	Kafka 高级用户组。添加入本组的用户，拥有所有主题的读写权限。
kafka	Kafka 普通用户组。添加入本组的用户，需要被 kafkaadmin 组用户授予特定主题的读写权限，才能访问对应主题。

前提条件

- 已安装客户端。
- 用户已明确业务需求，并准备一个属于 kafkaadmin 组的用户，作为 Kafka 管理员用户。例如“admin”。

操作步骤

步骤 1 进入 ZooKeeper 实例页面：

- MRS 3.x 之前版本，单击集群名称，登录集群详情页面，选择“组件管理 > ZooKeeper > 实例”。

  说明

  若集群详情页面没有“组件管理”页签，请先完成 IAM 用户同步（在集群详情页的“概览”页签，单击“IAM 用户同步”右侧的“同步”进行 IAM 用户同步）。

- MRS 3.x 及后续版本，登录 FusionInsight Manager，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本）。然后选择“集群 > *待操作的集群名称* > 服务 > ZooKeeper > 实例”。

步骤 2 查看 ZooKeeper 角色实例的 IP 地址。

记录 ZooKeeper 角色实例其中任意一个的 IP 地址即可。

步骤 3 根据业务情况，准备好客户端，登录安装客户端的节点。

请根据客户端所在位置，参考“用户指南 > 使用 MRS 客户端 > 安装客户端”章节，登录安装客户端的节点。

步骤 4 执行以下命令，切换到客户端目录，例如“/opt/client/Kafka/kafka/bin”。

cd /opt/client/Kafka/kafka/bin

步骤 5 执行以下命令，配置环境变量。

source /opt/client/bigdata_env

步骤 6 执行以下命令，进行用户认证。

kinit *组件业务用户*

步骤 7 MRS 3.x 之前版本：选择业务需要对应的场景，管理 Kafka 用户权限。

- 查看某个主题的权限控制列表

  sh kafka-acls.sh --authorizer-properties zookeeper.connect=*ZooKeeper 角色实例所在节点 IP 地址*:2181/kafka --list --topic *主题名称*

- 为某个用户添加生产者的权限

  sh kafka-acls.sh --authorizer-properties zookeeper.connect=*ZooKeeper 角色实例所在节点 IP 地址*:2181/kafka --add --allow-principal User:*用户名* --producer --topic *主题名称*

- 删除某个用户的生产者权限

  sh kafka-acls.sh --authorizer-properties zookeeper.connect=*ZooKeeper 角色实例所在节点 IP 地址*:2181/kafka --remove --allow-principal User:*用户名* --producer --topic *主题名称*

- 为某个用户添加消费者的权限

  sh kafka-acls.sh --authorizer-properties zookeeper.connect=*ZooKeeper 角色实例所在节点 IP 地址*:2181/kafka --add --allow-principal User:*用户名* --consumer --topic *主题名称* --group *消费者组名称*

- 删除某个用户的消费者权限

  sh kafka-acls.sh --authorizer-properties zookeeper.connect=*ZooKeeper 角色实例所在节点 IP 地址*:2181/kafka --remove --allow-principal User:*用户名* --consumer --topic *主题名称* --group *消费者组名称*

说明

删除权限时需要输入两次“y”确认删除权限。

步骤 8 MRS 3.x 及后续版本：使用“kafka-acl.sh”进行用户授权常用命令如下。

- 查看某 Topic 权限控制列表：

  ./kafka-acls.sh --authorizer-properties zookeeper.connect=<*ZooKeeper 的任意一个节点的业务 IP:2181/kafka* > **--list --topic <*Topic 名称*>**

  ./kafka-acls.sh --bootstrap-server <*Kafka 集群 IP:21007*> **--command-config ../config/client.properties --list --topic** <*Topic 名称*>

- 添加给某用户 Producer 权限：

  ./kafka-acls.sh --authorizer-properties zookeeper.connect=<*ZooKeeper 的任意一个节点的业务 IP:2181/kafka* > **--add --allow-principal User:**<*用户名*> **--producer --topic** <*Topic 名称*>

  ./kafka-acls.sh --bootstrap-server <*Kafka 集群 IP:21007*> **--command-config ../config/client.properties --add --allow-principal User:**<*用户名*> **--producer --topic** <*Topic 名称*>

- 给某用户批量添加 Producer 权限

  ./kafka-acls.sh --authorizer-properties zookeeper.connect=<*ZooKeeper 的任意一个节点的业务 IP:2181/kafka* > **--add --allow-principal User:**<*用户名*> **--producer --topic** <*Topic 名称*> **--resource-pattern-type prefixed**

  ./kafka-acls.sh --bootstrap-server <*Kafka 集群 IP:21007*> **--command-config ../config/client.properties --add --allow-principal User:**<*用户名*> **--producer --topic** <*Topic 名称*>**--resource-pattern-type prefixed**

- 删除某用户 Producer 权限：

  ./kafka-acls.sh --authorizer-properties zookeeper.connect=<*ZooKeeper 的任意一个节点的业务 IP:2181/kafka* > **--remove --allow-principal User:**<*用户名*> **--producer --topic** <*Topic 名称*>

  ./kafka-acls.sh --bootstrap-server <*Kafka 集群 IP:21007*> **--command-config ../config/client.properties --remove --allow-principal User:**<*用户名*> **--producer --topic** <*Topic 名称*>

- 批量删除某用户 Producer 权限:

  ./kafka-acls.sh --authorizer-properties zookeeper.connect=<*ZooKeeper 的任意一个节点的业务 IP:2181/kafka* > **--remove --allow-principal User:**<*用户名*> **--producer --topic** <*Topic 名称*> **--resource-pattern-type prefixed**

  ./kafka-acls.sh --bootstrap-server <*Kafka 集群 IP:21007*> **--command-config ../config/client.properties --remove --allow-principal User:**<*用户名*> **--producer --topic** <*Topic 名称*>**--resource-pattern-type prefixed**

- 添加给某用户 Consumer 权限：

  ./kafka-acls.sh --authorizer-properties zookeeper.connect=<*ZooKeeper 的任意一个节点的业务 IP:2181/kafka* > **--add --allow-principal User:**<*用户名*> **--consumer --topic** <*Topic 名称*> **--group** <*消费者组名称*>

  ./kafka-acls.sh --bootstrap-server <*Kafka 集群 IP:21007*> **--command-config ../config/client.properties --add --allow-principal User:**<*用户名*> **--consumer --topic** <*Topic 名称*> **--group** <*消费者组名称*>

- 给某用户批量添加 Consumer 权限

  ./kafka-acls.sh --authorizer-properties zookeeper.connect=<*ZooKeeper 的任意一个节点的业务 IP:2181/kafka* > **--add --allow-principal User:**<*用户名*> **--consumer --topic** <*Topic 名称*> **--group** <*消费者组名称*> **--resource-pattern-type prefixed**

  ./kafka-acls.sh --bootstrap-server <*Kafka 集群 IP:21007*> **--command-config ../config/client.properties --add --allow-principal User:**<*用户名*> **--consumer --topic** <*Topic 名称*> **--group** <*消费者组名称*> **--resource-pattern-type prefixed**

- 删除某用户 Consumer 权限：

  ./kafka-acls.sh --authorizer-properties zookeeper.connect=<*ZooKeeper 的任意一个节点的业务 IP:2181/kafka* > **--remove --allow-principal User:**<*用户名*> **--consumer --topic** <*Topic 名称*> **--group** <*消费者组名称*>

  ./kafka-acls.sh --bootstrap-server <*Kafka 集群 IP:21007*> **--command-config ../config/client.properties --remove --allow-principal User:**<*用户名*> **--*consumer --topic*** <*Topic 名称*> **--group** <*消费者组名称*>

- 批量删除某用户 Consumer 权限:

  ./kafka-acls.sh --authorizer-properties zookeeper.connect=<*ZooKeeper 的任意一个节点的业务 IP:2181/kafka* > **--remove --allow-principal User:**<*用户名*> **--consumer --topic** <*Topic 名称*> **--group** <*消费者组名称*> **--resource-pattern-type prefixed**

  ./kafka-acls.sh --bootstrap-server <*Kafka 集群 IP:21007*> **--command-config ../config/client.properties --remove --allow-principal User:**<*用户名*> **--consumer --topic** <*Topic 名称*> **--group** <*消费者组名称*> **--resource-pattern-type prefixed**

----结束

15.5 管理 Kafka 主题中的消息

操作场景

用户可以根据业务需要，使用 MRS 集群客户端，在 Kafka 主题中产生消息，或消费消息。启用 Kerberos 认证的集群，需要用户拥有在 Kafka 主题中执行相应操作的权限。

前提条件

已安装客户端。

操作步骤

步骤 1 进入 Kafka 服务页面：

- MRS 3.x 之前版本，单击集群名称，登录集群详情页面，选择“组件管理 > Kafka”。

  说明

  若集群详情页面没有“组件管理”页签，请先完成 IAM 用户同步（在集群详情页的“概览”页签，单击“IAM 用户同步”右侧的“同步”进行 IAM 用户同步）。

- MRS 3.x 及后续版本，登录 FusionInsight Manager，然后选择“集群 > *待操作的集群名*称 > 服务 > Kafka”。

步骤 2 单击“实例”，查看 Kafka 角色实例的 IP 地址。

记录 Kafka 角色实例其中任意一个的 IP 地址即可。

步骤 3 根据业务情况，准备好客户端，登录安装客户端的节点。

请根据客户端所在位置，参考“用户指南 > 使用 MRS 客户端 > 安装客户端”章节，登录安装客户端的节点。

步骤 4 执行以下命令，切换到客户端目录，例如“/opt/client/Kafka/kafka/bin”。

cd /opt/client/Kafka/kafka/bin

步骤 5 执行以下命令，配置环境变量。

source /opt/client/bigdata_env

步骤 6 启用 Kerberos 认证的集群，执行以下命令认证用户身份。未启用 Kerberos 认证的集群无需执行。

kinit *Kafka 用户*

例如：

kinit admin

步骤 7 根据业务需要，管理 Kafka 主题中的消息。

- 在主题中产生消息

  sh kafka-console-producer.sh --broker-list *Kafka 角色实例所在节点的 IP 地址*:9092 --topic *主题名称* --producer.config /opt/client/Kafka/kafka/config/producer.properties

  用户可以输入指定的内容作为生产者产生的消息，输入完成后按回车发送消息。如果需要结束产生消息，使用“Ctrl + C”退出任务。

- 消费主题中的消息

  sh kafka-console-consumer.sh --topic *主题名称* --bootstrap-server *Kafka 角色实例所在节点的 IP 地址*:9092 --consumer.config /opt/client/Kafka/kafka/config/consumer.properties

  配置文件中“group.id”指定的消费者组默认为“example-group1”。用户可根据业务需要，自定义其他消费者组。每次消费时生效。

  执行命令时默认会读取当前消费者组中未被处理的消息。如果在配置文件指定了新的消费者组且命令中增加参数“--from-beginning”，则会读取所有 Kafka 中未被自动删除的消息。

说明

- Kafka 角色实例所在节点 IP 地址，填写 Broker 角色实例其中任意一个的 IP 地址即可。
- 如果集群启用 Kerberos 认证，则端口需要修改为“21007”。
- 默认情况下，ZooKeeper 的“clientPort”为“2181”。

----结束

15.6 基于 binlog 的 MySQL 数据同步到 MRS 集群中

本章节为您介绍使用 Maxwell 同步工具将线下基于 binlog 的数据迁移到 MRS Kafka 集群中的指导。

Maxwell 是一个开源程序（https://maxwells-daemon.io），通过读取 MySQL 的 binlog 日志，将增删改等操作转为 JSON 格式发送到输出端(如控制台/文件/Kafka 等)。Maxwell 可部署在 MySQL 机器上，也可独立部署在其他与 MySQL 网络可通的机器上。

Maxwell 运行在 Linux 服务器上，常见的有 EulerOS、Ubuntu、Debian、CentOS、OpenSUSE 等，且需要 Java 1.8+支持。

同步数据具体内容如下。

1. 配置 MySQL
2. 安装 Maxwell
3. 配置 Maxwell
4. 启动 Maxwell
5. 验证 Maxwell
6. 停止 Maxwell
7. Maxwell 生成的数据格式及常见字段含义

配置 MySQL

步骤 1 开启 binlog，在 MySQL 中打开 my.cnf 文件，在[mysqld] 区块检查是否配置 server_id，log-bin 与 binlog_format，若没有配置请执行如下命令添加配置项并重启 MySQL，若已经配置则忽略此步骤。

```
$ vi my.cnf

[mysqld]
server_id=1
log-bin=master
binlog_format=row
```

步骤 2 Maxwell 需要连接 MySQL，并创建一个名称为 maxwell 的数据库存储元数据，且需要能访问需要同步的数据库，所以建议新创建一个 MySQL 用户专门用来给 Maxwell 使用。使用 root 登录 MySQL 之后，执行如下命令创建 maxwell 用户（其中 XXXXXX 是密码，请修改为实际值）。

- 若 Maxwell 程序部署在非 MySQL 机器上，则创建的 maxwell 用户需要有远程登录数据库的权限，此时创建命令为

  mysql> GRANT ALL on maxwell.* to 'maxwell'@'%' identified by 'XXXXXX';

  mysql> GRANT SELECT, REPLICATION CLIENT, REPLICATION SLAVE on *.* to 'maxwell'@'%';
- 若 Maxwell 部署在 MySQL 机器上，则创建的 maxwell 用户可以设置为只能在本机登录数据库，此时创建命令为

  mysql> GRANT SELECT, REPLICATION CLIENT, REPLICATION SLAVE on *.* to 'maxwell'@'localhost' identified by 'XXXXXX';

  mysql> GRANT ALL on maxwell.* to 'maxwell'@'localhost';

----结束

安装 Maxwell

步骤 1 下载安装包，下载路径为 https://github.com/zendesk/maxwell/releases，选择名为 maxwell-XXX.tar.gz 的二进制文件下载，其中 XXX 为版本号。

步骤 2 将 tar.gz 包上传到任意目录下（本示例路径为 Master 节点的/opt）。

步骤 3 登录部署 Maxwell 的服务器，并执行如下命令进入 tar.gz 包所在目录。

cd /opt

步骤 4 执行如下命令解压“maxwell-XXX.tar.gz”压缩包，并进入“maxwell-XXX”文件夹。

tar -zxvf maxwell-XXX.tar.gz

cd maxwell-XXX

----结束

配置 Maxwell

在 maxwell-XXX 文件夹下若有 conf 目录则配置 config.properties 文件，配置项说明请参见表 15-2。若没有 conf 目录，则是在 maxwell-XXX 文件夹下将 config.properties.example 修改成 config.properties。

表15-2 Maxwell 配置项说明

配置项	是否必填	说明	默认值
user	是	连接 MySQL 的用户名，即步骤 2 中新创建的用户	-
password	是	连接 MySQL 的密码	-
host	否	MySQL 地址	localhost
port	否	MySQL 端口	3306
log_level	否	日志打印级别，可选值为 • debug • info • warn • error	info
output_ddl	否	是否发送 DDL(数据库与数据表的定义修改)事件 • true：发送 DDL 事件 • false：不发送 DDL 事件	false
producer	是	生产者类型，配置为 kafka • stdout：将生成的事件打印在日志中 • kafka：将生成的事件发送到 kafka	stdout
producer_partition_by	否	分区策略，用来确保相同一类的数据写入到 kafka 同一分区 • database：使用数据库名称做分区，保证同一个数据库的事件写入到 kafka 同一个分区中 • table：使用表名称做分区，保证同一个表的事件写入到 kafka 同一个分区中	databa
ignore_producer_error	否	是否忽略生产者发送数据失败的错误 • true：在日志中打印错误信息并跳过错误的数据，程序继续运行 • false：在日志中打印错误信息并终止程序	true
metrics_slf4j_interval	否	在日志中输出上传 kafka 成功与失败数据的数量统计的时间间隔，单位为秒	60

配置项	是否必填	说明	默认值
kafka.bootstrap.servers	是	kafka 代理节点地址，配置形式为 HOST:PORT[,HOST:PORT]	-
kafka_topic	否	写入 kafka 的 topic 名称	maxwell
dead_letter_topic	否	当发送某条记录出错时，记录该条出错记录主键的 kafka topic	-
kafka_version	否	Maxwell 使用的 kafka producer 版本号，不能在 config.properties 中配置，需要在启动命令时用--kafka_version xxx 参数传入	-
kafka_partition_hash	否	划分 kafka topic partition 的算法，支持 default 或 murmur3	default
kafka_key_format	否	Kafka record 的 key 生成方式，支持 array 或 Hash	Hash
ddl_kafka_topic	否	当 output_ddl 配置为 true 时，DDL 操作写入的 topic	{kafka_topic}
filter	否	过滤数据库或表。 • 若只想采集 mydatabase 的库，可以配置为 exclude: *.*,include: mydatabase.* • 若只想采集 mydatabase.mytable 的表，可以配置为 exclude: *.*,include: mydatabase.mytable • 若只想采集 mydatabase 库下的 mytable，mydate_123, mydate_456 表，可以配置为 exclude: *.*,include: mydatabase.mytable, include: mydatabase./mydate_\\d*/	-

启动 Maxwell

步骤 1 登录 Maxwell 所在的服务器。

步骤 2 执行如下命令进入 Maxwell 安装目录。

cd /opt/maxwell-1.21.0/

说明

如果是初次使用 Maxwell，建议将 conf/config.properties 中的 log_level 改为 debug(调试级别)，以便观察启动之后是否能正常从 MySQL 获取数据并发送到 kafka，当整个流程调试通过之后，再把 log_level 修改为 info，然后先停止再启动 Maxwell 生效。

```
# log level [debug | info | warn | error]
log_level=debug
```

步骤 3 执行如下命令启动 Maxwell。

source /opt/client/bigdata_env

bin/Maxwell

**bin/maxwell --user='maxwell' --password='XXXXXX' --host='127.0.0.1' **

--producer=kafka --kafka.bootstrap.servers=kafkahost:9092 --kafka_topic=Maxwell

其中，user，password 和 host 分别表示 MySQL 的用户名，密码和 IP 地址，这三个参数可以通过修改配置项配置也可以通过上述命令配置，kafkahost 为流式集群的 Core 节点的 IP 地址。

显示类似如下信息，表示 Maxwell 启动成功。

```
Success to start Maxwell [78092].
```

----结束

验证 Maxwell

步骤 1 登录 Maxwell 所在的服务器。

步骤 2 查看日志。如果日志里面没有 ERROR 日志，且有打印如下日志，表示与 MySQL 连接正常。

```
BinlogConnectorLifecycleListener - Binlog connected.
```

步骤 3 登录 MySQL 数据库，对测试数据进行更新/创建/删除等操作。操作语句可以参考如下示例。

```
-- 创建库
create database test;
-- 创建表
create table test.e (
  id int(10) not null primary key auto_increment,
  m double,
  c timestamp(6),
  comment varchar(255) charset 'latin1'
);
-- 增加记录
insert into test.e set m = 4.2341, c = now(3), comment = 'I am a creature of
light.';
-- 更新记录
update test.e set m = 5.444, c = now(3) where id = 1;
-- 删除记录
delete from test.e where id = 1;
-- 修改表
alter table test.e add column torvalds bigint unsigned after m;
-- 删除表
drop table test.e;
-- 删除库
drop database test;
```

步骤 4 观察 Maxwell 的日志输出，如果没有 WARN/ERROR 打印，则表示 Maxwell 安装配置正常。

若要确定数据是否成功上传，可设置 config.properties 中的 log_level 为 debug，则数据上传成功时会立刻打印如下 JSON 格式数据，具体字段含义请参考 Maxwell 生成的数据格式及常见字段含义。

```
{"database":"test","table":"e","type":"insert","ts":1541150929,"xid":60556,"commit"
:true,"data":{"id":1,"m":4.2341,"c":"2018-11-02 09:28:49.297000","comment":"I am a
creature of light."}}
……
```

说明

当整个流程调试通过之后，可以把 config.properties 文件中的配置项 log_level 修改为 info，减少日志打印量，并重启 Maxwell。

```
# log level [debug | info | warn | error]
log_level=info
```

----结束

停止 Maxwell

步骤 1 登录 Maxwell 所在的服务器。

步骤 2 执行如下命令，获取 Maxwell 的进程标识（PID）。输出的第二个字段即为 PID。

ps -ef | grep Maxwell | grep -v grep

步骤 3 执行如下命令，强制停止 Maxwell 进程。

kill -9 PID

----结束

Maxwell 生成的数据格式及常见字段含义

Maxwell 生成的数据格式为 JSON，常见字段含义如下：

- type：操作类型，包含 database-create，database-drop，table-create，table-drop，table-alter，insert，update，delete
- database：操作的数据库名称
- ts：操作时间，13 位时间戳
- table：操作的表名
- data：数据增加/删除/修改之后的内容
- old：数据修改前的内容或者表修改前的结构定义
- sql：DDL 操作的 SQL 语句
- def：表创建与表修改的结构定义
- xid：事物唯一 ID
- commit：数据增加/删除/修改操作是否已提交

15.7 创建 Kafka 角色

操作场景

该任务指导系统管理员创建并设置 Kafka 的角色。

本章节内容适用于 MRS 3.x 及后续版本。

说明

安全模式支持创建 Kafka 角色，普通模式不支持创建 Kafka 角色。

如果当前组件使用了 Ranger 进行权限控制，须基于 Ranger 配置相关策略进行权限管理，具体操作可参考添加 Kafka 的 Ranger 访问权限策略。

前提条件

系统管理员已明确业务需求。

操作步骤

步骤 1 登录 FusionInsight Manager，选择“系统 > 权限 > 角色”。

步骤 2 单击“添加角色”，然后在“角色名称”和“描述”输入角色名字与描述。

步骤 3 在“配置资源权限”中，选择“*待操作集群的名称* > Kafka”。

步骤 4 根据业务需求选择权限，具体配置项，请参见表 15-3

表15-3 配置项说明

任务场景	角色授权操作
设置 Kafka 管理员权限	在“配置资源权限”的表格中选择“*待操作集群的名称* > Kafka > Kafka Manager 权限”。 说明 设置此权限，拥有 Topic 的创建、删除等权限，但是不具备任何 Topic 的生产和消费权限。
设置用户对 Topic 的生产权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > Kafka > Kafka Topic 生产和消费权限”。 2. 在指定 Topic 的“权限”列，勾选“Kafka 生产者权限”。
设置用户对 Topic 的消费权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > Kafka > Kafka Topic 生产和消费权限”。 2. 在指定 Topic 的“权限”列，勾选“Kafka 消费者权限”。

步骤 5 单击“确定”完成，返回“角色”。

----结束

15.8 Kafka 常用参数

本章节内容适用于 MRS 3.x 及后续版本。

参数入口

参数入口，请参考修改集群服务配置参数。

常用参数

表15-4 参数说明

配置参数	说明	缺省值
log.dirs	Kafka 数据存储目录列表，以逗号分隔多个目录。	%{@auto.detect.datapart.bk.log.logs}
KAFKA_HEAP_OPTS	Kafka 启动 Broker 时使用的 jvm 选项。建议根据业务需要进行设置。	-Xmx6G -Xms6G
auto.create.topics.enable	是否自动创建 Topic，若参数设置为 false，发消息前需要通过命令创建 Topic。	true
default.replication.factor	自动创建 Topic 时的默认副本数。	2
monitor.preInitDelay	服务启动后，第一次健康检查的延迟时间。如果启动需要较长时间，可以通过调大参数，来完成启动。单位为毫秒。	600000

超时参数

表15-5 Broker 相关超时参数

参数名称	参数说明	默认值	影响分析

参数名称	参数说明	默认值	影响分析
controller.socket.timeout.ms	Controller 连接 Broker 的超时时间。单位：毫秒。	30000	Controller 连接 Broker 的超时时间，一般不需要调整。
group.max.session.timeout.ms	Consumer 注册时允许的最大会话超时时间。单位：毫秒。	1800000	允许 Consumer 配置的 session.timeout.ms 的最大值（不包含此值）。
group.min.session.timeout.ms	Consumer 注册时允许的最小会话超时时间。单位：毫秒。	6000	允许 Consumer 配置的 session.timeout.ms 的最小值（不包含此值）。
offsets.commit.timeout.ms	Offset 提交请求的超时时间。单位：毫秒。	5000	Offset 提交时被延迟处理的最大超时时间。
replica.socket.timeout.ms	副本数据同步请求的超时时间，配置值不得小于 replica.fetch.wait.max.ms。单位：毫秒。	30000	同步线程在发送同步请求之前等待通道建立的最大超时时间，要求配置大于 replica.fetch.wait.max.ms。
request.timeout.ms	设置客户端发送连接请求后，等待响应的超时时间。如果在超时时间内没有接收到响应，那么客户端重新发送，并在达到重试次数后返回请求失败。单位：毫秒。	30000	Broker 节点上的 Controller、Replica 线程中传入 networkclient 连接的超时参数。
transaction.max.timeout.ms	事务允许的最大超时。如果客户端的请求时间超过该值，则 Broker 将在 InitProducerIdRequest 中返回一个错误。这样可以防止客户端超时时间过长，而导致消费者无法接收 topic。单位：毫秒。	900000	事务最大超时时间。
user.group.cache.timeout.sec	指定缓存中保存用户对应组信息的时间。单位：秒。	300	缓存中用户和组对应关系缓存时间，超过此时间用户信息才会再次通过 id -Gn 命令查询，在此期间，仅使用缓存中的用户和组对应关系。
zookeeper.connection.timeout.ms	连接 ZooKeeper 的超时时间。单位：毫秒。	45000	ZooKeeper 连接超时时间，这个时间决定了 zkclient 中初次连接建立过程时允许消耗的时

参数名称	参数说明	默认值	影响分析
			间，超过该时间，zkclient 会主动断开。
zookeeper.session.timeout.ms	ZooKeeper 会话超时时间。如果 Broker 在此时间内未向 ZooKeeper 上报心跳，则被认为失效。单位：毫秒。	45000	ZooKeeper 会话超时时间。 作用一：这个时间结合传入的 ZKURL 中 ZooKeeper 的地址个数，ZooKeeper 客户端以（sessionTimeout/传入 ZooKeeper 地址个数）为连接一个节点的超时时间，超过此时间未连接成功，则尝试连接下一个节点。 作用二：连接建立后，一个会话的超时时间，如 ZooKeeper 上注册的临时节点 BrokerId，当 Broker 被停止，则该 BrokerId，会经过一个 sessionTimeout 才会被 ZooKeeper 清理。

表15-6 Producer 相关超时参数

配置名称	说明	默认值	影响分析
request.timeout.ms	指定发送消息请求的请求超时时间。	30000	请求超时时间，出现网络问题时，需调大此参数；配置过小，则容易出现 Batch Expire 异常。

表15-7 Consumer 相关超时参数

配置名称	说明	默认值	影响分析
connections.max.idle.ms	空闲连接的保留时间。	600000	空闲连接的保留时间，连接空闲时间大于此时

配置名称	说明	默认值	影响分析
			间，则会销毁该连接，有需要时重新创建连接。
request.timeout.ms	消费请求的超时时间。	30000	请求超时时间，请求超时会失败然后不断重试。

15.9 Kafka 安全使用说明

本章节内容适用于 MRS 3.x 及后续版本。

Kafka API 简单说明

- Producer API

  指 **org.apache.kafka.clients.producer.KafkaProducer** 中定义的接口，在使用“kafka-console-producer.sh”时，默认使用此 API。

- Consumer API

  指 **org.apache.kafka.clients.consumer.KafkaConsumer** 中定义的接口，在使用“kafka-console-consumer.sh”时，默认会调用此 API。

说明

MRS 3.x 后，Kafka 不支持旧 Producer API 和旧 Consumer API。

Kafka 访问协议说明

Kafka 当前支持四种协议类型的访问：PLAINTEXT、SSL、SASL_PLAINTEXT、SASL_SSL。

Kafka 服务启动时，默认会启动 PLAINTEXT 和 SASL_PLAINTEXT 两种协议类型的访问监听。可通过设置 Kafka 服务配置“ssl.mode.enable”为“true”，来启动 SSL 和 SASL_SSL 两种协议类型的访问监听。下表是四中协议类型的简单说明：

协议类型	说明	默认端口
PLAINTEXT	支持无认证的明文访问	9092
SASL_PLAINTEXT	支持 Kerberos 认证的明文访问	21007
SSL	支持无认证的 SSL 加密访问	9093
SASL_SSL	支持 Kerberos 认证的 SSL 加密访问	21009

Topic 的 ACL 设置

Topic 的权限信息，需要在 Linux 客户端上，使用“kafka-acls.sh”脚本进行查看和设置，具体可参考管理 Kafka 用户权限。

针对不同的 Topic 访问场景，Kafka 中 API 使用说明

- 场景一：访问设置了 ACL 的 Topic

使用的 API	用户属组	客户端参数	服务端参数	访问的端口
API	用户需满足以下条件之一即可： • 属于系统管理员组 • 属于 kafkaadmin 组 • 属于 kafkasuperuser 组 • 被授权的 kafka 组的用户	security.inter.broker.protocol=SASL_PLAINTEXT sasl.kerberos.service.name = kafka	-	sasl.port（默认 21007）
		security.protocol=SASL_SSL sasl.kerberos.service.name = kafka	“ssl.mode.enable”配置为 true	sasl-ssl.port（默认 21009）

- 场景二：访问未设置 ACL 的 Topic

使用的 API	用户属组	客户端参数	服务端参数	访问的端口
API	用户需满足以下条件之一： • 属于系统管理员组 • 属于 kafkaadmin 组 • 属于 kafkasuperuser 组	security.protocol=SASL_PLAINTEXT sasl.kerberos.service.name = kafka	-	sasl.port（默认 21007）
	用户属于 kafka 组		“allow.everyone.if.no.acl.found”配置为 true 说明 普通集群下不涉及服务端参数“allow.everyone.if	sasl.port（默认 21007）

使用的API	用户属组	客户端参数	服务端参数	访问的端口
			.no.acl.found”的修改	
	用户需满足以下条件之一： • 属于系统管理员组 • 属于 kafkaadmin 组 • kafkasuperuser 组用户	security.protocol=SASL_SSL sasl.kerberos.service.name = kafka	“ssl.mode.enable”配置为“true”	sasl-ssl.port（默认21009）
	用户属于 kafka 组		1. “allow.everyone.if.no.acl.found”配置为“true” 2. “ssl.mode.enable”配置为“true”	sasl-ssl.port（默认21009）
	-	security.protocol=PLAINTEXT	“allow.everyone.if.no.acl.found”配置为“true”	port（默认9092）
	-	security.protocol=SSL	1. “allow.everyone.if.no.acl.found”配置为“true” 2. “ssl.mode.enable”配置为“true”	ssl.port（默认9063）

15.10 Kafka 业务规格说明

本章节内容适用于 MRS 3.x 及后续版本。

支持的 Topic 上限

支持 Topic 的个数，受限于进程整体打开的文件句柄数（现场环境一般主要是数据文件和索引文件占用比较多）。

1. 可通过 **ulimit -n** 命令查看进程最多打开的文件句柄数；
2. 执行 **lsof -p** <*Kafka PID*>命令，查看当前单节点上 Kafka 进程打开的文件句柄（会继续增加）；

3. 权衡当前需要创建的 Topic 创建完成后，会不会达到文件句柄上限，每个 Partition 文件夹下会最多保存多大的数据，会产生多少个数据文件（*.log 文件，默认配置为 1GB，可通过修改 **log.segment.bytes** 来调整大小）和索引文件（*.index 文件，默认配置为 10MB，可通过修改 **log.index.size.max.bytes** 来调整大小），是否会影响 Kafka 正常运行。

Consumer 的并发量

在一个应用中，同一个 Group 的 Consumer 并发量建议与 Topic 的 Partition 个数保持一致，保证每个 Consumer 对应消费一个 Partition 上的数据。若 Consumer 的并发量多于 Partition 个数，那么多余的 Consumer 将消费不到数据。

Topic 和 Partition 的划分关系说明

- 假设集群中部署了 K 个 Kafka 节点，每个节点上配置的磁盘个数为 N，每块磁盘大小为 M，集群中共有 n 个 Topic（T1,T2…Tn），并且其中第 m 个 Topic 的每秒输入数据总流量为 X(Tm) MB/s，配置的副本数为 R(Tm)，配置数据保存时间为 Y(Tm)小时，那么整体必须满足：

$$M \times N \times K > \sum_{i=T1}^{Tn} (X(i)R(i)Y(i) \times 3600)$$

- 假设单个磁盘大小为 M，该磁盘上有 n 个 Partition（P0,P1……Pn），并且其中第 m 个 Partition 的每秒写入数据流量为 Q(Pm) MB/s（计算方法：所属 Topic 的数据流量除以 Partition 数） 、数据保存时间为 T(Pm)小时，那么单个磁盘必须满足：

$$M > \sum_{i=P0}^{Pn} (Q(i)T(i) \times 3600)$$

- 根据吞吐量粗略计算，假设生产者可以达到的吞吐量为 P，消费者可以达到的吞吐量为 C，预期 Kafka 吞吐量为 T，那么建议该 Topic 的 Partition 数目设置为 Max(T/P , T/C)。

说明

- 在 Kafka 集群中，分区越多吞吐量越高，但是分区过多也存在潜在影响，例如文件句柄增加、不可用性增加（如：某个节点故障后，部分 Partition 重选 Leader 后时间窗口会比较大）及端到端时延增加等。
- 建议：单个 Partition 的磁盘占用最大不超过 100GB；单节点上 Partition 数目不超过 3000；整个集群的分区总数不超过 10000。

15.11 使用 Kafka 客户端

操作场景

该任务指导用户在运维场景或业务场景中使用 Kafka 客户端。

本章节适用于 MRS 3.x 及后续版本。

前提条件

- 已安装客户端。例如安装目录为“/opt/client”。
- 各组件业务用户由系统管理员根据业务需要创建。“机机”用户需要下载 keytab 文件。“人机”用户第一次登录时需修改密码。（普通模式不涉及）
- 在修改集群域名后，需要重新下载客户端，以保证客户端配置文件中 kerberos.domain.name 配置为正确的服务端域名。

操作步骤

步骤 1 以客户端安装用户，登录安装客户端的节点。

步骤 2 执行以下命令，切换到客户端安装目录。

cd /opt/client

步骤 3 执行以下命令配置环境变量。

source bigdata_env

步骤 4 执行以下命令，进行用户认证。（普通模式跳过此步骤）

kinit *组件业务用户*

步骤 5 执行以下命令切换到 Kafka 客户端安装目录。

cd Kafka/kafka/bin

步骤 6 执行以下命令使用客户端工具查看帮助并使用。

- **./kafka-console-consumer.sh**：Kafka 消息读取工具
- **./kafka-console-producer.sh**：Kafka 消息发布工具
- **./kafka-topics.sh**：Kafka Topic 管理工具

----结束

15.12 配置 Kafka 高可用和高可靠参数

操作场景

Kafka 消息传输保障机制，可以通过配置不同的参数来保障消息传输，进而满足不同的性能和可靠性要求。本章节介绍如何配置 Kafka 高可用和高可靠参数。

本章节内容适用于 MRS 3.x 及后续版本。

对系统的影响

- 配置高可用、高性能的影响：

须知

配置高可用、高性能模式后，数据可靠性会降低。在磁盘故障、节点故障等场景下存在数据丢失风险。

- 配置高可靠性的影响：
  - 性能降低：

    在生产数据时，配置了高可靠参数 ack=-1 之后，需要多个副本均写入成功之后才认为是写入成功。这样会导致单条消息时延增加，客户端处理能力下降。具体性能以现场实际测试数据为准。
  - 可用性降低：

    不允许不在 ISR 中的副本被选举为 Leader。如果 Leader 下线时，其他副本均不在 ISR 列表中，那么该分区将保持不可用，直到 Leader 节点恢复。当分区的一个副本所在节点故障时，无法满足最小写入成功的副本数，那么将会导致业务写入失败。
- 参数配置项为服务级配置需要重启 Kafka，建议在变更窗口做服务级配置修改。

参数描述

- 如果业务需要保证高可用和高性能。

  在服务端配置如表 15-8 中参数，参数配置入口请参考修改集群服务配置参数。

表15-8 服务端高可用性和高性能参数说明

参数	默认值	说明
unclean.leader.election.enable	true	是否允许不在 ISR 中的副本被选举为 Leader，若设置为 true，可能会造成数据丢失。
auto.leader.rebalance.enable	true	是否使用 Leader 自动均衡功能。 如果设为 true，Controller 会周期性的为所有节点的每个分区均衡 Leader，将 Leader 分配给更优先的副本。
min.insync.replicas	1	当 Producer 设置 acks 为-1 时，指定需要写入成功的副本的最小数目。

在客户端配置文件 producer.properties 中配置如表 15-9 中参数，producer.properties 存放路径为：/opt/client/Kafka/kafka/config/producer.properties，其中/opt/client 为 Kafka 客户端安装目录。

表15-9 客户端高可用性和高性能参数说明

参数	默认值	说明
acks	1	需要 Leader 确认消息是否

参数	默认值	说明
		已经接收并认为已经处理完成。该参数会影响消息的可靠性和性能。 • acks=0 ： Producer 将不会等待服务端任何响应。消息将会被认为成功。 • acks=1 ： 当副本所在 Leader 确认数据已写入，但是其不会等待所有的副本完全写入即返回响应。在这种情况下，如果 Leader 确认后但是副本未同步完成时 Leader 异常，那么数据就会丢失。 • acks=-1 ： 意味着等待所有的同步副本确认后才认为成功，配合“min.insync.replicas”可以确保多副本写入成功，只要有一个副本保持活跃状态，记录将不会丢失。

- 如果业务需要保证数据高可靠性。

  在服务端配置如表 15-10 参数，参数配置入口请参考修改集群服务配置参数。

表15-10 服务端高可靠性参数说明

参数	建议值	说明
unclean.leader.election.enable	false	不允许不在 ISR 中的副本被选举为 Leader。
min.insync.replicas	2	当 Producer 设置 acks 为-1 时，指定需要写入成功的副本的最小数目。 需要满足 min.insync.replicas <= replication.factor。

在客户端配置文件 producer.properties 中配置如表 15-11 中参数，producer.properties 存放路径为：/opt/client/Kafka/kafka/config/producer.properties，其中/opt/client 为 Kafka 客户端安装目录。

表15-11 客户端高可靠性参数说明

参数	建议值	说明
acks	-1	Producer 需要 Leader 确认消息是否已经接收并认为已经处理完成。 acks=-1 需要等待在 ISR 列表的副本都确认接收到消息并处理完成才表示消息成功。配合“min.insync.replicas”可以确保多副本写入成功，只要有一个副本保持活跃状态，记录将不会丢失，此参数配置为-1 时，会降低生产性能，请权衡后配置。

配置建议

请根据以下业务场景对可靠性和性能要求进行评估，采用合理参数配置。

- 对于价值数据，这两种场景下建议 Kafka 数据目录磁盘配置 raid1 或者 raid5，从而提高单个磁盘故障情况下数据可靠性。
- 参数配置项均为 Topic 级别可修改的参数，默认采用服务级配置。

  可针对不同 Topic 可靠性要求对 Topic 进行单独配置。以 root 用户登录 Kafka 客户端节点，在客户端安装目录下配置 Topic 名称为 test 的可靠性参数命令：

  cd Kafka/kafka/bin

  kafka-topics.sh --zookeeper 192.168.1.205:2181/kafka --alter --topic test --config unclean.leader.election.enable=false --config min.insync.replicas=2

  其中 192.168.1.205 为 ZooKeeper 业务 IP 地址。
- 参数配置项为服务级配置需要重启 Kafka，建议在变更窗口做服务级配置修改。

15.13 更改 Broker 的存储目录

操作场景

本章节内容适用于 MRS 3.x 及后续版本。

增加 Broker 的存储目录时，系统管理员需要在 FusionInsight Manager 中修改 Broker 的存储目录，以保证 Kafka 正常工作，新创建的主题分区将在分区最少的目录中生成。适用于以下场景：

说明

由于 Kafka 不感知磁盘容量，建议各 Broker 实例配置的磁盘个数和容量保持一致。

- 更改 Broker 角色的存储目录，所有 Broker 实例的存储目录将同步修改。
- 更改 Broker 单个实例的存储目录，只对单个实例生效，其他节点 Broker 实例存储目录不变。

对系统的影响

- 更改 Broker 角色的存储目录需要重新启动服务，服务重启时无法访问。
- 更改 Broker 单个实例的存储目录需要重新启动实例，该节点 Broker 实例重启时无法提供服务。
- 服务参数配置如果使用旧的存储目录，需要更新为新目录。

前提条件

- 在各个数据节点准备并安装好新磁盘，并格式化磁盘。
- 已安装好 Kafka 客户端。
- 更改 Broker 单个实例的存储目录时，保持活动的 Broker 实例数必须大于创建主题时指定的备份数。

操作步骤

更改 Kafka 角色的存储目录

步骤 1 以 **root** 用户登录到安装 Kafka 服务的各个数据节点中，执行如下操作。

1. 创建目标目录。

   例如目标目录为“${BIGDATA_DATA_HOME}/kafka/data2”：

   执行 **mkdir ${BIGDATA_DATA_HOME}/kafka/data2**。

2. 挂载目录到新磁盘。例如挂载“${BIGDATA_DATA_HOME}/kafka/data2”到新磁盘。

3. 修改新目录的权限。

   例如新目录路径为“${BIGDATA_DATA_HOME}/kafka/data2”：

   执行 **chmod 700 ${BIGDATA_DATA_HOME}/kafka/data2 -R** 和 **chown omm:wheel ${BIGDATA_DATA_HOME}/kafka/data2 -R**。

步骤 2 MRS 3.x 及后续版本，登录 FusionInsight Manager，然后选择“集群 > 服务 > Kafka > 配置”。

步骤 3 添加新目录到“log.dirs”的默认值后面。

在搜索框中输入“log.dirs”进行搜索，将新目录添加到配置项“log.dirs”的默认值后面，多个目录使用逗号分隔。例如“

${BIGDATA_DATA_HOME}/kafka/data1/kafka-logs,${BIGDATA_DATA_HOME}/kafka/data2/kafka-logs”。

步骤 4 单击“保存”，并单击“确定”。界面提示“操作成功”，单击“完成”。

步骤 5 选择“集群 > 服务 > Kafka ，右上角选择“更多 > 重启服务”，重启 Kafka 服务。

更改 Kafka 单个实例的存储目录

步骤 6 以 **root** 用户登录到 Broker 节点，执行如下操作。

1. 创建目标目录。

   例如目标目录为“${BIGDATA_DATA_HOME}/kafka/data2”：

   执行 **mkdir ${BIGDATA_DATA_HOME}/kafka/data2**。

2. 挂载目录到新磁盘。例如挂载“${BIGDATA_DATA_HOME}/kafka/data2”到新磁盘。

3. 修改新目录的权限。

   例如新目录路径为“${BIGDATA_DATA_HOME}/kafka/data2”：

   执行 **chmod 700 ${BIGDATA_DATA_HOME}/kafka/data2 -R** 和 **chown omm:wheel ${BIGDATA_DATA_HOME}/kafka/data2 -R**。

步骤 7 MRS 3.x 及后续版本，登录 FusionInsight Manager，然后选择“集群 > 服务 > Kafka > 实例”。

步骤 8 单击指定的 Broker 实例并切换到“实例配置”。

在搜索框中输入“log.dirs”进行搜索，将新目录添加到配置项“log.dirs”的默认值后面，多个目录使用逗号分隔。例如“${BIGDATA_DATA_HOME}/kafka/data1/kafka-logs,${BIGDATA_DATA_HOME}/kafka/data2/kafka-logs”。

步骤 9 单击“保存”，并单击“确定”，界面提示“操作成功”，单击“完成”。

步骤 10 在 Broker 实例页面选择“更多 > 重启实例”，重启 Broker 实例。

----结束

15.14 查看 Consumer Group 消费情况

操作场景

该任务指导系统管理员根据业务需求，在客户端中查看当前消费情况。

本章节内容适用于 MRS 3.x 及后续版本。

前提条件

- 系统管理员已明确业务需求，并准备一个系统用户。
- 已安装 Kafka 客户端。

操作步骤

步骤 1 以客户端安装用户，登录安装 Kafka 客户端的节点。

步骤 2 切换到 Kafka 客户端安装目录，例如“/opt/kafkaclient”。

cd /opt/kafkaclient

步骤 3 执行以下命令，配置环境变量。

source bigdata_env

步骤 4 执行以下命令，进行用户认证。（普通模式跳过此步骤）

kinit *组件业务用户*

步骤 5 执行以下命令，切换到 Kafka 客户端安装目录。

cd Kafka/kafka/bin

步骤 6 使用 **kafka-consumer-groups.sh** 查看当前消费情况。

- 查看 Offset 保存在 Kafka 上的 Consumer Group 列表：

  ./kafka-consumer-groups.sh --list --bootstrap-server <*Broker 的任意一个节点的业务 IP:21007*> **--command-config ../config/consumer.properties**

  eg:./kafka-consumer-groups.sh --bootstrap-server 192.168.1.1:21007 --list --command-config ../config/consumer.properties

- 查看 Offset 保存在 Kafka 上的 Consumer Group 消费情况：

  ./kafka-consumer-groups.sh --describe --bootstrap-server <*Broker 的任意一个节点的业务 IP:21007*> *--group 消费组名称* **--command-config ../config/consumer.properties**

  eg:./kafka-consumer-groups.sh --describe --bootstrap-server 192.168.1.1:21007 --group example-group --command-config ../config/consumer.properties

须知

1. 确保当前 consumer 在线消费。
2. 确保配置文件 consumer.properties 中的 group.id 与命令中--group 的参数均配置为待查询的 group。
3. Kafka 集群 IP 端口号安全模式下是 21007，普通模式下是 9092。

----结束

15.15 Kafka 均衡工具使用说明

操作场景

该任务指导管理员根据业务需求，在客户端中执行 Kafka 均衡工具来均衡 Kafka 集群的负载，一般用于节点的退服、入服以及负载均衡的场景。

本章节内容适用于 MRS 3.x 及后续版本。3.x 之前版本请参考 Kafka 扩容节点后数据均衡

前提条件

- 系统管理员已明确业务需求，并准备一个 Kafka 管理员用户（属于 kafkaadmin 组，普通模式不需要）。
- 已安装 Kafka 客户端。

操作步骤

步骤 1 以客户端安装用户，登录已安装 Kafka 客户端的节点。

步骤 2 切换到 Kafka 客户端安装目录，例如“/opt/kafkaclient”。

cd /opt/kafkaclient

步骤 3 执行以下命令，配置环境变量。

source bigdata_env

步骤 4 执行以下命令，进行用户认证（普通模式跳过此步骤）。

kinit *组件业务用户*

步骤 5 执行以下命令，切换到 Kafka 客户端安装目录。

cd Kafka/kafka

步骤 6 使用“kafka-balancer.sh”进行用户集群均衡，常用命令如下：

- 使用--run 命令执行集群均衡：

  ./bin/kafka-balancer.sh --run --zookeeper <*ZooKeeper 的任意一个节点的业务 IP:zkPort/**kafka***> **--bootstrap-server** <*Kafka 集群 IP: port*> **--throttle 10000000 --consumer-config config/consumer.properties --enable-az-aware --show-details**

  该命令包含均衡方案的生成和执行两部分，其中--show-details 为可选参数，表示是否打印方案明细，--throttle 表示均衡方案执行时的带宽限制，单位:bytes/sec, --enable-az-aware 为可选参数，表明生成均衡方案时，开启跨 AZ 特性，使用此参数时，请务必保证集群已开启跨 AZ 特性。

- 使用--run 命令执行节点退服：

  ./bin/kafka-balancer.sh --run --zookeeper <*ZooKeeper 的任意一个节点的业务 IP:zkPort/**kafka***> **--bootstrap-server** <*Kafka 集群 IP: port*> **--throttle 10000000 --consumer-config config/consumer.properties --remove-brokers** <*BrokerId 列表*> **--enable-az-aware --force**

  其中--remove-brokers 表示要删除的 BrokerId 列表，多个间用逗号分隔，--force 参数为可选参数，表示忽略磁盘使用率告警，强制生成迁移方案， -enable-az-aware 为可选参数，表明生成均衡方案时，开启跨 AZ 特性，使用此参数时，请务必保证集群已开启跨 AZ 特性。

  说明

  此退服命令会将待退服 Broker 节点上的数据迁移至其他 Broker 节点。

- 查看执行状态：

  ./bin/kafka-balancer.sh --status --zookeeper <*ZooKeeper 的任意一个节点的业务 IP:zkPort/**kafka***>

- 生成均衡方案：

  ./bin/kafka-balancer.sh --generate --zookeeper <*ZooKeeper 的任意一个节点的业务 IP:zkPort/**kafka***> **--bootstrap-server** <*Kafka 集群 IP:port*> **--consumer-config config/consumer.properties --enable-az-aware**

  该命令仅根据集群当前状态生成迁移方案，并打印到控制台，其中--enable-az-aware 为可选参数，表明生成迁移方案时，开启跨 AZ 特性，使用此参数时，请务必保证集群已开启跨 AZ 特性。

- 清理中间状态

  ./bin/kafka-balancer.sh --clean --zookeeper <*ZooKeeper 的任意一个节点的业务 IP:zkPort/**kafka***>

  一般在迁移没有正常执行完成时用来清理 ZooKeeper 上的中间状态信息。

须知

Kafka 集群 IP 端口号安全模式下是 21007，普通模式下是 9092。

----结束

异常情况处理

在使用 Kafka 均衡工具进行 Partition 迁移的过程中，如果出现集群中 Broker 故障导致均衡工具的执行进度阻塞，这时需要人工介入来恢复，分为以下几种场景：

- 存在 Broker 因为磁盘占有率达到 100%导致 Broker 故障的情况。
  a. 登录 FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > Kafka > 实例”，将运行状态为“正在恢复”的 Broker 实例停止并记录实例所在节点的管理 IP 地址以及对应的“broker.id”，该值可通过单击角色名称，在“实例配置”页面中选择“全部配置”，搜索“broker.id”参数获取。
  b. 以 **root** 用户登录记录的管理 IP 地址，并执行 **df -lh** 命令，查看磁盘占用率为 100%的挂载目录，例如“${BIGDATA_DATA_HOME}/kafka/data1”。
  c. 进入该目录，执行 **du -sh ***命令，查看该目录下各文件夹的大小。查看是否存在除“kafka-logs”目录外的其他文件，并判断是否可以删除或者迁移。
     - 是，删除或者迁移相关数据，然后执行 8。
     - 否，执行 4。
  d. 进入“kafka-logs”目录，执行 **du -sh ***命令，选择一个待移动的 Partition 文件夹，其名称命名规则为“Topic 名称-Partition 标识”，记录 Topic 及 Partition。
  e. 修改“kafka-logs”目录下的“recovery-point-offset-checkpoint”和“replication-offset-checkpoint”文件（两个文件做同样的修改）。
     i. 减少文件中第二行的数字（若移出多个目录，则减少的数字为移出的目录个数）。

ii. 删除待移出的 Partition 所在的行（行结构为“Topic 名称 Partition 标识 Offset”，删除前先将该行数据保存，后续此内容还要添加到目的目录下的同名文件中）。

f. 修改目的数据目录下（例如：“${BIGDATA_DATA_HOME}/kafka/data2/kafka-logs”）的“recovery-point-offset-checkpoint”和“replication-offset-checkpoint”文件（两个文件做同样的修改）。

- 增加文件中第二行的数字（若移入多个 Partition 目录，则增加的数字为移入的 Partition 目录个数）。
- 添加待移入的 Partition 行到文件末尾（行结构为“Topic 名称 Partition 标识 Offset”，直接复制 5 中保存的行数据即可）。

g. 移动数据，将待移动的 Partition 文件夹移动到目的目录下，移动完成后执行 **chown omm:wheel -R** *Partition 目录*命令修改 Partition 目录属组。

h. 登录 FusionInsight Manager，选择“集群 > 待操作集群的名称 > 服务 > Kafka > 实例”，启动停止的 Broker 实例。

i. 等待 5 至 10 分钟后查看 Broker 实例的运行状态是否为“良好”。

- 是，修复完成后按照“ALM-38001 Kafka 磁盘容量不足”告警指导彻底解决磁盘容量不足问题。
- 否，联系运维人员。

按照上述步骤将故障 Broker 进行恢复后，阻塞的均衡任务会继续执行，可使用--status 命令来查看任务的执行进度。

- 存在由其他原因导致的 Broker 故障，且问题场景单一明确，短时间内可以恢复 Broker 的情况。

  a. 根据问题根因指定恢复方案，恢复故障 Broker。

  b. 故障 Broker 恢复后，阻塞的均衡任务会继续执行，可使用--status 命令来查看任务的执行进度。

- 存在由其他原因导致的 Broker 故障，且问题场景复杂，短时间内无法恢复 Broker 的情况。

  a. 执行 **kinit** *Kafka 管理员用户*。（普通模式跳过此步骤）

  b. 使用 **zkCli.sh -server** <*ZooKeeper 集群业务 IP:zkPort/**kafka***>登录 ZooKeeper Shell。

  c. 执行 **addauth krbgroup**。（普通模式跳过此步骤）

  d. 删除“/admin/reassign_partitions”目录和“/controller”目录。

  e. 通过以上步骤强行终止迁移，待集群恢复后使用 **kafka-reassign-partitions.sh** 命令手动将中间过程中导致的多余的副本删除。

15.16 Kafka 扩容节点后数据均衡

操作场景

该任务指导管理员在 Kafka 扩容节点后，在客户端中执行 Kafka 均衡工具来均衡 Kafka 集群的负载。

本章节内容适用于 MRS 3.x 之前版本。3.x 及之后版本请参考 Kafka 均衡工具使用说明。

前提条件

- 系统管理员已明确业务需求，并准备一个 Kafka 管理员用户（属于 kafkaadmin 组，普通模式不需要）。
- 已安装 Kafka 客户端，客户端安装目录如“/opt/kafkaclient”。
- 本示例需创建两个 Topic，可参考步骤 7，分别命名为“test_2”和“test_3”，并创建“move-kafka-topic.json”文件，创建路径如“/opt/kafkaclient/Kafka/kafka”，Topic 格式内容如下：

```
{
"topics":
[{"topic":"test_2"},{"topic":"test_3"}],
"version":1
}
```

操作步骤

步骤 1 以客户端安装用户，登录安装 Kafka 客户端的节点。

步骤 2 切换到 Kafka 客户端安装目录。

cd /opt/kafkaclient

步骤 3 执行以下命令，配置环境变量。

source bigdata_env

步骤 4 执行以下命令，进行用户认证。（普通模式跳过此步骤）

kinit *组件业务用户*

步骤 5 执行以下命令进入 Kafka 客户端的 bin 目录。

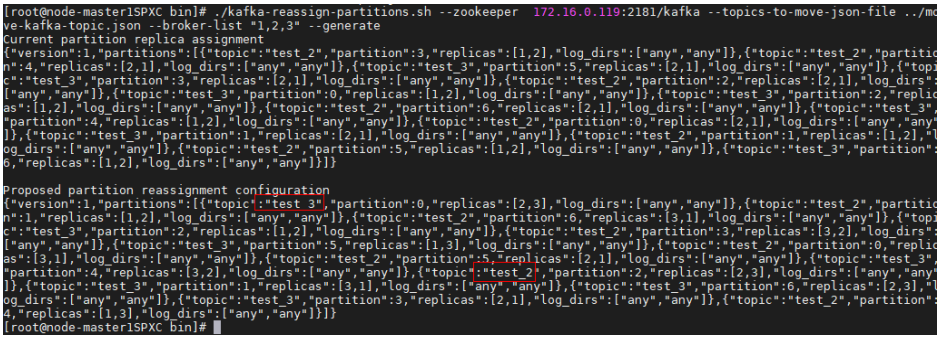
cd Kafka/kafka/bin

步骤 6 执行以下命令生成执行计划。

./kafka-reassign-partitions.sh --zookeeper *172.16.0.119:***2181/kafka --topics-to-move-json-file ../move-kafka-topic.json --broker-list "***1,2,3***" --generate**

说明

- 172.16.0.119：ZooKeeper 实例的业务 IP。
- --broker-list "1,2,3"：参数中的“1,2,3”为扩容后的所有 broker_id。



步骤 7 执行 **vim ../reassignment.json** 创建“reassignment.json”文件并保存，保存路径为“/opt/kafkaclient/Kafka/kafka”。

拷贝步骤 6 中生成的“Proposed partition reassignment configuration”下的内容至“reassignment.json”文件，如下所示：

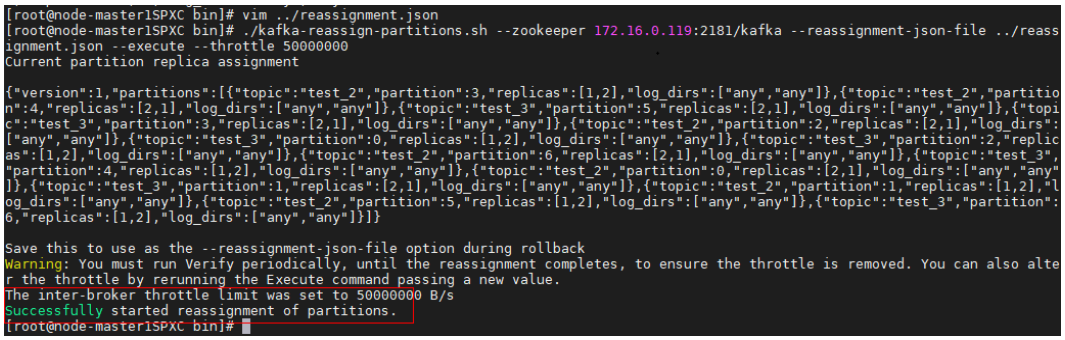
```
{"version":1,"partitions":[{"topic":"test","partition":4,"replicas":[1,2],"log_dirs":["any","any"]},{"topic":"test","partition":1,"replicas":[1,3],"log_dirs":["any","any"]},{"topic":"test","partition":3,"replicas":[3,1],"log_dirs":["any","any"]},{"topic":"test","partition":0,"replicas":[3,2],"log_dirs":["any","any"]},{"topic":"test","partition":2,"replicas":[2,1],"log_dirs":["any","any"]}]}
```

步骤 8 执行以下命令进行分区重分布。

./kafka-reassign-partitions.sh --zookeeper *172.16.0.119:***2181/kafka --reassignment-json-file ../reassignment.json --execute --throttle** *50000000*

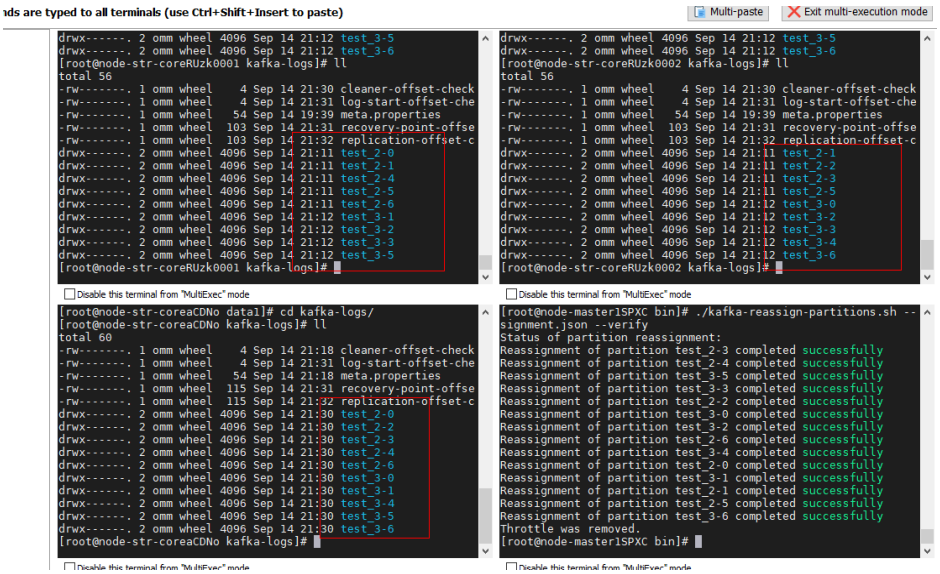
说明

--throttle 50000000：限制网络带宽为 50MB。带宽可根据数据量大小及客户对均衡时间的要求进行调整，5TB 数据量，使用 50MB 带宽，均衡时长约 8 小时。



步骤 9 执行以下命令查看迁移状态。

./kafka-reassign-partitions.sh --zookeeper *172.16.0.119:***2181/kafka --reassignment-json-file ../reassignment.json --verify**



----结束

15.17 Kafka Token 认证机制工具使用说明

操作场景

使用 Token 认证机制时对 Token 的操作。

本章节内容适用于 MRS 3.x 及后续版本的安全集群。

前提条件

- 系统管理员已明确业务需求，并准备一个系统用户。
- 已安装 Kafka 客户端。

操作步骤

步骤 1 以客户端安装用户，登录安装 Kafka 客户端的节点。

步骤 2 切换到 Kafka 客户端安装目录，例如“/opt/kafkaclient”。

cd /opt/kafkaclient

步骤 3 执行以下命令，配置环境变量。

source bigdata_env

步骤 4 执行以下命令，进行用户认证。

kinit *组件业务用户*

步骤 5 执行以下命令，切换到 Kafka 客户端安装目录。

cd Kafka/kafka/bin

步骤 6 使用 **kafka-delegation-tokens.sh** 对 Token 进行操作

- 为用户生成 Token

  ./kafka-delegation-tokens.sh --create --bootstrap-server <*IP1:PORT, IP2:PORT,...*> **--max-life-time-period** <*Long: max life period in milliseconds*> **--command-config** <*config file*> **--renewer-principal User:**<*user name*>

  例如： **./kafka-delegation-tokens.sh --create --bootstrap-server 192.168.1.1:21007,192.168.1.2:21007,192.168.1.3:21007 --command-config ../config/producer.properties --max-life-time-period -1 --renewer-principal User:username**

- 列出归属在特定用户下的所有 Token 信息

  ./kafka-delegation-tokens.sh --describe --bootstrap-server <*IP1:PORT, IP2:PORT,...*> **--command-config** <*config file*> **--owner-principal User:**<*user name*>

  例如： **./kafka-delegation-tokens.sh --describe --bootstrap-server 192.168.1.1:21007,192.168.1.2:21007,192.168.1.3:21007 --command-config ../config/producer.properties --owner-principal User:username**

- Token 有效期刷新

  ./kafka-delegation-tokens.sh --renew --bootstrap-server <*IP1:PORT, IP2:PORT,...*> **--renew-time-period** <*Long: renew time period in milliseconds*> **--command-config** <*config file*> **--hmac** <*String: HMAC of the delegation token*>

  例如： **./kafka-delegation-tokens.sh --renew --bootstrap-server 192.168.1.1:21007,192.168.1.2:21007,192.168.1.3:21007 --renew-time-period -1 --command-config ../config/producer.properties --hmac ABCDEFG**

- 销毁 Token

  ./kafka-delegation-tokens.sh --expire --bootstrap-server <*IP1:PORT, IP2:PORT,...*> **--expiry-time-period** <*Long: expiry time period in milliseconds*> **--command-config** <*config file*> **--hmac** <*String: HMAC of the delegation token*>

  例如： **./kafka-delegation-tokens.sh --expire --bootstrap-server 192.168.1.1:21007,192.168.1.2:21007,192.168.1.3:21007 --expiry-time-period -1 --command-config ../config/producer.properties --hmac ABCDEFG**

----结束

15.18 使用 KafkaUI

15.18.1 访问 KafkaUI

操作场景

MRS 集群安装 Kafka 组件后，通过 KafkaUI，用户能够便捷查询集群信息、节点状态、topic 分区、数据的生产、消费详情等多维度信息。KafkaUI 将 topic 创建、删除、配置修改、扩展分区、分区迁移等复杂易出错的管理操作界面化，降低用户使用门槛，提高运维效率。

说明

本章节内容仅适用于 MRS 3.1.2 及之后版本。

前提条件

已创建具有 KafkaUI 页面访问权限的用户，如需在页面上进行相关操作，例如创建 Topic，需同时授予用户相关权限，请参考管理 Kafka 用户权限。

对系统的影响

第一次访问 Manager 和 KafkaUI，需要在浏览器中添加站点信任以继续访问 KafkaUI。

操作步骤

步骤 1 在“KafkaManager WebUI”右侧，单击 URL 链接，访问 KafkaUI 的页面。

KafkaUI 界面支持以下功能：

- 集群内部分区重分布
- 创建、查看和删除 topic
- 对已有 topic 进行加分区、配置修改
- 查看 topic 生产数据信息
- 查看 Broker 实例信息
- 查看 Consumer Group 消费情况

----结束

15.18.2 KafkaUI 概览

操作场景

用户通过登录 KafkaUI 可在主页查看当前集群已有的 Cluster、Topic、Broker 和 Consumer Group 的基本情况，对 Topic 执行创建、删除、增加分区、修改配置操作，还可以执行集群内分区迁移。

说明

本章节内容仅适用于 MRS 3.1.2 及之后版本。

操作步骤

Cluster Summary

步骤 1 进入 KafkaUI，请参考访问 KafkaUI。

步骤 2 在“Cluster Summary”栏，可查看当前集群已有的 Topic、Broker 和 Consumer Group 数量。

步骤 3 单击“Brokers”下方的数字，可自动跳转至“Brokers”页面，在该页面的具体操作请参考使用 KafkaUI 查看 Broker。

单击“Topics”下方的数字，可自动跳转至“Topics”页面，在该页面的具体操作请参考使用 KafkaUI 管理 Topic。

单击“Consumer Group”下方的数字，可自动跳转至“Consumers”页面，在该页面的具体操作请参考使用 KafkaUI 查看 Consumer Group。

----结束

Cluster Action

步骤 1 进入 KafkaUI，请参考访问 KafkaUI。

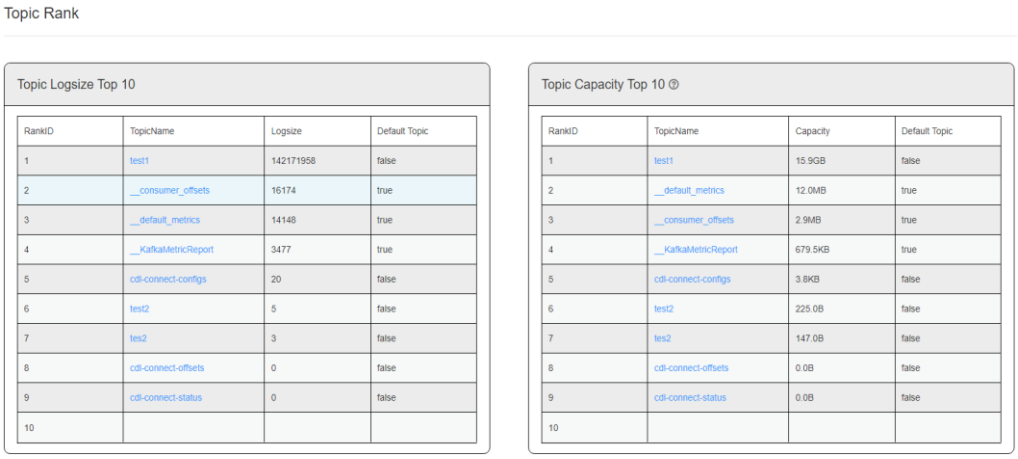
步骤 2 在“Cluster Action”栏，可创建 Topic 与分区迁移，具体操作请分别参考在 KafkaUI 创建 Topic 和在 KafkaUI 进行分区迁移章节。

----结束

Topic Rank

步骤 1 进入 KafkaUI，请参考访问 KafkaUI。

步骤 2 在“Topic Rank”栏，可查看当前集群 Topic 日志条数、数据体积大小、数据流入量、数据流出量前十名的 Topic。



步骤 3 单击“TopicName”可进入到该 Topic 的详情页面中，在该页面的具体操作请参考使用 KafkaUI 管理 Topic。

----结束

15.18.3 在 KafkaUI 创建 Topic

操作场景

通过 KafkaUI 创建 Topic。

说明

本章节内容仅适用于 MRS 3.1.2 及之后版本。

创建 Topic

步骤 1 进入 KafkaUI，请参考访问 KafkaUI。

步骤 2 单击“Create Topic”进入创建 Topic 页面。在弹出的页面中参考表 15-12 填写信息，单击“Create”，完成 Topic 创建。

表15-12 创建 Topic 信息

参数名称	参数描述	备注
Topic	Topic 的名称，只能包含英文字母、数字、中划线和下划线，且不能多于 249 个字符。	例如：kafka_ui
Partitions	Topic 的分区数量，取值范围大于等于 1，默认为 3。	-
Replication Factor	Topic 的副本因子，取值范围为 1~*N*，N 为当前集群 Broker 个数，默认为 2。	-

说明

- 用户可根据业务需要单击“Advanced Options”配置 topic 相关高级参数，通常保持默认即可。
- 安全模式集群下，执行 Create Topic 操作的用户需属于“kafkaadmin”用户组，否则将会由于鉴权失败导致无法创建。
- 非安全模式集群下，执行 Create Topic 操作不作鉴权，即任意用户都可执行 Create Topic 操作。

----结束

15.18.4 在 KafkaUI 进行分区迁移

操作场景

通过 KafkaUI 进行分区迁移。

说明

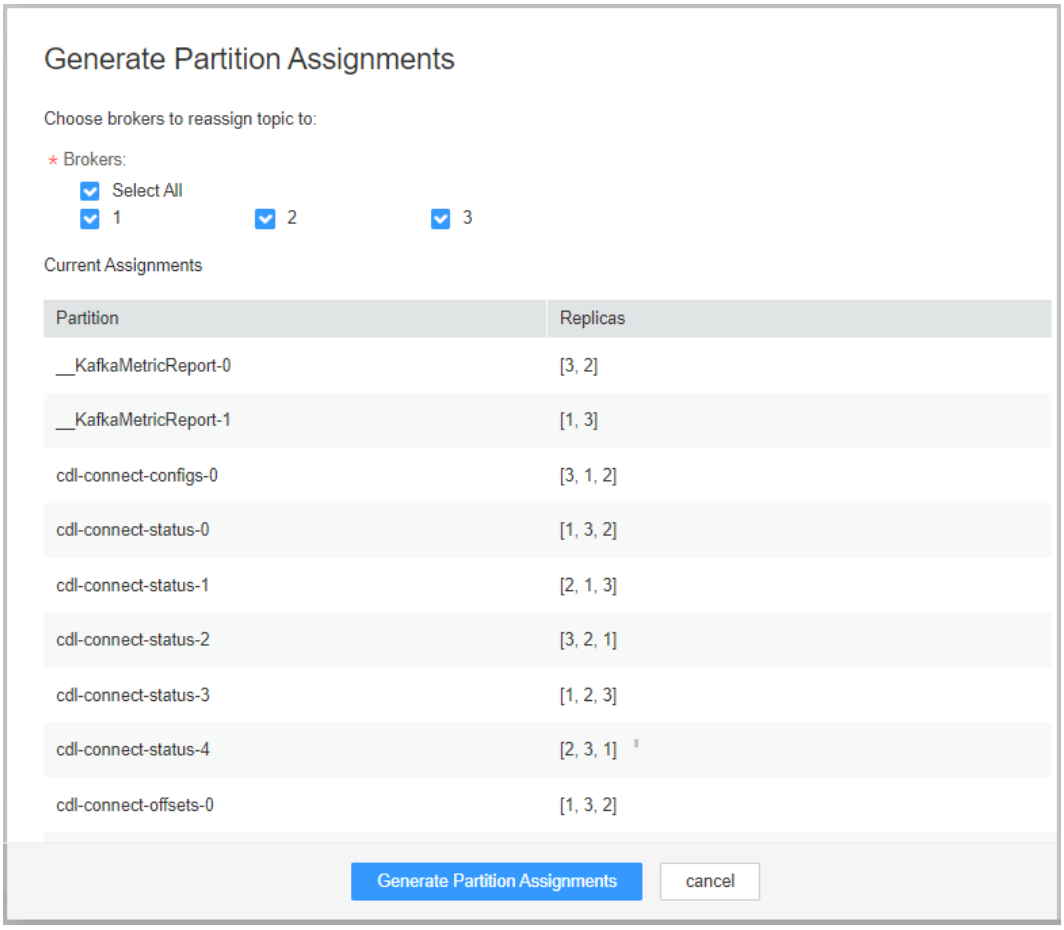
- 安全模式集群下，执行分区迁移操作的用户需属于“kafkaadmin”用户组，否则将会由于鉴权失败导致操作失败。
- 非安全模式下，KafkaUI 对任意操作不作鉴权处理。
- 本章节内容仅适用于 MRS 3.1.2 及之后版本。

分区迁移

步骤 1 进入 KafkaUI，请参考访问 KafkaUI。单击“Generate assignment”进入分区迁移页面。

步骤 2 在“Brokers”处选择要将主题重新分配的 Broker。

步骤 3 单击“Generate Partition Assignments”生成分区迁移方案。



步骤 4　继续单击“Run assignment”执行分区迁移方案，完成分区迁移。

----结束

15.18.5 使用 KafkaUI 管理 Topic

操作场景

通过 KafkaUI 查看 Topic 详情、修改 Topic Configs、增加 Topic 分区个数、删除 Topic，并可实时查看不同时段的生产数据条数。

说明

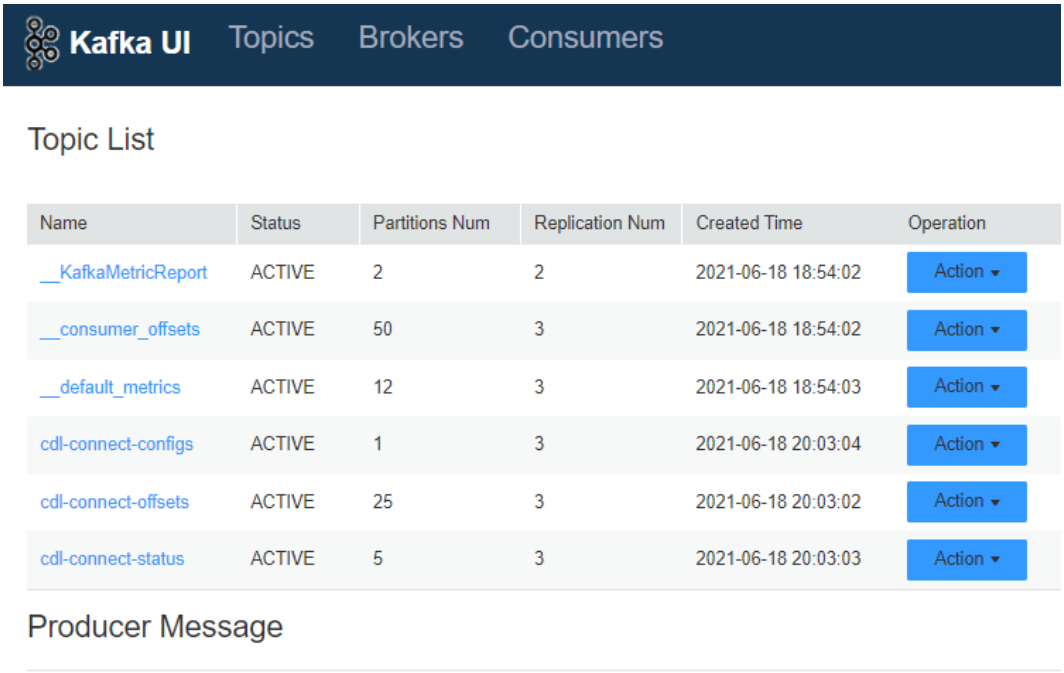
- 安全模式下，KafkaUI 对查看 Topic 详情操作不作鉴权处理，即任何用户都可以查询 Topic 信息；对于修改 Topic Configs、增加 Topic 分区个数、删除 Topic 场景，需保证 KafkaUI 登录用户属于“kafkaadmin”用户组或者单独给用户授予对应操作权限，否则将会鉴权失败。
- 非安全模式下，KafkaUI 对所有操作不作鉴权处理。
- 本章节内容仅适用于 MRS 3.1.2 及之后版本。

查看 Topic 详情

步骤 1　进入 KafkaUI，请参考访问 KafkaUI。

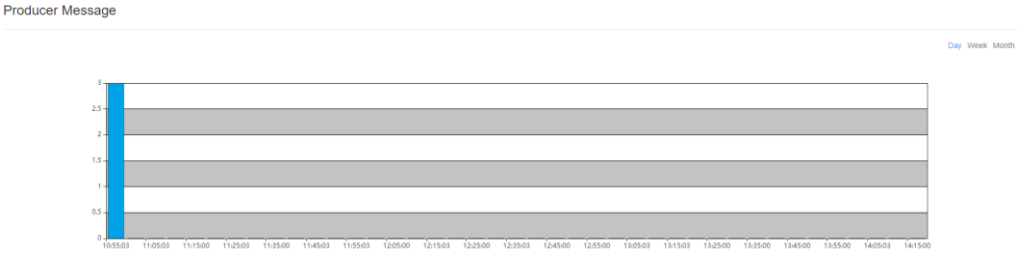
步骤 2 单击“Topics”，进入 Topic 管理页面。

步骤 3 在“Topic List”栏可查看当前集群已创建的 Topic 的名称、状态、分区数量、创建时间和副本个数等信息。



步骤 4 单击 Topic 名称可进入 Topic 详情页面。在该页面可查看 Topic 与分区的详细信息。

步骤 5 在“Producer Message”栏可根据业务需求选择“Day”、“Week”、“Month”不同时段查看此 Topic 生产数据条数。



----结束

修改 Topic 配置

步骤 1　进入 KafkaUI，请参考访问 KafkaUI。

步骤 2　单击“Topics”，进入 Topic 管理页面。

步骤 3　在待修改项的“Operation”列单击“Action > Config”，弹出的页面中可修改 Topic 的“Key”和“Value”值，如需要添加多条，可单击+添加。

步骤 4　单击“OK”完成修改。

----结束

搜索 Topic

步骤 1　进入 KafkaUI，请参考访问 KafkaUI。

步骤 2　单击“Topics”，进入 Topic 管理页面。

步骤 3　在页面右上角，用户可以输入 Topic 名称搜索查看该 Topic 信息。

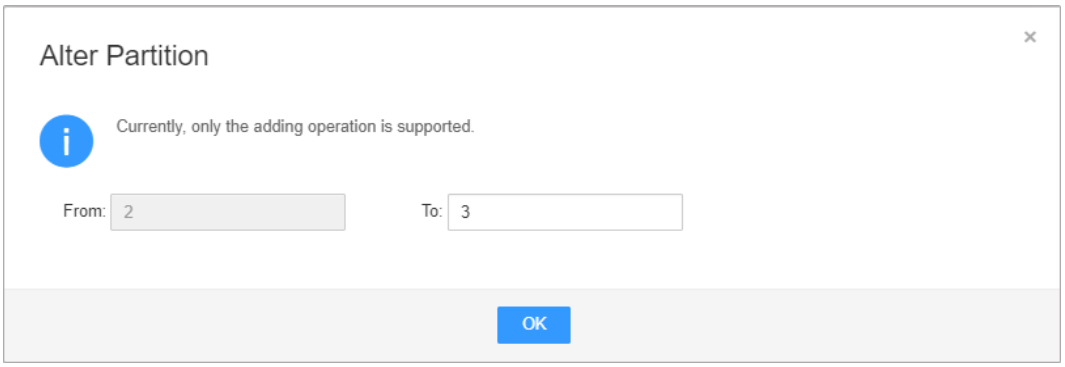
----结束

增加分区

步骤 1　进入 KafkaUI，请参考访问 KafkaUI。

步骤 2　单击“Topics”，进入 Topic 管理页面。

步骤 3　在待修改项的“Operation”列单击“Action > Alter”，弹出的页面中修改 Topic 分区。



说明

目前集群只支持增加分区操作，即修改的分区个数要大于原设置的分区个数。

步骤 4　单击“OK”完成修改。

----结束

删除 Topic

步骤 1　进入 KafkaUI，请参考访问 KafkaUI。

步骤 2 单击“Topics”，进入 Topic 管理页面。

步骤 3 在待修改项的“Operation”列单击“Action > Delete”。

步骤 4 在弹出的确认信息页面中单击“OK”即可完成删除。

说明

系统默认内置的 Topic 不支持删除操作。

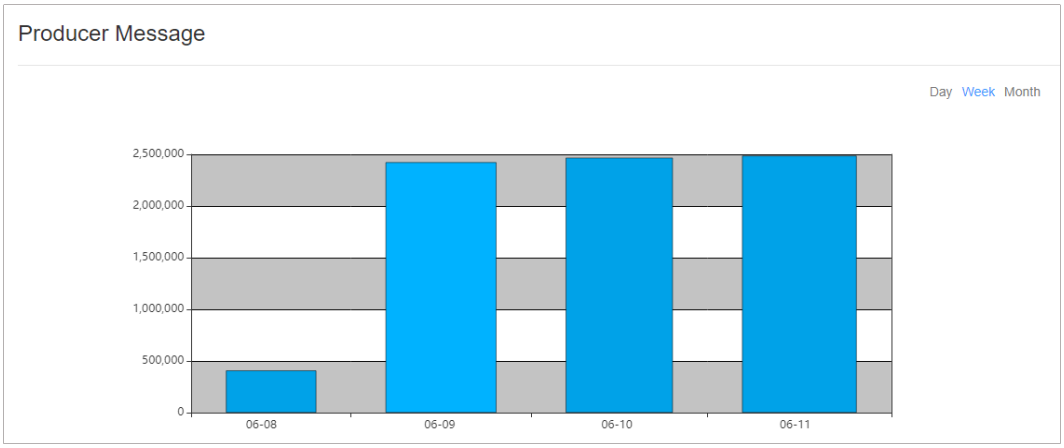
----结束

查看生产数据条数

步骤 1 进入 KafkaUI，请参考访问 KafkaUI。

步骤 2 单击“Topics”，进入 Topic 管理页面。

步骤 3 在“Producer Message”栏可选择“Day”、“Week”、“Month”不同时段查看当前集群所有集群生产数据条数。



----结束

15.18.6 使用 KafkaUI 查看 Broker

操作场景

通过 KafkaUI 可查看的 Broker 的详情信息与 Broker 节点数据流量的 jmx 指标。

说明

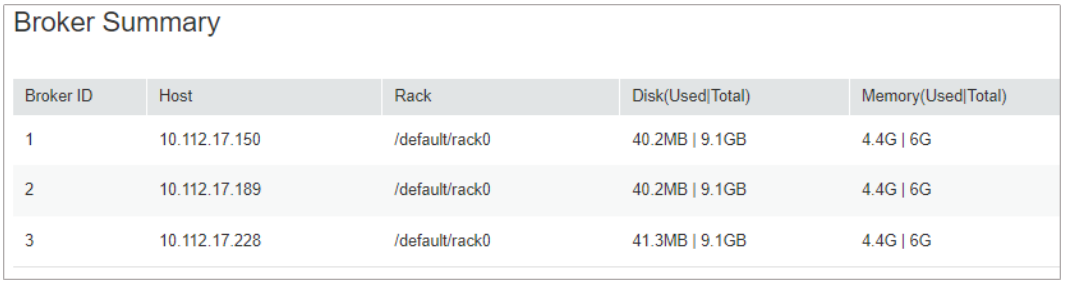
本章节内容仅适用于 MRS 3.1.2 及之后版本。

查看 Broker

步骤 1 进入 KafkaUI，请参考访问 KafkaUI。

步骤 2 单击“Brokers”，进入 Broker 详情页面。

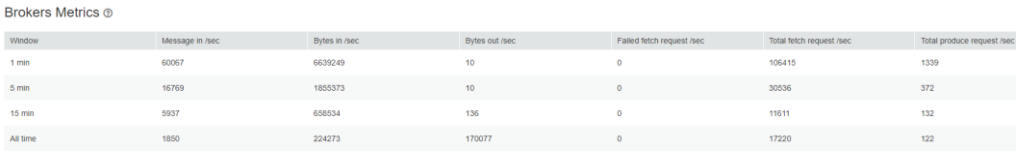
步骤 3 在“Broker Summary”一栏可查看 Broker 的“Broker ID”、“Host”、“Rack”、“Disk(Used|Total)”和“Memory(Used|Total)”。



Broker Summary

Broker ID	Host	Rack	Disk(Used|Total)	Memory(Used|Total)
1	10.112.17.150	/default/rack0	40.2MB | 9.1GB	4.4G | 6G
2	10.112.17.189	/default/rack0	40.2MB | 9.1GB	4.4G | 6G
3	10.112.17.228	/default/rack0	41.3MB | 9.1GB	4.4G | 6G

步骤 4 在“Brokers Metrics”处可查看 Broker 节点数据流量的 jmx 指标，包括在不同时段的时间窗口内，Broker 节点平均每秒流入消息条数， 每秒流入消息字节数， 每秒流出消息字节数，每秒失败的请求数，每秒总的请求数和每秒生产的请求数。



Brokers Metrics

Window	Message in /sec	Bytes in /sec	Bytes out /sec	Failed fetch request /sec	Total fetch request /sec	Total produce request /sec
1 min	60067	6639249	10	0	106415	1339
5 min	16769	1855373	10	0	30536	372
15 min	5937	658534	136	0	11611	132
All time	1850	224273	170077	0	17220	122

----结束

搜索 Broker

步骤 1 进入 KafkaUI，请参考访问 KafkaUI。

步骤 2 单击“Brokers”，进入 Broker 详情页面。

步骤 3 在页面右上角，用户可以输入主机 IP 地址或者机架配置信息搜索查看该 Broker 信息。

----结束

15.18.7 使用 KafkaUI 查看 Consumer Group

操作场景

通过 KafkaUI 可查看消费组的基本信息以及组内包含的 Topic 的消费状态。

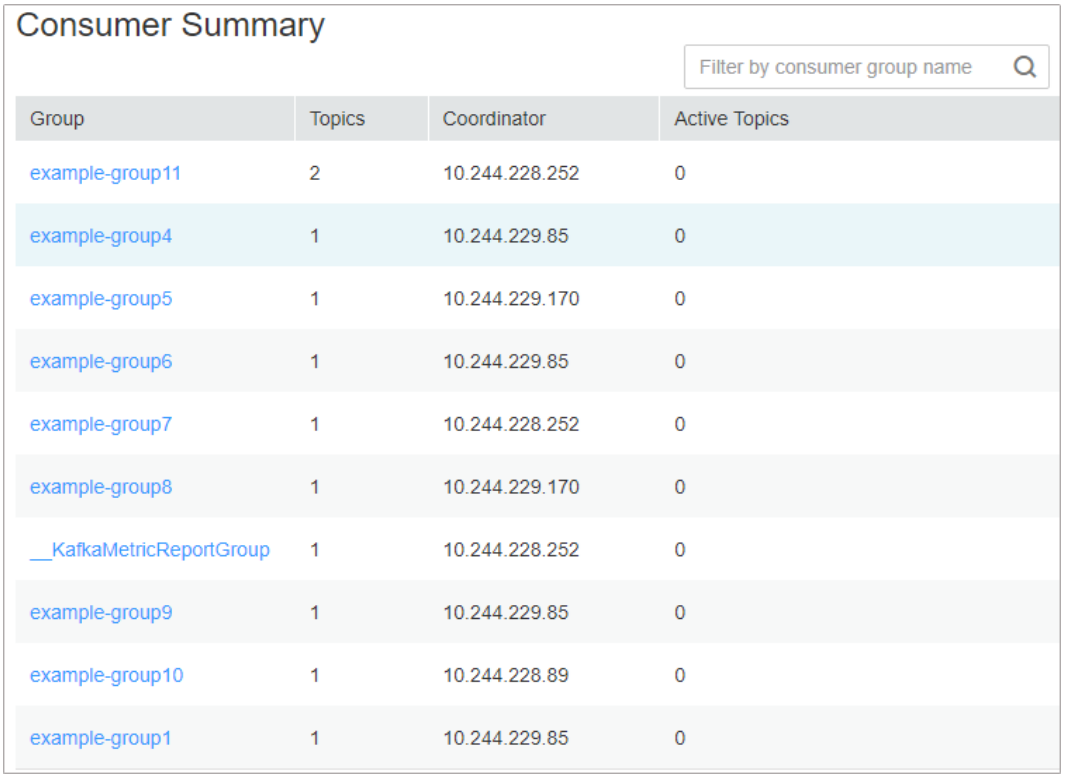
说明

本章节内容仅适用于 MRS 3.1.2 及之后版本。

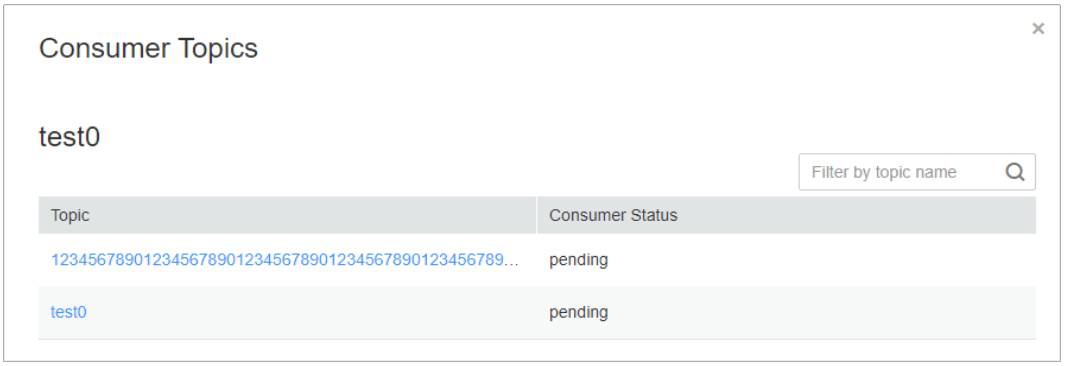
查看消费组

步骤 1 进入 KafkaUI，请参考访问 KafkaUI。

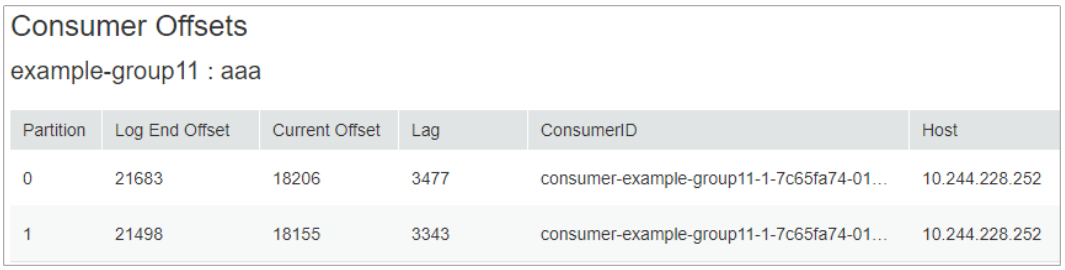
步骤 2 单击“Consumers”，进入消费组详情页面，可以查看当前集群内的所有 ConsumerGroups，并可以查看各个 ConsumerGroups Coordinator 所在节点 IP，在页面右上角，用户可以输入 ConsumerGroup 来搜索指定的 ConsumerGroup 信息。



步骤 3 在 Consumer Summary 一栏，可查看当前集群已存在的消费组，单击消费组名称，可查看该消费组所消费过的 Topic，消费过的 Topic 有两种状态：“pending”和“running”，分别表示“曾经消费过但现在未消费”和“现在正在消费”，在弹框右上角，可以输入 Topic 名来进行过滤。



步骤 4 单击 Topic 名称，进入 Consumer Offsets 页面，可查看 Topic 消费详情。

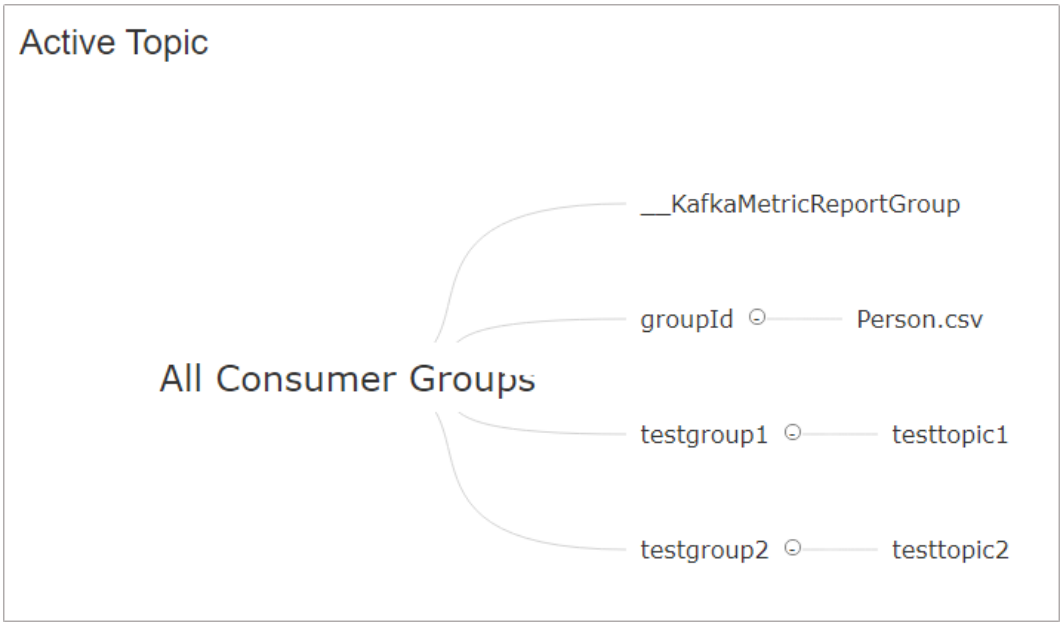


----结束

查看消费血缘图

步骤 1 进入 KafkaUI，请参考访问 KafkaUI。

步骤 2 单击“Consumers”，进入消费组详情页面。在 Active Topic 处可以查看当前集群所有的消费组，以及各个 Consumer Group 正在消费的 Topic。



说明

MRS 集群当前不支持单击消费组名称进行跳转。

----结束

15.19 Kafka 日志介绍

本章节内容适用于 MRS 3.x 及后续版本。

日志描述

日志路径：Kafka 相关日志的默认存储路径为“/var/log/Bigdata/kafka”，审计日志的默认存储路径为“/var/log/Bigdata/audit/kafka”。

- Broker：“/var/log/Bigdata/kafka/broker”（运行日志）

日志归档规则：Kafka 的日志启动了自动压缩归档功能，默认情况下，当日志大小超过 30MB 的时候，会自动压缩，压缩后的日志文件名规则为：“<原有日志名>-<yyyy-mm-dd_hh-mm-ss>.[编号].log.zip”。默认最多保留最近的 20 个压缩文件，压缩文件保留个数和压缩文件阈值可以配置。

表15-13 Broker 日志列表

日志类型	日志文件名	描述

日志类型	日志文件名	描述
运行日志	server.log	Broker 进程的 server 运行日志。
	controller.log	Broker 进程的 controller 运行日志。
	kafka-request.log	Broker 进程的 request 运行日志。
	log-cleaner.log	Broker 进程的 cleaner 运行日志。
	state-change.log	Broker 进程的 state-change 运行日志。
	kafkaServer-<SSH_USER>-<DATE>-<PID>-gc.log	Broker 进程的 GC 日志。
	postinstall.log	Broker 安装后的工作日志。
	prestart.log	Broker 启动前的工作日志。
	checkService.log	Broker 启动是否成功的检查日志。
	start.log	Broker 进程启动日志。
	stop.log	Broker 进程停止日志。
	checkavailable.log	Kafka 服务健康状态检查日志。
	checkInstanceHealth.log	Broker 实例健康状态检测日志。
	kafka-authorizer.log	Broker 鉴权日志。
	kafka-root.log	Broker 基础日志。
	cleanup.log	Broker 卸载的清理日志。
	metadata-backup-recovery.log	Broker 备份恢复日志。
	ranger-kafka-plugin-enable.log	Broker 启动 Ranger 插件日志。
	server.out	Broker jvm 日志。
	audit.log	Ranger 鉴权插件鉴权日志。此日志统一归档在“/var/log/Bigdata/audit/kafka”目录下。

日志级别

Kafka 提供了如表 15-14 所示的日志级别。

运行日志的级别优先级从高到低分别是 ERROR、WARN、INFO、DEBUG，程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。

表15-14 日志级别

级别	描述
ERROR	ERROR 表示系统运行的错误信息。
WARN	WARN 表示当前事件处理存在异常信息。
INFO	INFO 表示记录系统及各事件正常运行状态信息。
DEBUG	DEBUG 表示记录系统及系统的调试信息。

如果您需要修改日志级别，请执行如下操作：

步骤 1 请参考修改集群服务配置参数，进入 Kafka 的“全部配置”页面。

步骤 2 左边菜单栏中选择所需修改的角色所对应的日志菜单。

步骤 3 选择所需修改的日志级别。

步骤 4 保存配置，在弹出窗口中单击“确定”使配置生效。

----结束

日志格式

Kafka 的日志格式如下所示

表15-15 日志格式

日志类型	格式	示例
运行日志	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<产生该日志的线程名字>|<log 中的 message>|<日志事件调用类全名>(<日志打印文件>:<行号>)	2015-08-08 11:09:53,483	INFO	[main]	Loading logs.	kafka.log.LogManager (Logging.scala:68)
	<yyyy-MM-dd HH:mm:ss><HostName><组件名><logLevel><Message>	2015-08-08 11:09:51 10-165-0-83 Kafka INFO Running kafka-start.sh.

15.20 性能调优

15.20.1 Kafka 性能调优

操作场景

通过调整 Kafka 服务端参数，可以提升特定业务场景下 Kafka 的处理能力。

参数调优

修改服务配置参数，请参考修改集群服务配置参数。调优参数请参考表 15-16。

表15-16 调优参数

配置参数	缺省值	调优场景
num.recovery.threads.per.data.dir	10	在 Kafka 启动过程中，数据量较大情况下，可调大此参数，可以提升启动速度。
background.threads	10	Broker 后台任务处理的线程数目。数据量较大的情况下，可适当调大此参数，以提升 Broker 处理能力。
num.replica.fetchers	1	副本向 Leader 请求同步数据的线程数，增大这个数值会增加副本的 I/O 并发度。
num.io.threads	8	Broker 用来处理磁盘 I/O 的线程数目，这个线程数目建议至少等于硬盘的个数。
KAFKA_HEAP_OPTS	-Xmx6G -Xms6G	Kafka JVM 堆内存设置。当 Broker 上数据量较大时，应适当调整堆内存大小。

15.21 Kafka 特性说明

Kafka Idempotent 特性

特性说明：Kafka 从 0.11.0.0 版本引入了创建幂等性 Producer 的功能，开启此特性后，Producer 自动升级成幂等性 Producer，当 Producer 发送了相同字段值的消息后，Broker 会自动感知消息是否重复，继而避免数据重复。需要注意的是，这个特性只能保证单分区上的幂等性，即一个幂等性 Producer 能够保证某个主题的一个分区内不出现重复消息；只能实现单会话上的幂等性，这里的会话指的是 Producer 进程的一次运行，即重启 Producer 进程后，幂等性不保证。

开启方法：

1. 二次开发代码中添加 “props.put(“enable.idempotence”，true)”。
2. 客户端配置文件中添加 “enable.idempotence = true”。

Kafka Transaction 特性

特性说明：Kafka 在 0.11 版本中，引入了事务特性，Kafka 事务特性指的是一系列的生产者生产消息和消费者提交偏移量的操作在一个事务中，或者说是一个原子操作，生产消息和提交偏移量同时成功或者失败，此特性提供的是 read committed 隔离级别的事务，保证多条消息原子性的写入到目标分区，同时也能保证 Consumer 只能看到成功提交的事务消息。Kafka 中的事务特性主要用于以下两种场景：

1. 生产者发送多条数据可以封装在一个事务中，形成一个原子操作。多条消息要么都发送成功，要么都发送失败。
2. read-process-write 模式：将消息消费和生产封装在一个事务中，形成一个原子操作。在一个流式处理的应用中，常常一个服务需要从上游接收消息，然后经过处理后送达到下游，这就对应着消息的消费和生产。

二次开发代码样例如下：

```
// 初始化配置,开启事务特性
Properties props = new Properties();
props.put("enable.idempotence", true);
props.put("transactional.id", "transaction1");
...

KafkaProducer producer = new KafkaProducer<String, String>(props);

// init 事务
producer.initTransactions();
try {
    // 开启事务
    producer.beginTransaction();
    producer.send(record1);
    producer.send(record2);
    // 结束事务
    producer.commitTransaction();
} catch (KafkaException e) {
    // 事务 abort
    producer.abortTransaction();
}
```

就近消费特性

特性说明：Kafka 2.4.0 之前版本，客户端的生产、消费都是面向各个 partition 的 leader 副本，follower 副本仅用来做数据冗余，不对外提供服务，常会导致 leader 副本压力较大，且在跨机房、机架的消费场景下，常会导致大量的机房、机架间的数据传输；Kafka 2.4.0 及之后版本，Kafka 内核支持从 follower 副本消费数据，在跨机房、机架的场景中，会大大降低数据传输量，减轻网络带宽压力。社区开放了 ReplicaSelector 接口来支持此特性，MRS Kafka 中默认提供两种实现此接口的方式。

1. RackAwareReplicaSelector：优先从相同机架的副本进行消费（机架内就近消费特性）。
2. AzAwareReplicaSelector：优先从相同 AZ 内的节点上的副本进行消费（AZ 内就近消费特性）。

以 RackAwareReplicaSelector 为例，描述实现就近消费副本的选取：

```
public class RackAwareReplicaSelector implements ReplicaSelector {

   @Override
   public Optional<ReplicaView> select(TopicPartition topicPartition,
                                       ClientMetadata clientMetadata,
                                       PartitionView partitionView) {
      if (clientMetadata.rackId() != null && !clientMetadata.rackId().isEmpty()) {
         Set<ReplicaView> sameRackReplicas = partitionView.replicas().stream()
                // 过滤与客户端处于相同 Rack 的副本
                .filter(replicaInfo ->
clientMetadata.rackId().equals(replicaInfo.endpoint().rack()))
                .collect(Collectors.toSet());
         if (sameRackReplicas.isEmpty()) {
            // 如果没有副本与客户端处于相同 Rack，则返回 leader 副本
            return Optional.of(partitionView.leader());
         } else {
            // 到这里说明存在与客户端位于同一 Rack 的副本
            if (sameRackReplicas.contains(partitionView.leader())) {
               // 如果客户端和 leader 在同一个机架，则优先返回 leader 副本
               return Optional.of(partitionView.leader());
            } else {
               // 否则，返回和 leader 同步最新的副本
               return sameRackReplicas.stream().max(ReplicaView.comparator());
            }
         }
      } else {
         // 如果客户端请求中不包含机架信息，则默认返回 leader 副本
         return Optional.of(partitionView.leader());
      }
   }
}
```

开启方法：

1. 服务端：根据不同特性更新“replica.selector.class”配置项：
   - 开启“机架内就近消费特性”，配置为“org.apache.kafka.common.replica.RackAwareReplicaSelector”。
   - 开启“AZ 内就近消费特性”，配置为“org.apache.kafka.common.replica.AzAwareReplicaSelector”。
2. 客户端：在“{*客户端安装目录*}/Kafka/kafka/config”目录中的“consumer.properties”消费配置文件里添加“client.rack”配置项：
   - 若服务端开启“机架内就近消费特性”，添加客户端所处的机架信息，如 client.rack = /default0/rack1。
   - 若服务端开启“AZ 内就近消费特性”，添加客户端所处的机架信息，如 client.rack = /AZ1/rack1。

Ranger 统一鉴权特性

特性说明：在 Kafka 2.4.0 之前版本，Kafka 组件仅支持社区自带的 SimpleAclAuthorizer 鉴权插件，Kafka 2.4.0 及之后版本，MRS Kafka 同时支持 Ranger 鉴权插件和社区自带鉴权插件。默认使用 Ranger 鉴权，基于 Ranger 鉴权插件，可进行细粒度的 Kafka Acl 管理。

说明

服务端使用 Ranger 鉴权插件时，若“allow.everyone.if.no.acl.found”配置为“true”，使用非安全端口访问时，所有行为将直接放行。建议使用 Ranger 鉴权插件的安全集群，不要开启“allow.everyone.if.no.acl.found”。

15.22 Kafka 节点内数据迁移

操作场景

该任务指导管理员根据业务需求，通过 Kafka 客户端命令，在不停止服务的情况下，进行节点内磁盘间的分区数据迁移。

前提条件

- 系统管理员已明确业务需求，并准备一个 Kafka 用户（属于 kafkaadmin 组，普通模式不需要）。
- 已安装 Kafka 客户端。
- Kafka 实例状态和磁盘状态均正常。
- 根据待迁移分区当前的磁盘空间占用情况，评估迁移后，不会导致新迁移后的磁盘空间不足。

操作步骤

步骤 1 以客户端安装用户，登录已安装 Kafka 客户端的节点。

步骤 2 执行以下命令，切换到 Kafka 客户端安装目录，例如“/opt/kafkaclient”。

cd /opt/kafkaclient

步骤 3 执行以下命令，配置环境变量。

source bigdata_env

步骤 4 执行以下命令，进行用户认证（普通模式跳过此步骤）。

kinit *组件业务用户*

步骤 5 执行以下命令，切换到 Kafka 客户端目录。

cd Kafka/kafka/bin

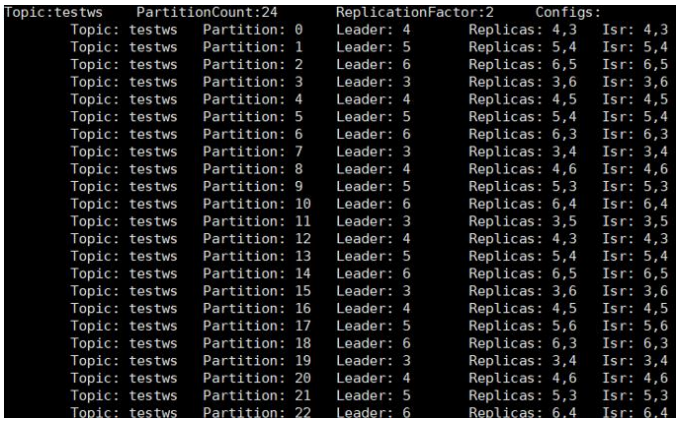
步骤 6 执行以下命令，查看待迁移的 Partition 对应的 Topic 的详细信息。

安全模式：

./kafka-topics.sh --describe --bootstrap-server *Kafka 集群 IP:21007* **--command-config ../config/client.properties --topic** *主题名称*

普通模式：

./kafka-topics.sh --describe --bootstrap-server *Kafka 集群 IP:21005* **--command-config ../config/client.properties --topic** *主题名称*



步骤 7 执行以下命令，查询 Broker_ID 和 IP 对应关系。

./kafka-broker-info.sh --zookeeper *ZooKeeper 的 quorumpeer 实例业务 IP:ZooKeeper 客户端端口号***/kafka**

```
Broker_ID    IP_Address
--------------------------
4          192.168.0.100
5          192.168.0.101
6          192.168.0.102
```

说明

- ZooKeeper 的 quorumpeer 实例业务 IP：

  ZooKeeper 服务所有 quorumpeer 实例业务 IP。登录 FusionInsight Manager，选择“集群 > 服务 > ZooKeeper > 实例”，可查看所有 quorumpeer 实例所在主机业务 IP 地址。

- ZooKeeper 客户端端口号：

  登录 FusionInsight Manager，选择“集群 > 服务 > ZooKeeper”，在“配置”页签查看“clientPort”的值。默认为 24002。

步骤 8 从步骤 6 和步骤 7 回显中获取分区的分布信息和节点信息，在当前目录下创建执行重新分配的 json 文件。

以迁移的是 Broker_ID 为 6 的节点的分区为例，迁移到"/srv/BigData/hadoop/data1/kafka-logs”，完成迁移所需的 json 配置文件，内容如下。

```
{"partitions":[{"topic": "testws","partition": 2,"replicas": [6,5],"log_dirs":
["/srv/BigData/hadoop/data1/kafka-logs","any"]}],"version":1}
```

说明

- topic 为 Topic 名称，此处以 testws 为例，具体以实际为准。
- partition 为 Topic 分区。

- replicas 中的数字对应 Broker_ID。
- log_dirs 为需要迁移的磁盘路径。此样例迁移的是 Broker_ID 为 6 的节点，Broker_ID 为 5 的节点对应的 log_dirs 可设置为“any”，Broker_ID 为 6 的节点对应的 log_dirs 设置为“/srv/BigData/hadoop/data1/kafka-logs”。**注意路径需与节点对应。**

步骤 9 使用如下命令，执行重分配操作。

安全模式：

./kafka-reassign-partitions.sh --bootstrap-server *Broker 业务 IP*:**21007 --command-config ../config/client.properties --zookeeper** *{zk_host}:{port}*/**kafka --reassignment-json-file** *步骤 8 中编写的 json 文件路径* **--execute**

普通模式：

./kafka-reassign-partitions.sh --bootstrap-server *Broker 业务 IP*:**21005 --command-config ../config/client.properties --zookeeper** *{zk_host}:{port}*/**kafka --reassignment-json-file** *步骤 8 中编写的 json 文件路径* **--execute**

提示“Successfully started reassignment of partitions”表示执行成功。

----结束

15.23 Kafka 常见问题

15.23.1 如何解决 Kafka topic 无法删除的问题

问题

删除 Kafka topic 后发现未成功删除，如何正常删除？

回答

- 可能原因一：配置项“delete.topic.enable”未配置为“true”，只有配置为“true”才能执行真正删除。
- 可能原因二：“auto.create.topics.enable”配置为“true”，其他应用程序有使用该 Topic，并且一直在后台运行。

解决方法：

- 针对原因一：配置页面上将“delete.topic.enable”设置为“true”。
- 针对原因二：先停掉后台使用该 Topic 的应用程序，或者“auto.create.topics.enable”配置为“false”（需要重启 Kafka 服务），然后再做删除操作。

16 使用 KafkaManager

16.1 KafkaManager 介绍

KafkaManager 是 Apache Kafka 的管理工具，提供 Kafka 集群界面化的 Metric 监控和集群管理。

通过 KafkaManager 可以：

- 支持管理多个 Kafka 集群
- 支持界面检查集群状态（主题，消费者，偏移量，分区，副本，节点）
- 支持界面执行副本的 leader 选举
- 使用选择生成分区分配以选择要使用的分区方案
- 支持界面执行分区重新分配（基于生成的分区方案）
- 支持界面选择配置创建主题（支持多种 Kafka 版本集群）
- 支持界面删除主题（仅支持 0.8.2+并设置了 delete.topic.enable = true）
- 支持批量生成多个主题的分区分配，并可选择要使用的分区方案
- 支持批量运行重新分配多个主题的分区
- 支持为已有主题增加分区
- 支持更新现有主题的配置
- 可以为分区级别和主题级别度量标准启用 JMX 查询
- 可以过滤掉 zookeeper 中没有 ids / owner /＆offsets /目录的使用者。

16.2 访问 KafkaManager 的 WebUI

用户可以通过 KafkaManager 的 WebUI，在图形化界面监控管理 Kafka 集群。

前提条件

- 已安装 KafkaManager 服务的集群。
- 获取用户“admin”帐号密码。“admin”密码在创建 MRS 集群时由用户指定。

访问 KafkaManager 的 WebUI

步骤 1 登录集群详情页面，选择“组件管理 > KafkaManager”。

说明

若集群详情页面没有“组件管理”页签，请先完成 IAM 用户同步（在集群详情页的“概览”页签，单击“IAM 用户同步”右侧的“同步”进行 IAM 用户同步）。

步骤 2 在 KafkaManager 概述的“KafkaManager WebUI”中单击任意一个 UI 链接，打开 KafkaManager 的 WebUI 页面。

KafkaManager 的 WebUI 支持查看以下信息：

- Kafka 集群列表
- Kafka 集群 Broker 节点列表和 Metric 监控
- Kafka 集群副本监控
- Kafka 集群 Consumer 监控

说明

在 KafkaManager 的任何子页面单击左上角 KafkaManager 的 Logo 都可以回到 KafkaManager 的 WebUI 主界面，显示集群列表信息。

----结束

16.3 管理 Kafka 集群

管理 Kafka 集群包含以下内容：

- 添加集群到 KafkaManager 的 WebUI 界面
- 更新集群参数
- 删除 KafkaManager 的 WebUI 界面的集群

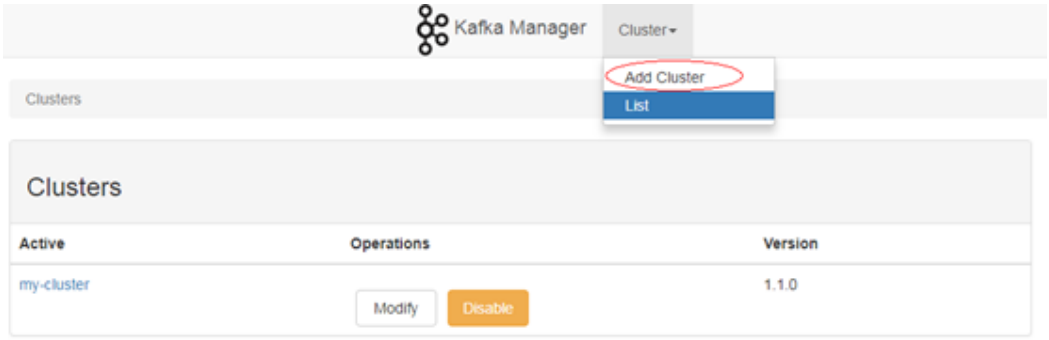
添加集群到 KafkaManager 的 WebUI 界面

首次创建 Kafka 集群后会在 KafkaManager 的 WebUI 界面创建名为 my-cluster 的默认 Kafka 集群，用户也可以在 KafkaManager 的 WebUI 界面自行添加已经通过 MRS 控制台创建的 Kafka 集群，用于管理多个 Kafka 集群。

步骤 1 登录 KafkaManager 的 WebUI 界面。

步骤 2 在页面上方选择“Cluster > Add Cluster” 。

图16-1 添加集群



步骤 3　设置待添加集群的参数，如下参数请参考样例，其他参数默认不需要修改。

表16-1 需修改的集群参数

参数名称	取值样例	说明
Cluster Name	mrs-demo	待添加集群在 KafkaManager 的 WebUI 界面中显示的名称。
Cluster Zookeeper Hosts	zk1_ip:zk1_port, zk2_ip:zk2_port/kafka	待添加集群的 Zookeeper 地址。
Kafka Version	1.1.0	待添加集群的 Kafka 版本，默认 1.1.0。
Enable JMX Polling (Set JMX_PORT env variable before starting kafka server)	勾选	-
Poll consumer information (Not recommended for large # of consumers)	勾选	-
Enable Active OffsetCache (Not recommended for large # of consumers)	勾选	-
Display Broker and Topic Size (only works after applying this patch)	勾选	-
Security Protocol	PLAINTEXT	• 开启 Kerberos 的 Kafka 集群选择 SASL_PLAINTEXT • 未开启 Kerberos 集群选择 PLAINTEXT

步骤 4　单击“Save”完成添加集群。

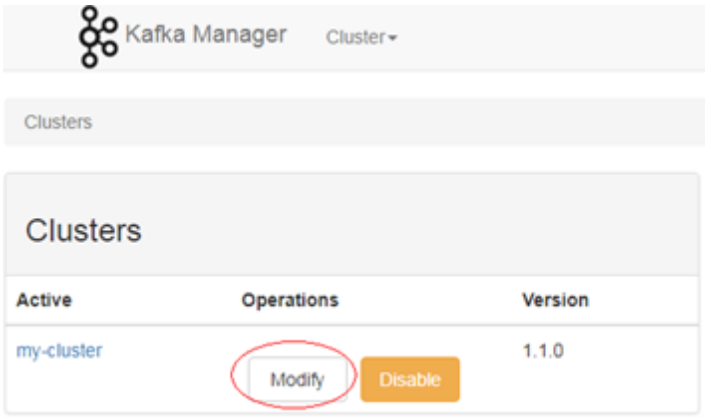
----结束

更新集群参数

步骤 1　登录 KafkaManager 的 WebUI 界面。

步骤 2　在对应集群的“Operations”列单击“Modify”。

图16-2 更新集群参数



步骤 3　进入集群配置参数页面，修改集群参数。

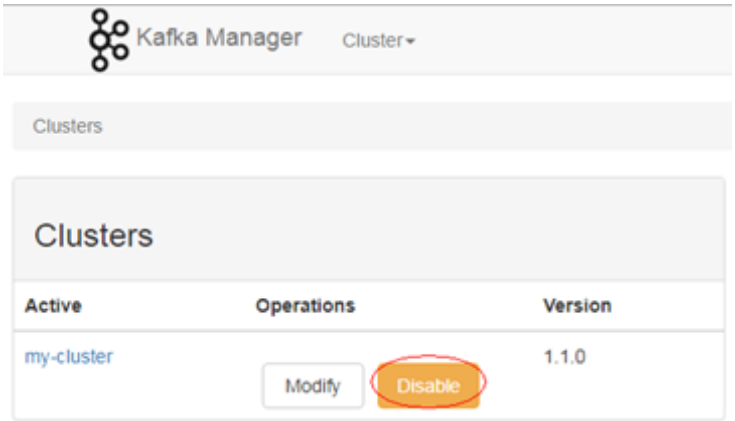
----结束

删除 KafkaManager 的 WebUI 界面的集群

步骤 1　登录 KafkaManager 的 WebUI 界面。

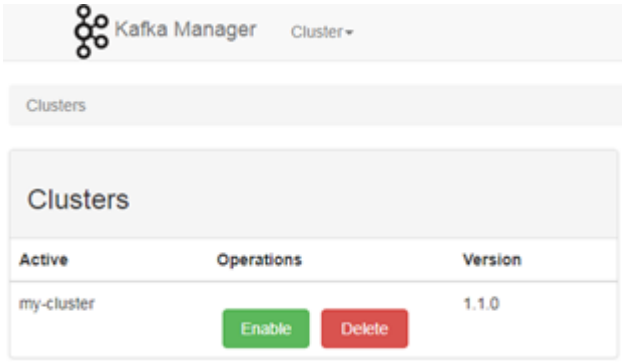
步骤 2　在对应集群的“Operations”列单击“Disable”。

图16-3 停用集群



步骤 3 等待集群列表页面的“Operations”列出现“Delete”或“Enable”时，单击“Delete”删除集群。也可以单击“Enable”启用集群。

图16-4 启用或删除集群



----结束

16.4 Kafka 集群监控管理

Kafka 集群监控管理包含以下内容：

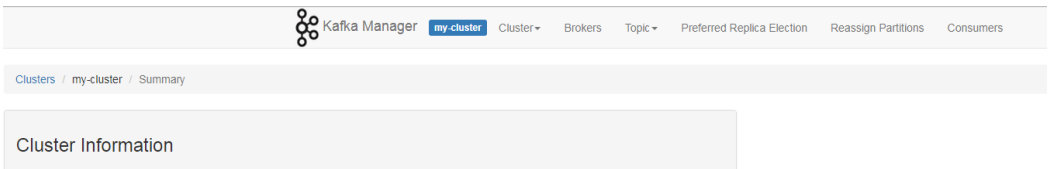
- 查看 Broker 信息
- 查看 Topic 信息
- 查看 Consumers 信息
- 通过 KafkaManager 修改 Topic 的 partition

查看 Broker 信息

步骤 1 登录 KafkaManager 的 WebUI 界面。

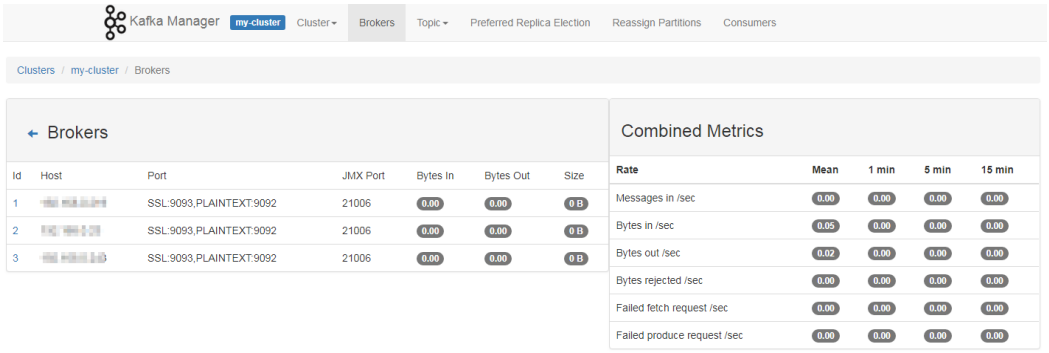
步骤 2 在集群列表页面单击对应集群名称进入集群 Summary 页面。

图16-5 集群 Summary 页面



步骤 3 单击“Brokers”进入 Broker 监控页面，该页面包括 Broker 列表和 Broker 节点的 IO 统计信息。

图16-6 Broker 监控页面



----结束

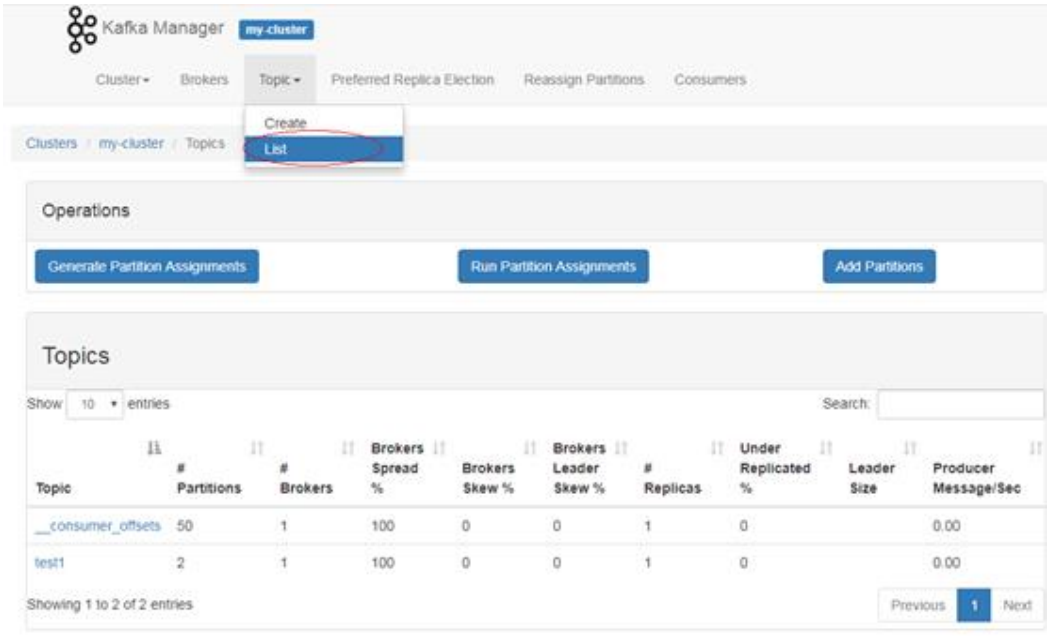
查看 Topic 信息

步骤 1　登录 KafkaManager 的 WebUI 界面。

步骤 2　在集群列表页面单击对应集群名称进入集群 Summary 页面。

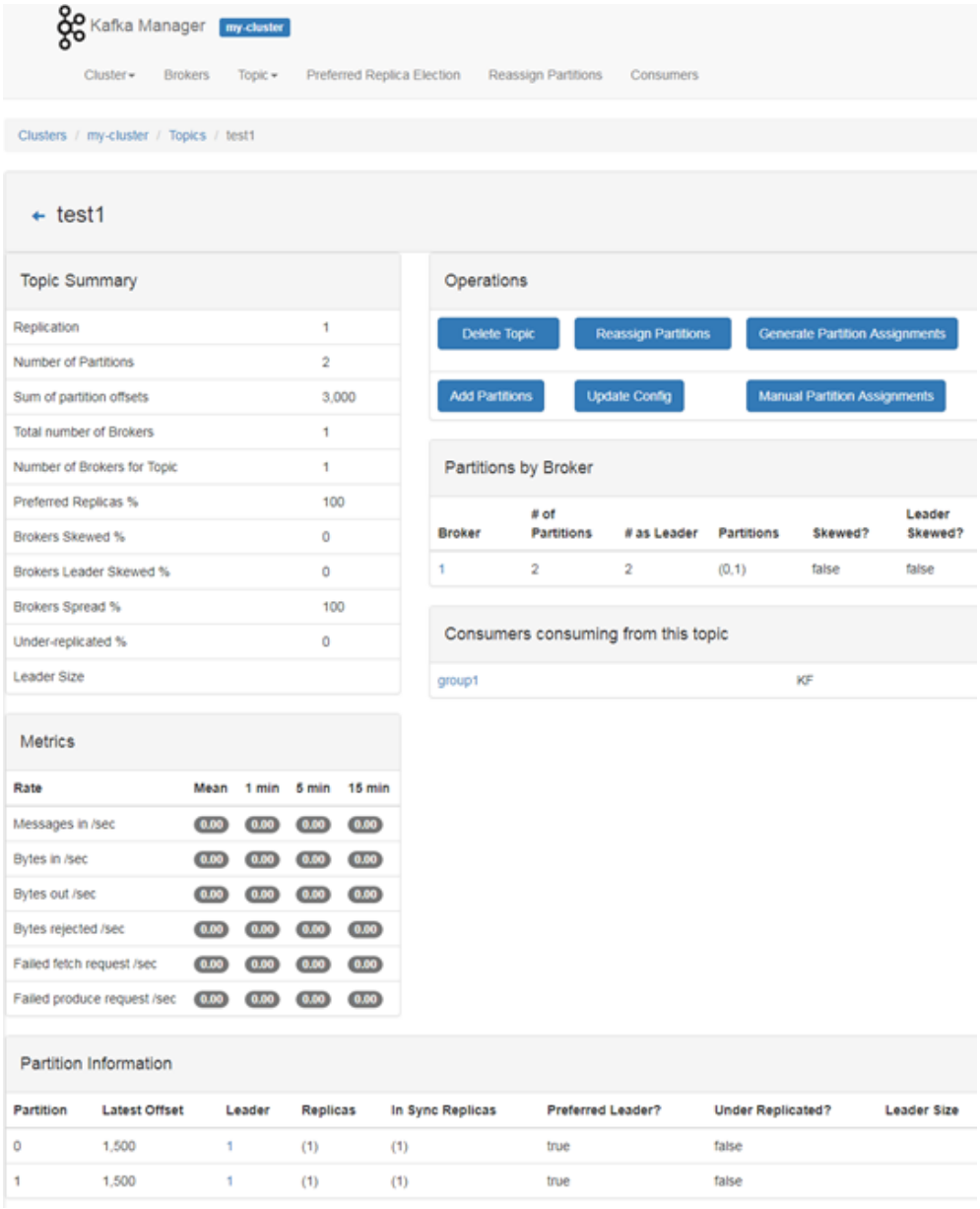
步骤 3　单击“Topic > List”查看当前集群的 Topic 列表及每个 Topic 的相关信息。

图16-7 Topic 列表



步骤 4　单击具体的 Topic 名称查看该 Topic 的详细信息。

图16-8 Topic 的详细信息



----结束

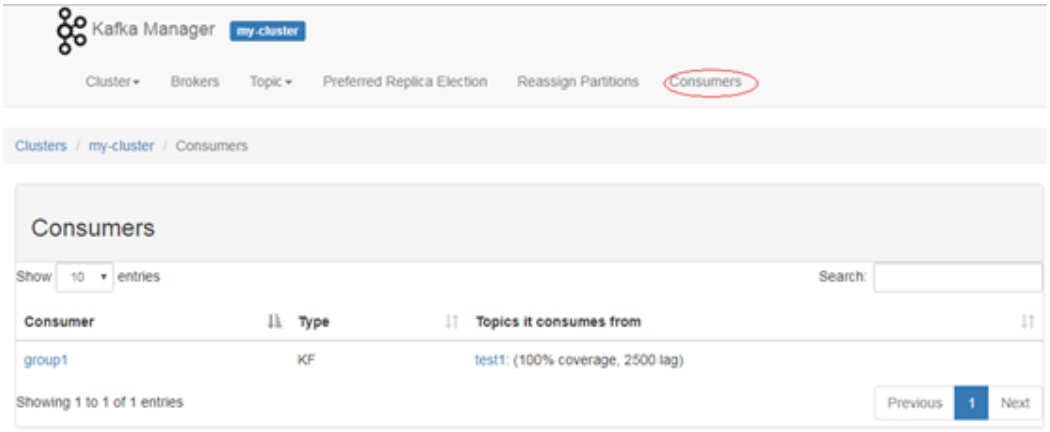
查看 Consumers 信息

步骤 1 登录 KafkaManager 的 WebUI 界面。

步骤 2 在集群列表页面单击对应集群名称进入集群 Summary 页面。

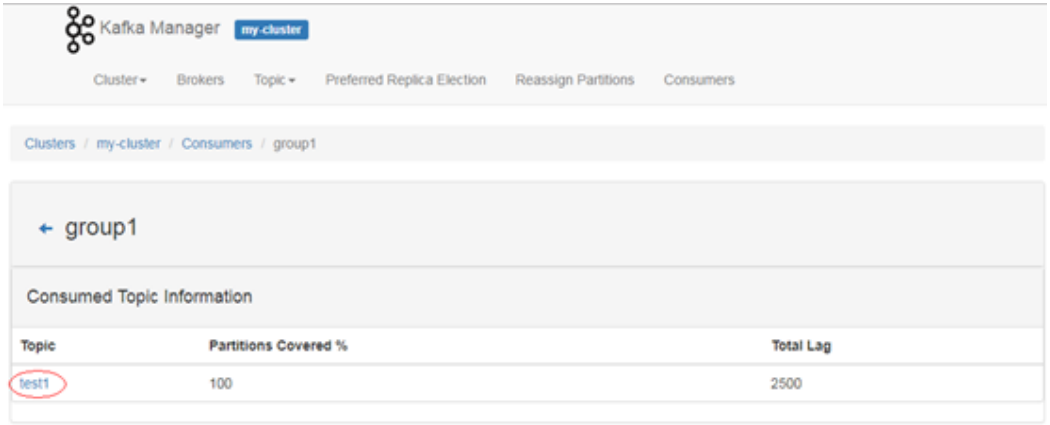
步骤 3 单击“Consumers”查看当前集群的 Consumers 列表及每个 Consumer 的消费信息。

图16-9 Consumers 列表



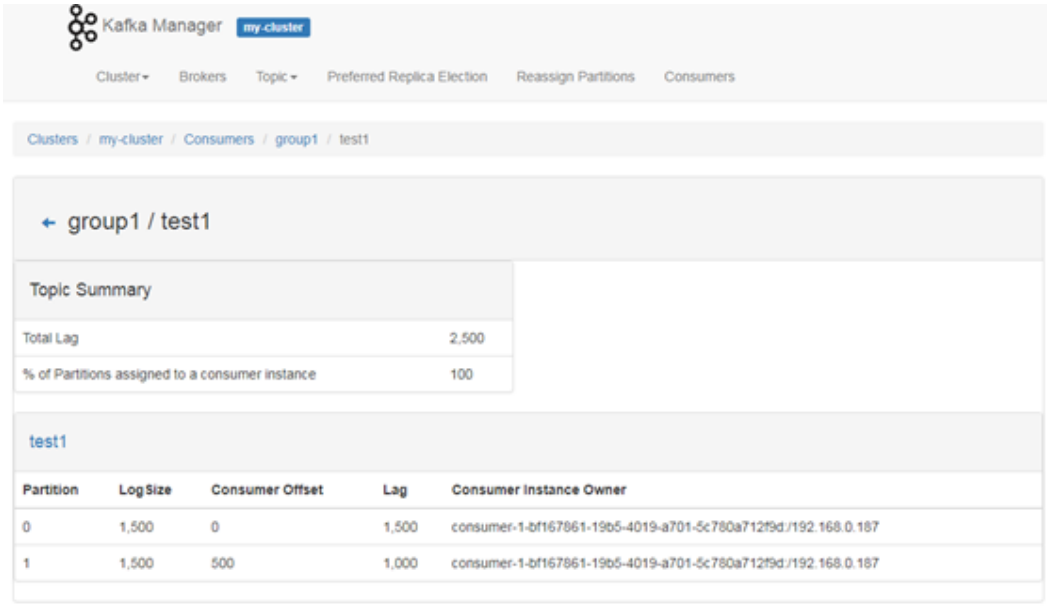
步骤 4 单击 Consumer 的名称查看消费的 Topic 列表。

图16-10 Consumer 消费的 Topic 列表



步骤 5 单击 Consumer 下 Topic 列表中的 Topic 名称，查看该 Consumer 对 Topic 的具体消费情况。

图16-11 Consumer 对 Topic 的具体消费情况



----结束

通过 KafkaManager 修改 Topic 的 partition

步骤 1 登录 KafkaManager 的 WebUI 界面。

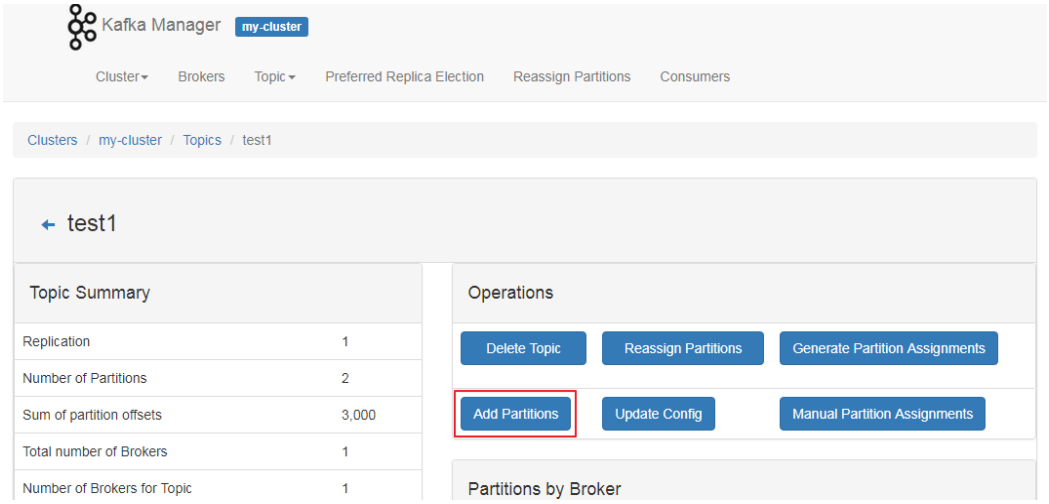
步骤 2 在集群列表页面单击对应集群名称进入集群 Summary 页面。

步骤 3 单击“Topic > List”进入当前集群的 Topic 列表页面。

步骤 4 单击具体的 Topic 名称进入 Topic Summary 页面。

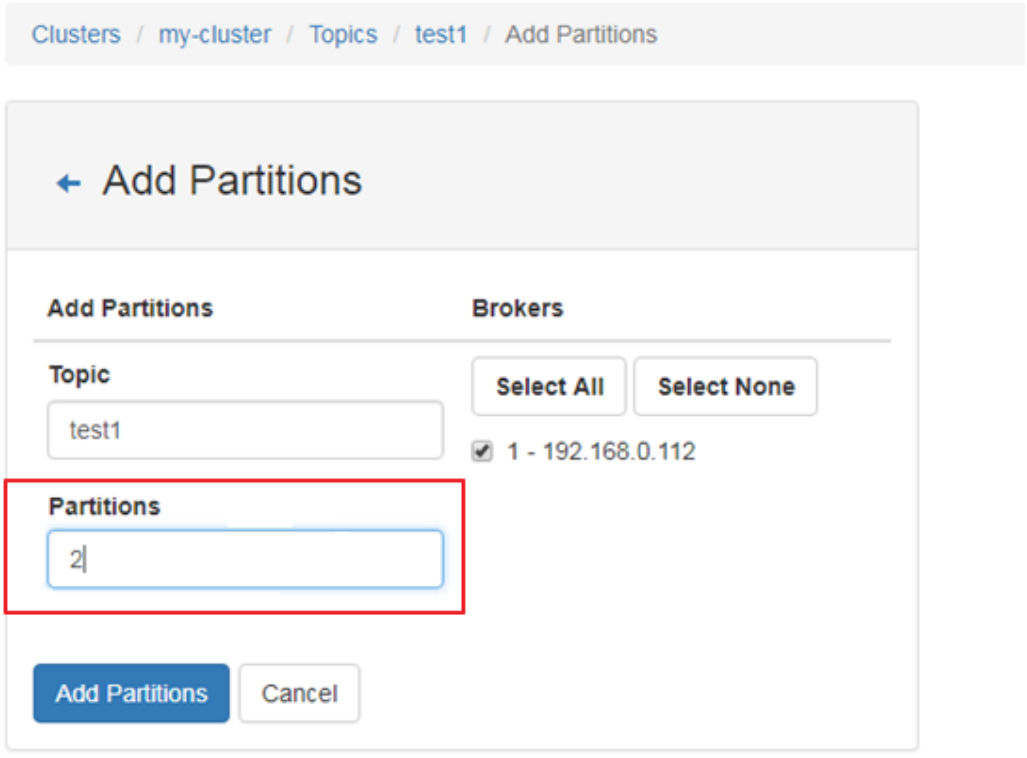
步骤 5 单击“add partitions”，进入添加分区页面。

图16-12 添加分区



步骤 6　确认 Topic 名称并修改“Partitions”数量，单击“Add Partitions”进行分区添加。

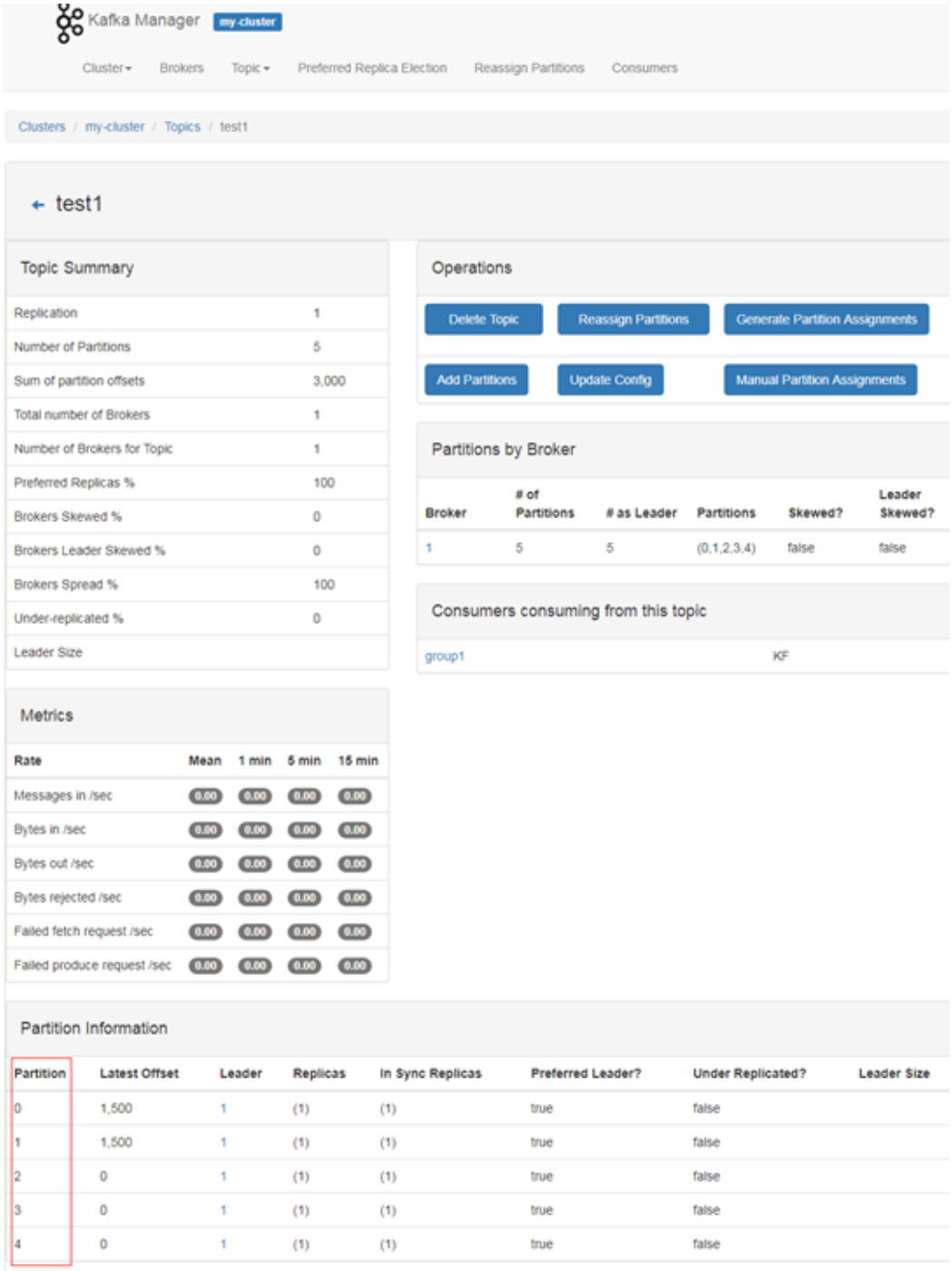
图16-13 修改 Partitions 数量



步骤 7　分区添加成功后，单击“Go to topic view.”返回 Topic Summary 页面。

步骤 8　在 Topic Summary 页面的下方“Partition Information”中确认 partition 数量。

图16-14 Partition Information



步骤 9 （可选）若对分配的分区不满意，可以执行 Partition 的重新分配功能来重新自动分配分区。

1. 在 Topic Summary 页面单击“Generate Partition Assignments”。
2. 勾选 broker 实例，单击“Generate Partition Assignments”生成分区。
3. 分区生成完成，单击“Go to topic view.”返回 Topic Summary 页面。

4. 在 Topic Summary 页面单击“Reassign Partitions”可以在集群的 broker 实例上重新自动分配分区。
5. 单击“Go to reassign partitions.”查看重新分配的分区详情。

步骤 10 （可选）若对自动分配的分区不满意，可以执行手动分配来重新分配分区。

1. 在 Topic Summary 页面单击“Manual Partition Assignments”进入手动分配分区页面。
2. 手动为每个分区的副本分配 Broker id，然后单击“Save Partition Assignment”保存修改。
3. 单击“Go to topic view.”返回 Topic Summary 页面，查看分区详情。

----结束

17 使用 Loader

17.1 从零开始使用 Loader

用户可以使用 Loader 将数据从 SFTP 服务器导入到 HDFS。

本章节适用于 MRS 3.x 之前版本。

前提条件

- 已准备业务数据。
- 已创建分析集群。

操作步骤

步骤 1 访问 Loader 页面。

1. 登录集群详情页面，选择“服务管理”。
2. 选择“Hue”，在“Hue 概述”的“Hue WebUI”，单击“Hue (主)”，打开 Hue 的 WebUI。
3. 选择“Data Browsers > Sqoop”。

   默认显示 Loader 页面中的作业管理界面。

步骤 2 在 Loader 页面，单击“管理连接”。

步骤 3 单击“新建连接”，参考文件服务器连接，创建 sftp-connector。

步骤 4 单击“新建连接”，输入连接名称，选择连接器为 hdfs-connector，创建 hdfs-connector。

步骤 5 访问 Loader 页面，单击“管理作业”。

步骤 6 单击“新建作业”。

步骤 7 在“基本信息”填写参数。

1. 在“名称”填写一个作业的名称。
2. 选择步骤 3 创建的“源连接”和步骤 4 创建的“目的连接”。

步骤 8 在“自”填写源连接的作业配置。

具体请参见 ftp-connector 或 sftp-connector。

步骤 9 在“至”填写目的连接的作业配置。

具体请参见 hdfs-connector。

步骤 10 在“任务配置”填写作业的运行参数。

表17-1 Loader 作业运行属性

参数	说明
抽取并发数	设置 map 任务的个数。
加载(写入)并发数	设置 reduce 任务的个数。 该参数只有在目的字段为 Hbase 和 Hive 时才会显示。
单个分片的最大错误记录数	设置一个错误阈值，如果单个 map 任务的错误记录超过设置阈值则任务自动结束，已经获取的数据不回退。 说明 “generic-jdbc-connector”的“MYSQL”和“MPPDB”默认批量读写数据，每一批次数据最多只记录一次错误记录。
脏数据目录	设置一个脏数据目录，在出现脏数据的场景中在该目录保存脏数据。如果不设置则不保存。

步骤 11 单击“保存”。

----结束

17.2 Loader 使用简介

本章节适用于 MRS 3.x 之前版本。

使用流程

通过 Loader 迁移用户数据时，基本流程如下所示。

1. 访问 Hue WebUI 的 Loader 页面。
2. 管理 Loader 连接。
3. 创建作业，选择数据源的连接以及保存数据的连接。
4. 运行作业，完成数据迁移。

Loader 页面介绍

Loader 页面是基于开源 Sqoop WebUI 的图形化数据迁移管理工具，该页面托管在 Hue 的 WebUI 中。进入 Loader 页面请执行以下操作：

1. 访问 Hue WebUI，参见访问 Hue 的 WebUI。
2. 选择“Data Browsers > Sqoop”。

   默认显示 Loader 页面中的作业管理界面。

Loader 连接介绍

Loader 连接保存了数据具体位置的相关信息，Loader 使用连接来访问数据，或将数据保存到指定的位置。进入 Loader 连接管理页面请执行以下操作：

1. 进入 Loader 页面。
2. 单击“管理连接”。

   显示 Loader 连接管理页面。

   可单击“管理作业”回到作业管理页面。
3. 单击“新建连接”，进入配置页面，并填写参数创建一个 Loader 连接。

Loader 作业介绍

Loader 作业用于管理数据迁移任务，每个作业包含一个源数据的连接，和一个目的数据的连接，通过从源连接读取数据，再将数据保存到目的连接，完成数据迁移任务。

17.3 Loader 连接配置说明

本章节适用于 MRS 3.x 之前版本。

基本介绍

Loader 支持以下多种连接，每种连接的配置介绍可根据本章节内容了解。

- obs-connector
- generic-jdbc-connector
- ftp-connector 或 sftp-connector
- hbase-connector、hdfs-connector 或 hive-connector

OBS 连接

OBS 连接是 Loader 与 OBS 进行数据交换的通道，配置参数如表 17-2 所示。

表17-2 obs-connector 配置

参数	说明
名称	指定一个 Loader 连接的名称。
OBS 服务器	输入 OBS endpoint 地址，一般格式为 **OBS.***Region***.***DomainName*。 例如执行如下命令查看 OBS endpoint 地址：

参数	说明
	cat /opt/Bigdata/apache-tomcat-7.0.78/webapps/web/WEB-INF/classes/cloud-obs.properties
端口	访问 OBS 数据的端口。默认值为“443”。
访问标识(AK)	表示访问 OBS 的用户的访问密钥 AK。
密钥(SK)	表示访问密钥对应的 SK。

关系型数据库连接

关系型数据库连接是 Loader 与关系型数据库进行数据交换的通道，配置参数如表 17-3 所示。

说明

部分参数需要单击“显示高级属性”后展开，否则默认隐藏。

表17-3 generic-jdbc-connector 配置

参数	说明
名称	指定一个 Loader 连接的名称。
数据库类型	表示 Loader 连接支持的数据，可以选择“ORACLE”、“MYSQL”和“MPPDB”。
数据库服务器	表示数据库的访问地址，可以是 IP 地址或者域名。
端口	表示数据库的访问端口。
数据库名称	表示保存数据的具体数据库名。
用户名	表示连接数据库使用的用户名称。
密码	表示此用户对应的密码。需要与实际密码保持一致。

表17-4 高级属性配置

参数	说明
一次请求行数	表示每次连接数据库时，最多可获取的数据量。
连接属性	不同类型数据库支持该数据库连接特有的驱动属性，例如 MYSQL 的“autoReconnect”。如果需要定义驱动属性，单击“添加”。
引用符号	表示数据库的 SQL 中保留关键字的定界符，不同类型数据库定义的定界符不完全相同。

文件服务器连接

文件服务器连接包含 FTP 连接和 SFTP 连接，是 Loader 与文件服务器进行数据交换的通道，配置参数如表 17-5 所示。

表17-5 ftp-connector 或 sftp-connector 配置

参数	说明
名称	指定一个 Loader 连接的名称。
主机名或 IP	输入文件服务器的访问地址，可以是服务器的主机名或者 IP 地址。
端口	访问文件服务器的端口。 • FTP 协议请使用端口“21”。 • SFTP 协议请使用端口“22”。
用户名	表示文件服务器的用户名称。
密码	表示此用户对应的密码。

MRS 集群连接

MRS 集群连接包含 HBase 连接、HDFS 连接和 Hive 连接，是 Loader 与对应各数据进行数据交换的通道。

配置 MRS 集群连接时，需要设置名称、选择对应的连接器“hbase-connector”、“hdfs-connector”或“hive-connector”，然后保存即可。

17.4 管理 Loader 连接（MRS 3.x 之前版本）

操作场景

Loader 页面支持创建、查看、编辑和删除连接。

本章节适用于 MRS 3.x 之前版本。

前提条件

已访问 Loader 页面，参见 Loader 页面介绍。

创建连接

步骤 1 在 Loader 页面，单击“管理连接”。

步骤 2 单击“新建连接”，配置连接参数。

参数介绍具体可参见 Loader 连接配置说明。

步骤 3 单击“保存”。

如果连接配置，例如 IP 地址、端口、访问用户等信息不正确，将导致验证连接失败无法保存。另外 VPC 相关设置，也可能影响网络连通性。

说明

用户可以直接单击“测试”立即检测连接是否可用。

----结束

查看连接

步骤 1 在 Loader 页面，单击“管理连接”。

- 如果集群启用了 Kerberos 认证，则默认显示所有当前用户创建的连接，不支持显示其他用户创建的连接。
- 如果集群未启用 Kerberos 认证，则显示集群中全部的 Loader 连接。

步骤 2 在“Sqoop 连接”中输入指定连接的名称，可以筛选该连接。

----结束

编辑连接

步骤 1 在 Loader 页面，单击“管理连接”。

步骤 2 单击指定连接的名称，进入编辑页面。

步骤 3 根据业务需要，修改连接配置参数。

步骤 4 单击“测试”。

如果显示测试成功，则执行步骤 5；如果显示不能连接至 OBS Server，则需要重复步骤 3。

步骤 5 单击“保存”。

如果某个 Loader 作业已集成一个 Loader 连接，那么编辑连接参数后可能导致 Loader 作业运行效果也产生变化。

----结束

删除连接

步骤 1 在 Loader 页面，单击“管理连接”。

步骤 2 在指定连接所在行，单击“删除”。

步骤 3 在弹出的对话框窗口，单击“是，将其删除”。

如果某个 Loader 作业已集成一个 Loader 连接，那么该连接不可以被删除。

----结束

17.5 Loader 作业源连接配置说明

基本介绍

Loader 作业需要从不同数据源获取数据时，应该选择对应类型的连接，每种连接在该场景中需要配置连接的属性。

本章节适用于 MRS 3.x 之前版本。

obs-connector

表17-6 obs-connector 数据源连接属性

参数	说明
桶名	保存源数据的 OBS 文件系统。
源目录或文件	源数据实际存储的形态，可能是文件系统包含一个目录中的全部数据文件，或者是文件系统包含的单个数据文件。
文件格式	Loader 支持 OBS 中存储数据的文件格式，默认支持以下两种： • CSV_FILE：表示文本格式文件。目的连接为数据库型连接时，只支持文本格式。 • BINARY_FILE：表示文本格式以外的二进制文件。
换行符	源数据的每行结束标识字符。
字段分割符	源数据的每个字段分割标识字符。
编码类型	源数据的文本编码类型。只对文本类型文件有效。
文件分割方式	支持以下两种： • File：按总文件个数分配 map 任务处理的文件数量，计算规则为“文件总个数/抽取并发数”。 • Size：按文件总大小分配 map 任务处理的文件大小，计算规则为“文件总大小/抽取并发数”。

generic-jdbc-connector

表17-7 generic-jdbc-connector 数据源连接属性

参数	说明

参数	说明
模式或表空间	表示源数据对应的数据库名称，支持通过界面查询并选择。
表名	存储源数据的数据表，支持通过界面查询并选择。
抽取分区字段	分区字段，如果需读取多个字段，使用该字段分割结果并获取数据。
Where 子句	表示读取数据库时使用的查询语句。

ftp-connector 或 sftp-connector

表17-8 ftp-connector 或 sftp-connector 数据源连接属性

参数	说明
源目录或文件	源数据实际存储的形态，可能是文件服务器包含一个目录中的全部数据文件，或者是单个数据文件。
文件格式	Loader 支持文件服务器中存储数据的文件格式，默认支持以下两种： • CSV_FILE：表示文本格式文件。目的连接为数据库型连接时，只支持文本格式。 • BINARY_FILE：表示文本格式以外的二进制文件。
换行符	源数据的每行结束标识字符。 说明 ftp 或 sftp 作为源连接时，当“文件格式”配置为 BINARY_FILE 时，高级属性中的“换行符”配置无效
字段分割符	源数据的每个字段分割标识字符。 说明 ftp 或 sftp 作为源连接时，当“文件格式”配置为 BINARY_FILE 时，高级属性中的“字段分割符”配置无效
编码类型	源数据的文本编码类型。只对文本类型文件有效。
文件分割方式	支持以下两种： • File：按总文件个数分配 map 任务处理的文件数量，计算规则为“文件总个数/抽取并发数”。 • Size：按文件总大小分配 map 任务处理的文件大小，计算规则为“文件总大小/抽取并发数”。

hbase-connector

表17-9 hbase-connector 数据源连接属性

参数	说明
表名	源数据实际存储的 HBase 表。

hdfs-connector

表17-10 hdfs-connector 数据源连接属性

参数	说明
源目录或文件	源数据实际存储的形态，可能是 HDFS 包含一个目录中的全部数据文件，或者是单个数据文件。
文件格式	Loader 支持 HDFS 中存储数据的文件格式，默认支持以下两种： • CSV_FILE：表示文本格式文件。目的连接为数据库型连接时，只支持文本格式。 • BINARY_FILE：表示文本格式以外的二进制文件。
换行符	源数据的每行结束标识字符。 说明 hdfs 作为源连接时，当“文件格式”配置为 BINARY_FILE 时，高级属性中的“换行符”配置无效。
字段分割符	源数据的每个字段分割标识字符。 说明 hdfs 作为源连接时，当“文件格式”配置为 BINARY_FILE 时，高级属性中的“字段分割符”配置无效。
文件分割方式	支持以下两种： • File：按总文件个数分配 map 任务处理的文件数量，计算规则为“文件总个数/抽取并发数”。 • Size：按文件总大小分配 map 任务处理的文件大小，计算规则为“文件总大小/抽取并发数”。

hive-connector

表17-11 hive-connector 数据源连接属性

参数	说明

参数	说明
数据库名称	数据源的 Hive 数据库名称，支持通过界面查询并选择。
表名	数据源的 Hive 表名称，支持通过界面查询并选择。

17.6 Loader 作业目的连接配置说明

基本介绍

Loader 作业需要将数据保存到不同目的存储位置时，应该选择对应类型的目的连接，每种连接在该场景中需要配置连接的属性。

obs-connector

表17-12 obs-connector 目的连接属性

参数	说明
桶名	保存最终数据的 OBS 文件系统。
写入目录	最终数据在文件系统保存时的具体目录。必须指定一个目录。
文件格式	Loader 支持 OBS 中存储数据的文件格式，默认支持以下两种： • CSV_FILE：表示文本格式文件。目的连接为数据库型连接时，只支持文本格式。 • BINARY_FILE：表示文本格式以外的二进制文件。
换行符	最终数据的每行结束标识字符。
字段分割符	最终数据的每个字段分割标识字符。
编码类型	最终数据的文本编码类型。只对文本类型文件有效。

generic-jdbc-connector

表17-13 generic-jdbc-connector 目的连接属性

参数	说明
模式名称	保存最终数据的数据库名称。
表名	保存最终数据的数据表名称。

ftp-connector 或 sftp-connector

表17-14 ftp-connector 或 sftp-connector 目的连接属性

参数	说明
写入目录	最终数据在文件服务器保存时的具体目录。必须指定一个目录。
文件格式	Loader 支持文件服务器中存储数据的文件格式，默认支持以下两种： • CSV_FILE：表示文本格式文件。目的连接为数据库型连接时，只支持文本格式。 • BINARY_FILE：表示文本格式以外的二进制文件。
换行符	最终数据的每行结束标识字符。 说明 ftp 或 sftp 作为目的连接时，当“文件格式”配置为 BINARY_FILE 时，高级属性中的“换行符”配置无效。
字段分割符	最终数据的每个字段分割标识字符。 说明 ftp 或 sftp 作为目的连接时，当“文件格式”配置为 BINARY_FILE 时，高级属性中的“字段分割符”配置无效
编码类型	最终数据的文本编码类型。只对文本类型文件有效。

hbase-connector

表17-15 hbase-connector 目的连接属性

参数	说明
表名	保存最终数据的 HBase 表名称，支持通过界面查询并选择。
导入方式	支持 BULKLOAD、PUTLIST 两种方式导入数据到 HBase 表。
导入前清空数据	标识是否需要清空目标 HBase 表中的数据，支持以下两种类型： • True：清空表中的数据。 • False：不清空表中的数据，选择 False 时如果表中存在数据，则作业运行会报错。

hdfs-connector

表17-16 hdfs-connector 目的连接属性

参数	说明
写入目录	最终数据在 HDFS 保存时的具体目录。必须指定一个目录。
文件格式	Loader 支持 HDFS 中存储数据的文件格式，默认支持以下两种： • CSV_FILE：表示文本格式文件。目的连接为数据库型连接时，只支持文本格式。 • BINARY_FILE：表示文本格式以外的二进制文件。
压缩格式	文件在 HDFS 保存时的压缩行为。支持 NONE、DEFLATE、GZIP、BZIP2、LZ4 和 SNAPPY。
是否覆盖	文件在导入 HDFS 时对写入目录中原有文件的处理行为，支持以下两种： • True：默认清空目录中的文件并导入新文件。 • False：不清空文件。如果写入目录中有文件，则作业运行失败。
换行符	最终数据的每行结束标识字符。 说明 hdfs 作为目的连接时，当“文件格式”配置为 BINARY_FILE 时，高级属性中的“换行符”配置无效。
字段分割符	最终数据的每个字段分割标识字符。 说明 hdfs 作为目的连接时，当“文件格式”配置为 BINARY_FILE 时，高级属性中的“字段分割符”配置无效

hive-connector

表17-17 hive-connector 目的连接属性

参数	说明
数据库名称	保存最终数据的 Hive 数据库名称，支持通过界面查询并选择。
表名	保存最终数据的 Hive 表名称，支持通过界面查询并选择。

17.7 管理 Loader 作业

操作场景

Loader 页面支持创建、查看、编辑和删除作业。

本章节适用于 MRS 3.x 之前版本。

前提条件

已访问 Loader 页面，参见 Loader 页面介绍。

创建作业

步骤 1 访问 Loader 页面，单击“新建作业”。

步骤 2 在“基本信息”填写参数。

1. 在“名称”填写一个作业的名称。
2. 在“源连接”和“目的连接”选择对应的连接。
   选择某个类型的连接，表示从指定的源获取数据，并保存到目的位置。

   说明

   如果没有需要的连接，可单击“添加新连接”。

步骤 3 在“自”填写源连接的作业配置。

具体请参见 Loader 作业源连接配置说明。

步骤 4 在“至”填写目的连接的作业配置。

具体请参见 Loader 作业目的连接配置说明。

步骤 5 在“目的连接”是否选择了数据库类型的连接？

数据库类型的连接包含以下几种：

- generic-jdbc-connector
- hbase-connector
- hive-connector

“目的连接”选择数据库类型的连接时，还需要配置业务数据与数据库表字段的对应关系：

- 是，请执行步骤 6。
- 否，请执行步骤 7。

步骤 6 在“字段映射”填写字段对应关系。然后执行步骤 7。

“字段映射”的对应关系，表示用户数据中每一列与数据库的表字段的匹配关系。

表17-18“字段映射”属性

参数	说明
列号	表示业务数据的字段顺序。
样本	表示业务数据的第一行值样例。
列族	“目的连接”为 hbase-connector 类型时，支持定义保存数据的具体列族。
目的字段	配置保存数据的具体字段。
类型	显示用户选择字段的类型。
行键	“目的连接”为 hbase-connector 类型时，需要勾选作为行键的“目的字段”。

说明

如果 From 是 sftp/ftp/obs/hdfs 等文件类型连接器，Field Mapping 样值取自文件第一行数据，需要保证第一行数据是完整的，Loader 作业不会抽取没有 Mapping 上的列。

步骤 7 在“任务配置”填写作业的运行参数。

表17-19 Loader 作业运行属性

参数	说明
抽取并发数	设置 map 任务的个数。
加载(写入)并发数	设置 reduce 任务的个数。 该参数只有在目的字段为 Hbase 和 Hive 时才会显示。
单个分片的最大错误记录数	设置一个错误阈值，如果单个 map 任务的错误记录超过设置阈值则任务自动结束，已经获取的数据不回退。 说明 “generic-jdbc-connector”的“MYSQL”和“MPPDB”默认批量读写数据，每一批次数据最多只记录一次错误记录。
脏数据目录	设置一个脏数据目录，在出现脏数据的场景中在该目录保存脏数据。如果不设置则不保存。

步骤 8 单击“保存”。

----结束

查看作业

步骤 1 访问 Loader 页面，默认显示 Loader 作业管理页面。

- 如果集群启用了 Kerberos 认证，则默认显示所有当前用户创建的作业，不支持显示其他用户的作业。
- 如果集群未启用 Kerberos 认证，则显示集群中全部的作业。

步骤 2 在“Sqoop 作业”中输入指定作业的名称或连接类型，可以筛选该作业。

步骤 3 单击“刷新列表”，可以获取作业的最新状态。

----结束

编辑作业

步骤 1 访问 Loader 页面，默认显示 Loader 作业管理页面。

步骤 2 单击指定作业的名称，进入编辑页面。

步骤 3 根据业务需要，修改作业配置参数。

步骤 4 单击“保存”。

说明

左侧导航栏支持作业的基本操作，包含“运行”、“复制”、“删除”、“激活”、“历史记录”和“显示作业 JSON 定义”。

----结束

删除作业

步骤 1 访问 Loader 页面。

步骤 2 在指定作业所在行，单击。

您还可以勾选一个或多个作业，单击作业列表右上方的“删除作业”。

步骤 3 在弹出的对话框窗口，单击“是，将其删除”。

如果某个 Loader 作业正处于“运行中”的状态，则无法删除作业。

----结束

17.8 准备 MySQL 数据库连接的驱动

操作场景

Loader 作为批量数据导出的组件，可以通过关系型数据库导入、导出数据。

前提条件

已准备业务数据。

操作步骤

MRS 3.x 之前版本：

步骤 1 从 MySQL 官网下载 MySQL jdbc 驱动程序“mysql-connector-java-5.1.21.jar”，具体 MySQL jdbc 驱动程序选择参见下表。

表17-20 版本信息

jdbc 驱动程序版本	MySQL 版本
Connector/J 5.1	MySQL 4.1、MySQL 5.0、MySQL 5.1、MySQL 6.0 alpha
Connector/J 5.0	MySQL 4.1、MySQL 5.0 servers、distributed transaction (XA)
Connector/J 3.1	MySQL 4.1、MySQL 5.0 servers、MySQL 5.0 except distributed transaction (XA)
Connector/J 3.0	MySQL 3.x、MySQL 4.1

步骤 2 将“mysql-connector-java-5.1.21.jar”上传至 MRS master 主备节点 loader 安装目录

- 针对 MRS 3.x 之前版本，上传至“/opt/Bigdata/MRS_XXX/install/FusionInsight-Sqoop-1.99.7/FusionInsight-Sqoop-1.99.7/server/jdbc/”

  其中“XXX”为 MRS 版本号，请根据实际情况修改。

步骤 3 修改“mysql-connector-java-5.1.21.jar”包属主为“omm:wheel”。

步骤 4 修改配置文件“jdbc.properties”。

将“MYSQL”的键值修改为上传的 jdbc 驱动包名“mysql-connector-java-5.1.21.jar”，例如：MYSQL=mysql-connector-java-5.1.21.jar。

步骤 5 重启 Loader 服务。

----结束

MRS 3.x 及之后版本：

修改关系型数据库对应的驱动 jar 包文件权限。

步骤 1 登录 Loader 服务的主备管理节点，获取关系型数据库对应的驱动 jar 包保存在 Loader 服务主备节点的 lib 路径：“${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib”。

说明

此处版本号 8.1.0.1 为示例，具体以实际环境的版本号为准。

步骤 2 使用 **root** 用户在 Loader 服务主备节点分别执行以下命令修改权限：

cd ${BIGDATA_HOME}/FusionInsight_Porter_8.1.0.1/install/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib

chown omm:wheel *jar 包文件名*

chmod 600 *jar 包文件名*

步骤 3 登录 FusionInsight Manager 系统，选择“集群 > *待操作集群名称* > 服务 > Loader > 更多 > 重启服务”输入管理员密码重启 Loader 服务。

----结束

17.9 Loader 日志介绍

日志描述

日志存储路径：Loader 相关日志的默认存储路径为“/var/log/Bigdata/loader/日志分类”。

- runlog：“/var/log/Bigdata/loader/runlog”（运行日志）
- scriptlog：“/var/log/Bigdata/loader/scriptlog/”（脚本的执行日志）
- catalina：“/var/log/Bigdata/loader/catalina”（tomcat 的启停日志）
- audit： “/var/log/Bigdata/loader/audit”（审计日志）

日志归档规则：

Loader 的运行日志和审计日志，启动了自动压缩归档功能，默认情况下，当日志大小超过 10MB 的时候，会自动压缩，压缩后的日志文件名规则为：“<原有日志名>-<yyyy-mm-dd_hh-mm-ss>.[编号].log.zip”。最多保留最近的 20 个压缩文件，压缩文件保留个数可以在 Manager 界面中配置。

表17-21 Loader 日志列表

日志类型	日志文件名	描述
运行日志	loader.log	Loader 运行日志，记录 Loader 系统运行时候所产生的大部分日志。
	loader-omm-***-pid***-gc.log.*.current	Loader 进程 gc 日志
	sqoopInstanceCheck.log	Loader 实例健康检查日志
审计日志	default.audit	Loader 操作审计日志（例如：作业的增删改查、用户的登录）。
tomcat 日志	catalina.out	tomcat 的运行日志
	catalina. <yyyy-mm-dd >.log	tomcat 的运行日志

日志类型	日志文件名	描述
	host-manager. <yyyy-mm-dd >.log	tomcat 的运行日志
	localhost_access_log. <yyyy-mm-dd >.txt	tomcat 的运行日志
	manager <yyyy-mm-dd >.log	tomcat 的运行日志
	localhost. <yyyy-mm-dd >.log	tomcat 的运行日志
脚本日志	postInstall.log	Loader 安装脚本日志。 执行 loader 安装脚本（postInstall.sh）时产生的日志。
	preStart.log	Loader 服务的预启动脚本日志。Loader 服务启动时，需要先执行一系列的准备操作（preStart.sh），例如生成 keytab 文件等，该日志正是记录了这些操作信息。
	loader_ctl.log	Loader 执行服务启停脚本（sqoop.sh）的日志。

日志级别

Loader 中提供了如表 17-22 所示的日志级别，日志级别优先级从高到低分别是 ERROR、WARN、INFO、DEBUG，程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。

表17-22 日志级别

级别	描述
ERROR	ERROR 表示错误日志输出。
WARN	WARN 表示当前事件处理存在异常信息。
INFO	INFO 表示系统及各事件正常运行状态信息。
DEBUG	DEBUG 表示系统及系统调试信息。

如果您需要修改日志级别，请执行如下操作：

步骤 1 请参考修改集群服务配置参数，进入 Loader 的“全部配置”页面。

步骤 2 左边菜单栏中选择所需修改的角色所对应的日志菜单。

步骤 3 选择所需修改的日志级别。

步骤 4 保存配置，在弹出窗口中单击“确定”使配置生效。

说明

配置完成后即生效，不需要重启服务。

----结束

日志格式

Loader 的日志格式如下所示：

表17-23 日志格式

日志类型	格式	示例
运行日志	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<产生该日志的线程名字>|<log 中的 message>|<日志事件的发生位置>	2015-06-29 14:54:35,553	INFO	[localhost-startStop-1]	ConnectionRequestHandler initialized	org.apache.sqoop.handler.ConnectionRequestHandler.<init>(ConnectionRequestHandler.java:100)
审计日志	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|default|<log 中的 message>|<日志事件的发生位置>	2015-06-29 15:35:40,969 INFO default: UserName=admin, UserIP=10.52.0.111, Time=2015-06-29 15:35:40,969, Operation=submit, Resource=submission@21, Result=Failure, Detail={[reason:GET_SFTP_SESSION_FAILED:Failed to get sftp session - 10.162.0.35 (caused by: Auth cancel)];[config:null]}

17.10 样例：通过 Loader 将数据从 OBS 导入 HDFS

操作场景

用户需要将大量数据从集群外导入集群内的时候，可以选择从 OBS 导入到 HDFS 的方式。

前提条件

- 已准备业务数据。
- 已创建分析集群。

操作步骤

步骤 1 将业务数据上传到用户的 OBS 文件系统。

步骤 2 获取用户的 AK/SK 信息，然后创建一个 OBS 连接和一个 HDFS 连接。

具体可参见 Loader 连接配置说明。

步骤 3 访问 Loader 页面。

如果是启用了 Kerberos 认证的分析集群，可参见访问 Hue 的 WebUI。

步骤 4 单击“新建作业”。

步骤 5 在“基本信息”填写参数。

1. 在“名称”填写一个作业的名称。例如“obs2hdfs”。
2. 在“源连接”选择已创建的 OBS 连接。
3. “目的连接”选择已创建的 HDFS 连接。

步骤 6 在“自”填写源连接参数。

1. 在“桶名”填写业务数据所保存的 OBS 文件系统名称。
2. 在“源目录或文件”填写业务数据在文件系统的具体位置。

   如果是单个文件，需要填写包含文件名的完整路径。如果是目录，填写目录的完整路径
3. “文件格式”填写业务数据文件的类型。

可参见 obs-connector。

步骤 7 在“至”填写目的连接参数。

1. 在“定入目录”填写业务数据在 HDFS 要保存的目录名称。

   如果是启用 Kerberos 认证的集群，当前访问 Loader 的用户对保存数据的目录需要有写入权限。
2. 在“文件格式”填写业务数据文件的类型。

   需要与步骤 6.3 的类型对应。
3. 在“压缩格式”填写一种压缩的算法。例如选择不压缩“NONE”。

4. 在“是否覆盖”选择已有文件的处理方式，选择“True”。
5. 单击“显示高级属性”，在“换行符”填写业务数据保存时，系统填充的换行字符。
6. 在“字段分割符”填写业务数据保存时，系统填充的分割字符。

可参见 hdfs-connector。

步骤 8 在“任务配置”填写作业的运行参数。

1. 在“抽取并发数”填写 map 任务的个数。
2. 在“加载(写入)并发数”填写 reduce 任务的个数。

   目的连接为 HDFS 连接时，不显示“加载(写入)并发数”参数。
3. “单个分片的最大错误记录数”填写错误记录阈值。
4. 在“脏数据目录”填写一个脏数据的保存位置，例如“/user/sqoop/obs2hdfs-dd”。

步骤 9 单击“保存并运行”。

在“管理作业界面”，查看作业运行结果。可以单击“刷新列表”获取作业的最新状态。

----结束

17.11 Loader 常见问题

17.11.1 IE 10&IE 11 浏览器无法保存数据

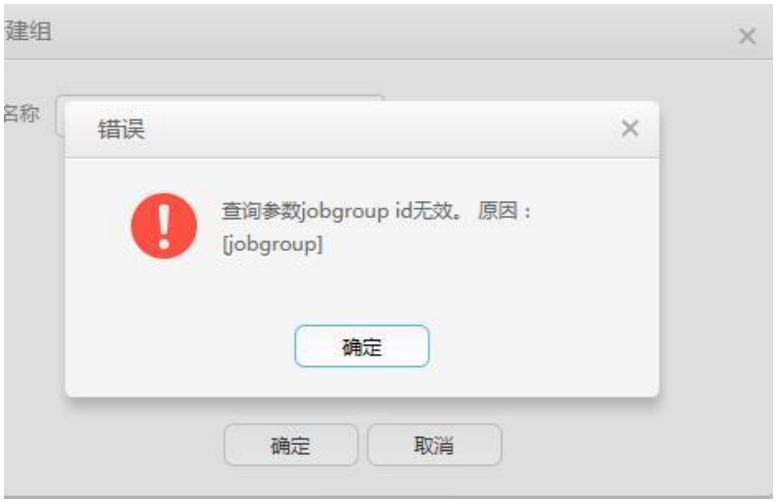
问题

通过 IE 10&IE 11 浏览器访问 Loader 界面，提交数据后，会报错。

回答

- 现象

  保存提交数据，出现类似报错。



- 原因

  IE 11 浏览器的某些版本在接收到 HTTP 307 响应时，会将 POST 请求转化为 GET 请求，从而使得 POST 数据无法下发到服务端。

- 解决建议

  使用 Google Chrome 浏览器。

17.11.2 将 Oracle 数据库中的数据导入 HDFS 时各连接器的区别

问题

使用 Loader 将 Oracle 数据库中的数据导入到 HDFS 中时，可选择的连接器有 generic-jdbc-connector、oracle-connector、oracle-partition-connector 三种，要怎么选？有什么区别？

答案

- generic-jdbc-connector

  使用 JDBC 方式从 Oracle 数据库读取数据，适用于支持 JDBC 的数据库。

  在这种方式下，Loader 加载数据的性能受限于分区列的数据分布是否均匀。当分区列的数据偏斜（数据集中在一个或者几个值）时，个别 Map 需要处理绝大部分数据，进而导致索引失效，造成 SQL 查询性能急剧下降。

  generic-jdbc-connector 支持视图的导入导出，而 oracle-partition-connector 和 oracle-connector 暂不支持，因此导入视图只能选择该连接器。

- oracle-partition-connector 和 oracle-connector

  这两种连接器都支持按照 Oracle 的 ROWID 进行分区（oracle-partition-connector 是自研，oracle-connector 是社区开源版本），二者的性能较为接近。

  oracle-connector 需要的系统表权限较多，下面是各自需要的系统表，需要赋予读权限。

  - oracle-connector：dba_tab_partitions、dba_constraints、dba_tables、dba_segments、v$version、dba_objects、v$instance、SYS_CONTEXT 函数、dba_extents、dba_tab_subpartitions。
  - oracle-partition-connector：DBA_OBJECTS、DBA_EXTENTS。

相比于 generic-jdbc-connector，oracle-partition-connector 和 oracle-connector 具有以下优点：

a. 负载均匀，数据分片的个数和范围与源表的数据无关，而是由源表的存储结构（数据块）确定，颗粒度可以达到“每个数据块一个分区”。

b. 性能稳定，完全消除“数据偏斜”和“绑定变量窥探”导致的“索引失效”。

c. 查询速度快，数据分片的查询速度比用索引快。

d. 水平扩展性好，如果数据量越大，产生的分片就越多，所以只要增加任务的并发数，就可以获得较理想的性能；反之，减少任务并发数，就可以节省资源。

e. 简化数据分片逻辑，不需要考虑“精度丢失”、“类型兼容”和“绑定变量”等问题。

f. 易用性得到增强，用户不需要专门为 Loader 创建分区列、分区表。

18 使用 Kudu

18.1 从零开始使用 Kudu

Kudu 是专为 Apache Hadoop 平台开发的列式存储管理器。Kudu 具有 Hadoop 生态系统应用程序的共同技术特性：可水平扩展，并支持高可用性操作。

前提条件

已安装集群客户端，例如安装目录为“/opt/hadoopclient”，以下操作的客户端目录只是举例，请根据实际安装目录修改。

操作步骤

步骤 1 以客户端安装用户，登录安装客户端的节点。

步骤 2 执行以下命令，切换到客户端安装目录。

cd /opt/hadoopclient

步骤 3 执行以下命令配置环境变量。

source bigdata_env

步骤 4 运行 Kudu 命令行工具。

直接执行 Kudu 组件的命令行工具，查看帮助。

kudu -h

回显信息如下：

```
Usage: kudu <command> [<args>]

<command> can be one of the following:
       cluster   Operate on a Kudu cluster
      diagnose   Diagnostic tools for Kudu servers and clusters
            fs   Operate on a local Kudu filesystem
           hms   Operate on remote Hive Metastores
  local_replica   Operate on local tablet replicas via the local filesystem
        master   Operate on a Kudu Master
           pbc   Operate on PBC (protobuf container) files
```

```
         perf   Measure the performance of a Kudu cluster
remote_replica   Operate on remote tablet replicas on a Kudu Tablet Server
        table   Operate on Kudu tables
       tablet   Operate on remote Kudu tablets
         test   Various test actions
      tserver   Operate on a Kudu Tablet Server
          wal   Operate on WAL (write-ahead log) files
```

说明

kudu 命令行工具不提供 DDL、DML 等操作，但提供针对 cluster、master、tserver、fs、table 等的细化查询功能。

常用操作：

- 查看当前集群下有哪些表。

  kudu table list *KuduMaster 实例 IP1:7051, KuduMaster 实例 IP2:7051, KuduMaster 实例 IP3:7051*

- 查询 Kudu 服务 KuduMaster 实例的配置信息。

  kudu master get_flags *KuduMaster 实例 IP:7051*

- 查询表的 schema。

  kudu table describe *KuduMaster 实例 IP1:7051, KuduMaster 实例 IP2:7051, KuduMaster 实例 IP3:7051 tablename*

- 删除表。

  kudu table delete *KuduMaster 实例 IP1:7051, KuduMaster 实例 IP2:7051, KuduMaster 实例 IP3:7051 tablename*

说明

KuduMaster 实例 IP 获取方式：在集群详情页面，选择“组件管理 > Kudu > 实例”，获取角色 KuduMaster 的 IP 地址。

----结束

18.2 访问 Kudu 的 WebUI

用户可以通过 Kudu 的 WebUI，在图形化界面查看 Kudu 作业的相关信息。

前提条件

已安装 Kudu 服务的集群。

访问 KuduMaster WebUI（MRS 3.x 及之后版本）

步骤 1 登录 Manager 页面，请参见访问 FusionInsight Manager（MRS 3.x 及之后版本）。

步骤 2 选择“集群 > 服务 > Kudu”。

步骤 3 在“Kudu 概览”的“KuduMaster WebUI”中单击“KuduMaster(KuduMaster)”，打开 KuduMaster 的 WebUI 页面。

----结束

访问 KuduMaster WebUI（MRS 3.x 之前版本）

步骤 1 登录 Manager 页面，请参见访问 MRS Manager（MRS 3.x 之前版本）。

步骤 2 选择“服务管理 > Kudu”。

步骤 3 在“Kudu 概述”的“KuduMaster WebUI”中单击“KuduMaster(KuduMaster)”，打开 KuduMaster 的 WebUI 页面。

----结束

19 使用 Mapreduce

19.1 配置日志归档和清理机制

配置场景

执行一个 MapReduce 应用会产生两种类型日志文件：作业日志和任务日志。

- 作业日志由 MRApplicationMaster 产生，详细记录了作业启动时间、运行时间，每个任务启动时间、运行时间、Counter 值等信息。此日志内容被 HistoryServer 解析以后用于查看作业执行的详细信息。
- 任务日志记录了每个运行在 Container 中的任务输出的日志信息。默认情况下，任务日志只会存放在各 NodeManager 的本地磁盘上。打开日志聚合功能后，NodeManager 会在作业运行完成后将本地的任务日志进行合并，写入到 HDFS 中。

由于 MapReduce 的作业日志和任务日志（聚合功能开启的情况下）都保存在 HDFS 上。对于计算任务量大的集群，如果不进行合理的配置对日志文件进行定期归档和删除，日志文件将占用 HDFS 大量内存空间，增加集群负载。

日志归档是通过 Hadoop Archives 功能实现的，Hadoop Archives 启动的并行归档任务数（Map 数）与待归档的日志文件总大小有关。计算公式为：并行归档任务数=待归档的日志文件总大小/归档文件大小。

配置描述

进入 Mapreduce 服务参数“全部配置”界面，具体操作请参考修改集群服务配置参数章节。

在搜索框中输入参数名称，修改并保存配置。然后在 Mapreduce 服务“概览”页面选择“更多 > 同步配置”。同步完成后重启 Mapreduce 服务。

- 作业日志参数：

表19-1 参数说明

参数	描述	默认值

参数	描述	默认值
mapreduce.jobhistory.cleaner.enable	是否开启作业日志文件清理功能。	true
mapreduce.jobhistory.cleaner.interval-ms	作业日志文件清理启动周期。只有保留时间比“mapreduce.jobhistory.max-age-ms”更长的日志文件才会被清除。	86400000（1天）
mapreduce.jobhistory.max-age-ms	比此项设置的毫秒数保留时间更长的作业日志文件将被清理。	1296000000（15天）

- 任务日志参数：

表19-2 参数说明

参数	描述	默认值
yarn.log-aggregation.archive.files.minimum	设置 Mapreduce 任务日志归档最小文件数。当“yarn.nodemanager.remote-app-log-dir”文件夹下文件数大于等于该设置的值时归档任务启动。 该参数适用于 MRS 3.x 版本。	5000
yarn.log-aggregation.archive-check-interval-seconds	设置 Mapreduce 任务日志归档任务启动周期（秒）。只有日志文件数达到“yarn.log-aggregation.archive.files.minimum”设置值时日志文件才会被归档。周期设置为“0”或“-1”时归档功能禁用。 该参数适用于 MRS 3.x 版本。	-1
yarn.log-aggregation.retain-seconds	设置 Mapreduce 任务日志在 HDFS 上的保留时间。设置为“-1”时日志文件永久保存。	1296000
yarn.log-aggregation.retain-check-interval-seconds	设置 Mapreduce 任务日志清理任务的检查周期（秒）。设置为“-1”时检查周期为日志保留时间的十分之一。	86400

19.2 降低客户端应用的失败率

配置场景

当网络不稳定或者集群 IO、CPU 负载过高的情况下，通过调整如下参数值，降低客户端应用的失败率，保证应用的正常运行。

配置描述

在客户端的“mapred-site.xml”配置文件中调整如下参数。

说明

“mapred-site.xml”配置文件在客户端安装路径的 conf 目录下，例如“/opt/client/Yarn/config”。

表19-3 参数说明

参数	描述	默认值
mapreduce.reduce.shuffle.max-host-failures	MR 任务在 reduce 过程中读取远端 shuffle 数据允许失败的次数。当设置次数大于 5 时，可以降低客户端应用的失败率。该参数适用于 MRS 3.x 版本。	5
mapreduce.client.submit.file.replication	MR 任务在运行时依赖的相关 job 文件在 HDFS 上的备份。当备份数大于 10 时，可以降低客户端应用的失败率。	10

19.3 将 MR 任务从 Windows 上提交到 Linux 上运行

配置场景

用户将 MapReduce 任务从 Windows 上提交到 Linux 上运行，则“mapreduce.app-submission.cross-platform”参数值需配置为“true”。若集群无此参数，或参数值为“false”，则表示集群不支持此功能，需要按照如下操作增加该参数或修改参数值进行开启。

说明

本章节操作适用于 MRS 3.x 及之后版本。

配置描述

在客户端的“mapred-site.xml”配置文件中进行如下配置。“mapred-site.xml”配置文件在客户端安装路径的 config 目录下，例如“/opt/client/Yarn/config”。

表19-4 参数说明

参数	描述	默认值
mapreduce.app-submission.cross-platform	支持在 Windows 上提交到 Linux 上运行 MR 任务的配置项。当该参数的值设为“true”时，表示支持。当该参数的值设为“false”时，表示不支持。	true

19.4 配置使用分布式缓存

配置场景

说明

本章节操作适用于 MRS 3.x 及之后版本。

分布式缓存在两种情况下非常有用。

滚动升级

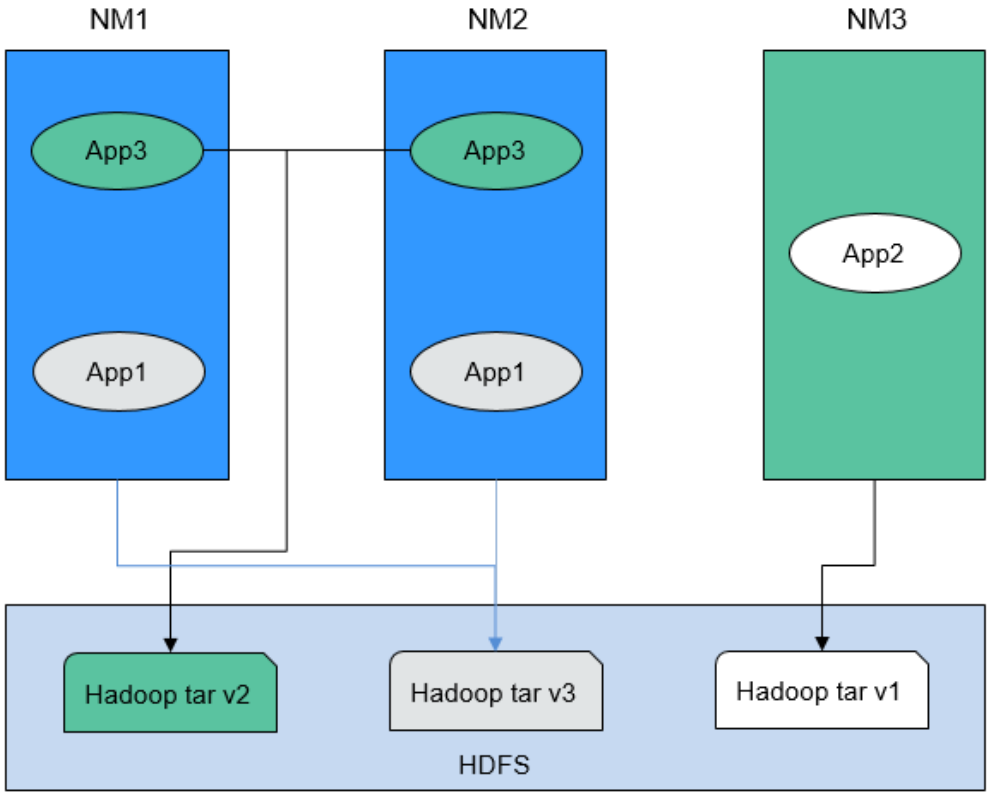
在升级过程中，应用程序必须保持文字内容（jar 文件或配置文件）不变。而这些内容并非基于当前版本的 YARN，而是要基于其提交时的版本。这是一个具有挑战性的问题。一般情况下，应用程序（例如 MapReduce、Hive、Tez 等）需要进行完整的本地安装，将库安装至所有的集群机器（客户端及服务器端机器）中。当集群内开始进行滚动升级或降级时，本地安装的库的版本必然会在应用运行过程时发生改变。在滚动升级过程中，首先只会对少数 NodeManager 进行升级，这些 NodeManager 会获得新版本的软件。这导致了行为的不一致，并可能发生运行时错误。

同时存在多个 YARN 版本

集群管理员可能会在一个集群内运行使用多个版本 YARN 及 Hadoop jars 的任务。这在当前很难实现，因为 jars 已被本地化且只有一个版本。

MapReduce 应用框架可以通过分布式缓存进行部署，且无需依赖安装中复制的静态版本。因此，可以在 HDFS 中存放多版本的 Hadoop，并通过配置“mapred-site.xml”文件指定任务默认使用的版本。只需设置适当的配置属性，用户就可以运行不同版本的 MapReduce，而无需使用部署在集群中的版本。

图19-1 具有多个版本 NodeManagers 及 Applications 的集群



在图 19-1 中：可以看出，应用程序可以使用 HDFS 中的 Hadoop jars，而无需使用本地版本。因此在滚动升级中，即使 NodeManager 已经升级，应用程序仍然可以运行旧版本的 Hadoop。

配置描述

步骤 1 首先，需要将指定版本的 MapReduce tar 包存放至 HDFS 中应用程序可以访问的目录下，如下所示：

$HADOOP_HOME/**bin/hdfs dfs -put** *hadoop-x.tar.gz /mapred/framework/*

步骤 2 根据表 19-5，对“mapred-site.xml”文件中的参数进行设置。

表19-5 分布式缓存相关参数

参数	说明	默认值
mapreduce.application.framework.path	此参数值为指向存档位置的 URL。 说明 如果对 URL 片段标示名称进行如下指定，该属性还可以为存档创建别名。作为示例，这里将别名设为了 mr-framework。 <property> <name>mapreduce.application.framework.path</name>	NA

参数	说明	默认值
	<value>hdfs:/mapred/framework/hadoop-x.tar.gz#mr-framework</value> </property>	
mapreduce.application.classpath	设定属性 mapreduce.application.classpath，使其可以包含类目录中相关的 MR jars。 说明 例如，此处利用在框架路径中使用过的别名“mr-framework”对目录进行匹配。 <property> <name>mapreduce.application.classpath</name> <value>$PWD/mr-framework/hadoop/share/hadoop/mapreduce/*:$PWD/mr-framework/hadoop/share/hadoop/mapreduce/lib/*:$PWD/mr-framework/hadoop/share/hadoop/common/*:$PWD/mr-framework/hadoop/share/hadoop/common/lib/*:$PWD/mr-framework/hadoop/share/hadoop/yarn/*:$PWD/mr-framework/hadoop/share/hadoop/yarn/lib/*:$PWD/mr-framework/hadoop/share/hadoop/hdfs/*:$PWD/mr-framework/hadoop/share/hadoop/hdfs/lib/*:/etc/hadoop/conf/secure</value></property>	NA

可以将多个版本的 MR tarball 上传至 HDFS。不同的“mapred-site.xml”文件可以指向不同的位置。用户在此之后可以针对特定的“mapred-site.xml”文件运行任务。以下是一个针对 x 版本的 MR tarball 运行 MR 任务的例子：

hadoop jar *share/hadoop/mapreduce/hadoop-mapreduce-examples-*.jar pi* **-conf etc/hadoop-x/mapred-site.xml** *10 10*

----结束

19.5 配置 MapReduce shuffle address

配置场景

当 MapReduce shuffle 服务启动时，它尝试基于 localhost 绑定 IP。如果需要 MapReduce shuffle 服务去连接特定 IP，那么没有可用的配置。下面的描述允许您配置连接到特定的 IP。

配置描述

当需要 MapReduce shuffle 服务绑定特定 IP 时，需要在 NodeManager 实例所在节点的配置文件“mapred-site.xml”中设置下面的参数。

表19-6 参数描述

参数	描述	默认值

参数	描述	默认值
mapreduce.shuffle.address	指定地址来运行 shuffle 服务，格式是 IP:PORT，参数的默认值为空。当参数值为空时，将绑定 localhost，默认端口为 13562。 说明 如果涉及到的 PORT 值和配置的 mapreduce.shuffle.port 值不一样时，mapreduce.shuffle.port 将不会生效。	-

19.6 配置集群管理员列表

配置场景

该功能主要用于指定 MapReduce 集群管理员。

其中，管理员列表由参数“mapreduce.cluster.administrators”指定，集群管理员 admin 具有所有可以操作的权限。

配置描述

进入 Mapreduce 服务参数“全部配置”界面，在搜索框中输入参数名称。具体操作请参考修改集群服务配置参数章节。

表19-7 参数描述

参数	描述	默认值
mapreduce.cluster.acls.enabled	是否开启对 Job History Server 权限控制的开关。	true
mapreduce.cluster.administrators	用于指定 MapReduce 集群的管理员列表，可以配置用户和用户组，用户或者用户组之间用逗号间隔，用户和用户组之间用空格间隔，举例：userA,userB groupA,groupB。当配置为*时表示所有用户或用户组。	MRS 3.x 之前版本：mapred MRS 3.x 及之后版本： mapred supergroup,System_administrator_186

19.7 MapReduce 日志介绍

日志描述

日志默认存储路径：

- JobhistoryServer：“/var/log/Bigdata/mapreduce/jobhistory”（运行日志），“/var/log/Bigdata/audit/mapreduce/jobhistory”（审计日志）
- Container：“/srv/BigData/hadoop/data1/nm/containerlogs/application_${appid}/container_{$contid}”

说明

运行中的任务日志存储在以上路径中，运行结束后会基于 YARN 的配置是否汇聚到 HDFS 目录中，详情请参见 Yarn 常用参数。

日志归档规则：

MapReduce 的日志启动了自动压缩归档功能，缺省情况下，当日志大小超过 50MB 的时候，会自动压缩，压缩后的日志文件名规则为：“*<原有日志名>*-<yyyy-mm-dd_hh-mm-ss>.[*编号*].log.zip”。最多保留最近的 100 个压缩文件，压缩文件保留个数可以在参数配置界面中配置。

在 MapReduce 服务中，JobhistoryServer 会定时去清理 HDFS 上存储的旧的日志文件（默认目录为 HDFS 文件系统中的“/mr-history/done”），具体清理的时间间隔参数配置为 mapreduce.jobhistory.max-age-ms，默认值为 1296000000，即 15 天。

表19-8 MR 日志列表

日志类型	日志文件名	描述
运行日志	jhs-daemon-start-stop.log	守护进程（Daemon）的启动日志。
	hadoop-<SSH_USER>-jhshadaemon-<hostname>.log	守护进程（Daemon）的运行日志。
	hadoop-<SSH_USER>-<process_name>-<hostname>.out	MR 运行环境信息日志。
	historyserver-<SSH_USER>-<DATE>-<PID>-gc.log	MR 服务垃圾回收日志。
	jhs-haCheck.log	MR 实例主备状态检查日志。
	yarn-start-stop.log	MR 服务启停操作日志。
	yarn-prestart.log	MR 服务启动前集群操作的记录日志。
	yarn-postinstall.log	MR 服务安装后启动前的

日志类型	日志文件名	描述
		工作日志。
	yarn-cleanup.log	MR 服务卸载时候的清理日志。
	mapred-service-check.log	MR 服务健康状态检测日志。
	container_{$contid}	Container 日志。
	hadoop-<SSH_USER>-<process_name>-<hostname>.log	MR 运行日志。
	mapred-switch-jhs.log	MR 主备倒换日志。
	env.log	实例启停前的环境信息日志。
审计日志	mapred-audit-jobhistory.log	MR 操作审计日志。
	SecurityAuth.audit	MR 安全审计日志。

日志级别

MapReduce 中提供了如表 19-9 所示的日志级别。其中日志级别优先级从高到低分别是 FATAL、ERROR、WARN、INFO、DEBUG。程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。

表19-9 日志级别

级别	描述
FATAL	FATAL 表示当前事件处理存在严重错误信息。
ERROR	ERROR 表示当前事件处理存在错误信息。
WARN	WARN 表示当前事件处理存在异常告警信息。
INFO	INFO 表示记录系统及各事件正常运行状态信息。
DEBUG	DEBUG 表示系统及系统的调试信息。

如果您需要修改日志级别，请执行如下操作：

步骤 1 进入 MapReduce 服务参数“全部配置”界面，具体操作请参考修改集群服务配置参数。

步骤 2 左边菜单栏中选择所需修改的角色所对应的日志菜单。

步骤 3 选择所需修改的日志级别。

步骤 4 保存配置，在弹出窗口中单击“确定”使配置生效。

说明

配置完成后立即生效，不需要重启服务。

----结束

日志格式

MapReduce 日志格式如下所示：

表19-10 日志格式

日志类型	格式	示例
运行日志	<yyyy-MM-dd HH:mm:ss,SSS>|<*Log Level*>|<*产生该日志的线程名字*>|<*log 中的message*>|<*日志事件的发生位置*>	2020-01-26 14:18:59,109 | INFO | main | Client environment:java.compiler=<NA> | org.apache.zookeeper.Environment.logEnv(Environment.java:100)
审计日志	<yyyy-MM-dd HH:mm:ss,SSS>|<*Log Level*>|<*产生该日志的线程名字*>|<*log 中的message*>|<*日志事件的发生位置*>	2020-01-26 14:24:43,605 | INFO | main-EventThread | USER=omm OPERATION=refreshAdminAcls TARGET=AdminService RESULT=SUCCESS | org.apache.hadoop.yarn.server.resourcemanager.RMAuditLogger$LogLevel$6.printLog(RMAuditLogger.java:91)

19.8 MapReduce 性能调优

19.8.1 多 CPU 内核下的调优配置

操作场景

当 CPU 内核数很多时，如 CPU 内核为磁盘数的 3 倍时的调优配置。

操作步骤

以下参数有如下两个配置入口：

- 服务器端配置

进入 Yarn 服务参数“全部配置”界面，在搜索框中输入参数名称。具体操作请参考修改集群服务配置参数章节。

- 客户端配置

  直接在客户端中修改相应的配置文件。

说明

- HDFS 客户端配置文件路径：*客户端安装目录*/HDFS/hadoop/etc/hadoop/hdfs-site.xml。
- Yarn 客户端配置文件路径：*客户端安装目录*/HDFS/hadoop/etc/hadoop/yarn-site.xml。
- MapReduce 客户端配置文件路径：*客户端安装目录*/HDFS/hadoop/etc/hadoop/mapred-site.xml。

表19-11 多 CPU 内核设置

配置	描述	参数	默认值	Server/Client	影响	备注
节点容器槽位数	如下配置组合决定了每节点任务(map、reduce)的并发数。 • “yarn.nodemanager.resource.memory-mb” • “mapreduce.map.memory.mb” • “mapreduce.reduce.memory.mb”	yarn.nodemanager.resource.memory-mb 说明 MRS 3.x 之前版本：需要在 MRS 控制台中进行配置。 MRS 3.x 及之后版本：需要在 FusionInsight Manager 系统进行配置。	MRS 3.x 之前版本：8192 MRS 3.x 及之后版本：16384	Server	如果所有的任务(map/reduce)需要读写数据至磁盘，多个进程将会同时访问一个磁盘。这将会导致磁盘的 IO 性能非常的低下。为了改善磁盘的性能，请确保客户端并发访问磁盘的数不大于3。	最大并发的 container 数量应该为[2.5 * Hadoop 中磁盘配置数]。
		mapreduce.map.memory.mb 说明 需要在客户端进行配置，配置文件路径：*客户端安装目录*/HDFS/hadoop/etc/hadoop/mapred-site.xml。	4096	Client		
		mapreduce.reduce.memory.mb	4096	Client		

配置	描述	参数	默认值	Server/Client	影响	备注
		说明 需要在客户端进行配置，配置文件路径：*客户端安装目录*/HDFS/hadoop/etc/hadoop/mapred-site.xml。				
Map输出与压缩	Map任务所产生的输出可以在写入磁盘之前被压缩，这样可以节约磁盘空间并得到更快的写盘速度，同时可以减少至Reducer的数据传输量。需要在客户端进行配置。 • mapreduce.map.output.compress 指定了Map任务输出结果可以在网络传输前被压缩。这是一个per-job的配置。 • mapreduce.map.output.compress.c	mapreduce.map.output.compress 说明 需要在客户端进行配置，配置文件路径：*客户端安装目录*/HDFS/hadoop/etc/hadoop/mapred-site.xml。	true	Client	在这种情况下，磁盘的IO是主要瓶颈。所以可以选择一种压缩率非常高的压缩算法。	编解码器可配置为Snappy，Benchmark测试结果显示Snappy是非常平衡以及高效的编码器。
		mapreduce.map.output.compress.codec 说明 需要在客户端进行配置，配置文件路径：*客户端安装目录*/HDFS/hadoop/etc/hadoop/mapred-site.xml。	org.apache.hadoop.io.compress.Lz4Codec	Client		

配置	描述	参数	默认值	Server/Client	影响	备注
	odec 指定用于压缩的编解码器。					
Spills	mapreduce.map.sort.spill.percent	mapreduce.map.sort.spill.percent 说明 需要在客户端进行配置，配置文件路径：*客户端安装目录*/HDFS/hadoop/etc/hadoop/mapred-site.xml。	0.8	Client	磁盘 IO 是主要瓶颈，合理配置“mapreduce.task.io.sort.mb”可以使溢出至磁盘的内容最小化。	-
数据包大小	当 HDFS 客户端写数据至数据节点时，数据会被累积，直到形成一个包。然后这个数据包会通过网络传输。dfs.client-write-packet-size 配置项可以指定该数据包的大小。这个可以通过每个 job 进行指定。	dfs.client-write-packet-size 说明 需要在客户端进行配置，配置文件路径：*客户端安装目录*/HDFS/hadoop/etc/hadoop/hdfs-site.xml。	262144	Client	数据节点从 HDFS 客户端接收数据包，然后将数据包里的数据单线程写入磁盘。当磁盘处于并发写入状态时，增加数据包的大小可以减少磁盘寻道时间，从而提升 IO 性能。	dfs.client-write-packet-size = 262144

19.8.2 确定 Job 基线

操作场景

确定 Job 基线是调优的基础，一切调优项效果的检查，都是通过和基线数据做对比来获得。

Job 基线的确定有如下三个原则：

- 充分利用集群资源
- reduce 阶段尽量放在一轮
- 每个 task 的执行时间要合理

操作步骤

- **原则一：充分利用集群资源。**

  Job 运行时，会让所有的节点都有任务处理，且处于繁忙状态，这样才能保证资源充分利用，任务的并发度达到最大。可以通过调整处理的数据量大小，以及调整 map 和 reduce 个数来实现。

  Reduce 个数的控制使用“mapreduce.job.reduces”。

  Map 个数取决于使用了哪种 InputFormat，以及待处理的数据文件是否可分割。默认的 TextFileInputFormat 将根据 block 的个数来分配 map 数(一个 block 一个 map)。通过如下配置参数进行调整。

  参数入口：

  进入 Yarn 服务参数“全部配置”界面，在搜索框中输入参数名称。具体操作请参考修改集群服务配置参数章节。

参数	描述	默认值
mapreduce.input.fileinputformat.split.maxsize	map 输入信息应被拆分成的数据块的最大大小。 由用户定义的分片大小的设置及每个文件 block 大小的设置，可以计算分片的大小。计算公式如下： `splitSize = Math.max(minSize, Math.min(maxSize, blockSize))` 如果 maxSize 设置大于 blockSize，那么每个 block 就是一个分片，否则就会将一个 block 文件分隔为多个分片，如果 block 中剩下的一小段数据量小于 splitSize，还是认为它是独立的分片。	-
mapreduce.input.fileinputformat.split.minsize	可以设置数据分片的数据最小值。	0

- **原则二：控制 reduce 阶段在一轮中完成。**

避免以下两种场景：

- 大部分的 reduce 在第一轮运行完后，剩下唯一一个 reduce 继续运行。这种情况下，这个 reduce 的执行时间将极大影响这个 job 的运行时间。因此需要将 reduce 个数减少。
- 所有的 map 运行完后，只有个别节点有 reduce 在运行。这时候集群资源没有得到充分利用，需要增加 reduce 的个数以便每个节点都有任务处理。

- **原则三：每个 task 的执行时间要合理。**

  如果一个 job，每个 map 或 reduce 的执行时间只有几秒钟，就意味着这个 job 的大部分时间都消耗在 task 的调度和进程启停上了，因此需要增加每个 task 处理的数据大小。建议一个 task 处理时间为 1 分钟。

  控制单个 task 处理时间的大小，可以通过如下配置来调整。

  参数入口：

  进入 Yarn 服务参数“全部配置”界面，在搜索框中输入参数名称。具体操作请参考修改集群服务配置参数章节。

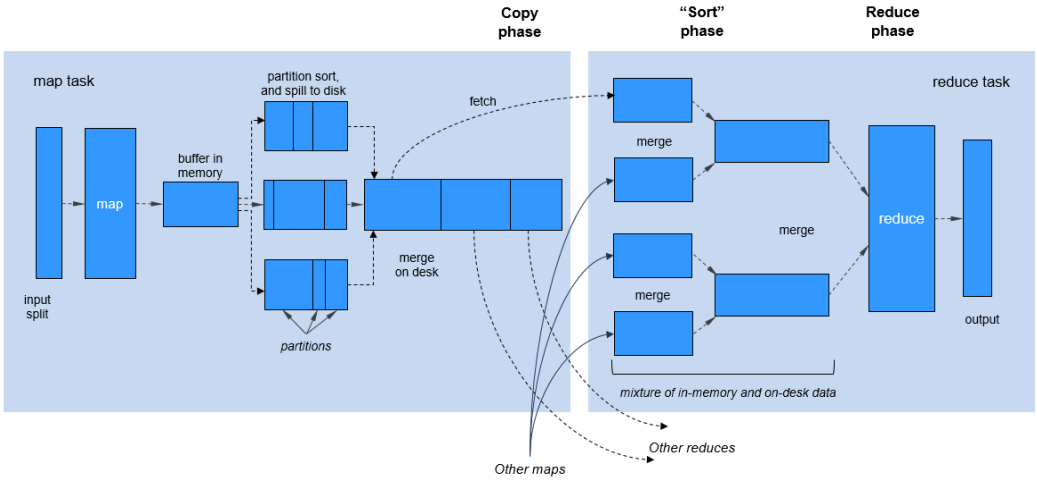
参数	描述	默认值
mapreduce.input.fileinputformat.split.maxsize	map 输入信息应被拆分成的数据块的最大大小。 由用户定义的分片大小的设置及每个文件 block 大小的设置，可以计算分片的大小。计算公式如下： `splitSize = Math.max(minSize, Math.min(maxSize, blockSize))` 如果 maxSize 设置大于 blockSize，那么每个 block 就是一个分片，否则就会将一个 block 文件分隔为多个分片，如果 block 中剩下的一小段数据量小于 splitSize，还是认为它是独立的分片。	-
mapreduce.input.fileinputformat.split.minsize	可以设置数据分片的数据最小值。	0

19.8.3 Shuffle 调优

操作场景

Shuffle 阶段是 MapReduce 性能的关键部分，包括了从 Map task 将中间数据写到磁盘一直到 Reduce task 拷贝数据并最终放到 reduce 函数的全部过程。这一块 Hadoop 提供了大量的调优参数。

图19-2 Shuffle 过程



操作步骤

1. **Map 阶段的调优**

   - 判断 Map 使用的内存大小

     判断 Map 分配的内存是否足够，一个简单的办法是查看运行完成的 job 的 Counters 中，对应的 task 是否发生过多次 GC，以及 GC 时间占总 task 运行时间之比。通常，GC 时间不应超过 task 运行时间的 10%，即 GC time elapsed (ms)/CPU time spent (ms)<10%。

     主要通过如下参数进行调整。

     参数入口：

     进入 Yarn 服务参数“全部配置”界面，在搜索框中输入参数名称。具体操作请参考修改集群服务配置参数章节。

表19-12 参数说明

参数	描述	默认值
mapreduce.map.memory.mb	map 任务的内存限制。	4096
mapreduce.map.java.opts	map 子任务的 JVM 参数。如果设置，会替代 mapred.child.java.opts 参数；如果未设置-Xmx，Xmx 值从 mapreduce.map.memory.mb*mapreduce.job.heap.memory-mb.ratio 计算获取。	MRS 3.x 之前版本：-Xmx2048M -Djava.net.preferIPv4Stack=true MRS 3.x 及之后版本： • 集群已开启 Kerberos 认证：-Djava.net.preferIPv4Stack=true -Djava.net.preferIPv6Addresses=false -

参数	描述	默认值
		Djava.security.krb5.conf=*${BIGDATA_HOME}*/common/runtime/krb5.conf -Dbeetle.application.home.path=*${BIGDATA_HOME}*/common/runtime/security/config • 集群未开启 Kerberos 认证：-Djava.net.preferIPv4Stack=true -Djava.net.preferIPv6Addresses=false -Dbeetle.application.home.path=*${BIGDATA_HOME}*/common/runtime/security/config

建议：配置“mapreduce.map.java.opts”参数中“-Xmx”值为“mapreduce.map.memory.mb”参数值的 0.8 倍。

- 使用 Combiner

  在 Map 阶段，有一个可选过程，将同一个 key 值的中间结果合并，叫做 combiner。一般将 reduce 类设置为 combiner 即可。通过 combine，一般情况下可以显著减少 Map 输出的中间结果，从而减少 shuffle 过程的网络带宽占用。可通过如下接口为一个任务设置 Combiner 类。

表19-13 Combiner 设置接口

类名	接口名	描述
org.apache.hadoop.mapreduce.Job	public void setCombinerClass(Class<? extends Reducer> cls)	为 Job 设置一个 combiner 类。

2. **Copy 阶段的调优**

   - 数据是否压缩

     对 Map 的中间结果进行压缩，当数据量大时，会显著减少网络传输的数据量，但是也因为多了压缩和解压，带来了更多的 CPU 消耗。因此需要做好权衡。当任务属于网络瓶颈类型时，压缩 Map 中间结果效果明显。针对 bulkload 调优，压缩中间结果后性能提升 60%左右。

     配置方法：将“mapreduce.map.output.compress”参数值设置为“true”，将“mapreduce.map.output.compress.codec”参数值设置为“org.apache.hadoop.io.compress.SnappyCodec”。

3. **Merge 阶段的调优**

   通过调整如下参数减少 reduce 写磁盘的次数。

   参数入口：

   进入 Yarn 服务参数“全部配置”界面，在搜索框中输入参数名称。具体操作请参考修改集群服务配置参数章节。

表19-14 参数说明

参数	描述	默认值
mapreduce.reduce.merge.inmem.threshold	内存合并进程的文件数阈值。累计文件数达到阈值时会发起内存合并及溢出到磁盘。小于等于 0 的值表示该阈值不生效且仅基于 ramfs 的内存使用情况来触发合并。	1000
mapreduce.reduce.shuffle.merge.percent	发起内存合并的使用率阈值，表示为分配给映射输出信息的内存的比例（是由 mapreduce.reduce.shuffle.input.buffer.percent 设置的）。	0.66
mapreduce.reduce.shuffle.input.buffer.percent	shuffle 过程中分配给映射输出信息的内存占最大堆大小的比例。	0.70
mapreduce.reduce.input.buffer.percent	Reduce 过程中保存映射输出信息的内存相对于最大堆大小的比例。当 shuffle 结束时，需保证 reduce 开始前内存中所有剩余的映射输出信息所使用的内存小于该阈值。	0.0

19.8.4 大任务的 AM 调优

操作场景

任务场景：运行的一个大任务（map 总数达到了 10 万的规模），但是一直没有跑成功。经过查询，发现是 ApplicationMaster（以下简称 AM）反应缓慢，最终超时失败。

此任务的问题是，task 数量变多时，AM 管理的对象也线性增长，因此就需要更多的内存来管理。AM 默认分配的内存堆大小是 1GB。

操作步骤

通过调大如下的参数来进行 AM 调优。

参数入口：

在 Yarn 客户端的“mapred-site.xml”配置文件中调整如下参数。“mapred-site.xml”配置文件在客户端安装路径的 conf 目录下，例如“/opt/client/Yarn/config”。

参数	描述	默认值
yarn.app.mapreduce.am.resource.mb	该参数值必须大于下面参数的堆大小。单位：MB	1536
yarn.app.mapreduce.am.command-opts	传递到 MapReduce ApplicationMaster 的 JVM 启动参数。	MRS 3.x 之前版本：-Xmx1024m -XX:CMSFullGCsBeforeCompaction=1 -XX:+UseConcMarkSweepGC -XX:+CMSParallelRemarkEnabled -XX:+UseCMSCompactAtFullCollection -verbose:gc MRS 3.x 及之后版本：-Xmx1024m -XX:+UseConcMarkSweepGC -XX:+CMSParallelRemarkEnabled -verbose:gc -Djava.security.krb5.conf=${KRB5_CONFIG} -Dhadoop.home.dir=*${BIGDATA_HOME}*/FusionInsight_HD_*xxx*/install/FusionInsight-Hadoop-*xxx*/hadoop

19.8.5 推测执行

操作场景

当集群规模很大时（如几百上千台节点的集群），个别机器出现软硬件故障的概率就变大了，并且会因此延长整个任务的执行时间（跑完的任务都在等出问题的机器跑结束）。推测执行通过将一个 task 分给多台机器跑，取先运行完的那个，会很好的解决这个问题。对于小集群，可以将这个功能关闭。

操作步骤

参数入口：

进入 Yarn 服务参数“全部配置”界面，在搜索框中输入参数名称。具体操作请参考修改集群服务配置参数章节。

参数	描述	默认值
mapreduce.map.speculative	设置是否并行执行某些映射任务的多个实例。true 表示开启。	false
mapreduce.reduce.speculative	设置是否并行执行某些 reduce 任务的多个实例。true 表示开启。	false

19.8.6 通过“Slow Start”调优

操作场景

Slow Start 特性指定 Map 任务完成度为多少时 Reduce 任务可以启动，过早启动 Reduce 任务会导致资源占用，影响任务运行效率，但适当的提早启动 Reduce 任务会提高 Shuffle 阶段的资源利用率，提高任务运行效率。例如：某集群可启动 10 个 Map 任务，MapReduce 作业共 15 个 Map 任务，那么在一轮 Map 任务执行完成后只剩 5 个 Map 任务，集群还有剩余资源，在这种场景下，配置 Slow Start 参数值小于 1，比如 0.8，则 Reduce 就可以利用集群剩余资源。

操作步骤

参数入口：

进入 Mapreduce 服务参数“全部配置”界面，在搜索框中输入参数名称。具体操作请参考修改集群服务配置参数章节。

参数	描述	默认值
mapreduce.job.reduce.slowstart.completedmaps	为 job 安排 reduce 前应完成的映射数的分数形式。默认 100%的 Map 跑完后开始起 Reduce。	1.0

19.8.7 MR job commit 阶段优化

操作场景

默认情况下，如果一个 MR 任务会产生大量的输出结果文件，那么该 job 在最后的 commit 阶段会耗费较长的时间将每个 task 的临时输出结果 commit 到最终的结果输出目录。特别是在大集群中，大 Job 的 commit 过程会严重影响任务的性能表现。

针对以上情况，可以通过将以下参数“mapreduce.fileoutputcommitter.algorithm.version”配置为“2”，来提升 MR Job commit 阶段的性能。

操作步骤

参数入口：

进入 Yarn 服务参数“全部配置”界面，在搜索框中输入参数名称。具体操作请参考修改集群服务配置参数章节。

表19-15 参数说明

参数	描述	默认值
mapreduce.fileoutputcommi	用于指定 Job 的最终输出	2

参数	描述	默认值
tter.algorithm.version	文件提交的算法版本，取值为“1”或“2”。 说明 版本 2 为建议的优化算法版本。该算法通过让任务直接将每个 task 的输出结果提交到最终的结果输出目录，从而减少大作业的输出提交时间。	

19.9 MapReduce 常见问题

19.9.1 ResourceManager 进行主备切换后，任务中断后运行时间过长

问题

在 MapReduce 任务运行过程中，ResourceManager 发生主备切换，切换完成后，MapReduce 任务继续执行，此时任务的运行时间过长。

回答

因为 ResourceManager HA 已启用，但是 Work-preserving RM restart 功能未启用。

如果 Work-preserving RM restart 功能未启用，ResouceManager 切换时 container 会被 kill，然后导致 Application Master 超时。Work-preserving RM restart 功能介绍请参见：http://hadoop.apache.org/docs/r3.1.1/hadoop-yarn/hadoop-yarn-site/ResourceManagerRestart.html

可以通过如下方式解决此问题：

设置如下参数启用 Work-preserving RM restart 功能。

“yarn.resourcemanager.work-preserving-recovery.enabled”=“true”

19.9.2 MapReduce 任务长时间无进展

问题

MapReduce 任务长时间无进展。

回答

一般是因为内存太少导致的。当内存较小时，任务中拷贝 map 输出的时间将显著增加。

为了减少等待时间，您可以适当增加堆内存空间。

任务的配置可根据 mapper 的数量和各 mapper 的数据大小来进行优化。根据输入数据的大小，优化如下参数：

- “mapreduce.reduce.memory.mb”
- “mapreduce.reduce.java.opts”

例如：如果 10 个 mapper 的数据大小为 5GB，那么理想的堆内存是 1.5GB。随着数据大小的增加而增加堆内存大小。

19.9.3 运行任务时，客户端不可用

问题

当运行任务时，将 MR ApplicationMaster 或 ResourceManager 移动为 D 状态，为什么此时客户端会不可用？

回答

当运行任务时，将 MR ApplicationMaster 或 ResourceManager 移动为 D 状态（不间断睡眠状态）或 T 状态（停止状态），客户端会等待返回任务运行的状态，由于 AM 无返回，客户端会一直处于等待状态。

为避免出现上述场景，使用“core-site.xml”中的“ipc.client.rpc.timeout”配置项设置客户端超时时间。

该参数的参数值为毫秒。默认值为 0，表示无超时。客户端超时的取值范围可以为 0～2147483647 毫秒。

说明

- 如果 Hadoop 进程已处于 D 状态，重启该进程所处的节点。
- “core-site.xml” 配置文件在客户端安装路径的 conf 目录下，例如“/opt/hadoopClient/Yarn/config”。

19.9.4 在缓存中找不到 HDFS_DELEGATION_TOKEN

问题

安全模式下，为什么在缓存中找不到 HDFS_DELEGATION_TOKEN？

回答

在 MapReduce 中，默认情况下，任务完成之后，HDFS_DELEGATION_TOKEN 将会被删除。因此如果在下一个任务中再次使用 HDFS_DELEGATION_TOKEN，缓存中将会找不到 HDFS_DELEGATION_TOKEN。

为了能够在随后的工作中再次使用同一个 Token，为 MapReduce 任务配置参数。当参数为 false 时，用户能够再次使用同一个 Token。

```
jobConf.setBoolean("mapreduce.job.complete.cancel.delegation.tokens", false);
```

19.9.5 如何在提交 MapReduce 任务时设置任务优先级

问题

如何在提交 MapReduce 任务时设置任务优先级？

回答

当您在客户端提交 MapReduce 任务时，可以在命令行中增加“-Dmapreduce.job.priority=<priority>”参数来设置任务优先级。格式如下：

yarn jar *<jar> [mainClass]* ***-Dmapreduce.job.priority=****<priority> [path1] [path2]*

命令行中参数含义为：

- <jar>：指定需要运行的 jar 包名称。
- [mainClass]：指 jar 包应用工程中的类得 main 方法。
- <priority>：指定任务的优先级，其取值可为：VERY_HIGH、HIGH、NORMAL、LOW、VERY_LOW。
- [path1]：指数据输入路径。
- [path2]：指数据输出路径。

例如，将“/opt/client/HDFS/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples*.jar”包设置为高优先级任务。

yarn jar /opt/client/HDFS/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples*.jar wordcount -Dmapreduce.job.priority=VERY_HIGH /DATA.txt /out/

19.9.6 MapReduce 任务运行失败，ApplicationMaster 出现物理内存溢出异常

问题

HBase bulkload 任务有 210000 个 map 和 10000 个 reduce，MapReduce 任务运行失败，ApplicationMaster 出现物理内存溢出异常。

```
For more detailed output, check the application tracking page:https://bigdata-55:8090/cluster/app/application_1449841777199_0003
Then click on links to logs of each attempt.
Diagnostics: Container [pid=21557,containerID=container_1449841777199_0003_02_000001] is running beyond physical memory limits
Current usage: 1.0 GB of 1 GB physical memory used; 3.6 GB of 5 GB virtual memory used. Killing container.
Dump of the process-tree for container_1449841777199_0003_02_000001 :
|- PID PPID PGRPID SESSID CMD_NAME USER_MODE_TIME(MILLIS) SYSTEM_TIME(MILLIS) VMEM_USAGE(BYTES) RSSMEM_USAGE(PAGES) FULL_CMD_LINE
|- 21584 21557 21557 21557 (java) 12342 1627 3871748096 271331 /opt/xxx/Bigdata/jdk1.8.0_51//bin/java
```

```
-
Djava.io.tmpdir=/srv/BigData/hadoop/data1/nm/localdir/usercache/hbase/appcache/appl
ication_1449841777199_0003/container_1449841777199_0003_02_000001/tmp -
Dlog4j.configuration=container-log4j.properties
-
Dyarn.app.container.log.dir=/srv/BigData/hadoop/data1/nm/containerlogs/application_
1449841777199_0003/container_1449841777199_0003_02_000001 -
Dyarn.app.container.log.filesize=0 -Dhadoop.root.logger=INFO,CLA
-Dhadoop.root.logfile=syslog -Xmx784m
org.apache.hadoop.mapreduce.v2.app.MRAppMaster
|- 21557 21547 21557 21557 (bash) 0 0 13074432 368 /bin/bash -c
/opt/xxx/Bigdata/jdk1.8.0_51//bin/java
-
Djava.io.tmpdir=/srv/BigData/hadoop/data1/nm/localdir/usercache/hbase/appcache/appl
ication_1449841777199_0003/container_1449841777199_0003_02_000001/tmp -
Dlog4j.configuration=container-log4j.properties
-
Dyarn.app.container.log.dir=/srv/BigData/hadoop/data1/nm/containerlogs/application_
1449841777199_0003/container_1449841777199_0003_02_000001 -
Dyarn.app.container.log.filesize=0 -Dhadoop.root.logger=INFO,CLA
-Dhadoop.root.logfile=syslog -Xmx784m
org.apache.hadoop.mapreduce.v2.app.MRAppMaster
1>/srv/BigData/hadoop/data1/nm/containerlogs/application_1449841777199_0003/contain
er 1449841777199 0003 02 000001/stdout
2>/srv/BigData/hadoop/data1/nm/containerlogs/application 1449841777199 0003/contain
er 1449841777199 0003 02 000001/stderr
Container killed on request. Exit code is 143
Container exited with a non-zero exit code 143
Failing this attempt. Failing the application.
```

回答

这是性能规格的问题，MapReduce 任务运行失败的根本原因是由于 ApplicationMaster 的内存溢出导致的，即物理内存溢出导致被 NodeManager kill。

解决方案：

将 ApplicationMaster 的内存配置调大，在客户端“mapred-site.xml”配置文件中优化如下参数：

- “yarn.app.mapreduce.am.resource.mb”
- “yarn.app.mapreduce.am.command-opts”，该参数中-Xmx 值建议为 0.8*“yarn.app.mapreduce.am.resource.mb”

参考规格：

ApplicationMaster 配置如下时，可以同时支持并发 Container 数为 2.4 万个。

- “yarn.app.mapreduce.am.resource.mb”=2048
- “yarn.app.mapreduce.am.command-opts”该参数中-Xmx=1638m

19.9.7 MapReduce JobHistoryServer 服务地址变更后，为什么运行完的 MapReduce 作业信息无法通过 ResourceManager Web UI 页面的 Tracking URL 打开

问题

MapReduce JobHistoryServer 服务地址变更后，为什么运行完的 MapReduce 作业无法通过 ResourceManager Web UI 页面打开？

回答

JobHistoryServer 地址（mapreduce.jobhistory.address / mapreduce.jobhistory.webapp.<https.>address）是 MapReduce 参数，MapReduce 客户端提交作业时，会将此地址随任务一起提交给 ResourceManager。ResourceManager 在作业完成后，将此参数作为查看作业历史信息的跳转地址保存在 RMStateStore 中。

JobHistoryServer 服务地址变更后，需要将新的服务地址及时更新到 MapReduce 客户端配置文件中，否则，新运行的作业在查看作业历史信息时，仍然会指向原 JobHistoryServer 地址，导致无法正常跳转到作业历史信息页面。服务地址变更前运行的 MapReduce 作业，由于其跳转信息已经保存在 RMStateStore 中，无法变更，因此从 ResourceManager Web UI 页面是无法进行正常跳转的，但可以直接访问新的 JobHistoryServer 服务地址进行查找，作业信息不会丢失。

19.9.8 多个 NameService 环境下，运行 MapReduce 任务失败

问题

多个 NameService 环境下，运行使用 viewFS 功能的 MapReduce 或 YARN 任务失败。

回答

当使用 viewFS 时，只有在 viewFS 中挂载的目录才能被访问到。所以最可能的原因是配置的路径没有在 viewFS 的挂载点上。例如：

```
<property>
<name>fs.defaultFS</name>
<value>viewfs://ClusterX/</value>
</property>
<property>
<name>fs.viewfs.mounttable.ClusterX.link./folder1</name>
<value>hdfs://NS1/folder1</value>
</property>
<property>
<name>fs.viewfs.mounttable.ClusterX.link./folder2</name>
<value>hdfs://NS2/folder2</value>
</property>
```

对于依赖 HDFS 的 MR 配置中，需要使用已挂载的目录。

错误示例：

```
<property>
<name>yarn.app.mapreduce.am.staging-dir</name>
<value>/tmp/hadoop-yarn/staging</value>
</property>
```

根目录（/）在 viewFS 中是无法访问的。

正确示例：

```
<property>
<name>yarn.app.mapreduce.am.staging-dir</name>
<value>/folder1/tmp/hadoop-yarn/staging</value>
</property>
```

19.9.9 基于分区的任务黑名单

问题

Map&Reduce 任务失败，并且故障节点数与集群总节点数的比值低于“yarn.resourcemanager.am-scheduling.node-blacklisting-disable-threshold”配置的黑名单阈值，为什么 Map&Reduce 任务故障节点没有加入黑名单？

回答

当集群中有超过阈值的节点都被加入黑名单时，黑名单会释放这些节点，其中阈值为故障节点数与集群总节点数的比值。现在每个节点都有其标签表达式，黑名单阈值应根据有效节点标签表达式关联的节点数进行计算，其值为故障节点数与有效节点标签表达式关联的节点数的比值。

假设集群中有 100 个节点，其中有 10 个节点为有效节点标签表达式关联的节点（labelA）。其中所有有效节点标签表达式关联的节点都已经故障，黑名单节点释放阈值默认值为 0.33，按照传统的计算方式，10/100=0.1，远小于该阈值。这就造成这 10 个节点永远无法得到释放，Map&Reduce 任务一直无法获取节点，应用程序无法正常运行。实际需要根据与 Map&Reduce 任务的有效节点关联的节点总数进行计算，即 10/10=1，大于黑名单节点释放阈值，节点被释放。

因此即使故障节点数与集群总节点数的比值没有超过阈值，也存在黑名单将这些节点释放的情况。

20 使用 Oozie

20.1 从零开始使用 Oozie

Oozie 是一个基于工作流引擎的开源框架，能够提供对 Hadoop 作业的任务调度与协调。

Oozie 支持提交多种类型任务，例如 Hive、Spark2x、Loader、Mapreduce、Java、DistCp、Shell、HDFS、SSH、SubWorkflow、Streaming、定时任务等。

本章节指导用户通过使用 Oozie 客户端提交 MapReduce 任务。

前提条件

已安装客户端。例如安装目录为“/opt/client”，以下操作的客户端目录只是举例，请根据实际安装目录修改。

操作步骤

步骤 1 以客户端安装用户，登录安装客户端的节点。

步骤 2 执行以下命令，切换到客户端安装目录。假如客户端安装目录为：/opt/client，请根据实际安装目录修改。

cd /opt/client

步骤 3 执行以下命令配置环境变量。

source bigdata_env

步骤 4 判断集群认证模式。

- 安全模式，执行以下命令进行用户认证。*UserOozie* 为提交任务的用户。

  kinit *UserOozie*
- 普通模式，执行步骤 5。

步骤 5 上传 Oozie 配置文件以及 Jar 包至 HDFS：

hdfs dfs -mkdir /user/*UserOozie*

hdfs dfs -put -f /opt/client/Oozie/oozie-client-*/examples /user/*UserOozie*/

说明

- “/opt/client/”为客户端安装目录，请根据实际安装目录修改。
- UserOozie 为提交任务的用户。

步骤 6 修改任务执行配置文件：

cd /opt/client/Oozie/oozie-client-*/examples/apps/map-reduce/

vi job.properties

```
nameNode=hdfs://hacluster
resourceManager=10.64.35.161:8032（10.64.35.161 为 Yarn resourceManager（Active）节点业务平面 IP；8032 为 yarn.resourcemanager.port）
queueName=default
examplesRoot=examples
user.name=admin
oozie.wf.application.path=${nameNode}/user/${user.name}/${examplesRoot}/apps/map-reduce #hdfs 上传路径
outputDir=map-reduce
oozie.wf.rerun.failnodes=true
```

步骤 7 运行 oozie 任务：

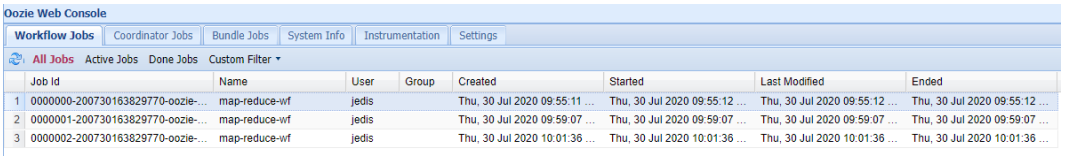
oozie job -oozie https://*oozie 角色的主机名*:21003/oozie/ -config job.properties -run

```
[root@kwephispra44947 map-reduce]# oozie job -oozie
https://kwephispra44948:21003/oozie/ -config job.properties -run
......
job: 0000000-200730163829770-oozie-omm-W
```

步骤 8 登录 FusionInsight Manager。具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本）。

步骤 9 选择“集群 > *待操作集群的名称* > 服务 > Oozie ”，单击“oozie WebUI”后的超链接进入 Oozie 页面，在 Oozie 的 WebUI 上查看任务运行结果。

图20-1 任务运行结果



----结束

20.2 使用 Oozie 客户端

操作场景

该任务指导用户在运维场景或业务场景中使用 Oozie 客户端。

前提条件

- 已安装客户端。例如安装目录为“/opt/client”，以下操作的客户端目录只是举例，请根据实际安装目录修改。
- 各组件业务用户由系统管理员根据业务需要创建。安全模式下，“机机”用户需要下载 keytab 文件。“人机”用户第一次登录时需修改密码。

使用 Oozie 客户端

步骤 1 以客户端安装用户，登录安装客户端的节点。

步骤 2 执行以下命令，切换到客户端安装目录，该操作的客户端目录只是举例，请根据实际安装目录修改。

cd /opt/client

步骤 3 执行以下命令配置环境变量。

source bigdata_env

步骤 4 判断集群认证模式。

- 安全模式，执行以下命令进行用户认证。*exampleUser* 为提交任务的用户名。

  kinit *exampleUser*
- 普通模式，执行步骤 5。

步骤 5 配置 Hue。

1. spark2x 环境配置（如果不涉及 spark2x 任务，可以跳过此步骤）：

   hdfs dfs -put /opt/client/Spark2x/spark/jars/*.jar /user/oozie/share/lib/spark2x/

   当 HDFS 目录“/user/oozie/share”中的 Jar 包发生变化时，需要重启 Oozie 服务。
2. 上传 Oozie 配置文件以及 Jar 包至 HDFS：

   hdfs dfs -mkdir /user/*exampleUser*

   hdfs dfs -put -f /opt/client/Oozie/oozie-client-*/examples /user/*exampleUser*/

说明

- *exampleUser* 为提交任务的用户名。
- 在提交任务的用户和非 job.properties 文件均无变更的前提下，客户端安装目录/Oozie/oozie-client-*/examples 目录一经上传 HDFS，后续可重复使用，无需多次提交。
- 解决 Spark 和 Yarn 关于 jetty 的 jar 冲突。

  hdfs dfs -rm -f /user/oozie/share/lib/spark/jetty-all-9.2.22.v20170606.jar
- 普通模式下，上传过程如果遇到“Permission denied”的问题，可执行以下命令进行处理。

  su - omm

  source /opt/client/bigdata_env

  hdfs dfs -chmod -R 777 /user/oozie

  exit

----结束

20.3 开启 Oozie HA 机制

操作场景

Oozie 多个节点同时提供服务的时候，通过 ZooKeeper 来提供高可用（HA）功能，防止单节点故障以及多节点同时处理一个任务。

说明

本章节内容仅适用于 MRS 3.1.2 及之后版本。

对系统影响

操作过程中需要重启 Oozie 服务。重启过程中，Oozie 服务无法提供服务。

前提条件

- 已安装 Oozie、ZooKeeper 服务，且服务正常运行。
- 没有任务正在运行。
- 如果当前集群不是安装最新的版本包，需要从“$BIGDATA_HOME/FusionInsight_Porter_*x.x.x*/install/FusionInsight-Oozie-*x.x.x*/oozie-*x.x.x*/embedded-oozie-server/webapp/WEB-INF/lib”路径拷贝“curator-x-discovery-*x.x.x*.jar”包到“$BIGDATA_HOME/FusionInsight_Porter_*x.x.x*/install/FusionInsight-Oozie-*x.x.x*/oozie-*x.x.x*/lib”目录下。

操作步骤

步骤 1 在 FusionInsight Manager 界面选择“集群 > 服务 > Oozie > 配置 > 全部配置”，在“自定义”的“oozie.site.configs”参数中添加如下四个配置项。修改完成后单击“保存”，在弹框中单击“确定”保存配置。

名称	值	参数说明
oozie.services.ext	org.apache.oozie.service.ZKLocksService,org.apache.oozie.service.ZKXLogStreamingService,org.apache.oozie.service.ZKJobsConcurrencyService,org.apache.oozie.service.ZKUUIDService	HA 启用的功能
oozie.zookeeper.connection.string	*ZooKeeper 实例的业务 IP:端口*（多个地址以逗号隔开）	ZooKeeper 连接信息
oozie.zookeeper.namespace	oozie	Oozie 在 ZooKeeper 的路径
oozie.zookeeper.secure	安全集群：true 普通集群：无需配置该参数	ZooKeeper 是否启用 kerberos

步骤 2 在 Oozie 的“概览”界面，选择右上角“更多 > 重启服务”，重启 Oozie 集群。

----结束

20.4 使用 Oozie 客户端提交作业

20.4.1 提交 Hive 任务

操作场景

该任务指导用户在使用 Oozie 客户端提交 Hive 任务

Hive 任务有如下类型：

- Hive 作业
  使用 JDBC 方式连接的 Hive 作业。
- Hive2 作业
  使用 Beeline 方式连接的 Hive 作业。

本文以使用 Oozie 客户端提交 Hive 作业为例介绍。

说明

- 使用 Oozie 客户端提交 Hive2 作业与提交 Hive 作业操作步骤一致，只需将操作步骤中对应路径的“/Hive”改成“/Hive2”即可。

  例如，Hive 作业运行目录“/opt/client/Oozie/oozie-client-*/examples/apps/hive/”，则 Hive2 对应的运行目录为“/opt/client/Oozie/oozie-client-*/examples/apps/hive2/”。
- 建议下载使用最新版本的客户端。

前提条件

- Hive 和 Oozie 组件及客户端已经安装，并且正常运行。
- 已创建或获取访问 Oozie 服务的人机用户帐号及密码。

说明

- 该用户需要从属于 hadoop、supergroup、hive 组，同时添加 Oozie 的角色操作权限。若使用 Hive 多实例，该用户还需要从属于具体的 Hive 实例组，如 hive3。
- 用户同时还需要至少有 manager_viewer 权限的角色。

- 获取运行状态的 Oozie 服务器（任意实例）URL，如“https://10.1.130.10:21003/oozie”。
- 获取运行状态的 Oozie 服务器主机名，如“10-1-130-10”。
- 获取 Yarn ResourceManager 主节点 IP，如 10.1.130.11。

操作步骤

步骤 1 以客户端安装用户，登录安装 Oozie 客户端的节点。

步骤 2 执行以下命令，获取安装环境信息。其中“/opt/client/”为客户端安装路径，该操作的客户端目录只是举例，请根据实际安装目录修改。

source /opt/client/bigdata_env

步骤 3 判断集群认证模式。

- 安全模式，执行 **kinit** 命令进行用户认证。

  例如，使用 **oozieuser** 用户进行认证。

  kinit oozieuser
- 普通模式，执行步骤 4。

步骤 4 执行以下命令，进入样例目录。

cd /opt/client/Oozie/oozie-client-*/examples/apps/hive/

该目录下需关注文件如表 20-1 所示。

表20-1 文件说明

文件名称	描述
hive-site.xml	Hive 任务的配置文件。
job.properties	工作流的参数变量定义文件。
script.q	Hive 任务的 SQL 脚本。
workflow.xml	工作流的规则定制文件。

步骤 5 执行以下命令，编辑“job.properties”文件。

vi job.properties

修改如下内容：

更改“userName”的参数值为提交任务的人机用户名，例如“userName=oozieuser”。

步骤 6 执行 **oozie job** 命令，运行工作流文件。

oozie job -oozie https://*oozie 角色的主机名*:21003/oozie/ -config job.properties -run

说明

- 命令参数解释如下：

  -oozie 实际执行任务的 Oozie 服务器 URL

  -config 工作流属性文件

  -run 运行工作流

- 执行完工作流文件，显示 job id 表示提交成功，例如：job: 0000021-140222101051722-oozie-omm-W。登录 Oozie 管理页面，查看运行情况。

  使用 **oozieuser** 用户，登录 Oozie WebUI 页面：https://*oozie 角色的 ip 地址*:21003/oozie 。

  Oozie 的 WebUI 界面中，可在页面表格根据 jobid 查看已提交的工作流信息。

----结束

20.4.2 提交 Spark2x 任务

操作场景

该任务指导用户在使用 Oozie 客户端提交 Spark2x 任务。

说明

请下载使用最新版本的客户端。

前提条件

- Spark2x 和 Oozie 组件安装完成且运行正常，客户端安装成功。

  如果当前客户端为旧版本，需要重新下载和安装客户端。
- 已创建或获取访问 Oozie 服务的人机用户帐号及密码。

说明

- 该用户需要从属于 hadoop、supergroup、hive 组，同时添加 Oozie 的角色操作权限。若使用 Hive 多实例，该用户还需要从属于具体的 Hive 实例组，如 hive3。
- 用户同时还需要至少有 manager_viewer 权限的角色。

- 获取运行状态的 Oozie 服务器（任意实例）URL，如“https://*10.1.130.10:21003*/oozie”。
- 获取运行状态的 Oozie 服务器主机名，如“*10-1-130-10*”。
- 获取 Yarn ResourceManager 主节点 IP，如“*10.1.130.11*”。

操作步骤

步骤 1 以客户端安装用户登录安装 Oozie 客户端的节点。

步骤 2 执行以下命令，获取安装环境信息。其中“/opt/client/”为客户端安装路径，该操作的客户端目录只是举例，请根据实际安装目录修改。

source /opt/client/bigdata_env

步骤 3 判断集群认证模式。

- 安全模式，执行 **kinit** 命令进行用户认证。

  例如，使用 **oozieuser** 用户进行认证。

  kinit oozieuser
- 普通模式，执行步骤 4。

步骤 4 执行以下命令，进入样例目录。

cd /opt/client/Oozie/oozie-client-*/examples/apps/spark2x/

该目录下需关注文件如表 20-2 所示。

表20-2 文件说明

文件名称	描述
job.properties	工作流的参数变量定义文件。
workflow.xml	工作流的规则定制文件。
lib	工作流运行依赖的 jar 包目录。

步骤 5 执行以下命令，编辑“job.properties”文件。

vi job.properties

修改如下内容：

更改“userName”的参数值为提交任务的人机用户名，例如“userName=oozieuser”。

步骤 6 执行 **oozie job** 命令，运行工作流文件。

oozie job -oozie https://*oozie 角色的主机名*:21003/oozie/ -config job.properties -run

说明

- 命令参数解释如下：

  -oozie 实际执行任务的 Oozie 服务器 URL

  -config 工作流属性文件

  -run 运行工作流

- 执行完工作流文件，显示“job id”表示提交成功，例如“job: 0000021-140222101051722-oozie-omm-W”。登录 Oozie 管理页面，查看运行情况。

  使用 **oozieuser** 用户，登录 Oozie WebUI 页面：https://*oozie 角色的 ip 地址*:21003/oozie 。

  Oozie 的 WebUI 界面中，可在页面表格根据“job id”查看已提交的工作流信息。

----结束

20.4.3 提交 Loader 任务

操作场景

该任务指导用户在使用 Oozie 客户端提交 Loader 任务。

说明

请下载使用最新版本的客户端。

前提条件

- Loader 和 Oozie 组件及客户端已经安装，并且正常运行。
- 已创建或获取访问 Oozie 服务的人机用户帐号及密码。

  说明

  - 该用户需要从属于 hadoop、supergroup、hive 组，同时添加 Oozie 的角色操作权限。若使用 Hive 多实例，该用户还需要从属于具体的 Hive 实例组，如 hive3。
  - 用户同时还需要至少有 manager_viewer 权限的角色。
- 获取运行状态的 Oozie 服务器（任意实例）URL，如“https://10.1.130.10:21003/oozie”。
- 获取运行状态的 Oozie 服务器主机名，如“10-1-130-10”。
- 获取 Yarn ResourceManager 主节点 IP，如 10.1.130.11。
- 创建需要调度的 Loader 作业，并获取该作业 ID。

操作步骤

步骤 1 以客户端安装用户，登录安装 Oozie 客户端的节点。

步骤 2 执行以下命令，获取安装环境信息。其中“/opt/client/”为客户端安装路径，该操作的客户端目录只是举例，请根据实际安装目录修改。

source /opt/client/bigdata_env

步骤 3 判断集群认证模式。

- 安全模式，执行 **kinit** 命令进行用户认证。

  例如，使用 **oozieuser** 用户进行认证。

  kinit oozieuser
- 普通模式，执行步骤 4。

步骤 4 执行以下命令，进入样例目录。

cd /opt/client/Oozie/oozie-client-*/examples/apps/sqoop/

该目录下需关注文件如表 20-3 所示。

表20-3 文件说明

文件名称	描述
job.properties	工作流的参数变量定义文件。
workflow.xml	工作流的规则定制文件。

步骤 5 执行以下命令，编辑“job.properties”文件。

vi job.properties

修改如下内容：

更改“userName”的参数值为提交任务的人机用户名，例如“userName=oozieuser”。

步骤 6 执行以下命令，编辑“workflow.xml”文件。

vi workflow.xml

修改如下内容：

“command”的值修改为需要调度的已有 Loader 作业 ID，例如 1。

将“workflow.xml”文件上传至 "job.properties" 文件中的 HDFS 路径。

hdfs dfs -put -f workflow.xml /user/userName/examples/apps/sqoop

步骤 7 执行 **oozie job** 命令，运行工作流文件。

oozie job -oozie https://*oozie 角色的主机名*:21003/oozie/ -config job.properties -run

说明

- 命令参数解释如下：

  -oozie 实际执行任务的 Oozie 服务器 URL

  -config 工作流属性文件

  -run 运行工作流

- 执行完工作流文件，显示 job id 表示提交成功，例如：job: 0000021-140222101051722-oozie-omm-W。登录 Oozie 管理页面，查看运行情况。

  使用 **oozieuser** 用户，登录 Oozie WebUI 页面：https://*oozie 角色的 ip 地址*:21003/oozie 。

  Oozie 的 WebUI 界面中，可在页面表格根据 jobid 查看已提交的工作流信息。

----结束

20.4.4 提交 DistCp 任务

操作场景

该任务指导用户在使用 Oozie 客户端提交 DistCp 任务。

说明

请下载使用最新版本的客户端。

前提条件

- HDFS 和 Oozie 组件安装完成且运行正常，客户端安装成功。
  如果当前客户端为旧版本，需要重新下载和安装客户端。
- 已创建或获取访问 Oozie 服务的人机用户帐号及密码。

说明

- 该用户需要从属于 hadoop、supergroup、hive 组，同时添加 Oozie 的角色操作权限。若使用 Hive 多实例，该用户还需要从属于具体的 Hive 实例组，如 hive3。

- 用户同时还需要至少有 manager_viewer 权限的角色。

- 已获取运行状态的 Oozie 服务器（任意实例）URL，如“https://*10.1.130.10:21003*/oozie”。
- 已获取运行状态的 Oozie 服务器主机名，如“*10-1-130-10*”。
- 已获取 Yarn ResourceManager 主节点 IP，如“*10.1.130.11*”。

操作步骤

步骤 1 以客户端安装用户登录安装 Oozie 客户端的节点。

步骤 2 执行以下命令，获取安装环境信息。其中“/opt/client/”为客户端安装路径，该操作的客户端目录只是举例，请根据实际安装目录修改。

source /opt/client/bigdata_env

步骤 3 判断集群认证模式。

- 安全模式，执行 **kinit** 命令进行用户认证。

  例如，使用 **oozieuser** 用户进行认证。

  kinit oozieuser
- 普通模式，执行步骤 4。

步骤 4 执行以下命令，进入样例目录。

cd /opt/client/Oozie/oozie-client-*/examples/apps/distcp/

该目录下需关注文件如表 20-4 所示。

表20-4 文件说明

文件名称	描述
job.properties	工作流的参数变量定义文件。
workflow.xml	工作流的规则定制文件。

步骤 5 执行以下命令，编辑“job.properties”文件。

vi job.properties

修改如下内容：

更改“userName”的参数值为提交任务的人机用户名，例如“userName=oozieuser”。

步骤 6 是否是跨安全集群的 DistCp。

- 是，执行步骤步骤 7。
- 否，则执行步骤步骤 9。

步骤 7 对两个集群进行跨 Manager 集群互信。

步骤 8 备份并且修改 workflow.xml 的文件内容，命令如下：

cp workflow.xml workflow.xml.bak

vi workflow.xml

修改以下内容：

```
<workflow-app xmlns="uri:oozie:workflow:1.0" name="distcp-wf">
   <start to="distcp-node"/>
   <action name="distcp-node">
      <distcp xmlns="uri:oozie:distcp-action:1.0">
         <resource-manager>${resourceManager}</resource-manager>
         <name-node>${nameNode}</name-node>
         <prepare>
            <delete
path="hdfs://target_ip:target_port/user/${userName}/${examplesRoot}/output-
data/${outputDir}"/>
         </prepare>
         <configuration>
            <property>
               <name>mapred.job.queue.name</name>
               <value>${queueName}</value>
            </property>
            <property>
               <name>oozie.launcher.mapreduce.job.hdfs-servers</name>

<value>hdfs://source_ip:source_port,hdfs://target_ip:target_port</value>
            </property>
         </configuration>
         <arg>${nameNode}/user/${userName}/${examplesRoot}/input-
data/text/data.txt</arg>

<arg>hdfs://target_ip:target_port/user/${userName}/${examplesRoot}/output-
data/${outputDir}/data.txt</arg>
         </distcp>
      <ok to="end"/>
      <error to="fail"/>
   </action>
   <kill name="fail">
      <message>DistCP failed, error
message[${wf:errorMessage(wf:lastErrorNode())}]</message>
   </kill>
   <end name="end"/>
</workflow-app>
```

其中“target_ip:target_port”为另一个互信集群的 HDFS active namenode 地址，例如：10.10.10.233:25000。

“source_ip:source_port ”为源集群的 HDFS active namenode 地址，例如：10.10.10.223:25000。

两个 IP 地址和端口都需要根据自身的集群实际情况修改。

步骤 9 执行 **oozie job** 命令，运行工作流文件。

oozie job -oozie https://*oozie 角色的主机名***:21003/oozie/ -config job.properties -run**

说明

- 命令参数解释如下：

  -oozie　　实际执行任务的 Oozie 服务器 URL

  -config　　工作流属性文件

  -run　　运行工作流

- 执行完工作流文件，显示“job id”表示提交成功，例如“job: 0000021-140222101051722-oozie-omm-W”。登录 Oozie 管理页面，查看运行情况。

  使用 **oozieuser** 用户，登录 Oozie WebUI 页面：https://*oozie 角色的 ip 地址*:21003/oozie 。

  Oozie 的 WebUI 界面中，可在页面表格根据“job id”查看已提交的工作流信息。

----结束

20.4.5 提交其它任务

操作场景

除了 Hive、Spark2x、Loader 任务，也支持使用 Oozie 客户端提交 MapReduce、Java、Shell、HDFS、SSH、SubWorkflow、Streaming、定时等任务。

说明

请下载使用最新版本的客户端。

前提条件

- Oozie 组件及客户端已经安装，并且正常运行。
- 已创建或获取访问 Oozie 服务的人机用户帐号及密码。

说明

- Shell 任务：

  该用户需要从属于 hadoop、supergroup 组，添加 Oozie 的角色操作权限，并确保 Shell 脚本在每个 nodemanager 节点都有执行权限。

- SSH 任务：

  该用户需要从属于 hadoop、supergroup 组，添加 Oozie 的角色操作权限，并完成互信配置。

- 其他任务：

  该用户需要从属于 hadoop、supergroup 组，添加 Oozie 的角色操作权限，并具备对应任务类型所需的权限。

- 用户同时还需要至少 manager_viewer 权限的角色。

- 获取运行状态的 Oozie 服务器（任意实例）URL，如“https://10.1.130.10:21003/oozie”。
- 获取运行状态的 Oozie 服务器主机名，如“10-1-130-10”。
- 获取 Yarn ResourceManager 主节点 IP，如 10.1.130.11。

操作步骤

步骤 1 以客户端安装用户，登录安装 Oozie 客户端的节点。

步骤 2 执行以下命令，获取安装环境信息。其中“/opt/client/”为客户端安装路径，该操作的客户端目录只是举例，请根据实际安装目录修改。

source /opt/client/bigdata_env

步骤 3 判断集群认证模式。

- 安全模式，执行 **kinit** 命令进行用户认证。

  例如，使用 **oozieuser** 用户进行认证。

  kinit oozieuser
- 普通模式，执行步骤 4。

步骤 4 根据提交任务类型，进入对应样例目录。

表20-5 样例目录列表

任务类型	样例目录
Mapreduce 任务	*客户端安装目录*/Oozie/oozie-client-*/examples/apps/map-reduce
Java 任务	*客户端安装目录*/Oozie/oozie-client-*/examples/apps/java-main
Shell 任务	*客户端安装目录*/Oozie/oozie-client-*/examples/apps/shell
Streaming 任务	*客户端安装目录*/Oozie/oozie-client-*/examples/apps/streaming
SubWorkflow 任务	*客户端安装目录*/Oozie/oozie-client-*/examples/apps/subwf
SSH 任务	*客户端安装目录*/Oozie/oozie-client-*/examples/apps/ssh
定时任务	*客户端安装目录*/Oozie/oozie-client-*/examples/apps/cron

说明

其他任务样例中已包含 HDFS 任务样例。

样例目录下需关注文件如表 20-6 所示。

表20-6 文件说明

文件名称	描述
job.properties	工作流的参数变量定义文件。
workflow.xml	工作流的规则定制文件。
lib	工作流运行依赖的 jar 包目录。
coordinator.xml	“cron”目录下存在，定时任务配置文

文件名称	描述
	件，用于设置定时策略。
oozie_shell.sh	“shell”目录下存在，提交 Shell 任务需要的 Shell 脚本文件。

步骤 5 执行以下命令，编辑“job.properties”文件。

vi job.properties

修改如下内容：

更改“userName”的参数值为提交任务的人机用户名，例如“userName=oozieuser”。

步骤 6 执行 **oozie job** 命令，运行工作流文件。

oozie job -oozie https://*oozie 角色的主机名***:21003/oozie -config** *job.properties 文件所在路径* **-run**

例如：

oozie job -oozie https://10-1-130-10:21003/oozie -config

/opt/client/Oozie/oozie-client-*/examples/apps/map-reduce/job.properties -run

说明

- 命令参数解释如下：

  -oozie 实际执行任务的 Oozie 服务器 URL

  -config 工作流属性文件

  -run 运行工作流

- 执行完工作流文件，显示 job id 表示提交成功，例如：job: 0000021-140222101051722-oozie-omm-W。登录 Oozie 管理页面，查看运行情况。

  使用 **oozieuser** 用户，登录 Oozie WebUI 页面：https://*oozie 角色的 ip 地址*:21003/oozie 。

  Oozie 的 WebUI 界面中，可在页面表格根据 jobid 查看已提交的工作流信息。

----结束

20.5 使用 Hue 提交 Oozie 作业

20.5.1 创建工作流

操作场景

用户通过 Hue 管理界面可以进行提交 Oozie 作业，提交作业之前，首先需要创建一个工作流。

前提条件

使用 Hue 提交 Oozie 作业之前，需要提前配置好 Oozie 客户端，并上传样例配置文件和 jar 至 HDFS 指定目录，具体操作请参考使用 Oozie 客户端章节。

操作步骤

步骤 1 准备一个具有对应组件操作权限的用户。

例如：使用 **admin** 用户登录 FusionInsight Manager，选择“系统 > 用户 > 添加用户”，创建一个“人机”用户“hueuser”，并加入“hive”、“hadoop”、“supergroup”组和“System_administrator”角色，主组为“hive”。

步骤 2 使用**步骤 1** 创建的用户登录 FusionInsight Manager（首次登录需要修改密码），选择“集群 > 服务 > Hue”，单击“Hue WebUI”右侧的链接，进入 Hue WebUI 界面。

步骤 3 在界面左侧导航栏单击  ，选择“Workflow”，打开 Workflow 编辑器。

步骤 4 单击“文档”后的下拉框选择“操作”， 在操作列表中选择需要创建的作业类型，将其拖到操作界面中即可。

不同类型作业提交请参考以下章节：

- 提交 Hive2 作业
- 提交 Spark2x 作业
- 提交 Java 作业
- 提交 Loader 作业
- 提交 Mapreduce 作业
- 提交 Sub workflow 作业
- 提交 Shell 作业
- 提交 HDFS 作业

- 提交 Streaming 作业
- 提交 Distcp 作业

----结束

20.5.2 提交 Workflow 工作流作业

20.5.2.1 提交 Hive2 作业

操作场景

该任务指导用户通过 Hue 界面提交 Hive2 类型的 Oozie 作业。

操作步骤

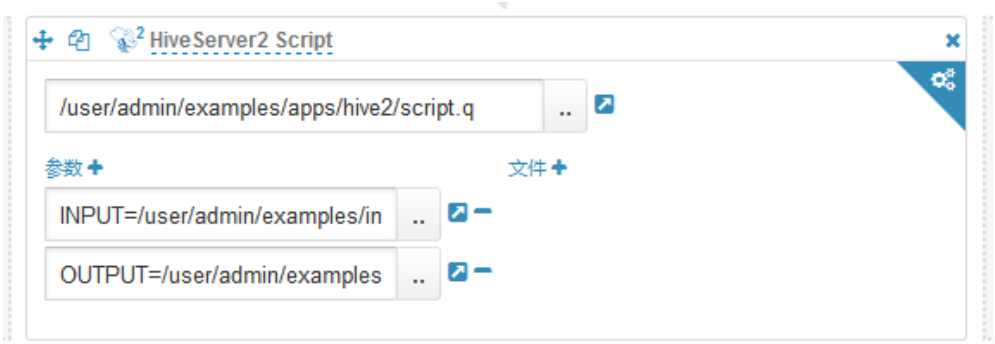
步骤 1 创建工作流，请参考创建工作流。

步骤 2 在工作流编辑页面，选择“HiveServer2 脚本”按钮  ，将其拖到操作区中。

步骤 3 在弹出的“HiveServer2 Script”窗口中配置 HDFS 上的脚本路径，例如“/user/admin/examples/apps/hive2/script.q”，然后单击“添加”。

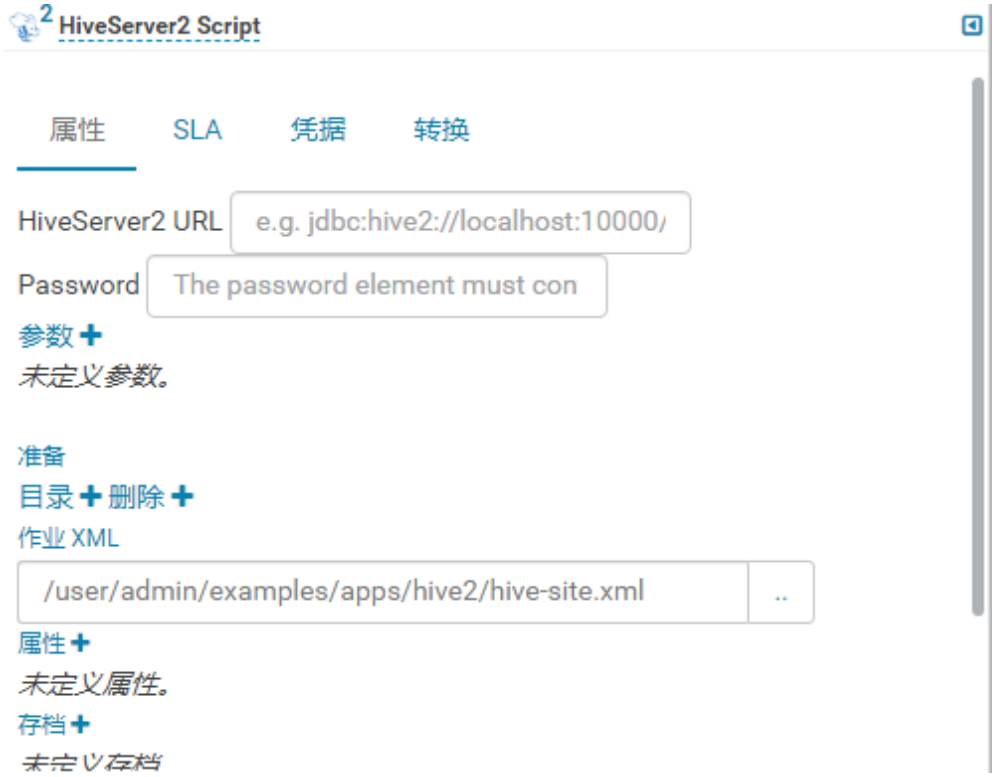
步骤 4 单击“参数+”，添加输入输出参数。

例如输入参数为“INPUT=/user/admin/examples/input-data/table”，输出参数为“OUTPUT=/user/admin/examples/output-data/hive2_workflow”。



步骤 5 单击右上角的配置按钮  。在打开的配置界面中，单击“删除+”，添加删除目录，例如“/user/admin/examples/output-data/hive2_workflow”。

步骤 6 配置“作业 XML”，例如配置为 hdfs 路径“/user/admin/examples/apps/hive2/hive-site.xml”。



说明

若以上的参数和值在使用过程中发生了修改，可在“Oozie 客户端安装目录/oozie-client-*/conf/hive-site.xml”文件中查询。

步骤 7 单击 Oozie 编辑器右上角的 。

保存前如果需要修改作业名称（默认为“My Workflow”），可以直接单击该名称进行修改，例如“Hive2-Workflow”。

步骤 8 保存完成后，单击 ，提交该作业。

作业提交后，可通过 Hue 界面查看作业的详细信息、日志、进度等相关内容。

----结束

20.5.2.2 提交 Spark2x 作业

操作场景

该任务指导用户通过 Hue 界面提交 Spark2x 类型的 Oozie 作业。

操作步骤

步骤 1 创建工作流，请参考创建工作流。

步骤 2 在工作流编辑页面，选择“Spark 程序”按钮  ，将其拖到操作区中。

步骤 3 在弹出的“Spark”窗口配置“Files”，例如“hdfs://hacluster/user/admin/examples/apps/spark2x/lib/oozie-examples.jar”。配置“jar/py name”，例如“oozie-examples.jar” ，配置完成后单击“添加”。

步骤 4 配置“Main class”的值。例如“org.apache.oozie.example.SparkFileCopy”。

步骤 5 单击“参数+”，添加输入输出相关参数。

例如添加：

- “hdfs://hacluster/user/admin/examples/input-data/text/data.txt”
- “hdfs://hacluster/user/admin/examples/output-data/spark_workflow”

步骤 6 在“Options list”文本框指定 spark 参数 ，例如“--conf spark.yarn.archive=hdfs://hacluster/user/spark2x/jars/8.1.0.1/spark-archive-2x.zip --conf spark.eventLog.enabled=true --conf spark.eventLog.dir=hdfs://hacluster/spark2xJobHistory2x”。

说明

此处版本号 8.1.0.1 为示例，具体以实际环境的版本号为准。

步骤 7 单击右上角的配置按钮  。配置“Spark Master”的值，例如“yarn-cluster”。配置“Mode”的值，例如“cluster”。

步骤 8 在打开的配置界面中，单击“删除+”，添加删除目录，例如“hdfs://hacluster/user/admin/examples/output-data/spark_workflow”。

步骤 9 单击“属性+”，添加 oozie 使用的 sharelib，左边文本框填写属性名称“oozie.action.sharelib.for.spark”，右边文本框填写属性值“spark2x”。

步骤 10 单击 Oozie 编辑器右上角的  。

保存前如果需要修改作业名称（默认为“My Workflow”），可以直接单击该名称进行修改，例如“Spark-Workflow”。

步骤 11 保存完成后，单击  ，提交该作业。

作业提交后，可通过 Hue 界面查看作业的详细信息、日志、进度等相关内容。

----结束

20.5.2.3 提交 Java 作业

操作场景

该任务指导用户通过 Hue 界面提交 Java 类型的 Oozie 作业。

操作步骤

步骤 1 创建工作流，请参考创建工作流。

步骤 2 在工作流编辑页面，选择“Java 程序”按钮  ，将其拖到操作区中。

步骤 3 在弹出的“Java program”窗口中配置“Jar name”的值，例如“/user/admin/examples/apps/java-main/lib/oozie-examples-5.1.0.jar”。配置“Main class”的值，例如“org.apache.oozie.example.DemoJavaMain”。然后单击“添加”。

步骤 4 单击 Oozie 编辑器右上角的  。

保存前如果需要修改作业名称（默认为“My Workflow”），可以直接单击该名称进行修改，例如“Java-Workflow”。

步骤 5 保存完成后，单击  ，提交该作业。

作业提交后，可通过 Hue 界面查看作业的详细信息、日志、进度等相关内容。

----结束

20.5.2.4 提交 Loader 作业

操作场景

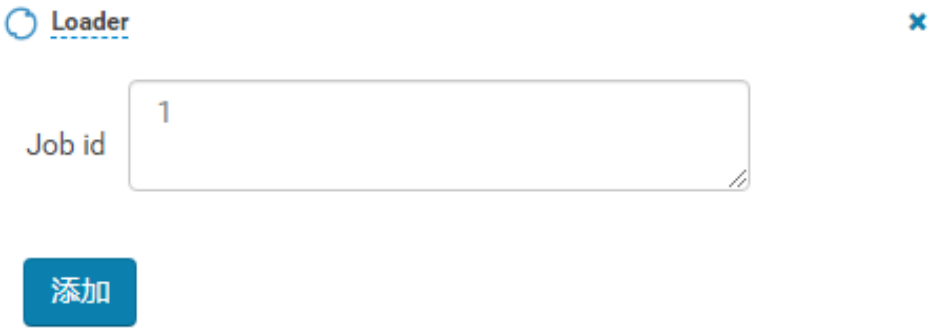
该任务指导用户通过 Hue 界面提交 Loader 类型的 Oozie 作业。

操作步骤

步骤 1 创建工作流，请参考创建工作流。

步骤 2 在工作流编辑页面，选择“Loader”按钮  ，将其拖到操作区中。

步骤 3 在弹出的“Loader”窗口中配置“Job id”的值，例如“1”。然后单击“添加”。



说明

“Job id”是需要编排的 Loader 作业的 ID 值，可从 Loader 页面获取。

创建需要调度的 Loader 作业，并获取该作业 ID，具体操作请参见使用 Loader 相关章节。

步骤 4 单击 Oozie 编辑器右上角的 。

保存前如果需要修改作业名称（默认为“My Workflow”），可以直接单击该名称进行修改，例如“Loader-Workflow”。

步骤 5 保存完成后，单击 ，提交该作业。

作业提交后，可通过 Hue 界面查看作业的详细信息、日志、进度等相关内容。

----结束

20.5.2.5 提交 Mapreduce 作业

操作场景

该任务指导用户通过 Hue 界面提交 Mapreduce 类型的 Oozie 作业。

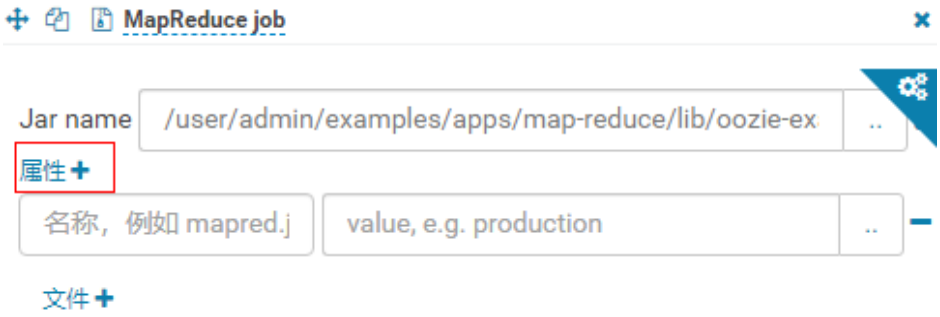
操作步骤

步骤 1 创建工作流，请参考创建工作流。

步骤 2 在工作流编辑页面，选择“MapReduce 作业”按钮 ，将其拖到操作区中。

步骤 3 在弹出的“MapReduce job”窗口中配置“Jar name”的值，例如“/user/admin/examples/apps/map-reduce/lib/oozie-examples-5.1.0.jar”。然后单击“添加”。

步骤 4 单击“属性+”，添加输入输出相关属性。



例如配置“mapred.input.dir”的值为“/user/admin/examples/input-data/text”，配置“mapred.output.dir”的值为“/user/admin/examples/output-data/map-reduce_workflow”。

步骤 5 单击右上角的配置按钮 。在打开的配置界面中，单击“删除+”，添加删除目录，例如“/user/admin/examples/output-data/map-reduce_workflow”。

步骤 6 单击 Oozie 编辑器右上角的 。

保存前如果需要修改作业名称（默认为“My Workflow”），可以直接单击该名称进行修改，例如“MapReduce-Workflow”。

步骤 7 保存完成后，单击  ，提交该作业。

作业提交后，可通过 Hue 界面查看作业的详细信息、日志、进度等相关内容。

----结束

20.5.2.6 提交 Sub workflow 作业

操作场景

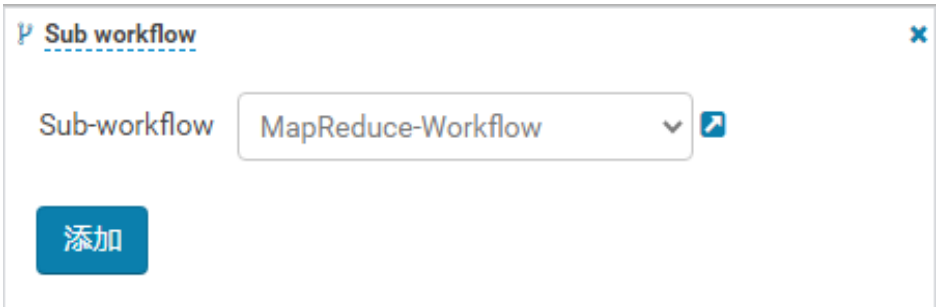
该任务指导用户通过 Hue 界面提交 Sub Workflow 类型的 Oozie 作业。

操作步骤

步骤 1 创建工作流，请参考创建工作流。

步骤 2 在工作流编辑页面，选择“子 Workflow”按钮  ，将其拖到操作区中。

步骤 3 在弹出的“Sub workflow”窗口中配置“Sub-workflow”的值，例如从下拉列表中选取“Java-Workflow”（这个值是已经创建好的工作流之一），然后单击“添加”。



步骤 4 单击 Oozie 编辑器右上角的  。

保存前如果需要修改作业名称（默认为“My Workflow”），可以直接单击该名称进行修改，例如“Subworkflow-Workflow”。

步骤 5 保存完成后，单击  ，提交该作业。

作业提交后，可通过 Hue 界面查看作业的详细信息、日志、进度等相关内容。

----结束

20.5.2.7 提交 Shell 作业

操作场景

该任务指导用户通过 Hue 界面提交 Shell 类型的 Oozie 作业。

操作步骤

步骤 1 创建工作流，请参考创建工作流。

步骤 2 在工作流编辑页面，选择“Shell”按钮  ，将其拖到操作区中。

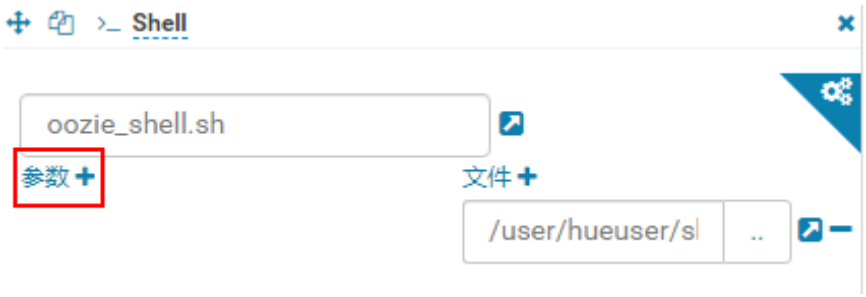
步骤 3 在弹出的“Shell”窗口中配置“Shell command”的值，例如“oozie_shell.sh”，然后单击“添加”。

步骤 4 单击“文件+”，添加 Shell 命令执行文件或 Oozie 样例执行文件，可以选择存储在 HDFS 的文件或本地文件。

- 若文件存储在 HDFS 上，选择“.sh”文件所在路径即可，例如“user/hueuser/shell/oozie_shell.sh”。

- 若选择本地文件，则需在“选择文件”界面，单击“上传文件”，上传本地文件，文件上传成功后，选择该文件即可。

步骤 5 如果执行的 Shell 文件需要传递参数，可单击“参数+”设置参数。



说明

传递参数的顺序需要和 Shell 脚本中保持一致。

步骤 6 单击 Oozie 编辑器右上角的 。

保存前如果需要修改作业名称（默认为“My Workflow”），可以直接单击该名称进行修改，例如“Shell-Workflow”。

步骤 7 保存完成后，单击 ，提交该作业。

作业提交后，可通过 Hue 界面查看作业的详细信息、日志、进度等相关内容。

说明

- 配置 Shell 命令为 Linux 指令时，请指定为原始指令，不要使用快捷键指令。例如：**ls -l** ,不要配置成 **ll**。可配置成 Shell 命令 **ls** ，参数添加一个 “-l”。
- Windows 上传 Shell 脚本到 HDFS 时，请保证 Shell 脚本的格式为 Unix，格式不正确会导致 Shell 作业提交失败。

----结束

20.5.2.8 提交 HDFS 作业

操作场景

该任务指导用户通过 Hue 界面提交 HDFS 类型的 Oozie 作业。

操作步骤

步骤 1 创建工作流，请参考创建工作流。

步骤 2 在工作流编辑页面，选择“Fs”按钮  ，将其拖到操作区中。

步骤 3 在弹出的“Fs”窗口中单击“添加”。

步骤 4 单击“CREATE DIRECTORY+”，添加待创建的 HDFS 目录。例如“/user/admin/examples/output-data/mkdir_workflow”和“/user/admin/examples/output-data/mkdir_workflow1”。

步骤 5 单击 Oozie 编辑器右上角的  。

保存前如果需要修改作业名称（默认为“My Workflow”），可以直接单击该名称进行修改，例如“HDFS-Workflow”。

步骤 6 保存完成后，单击  ，提交该作业。

作业提交后，可通过 Hue 界面查看作业的详细信息、日志、进度等相关内容。

----结束

20.5.2.9 提交 Streaming 作业

操作场景

该任务指导用户通过 Hue 界面提交 Streaming 类型的 Oozie 作业。

操作步骤

步骤 1 创建工作流，请参考创建工作流。

步骤 2 在工作流编辑页面，选择“数据流”按钮  ，将其拖到操作区中。

步骤 3 在弹出的“Streaming”窗口中配置“Mapper”的值，例如“/bin/cat”。配置“Reducer”的值，例如“/usr/bin/wc”。然后单击“添加”。

步骤 4 单击“文件+”，添加运行所需的文件。

例如“/user/oozie/share/lib/mapreduce-streaming/hadoop-streaming-3.1.1.jar”和“/user/oozie/share/lib/mapreduce-streaming/oozie-sharelib-streaming-5.1.0.jar”。

步骤 5 单击右上角的配置按钮  。在打开的配置界面中，单击“删除+”，添加删除目录，例如“/user/admin/examples/output-data/streaming_workflow”。

步骤 6 单击“属性+”，添加下列属性。

- 左边框填写属性名称“mapred.input.dir”，右边框填写属性值“/user/admin/examples/input-data/text”。
- 左边框填写属性名称“mapred.output.dir”，右边框填写属性值“/user/admin/examples/output-data/streaming_workflow”。

步骤 7 单击 Oozie 编辑器右上角的  。

保存前如果需要修改作业名称（默认为“My Workflow”），可以直接单击该名称进行修改，例如“Streaming-Workflow”。

步骤 8 保存完成后，单击  ，提交该作业。

作业提交后，可通过 Hue 界面查看作业的详细信息、日志、进度等相关内容。

----结束

20.5.2.10 提交 Distcp 作业

操作场景

该任务指导用户通过 Hue 界面提交 Distcp 类型的 Oozie 作业。

操作步骤

步骤 1 创建工作流，请参考创建工作流。

步骤 2 在工作流编辑页面，选择“DistCp”按钮  ，将其拖到操作区中。

步骤 3 当前 DistCp 操作是否是跨集群操作。

- 是，执行步骤 4。
- 否，执行步骤 7。

步骤 4 对两个集群进行跨 Manager 集群互信。

步骤 5 在弹出的“Distcp”窗口中配置“源”的值，例如“hdfs://hacluster/user/admin/examples/input-data/text/data.txt”。配置“目标”的值，例如“hdfs://target_ip:target_port/user/admin/examples/output-data/distcp-workflow/data.txt”。然后单击“添加”。

步骤 6 单击右上角的配置按钮，在打开的“属性”页签配置界面中，单击“属性+”，在左边文本框中填写属性名称“oozie.launcher.mapreduce.job.hdfs-servers”，在右边文本框中填写属性值“hdfs://*source_ip:source_port*,hdfs://*target_ip:target_port*”，执行步骤 8。

说明

source_ip：源集群的 HDFS 的 NameNode 的业务地址。

source_port：源集群的 HDFS 的 NameNode 的端口号。

target_ip：目标集群的 HDFS 的 NameNode 的业务地址。

target_port：目标集群的 HDFS 的 NameNode 的端口号。

步骤 7 在弹出的“Distcp”窗口中配置“源”的值，例如“/user/admin/examples/input-data/text/data.txt”。配置“目标”的值，例如“/user/admin/examples/output-data/distcp-workflow/data.txt”。然后单击“添加”。

步骤 8 单击右上角的配置按钮，在打开的配置界面中，单击“删除+”，添加删除目录，例如“/user/admin/examples/output-data/distcp-workflow”。

步骤 9 单击 Oozie 编辑器右上角的 。

保存前如果需要修改作业名称（默认为“My Workflow”），可以直接单击该名称进行修改，例如“Distcp-Workflow”。

步骤 10 保存完成后，单击，提交该作业。

作业提交后，可通过 Hue 界面查看作业的详细信息、日志、进度等相关内容。

----结束

20.5.2.11 互信操作示例

操作场景

在使用 Oozie 节点通过 SSH 作业执行外部节点的 Shell，需要单向免密互信时，可以参考此示例。

前提条件

已经安装 Oozie，而且能与外部节点（SSH 连接的节点）通信。

操作步骤

步骤 1 在外部节点上确保连接 SSH 时使用的用户存在，且该用户“~/.ssh”目录存在。

步骤 2 在 Oozie 所在节点上用 **omm** 用户登录，执行 **ssh-keygen -t rsa**，生成公私钥。

步骤 3 执行语句 **cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys**，把公钥添加到“authorized_keys”里。

步骤 4 以 **root** 用户将 **id_rsa.pub** 文件传给用户所在外部节点的某个已存在的目录下，例如/“opt/”下。

scp ~/.ssh/id_rsa.pub root@*外部节点 ip***:/opt/id_rsa.pub**

步骤 5 登录 Shell 所在外部节点，进入步骤 4 的目录，可以看到“id_rsa.pub”这个文件。

执行 **cat id_rsa.pub >> ~/.ssh/authorized_keys** 语句，把公钥也添加到 Shell 所在的用户的“authorized_keys”里。

步骤 6 更改目录的权限。

chmod 700 ~/.ssh

chmod 600 /opt/id_rsa.pub

chmod 600 ~/.ssh/authorized_keys

说明

- Shell 所在节点（外部节点）的帐户需要有权限执行 Shell 脚本并对于所有 Shell 脚本里涉及到的所有目录文件有足够权限。
- 如果 Oozie 具有多个节点，需要在所有 Oozie 节点执行步骤 2~步骤 6。

----结束

20.5.2.12 提交 SSH 作业

操作场景

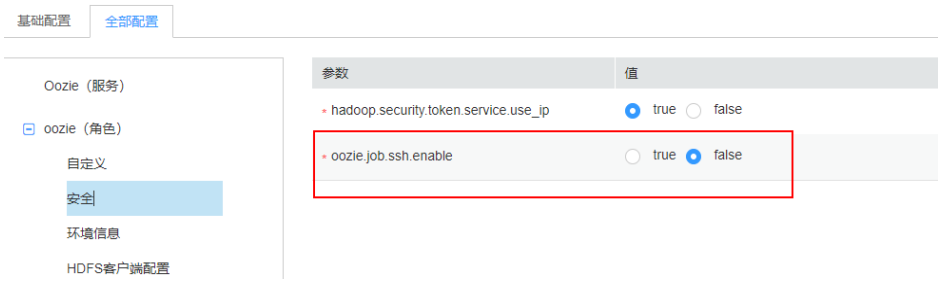
该任务指导用户通过 Hue 界面提交 SSH 类型的 Oozie 作业。

由于有安全攻击的隐患，所以默认是无法提交 SSH 作业的，如果想使用 SSH 功能，需要手动开启。

操作步骤

步骤 1 开启 SSH 功能：

1. 在 FusionInsight Manager 界面，选择“集群 > 服务 > Oozie > 配置 > 全部配置 > oozie（角色） > 安全”，修改“oozie.job.ssh.enable”的值为“true”，单击“保存”，在弹出的“保存配置”界面单击“确定”，保存配置。



2. 在 Oozie 的“概览”界面，选择右上角“更多 > 重启服务”，重启 Oozie 服务。

步骤 2 创建工作流，请参考创建工作流。

步骤 3 添加互信操作，请参考互信操作示例。

步骤 4 在工作流编辑页面，选择“Ssh”按钮  ，将其拖到操作区中。

步骤 5 在弹出的“Ssh”窗口中配置“User and Host”和“Ssh command”的值，然后单击“添加”。

步骤 6 单击 Oozie 编辑器右上角的  。

保存前如果需要修改作业名称（默认为“My Workflow”），可以直接单击该名称进行修改，例如“Ssh-Workflow”。

步骤 7 保存完成后，单击  ，提交该作业。

作业提交后，可通过 Hue 界面查看作业的详细信息、日志、进度等相关内容。

----结束

20.5.2.13 提交 Hive 脚本

操作场景

该任务指导用户通过 Hue 界面提交 Hive 脚本作业。

操作步骤

步骤 1 访问 Hue WebUI，请参考访问 Hue 的 WebUI。

步骤 2 在界面左侧导航栏选择“ > Workflow”，打开 Workflow 编辑器。

步骤 3 单击“文档”， 在操作列表中选择 Hive 脚本，将其拖到操作界面中。

步骤 4 在弹出的“HiveServer2 Script”框中，选择之前保存的 Hive 脚本，关于保存 Hive 脚本参考在 Hue WebUI 使用 HiveQL 编辑器章节。选择脚本后单击“添加”。

步骤 5 配置“作业 XML”，例如配置为 hdfs 路径“/user/admin/examples/apps/hive2/hive-site.xml”，配置方式参考提交 Hive2 作业。

步骤 6 单击 Oozie 编辑器右上角的 。

步骤 7 保存完成后，单击，提交该作业。

作业提交后，可通过 Hue 界面查看作业的详细信息、日志、进度等相关内容。

----结束

20.5.3 提交 Coordinator 定时调度作业

操作场景

该任务指导用户通过 Hue 界面提交定时调度类型的作业。

前提条件

提交 Coordinator 任务之前需要提前配置好相关的 workflow 作业。

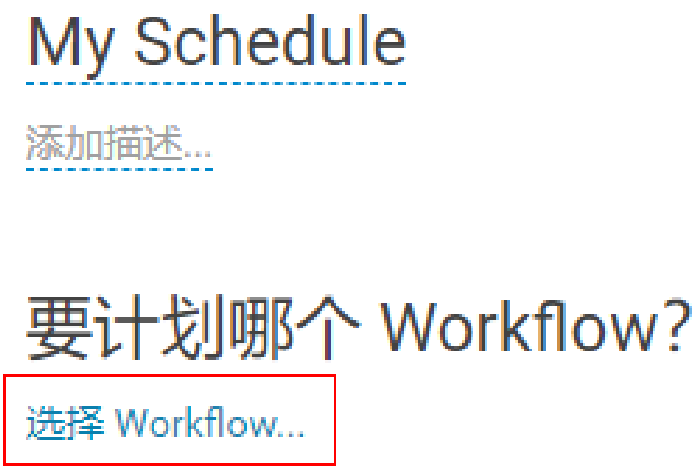
操作步骤

步骤 1 访问 Hue WebUI，请参考访问 Hue 的 WebUI。

步骤 2 在界面左侧导航栏单击，选择“计划”，打开 Coordinator 编辑器。

步骤 3 在作业编辑界面中单击“My Schedule”修改作业的名称。

步骤 4 单击“选择 Workflow...”选择需要编排的 Workflow。



步骤 5 选择好 Workflow，根据界面提示设置作业执行的频率，如果执行的 Workflow 需要传递参数，可单击“+添加参数”设置参数，然后单击右上角的 保存作业。

说明

因时区转化的原因，此处时间有可能会与当地系统实际时间差异数个小时。

步骤 6 单击编辑器右上角的，设置定时任务执行的时间范围的起始值与结束值，然后单击“提交”提交作业。

说明

因时区转化的原因，此处时间有可能会与当地系统实际时间差异数个小时。

----结束

20.5.4 提交 Bundle 批处理作业

操作场景

当同时存在多个定时任务的情况下，用户可以通过 Bundle 任务进行批量管理作业。该任务指导用户通过 Hue 界面提交批量类型的作业。

前提条件

提交 Bundle 批处理之前需要提前配置好相关的 Workflow 和 Coordinator 作业。

操作步骤

步骤 1 访问 Hue WebUI，请参考访问 Hue 的 WebUI。

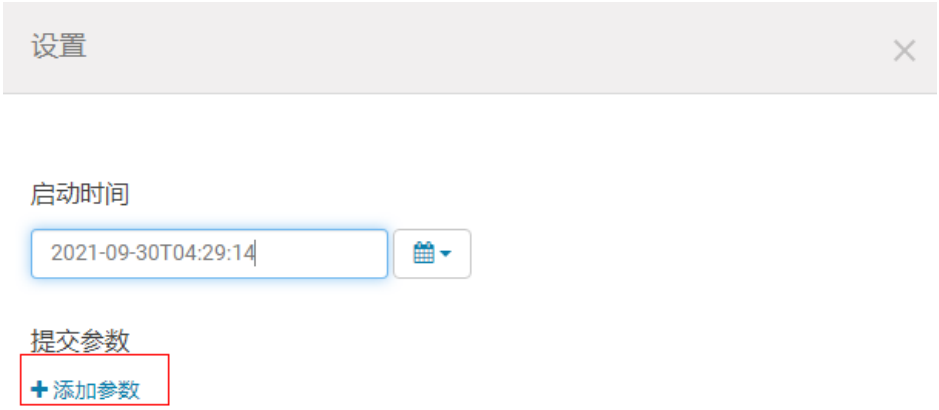
步骤 2 在界面左侧导航栏单击，选择“Bundle”，打开 Bundle 编辑器。

步骤 3 在作业编辑界面中单击“My Bundle”修改作业的名称。

步骤 4 单击“+添加 Coordinator”选择需要编排的 Coordinator 作业。

步骤 5 根据界面提示设置 Coordinator 任务调度的开始、结束时间，然后单击右上角的 保存作业。

步骤 6 单击编辑器右上角的 ，在弹出菜单选择 ，设置 Bundle 任务的启动时间，根据实际需求单击“+添加参数”设置提交参数，然后关闭对话框保存设置。



说明

因时区转化的原因，此处时间有可能会与当地系统实际时间差异数个小时。

步骤 7 单击编辑器右上角的 ，在弹出的确认界面中单击“提交”提交作业。

----结束

20.5.5 作业结果查询

操作场景

提交作业后，可以通过 Hue 界面查看具体作业的执行情况。

操作步骤

步骤 1 访问 Hue WebUI，请参考访问 Hue 的 WebUI。

步骤 2 单击菜单左侧的 ，在打开的页面中可以查看 Workflow、计划、Bundles 任务的相关信息。

----结束

20.6 Oozie 日志介绍

日志描述

日志路径：Oozie 相关日志的默认存储路径为：

- 运行日志：“/var/log/Bigdata/oozie”。
- 审计日志：“/var/log/Bigdata/audit/oozie”。

日志归档规则：Oozie 的日志分三类：运行日志、 脚本日志和审计日志。运行日志每个文件最大 20M，最多 20 个。审计日志每个文件最大 20M，最多 20 个。

说明

“oozie.log”日志每小时生成一个日志压缩文件，默认保留 720 个（一个月的日志）。

表20-7 Oozie 日志列表

日志类型	日志文件名	描述
运行日志	jetty.log	Oozie 内置 jetty 服务器日志，处理 OozieServlet 的 request/response 信息
	jetty.out	Oozie 进程启动日志
	oozie_db_temp.log	Oozie 数据库连接日志
	oozie-instrumentation.log	Oozie 仪表盘日志，主要记录 Oozie 运行状态，各组件的配置信息
	oozie-jpa.log	openJPa 运行日志
	oozie.log	Oozie 运行日志
	oozie-<SSH_USER>-<DATE>-<PID>-gc.log	Oozie 服务垃圾回收日志
	oozie-ops.log	Oozie 操作日志
	check-serviceDetail.log	Oozie 健康检查日志
	oozie-error.log	Oozie 运行错误日志
	threadDump-<DATE>.log	记录服务进程正常退出时堆栈信息的日志
脚本日志	postinstallDetail.log	安装后启动前的工作日志
	prestartDetail.log	预启动日志
	startDetail.log	服务启动日志
	stopDetail.log	服务停止日志
	upload-sharelib.log	sharelib 上传操作日志

日志类型	日志文件名	描述
审计日志	oozie-audit.log	审计日志

日志级别

Oozie 中提供了如表 20-8 所示的日志级别。

日志级别优先级从高到低分别是 ERROR、WARN、INFO、DEBUG，程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。

表20-8 日志级别

级别	描述
ERROR	ERROR 表示错误日志，可能会导致进程异常。
WARN	WARN 表示当前事件处理存在异常信息。
INFO	INFO 表示系统及各事件正常运行状态信息。
DEBUG	DEBUG 表示记录系统及数据库底层数据传输的信息。

如果您需要修改日志级别，请执行如下操作：

步骤 1 登录 FusionInsight Manager 系统。

步骤 2 选择“集群 > *待操作集群的名称* > 服务 > Oozie > 配置”。

步骤 3 选择“全部配置”。

步骤 4 左边菜单栏中选择所需修改的角色所对应的日志菜单。

步骤 5 选择所需修改的日志级别。

步骤 6 单击“保存”，单击“确定”，处理结束后生效。

----结束

日志格式

Oozie 的日志格式如下所示。

表20-9 日志格式

日志类型	格式	示例
运行日	<yyyy-MM-dd	2015-05-29 21:01:45,268 INFO

日志类型	格式	示例
志	HH:mm:ss,SSS><Log Level><日志事件的发生位置>|<log 中的 message>	StatusTransitService$StatusTransitRunnable: 539 - USER[-] GROUP[-] Released lock for [org.apache.oozie.service.StatusTransitService]
脚本日志	<yyyy-MM-dd HH:mm:ss,SSS><主机名><Log Level><log 中的 message>	2015-06-01 17:18:03 001 suse11-192-168-0-111 oozie INFO Running oozie service check script
审计日志	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<线程名称>|<log 中的 message>|<日志事件的发生位置>	2015-06-01 22:38:41,323 | INFO | http-bio-21003-exec-8 | IP [192.168.0.111] USER [null], GROUP [null], APP [null], JOBID [null], OPERATION [null], PARAMETER [null], RESULT [SUCCESS], HTTPCODE [200], ERRORCODE [null], ERRORMESSAGE [null] | org.apache.oozie.util.XLog.log(XLog.java:539)

20.7 Oozie 常见问题

20.7.1 Oozie 定时任务没有准时运行

问题

在 Hue 或者 Oozie 客户端设置执行 Coordinator 定时任务，但没有准时执行。

回答

需要使用 UTC 时间，例如在“job.properties”中配置“start=2016-12-20T09:00Z”。

20.7.2 HDFS 上更新了 oozie 的 share lib 目录但没有生效

问题

在 HDFS 的“/user/oozie/share/lib”目录上传了新的 jar 包，但执行任务时仍然报找不到类的错误。

回答

在客户端执行如下命令刷新目录：

oozie admin -oozie https://xxx.xxx.xxx.xxx:21003/oozie -sharelibupdate

20.7.3 Oozie 常用排查手段

1. 根据任务在 Yarn 上的任务日志排查，首先把实际的运行任务，比如 Hive SQL 通过 beeline 运行一遍，确认 Hive 无问题。
2. 出现“classnotfoundException”等报错，排查“/user/oozie/share/lib”路径下各组件有没有报错的类的 Jar 包，如果没有，添加 Jar 包并执行 HDFS 上更新了 oozie 的 share lib 目录但没有生效。如果执行了更新“share lib”目录依然报找不到类，那么可以查看执行更新“share lib”的命令打印出来的路径“sharelibDirNew”是否是“ /user/oozie/share/lib”， 一定不能是其它目录。

3. 出现 NosuchMethodError，排查“/user/oozie/share/lib”路径下各组件的 Jar 包是不是有多个版本，注意业务本身上传的 Jar 包冲突，可通过 Oozie 在 Yarn 上的运行日志打印的加载的 Jar 包排查是否有 Jar 包冲突。
4. 自研代码运行异常，可以先运行 Oozie 的自带样例，排除 Oozie 自身的异常。
5. 寻求技术人员的支持，需要收集 Yarn 上 Oozie 任务运行日志、Oozie 自身的日志及组件的运行的日志，例如使用 Oozie 运行 Hive 报异常，需收集 Hive 的日志。

21 使用 OpenTSDB

21.1 使用 MRS 客户端操作 OpenTSDB 指标数据

用户可以根据业务需要，在 MRS 集群的客户端中进行交互式操作。启用 Kerberos 认证的集群，需要操作的用户属于“opentsdb，hbase，opentsdbgroup 和 supergroup”组且拥有 HBase 权限。

前提条件

- 获取用户“admin”帐号密码。“admin”密码在创建 MRS 集群时由用户指定。
- 已安装集群客户端，例如安装目录为“/opt/client”，以下操作的客户端目录只是举例，请根据实际安装目录修改。更新客户端，具体请参见更新客户端（3.x 之前版本）。

使用客户端

步骤 1 如果当前集群已启用 Kerberos 认证，登录 MRS Manager 页面，创建属于“opentsdb，hbase，opentsdbgroup 和 supergroup”组且拥有 HBase 权限的用户，例如创建用户为 opentsdbuser。如果当前集群未启用 Kerberos 认证，则无需执行此步骤。

步骤 2 根据业务情况，准备好客户端，并登录安装客户端的节点。

例如在 Master2 节点更新客户端，则登录该节点使用客户端，具体参见更新客户端（3.x 之前版本）。

步骤 3 执行以下命令切换用户。

sudo su - omm

步骤 4 执行以下命令，切换到客户端目录，例如“/opt/client”。

cd /opt/client

步骤 5 执行以下命令，配置环境变量。

source bigdata_env

步骤 6 如果当前集群已启用 Kerberos 认证，执行以下命令认证当前用户。如果当前集群未启用 Kerberos 认证，则无需执行此命令。

- 当用户为“人机”用户时：执行 **kinit opentsdbuser** 认证用户
- 当用户为“机机”用户时：下载用户认证凭据文件，保存并解压获取用户的 user.keytab 文件与 krb5.conf 文件，进入解压后的 user.keytab 目录下，执行 **kinit -kt user.keytab opentsdbuser** 认证用户

步骤 7 操作 Opentsdb 数据，具体请参见操作数据。

----结束

操作数据

- 查看帮助

  执行 **tsdb** 命令打印出当前 opentsdb 所支持的所有命令。如，fsck, import, mkmetric, query, tsd, scan, search, uid, version。

  回显信息：

  ```
  tsdb: error: unknown command ''
  usage: tsdb <command> [args]
  Valid commands: fsck, import, mkmetric, query, tsd, scan, search, uid, version
  ```

- 创建 OpenTSDB 指标

  执行 **tsdb mkmetric** 命令创建指标。例如执行 **tsdb mkmetric sys.cpu.user** 命令创建名为 sys.cpu.user 的指标。

  回显信息：

  ```
  Start run net.opentsdb.tools.UidManager, args: assign metrics sys.cpu.user
  metrics sys.cpu.user: [0, 0, 6]
  ```

- 向 OpenTSDB 指标中导入数据

  a. 准备指标文件，例如包含如下信息的 importData.txt 文件。

  sys.cpu.user 1356998400 41 host=web01 cpu=0

  sys.cpu.user 1356998401 42 host=web01 cpu=0

  sys.cpu.user 1356998402 44 host=web01 cpu=0

  sys.cpu.user 1356998403 47 host=web01 cpu=0

  sys.cpu.user 1356998404 42 host=web01 cpu=0

  sys.cpu.user 1356998405 42 host=web01 cpu=0

  b. 执行 **tsdb import** 命令导入指标数据。例如执行 **tsdb import importData.txt** 命令导入 importData.txt 文件。

  ```
  Start run net.opentsdb.tools.TextImporter, args: importData.txt
  2019-06-26
  15:45:22,091 INFO  [main] TextImporter:
  reading from file:importData.txt
  2019-06-26
  15:45:22,102 INFO  [main] TextImporter:
  Processed importData.txt in 11 ms, 6 data points (545.5 points/s)
  2019-06-26
  15:45:22,102 INFO  [main] TextImporter:
  Total: imported 6 data points in 0.012s (504.0 points/s)
  ```

- 查询 OpenTSDB 指标

执行 **tsdb uid metrics** 命令获取当前 OpenTSDB 中存入的指标。例如执行 **tsdb uid metrics sys.cpu.user** 命令查询 sys.cpu.user 的数据。

回显信息：

```
Start run net.opentsdb.tools.UidManager, args: metrics sys.cpu.user
metrics sys.cpu.user: [0, 0, 6]
```

如需获得更多信息，请执行 **tsdb uid** 命令。

```
Start run net.opentsdb.tools.UidManager, args:
Not enough arguments
Usage: uid <subcommand> args
Sub commands:
  grep [kind] <RE>: Finds matching IDs.
  assign <kind> <name> [names]: Assign an ID for the given name(s).
  rename <kind> <name> <newname>: Renames this UID.
  delete <kind> <name>: Deletes this UID.
  fsck: [fix] [delete_unknown] Checks the consistency of UIDs.
       fix          - Fix errors. By default errors are logged.
       delete_unknown - Remove columns with unknown qualifiers.
                    The "fix" flag must be supplied as well.
  [kind] <name>: Lookup the ID of this name.
  [kind] <ID>: Lookup the name of this ID.
  metasync: Generates missing TSUID and UID meta entries, updates created
timestamps
  metapurge: Removes meta data entries from the UID table
  treesync: Process all timeseries meta objects through tree rules
  treepurge <id> [definition]: Purge a tree and/or the branches from storage.
Provide an integer Tree ID and
optionally add "true" to delete the tree definition
Example values for [kind]: metrics, tagk (tag name), tagv (tag value).
  --config=PATH    Path to a configuration file (default: Searches for file see
docs).
  --idwidth=N      Number of bytes on which the UniqueId is encoded.
  --ignore-case    Ignore case distinctions when matching a regexp.
  --table=TABLE    Name of the HBase table where to store the time series
(default: tsdb).
  --uidtable=TABLE Name of the HBase table to use for Unique IDs (default:
tsdb-uid).
  --verbose       Print more logging messages and not just errors.
  --zkbasedir=PATH Path under which is the znode for the -ROOT- region (default:
/hbase).
  --zkquorum=SPEC  Specification of the ZooKeeper quorum to use (default:
localhost).
  -i             Short for --ignore-case.
  -v             Short for --verbose.
```

- 扫描 Opentsdb 的指标数据

  执行 **tsdb query** 命令批量查询导入的指标数据，命令格式如下： tsdb query <START-DATE> <END-DATE> <aggregator> <metric> <tagk=tagv>，例如执行 **tsdb query 0 1h-ago sum sys.cpu.user host=web01**

```
Start run net.opentsdb.tools.CliQuery, args: 0 1h-ago sum sys.cpu.user
host=web01
sys.cpu.user 1356998400000 41 {host=web01, cpu=0}
sys.cpu.user 1356998401000 42 {host=web01, cpu=0}
sys.cpu.user 1356998402000 44 {host=web01, cpu=0}
```

```
sys.cpu.user 1356998403000 47 {host=web01, cpu=0}
sys.cpu.user 1356998404000 42 {host=web01, cpu=0}
sys.cpu.user 1356998405000 42 {host=web01, cpu=0}
```

说明

<START-DATE>:要查询指标的起始时间点。

<END-DATE>:要查询指标的结束时间点。

<aggregator>:查询数据的聚合方式。

<metric>:所需查询的指标名称。

<tagk=tagv>:标签的 key 和 value。

- 删除录入的 Opentsdb 指标

  执行命令 **tsdb uid delete** 命令删除录入的指标及值。例如删除 sys.cpu.user 指标可执行命令 **tsdb uid delete metrics sys.cpu.user**。

```
Start run net.opentsdb.tools.UidManager, args: delete metrics sys.cpu.user
```

21.2 使用 curl 命令操作 OpenTSDB

写入数据

例如，录入一个指标名称为 testdata，时间戳为 1524900185，值为 true，标签为 key，value 的指标数据。

```
curl -ki -X POST -d '{"metric":"testdata", "timestamp":1524900185,
"value":"true", "tags":{"key":"value"}}'
https://<tsd_ip>:4242/api/put?sync
```

<tsd_ip>表示所需写入数据的 Opentsdb 服务的 TSD 实例的 IP 地址。

```
HTTP/1.1 204 No Content
Content-Type: application/json; charset=UTF-8
Content-Length:0
```

查询数据

例如，可查询指标 testdata 在过去三年中中的汇总信息。

```
curl -ks https://<tsd_ip>:4242/api/query?start=3y-ago\&m=sum:testdata |
python -m json.tool
```

- <tsd_ip>：所需访问 Opentsdb 服务的 TSD 实例 IP 或主机名。
- <start=3y-ago\&m=sum:testdata>：在请求中可能无法识别“&”符号，需对其进行转义。
- <python -m json.tool>（可选）：把响应的请求转换为 json 格式。

```
[
   {
      "aggregateTags": [],
```

```
        "dps": {
            "1524900185": 1
        },
        "metric": "testdata",
        "tags": {
            "key": "value"
        }
    }
]
```

查询 tsd 状态信息

例如，可查询连接 HBase 的客户端信息。

```
curl -ks https://<tsd_ip>:4242/api/stats/region_clients | python -m
json.tool
```

<tsd_ip>：所需访问 Opentsdb 服务的 TSD 实例 IP 地址。

```
[
    {
        "dead": false,
        "endpoint":"/xx.xx.xx.xx:16020",
        "inflightBreached": 0,
        "pendingBatchedRPCs": 0,
        "pendingBreached": 0,
        "pendingRPCs": 0,
        "rpcResponsesTimedout": 0,
        "rpcResponsesUnknown": 0,
        "rpcid": 78,
        "rpcsInFlight": 0,
        "rpcsSent": 79,
        "rpcsTimedout": 0,
        "writesBlocked": 0
    }
]
```

22 使用 Presto

22.1 访问 Presto 的 WebUI

用户可以通过 Presto 的 WebUI，在图形化界面查看 Presto 的统计信息。Presto 的 WebUI 界面不支持使用 IE 浏览器访问，建议使用 Google 浏览器访问。

前提条件

- 已安装 Presto 服务的集群。
- 已安装集群客户端，例如安装目录为“/opt/client”。以下操作的客户端目录只是举例，请根据实际安装目录修改。

访问 Presto 的 WebUI

- 方法一（适用于 MRS 3.x 及之后版本）：
  a. 登录 FusionInsight Manager，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本）。然后选择“集群 > *待操作的集群名*称 > 服务 ”。
  b. 选择“Presto”并在“基本信息”的“Coordinator WebUI”中单击“Coordinator(Coordinator)”，打开 Presto 的 WebUI 页面。
- 方法二（适用于 MRS 3.x 之前版本）：
  a. 登录 MRS Manager 页面，选择“服务管理”。
  b. 选择“Presto”并在“Presto 概述”的“Presto WebUI”中单击“Coordinator (主)”，打开 Presto 的 WebUI 页面。

  说明

  第一次访问 Presto WebUI，需要在浏览器中添加站点信任以继续打开页面。
- 方法三（适用于 MRS 1.9.2 及之后版本）：
  a. 在集群列表页面，单击集群名称，登录集群详情页面，选择“组件管理”。

  说明

  若集群详情页面没有“组件管理”页签，请先完成 IAM 用户同步（在集群详情页的“概览”页签，单击“IAM 用户同步”右侧的“同步”进行 IAM 用户同步）。

b. 选择“Presto”并在“Presto 概述”的“Presto WebUI”中单击“Coordinator(主)”，打开 Presto 的 WebUI 页面。

22.2 使用客户端执行查询语句

用户可以根据业务需要，在 MRS 集群的客户端中进行交互式查询。启用 Kerberos 认证的集群，需要提交拓扑的用户属于“presto”组。

MRS 3.x 版本 Presto 组件暂不支持开启 Kerberos 认证。

前提条件

- 获取用户“admin”帐号密码。“admin”密码在创建 MRS 集群时由用户指定。
- 已刷新客户端。
- 3.x 版本的集群需要手动安装 Presto 客户端。

操作步骤

步骤 1 启用 Kerberos 认证的集群，登录 MRS Manager 页面，创建拥有“Hive Admin Privilege”权限的角色，创建角色请参考“用户指南 > 管理现有集群 > MRS 多用户权限管理 > 创建角色”。

步骤 2 创建属于“Presto”和“Hive”组的用户，同时为该用户绑定步骤 1 中创建的角色，然后下载用户认证文件，参见“用户指南 > 管理现有集群 > MRS 多用户权限管理 > 创建用户”，“用户指南 > 管理现有集群 > MRS 多用户权限管理 > 下载用户认证文件”。

步骤 3 将下载的 user.keytab 文件和 krb5.conf 上传到 MRS 客户端所在节点。

说明

步骤步骤 2-步骤 3 仅启用 Kerberos 认证的集群执行，普通集群请请直接从步骤步骤 4 开始执行。

步骤 4 根据业务情况，准备好客户端，并登录安装客户端的节点。

例如在 Master2 节点更新客户端，则登录该节点使用客户端，具体参见“用户指南 > 连接集群 > 使用 MRS 客户端 > 更新客户端”。

步骤 5 执行以下命令切换用户。

sudo su - omm

步骤 6 执行以下命令，切换到客户端目录，例如“/opt/client”。

cd /opt/client

步骤 7 执行以下命令，配置环境变量。

source bigdata_env

步骤 8 连接 Presto Server。根据客户端的不同，提供如下两种客户端的链接方式。

- 使用 MRS 提供的客户端。
  - 未启用 Kerberos 认证的集群，执行以下命令连接本集群的 Presto Server。

    presto_cli.sh
  - 未启用 Kerberos 认证的集群，执行以下命令连接其他集群的 Presto Server，其中 ip 为对应集群的 Presto 的浮动 IP（可通过在 Presto 配置项中搜索 PRESTO_COORDINATOR_FLOAT_IP 的值获得），port 为 Presto Server 的端口号，默认为 7520。

    presto_cli.sh --server *http://ip:port*
  - 启用 Kerberos 认证的集群，执行以下命令连接本集群的 Presto Server。

    presto_cli.sh --krb5-config-path *krb5.conf 文件路径* --krb5-principal *用户 principal* --krb5-keytab-path *user.keytab 文件路径* --user *presto 用户名*
  - 启用 Kerberos 认证的集群，执行以下命令连接其他集群的 Presto Server，其中 ip 为对应集群的 Presto 的浮动 IP（可通过在 Presto 配置项中搜索 PRESTO_COORDINATOR_FLOAT_IP 的值获得），port 为 Presto Server 的端口号，默认为 7521。

    presto_cli.sh --krb5-config-path *krb5.conf 文件路径* --krb5-principal *用户 principal* --krb5-keytab-path *user.keytab 文件路径* --server *https://ip:port* --krb5-remote-service-name *Presto Server name*
- 使用原生客户端

  Presto 原生客户端为客户端目录下的 Presto/presto/bin/presto，使用方式参见 https://prestodb.io/docs/0.215/installation/cli.html 和 https://prestodb.io/docs/0.215/security/cli.html。

步骤 9 执行 Query 语句，如“show catalogs”，更多语句请参阅 https://prestodb.io/docs/0.215/sql.html。

说明

启用 Kerberos 认证的集群使用 Presto 查询 Hive Catalog 的数据时，运行 Presto 客户端的用户需要有 Hive 表的访问权限，并且需要在 Hive beeline 中执行命令 **grant all on table [table_name] to group hive**，给 Hive 组赋权限。

步骤 10 查询结束后，执行以下命令退出客户端。

quit

----结束

22.3 在 DLF 中使用 Presto 转储

前提条件

- MRS 集群中安装了 Presto 组件。
- 已完成 IAM 用户同步（在集群详情页的“概览”页签，单击“IAM 用户同步”右侧的“同步”进行 IAM 用户同步）。
- 用户拥有操作 OBS 文件系统的权限，具体参考和。

- 用户已完成 Presto 权限配置，具体请参考配置 Presto 组件权限。

在 DLF 创建 MRS PrestoSQL 类型的数据连接

步骤 1 在 DLF 控制台的左侧导航栏，选择“数据管理 > 连接管理”。

步骤 2 在页面的右上方，单击“新建数据连接”。

步骤 3 参考表 22-1 配置相关参数。

表22-1 新建数据连接

参数	说明
数据链接类型	选择“MapReduce 服务（MRS PrestoSQL）”。
数据连接名称	数据连接的名称，只能包含英文字母、数字、“_”、“-”，且长度为 1~100 个字符。
集群名	选择 Presto 所属的 MRS 集群。

步骤 4 单击“测试”，测试数据连接的连通性。如果无法连通，数据连接将无法创建。

步骤 5 单击“确定”，创建数据连接。

----结束

在 DLF 脚本开发页面新建并执行 SQL 脚本

⚠ 注意

该场景下转储到 OBS 上的查询结果，最大只会保留最近的 10000 次查询结果，当查询次数超过该限制时，会自动按照查询时间先后顺序老化历史查询结果。请用户注意并合理使用，避免造成数据丢失风险。

步骤 1 在 DLF 控制台的左侧导航栏，选择“数据开发 > 脚本开发”。

步骤 2 在“右侧区域”，单击“新建 SQL 脚本”，选择“Presto”。

步骤 3 在编辑器右上方“数据连接”处选择在 DLF 创建 MRS PrestoSQL 类型的数据连接中创建的连接。

步骤 4 在编辑器右上方“模式”处选择所选连接对应模式。

步骤 5 在编辑器中输入 SQL 语句（支持输入多条 SQL 语句），如需单独执行某部分 SQL 语句，请选中 SQL 语句再运行。

步骤 6 在编辑器上方，单击“运行”。SQL 语句运行完成后，在编辑器下方可以查看脚本的执行历史、执行结果。

说明

- 不支持管理员类型的操作，即需要执行“set role admin”才能执行的命令都不支持。
- 由于每条语句都是单独执行的，所以对于设置上下文的语句（如“use”），执行后不会生效。
- 开启 Presto 授权情况下，各类用户默认权限如下：
- hive 下的 mrs_reserved 的数据库，默认所有人均有读写权限。
- hive 下的 default 数据库，具备 MRS CommonOperations、MRS FullAccess、MRS Administrator、Tenant Administrator 策略的 IAM 用户，均有读写权限；MRS ReadOnlyAccess 策略的用户，对该数据库有只读权限。
- 集群的 admin 用户以及具备 MRS FullAccess、MRS Administrator、Tenant Administrator 策略的 IAM 用户，具备 hive 数据库的 admin 角色。

----结束

22.4 配置 Presto 组件权限

MRS 3.x 版本不支持该操作。

安全集群配置 Presto 组件权限

安全集群默认已经开放 Presto 组件 Hive Catalog 授权，权限配置如下：

步骤 1 参考访问 FusionInsight Manager（MRS 3.x 及之后版本）页面登录 Manager 页面。

步骤 2 选择“系统设置 > 角色管理”，配置拥有 Hive 数据库/表权限的角色并为用户绑定该角色。

----结束

普通集群配置 Presto 组件权限

普通集群默认不开放 Presto 授权，需要手动配置，操作如下：

步骤 1 登录 MRS 集群详情页面。

步骤 2 选择“组件管理 > Hive”。设置“参数类别”为“全部配置”，进入 Hive 配置界面修改参数配置。

步骤 3 搜索并修改如下参数。

- hive.security.authorization.enabled 配置为 true
- hive.security.authorization.manager 配置为 org.apache.hadoop.hive.ql.security.authorization.plugin.sqlstd.SQLStdHiveAuthorizerFactory

步骤 4 单击“保存配置”，并勾选“重新启动受影响的服务或实例。”重启 Hive 服务。

步骤 5 选择“组件管理 > Presto”。设置“参数类别”为“全部配置”，进入 Presto 配置界面修改参数配置。

步骤 6 搜索并修改参数 hive.security，配置为 sql-standard-with-group。

步骤 7 单击“保存配置”，并勾选“重新启动受影响的服务或实例。”重启 Presto 服务。

步骤 8 登录 MRS Manager 页面。

步骤 9 选择“系统设置 > OMS 数据库密码修改 > 重启 OMS 服务”。

步骤 10 选择“系统设置 > 角色管理”，配置拥有 Hive 数据库/表权限的角色并为用户绑定该角色。

----结束

23 使用 Ranger（MRS 3.x）

23.1 登录 Ranger 管理界面

Ranger 服务提供了集中式的权限管理框架，可以对 HDFS、HBase、Hive、Yarn 等组件进行细粒度的权限访问控制，并且提供了 Web UI 方便管理员进行操作。

Ranger 用户类型

Ranger 中的用户可分为 Admin、User、Auditor 等类型，不同用户具有的 Ranger 管理界面查看和操作权限不同。

- Admin：安全管理员，可查看所有页面内容，进行服务权限管理插件及权限访问控制策略的管理操作，可查看审计信息内容，可进行用户类型设置。
- Auditor：审计管理员，可查看服务权限管理插件及权限访问控制策略的内容。
- User：普通用户，可以被管理员赋予具体权限。

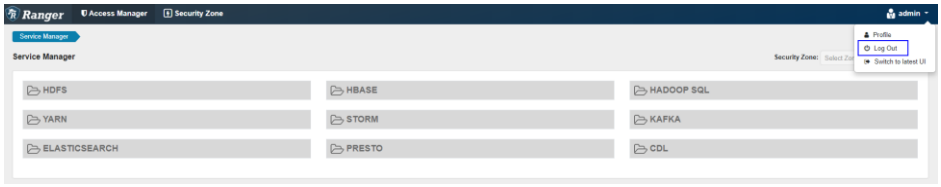
登录 Ranger 管理界面

安全模式（集群开启了 Kerberos 认证）

步骤 1 使用 **admin** 用户登录 FusionInsight Manager，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本）。选择“集群 > 服务 > Ranger”，进入 Ranger 服务概览页面。

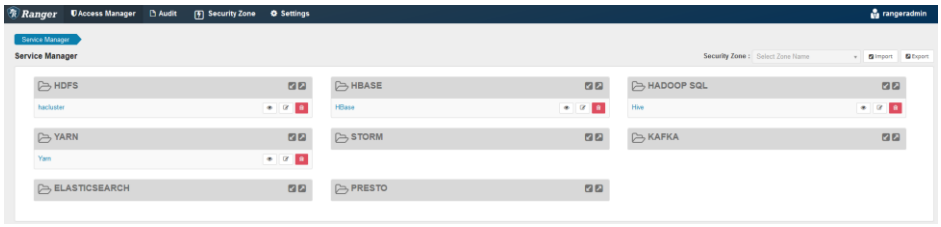
步骤 2 单击“基本信息”区域中的“RangerAdmin”，进入 Ranger WebUI 界面。

- **admin** 用户在 Ranger 中的用户类型为“User”，只能查看 Access Manager 和 Security Zone 页面。
- 如需查看所有管理页面，需要切换至 **rangeradmin** 用户或者其他具有 Ranger 管理员权限的用户：
  a. 在 Ranger WebUI 界面，单击右上角用户名，选择“Log Out”，退出当前用户。



b. 使用 **rangeradmin** 用户（默认密码为 Rangeradmin@123）或者其他具有 Ranger 管理员权限用户重新登录。

图23-1 Ranger WebUI



----结束

普通模式（集群关闭了 Kerberos 认证）：

步骤 1 使用 **admin** 用户登录 FusionInsight Manager，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本）。选择“集群 > 服务 > Ranger”，进入 Ranger 服务概览页面。

步骤 2 单击“基本信息”区域中的“RangerAdmin”，进入 Ranger WebUI 界面。

admin 用户在 Ranger 中的用户类型为“Admin”，能查看 Ranager 所有管理页面，无需切换至 **rangeradmin** 用户。

说明

普通模式下使用 **rangeradmin** 用户登录 Ranger WebUI 界面，页面报错 401。

----结束

在 Ranger 管理首页可查看当前 Ranger 已集成的各服务权限管理插件，用户可通过对应插件设置更细粒度的权限，具体主要操作页面功能描述参见表 23-1。

表23-1 Ranger 界面操作入口功能描述

入口	功能描述
Access Manager	查看当前 Ranger 已集成的各服务权限管理插件，用户可通过对应插件设置更细粒度的权限，具体操作请参考配置组件权限策略。
Audit	查看 Ranger 运行及权限管控相关审计日志信息，具体操作请参考查看 Ranger 审计信息。

入口	功能描述
Security Zone	配置安全区域，管理员可将各组件的资源切分为多个区域，由不同管理员为服务的指定资源设置安全策略，以便更好的管理，具体操作可参考配置 Ranger 安全区。
Settings	查看 Ranger 相关权限设置信息，例如查看用户、用户组、Role 等，具体操作可参考查看 Ranger 权限信息

23.2 启用 Ranger 鉴权

操作场景

该章节指导用户如何启用 Ranger 鉴权。安全模式默认开启 Ranger 鉴权，普通模式默认关闭 Ranger 鉴权。

操作步骤

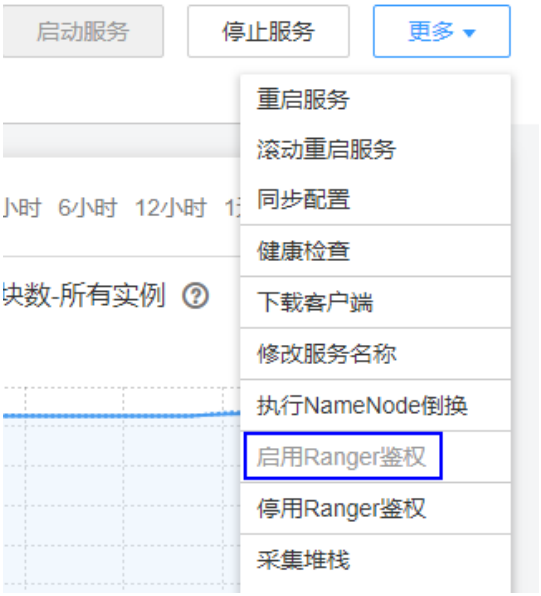
步骤 1 登录 FusionInsight Manager 页面，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本）。选择“集群 > 服务 > *需要启用 Ranger 鉴权的服务名称*”。

步骤 2 在服务“概览”页面右上角单击“更多”，选择“启用 Ranger 鉴权”。在弹出的对话框中输入密码，单击“确定”，操作成功后单击“完成”。

说明

如果“启用 Ranger 鉴权”是灰色，表示已开启 Ranger 鉴权，如图 23-2 所示。

图23-2 启用 Ranger 鉴权



步骤 3 滚动重启服务或者重启服务。

----结束

23.3 配置组件权限策略

新安装的 MRS 集群默认安装 Ranger 服务并启用了 Ranger 鉴权模型，管理员可以通过组件权限插件对组件资源的访问设置细粒度的安全访问策略。

目前安全模式集群中支持 Ranger 的组件包括：HDFS、Yarn、HBase、Hive、Spark2x、Kafka、Storm。

通过 Ranger 配置用户权限策略

步骤 1 使用管理员登录 Ranger 管理页面。

步骤 2 在 Ranger 首页的“Service Manager”区域内，单击组件名称下的权限插件名称，即可进入组件安全访问策略列表页面。

说明

各组件的策略列表中，系统默认会生成若干条目，用于保证集群内的部分默认用户或用户组的权限（例如 supergroup 用户组），请勿删除，否则系统默认用户或用户组的权限会受影响。

步骤 3 单击“Add New Policy”，根据业务场景规划配置相关用户或者用户组的资源访问策略。

不同组件的访问策略配置样例参考：

- 添加 HDFS 的 Ranger 访问权限策略

- 添加 HBase 的 Ranger 访问权限策略
- 添加 Hive 的 Ranger 访问权限策略
- 添加 Yarn 的 Ranger 访问权限策略
- 添加 Spark2x 的 Ranger 访问权限策略
- 添加 Kafka 的 Ranger 访问权限策略
- 添加 Storm 的 Ranger 访问权限策略

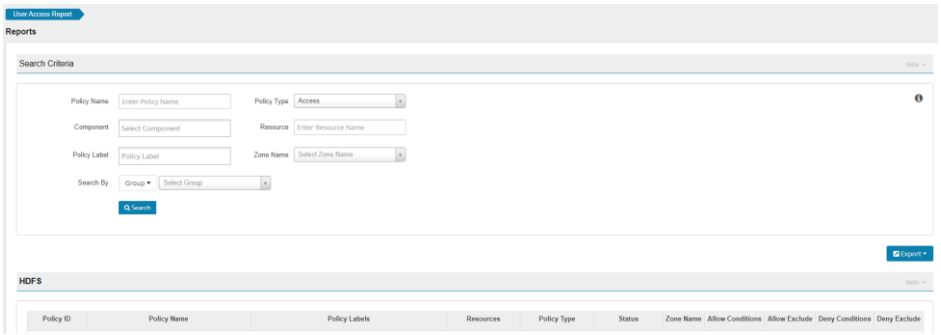
策略添加后，需等待 30 秒左右，待系统生效。

说明

组件每次启动都会检查组件默认的 Ranger Service 是否存在，如果不存在则会创建以及为其添加默认 Policy。如果用户在使用过程中误删了 Service，可以重启或者滚动重启相应组件服务来恢复，若是误删了默认 Policy，可先手动删除 Service，再重启组件服务。

步骤 4 单击“Access Manager > Reports”，可查看各组件所有的安全访问策略。

系统策略较多时，可通过策略名称、类型、组件、资源对象、策略标签、安全区域、用户或用户组等信息进行过滤搜索，也可以单击“Export”导出相关策略内容。



说明

- 对于同一个固定资源对象通常只能配置一条策略，多条策略针对的具体资源对象重复时将无法保存。
- 配置策略时，不同条件的优先级可参考 Ranger 权限策略条件判断优先级。

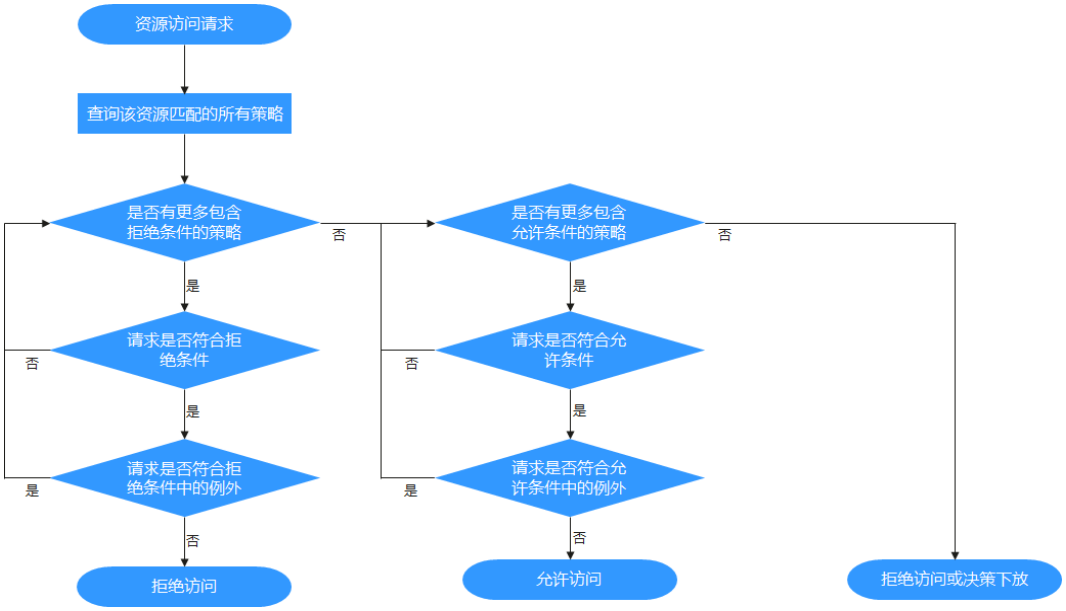
----结束

Ranger 权限策略条件判断优先级

配置资源的权限策略时，可配置针对该资源的允许条件（Allow Conditions）、允许例外条件（Exclude from Allow Conditions）、拒绝条件（Deny Conditions）以及拒绝例外条件（Exclude from Deny Conditions），以满足不同场景下的例外需求。

不同条件的优先级由高到低为：拒绝例外条件 > 拒绝条件 > 允许例外条件 > 允许条件。

系统判断流程可参考下图所示，如果组件资源请求未匹配到 Ranger 中的权限策略，系统默认将拒绝访问。但是对于 HDFS 和 Yarn，系统会将决策下放给组件自身的访问控制层继续进行判断。



例如要将一个文件夹 FileA 的读写权限授权给用户组 groupA，但是该用户组内某个用户 UserA 除外，这时可以增加一个允许条件及一个例外条件即可实现。

23.4 查看 Ranger 审计信息

管理员可通过 Ranger 界面查看 Ranger 运行审计日志及组件使用 Ranger 鉴权后权限管控审计日志信息。

查看 Ranger 审计信息内容

步骤 1 登录 Ranger 管理页面。

步骤 2 单击“Audit”，查看相关审计信息，各页签内容说明请参考表 23-2，条目较多时，单击搜索框可根据关键字字段进行筛选。

表23-2 Audit 信息

页签	内容描述
Access	用户通过 Ranger 鉴权访问组件资源的审计信息。
Admin	Ranger 上操作审计信息，例如安全访问策略的创建/更新/删除、组件权限策略的创建/删除、role 的创建/更新/删除等。
Login Sessions	登录 Ranger 的用户会话审计信息。
Plugins	Ranger 内组件权限策略信息。
Plugin Status	各组件节点权限策略的同步审计信息。
User Sync	Ranger 与 LDAP 用户同步审计信息。

----结束

23.5 配置 Ranger 安全区

Ranger 支持配置安全区，管理员可将各组件的资源切分为多个安全区，由对应管理员用户为区域的指定资源设置安全策略，以便更好的细分资源管理。安全区中定义的策略仅适用于区域中的资源，服务的资源被划分到安全区后，非安全区针对该资源的访问权限策略将不再生效。安全区的管理员只能在其作为管理员的安全区中设置策略。

添加安全区

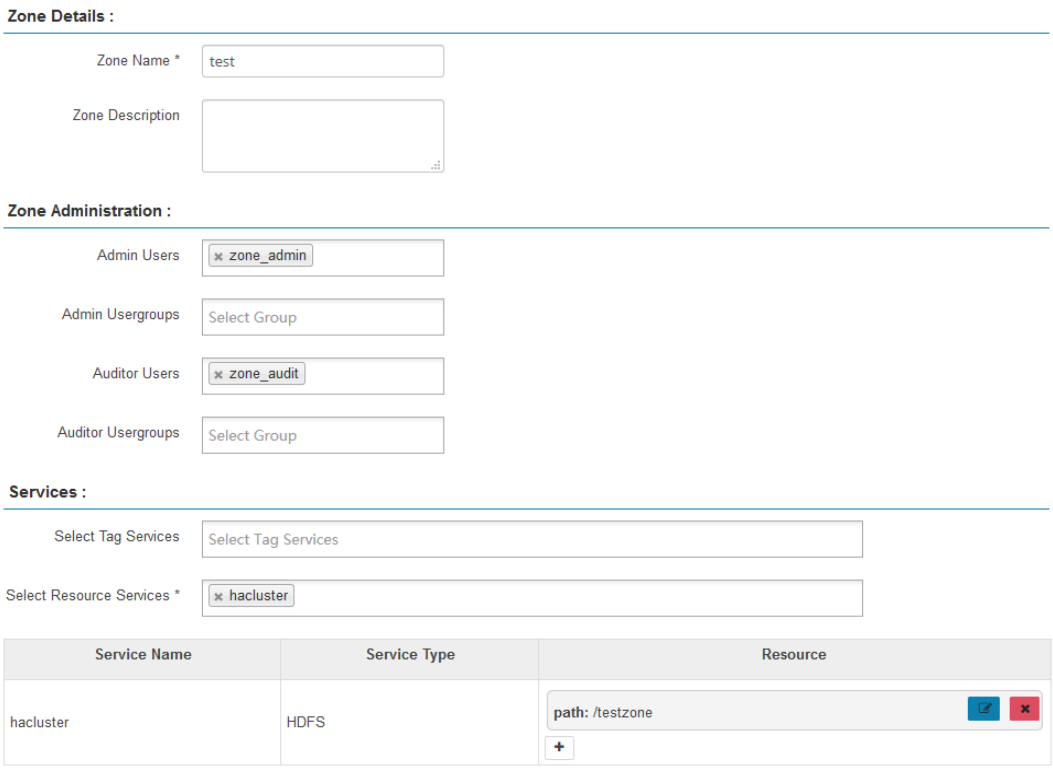
步骤 1 使用 Ranger 管理员登录 Ranger 管理界面。

步骤 2 单击“Security Zone”，在区域列表页面中单击  ，添加安全区。

表23-3 安全区配置参数

参数名称	描述	示例
Zone Name	配置安全区的名称。	test
Zone Description	配置安全区的描述信息。	-
Admin Users/Admin Usergroups	配置安全区的管理用户/用户组，可在安全区中添加及修改相关资源的权限策略。 必须至少配置一个用户或用户组。	zone_admin
Auditor Users/ Auditor Usergroups	添加审计用户/用户组，可在安全区中查看相关资源权限策略内容。 必须至少配置一个用户或用户组。	zone_user
Select Tag Services	选择服务的标签信息。	-
Select Resource Services	选择安全区内包含的服务及具体资源。 在“Select Resource Services”中选择服务后，需要在“Resource”列中添加具体的资源对象，例如 HDFS 服务器的文件目录、Yarn 的队列、Hive 的数据库及表、HBase 的表及列。	/testzone

例如针对 HDFS 中的“/testzone”目录创建一个安全区，配置如下：



步骤 3　单击“Save”，等待安全区添加成功。

Ranger 管理员可在“Security Zone”页面查看当前的所有安全区并单击“Edit”修改安全区的属性信息，当相关资源不需要在安全区中进行管理时，可单击“Delete”删除对应安全区。

----结束

在安全区中配置权限策略

步骤 1　使用安全区管理员用户登录 Ranger 管理页面。

步骤 2　在 Ranger 首页右上角的“Security Zone”选项的下拉列表中选择对应的安全区，即可切换至该安全区内的权限视图。

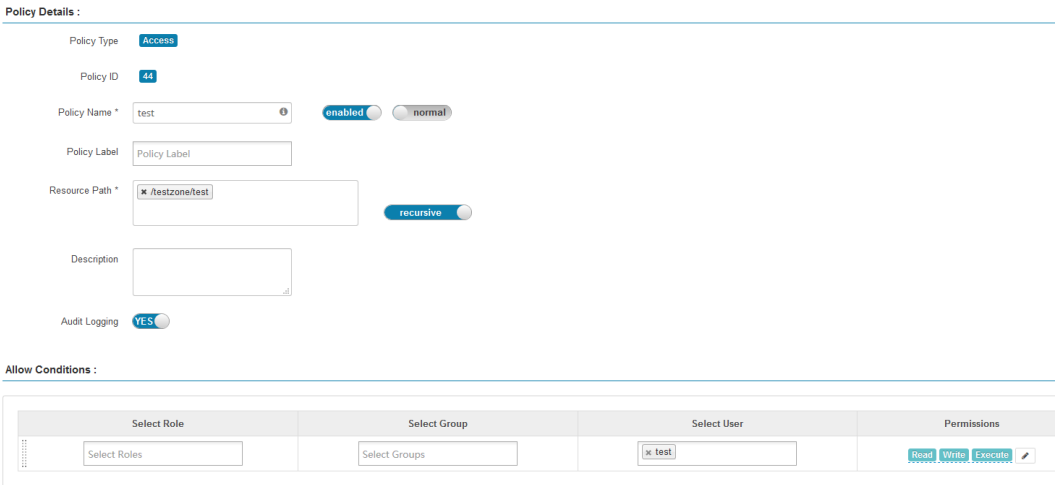
步骤 3　单击组件名称下的权限插件名称，即可进入组件安全访问策略列表页面。

说明

各组件的策略列表中，系统默认生成的条目会自动继承至安全区内，用于保证集群内的部分系统默认用户或用户组的权限。

步骤 4 单击“Add New Policy”，根据业务场景规划配置相关用户或者用户组的资源访问策略。

例如在本章节样例中，在安全区内配置一条允许“test”用户访问“/testzone/test”目录的策略：



其他不同组件的完整访问策略配置样例参考：

- 添加 HDFS 的 Ranger 访问权限策略
- 添加 HBase 的 Ranger 访问权限策略
- 添加 Hive 的 Ranger 访问权限策略
- 添加 Yarn 的 Ranger 访问权限策略
- 添加 Spark2x 的 Ranger 访问权限策略
- 添加 Kafka 的 Ranger 访问权限策略
- 添加 Storm 的 Ranger 访问权限策略

策略添加后，需等待 30 秒左右，待系统生效。

说明

- 安全区中定义的策略仅适用于区域中的资源，服务的资源被划分到安全区后，非安全区针对该资源的访问权限策略将不再生效。
- 如需配置针对当前安全区之外其他资源的访问策略，需在 Ranger 首页右上角的“Security Zone”选项中退出当前安全区后进行配置。

----结束

23.6 普通集群修改 Ranger 数据源为 Ldap

安全集群 Ranger 数据源默认为 FusionInsight Manager Ldap 用户。普通集群 Ranger 数据源默认为集群 Unix 用户。

前提条件

- 集群模式为普通模式。
- 已安装 Ranger 组件。

操作步骤

步骤 1 登录 MRS 管理控制台。

步骤 2 选择“集群列表 > 现有集群”，选中一个运行中的集群并单击集群名称，进入集群信息页面。

步骤 3 单击“节点管理”页签，选择“节点类型”为“Master”的节点组。

步骤 4 进入 **Master** 节点弹性云主机页面，单击“远程登录”按钮。

步骤 5 使用 **root** 用户登录 **Master** 节点，进入“ /opt/Bigdata/components/FusionInsight_HD_8.1.0.1/Ranger”目录，修改 configurations.xml 文件中参数“ranger.usersync.sync.source”值为“ldap”和“ranger.usersync.cookie.enabled”值为“false”。

```
<name>ranger.usersync.sync.source</name>
<value model="Sec">ldap</value>
<value model="NoSec">ldap</value>
<name>ranger.usersync.cookie.enabled</name>
<value>false</value>
```

说明

所有 Master 节点都需要修改该参数。

步骤 6 在**主 Master** 节点执行如下命令，重启 controller 进程。

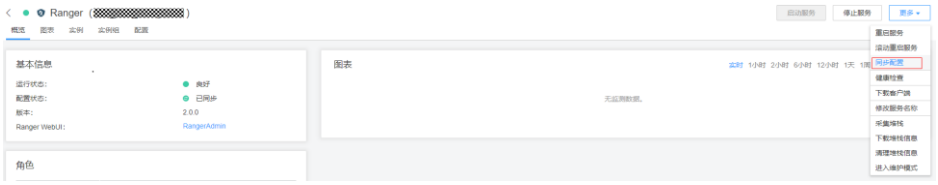
su - omm

sh /opt/Bigdata/om-server_8.1.0.1/om/sbin/restart-controller.sh

说明

重启 controller 进程会出现短暂的 Manager 页面不可访问现象，属于正常现象，待 controller 启动后，Manager 页面即可访问。

步骤 7 登录 FusionInsight Manager 页面，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本）。选择“集群 > 服务 > Ranger”。在服务“概览”页面右上角单击“更多”，选择“同步配置”。



步骤 8　在 Ranger 实例页面，勾选“UserSync”实例，选择“更多 > 重启实例”。

步骤 9　在 Ranger 服务“概览”页面，单击“RangerAdmin”，查看“Settings > Users/Groups/Roles”页面是否有 ldap 用户。

----结束

23.7 查看 Ranger 权限信息

查看 Ranger 相关权限设置信息，例如查看用户、用户组、Role。

查看 Ranger 权限信息

步骤 1　使用管理员登录 Ranger 管理页面。

步骤 2　选择“Settings > Users/Groups/Roles”，可查看系统中的用户、用户组、Roles 信息。

- Users：显示 Ranger 从 LDAP 或者 OS 同步的所有用户信息。
- Groups：显示 Ranger 从 LDAP 或者 OS 同步的所有用户组、角色信息。
- Roles：显示 Ranger 中创建的 Role 信息。

说明

- 在 FusionInsight Manager 上创建的用户、角色、用户组会定期自动同步至 Ranger，默认周期为 300000 毫秒（5 分钟）。FusionInsight Manager 中的角色和用户组在同步至 Ranger 后都变为用户组（Group）。只有被用户关联了的角色和用户组才会自动同步至 Ranger。
- Ranger 界面中创建的 Role 为用户或用户组的集合，用于灵活设置组件的权限访问策略，与 FusionInsight Manager 中的“角色”不同，请注意区分。

----结束

调整 Ranger 用户类型

步骤 1 登录 Ranger 管理页面。

调整 Ranger 用户类型须使用 Admin 类型的用户（例如 **admin**）进行操作，具体用户类型请参考 Ranger 用户类型。

步骤 2 选择“Settings > Users/Groups/Roles”，在“Users”用户列表中，单击待修改类型的用户名。

步骤 3 设置“Select Role”配置项为待修改的类型。

步骤 4 单击“Save”。

----结束

创建 Ranger Role

管理员在设置组件的权限访问策略时，可基于用户、用户组或者 Role 灵活配置，其中用户与用户组信息从 LDAP 中自动同步，Role 可手动添加。

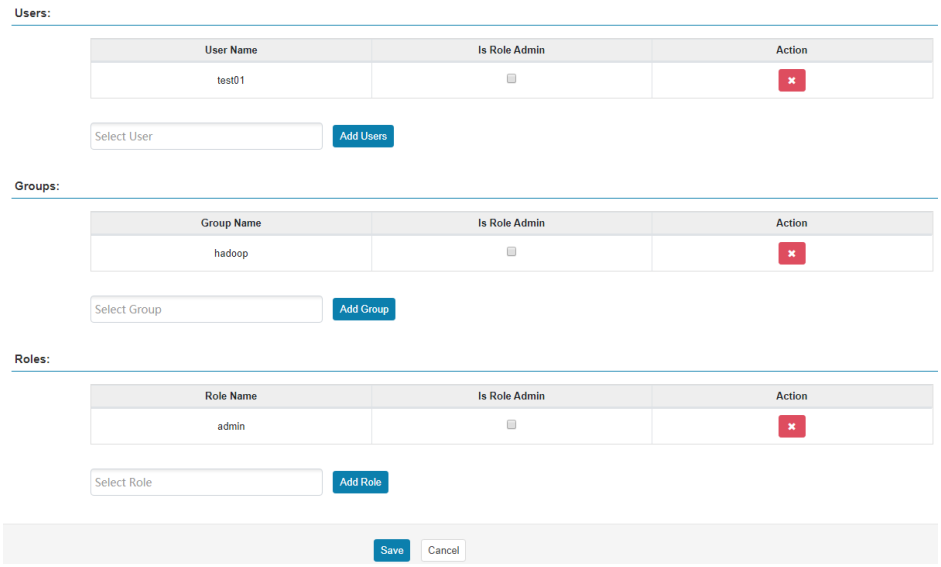
步骤 1 登录 Ranger 管理页面。

步骤 2 选择“Settings > Users/Groups/Roles > Roles > Add New Role”。

步骤 3 根据界面提示填写 Role 的名称与描述信息。

步骤 4 添加 Role 内需要包含的用户、用户组、子 Role 信息。

- 在“Users”区域，选择系统中已创建的用户，然后单击“Add Users”。
- 在“Groups”区域，选择系统中已创建的用户组，然后单击“Add Group”。
- 在“Roles”区域，选择系统中已创建的 Role，然后单击“Add Role”。



步骤 5　单击“Save”，Role 添加成功。

----结束

23.8 添加 HDFS 的 Ranger 访问权限策略

操作场景

管理员可通过 Ranger 为 HDFS 用户配置 HDFS 目录或文件的读、写和执行权限。

前提条件

- 已安装 Ranger 服务且服务运行正常。
- 已创建需要配置权限的用户、用户组或 Role。

操作步骤

步骤 1　登录 Ranger 管理页面。

步骤 2　在首页中单击“HDFS”区域的组件插件名称，例如“hacluster”。

步骤 3　单击“Add New Policy”，添加 HDFS 权限控制策略。

步骤 4　根据业务需求配置相关参数。

表23-4 HDFS 权限参数

参数名称	描述
Policy Name	策略名称，可自定义，不能与本服务内其他策略名称重复。
Policy Conditions	IP 过滤策略，可自定义，配置当前策略适用的主机节点，可填写一个或多个 IP 或 IP 段，并且 IP 填写支持“*”通配符，例如：

参数名称	描述
	192.168.1.10,192.168.1.20 或者 192.168.1.*。
Policy Label	为当前策略指定一个标签，可以根据这些标签搜索报告和筛选策略。
Resource Path	资源路径，配置当前策略适用的 HDFS 路径文件夹或文件，可填写多个值，支持使用通配符“*”（例如“/test/*”）。 如需子目录继承上级目录权限，可打开递归开关按钮。 如果父目录开启递归，同时子目录也配置了策略，以子目录策略为准。 • non-recursive：关闭递归 • recursive：打开递归
Description	策略描述信息。
Audit Logging	是否审计此策略。
Allow Conditions	策略允许条件，配置本策略内允许的权限及例外，例外条件优先级高于正常条件。 在“Select Role”、“Select Group”、“Select User”列选择已创建好的需要授予权限的 Role、用户组或用户，单击“Add Conditions”，添加策略适用的 IP 地址范围，单击“Add Permissions”，添加对应权限。 • Read：读权限 • Write：写权限 • Execute：执行权限 • Select/Deselect All：全选/取消全选 如需让当前条件中的用户或用户组管理本条策略，可勾选“Delegate Admin”使这些用户或用户组成为受委托的管理员。被委托的管理员可以更新、删除本策略，还可以基于原始策略创建子策略。 如需添加多条权限控制规则，可单击  按钮添加。如需删除权限控制规则，可单击  按钮删除。 Exclude from Allow Conditions：配置排除在允许条件之外的例外规则。
Deny All Other Accesses	是否拒绝其它所有访问。 • True：拒绝其它所有访问。 • False：设置为 false，可配置 Deny Conditions。
Deny Conditions	策略拒绝条件，配置本策略内拒绝的权限及例外，配置方法与“Allow Conditions”类似，拒绝条件的优先级高于“Allow Conditions”中配置的允许条件。

参数名称	描述
	Exclude from Deny Conditions：配置排除在拒绝条件之外的例外规则。

例如为用户“testuser”添加“/user/test”目录的写权限，配置如下：

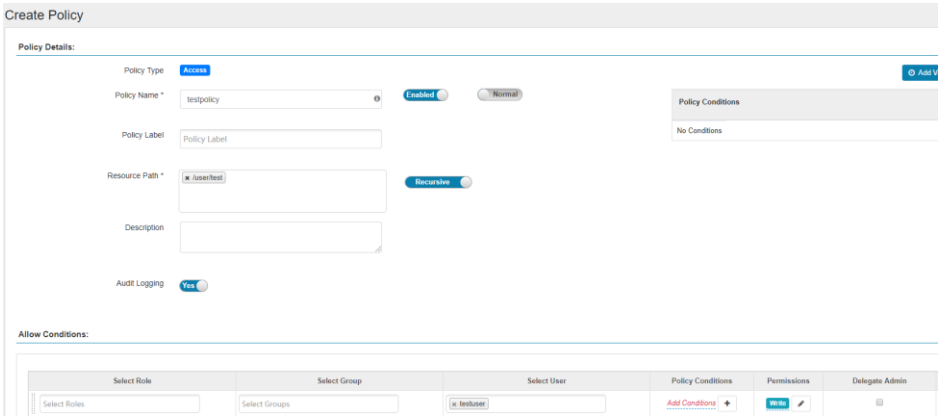


表23-5 设置权限

任务场景	角色授权操作
设置 HDFS 管理员权限	1. 在首页中单击“HDFS”区域的组件插件名称，例如“hacluster”。 2. 选择“Policy Name”为“all - path”的策略，单击  按钮编辑策略。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。
设置用户执行 HDFS 检查和 HDFS 修复的权限	1. 在“Resource Path”配置文件夹或文件。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Read”和“Execute”。
设置用户读取其他用户的目录或文件的权限	1. 在“Resource Path”配置文件夹或文件。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Read”和“Execute”。
设置用户在其他用户的文件写入数据的权限	1. 在“Resource Path”配置文件夹或文件。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Write”和“Execute”。
设置用户在其他用户的目录新	1. 在“Resource Path”配置文件夹或文件。

任务场景	角色授权操作
建或删除子文件、子目录的权限	2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Write”和“Execute”。
设置用户在其他用户的目录或文件执行的权限	1. 在“Resource Path”配置文件夹或文件。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Execute”。
设置子目录继承上级目录权限	1. 在“Resource Path”配置文件夹或文件。 2. 打开递归开关按钮，“Recursive”即为打开递归。

步骤 5 （可选）添加策略有效期。在页面右上角单击“Add Validity period”，设置“Start Time”和“End Time”，选择“Time Zone”。单击“Save”保存。如需添加多条策略有效期，可单击  按钮添加。如需删除策略有效期，可单击  按钮删除。

步骤 6 单击“Add”，在策略列表可查看策略的基本信息。等待策略生效后，验证相关权限是否正常。

如需禁用某条策略，可单击  按钮编辑策略，设置策略开关为“Disabled”。

如果不再使用策略，可单击  按钮删除策略。

----结束

23.9 添加 HBase 的 Ranger 访问权限策略

操作场景

管理员可通过 Ranger 为 HBase 用户配置 HBase 表和列族，列的权限。

前提条件

- 已安装 Ranger 服务且服务运行正常。
- 已创建需要配置权限的用户、用户组或 Role。

操作步骤

步骤 1 登录 Ranger 管理界面。

步骤 2 在首页中单击“HBASE”区域的组件插件名称如“HBase”。

步骤 3 单击“Add New Policy”，添加 HBase 权限控制策略。

步骤 4 根据业务需求配置相关参数。

表23-6 HBase 权限参数

参数名称	描述
Policy Name	策略名称，可自定义，不能与本服务内其他策略名称重复。
Policy Conditions	IP 过滤策略，可自定义，配置当前策略适用的主机节点，可填写一个或多个 IP 或 IP 段，并且 IP 填写支持“*”通配符，例如：192.168.1.10,192.168.1.20 或者 192.168.1.*。
Policy Label	为当前策略指定一个标签，可以根据这些标签搜索报告和筛选策略。
HBase Table	将适用该策略的表。 可支持通配符“*”，例如“table1:*“表示 table1 下的所有表。 “Include”策略适用于当前输入的对象，“Exclude”表示策略适用于除去当前输入内容之外的其他对象。 说明 Ranger 界面上 HBase 服务插件的“hbase.rpc.protection”参数值必须和 HBase 服务端的“hbase.rpc.protection”参数值保持一致。具体请参考 Ranger 界面添加或者修改 HBase 策略时，无法使用通配符搜索已存在的 HBase 表。
HBase Column-family	将适用该策略的列族。 “Include”策略适用于当前输入的对象，“Exclude”表示策略适用于除去当前输入内容之外的其他对象。
HBase Column	将适用该策略的列。 “Include”策略适用于当前输入的对象，“Exclude”表示策略适用于除去当前输入内容之外的其他对象。
Description	策略描述信息。
Audit Logging	是否审计此策略。
Allow Conditions	策略允许条件，配置本策略内允许的权限及例外。 在“Select Role”、“Select Group”、“Select User”列选择已创建好的需要授予权限的 Role、用户组或用户，单击“Add Conditions”，添加策略适用的 IP 地址范围，单击“Add Permissions”，添加对应权限。 • Read：读权限 • Write：写权限 • Create：创建权限 • Admin：管理权限 • Select/Deselect All：全选/取消全选 如需让当前条件中的用户或用户组管理本条策略，可勾选“Delegate Admin”使这些用户或用户组成为受委托的管理员。被委托的管理员

参数名称	描述
	可以更新、删除本策略，还可以基于原始策略创建子策略。 如需添加多条权限控制规则，可单击  按钮添加。如需删除权限控制规则，可单击  按钮删除。 Exclude from Allow Conditions：配置策略例外条件。
Deny All Other Accesses	是否拒绝其它所有访问。 • True：拒绝其它所有访问 • False：设置为 False，可配置 Deny Conditions。
Deny Conditions	策略拒绝条件，配置本策略内拒绝的权限及例外，配置方法与“Allow Conditions”类似。 拒绝条件的优先级高于“Allow Conditions”中配置的允许条件。 Exclude from Deny Conditions：配置排除在拒绝条件之外的例外规则。

表23-7 设置权限

任务场景	角色授权操作
设置 HBase 管理员权限	1. 在首页中单击“HBase”区域的组件插件名称，例如“HBase”。 2. 选择“Policy Name”为“all - table, column-family, column”的策略，单击  按钮编辑策略。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。
设置用户创建表的权限	1. 在“HBase Table”配置表名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Create”。 4. 该用户具有以下操作权限： create table drop table truncate table alter table enable table flush table flush region compact disable

任务场景	角色授权操作
	enable desc
设置用户写入数据的权限	1. 在“HBase Table”配置表名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Write”。 4. 该用户具有 put，delete，append，incr，bulkload 等操作权限。
设置用户读取数据的权限	1. 在“HBase Table”配置表名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Read”。 4. 该用户具有 get，scan 操作权限。
设置用户管理命名空间或表的权限	1. 在“HBase Table”配置表名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Admin”。 4. 该用户具有 rsgroup，peer，assign，balance 等操作权限。
设置列的读取或写入权限	1. 在“HBase Table”配置表名。 2. 在“HBase Column-family”配置列族名。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 4. 单击“Add Permissions”，勾选“Read”或者“Write”。

说明

如果用户在 hbase shell 中执行 desc 操作，需要同时给该用户赋予 hbase:qouta 表的读权限。

步骤 5 （可选）添加策略有效期。在页面右上角单击“Add Validity period”，设置“Start Time”和“End Time”，选择“Time Zone”。单击“Save”保存。如需添加多条策略有效期，可单击  按钮添加。如需删除策略有效期，可单击  按钮删除。

步骤 6 单击“Add”，在策略列表可查看策略的基本信息。等待策略生效后，验证相关权限是否正常。

如需禁用某条策略，可单击  按钮编辑策略，设置策略开关为“Disabled”。

如果不再使用策略，可单击  按钮删除策略。

----结束

23.10 添加 Hive 的 Ranger 访问权限策略

操作场景

管理员可通过 Ranger 为 Hive 用户进行相关的权限设置。Hive 默认管理员帐号为 hive，初始密码为 Hive@123。

前提条件

- 已安装 Ranger 服务且服务运行正常。
- 已创建用户需要配置权限的用户、用户组或 Role。
- 用户加入 hive 组。

操作步骤

步骤 1 登录 Ranger 管理界面。

步骤 2 在首页中单击“HADOOP SQL”区域的组件插件名称如“Hive”。

步骤 3 在“Access”页签单击“Add New Policy”，添加 Hive 权限控制策略。

步骤 4 根据业务需求配置相关参数。

表23-8 Hive 权限参数

参数名称	描述
Policy Name	策略名称，可自定义，不能与本服务内其他策略名称重复。
Policy Conditions	IP 过滤策略，可自定义，配置当前策略适用的主机节点，可填写一个或多个 IP 或 IP 段，并且 IP 填写支持“*”通配符，例如：192.168.1.10,192.168.1.20 或者 192.168.1.*。
Policy Label	为当前策略指定一个标签，您可以根据这些标签搜索报告和筛选策略。
database	将适用该策略的列 Hive 数据库名称。 “Include”策略适用于当前输入的对象，“Exclude”表示策略适用于除去当前输入内容之外的其他对象。
table	将适用该策略的 Hive 表名称。 如果需要添加基于 UDF 的策略，可切换为 UDF，然后输入 UDF 的名称。 “Include”策略适用于当前输入的对象，“Exclude”表示策略适用于除去当前输入内容之外的其他对象。
Hive Column	将适用该策略的列名，填写*时表示所有列。 “Include”策略适用于当前输入的对象，“Exclude”表示策略适用于除去

参数名称	描述
	当前输入内容之外的其他对象。
Description	策略描述信息。
Audit Logging	是否审计此策略。
Allow Conditions	策略允许条件，配置本策略内允许的权限及例外。 在“Select Role”、“Select Group”、“Select User”列选择已创建好的需要授予权限的 Role、用户组或用户，单击“Add Conditions”，添加策略适用的 IP 地址范围，然后在单击“Add Permissions”，添加对应权限。 • select：查询权限 • update：更新权限 • Create：创建权限 • Drop：drop 操作权限 • Alter：alter 操作权限 • Index：索引操作权限 • All：所有执行权限 • Read：可读权限 • Write：可写权限 • Temporary UDF Admin：临时 UDF 管理权限 • Select/Deselect All：全选/取消全选 如需添加多条权限控制规则，可单击  按钮添加。 如需当前条件中的用户或用户组管理本条策略，可勾选“Delegate Admin”，这些用户将成为受委托的管理员。被委托的管理员可以更新、删除本策略，它还可以基于原始策略创建子策略。
Deny Conditions	策略拒绝条件，配置本策略内拒绝的权限及例外，配置方法与“Allow Conditions”类型。

表23-9 设置权限

任务场景	角色授权操作
role admin 操作	1. 在首页中单击“Settings”，选择“Roles”。 2. 单击 Role Name 为 admin 的角色，在“Users”区域，单击“Select User”，选择对应用户名。 3. 点击 Add Users 按钮，在对应用户名所在行勾选“Is Role Admin”，单击“Save”保存配置。

任务场景	角色授权操作
	说明 Ranger 页面的“Settings”选项只有 rangeradmin 用户有权限访问。用户绑定 Hive 管理员角色后，在每个维护操作会话中，还需要执行以下操作： 1. 以客户端安装用户，登录安装 Hive 客户端的节点。 2. 执行以下命令配置环境变量。 例如，Hive 客户端安装目录为“/opt/hiveclient”，执行 **source /opt/hiveclient/bigdata_env** 3. 执行以下命令认证用户。 **kinit** *Hive 业务用户* 4. 执行以下命令登录客户端工具。 **beeline** 5. 执行以下命令更新用户的管理员权限。 **set role admin;**
创建库表操作	1. 在“Policy Name”填写策略名称。 2. “database”右侧填写或选择对应的数据库(如果是创建表则在“table”右侧填写或选择对应的表)，在“column”右侧填写或选择“*”。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 4. 单击“Add Permissions”，勾选“Create”。
删除库表操作	1. 在“Policy Name”填写策略名称。 2. “database”右侧填写或选择对应的数据库（如果是删除表则在“table”右侧填写或选择对应的表），在“column”右侧填写并选择“*”。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 4. 单击“Add Permissions”，勾选“Drop”。
查询操作(select、desc、show)	1. 在“Policy Name”填写策略名称。 2. “database”右侧填写或选择对应的数据库(如果是表则在“table”右侧填写或选择对应的表)，在“column”右侧填写并选择对应的列（*代表所有列）。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 4. 单击“Add Permissions”，勾选“select”。
Alter 操作	1. 在“Policy Name”填写策略名称。 2. “database”右侧填写并选择对应的数据库(如果是表则在“table”右侧填写或选择对应的表)，在“column”右侧填写或选择“*”。 3. 在“Allow Conditions”区域，单击“Select User”下选

任务场景	角色授权操作
	择框选择用户。 4. 单击“Add Permissions”，勾选“Alter”。
LOAD 操作	1. 在“Policy Name”填写策略名称。 2. “database”右侧填写或选择对应的数据库，在“table”右侧填写或选择对应的表，在“column”右侧填写并选择“*”。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 4. 单击“Add Permissions”，勾选“update”。
INSERT、DELETE 操作	1. 在“Policy Name”填写策略名称。 2. “database”右侧填写或选择对应的数据库，在“table”右侧填写或选择对应的表，在“column”右侧填写并选择“*”。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 4. 单击“Add Permissions”，勾选“update”。 5. 用户还需要具有 Yarn 任务队列的“submit”权限，权限配置参考添加 Yarn 的 Ranger 访问权限策略。
GRANT、REVOKE 操作	1. 在“Policy Name”填写策略名称。 2. “database”右侧填写或选择对应的数据库，在“table”右侧填写或选择对应的表，在“column”右侧填写并选择“*”。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 4. 勾选“Delegate Admin”。
ADD JAR 操作	1. 在“Policy Name”填写策略名称。 2. 单击“database”并在下拉菜单中选择“global”。在“global”右侧填写或选择“*”。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 4. 单击“Add Permissions”，勾选“Temporary UDF Admin”。
UDF 操作	1. 在“Policy Name”填写策略名称。 2. “database”右侧填写或选择对应的数据库，“udf”右侧填写对应的 udf 函数名。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 4. 单击“Add Permissions”，根据需求，给用户勾选相应权限（udf 支持 Create，select， Drop）。
VIEW 操作	1. 在“Policy Name”填写策略名称。 2. “database”右侧填写或选择对应的数据库，在“table”右侧填写或选择对应的 VIEW 名称，在“column”右侧

任务场景	角色授权操作
	填写并选择“*”。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 4. 单击“Add Permissions”，参照表格上述相关操作，根据需求，给用户勾选相应权限。
dfs 命令操作	执行 set role admin 操作才可使用。
其他用户库表操作	1. 参照表格上述相关操作添加对应权限。 2. 给用户添加其他用户库表的 HDFS 路径的读、写、执行权限，详情请参考添加 HDFS 的 Ranger 访问权限策略。

说明

- 如果用户在执行命令时指定了 HDFS 路径，需要给该用户添加 HDFS 路径的读、写、执行权限，详情请参考添加 HDFS 的 Ranger 访问权限策略。也可以不配置 HDFS 的 Ranger 策略，通过之前 Hive 权限插件的方式，给角色添加权限，然后把角色赋予对应用户。如果 HDFS Ranger 策略可以匹配到 Hive 库表的文件或目录权限，则优先使用 HDFS Ranger 策略。
- Ranger 策略中的 URL 策略是 hive 表存储在 obs 上的场景涉及，URL 填写对象在 obs 上的完整路径。与 URL 联合使用的 Read, Write 权限，其他场景不涉及 URL 策略。
- Ranger 策略中 global 策略仅用于和 Temprorary UDF Admin 权限联合使用，控制 UDF 包的上传。
- Ranger 策略中的 hiveservice 策略仅用于和 Service Admin 权限联合使用，用于控制命令：**kill query <queryId>** 结束正在执行的任务的权限。
- lock、index、refresh、replAdmin 权限暂不支持。
- 使用 **show grant** 命令查看表权限，表 owner 的 grantor 列统一显示为 hive 用户，其他用户 Ranger 页面赋权或后台采用 grant 命令赋权，则 grantor 显示为对应用户；若用户需要查看之前 Hive 权限插件的结果，可设置 hive-ext.ranger.previous.privileges.enable 为 true 后采用 **show grant** 查看。

步骤 5 单击“Add”，在策略列表可查看策略的基本信息。等待策略生效后，验证相关权限是否正常。

如需禁用某条策略，可单击  按钮编辑策略，设置策略开关为“Disabled”。

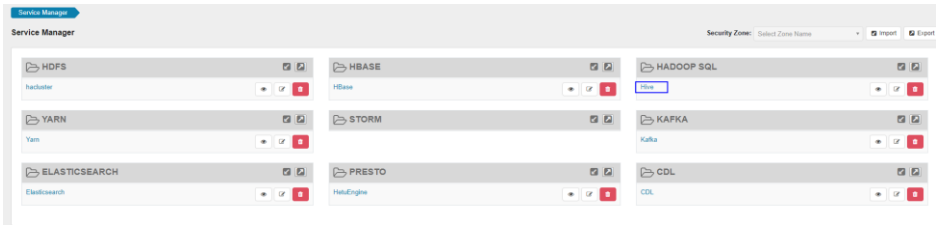
如果不再使用策略，可单击  按钮删除策略。

----结束

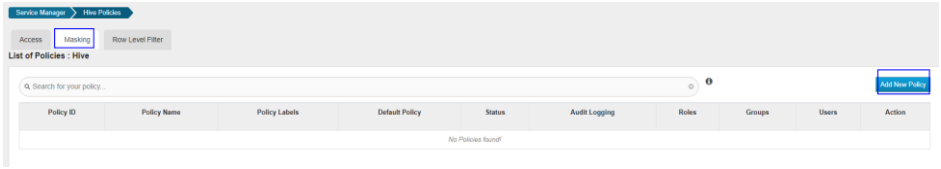
Hive 数据脱敏

Ranger 支持对 Hive 数据进行脱敏处理（Data Masking），可对用户执行的 select 操作的返回结果进行处理，以屏蔽敏感信息。

步骤 1 登录 Ranger WebUI 界面，在首页中单击“HADOOP SQL”区域的“Hive”



步骤 2 在“Masking”页签单击“Add New Policy”，添加 Hive 权限控制策略。



步骤 3 根据业务需求配置相关参数。

表23-10 Hive 数据脱敏参数

参数名称	描述
Policy Name	策略名称，可自定义，不能与本服务内其他策略名称重复。
Policy Conditions	IP 过滤策略，可自定义，配置当前策略适用的主机节点，可填写一个或多个 IP 或 IP 段，并且 IP 填写支持“*”通配符，例如：192.168.1.10,192.168.1.20 或者 192.168.1.*。
Policy Label	为当前策略指定一个标签，您可以根据这些标签搜索报告和筛选策略。
Hive Database	配置当前策略适用的 Hive 中数据库名称。
Hive Table	配置当前策略适用的 Hive 中的表名称。
Hive Column	可添加列名。
Description	策略描述信息。
Audit Logging	是否审计此策略。
Mask Conditions	在“Select Role”、“Select Group”、“Select User”列选择已创建好的需要授予权限的对象，单击“Add Conditions”，添加策略适用的 IP 地址范围，然后在单击“Add Permissions”，勾选“select”权限。 单击“Select Masking Option”，选择数据脱敏时的处理策略： • Redact：用 x 屏蔽所有字母字符，用 n 屏蔽所有数字字符。 • Partial mask: show last 4：只显示最后的 4 个字符，其他用 x

参数名称	描述
	代替。 • Partial mask: show first 4：只显示开始的 4 个字符，其他用 x 代替。 • Hash：用值的哈希值替换原值，采用的是 hive 的内置 mask_hash 函数，只对 string、char、varchar 类型的字段生效，其他类型的字段会返回 NULL 值。 • Nullify：用 NULL 值替换原值。 • Unmasked(retain original value)：原样显示。 • Date: show only year：仅显示日期字符串的年份部分，并将月份和日期默认为 01/01。 • Custom：可使用任何有效返回与被屏蔽的列中的数据类型相同的数据类型来自定义策略。 如需添加多列的脱敏策略，可单击  按钮添加。

步骤 4 单击“Add”，在策略列表可查看策略的基本信息。

步骤 5 用户通过 Hive 客户端对配置了数据脱敏策略的表执行 select 操作，系统将对数据进行处理后进行展示。

说明

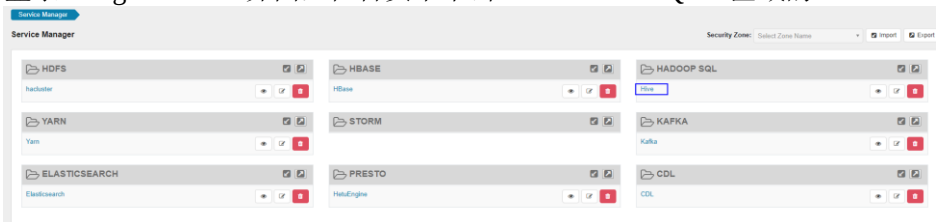
处理数据需要用户同时具有向 Yarn 队列提交任务的权限。

----结束

Hive 行级别数据过滤

Ranger 支持用户对 Hive 数据表执行 select 操作时进行行级别的数据过滤。

步骤 1 登录 Ranger WebUI 界面，在首页中单击“HADOOP SQL”区域的“Hive”。



步骤 2 在“Row Level Filter”页签单击“Add New Policy”，添加行数据过滤策略。

步骤 3 根据业务需求配置相关参数。

表23-11 Hive 行数据过滤参数

参数名称	描述
Policy Name	策略名称，可自定义，不能与本服务内其他策略名称重复。
Policy Conditions	IP 过滤策略，可自定义，配置当前策略适用的主机节点，可填写一个或多个 IP 或 IP 段，并且 IP 填写支持“*”通配符，例如：192.168.1.10,192.168.1.20 或者 192.168.1.*。
Policy Label	为当前策略指定一个标签，您可以根据这些标签搜索报告和筛选策略。
Hive Database	配置当前策略适用的 Hive 中数据库名称。
Hive Table	配置当前策略适用的 Hive 中的表名称。
Description	策略描述信息。
Audit Logging	是否审计此策略。
Row Filter Conditions	在“Select Role”、“Select Group”、“Select User”列选择已创建好的需要授予权限的对象，单击“Add Conditions”，添加策略适用的 IP 地址范围，然后在单击“Add Permissions”，勾选“select”权限。 单击“Row Level Filter”，填写数据过滤规则。 例如过滤表 A 中“name”列“zhangsan”行的数据，过滤规则为：name <> 'zhangsan'。更多信息可参考 Ranger 官方文档。 如需添加更多规则，可单击  按钮添加。

步骤 4 单击“Add”，在策略列表可查看策略的基本信息。

步骤 5 用户通过 Hive 客户端对配置了数据脱敏策略的表执行 select 操作，系统将对数据进行处理后进行展示。

说明

处理数据需要用户同时具有向 Yarn 队列提交任务的权限。

----结束

23.11 添加 Yarn 的 Ranger 访问权限策略

操作场景

管理员可通过 Ranger 为 Yarn 用户配置 Yarn 管理员权限以及 Yarn 队列资源管理权限。

前提条件

- 已安装 Ranger 服务且服务运行正常。
- 已创建需要配置权限的用户、用户组或 Role。

操作步骤

步骤 1 登录 Ranger 管理界面。

步骤 2 在首页中单击“YARN”区域的组件插件名称如“Yarn”。

步骤 3 单击“Add New Policy”，添加 Yarn 权限控制策略。

步骤 4 根据业务需求配置相关参数。

表23-12 Yarn 权限参数

参数名称	描述
Policy Name	策略名称，可自定义，不能与本服务内其他策略名称重复。
Policy Conditions	IP 过滤策略，可自定义，配置当前策略适用的主机节点，可填写一个或多个 IP 或 IP 段，并且 IP 填写支持“*”通配符，例如：192.168.1.10,192.168.1.20 或者 192.168.1.*。
Policy Label	为当前策略指定一个标签，可以根据这些标签搜索报告和筛选策略。
Queue	队列名称，支持通配符“*”。 如需子队列继承上级队列权限，可打开递归开关按钮。 • Non-recursive：关闭递归 • Recursive：打开递归
Description	策略描述信息。
Audit Logging	是否审计此策略。
Allow Conditions	策略允许条件，配置本策略内允许的权限及例外。 在“Select Role”、“Select Group”、“Select User”列选择已创建好的需要授予权限的 Role、用户组或用户，单击“Add Conditions”，添加策略适用的 IP 地址范围，单击“Add Permissions”，添加对应权限。 • submit-app：提交队列任务权限 • admin-queue：管理队列任务权限 • Select/Deselect All：全选/取消全选 如需让当前条件中的用户或用户组管理本条策略，可勾选“Delegate Admin”使这些用户成为受委托的管理员。被委托的管理员可以更新、删除本策略，它还可以基于原始策略创建子策略。

参数名称	描述
	如需添加多条权限控制规则，可单击  按钮添加。如需删除权限控制规则，可单击  按钮删除。 Exclude from Allow Conditions：配置策略例外条件。
Deny All Other Accesses	是否拒绝其它所有访问。 • True：拒绝其它所有访问 • False：设置为 False，可配置 Deny Conditions。
Deny Conditions	策略拒绝条件，配置本策略内拒绝的权限及例外，配置方法与“Allow Conditions”类似。拒绝条件的优先级高于“Allow Conditions”中配置的允许条件。 Exclude from Deny Conditions：配置排除在拒绝条件之外的例外规则。

表23-13 设置权限

任务场景	角色授权操作
设置 Yarn 管理员权限	1. 在首页中单击“YARN”区域的组件插件名称，例如“Yarn”。 2. 选择“Policy Name”为“all - queue”的策略，单击  按钮编辑策略。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。
设置用户在指定 Yarn 队列提交任务的权限	1. 在“Queue”配置队列名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“submit-app”。
设置用户在指定 Yarn 队列管理任务的权限	1. 在“Queue”配置队列名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“admin-queue”。

步骤 5 （可选）添加策略有效期。在页面右上角单击“Add Validity period”，设置“Start Time”和“End Time”，选择“Time Zone”。单击“Save”保存。如需添加多条策略有效期，可单击  按钮添加。如需删除策略有效期，可单击  按钮删除。

步骤 6 单击“Add”，在策略列表可查看策略的基本信息。等待策略生效后，验证相关权限是否正常。

如需禁用某条策略，可单击  按钮编辑策略，设置策略开关为“Disabled”。

如果不再使用策略，可单击  按钮删除策略。

----结束

说明

Ranger Yarn 上面各个权限之间相互独立，没有语义上的包含与被包含关系。当前支持下面两种权限：

- submit-app：提交队列任务权限
- admin-queue：管理队列任务权限

虽然 admin-queue 也有提交任务的权限，但和 submit-app 权限之间并没有包含关系。

23.12 添加 Spark2x 的 Ranger 访问权限策略

操作场景

管理员可通过 Ranger 为 Spark2x 用户进行相关的权限设置。

说明

1. Spark2x 开启或关闭 Ranger 鉴权后，需要重启 Spark2x 服务。
2. 需要重新下载客户端，或手动刷新客户端配置文件“*客户端安装目录*/Spark2x/spark/conf/spark-defaults.conf”：

   开启 Ranger 鉴权：spark.ranger.plugin.authorization.enable=true

   关闭 Ranger 鉴权：spark.ranger.plugin.authorization.enable=false
3. Spark2x 中，spark-beeline（即连接到 JDBCServer 的应用）支持 Ranger 的 IP 过滤策略（即 Ranger 权限策略中的 **Policy Conditions**），spark-submit 与 spark-sql 不支持。

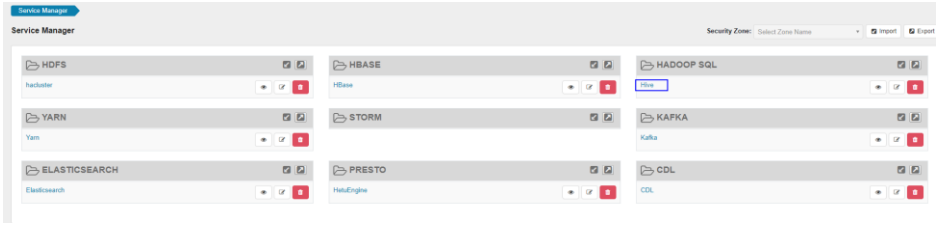
前提条件

- 已安装 Ranger 服务且服务运行正常。
- 已启用 Hive 服务的 Ranger 鉴权功能，并且重启 Hive 服务后，重启了 Spark2x 服务。
- 已创建用户需要配置权限的用户、用户组或 Role。
- 创建的用户已加入 hive 用户组。

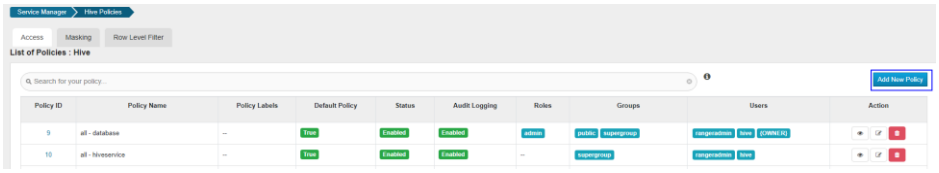
操作步骤

步骤 1 登录 Ranger 管理界面。

步骤 2 在首页中单击“HADOOP SQL”区域的组件插件名称如“Hive”。



步骤 3　在“Access”页签单击“Add New Policy”，添加 Spark2x 权限控制策略。



步骤 4　根据业务需求配置相关参数。

表23-14 Spark2x 权限参数

参数名称	描述
Policy Name	策略名称，可自定义，不能与本服务内其他策略名称重复。
Policy Conditions	IP 过滤策略，可自定义，配置当前策略适用的主机节点，可填写一个或多个 IP 或 IP 段，并且 IP 填写支持“*”通配符，例如：192.168.1.10,192.168.1.20 或者 192.168.1.*。
Policy Label	为当前策略指定一个标签，您可以根据这些标签搜索报告和筛选策略。
database	适用该策略的 Spark2x 数据库名称。 “Include”策略适用于当前输入的对象，“Exclude”表示策略适用于除去当前输入内容之外的其他对象。
table	适用该策略的 Spark2x 表名称。 如果需要添加基于 UDF 的策略，可切换为 UDF，然后输入 UDF 的名称。 “Include”策略适用于当前输入的对象，“Exclude”表示策略适用于除去当前输入内容之外的其他对象。
column	适用该策略的列名，填写*时表示所有列。 “Include”策略适用于当前输入的对象，“Exclude”表示策略适用于除去当前输入内容之外的其他对象。
Description	策略描述信息。
Audit Logging	是否审计此策略。
Allow Conditions	策略允许条件，配置本策略内允许的权限及例外。 在“Select Role”、“Select Group”、“Select User”列选择已创建好的需要授予权限的 Role、用户组或用户，单击“Add Conditions”，添加策略适用的 IP 地址范围，然后在单击“Add

参数名称	描述
	Permissions”，添加对应权限。 • select：查询权限 • update：更新权限 • Create：创建权限 • Drop：drop 操作权限 • Alter：alter 操作权限 • Index：索引操作权限 • All：所有执行权限 • Read：可读权限 • Write：可写权限 • Temporary UDF Admin：临时 UDF 管理权限 • Select/Deselect All：全选/取消全选 如需添加多条权限控制规则，可单击  按钮添加。 如需当前条件中的用户或用户组管理本条策略，可勾选“Delegate Admin”，这些用户将成为受委托的管理员。被委托的管理员可以更新、删除本策略，它还可以基于原始策略创建子策略。
Deny Conditions	策略拒绝条件，配置本策略内拒绝的权限及例外，配置方法与“Allow Conditions”类型。

表23-15 设置权限

任务场景	角色授权操作
role admin 操作	1. 在首页中单击“Settings”，选择“Roles > Add New Role”。 2. 设置“Role Name”为“admin”，在“Users”区域，单击“Select User”，选择对应用户名。 3. 点击 Add Users 按钮，在对应用户名所在行勾选“Is Role Admin”，单击“Save”保存配置。 说明 用户绑定 Hive 管理员角色后，在每个维护操作会话中，还需要执行以下操作： 1. 以客户端安装用户，登录安装 Hive 客户端的节点。 2. 执行以下命令配置环境变量。 例如，Spark2x 客户端安装目录为“/opt/client”，执行 **source /opt/client/bigdata_env** 3. 执行以下命令认证用户。

任务场景	角色授权操作
	kinit Spark2x *业务用户* 4. 执行以下命令登录客户端工具。 **spark-beeline** 5. 执行以下命令更新用户的管理员权限。 **set role admin;**
创建库表操作	1. 在“Policy Name”填写策略名称。 2. “database”右侧填写并选择对应的数据库（如果是创建库，需填写将要创建的库名称，或填写“*”表示任意名称的数据库，然后选择所写名称），在“table”与“column”右侧填写并选择对应的表名称、列名称，均支持通配符（“*”）匹配。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 4. 单击“Add Permissions”，勾选“Create”。
删除库表操作	1. 在“Policy Name”填写策略名称。 2. “database”右侧填写并选择对应的数据库（如果是删除库，需填写将要创建的库名称，或填写“*”表示任意名称的数据库，然后选择所写名称），在“table”与“column”右侧填写并选择对应的表名称、列名称，均支持通配符（“*”）匹配。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 4. 单击“Add Permissions”，勾选“Drop”。 说明 对于 CarbonData 表，只有对应表的 OWNER，才能执行“drop”操作。
ALTER 操作	1. 在“Policy Name”填写策略名称。 2. “database”右侧填写并选择对应的数据库，在“table”右侧填写并选择对应的表，在“column”右侧填写并选择对应的列名称，支持通配符（“*”）匹配。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 4. 单击“Add Permissions”，勾选“Alter”。
LOAD 操作	1. 在“Policy Name”填写策略名称。 2. “database”右侧填写并选择对应的数据库，在“table”右侧填写并选择对应的表，在“column”右侧填写并选择对应的列名称，支持通配符（“*”）匹配。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 4. 单击“Add Permissions”，勾选“update”。
INSERT 操作	1. 在“Policy Name”填写策略名称。

任务场景	角色授权操作
	2. “database”右侧填写并选择对应的数据库，在“table”右侧填写并选择对应的表，在“column”右侧填写并选择对应的列名称，支持通配符（“*”）匹配。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 4. 单击“Add Permissions”，勾选“update”。 5. 用户还需要具有 Yarn 任务队列的“submit-app”权限，默认情况下，hadoop 用户组具有向所有 Yarn 任务队列“submit-app”权限。具体配置请参考添加 Yarn 的 Ranger 访问权限策略。
GRANT 操作	1. 在“Policy Name”填写策略名称。 2. “database”右侧填写并选择对应的数据库，在“table”右侧填写并选择对应的表，在“column”右侧填写并选择对应的列名称，支持通配符（“*”）匹配。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 4. 勾选“Delegate Admin”。
ADD JAR 操作	1. 在“Policy Name”填写策略名称。 2. 单击“database”并在下拉菜单中选择“global”。在“global”右侧填写并选择“*”。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 4. 单击“Add Permissions”，勾选“Temporary UDF Admin”。
VIEW 与 INDEX 权限	1. 在“Policy Name”填写策略名称。 2. “database”右侧填写并选择对应的数据库，在“table”右侧填写并选择对应的 VIEW 或 INDEX 名称，在“column”右侧填写并选择“*”。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 4. 单击“Add Permissions”，参照表格上述相关操作，根据需求，给用户勾选相应权限。
其他用户库表操作	1. 参照表格上述操作添加对应权限。 2. 给当前用户添加其他用户库表的 HDFS 路径的读、写、执行权限，具体配置请参考添加 HDFS 的 Ranger 访问权限策略。

说明

在 Ranger 上为用户添加 Spark SQL 的访问策略后，需要在 HDFS 的访问策略中添加相应的路径访问策略，否则无法访问数据文件，具体请参考添加 HDFS 的 Ranger 访问权限策略。

- Ranger 策略中 global 策略仅用于联合 Temprorary UDF Admin 权限，用来控制 UDF 包的上传。
- 通过 Ranger 对 Spark SQL 进行权限控制时，不支持 empower 语法。

步骤 5 单击“Add”，在策略列表可查看策略的基本信息。等待策略生效后，验证相关权限是否正常。

如果需要禁用某条策略，可单击  按钮编辑该策略，设置策略开关为“Disabled”。

如果不再使用某条策略，可单击  按钮删除该策略。

----结束

Spark2x 表数据脱敏

Ranger 支持对 Spark2x 数据进行脱敏处理（Data Masking），可对用户执行的 select 操作的返回结果进行处理，以屏蔽敏感信息。

步骤 1 登录 Ranger WebUI 界面，在首页单击“HADOOP SQL”区域的组件插件名称如“Hive”。

步骤 2 在“Masking”页签单击“Add New Policy”，添加 Spark2x 权限控制策略。

步骤 3 根据业务需求配置相关参数。

表23-16 Spark2x 数据脱敏参数

参数名称	描述
Policy Name	策略名称，可自定义，不能与本服务内其他策略名称重复。
Policy Conditions	IP 过滤策略，可自定义，配置当前策略适用的主机节点，可填写一个或多个 IP 或 IP 段，并且 IP 填写支持“*”通配符，例如：192.168.1.10,192.168.1.20 或者 192.168.1.*。
Policy Label	为当前策略指定一个标签，您可以根据这些标签搜索报告和筛选策略。
Hive Database	配置当前策略适用的 Spark2x 中的数据库名称。
Hive Table	配置当前策略适用的 Spark2x 中的表名称。
Hive Column	配置当前策略适用的 Spark2x 中的列名称。
Description	策略描述信息。
Audit Logging	是否审计此策略。
Mask Conditions	在“Select Group”、“Select User”列选择已创建好的需要授予权限的用户组或用户，单击“Add Conditions”，添加策略适用的 IP 地址范围，然后在单击“Add Permissions”，勾选“select”权限。 单击“Select Masking Option”，选择数据脱敏时的处理策略：

参数名称	描述
	• Redact：用 x 屏蔽所有字母字符，用 n 屏蔽所有数字字符。 • Partial mask: show last 4：只显示最后的 4 个字符。 • Partial mask: show first 4：只显示开始的 4 个字符。 • Hash：对数据进行 Hash 处理。 • Nullify：用 NULL 值替换原值。 • Unmasked(retain original value)：不脱敏，显示原数据。 • Date: show only year：日期格式数据只显示年份信息。 • Custom：可使用任何有效 Hive UDF（返回与被屏蔽的列中的数据类型相同的数据类型）来自定义策略。 如需添加多列的脱敏策略，可单击  按钮添加。
Deny Conditions	策略拒绝条件，配置本策略内拒绝的权限及例外，配置方法与“Allow Conditions”类型。

----结束

Spark2x 行级别数据过滤

Ranger 支持用户对 Spark2x 数据表执行 select 操作时进行行级别的数据过滤。

步骤 1 登录 Ranger WebUI 界面，在首页单击“HADOOP SQL”区域的组件插件名称如“Hive”。

步骤 2 在“Row Level Filter”页签单击“Add New Policy”，添加行数据过滤策略。

步骤 3 根据业务需求配置相关参数。

表23-17 Spark2x 行数据过滤参数

参数名称	描述
Policy Name	策略名称，可自定义，不能与本服务内其他策略名称重复。
Policy Conditions	IP 过滤策略，可自定义，配置当前策略适用的主机节点，可填写一个或多个 IP 或 IP 段，并且 IP 填写支持“*”通配符，例如：192.168.1.10,192.168.1.20 或者 192.168.1.*。
Policy Label	为当前策略指定一个标签，您可以根据这些标签搜索报告和筛选策略。
Hive Database	配置当前策略适用的 Saprk2x 中的数据库名称。
Hive Table	配置当前策略适用的 Saprk2x 中的表名称。
Description	策略描述信息。

参数名称	描述
Audit Logging	是否审计此策略。
Row Filter Conditions	在“Select Role”、“Select Group”、“Select User”列选择已创建好的需要授予权限的对象，单击“Add Conditions”，添加策略适用的 IP 地址范围，然后在单击“Add Permissions”，勾选“select”权限。 单击“Row Level Filter”，填写数据过滤规则。 例如过滤表 A 中“name”列“zhangsan”行的数据，过滤规则为：name <> 'zhangsan'。更多信息可参考 Ranger 官方文档。 如需添加更多规则，可单击  按钮添加。

步骤 4 单击“Add”，在策略列表可查看策略的基本信息。

步骤 5 用户通过 Saprk2x 客户端对配置了数据脱敏策略的表执行 select 操作，系统将对数据进行处理后进行展示。

----结束

23.13 添加 Kafka 的 Ranger 访问权限策略

操作场景

管理员可通过 Ranger 为 Kafka 用户配置 Kafka 主题的读、写、管理权限以及集群的管理权限，本章节以为用户“test”添加 “test” 主题的“生产”权限。

前提条件

- 已安装 Ranger 服务且服务运行正常。
- 已创建用户需要配置权限的用户、用户组或 Role。

操作步骤

步骤 1 登录 Ranger 管理界面。

步骤 2 在首页中单击“KAFKA”区域的组件插件名称如“Kafka”。

步骤 3 单击“Add New Policy”，添加 Kafka 权限控制策略。

步骤 4 根据业务需求配置相关参数。

表23-18 Kafka 权限参数

参数名称	描述
Policy Type	Access。

参数名称	描述
Policy Conditions	IP 过滤策略，可自定义，配置当前策略适用的主机节点，可填写一个或多个 IP 或 IP 段，并且 IP 填写支持“*”通配符，例如：192.168.1.10,192.168.1.20 或者 192.168.1.*。
Policy Name	策略名称，可自定义，不能与本服务内其他策略名称重复。
Policy Label	为当前策略指定一个标签，您可以根据这些标签搜索报告和筛选策略。
topic	配置当前策略适用的 topic 名，可以填写多个值。这里支持通配符，例如：test、test*、*。 “Include”策略适用于当前输入的对象，“Exclude”表示策略适用于除去当前输入内容之外的其他对象。
Description	策略描述信息。
Audit Logging	是否审计此策略。
Allow Conditions	策略允许条件，配置本策略内允许的权限及例外，例外条件优先级高于正常条件。 在“Select Role”、“Select Group”、“Select User”列选择已创建好的需要授予权限的 Role、用户组或用户。 单击“Add Conditions”，添加策略适用的 IP 地址范围，单击“Add Permissions”，添加对应权限。 • Publish：生产权限。 • Consume：消费权限。 • Describe：查询权限。 • Create： 创建主题权限。 • Delete： 删除主题权限。 • Describe Configs：查询配置权限。 • Alter：修改 topic 的 partition 数量的权限。 • Alter Configs：修改配置权限。 • Select/Deselect All：全选/取消全选。 如需添加多条权限控制规则，可单击  按钮添加。 如需当前条件中的用户或用户组管理本条策略，可勾选“Delegate Admin”，这些用户将成为受委托的管理员。被委托的管理员可以更新、删除本策略，它还可以基于原始策略创建子策略。
Deny Conditions	策略拒绝条件，配置本策略内拒绝的权限及例外，配置方法与“Allow Conditions”类型，拒绝条件的优先级高于“Allow Conditions”中配置的允许条件。

例如为用户“testuser”添加“test”主题的生产权限，配置如下：

图23-3 Kafka 权限参数

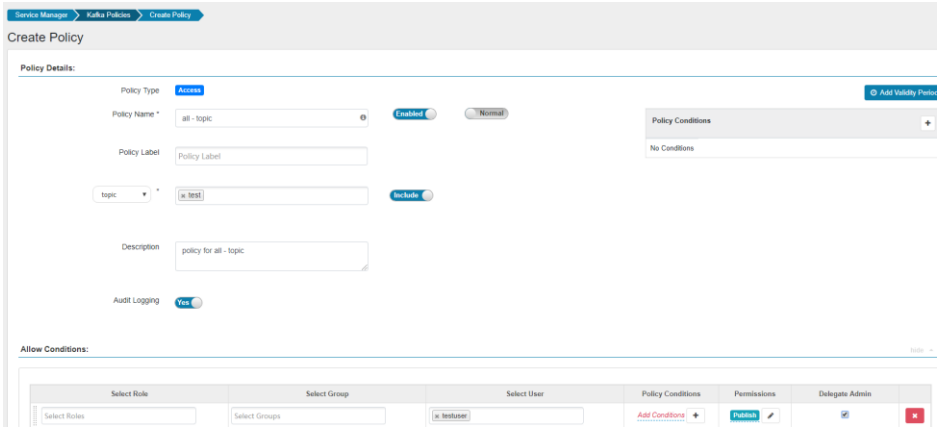


表23-19 设置权限

任务场景	角色授权操作
设置 Kafka 管理员权限	1. 在首页中单击“KAFKA”区域的组件插件名称，例如“Kafka”。 2. 选择“Policy Name”为“all - topic”的策略，单击  按钮编辑策略。 3. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 4. 单击“Add Permissions”，勾选“Select/Deselect All”。
设置用户对 Topic 的创建权限	1. 在“topic”配置 Topic 名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Create”。 说明 目前 Kafka 内核支持"--zookeeper"和"--bootstrap-server"两种方式创建 Topic,社区将会在后续的版本中删掉对"--zookeeper"的支持，所以建议用户使用"--bootstrap-server"的方式创建 Topic。 注意：目前 Kafka 只支持"--bootstrap-server"方式创建 Topic 行为的鉴权，不支持对"--zookeeper"方式的鉴权
设置用户对 Topic 的删除权限	1. 在“topic”配置 Topic 名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Delete”。

任务场景	角色授权操作
	说明 目前 Kafka 内核支持"--zookeeper"和"--bootstrap-server"两种方式删除 Topic，社区将会在后续的版本中删掉对"--zookeeper"的支持，所以建议用户使用"--bootstrap-server"的方式删除 Topic。 注意：目前 Kafka 只支持对"--bootstrap-server"方式删除 Topic 行为的鉴权，不支持对"--zookeeper"方式的鉴权
设置用户对 Topic 的查询权限	1. 在“topic”配置 Topic 名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Describe”和“Describe Configs”。 说明 目前 Kafka 内核支持"--zookeeper"和"--bootstrap-server"两种方式查询 Topic，社区将会在后续的版本中删掉对"--zookeeper"的支持，所以建议用户使用"--bootstrap-server"的方式查询 Topic。 注意：目前 Kafka 只支持对"--bootstrap-server"方式查询 Topic 行为的鉴权，不支持对"--zookeeper"方式的鉴权
设置用户对 Topic 的生产权限	1. 在“topic”配置 Topic 名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Publish”。
设置用户对 Topic 的消费权限	1. 在“topic”配置 Topic 名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Consume”。 说明 因为消费 Topic 时，涉及到 Offset 的管理操作，必须同时开启 ConsumerGroup 的“Consume”权限，详见“设置用户对 ConsumerGroup Offsets 的提交权限”
设置用户对 Topic 的扩容权限（增加分区）	1. 在“topic”配置 Topic 名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Alter”。
设置用户对 Topic 的配置修改权限	当前 Kafka 内核暂不支持基于“--bootstrap-server”的 Topic 参数修改行为，故当前 Ranger 不支持对此行为的鉴权操作。
设置用户对 Cluster 的所有管	1. 在“cluster”右侧输入并选择集群名。

任务场景	角色授权操作
理权限	2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Kafka Admin”。
设置用户对 Cluster 的创建权限	1. 在首页中单击“KAFKA”区域的组件插件名称，例如“Kafka”。 2. 选择“Policy Name”为“all - cluster”的策略，单击  按钮编辑策略。 3. 在“cluster”右侧输入并选择集群名。 4. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 5. 单击“Add Permissions”，勾选“Create”。 说明 对于 Cluster 的 Create 操作鉴权主要涉及以下两个场景： 1. 集群开启了“auto.create.topics.enable”参数后，客户端向服务的还未创建的 Topic 发送数据的场景，此时会判断用户是否有集群的 Create 权限 2. 对于用户创建大量 Topic 的场景，如果授予用户 Cluster Create 权限，那么该用户可以在集群内部创建任意 Topic
设置用户对 Cluster 的配置修改权限	1. 在“cluster”右侧输入并选择集群名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Alter Configs”。 说明 此处的配置修改权限，指的是 Broker、Broker Logger 的配置权限。 当授予用户配置修改权限后，即使不授予配置查询权限也可查询配置详情（配置修改权限高于且包含配置查询权限）。
设置用户对 Cluster 的配置查询权限	1. 在“cluster”右侧输入并选择集群名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Describe”和“Describe Configs”。 说明 此处查询指的是查询集群内的 Broker、Broker Logger 信息。该查询不涉及 Topic。
设置用户对 Cluster 的 Idempotent Write 权限	1. 在“cluster”右侧输入并选择集群名。 2. 在“Allow Conditions”区域，单击“Select User”

任务场景	角色授权操作
	下选择框选择用户。 3. 单击“Add Permissions”，勾选“Idempotent Write”。 说明 此权限会对用户客户端的 Idempotent Produce 行为进行鉴权。
设置用户对 Cluster 的分区迁移权限管理	1. 在“cluster”右侧输入并选择集群名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Alter”。 说明 Cluster 的 Alter 权限可以对以下三种场景进行权限控制： 1. Partition Reassign 场景下，迁移副本的存储目录。 2. 集群里各分区内部 leader 选举。 3. Acl 管理（添加或删除）。 其中步骤 4.1 和步骤 4.2 都是集群内部 Controller 与 Broker 间、Broker 与 Broker 间的操作，创建集群时，默认授予内置 kafka 用户此权限，普通用户授予此权限没有意义。 步骤 4.3 涉及 Acl 的管理，Acl 设计的就是用于鉴权，由于目前 kafka 鉴权已全部托管给 Ranger，所以这个场景也基本不涉及（配置后亦不生效）。
设置用户对 Cluster 的 Cluster Action 权限	1. 在“cluster”右侧输入并选择集群名。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Cluster Action”。 说明 此权限主要对集群内部副本主从同步、节点间通信进行控制，在集群创建时已经授权给内置 kakfa 用户，普通用户授予此权限没有意义。
设置用户对 TransactionalId 的权限	1. 在首页中单击“KAFKA”区域的组件插件名称，例如“Kafka”。 2. 选择“Policy Name”为“all - transactionalid”的策略，单击  按钮编辑策略。 1. 在“transactionalid”配置事务 ID。 2. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 3. 单击“Add Permissions”，勾选“Publish”和"Describe"。

任务场景	角色授权操作
	说明 “Publish”权限主要对用户开启了事务特性的客户端请求进行鉴权，例如事务开启、结束、提交 offset、事务性数据生产等行为。 “Describe”权限主要对于开启事务特性的客户端与 Coordinator 的请求进行鉴权。 建议在开启事务特性的场景下，给用户同时授予“Publish”和“Describe”权限。
设置用户对 DelegationToken 的权限	1. 在首页中单击“KAFKA”区域的组件插件名称，例如“Kafka”。 2. 选择“Policy Name”为“all - delegationtoken”的策略，单击  按钮编辑策略。 3. 在“delegationtoken”配置 delegationtoken。 4. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 5. 单击“Add Permissions”，勾选“ Describe”。 说明 当前 Ranger 对 DelegationToken 的鉴权控制仅限于对查询的权限控制，不支持对 DelegationToken 的 create、renew、expire 操作的权限控制。
设置用户对 ConsumerGroup Offsets 的查询权限	1. 在首页中单击“KAFKA”区域的组件插件名称，例如“Kafka”。 2. 选择“Policy Name”为“all - consumergroup”的策略，单击  按钮编辑策略。 3. 在“consumergroup”配置需要管理的 consumergroup。 4. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 5. 单击“Add Permissions”，勾选“Describe”。
设置用户对 ConsumerGroup Offsets 的提交权限	1. 在首页中单击“KAFKA”区域的组件插件名称，例如“Kafka”。 2. 选择“Policy Name”为“all - consumergroup”的策略，单击  按钮编辑策略。 3. 在“consumergroup”配置需要管理的 consumergroup。 4. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 5. 单击“Add Permissions”，勾选“Consume”。 说明

任务场景	角色授权操作
	当给用户授予了 ConsumerGroup 的“Consume”权限后，用户会同时被授予“Describe”权限。
设置用户对 ConsumerGroup Offsets 的删除权限	1. 在首页中单击“KAFKA”区域的组件插件名称，例如“Kafka”。 2. 选择“Policy Name”为“all - consumergroup”的策略，单击  按钮编辑策略。 3. 在“consumergroup”配置需要管理的consumergroup。 4. 在“Allow Conditions”区域，单击“Select User”下选择框选择用户。 5. 单击“Add Permissions”，勾选“Delete”。 说明 当给用户授予了 ConsumerGroup 的“Delete”权限后，用户会同时被授予“Describe”权限。

步骤 5 （可选）添加策略有效期。在页面右上角单击“Add Validity period”，设置“Start Time”和“End Time”，选择“Time Zone”。单击“Save”保存。如需添加多条策略有效期，可单击  按钮添加。如需删除策略有效期，可单击  按钮删除。

步骤 6 单击“Add”，在策略列表可查看策略的基本信息。等待策略生效后，验证相关权限是否正常。

如需禁用某条策略，可单击  按钮编辑策略，设置策略开关为“Disabled”。

如果不再使用策略，可单击  按钮删除策略。

----结束

23.14 添加 Storm 的 Ranger 访问权限策略

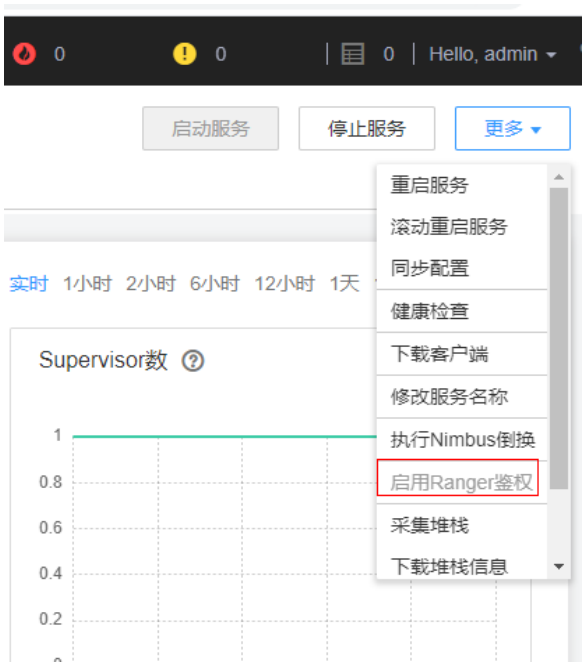
操作场景

管理员可通过 Ranger 为 Storm 用户进行相关的权限设置。

前提条件

- 已安装 Ranger 服务且服务运行正常。
- 已创建用户需要配置权限的用户、用户组或 Role。
- 页面已启用 Ranger 鉴权开关，该按钮控制是否启用 Ranger 插件进行权限管控，启用则使用 Ranger 鉴权，否则使用组件自身鉴权机制。

图23-4 启用 Ranger 鉴权



操作步骤

步骤 1 登录 Ranger WebUI 界面，在首页中单击“STORM”区域的“Storm”。

步骤 2 单击“Add New Policy”，添加 Storm 权限控制策略。

步骤 3 根据业务需求配置相关参数。

表23-20 Storm 权限参数

参数名称	描述
Policy Conditions	IP 过滤策略，可自定义，配置当前策略适用的主机节点，可填写一个或多个 IP 或 IP 段，并且 IP 填写支持“*”通配符，例如：192.168.1.10,192.168.1.20 或者 192.168.1.*。
Policy Name	策略名称，可自定义，不能与本服务内其他策略名称重复。 “include”策略适用于当前输入的对象，“exclude”表示策略适用于除去当前输入内容之外的其他对象。
Policy Label	为当前策略指定一个标签，您可以根据这些标签搜索报告和筛选策略。
Storm Topology	配置当前策略适用的拓扑名称。可以填写多个值。
Description	策略描述信息。

参数名称	描述
Audit Logging	是否审计此策略。
Allow Conditions	策略允许条件，配置本策略内允许的权限及例外。 在“Select Role”、“Select Group”、“Select User”列选择已创建好的需要授予权限的 Role、用户组或用户，单击“Add Conditions”，添加策略适用的 IP 地址范围，单击“Add Permissions”，添加对应权限。 • Submit Topology：提交拓扑。 说明 Submit Topology 权限只有在 Storm Topology 为*的情况下可以赋权生效。 • File Upload：文件上传。 • File Download：文件下载。 • Kill Topology：删除拓扑。 • Rebalance：Rebalance 操作权限。 • Activate：激活权限。 • Deactivate：去激活权限。 • Get Topology Conf：获取拓扑配置。 • Get Topology：获取拓扑。 • Get User Topology：获取用户拓扑。 • Get Topology Info：获取拓扑信息。 • Upload New Credential：上传新的凭证。 • Select/Deselect All：全选/取消全选。 如需添加多条权限控制规则，可单击  按钮添加。 如需当前条件中的用户或用户组管理本条策略，可勾选“Delegate Admin”，这些用户将成为受委托的管理员。被委托的管理员可以更新、删除本策略，它还可以基于原始策略创建子策略。
Deny Conditions	策略拒绝条件，配置本策略内拒绝的权限及例外，配置方法与“Allow Conditions”类似。

步骤 4 （可选）添加策略有效期。在页面右上角单击“Add Validity period”，设置“Start Time”和“End Time”，选择“Time Zone”。单击“Save”保存。如需添加多条策略有效期，可单击  按钮添加。如需删除策略有效期，可单击  按钮删除。

步骤 5 单击“Add”，在策略列表可查看策略的基本信息。等待策略生效后，验证相关权限是否正常。

如需禁用某条策略，可单击  按钮编辑策略，设置策略开关为“Disabled”。

如果不再使用策略，可单击  按钮删除策略。

----结束

23.15 Ranger 日志介绍

日志描述

日志存储路径：Ranger 相关日志的默认存储路径为“/var/log/Bigdata/ranger/*角色名*”

- RangerAdmin：“/var/log/Bigdata/ranger/rangeradmin”（运行日志）。
- TagSync：“/var/log/Bigdata/ranger/tagsync”（运行日志）。
- UserSync“/var/log/Bigdata/ranger/usersync”（运行日志）。

日志归档规则：Ranger 的日志启动了自动压缩归档功能，缺省情况下，当日志大小超过 20MB 的时，会自动压缩，压缩后的日志文件名规则为：“<原有日志名>-<yyyy-mm-dd_hh-mm-ss>.[编号].log.zip”，最多保留最近的 20 个压缩文件。

表23-21 HDFS 日志列表

日志类型	日志文件名	描述
RangerAdmin 运行日志	access_log.<*DATE*>.log	Tomcat 访问日志。
	catalina.out	Tomcat 服务运行日志。
	gc-worker.log	RangerAdmin 的 GC 日志。
	postinstallDetail.log	实例安装前启动后工作日志。
	prestartDetail.log	实例启动前准备工作日志。
	ranger-admin-<*hostname*>.log	RangerAdmin 运行日志。
	ranger_admin_sql-<*hostname*>.log	RangerAdmin 检索 DBService 的日志。
	startDetail.log	实例启动日志。
TagSync 运行日志	cleanupDetail.log	实例清理日志。
	gc-worker.log	实例 GC 日志。
	postinstallDetail.log	实例安装前启动后工作日志。

日志类型	日志文件名	描述
	prestartDetail.log	实例启动前准备工作日志。
	ranger-tagsync-<*hostname*>.log	TagSync 运行日志。
	startDetail.log	实例启动日志。
	tagsync.out	TagSync 的运行日志。
UserSync 运行日志	auth.log	unixauth 服务运行日志。
	cleanupDetail.log	实例清理日志。
	gc-worker.log	实例 GC 日志。
	postinstallDetail.log	实例安装前启动后工作日志。
	prestartDetail.log	实例启动前准备工作日志。
	ranger-usersync-<*hostname*>.log	USerSync 运行日志。
	startDetail.log	实例启动日志。

日志级别

HDFS 中提供了如表 23-22 所示的日志级别，日志级别优先级从高到低分别是 FATAL、ERROR、WARN、INFO、DEBUG，程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。

表23-22 日志级别

级别	描述
FATAL	FATAL 表示当前事件处理出现严重错误信息，可能导致系统崩溃。
ERROR	ERROR 表示当前事件处理出现错误信息，系统运行出错。
WARN	WARN 表示当前事件处理存在异常信息，但认为是正常范围，不会导致系统出错。
INFO	INFO 表示记录系统及各事件正常运行状态信息
DEBUG	DEBUG 表示记录系统及系统的调试信息。

如果您需要修改日志级别，请执行如下操作：

步骤 1 登录 FusionInsight Manager。

步骤 2 选择“集群 > 服务 > Ranger > 配置”。

步骤 3 选择“全部配置”。

步骤 4 左边菜单栏中选择所需修改的角色所对应的日志菜单。

步骤 5 选择所需修改的日志级别。

步骤 6 单击“保存”，在弹出窗口中单击“确定”使配置生效。

说明

配置完成后立即生效，不需要重启服务。

----结束

日志格式

Ranger 的日志格式如下所示：

表23-23 日志格式

日志类型	格式	示例
运行日志	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<产生该日志的线程名字>|<log 中的 message>|<日志事件的发生位置>	2020-04-29 20:09:28,543 | INFO | http-bio-21401-exec-56 | Request comes from API call, skip cas filter. | CasAuthenticationFilterWrapper.java:25

23.16 Ranger 常见问题

23.16.1 安装集群过程中，Ranger 启动失败

问题

安装集群过程中，Ranger 启动失败，Manager 进程任务列表里打印“ERROR：cannot drop sequence X_POLICY_REF_ACCESS_TYPE_SEQ”等关于数据库信息，如何解决并正常安装 Ranger？

回答

该现象可能出现在安装两个 RangerAmdin 实例的场景下，安装失败后，请先手动重启一个 RangerAdmin，然后再逐步重启其他实例。

23.16.2 如何判断某个服务是否使用了 Ranger 鉴权

问题

如何判断某个支持使用 Ranger 鉴权的服务当前是否启用了 Ranger 鉴权？

回答

登录 FusionInsight Manager，选择“集群 > 服务 > *服务名称*”，在服务详情页上继续单击“更多”，查看“启用 Ranger 鉴权”是否为可点击？

- 是，表示当前本服务未启用 Ranger 鉴权插件，可单击“启用 Ranger 鉴权”启用该功能。
- 否，表示当前本服务已启用 Ranger 鉴权插件，可通过 Ranger 管理界面配置访问该服务资源的权限策略。

23.16.3 新创建用户修改完密码后无法登录 Ranger

问题

使用新建用户登录 Ranger 页面，为什么在修改完密码后登录报 401 错误？

回答

由于 UserSync 同步用户数据有时间周期，默认是 5 分钟，因此在 Manager 上新创建的用户在用户同步成功前无法登录 Ranger，因为 Ranger 的 DB 里暂时还没有该用户信息，需要等待同步周期所设置的时间后再尝试登录。

非安全模式下，由于 Ranger 并不从 Manager 同步用户数据，因此，仅有 admin 用户可以登录 Ranger，暂时不支持其他用户登录。

23.16.4 Ranger 界面添加或者修改 HBase 策略时，无法使用通配符搜索已存在的 HBase 表

问题

添加 HBase 的 Ranger 访问权限策略时，在策略中使用通配符搜索已存在的 HBase 表时，搜索不到已存在的表，并且在/var/log/Bigdata/ranger/rangeradmin/ranger-admin-*log 中报以下错误

```
Caused by: javax.security.sasl.SaslException: No common protection layer between
client and server
at
com.sun.security.sasl.gsskerb.GssKrb5Client.doFinalHandshake(GssKrb5Client.java:253)
at
com.sun.security.sasl.gsskerb.GssKrb5Client.evaluateChallenge(GssKrb5Client.java:18
6)
at
org.apache.hadoop.hbase.security.AbstractHBaseSaslRpcClient.evaluateChallenge(Abstr
actHBaseSaslRpcClient.java:142)
```

```
at
org.apache.hadoop.hbase.security.NettyHBaseSaslRpcClientHandler$2.run(NettyHBaseSas
lRpcClientHandler.java:142)
at
org.apache.hadoop.hbase.security.NettyHBaseSaslRpcClientHandler$2.run(NettyHBaseSas
lRpcClientHandler.java:138)
at java.security.AccessController.doPrivileged(Native Method)
at javax.security.auth.Subject.doAs(Subject.java:422)
at
org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1761)
at
org.apache.hadoop.hbase.security.NettyHBaseSaslRpcClientHandler.channelRead0(NettyH
BaseSaslRpcClientHandler.java:138)
at
org.apache.hadoop.hbase.security.NettyHBaseSaslRpcClientHandler.channelRead0(NettyH
BaseSaslRpcClientHandler.java:42)
at
org.apache.hbase.thirdparty.io.netty.channel.SimpleChannelInboundHandler.channelRea
d(SimpleChannelInboundHandler.java:105)
at
org.apache.hbase.thirdparty.io.netty.channel.AbstractChannelHandlerContext.invokeCh
annelRead(AbstractChannelHandlerContext.java:362)
```

回答

Ranger 界面上 HBase 服务插件的“hbase.rpc.protection”参数值和 HBase 服务端的“hbase.rpc.protection”参数值必须保持一致。

步骤 1 参考登录 Ranger 管理界面章节，登录 Ranger 管理界面。

步骤 2 在首页中“HBASE”区域，单击组件插件名称，如 HBase 的  按钮

步骤 3 搜索配置项“hbase.rpc.protection”，修改配置项的 value 值，与 HBase 服务端的“hbase.rpc.protection”的值保持一致。

步骤 4 单击“保存”。

----结束

24 使用 Spark

24.1 使用前须知

本章节适用于 MRS 3.x 之前版本。

24.2 从零开始使用 Spark

本章节提供从零开始使用 Spark 提交 sparkPi 作业的操作指导，sparkPi 是最经典的 Spark 作业，它用来计算 Pi（π）值。

操作步骤

步骤 1 准备 sparkPi 程序。

开源的 Spark 的样例程序包含多个例子，其中包含 sparkPi。可以从 https://d3kbcqa49mib13.cloudfront.net/spark-2.1.0-bin-hadoop2.7.tgz 中下载 Spark 的样例程序。

解压后在“spark-2.1.0-bin-hadoop2.7/examples/jars”路径下获取“spark-examples_2.11-2.1.0.jar”，即为 Spark 的样例程序。spark-examples_2.11-2.1.0.jar 样例程序包含 sparkPi 程序。

步骤 2 上传数据至 OBS。

1. 登录 OBS 控制台。
2. 单击“并行文件系统 > 创建并行文件系统”，创建一个名称为 sparkpi 的文件系统。

   sparkpi 仅为示例，文件系统名称必须全局唯一，否则会创建并行文件系统失败。其他参数分别保持默认值。
3. 单击 sparkpi 文件系统名称，并选择“文件”。
4. 单击“新建文件夹”，分别创建 program 文件夹。
5. 进入 program 文件夹，单击上传文件，从本地选择步骤 1 中下载的程序包，“存储类别”选择“标准存储”。

步骤 3 登录 MRS 控制台，在左侧导航栏选择“集群列表 > 现有集群”，单击集群名称。

步骤 4 提交 sparkPi 作业。

在 MRS 控制台选择“作业管理”，单击“添加”，进入“添加作业”页面，具体请参见“用户指南 > 管理现有集群 > 作业管理 > 运行 Spark 作业”章节。

- 作业类型选择“SparkSubmit”。
- 作业名称为“sparkPi”。
- 执行程序路径配置为 OBS 上存放程序的地址。例如：obs://sparkpi/program/spark-examples_2.11-2.1.0.jar。
- 运行程序参数选择“--class”，值填写“org.apache.spark.examples.SparkPi”。
- 执行程序参数中填写的参数为： 10。
- 服务配置参数无需填写。

只有集群处于“运行中”状态时才能提交作业。

作业提交成功后默认为“已接受”状态，不需要用户手动执行作业。

步骤 5 查看作业执行结果。

1. 进入“作业管理”页面，查看作业是否执行完成。

   作业运行需要时间，作业运行结束后，刷新作业列表。

   作业执行成功或失败后都不能再次执行，只能新增作业，配置作业参数后重新提交作业。

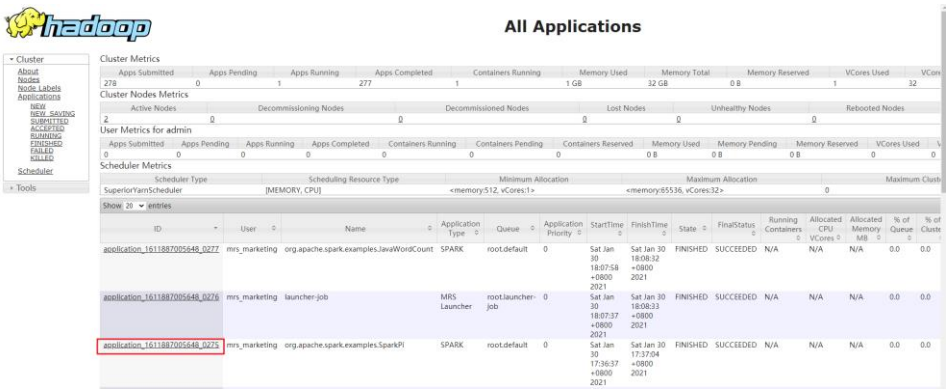
2. 进入 Yarn 原生界面，查看作业输出信息。

   a. 进入“作业管理”页面，单击对应作业所在行“操作”列的“查看详情”，获取“作业实际编号”。

   b. 登录 Manager 页面，选择“服务管理 > Yarn > ResourceManager WebUI > ResourceManager (主)”进入 Yarn 界面。

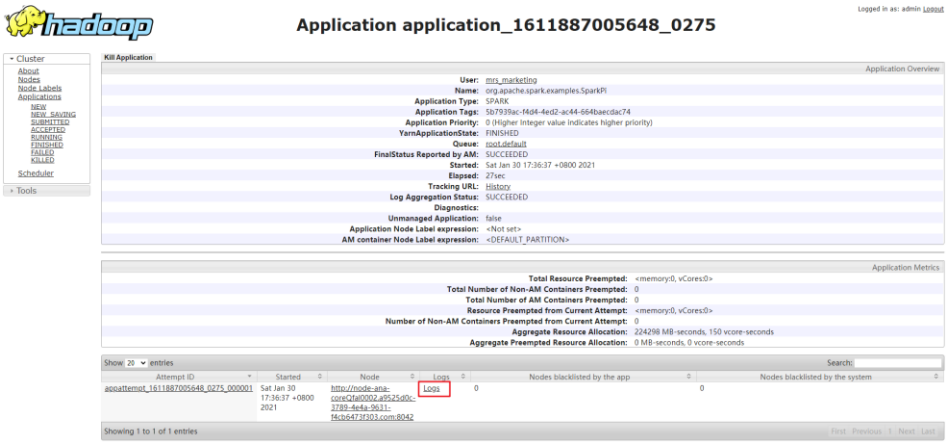
   c. 单击“作业实际编号”对应 ID。

   图24-1 Yarn 界面



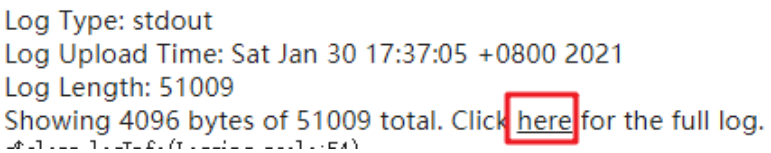
   d. 单击作业日志中的“Logs”。

图24-2 sparkPi 作业日志



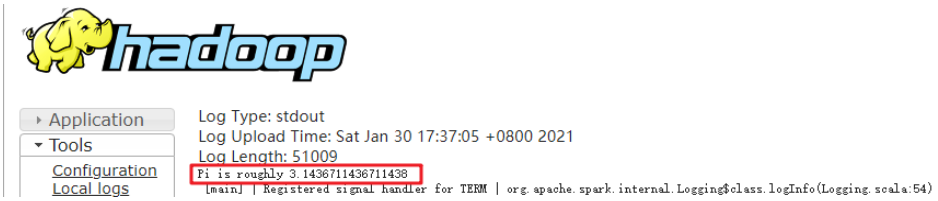
e. 单击“here”获取更详细日志。

图24-3 sparkPi 作业更详细日志



f. 获取作业执行结果。

图24-4 sparkPi 作业执行结果



----结束

24.3 从零开始使用 Spark SQL

Spark 提供类似 SQL 的 Spark SQL 语言操作结构化数据，本章节提供从零开始使用 Spark SQL，创建一个名称为 src_data 的表，然后在 src_data 表中每行写入一条数据，最后将数据存储在“mrs_20160907”集群中。再使用 SQL 语句查询 src_data 表中的数据，最后可将 src_data 表删除。

前提条件

将 OBS 数据源中的数据写入 Spark SQL 表中时，需要先获取 AK/SK。获取方法如下：

1. 登录管理控制台。
2. 单击用户名，在下拉列表中单击“我的凭证”。
3. 单击“访问密钥”。
4. 单击“新增访问密钥”，进入“新增访问密钥”页面。
5. 输入登录密码和短信验证码，单击“确定”，下载密钥，请妥善保。

操作步骤

步骤 1 准备使用 Spark SQL 分析的数据源。

样例 txt 文件如下：

```
abcd3ghji
efgh658ko
1234jjyu9
7h8kodfg1
kk99icxz3
```

步骤 2 上传数据至 OBS。

1. 登录 OBS 控制台。
2. 单击“并行文件系统 > 创建并行文件系统”，创建一个名称为 sparksql 的文件系统。

   sparksql 仅为示例，文件系统名称必须全局唯一，否则会创建并行文件系统失败。
3. 单击 sparksql 文件系统名称，并选择“文件”。
4. 单击“新建文件夹”，创建 input 文件夹。
5. 进入 input 文件夹，单击“上传文件 > 添加文件”，选择本地的 txt 文件，然后单击“上传”。

步骤 3 登录 MRS 控制台，在左侧导航栏选择“集群列表 > 现有集群”，单击集群名称。

步骤 4 将 OBS 中的 txt 文件导入至 HDFS 中。

1. 选择“文件管理”。
2. 在“HDFS 文件列表”页签中单击“新建”，创建一个名称为 userinput 的文件夹。
3. 进入 userinput 文件夹，单击“导入数据”。
4. 选择 OBS 和 HDFS 路径，单击“确定”。

   OBS 路径：obs://sparksql/input/sparksql-test.txt

   HDFS 路径：/user/userinput

步骤 5 提交 Spark SQL 语句。

1. 在 MRS 控制台选择“作业管理”，具体请参见“用户指南 > 管理现有集群 > 作业管理 > 运行 Spark 作业”章节。

只有“mrs_20160907”集群处于“运行中”状态时才能提交 Spark SQL 语句。

2. 输入创建表的 Spark SQL 语句。

   输入 Spark SQL 语句时，总字符数应当小于或等于 10000 字符，否则会提交语句失败。

   语法格式：

   CREATE [EXTERNAL] **TABLE** [IF NOT EXISTS] *table_name* [(col_name data_type [COMMENT col_comment], ...)] [COMMENT table_comment] [PARTITIONED **BY** (col_name data_type [COMMENT col_comment], ...)] [CLUSTERED **BY** (col_name, col_name, ...) [SORTED **BY** (col_name [ASC|DESC], ...)] INTO num_buckets BUCKETS] [ROW FORMAT row_format] [STORED **AS** file_format] [LOCATION hdfs_path];

   创建表样例存在以下两种方式。

   - 方式一：创建一个 src_data 表，将数据源中的数据一行一行写入 src_data 表中。
     - 数据源存储在 HDFS 的“/user/omm/userinput”文件夹下：***create external table*** *src_data****(line string) row format delimited fields terminated by*** *'\\n'* ***stored as textfile location*** *'/user/omm/userinput'*;
     - 数据源存储在 OBS 的“/sparksql/input”文件夹下：**create external table** *src_data***(line string) row format delimited fields terminated by '\\n' stored as textfile location** '*obs://AK:SK@sparksql/input*';

       AK/SK 获取方法，请参见前提条件。
   - 方式二：创建一个表 src_data1，将数据源中的数据批量 load 到 src_data1 表中。

     create table *src_data1* **(line string) row format delimited fields terminated by** ',' ;

     load data inpath '*/user/omm/userinput/sparksql-test.txt*' **into table** *src_data1*;

   说明

   采用方式二时，只能将 HDFS 上的数据 load 到新建的表中，OBS 上的数据不支持直接 load 到新建的表中。

3. 输入查询表的 Spark SQL 语句。

   语法格式：

   SELECT col_name **FROM** *table_name*;

   查询表样例，查询 src_data 表中的所有数据：

   select * from src_data;

4. 输入删除表的 Spark SQL 语句。

   语法格式：

   DROP TABLE [IF EXISTS] *table_name*;

   删除表样例：

   drop table src_data;

5. 单击“检查”，检查输入语句的语法是否正确。
6. 单击“确定”。

   Spark SQL 语句提交后，是否执行成功会在“执行结果”列中展示。

步骤 6 删除集群。

----结束

24.4 使用 Spark 客户端

MRS 集群创建完成后，可以通过客户端去创建和提交作业。客户端可以安装在集群内部节点或集群外部节点上：

- 集群内部节点：MRS 集群创建完成后，集群内的 master 和 core 节点默认已经安装好客户端，详情请参见“用户指南 > 连接集群 > 使用 MRS 客户端 > 集群内节点使用 MRS 客户端”章节，登录安装客户端的节点。
- 集群外部节点：用户可以将客户端安装在集群外部节点上，详情请参见“用户指南 > 连接集群 > 使用 MRS 客户端 > 集群外节点使用 MRS 客户端”章节，登录安装客户端的节点。

使用 Spark 客户端

步骤 1 请根据客户端所在位置，参考“用户指南 > 连接集群 > 使用 MRS 客户端 > 集群内节点使用 MRS 客户端”，或者“用户指南 > 连接集群 > 使用 MRS 客户端 > 集群外节点使用 MRS 客户端”章节，登录安装客户端的节点。

步骤 2 执行以下命令，切换到客户端安装目录。

cd /opt/client

步骤 3 执行以下命令配置环境变量。

source bigdata_env

步骤 4 如果集群为安全模式，执行以下命令进行用户认证。普通模式集群无需执行用户认证。

kinit *组件业务用户*

步骤 5 直接执行 Spark Shell 命令。例如：

spark-beeline

----结束

24.5 访问 Spark Web UI 界面

Spark Web UI 界面主要用于查看 Spark 应用程序运行情况，推荐使用 Google chrome 浏览器以获得更好的体验。

Spark 主要有两个 Web 页面。

- Spark UI 页面，用于展示正在执行的应用的运行情况。

  页面主要包括了 Jobs、Stages、Storage、Environment、Executors、SQL、JDBC/ODBC Server 等部分。Streaming 应用会多一个 Streaming 标签页。

- History Server 页面，用于展示已经完成的和未完成的 Spark 应用的运行情况。

  页面包括了应用 ID、应用名称、开始时间、结束时间、执行时间、所属用户等信息。

Spark UI

步骤 1 进入组件管理页面：

- MRS 3.x 之前版本，单击集群名称，登录集群详情页面，选择“组件管理”。

  说明

  若集群详情页面没有“组件管理”页签，请先完成 IAM 用户同步（在集群详情页的“概览”页签，单击“IAM 用户同步”右侧的“同步”进行 IAM 用户同步）。

- MRS 3.x 及后续版本，登录 FusionInsight Manager，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本）。然后选择“集群 > *待操作的集群名称* > 服务 ”。

步骤 2 选择“Yarn”并在“Yarn 概述”中“ResourceManager Web UI”中单击“ResourceManager Web UI”对应的“ResourceManager”进入 Web 界面。

步骤 3 查找到对应的 Spark 应用程序，单击应用信息的最后一列“ApplicationMaster”，即可进入 Spark UI 页面。

图24-5 ApplicationMaster

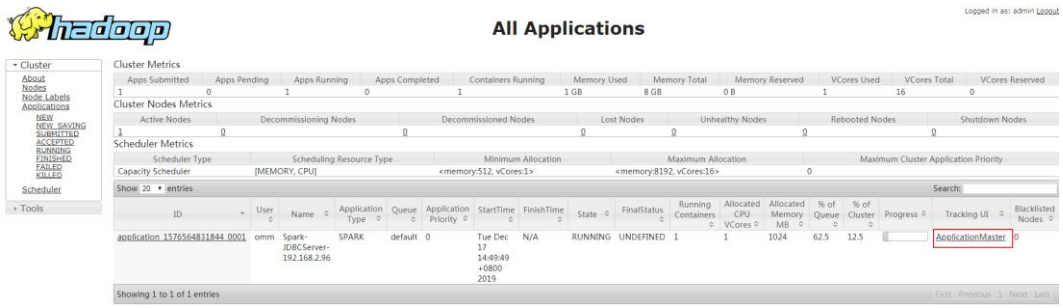


图24-6 Spark UI 页面

----结束

History Server

步骤 1 进入组件管理页面：

- MRS 3.x 之前版本，单击集群名称，登录集群详情页面，选择“组件管理”。

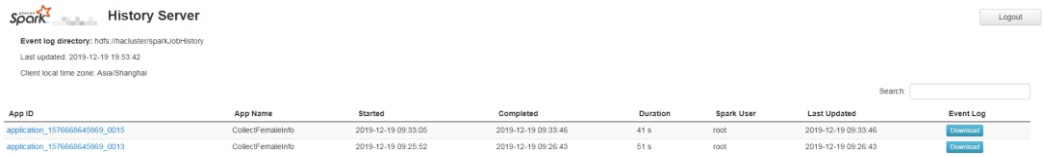
说明

若集群详情页面没有“组件管理”页签，请先完成 IAM 用户同步（在集群详情页的“概览”页签，单击“IAM 用户同步”右侧的“同步”进行 IAM 用户同步）。

- MRS 3.x 及后续版本，登录 FusionInsight Manager，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本）。然后选择“集群 > *待操作的集群名*称 > 服务 ”。

步骤 2 选择“Spark”并在“Spark 概述”中“Spark Web UI”中单击“Spark Web UI”对应的“JobHistory”进入 Web 界面。

图24-7 Spark History Server



----结束

24.6 Spark 对接 OpenTSDB

24.6.1 创建表关联 OpenTSDB

功能描述

MRS 的 Spark 实现了访问 OpenTSDB 的 Datasource，能够在 Spark 中创建关联表，查询和插入 OpenTSDB 数据。

使用 CREATE TABLE 命令创建表并关联 OpenTSDB 上已有的 metric。

说明

若 OpenTSDB 上不存在 metric，查询对应的表会报错。

语法格式

```
CREATE TABLE [IF NOT EXISTS] OPENTSDB_TABLE_NAME  USING OPENTSDB OPTIONS (
'metric' = 'METRIC_NAME',
'tags' = 'TAG1,TAG2'
);
```

关键字

参数	描述
metric	所创建的表对应的 OpenTSDB 中的指标名称。

参数	描述
tags	metric 对应的标签，用于归类、过滤、快速检索等操作。可以是 1 个到 8 个，以“,”分隔，包括对应 metric 下所有 tagk 的值。

注意事项

创建表时，不需要指定 timestamp 和 value 字段，系统会根据指定的 tags 自动构建字段，包含以下字段，其中 TAG1 和 TAG2 由 tags 指定。

- TAG1 String
- TAG2 String
- timestamp Timestamp
- value double

示例

创建 opentsdb_table 表并关联到 OpenTSDB 组件的 city.temp 这个 metric。

```
CREATE table opentsdb_table using opentsdb OPTIONS ('metric'='city.temp',
'tags'='city,location');
```

24.6.2 插入数据至 OpenTSDB 表

功能描述

使用 INSERT INTO 命令将表中的数据插入到已关联的 OpenTSDB metric 中。

语法格式

```
INSERT INTO TABLE_NAME SELECT * FROM SRC_TABLE;
INSERT INTO TABLE_NAME VALUES(XXX);
```

关键字

参数	描述
TABLE_NAME	所关联的 OpenTSDB 表名。
SRC_TABLE	获取数据的表名，普通表即可。

注意事项

- 插入的数据不能为 null；插入的数据相同，会覆盖原数据；插入的数据只有 value 值不同，也会覆盖原数据。
- 不支持 INSERT OVERWRITE 语法。

- 不建议对同一张表并发插入数据，因为有一定概率发生并发冲突，导致插入失败。
- 时间戳格式只支持 yyyy-MM-dd hh:mm:ss。

示例

在 opentsdb_table 表中插入数据。

```
insert into opentsdb_table values('shenzhen','futian','2018-05-03 00:00:00',21);
```

24.6.3 查询 OpenTSDB 表

SELECT 命令用于查询 OpenTSDB 表中的数据。

语法格式

```
SELECT * FROM table_name WHERE tagk=tagv LIMIT number;
```

关键字

参数	描述
LIMIT	对查询结果进行限制。
number	参数仅支持 INT 类型。

注意事项

- 所查询的表必须是已经存在的表，否则会出错。
- 查询的 tagv 必须是已经有的值，否则会出错。

示例

查询表 opentsdb_table 中的数据。

```
SELECT * FROM opentsdb_table LIMIT 100;
SELECT * FROM opentsdb_table WHERE city='shenzhen';
```

24.6.4 默认配置修改

默认会连接 Spark 的 Executor 所在节点本地的 TSD 进程，在 MRS 中一般使用默认配置即可，无需修改。

表24-1 OpenTSDB 数据源相关配置

配置名	描述	样例值
spark.sql.datasource.opentsdb.host	连接的 TSD 进程地址	空（默认值） xx.xx.xx.xx，多个地址间用英文逗号间

		隔。
spark.sql.datasource.opentsdb.port	TSD 进程端口号	4242（默认值）
spark.sql.datasource.opentsdb.randomSeed	当 spark.sql.datasource.opentsdb.host 配置多个地址时，是否使用随机种子。配置为否时，所有在相同节点的 executor 会连接相同的 host，这样可以配合 spark.blacklist.enabled=true 来实现 Task 容错。	false（默认）

示例

在 spark-sql，spark-beeline 执行 set 语句后，再执行其他 SQL：

```
set spark.sql.datasource.opentsdb.host = 192.168.2.143,192.168.2.158;
SELECT * FROM opentsdb_table ;
```

25 使用 Spark2x

25.1 使用前须知

本章节适用于 MRS 3.x 及后续版本。

25.2 基本操作

25.2.1 快速入门

本章节提供从零开始使用 Spark2x 提交 spark 应用程序，包括 Spark Core 及 Spark SQL。其中，Spark Core 为 Spark 的内核模块，主要负责任务的执行，用于编写 spark 应用程序；Spark SQL 为执行 SQL 的模块。

场景说明

假定用户有某个周末网民网购停留时间的日志文本，基于某些业务要求，要求开发 Spark 应用程序实现如下要求：

- 统计日志文件中本周末网购停留总时间超过 2 个小时的女性网民信息。
- 周末两天的日志文件第一列为姓名，第二列为性别，第三列为本次停留时间，单位为分钟，分隔符为“,”。

log1.txt：周六网民停留日志

```
LiuYang,female,20
YuanJing,male,10
GuoYijun,male,5
CaiXuyu,female,50
Liyuan,male,20
FangBo,female,50
LiuYang,female,20
YuanJing,male,10
GuoYijun,male,50
CaiXuyu,female,50
FangBo,female,60
```

log2.txt：周日网民停留日志

```
LiuYang,female,20
YuanJing,male,10
CaiXuyu,female,50
FangBo,female,50
GuoYijun,male,5
CaiXuyu,female,50
Liyuan,male,20
CaiXuyu,female,50
FangBo,female,50
LiuYang,female,20
YuanJing,male,10
FangBo,female,50
GuoYijun,male,50
CaiXuyu,female,50
FangBo,female,60
```

前提条件

- 在 Manager 界面创建用户并开通其 HDFS、YARN、Kafka 和 Hive 权限。
- 根据所用的开发语言安装并配置 IntelliJ IDEA 及 JDK 等工具。
- 已完成 Spark2x 客户端的安装及客户端网络连接的配置。
- 对于 Spark SQL 程序，需要先在客户端启动 Spark SQL 或 Beeline 以输入 SQL 语句。

操作步骤

步骤 1 获取样例工程并将其导入 IDEA，导入样例工程依赖 jar 包。通过 IDEA 配置并生成 jar 包。

步骤 2 准备样例工程所需数据。

将场景说明中的原日志文件放置在 HDFS 系统中。

1. 本地新建两个文本文件，分别将 log1.txt 及 log2.txt 中的内容复制保存到 input_data1.txt 和 input_data2.txt。
2. 在 HDFS 上建立一个文件夹“/tmp/input”，并上传 input_data1.txt、input_data2.txt 到此目录。

步骤 3 将生成的 jar 包上传至 Spark2x 运行环境下（Spark2x 客户端），如“/opt/female”。

步骤 4 进入客户端目录，执行以下命令加载环境变量并登录。若安装了 Spark2x 多实例或者同时安装了 Spark 和 Spark2x，在使用客户端连接具体实例时，请执行以下命令加载具体实例的环境变量。

source bigdata_env

source Spark2x/component_env

kinit <用于认证的业务用户>

步骤 5 在 bin 目录下调用以下脚本提交 Spark 应用程序。

spark-submit --class *com.xxx.bigdata.spark.examples.FemaleInfoCollection* ***--master yarn-client*** */opt/female/FemaleInfoCollection.jar <inputPath>*

说明

- FemaleInfoCollection.jar 为步骤 1 生成的 jar 包。
- <inputPath>是步骤 2.2 创建的目录。

步骤 6 （可选）在 bin 目录下调用 **spark-sql** 或 **spark-beeline** 脚本后便可直接输入 SQL 语句执行查询等操作。

如创建一个表，插入一条数据再对表进行查询。

```
spark-sql> CREATE TABLE TEST(NAME STRING, AGE INT);
Time taken: 0.348 seconds
spark-sql>INSERT INTO TEST VALUES('Jack', 20);
Time taken: 1.13 seconds
spark-sql> SELECT * FROM TEST;
Jack     20
Time taken: 0.18 seconds, Fetched 1 row(s)
```

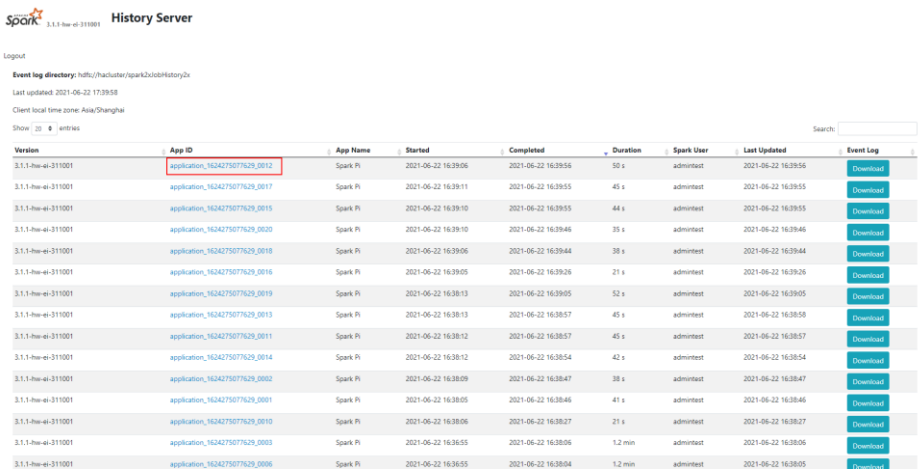
步骤 7 查看 Spark 应用运行结果。

- 通过指定文件查看运行结果数据。

  结果数据的存储路径和格式由 Spark 应用程序指定。
- 通过 Web 页面查看运行情况。
  a. 登入 Manager 主页面。在服务中选择 Spark2x。
  b. 进入 Spark2x 概览页面，单击 SparkWebUI 任意一个实例，如 JobHistory2x(host2)。
  c. 进入 History Server 页面。

     History Server 页面用于展示已完成和未完成的应用的运行情况。

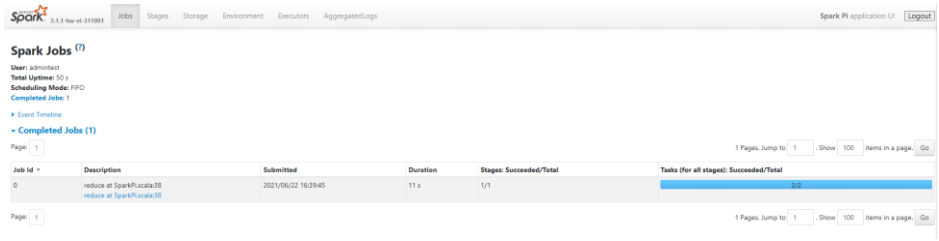
     图25-1 History Server 页面



  d. 选择一个应用 ID，单击此页面将跳转到该应用的 Spark UI 页面。

Spark UI 页面，用于展示正在执行的应用的运行情况。

图25-2 Spark UI 页面



- 通过查看 Spark 日志获取应用运行情况。
  通过查看 Spark2x 日志介绍了解应用运行情况，并根据日志信息调整应用程序。

----结束

25.2.2 快速配置参数

概述

本节介绍 Spark2x 使用过程中快速配置常用参数和不建议修改的配置参数。

快速配置常用参数

其他参数在安装集群时已进行了适配，以下参数需要根据使用场景进行调整。以下参数除特别指出外，一般在 Spark2x 客户端的“spark-defaults.conf”文件中配置。

表25-1 快速配置常用参数

配置项	说明	默认值
spark.sql.parquet.compression.codec	对于非分区 parquet 表，设置其存储文件的压缩格式。 在 JDBCServer 服务端的“spark-defaults.conf”配置文件中进行设置。	snappy
spark.dynamicAllocation.enabled	是否使用动态资源调度，用于根据规模调整注册于该应用的 executor 的数量。目前仅在 YARN 模式下有效。 JDBCServer 默认值为 true，client 默认值为 false。	false
spark.executor.memory	每个 Executor 进程使用的内存数量，与 JVM 内存设置字符串的格式相同（例如：512m，2g）。	4G
spark.sql.autoBroadcastJoinThreshold	当进行 join 操作时，配置广播的最大值。	10485760

配置项	说明	默认值
	• 当 SQL 语句中涉及的表中相应字段的大小小于该值时，进行广播。 • 配置为-1 时，将不进行广播。	
spark.yarn.queue	JDBCServer 服务所在的 Yarn 队列。 在 JDBCServer 服务端的“spark-defaults.conf”配置文件中进行设置。	default
spark.driver.memory	大集群下推荐配置 32~64g 驱动程序进程使用的内存数量，即 SparkContext 初始化的进程（例如：512m, 2g）。	4G
spark.yarn.security.credentials.hbase.enabled	是否打开获取 HBase token 的功能。如果需要 Spark-on-HBase 功能，并且配置了安全集群，参数值设置为“true”。否则设置为“false”。	false
spark.serializer	用于串行化将通过网络发送或需要缓存的对象的类以序列化形式展现。 Java 序列化的默认值适用于任何 Serializable Java 对象，但运行速度相当慢，所以建议使用 org.apache.spark.serializer.KryoSerializer 并配置 Kryo 序列化。可以是 org.apache.spark.serializer.Serializer 的任何子类。	org.apache.spark.serializer.JavaSerializer
spark.executor.cores	每个执行者使用的内核个数。 在独立模式和 Mesos 粗粒度模式下设置此参数。当有足够多的内核时，允许应用程序在同样的 worker 上执行多个执行程序；否则，在每个 worker 上，每个应用程序只能运行一个执行程序。	1
spark.shuffle.service.enabled	NodeManager 中一个长期运行的辅助服务，用于提升 Shuffle 计算性能。	fasle
spark.sql.adaptive.enabled	是否开启自适应执行框架。	false
spark.executor.memoryOverhead	每个执行器要分配的堆内存量（单位为兆字节）。 这是占用虚拟机开销的内存，类似于内部字符串，其他内置开销等等。会随着执行器大小（通常为 6-10%）而增长。	1GB
spark.streaming.kafka.direct.lifo	配置是否开启 Kafka 后进先出功能。	false

不建议修改的参数

以下参数在安装集群时已进行了适配，不建议用户进行修改。

表25-2 不建议修改的参数说明

配置项	说明	默认值或配置示例
spark.password.factory	用于选择密钥解析方式。	org.apache.spark.om.util.FIPasswordFactory
spark.ssl.ui.protocol	配置 ui 的 ssl 协议。	TLSv1.2
spark.yarn.archive	Spark jars 的存档，用于分发到 YARN 缓存。如果设置，此配置值将替换<code> spark.yarn.jars </code>，并存档在所有应用程序的容器中使用。存档应包含其根目录中的 jar 文件。与以前的选项一样，存档也可以在 HDFS 上托管，用来加快文件分发速度。	hdfs://hacluster/user/spark2x/jars/8.1.0.1/spark-archive-2x.zip 说明 此处版本号 8.1.0.1 为示例，具体以实际环境的版本号为准。
spark.yarn.am.extraJavaOptions	在 Client 模式下传递至 YARN Application Master 的一系列额外 JVM 选项。在 Cluster 模式下使用“spark.driver.extraJavaOptions”。	-Dlog4j.configuration=./__spark_conf__/__hadoop_conf__/log4j-executor.properties -Djava.security.auth.login.config=./__spark_conf__/__hadoop_conf__/jaas-zk.conf -Dzookeeper.server.principal=zookeeper/hadoop.*<系统域名>* -Djava.security.krb5.conf=./__spark_conf__/__hadoop_conf__/kdc.conf -Djdk.tls.ephemeralDHKeySize=2048
spark.shuffle.servicev2.port	Shuffle 服务监听数据获取请求的端口。	27338
spark.ssl.historyServer.enabled	配置 history server 是否使用 SSL。	true
spark.files.overwrite	当目标文件存在时，且其内容与源的文件不匹配。是否覆盖通过 SparkContext.addFile()添加的文件。	false
spark.yarn.clus	YARN-Cluster 模式	${BIGDATA_HOME}/common/runtime/securit

配置项	说明	默认值或配置示例
ter.driver.extraClassPath	下，Driver 使用的 extraClassPath，配置为服务端的路径和参数。	y
spark.driver.extraClassPath	附加至 driver 的 classpath 的额外 classpath 条目。	${BIGDATA_HOME}/common/runtime/security
spark.yarn.dist.innerfiles	配置 YARN 模式下 Spark 内部需要上传到 HDFS 的文件。	/Spark_path/spark/conf/s3p.file,/Spark_path/spark/conf/locals3.jceks *Spark_path* 为 Spark 客户端的安装路径。
spark.sql.bigdata.register.dialect	用于注册 sql 解析器。	org.apache.spark.sql.hbase.HBaseSQLParser
spark.shuffle.manager	处理数据的方式。有两种实现方式可用：sort 和 hash。sort shuffle 对内存的使用率更高，是 Spark 1.2 及后续版本的默认选项。	SORT
spark.deploy.zookeeper.url	Zookeeper 的地址，多个地址以逗号隔开。	For example: host1:2181,host2:2181,host3:2181
spark.broadcast.factory	使用的广播方式。	org.apache.spark.broadcast.TorrentBroadcastFactory
spark.sql.session.state.builder	指定会话状态构造器。	org.apache.spark.sql.hive.FIHiveACLSessionStateBuilder
spark.executor.extraLibraryPath	设置启动 executor JVM 时所使用的特殊的 library path。	${BIGDATA_HOME}/FusionInsight_HD_8.1.0.1/install/FusionInsight-Hadoop-3.1.1/hadoop/lib/native
spark.ui.customErrorPage	配置网页有错误时是否允许显示自定义的错误信息页面。	true
spark.httpdProxy.enable	配置是否使用 httpd 代理。	true
spark.ssl.ui.enabledAlgorithms	配置 ui ssl 算法。	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_D

配置项	说明	默认值或配置示例
		HE_RSA_WITH_AES_256_GCM_SHA384,TLS_DHE_DSS_WITH_AES_256_GCM_SHA384,TLS_DHE_RSA_WITH_AES_128_GCM_SHA256,TLS_DHE_DSS_WITH_AES_128_GCM_SHA256
spark.ui.logout.enabled	针对 Spark 组件的 WebUI，设置 logout 按钮。	true
spark.security.hideInfo.enabled	配置 UI 界面是否隐藏敏感信息。	true
spark.yarn.cluster.driver.extraLibraryPath	YARN-Cluster 模式下 driver 的 extraLibraryPath，配置成服务端的路径和参数。	${BIGDATA_HOME}/FusionInsight_HD_8.1.0.1/install/FusionInsight-Hadoop-3.1.1/hadoop/lib/native
spark.driver.extraLibraryPath	设置一个特殊的 library path 在启动驱动程序 JVM 时使用。	${DATA_NODE_INSTALL_HOME}/hadoop/lib/native
spark.ui.killEnabled	允许停止 Web UI 中的 stage 和相应的 job。	true
spark.yarn.access.hadoopFileSystems	Spark 可以访问多个 NameService。有多个 NameService 时，需要把所使用的 NameService 都配置进该配置项，之间以逗号分隔。	hdfs://hacluster,hdfs://hacluster
spark.yarn.cluster.driver.extraJavaOptions	传递至 Executor 的额外 JVM 选项。例如，GC 设置或其他日志记录。请注意不能通过此选项设置 Spark 属性或 heap 大小。Spark 属性应该使用 SparkConf 对象或调用 spark-submit 脚本时指定的 spark-defaults.conf 文件来设置。Heap 大小可以通过	-Xloggc:<LOG_DIR>/gc.log -XX:+PrintGCDetails -XX:-OmitStackTraceInFastThrow -XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=20 -XX:GCLogFileSize=10M -Dlog4j.configuration=./__spark_conf__/__hadoop_conf__/log4j-executor.properties -Djava.security.auth.login.config=./__spark_conf__/__hadoop_conf__/jaas-zk.conf -Dzookeeper.server.principal=zookeeper/hadoop.*<系统域名>* -Djava.security.krb5.conf=./__spark_conf__/__h

配置项	说明	默认值或配置示例
	spark.executor.memory 来设置。	adoop_conf__/kdc.conf -Djetty.version=x.y.z -Dorg.xerial.snappy.tempdir=${BIGDATA_HOME}/tmp/spark2x_app -Dcarbon.properties.filepath=./__spark_conf__/__hadoop_conf__/carbon.properties -Djdk.tls.ephemeralDHKeySize=2048
spark.driver.extraJavaOptions	传递至 driver（驱动程序）的一系列额外 JVM 选项。	-Xloggc:${SPARK_LOG_DIR}/indexserver-omm-%p-gc.log -XX:+PrintGCDetails -XX:-OmitStackTraceInFastThrow -XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps -XX:MaxDirectMemorySize=512M -XX:MaxMetaspaceSize=512M -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=20 -XX:GCLogFileSize=10M -XX:OnOutOfMemoryError='kill -9 %p' -Djetty.version=x.y.z -Dorg.xerial.snappy.tempdir=${BIGDATA_HOME}/tmp/spark2x/JDBCServer/snappy_tmp -Djava.io.tmpdir=${BIGDATA_HOME}/tmp/spark2x/JDBCServer/io_tmp -Dcarbon.properties.filepath=${SPARK_CONF_DIR}/carbon.properties -Djdk.tls.ephemeralDHKeySize=2048 -Dspark.ssl.keyStore=${SPARK_CONF_DIR}/child.keystore #{java_stack_prefer}
spark.eventLog.overwrite	是否覆盖任何现有的文件。	false
spark.eventLog.dir	如果 **spark.eventLog.enabled** 为 **true**，记录 Spark 事件的目录。在此目录下，Spark 为每个应用程序创建文件，并将应用程序的事件记录到文件中。用户也可设置为统一的与 HDFS 目录相似的地址，这样 History server 就可以读取历史文件。	hdfs://hacluster/spark2xJobHistory2x
spark.random.port.min	设置随机端口的最小值。	22600
spark.authenticate	是否 Spark 认证其内部连接。如果不是运行在 YARN 上，请	true

配置项	说明	默认值或配置示例
	参见 spark.authenticate.secret 的相关内容。	
spark.random.port.max	设置随机端口的最大值。	22899
spark.eventLog.enabled	是否记录 Spark 事件，用于应用程序在完成后重构 webUI。	true
spark.executor.extraJavaOptions	传递至 Executor 的额外 JVM 选项。例如，GC 设置或其他日志记录。请注意不能通过此选项设置 Spark 属性或 heap 大小。	-Xloggc:<LOG_DIR>/gc.log -XX:+PrintGCDetails -XX:-OmitStackTraceInFastThrow -XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=20 -XX:GCLogFileSize=10M -Dlog4j.configuration=./log4j-executor.properties -Djava.security.auth.login.config=./jaas-zk.conf -Dzookeeper.server.principal=zookeeper/hadoop.*<系统域名>* -Djava.security.krb5.conf=./kdc.conf -Dcarbon.properties.filepath=./carbon.properties -Xloggc:<LOG_DIR>/gc.log -XX:+PrintGCDetails -XX:-OmitStackTraceInFastThrow -XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=20 -XX:GCLogFileSize=10M -Dlog4j.configuration=./__spark_conf__/__hadoop_conf__/log4j-executor.properties -Djava.security.auth.login.config=./__spark_conf__/__hadoop_conf__/jaas-zk.conf -Dzookeeper.server.principal=zookeeper/hadoop.*<系统域名>* -Djava.security.krb5.conf=./__spark_conf__/__hadoop_conf__/kdc.conf -Dcarbon.properties.filepath=./__spark_conf__/__hadoop_conf__/carbon.properties -Djdk.tls.ephemeralDHKeySize=2048
spark.sql.authorization.enabled	配置 Hive client 是否开启认证。	true

25.2.3 常用参数

概述

本节介绍 Spark 使用过程中的常用配置项。以特性为基础划分子章节，以便用户快速搜索到相应的配置项。如果用户使用 MRS 集群，本节介绍的参数大部分已经适配好，用户无需再进行配置。少数需要用户根据实际场景配置的参数，请参见快速配置参数。

配置 Stage 失败重试次数

Spark 任务在遇到 FetchFailedException 时会触发 Stage 重试。为了防止 Stage 无限重试，对 Stage 重试次数进行限制。重试次数可以根据实际需要进行调整。

在 Spark 客户端的“spark-defaults.conf”文件中配置如下参数。

表25-3 参数说明

参数	说明	默认值
spark.stage.maxConsecutiveAttempts	Stage 失败重试最大次数。	4

配置是否使用笛卡尔积功能

要启动使用笛卡尔积功能，需要在 Spark 的“spark-defaults.conf”配置文件中进行如下设置。

表25-4 笛卡尔积参数说明

参数	说明	默认值
spark.sql.crossJoin.enabled	是否允许隐性执行笛卡尔积。 • “true”表示允许 • “false”表示不允许，此时只允许query 中显式包含 CROSS JOIN 语法。	true

说明

- JDBC 应用在服务端的“spark-defaults.conf”配置文件中设置该参数。
- Spark 客户端提交的任务在客户端配的“spark-defaults.conf”配置文件中设置该参数。

Spark 长时间任务安全认证配置

安全模式下，使用 Spark CLI（如 spark shell、spark sql、spark submit）时，如果使用 kinit 命令进行安全认证，当执行长时间运行任务时，会因为认证过期导致任务失败。

在客户端的“spark-defaults.conf”配置文件中设置如下参数，配置完成后，重新执行 Spark CLI 即可。

说明

当参数值为“true”时，需要保证“spark-defaults.conf”和“hive-site.xml”中的 Keytab 和 principal 的值相同。

表25-5 参数说明

参数名称	含义	默认值
spark.kerberos.principal	具有 Spark 操作权限的 principal。请联系管理员获取对应 principal。	-
spark.kerberos.keytab	具有 Spark 操作权限的 Keytab 文件名称和文件路径。请联系管理员获取对应 Keytab 文件。	-
spark.security.bigdata.loginOnce	Principal 用户是否只登录一次。true 为单次登录；false 为多次登录。 单次登录与多次登录的区别在于：Spark 社区使用多次 Kerberos 用户登录多次的方案，但容易出现 TGT 过期或者 Token 过期异常导致应用无法长时间运行。DataSight 修改了 Kerberos 登录方式，只允许用户登录一次，可以有效的解决过期问题。限制在于，Hive 相关的 principal 与 keytab 的配置项必须与 Spark 配置相同。 说明 当参数值为 true 时，需要保证“spark-defaults.conf”和“hive-site.xml”中的 Keytab 和 principal 的值相同。	true

Python Spark

Python Spark 是 Spark 除了 Scala、Java 两种 API 之外的第三种编程语言。不同于 Java 和 Scala 都是在 JVM 平台上运行，Python Spark 不仅会有 JVM 进程，还会有自身的 Python 进程。以下配置项只适用于 Python Spark 场景，而其他配置项也同样可以在 Python Spark 中生效。

表25-6 参数说明

参数	描述	默认值
spark.python.profile	在 Python worker 中开启 profiling。通过 sc.show_profiles()展示分析结果。或者在 driver 退出	false

参数	描述	默认值
	前展示分析结果。可以通过 sc.dump_profiles(path) 将结果转储到磁盘中。如果一些分析结果已经手动展示，那么在 Driver 退出前，它们将不会再自动展示。 默认使用 pyspark.profiler.BasicProfiler，可以在初始化 SparkContext 时传入指定的 profiler 来覆盖默认的 profiler。	
spark.python.worker.memory	聚合过程中每个 python worker 进程所能使用的内存大小，其值格式同指定 JVM 内存一致，如 512m，2g。如果进程在聚集期间所用的内存超过了该值，数据将会被写入磁盘。	512m
spark.python.worker.reuse	是否重用 python worker。如是，它将使用固定数量的 Python workers，那么下一批提交的 task 将重用这些 Python workers，而不是为每个 task 重新 fork 一个 Python 进程。 该功能在大型广播下非常有用，因为此时对下一批提交的 task 不需要将数据从 JVM 再一次传输至 Python worker。	true

Dynamic Allocation

动态资源调度是 On Yarn 模式特有的特性，并且必须开启 Yarn External Shuffle 才能使用这个功能。在使用 Spark 作为一个常驻的服务时候，动态资源调度将大大的提高资源的利用率。例如 JDBCServer 服务，大多数时间该进程并不接受 JDBC 请求，因此将这段空闲时间的资源释放出来，将极大的节约集群的资源。

表25-7 参数说明

参数	描述	默认值
spark.dynamicAllocation.enabled	是否使用动态资源调度，用于根据规模调整注册于该应用的 executor 的数量。注意目前仅在 YARN 模式下有效。 启用动态资源调度必须将 spark.shuffle.service.enabled 设置为 true。以下配置也与此相关：spark.dynamicAllocation.minExecutors、spark.dynamicAllocation.maxExecutors 和 spark.dynamicAllocation.initialExecutors。	• JDBCServer2x：true • SparkResource2x：false
spark.dynamicAllocation.minExecutors	最小 Executor 个数。	0
spark.dynamicAllocation.initialEx	初始 Executor 个数。	spark.dynamicAllocation.

参数	描述	默认值
ecutors		minExecutors
spark.dynamicAllocation.maxExecutors	最大 executor 个数。	2048
spark.dynamicAllocation.schedulerBacklogTimeout	调度第一次超时时间。单位为秒。	1s
spark.dynamicAllocation.sustainedSchedulerBacklogTimeout	调度第二次及之后超时时间。	1s
spark.dynamicAllocation.executorIdleTimeout	普通 Executor 空闲超时时间。单位为秒。	60
spark.dynamicAllocation.cachedExecutorIdleTimeout	含有 cached blocks 的 Executor 空闲超时时间。	• JDBCServer2x：2147483647s • IndexServer2x：2147483647s • SparkResource2x：120

Spark Streaming

Spark Streaming 是在 Spark 批处理平台提供的流式数据的处理能力，以“mini-batch”的方式处理从外部输入的数据。

在 Spark 客户端的“spark-defaults.conf”文件中配置如下参数。

表25-8 参数说明

参数	描述	默认值
spark.streaming.receiver.writeAheadLog.enable	启用预写日志（WAL）功能。所有通过 Receiver 接收的输入数据将被保存至预写日志，预写日志可以保证 Driver 程序出错后数据可以恢复。	false
spark.streaming.unpersist	由 Spark Streaming 产生和保存的 RDDs 自动从 Spark 的内存中强制移除。Spark Streaming 接收的原始输入数据也将自动清除。设置为 false 时原始	true

参数	描述	默认值
	输入数据和存留的 RDDs 不会自动清除，因此在 streaming 应用外部依然可以访问，但是这会占用更多的 Spark 内存。	

Spark Streaming Kafka

Receiver 是 Spark Streaming 一个重要的组成部分，它负责接收外部数据，并将数据封装为 Block，提供给 Streaming 消费。最常见的数据源是 Kafka，Spark Streaming 对 Kafka 的集成也是最完善的，不仅有可靠性的保障，而且也支持从 Kafka 直接作为 RDD 输入。

表25-9 参数说明

参数	描述	默认值
spark.streaming.kafka.maxRatePerPartition	使用 Kafka direct stream API 时，从每个 Kafka 分区读取数据的最大速率（每秒记录数量）。	-
spark.streaming.blockInterval	在被存入 Spark 之前 Spark Streaming Receiver 接收数据累积成数据块的间隔（毫秒）。推荐最小值为 50 毫秒。	200ms
spark.streaming.receiver.maxRate	每个 Receiver 接收数据的最大速率（每秒记录数量）。配置设置为 0 或者负值将不会对速率设限。	-
spark.streaming.receiver.writeAheadLog.enable	是否使用 ReliableKafkaReceiver。该 Receiver 支持流式数据不丢失。	false

Netty/NIO 及 Hash/Sort 配置

Shuffle 是大数据处理中最重要的一个性能点，网络是整个 Shuffle 过程的性能点。目前 Spark 支持两种 Shuffle 方式，一种是 Hash，另外一种 Sort。网络也有两种方式，Netty 和 NIO。

表25-10 参数说明

参数	描述	默认值
spark.shuffle.manager	处理数据的方式。有两种实现方式可用：sort 和 hash。sort shuffle 对内存的使用率更高，是 Spark 1.2 及后续版本的默认选项。	SORT
spark.shuffle.consolidateFiles	（仅 hash 方式）若要合并在 shuffle 过程中创建的中间文件，需要将该值设置为“true”。文件创建	false

参数	描述	默认值
	的少可以提高文件系统处理性能，降低风险。使用 ext4 或者 xfs 文件系统时，建议设置为“true”。由于文件系统限制，在 ext3 上该设置可能会降低 8 核以上机器的处理性能。	
spark.shuffle.sort.bypassMergeThreshold	该参数只适用于 spark.shuffle.manager 设置为 sort 时。在不做 map 端聚合并且 reduce 任务的 partition 数小于或等于该值时，避免对数据进行归并排序，防止系统处理不必要的排序引起性能下降。	200
spark.shuffle.io.maxRetries	（仅 Netty 方式）如果设为非零值，由于 IO 相关的异常导致的 fetch 失败会自动重试。该重试逻辑有助于大型 shuffle 在发生长 GC 暂停或者网络闪断时保持稳定。	12
spark.shuffle.io.numConnectionsPerPeer	（仅 Netty 方式）为了减少大型集群的连接创建，主机间的连接会被重新使用。对于拥有较多硬盘和少数主机的集群，此操作可能会导致并发性不足以占用所有磁盘，所以用户可以考虑增加此值。	1
spark.shuffle.io.preferDirectBufs	（仅 Netty 方式）使用 off-heap 缓冲区减少 shuffle 和高速缓存块转移期间的垃圾回收。对于 off-heap 内存被严格限制的环境，用户可以将其关闭以强制所有来自 Netty 的申请使用堆内内存。	true
spark.shuffle.io.retryWait	（仅 Netty 方式）等待 fetch 重试期间的时间（秒）。重试引起的最大延迟为 maxRetries * retryWait，默认是 15 秒。	5

普通 Shuffle 配置

表25-11 参数说明

参数	描述	默认值
spark.shuffle.spill	若设为“true”，通过将数据溢出至磁盘来限制 reduce 任务期间内存的使用量。	true
spark.shuffle.spill.compress	是否压缩 shuffle 期间溢出的数据。使用 spark.io.compression.codec 指定的算法进行数据压缩。	true
spark.shuffle.file.buffer	每个 shuffle 文件输出流的内存缓冲区大小（单位：KB）。这些缓冲区可以减少创建中间 shuffle 文件流过程中产生的磁盘寻道和系统调用次数。也可以通过配置项 spark.shuffle.file.buffer.kb 设	32KB

参数	描述	默认值
	置。	
spark.shuffle.compress	是否压缩 map 任务输出文件。建议压缩。使用 spark.io.compression.codec 进行压缩。	true
spark.reducer.maxSizeInFlight	从每个 reduce 任务同时 fetch 的 map 任务输出最大值（单位：MB）。由于每个输出要求创建一个缓冲区进行接收，这代表了每个 reduce 任务固定的内存开销，所以除非拥有大量内存，否则保持低值。也可以通过配置项 spark.reducer.maxMbInFlight 设置。	48MB

Driver 配置

Spark Driver 可以理解为 Spark 提交应用的客户端，所有的代码解析工作都在这个进程中完成，因此该进程的参数尤其重要。下面将以如下顺序介绍 Spark 中进程的参数设置：

- JavaOptions：Java 命令中“-D”后面的参数，可以由 System.getProperty 获取。
- ClassPath：包括 Java 类和 Native 的 Lib 加载路径。
- Java Memory and Cores：Java 进程的内存和 CPU 使用量。
- Spark Configuration：Spark 内部参数，与 Java 进程无关。

表25-12 参数说明

参数	描述	默认值
spark.driver.extraJavaOptions	传递至 driver（驱动程序）的一系列额外 JVM 选项。例如，GC 设置或其他日志记录。 注意：在 Client 模式中，该配置禁止直接在应用程序中通过 SparkConf 设置，因为驱动程序 JVM 已经启动。请通过--driver-java-options 命令行选项或默认 property 文件进行设置。	参考快速配置参数
spark.driver.extraClassPath	附加至 driver 的 classpath 的额外 classpath 条目。 注意：在 Client 模式中，该配置禁止直接在应用程序中通过 SparkConf 设置，因为驱动程序 JVM 已经启动。请通过--driver-java-options 命令行选项或默认 property 文件进行设置。	参考快速配置参数
spark.driver.userClassPathFirst	（试验性）当在驱动程序中加载类时，是否授权用户添加的 jar 优先于 Spark 自身的 jar。这种特性可用于减缓 Spark 依赖和用户依赖之间的冲突。目前该特性仍处于试验阶段，仅用于 Cluster 模式中。	false

参数	描述	默认值
spark.driver.extraLibraryPath	设置一个特殊的 library path 在启动驱动程序 JVM 时使用。 注意：在 Client 模式中，该配置禁止直接在应用程序中通过 SparkConf 设置，因为驱动程序 JVM 已经启动。请通过--driver-java-options 命令行选项或默认 property 文件进行设置。	• JDBCServer2x： ${SPARK_INSTALL_HOME}/spark/native • SparkResource2x： ${DATA_NODE_INSTALL_HOME}/hadoop/lib/native
spark.driver.cores	驱动程序进程使用的核数。仅适用于 Cluster 模式。	1
spark.driver.memory	驱动程序进程使用的内存数量，即 SparkContext 初始化的进程（例如：512M, 2G）。 注意：在 Client 模式中，该配置禁止直接在应用程序中通过 SparkConf 设置，因为驱动程序 JVM 已经启动。请通过--driver-java-options 命令行选项或默认 property 文件进行设置。	4G
spark.driver.maxResultSize	对每个 Spark action 操作（例如“collect”）的所有分区序列化结果的总量限制，至少 1M，设置成 0 表示不限制。如果总量超过该限制，工作任务会中止。限制值设置过高可能会引起驱动程序的内存不足错误（取决于 spark.driver.memory 和 JVM 的对象内存开销）。设置合理的限制可以避免驱动程序出现内存不足的错误。	1G
spark.driver.host	Driver 监听的主机名或 IP 地址，用于 Driver 与 Executor 进行通信。	(local hostname)
spark.driver.port	Driver 监听的端口，用于 Driver 与 Executor 进行通信。	(random)

ExecutorLaucher 配置

ExecutorLauncher 只有在 Yarn-Client 模式下才会存在的角色，Yarn-Client 模式下，ExecutorLauncher 和 Driver 不在同一个进程中，需要对 ExecutorLauncher 的参数进行特殊的配置。

表25-13 参数说明

参数	描述	默认值
spark.yarn.am.extraJavaOptions	在 Client 模式下传递至 YARN Application Master 的一系列额外 JVM 选项。在 Cluster 模式下使用 spark.driver.extraJavaOptions。	参考快速配置参数
spark.yarn.am.memory	针对 Client 模式下 YARN Application Master 使用的内存数量，与 JVM 内存设置字符串格式一致（例如：512m，2g）。在集群模式下，使用 spark.driver.memory。	1G
spark.yarn.am.memoryOverhead	和“spark.yarn.driver.memoryOverhead”一样，但只针对 Client 模式下的 Application Master。	-
spark.yarn.am.cores	针对 Client 模式下 YARN Application Master 使用的核数。在 Cluster 模式下，使用 spark.driver.cores。	1

Executor 配置

Executor 也是单独一个 Java 进程，但不像 Driver 和 AM 只有一个，Executor 可以有多个进程，而目前 Spark 只支持相同的配置，即所有 Executor 的进程参数都必然是一样的。

表25-14 参数说明

参数	描述	默认值
spark.executor.extraJavaOptions	传递至 Executor 的额外 JVM 选项。例如，GC 设置或其他日志记录。请注意不能通过此选项设置 Spark 属性或 heap 大小。Spark 属性应该使用 SparkConf 对象或调用 spark-submit 脚本时指定的 spark-defaults.conf 文件来设置。Heap 大小可以通过 spark.executor.memory 来设置。	参考快速配置参数
spark.executor.extraClassPath	附加至 Executor classpath 的额外的 classpath。这主要是为了向后兼容 Spark 的历史版本。用户一般不用设置此选项。	-
spark.executor.extraLibraryPath	设置启动 executor JVM 时所使用的特殊的 library path。	参考快速配置参数
spark.executor.userClassPathFirst	（试验性）与 spark.driver.userClassPathFirst 相同的功能，但应用于 Executor 实例。	false
spark.executor.memory	每个 Executor 进程使用的内存数量，与 JVM 内存设置字符串的格式相同（例如：512M，2G）。	4G

参数	描述	默认值
spark.executorEnv.[EnvironmentVariableName]	添加由 EnvironmentVariableName 指定的环境变量至 executor 进程。用户可以指定多个来设置多个环境变量。	-
spark.executor.logs.rolling.maxRetainedFiles	设置系统即将保留的最新滚动日志文件的数量。旧的日志文件将被删除。默认关闭。	-
spark.executor.logs.rolling.size.maxBytes	设置滚动 Executor 日志的文件的最大值。默认关闭。数值以字节为单位设置。若要自动清除旧日志，请查看 spark.executor.logs.rolling.maxRetainedFiles。	-
spark.executor.logs.rolling.strategy	设置 executor 日志的滚动策略。默认滚动关闭。可以设置为“time”（基于时间的滚动）或“size”（基于大小的滚动）。当设置为“time”，使用 spark.executor.logs.rolling.time.interval 属性的值作为日志滚动的间隔。当设置为“size”，使用 spark.executor.logs.rolling.size.maxBytes 设置滚动的最大文件大小滚动。	-
spark.executor.logs.rolling.time.interval	设置 executor 日志滚动的时间间隔。默认关闭。合法值为“daily”、“hourly”、“minutely”或任意秒。若要自动清除旧日志，请查看 spark.executor.logs.rolling.maxRetainedFiles。	daily

WebUI

WebUI 展示了 Spark 应用运行的过程和状态。

表25-15 参数说明

参数	描述	默认值
spark.ui.killEnabled	允许停止 Web UI 中的 stage 和相应的 job。 说明 出于安全考虑，将此配置项的默认值设置成 false，以避免用户发生误操作。如果需要开启此功能，则可以在 spark-defaults.conf 配置文件中将此配置项的值设为 true。请谨慎操作。	true
spark.ui.port	应用程序 dashboard 的端口，显示内存和工作量数据。	• JDBC Server 2x：4040 • Spark Resou

参数	描述	默认值
		rce2x：0 • Index Server 2x：22901
spark.ui.retainedJobs	在垃圾回收之前 Spark UI 和状态 API 记住的 job 数。	1000
spark.ui.retainedStages	在垃圾回收之前 Spark UI 和状态 API 记住的 stage 数。	1000

HistoryServer

HistoryServer 读取文件系统中的 EventLog 文件，展示已经运行完成的 Spark 应用在运行时的状态信息。

表25-16 参数说明

参数	描述	默认值
spark.history.fs.logDirectory	History server 的日志目录	-
spark.history.ui.port	JobHistory 侦听连接的端口。	18080
spark.history.fs.updateInterval	History server 所显示信息的更新周期，单位为秒。每次更新检查持久存储中针对事件日志进行的更改。	10s
spark.history.fs.update.interval.seconds	每个事件日志更新检查的间隔。与 spark.history.fs.updateInterval 功能相同，推荐使用 spark.history.fs.updateInterval。	10s
spark.history.updateInterval	该配置项与 spark.history.fs.update.interval.seconds 和 spark.history.fs.updateInterval 功能相同，推荐使用 spark.history.fs.updateInterval。	10s

HistoryServer UI 超时和最大访问数

表25-17 参数说明

参数	描述	默认值

参数	描述	默认值
spark.session.maxAge	设置会话的超时时间，单位秒。此参数只适用于安全模式。普通模式下，无法设置此参数。	600
spark.connection.maxRequest	设置客户端访问 Jobhistory 的最大并发数量。	5000

EventLog

Spark 应用在运行过程中，实时将运行状态以 JSON 格式写入文件系统，用于 HistoryServer 服务读取并重现应用运行时状态。

表25-18 参数说明

参数	描述	默认值
spark.eventLog.enabled	是否记录 Spark 事件，用于应用程序在完成后重构 webUI。	true
spark.eventLog.dir	如果 **spark.eventLog.enabled** 为 **true**，记录 Spark 事件的目录。在此目录下，Spark 为每个应用程序创建文件，并将应用程序的事件记录到文件中。用户也可设置为统一的与 HDFS 目录相似的地址，这样 History server 就可以读取历史文件。	hdfs://hacluster/spark2xJobHistory2x
spark.eventLog.compress	spark.eventLog.enabled 为 true 时，是否压缩记录的事件。	false

EventLog 的周期清理

JobHistory 上的 Event log 是随每次任务的提交而累积的，任务提交的次数多了之后会造成太多文件的存放。Spark 提供了周期清理 Evnet log 的功能，用户可以通过配置开关和相应的清理周期参数来进行控制。

表25-19 参数说明

参数	描述	默认值
spark.history.fs.cleaner.enabled	是否打开清理功能。	true
spark.history.fs.cleaner.interval	清理功能的检查周期。	1d
spark.history.fs.cleaner.maxAge	日志的最长保留时间。	4d

Kryo

Kryo 是一个非常高效的 Java 序列化框架，Spark 中也默认集成了该框架。几乎所有的 Spark 性能调优都离不开将 Spark 默认的序列化器转化为 Kryo 序列化器的过程。目前 Kryo 序列化只支持 Spark 数据层面的序列化，还不支持闭包的序列化。设置 Kryo 序列元，需要将配置项“spark.serializer”设置为“org.apache.spark.serializer.KryoSerializer”，同时也搭配设置以下的配置项，优化 Kryo 序列化的性能。

表25-20 参数说明

参数	描述	默认值
spark.kryo.classesToRegister	使用 Kryo 序列化时，需要注册到 Kryo 的类名，多个类之间用逗号分隔。	-
spark.kryo.referenceTracking	当使用 Kryo 序列化数据时，是否跟踪对同一个对象的引用情况。适用于对象图有循环引用或同一对象有多个副本的情况。否则可以设置为关闭以提升性能。	true
spark.kryo.registrationRequired	是否需要使用 Kryo 来注册对象。当设为“true”时，如果序列化一个未使用 Kryo 注册的对象则会抛出异常。当设为“false”（默认值）时，Kryo 会将未注册的类名称一同写到序列化对象中。该操作会带来大量性能开销，所以在用户还没有从注册队列中删除相应的类时应该开启该选项。	false
spark.kryo.registrator	如果使用 Kryo 序列化，使用 Kryo 将该类注册至定制类。如果需要以定制方式注册类，例如指定一个自定义字段序列化器，可使用该属性。否则 spark.kryo.classesToRegister 会更简单。它应该设置为一个扩展 KryoRegistrator 的类。	-
spark.kryoserializer.buffer.max	Kryo 序列化缓冲区允许的最大值，单位为兆字节。这个值必须大于尝试序列化的对象。当在 Kryo 中遇到“buffer limit exceeded”异常时可以适当增大该值。也可以通过配置项 spark.kryoserializer.buffer.max 配置。	64MB
spark.kryoserializer.buffer	Kryo 序列化缓冲区的初始值，单位为兆字节。每个 worker 的每个核心都会有一个缓冲区。如果有需要，缓冲区会增大到 spark.kryoserializer.buffer.max 设置的值。也可以通过配置项 spark.kryoserializer.buffer 配置。	64KB

Broadcast

Broadcast 用于 Spark 进程间数据块的传输。Spark 中无论 Jar 包、文件还是闭包以及返回的结果都会使用 Broadcast。目前的 Broadcast 支持两种方式，Torrent 与 HTTP。前者将会把数据切成小片，分布到集群中，有需要时从远程获取；后者将文件存入到本地磁盘，有需要时通过 HTTP 方式将整个文件传输到远端。前者稳定性优于后者，因此 Torrent 为默认的 Broadcast 方式。

表25-21 参数说明

参数	描述	默认值
spark.broadcast.factory	使用的广播方式。	org.apache.spark.broadcast.TorrentBroadcastFactory
spark.broadcast.blockSize	TorrentBroadcastFactory 的块大小。该值过大会降低广播时的并行度（速度变慢），过小可能会影响 BlockManager 的性能。	4096
spark.broadcast.compress	在发送广播变量之前是否压缩。建议压缩。	true

Storage

内存计算是 Spark 的最大亮点，Spark 的 Storage 主要管理内存资源。Storage 中主要存储 RDD 在 Cache 过程中产生的数据块。JVM 中堆内存是整体的，因此在 Spark 的 Storage 管理中，“Storage Memory Size”变成了一个非常重要的概念。

表25-22 参数说明

参数	描述	默认值
spark.storage.memoryMapThreshold	超过该块大小的 Block，Spark 会对该磁盘文件进行内存映射。这可以防止 Spark 在内存映射时映射过小的块。一般情况下，对接近或低于操作系统的页大小的块进行内存映射会有高开销。	2m

PORT

表25-23 参数说明

参数	描述	默认值
spark.ui.port	应用仪表盘的端口，显示内存和工作负载数据。	• JDBCServer2x

参数	描述	默认值
		:4040 • SparkResource2x：0
spark.blockManager.port	所有 BlockManager 监听的端口。这些同时存在于 Driver 和 Executor 上。	随机端口范围
spark.driver.port	Driver 监听的端口，用于 Driver 与 Executor 进行通信。	随机端口范围

随机端口范围

所有随机端口必须在一定端口范围内。

表25-24 参数说明

参数	描述	默认值
spark.random.port.min	设置随机端口的最小值。	22600
spark.random.port.max	设置随机端口的最大值。	22899

TIMEOUT

Spark 默认配置能很好的处理中等数据规模的计算任务，但一旦数据量过大，会经常出现超时导致任务失败的场景。在大数据量场景下，需调大 Spark 中的超时参数。

表25-25 参数说明

参数	描述	默认值
spark.files.fetchTimeout	获取通过驱动程序的 SparkContext.addFile()添加的文件时的通信超时（秒）。	60s
spark.network.timeout	所有网络交互的默认超时（秒）。如未配置，则使用该配置代替 spark.core.connection.ack.wait.timeout, spark.akka.timeout, spark.storage.blockManagerSlaveTimeoutMs 或 spark.shuffle.io.connectionTimeout。	360s
spark.core.connection.ack.wait.timeout	连接时应答的超时时间（单位：秒）。为了避免由于 GC 带来的长时间等待，可以设置更大的值。	60

加密

Spark 支持 Akka 和 HTTP（广播和文件服务器）协议的 SSL，但 WebUI 和块转移服务仍不支持 SSL。

SSL 必须在每个节点上配置，并使用特殊协议为通信涉及到的每个组件进行配置。

表25-26 参数说明

参数	描述	默认值
spark.ssl.enabled	是否在所有被支持协议上开启 SSL 连接。 与 spark.ssl.xxx 类似的所有 SSL 设置指示了所有被支持协议的全局配置。为了覆盖特殊协议的全局配置，在协议指定的命名空间中必须重写属性。 使用“spark.ssl.YYY.XXX”设置覆盖由 YYY 指示的特殊协议的全局配置。目前 YYY 可以是基于 Akka 连接的 akka 或广播与文件服务器的 fs。	false
spark.ssl.enabledAlgorithms	以逗号分隔的密码列表。指定的密码必须被 JVM 支持。	-
spark.ssl.keyPassword	key-store 的私人密钥密码。	-
spark.ssl.keyStore	key-store 文件的路径。该路径可以绝对或相对于开启组件的目录。	-
spark.ssl.keyStorePassword	key-store 的密码。	-
spark.ssl.protocol	协议名。该协议必须被 JVM 支持。本页所有协议的参考表。	-
spark.ssl.trustStore	trust-store 文件的路径。该路径可以绝对或相对于开启组件的目录。	-
spark.ssl.trustStorePassword	trust-store 的密码。	-

安全性

Spark 目前支持通过共享密钥认证。可以通过 spark.authenticate 配置参数配置认证。该参数控制 Spark 通信协议是否使用共享密钥执行认证。该认证是确保双边都有相同的共享密钥并被允许通信的基本握手。如果共享密钥不同，通信将不被允许。共享密钥通过如下方式创建：

- 对于 YARN 部署的 Spark，将 spark.authenticate 配置为真会自动处理生成和分发共享密钥。每个应用程序会独占一个共享密钥。

- 对于其他类型部署的 Spark，应该在每个节点上配置 Spark 参数 spark.authenticate.secret。所有 Master/Workers 和应用程序都将使用该密钥。

表25-27 参数说明

参数	描述	默认值
spark.acls.enable	是否开启 Spark acls。如果开启，它将检查用户是否有访问和修改 job 的权限。请注意这要求用户可以被识别。如果用户被识别为无效，检查将不被执行。UI 可以使用过滤器认证和设置用户。	true
spark.admin.acls	逗号分隔的有权限访问和修改所有 Spark job 的用户/管理员列表。如果在共享集群上运行并且工作时有管理员或开发人员帮助调试，可以使用该列表。	admin
spark.authenticate	是否 Spark 认证其内部连接。如果不是运行在 YARN 上，请参见 spark.authenticate.secret。	true
spark.authenticate.secret	设置 Spark 各组件之间验证的密钥。如果不是运行在 YARN 上且认证未开启，需要设置该项。	-
spark.modify.acls	逗号分隔的有权限修改 Spark job 的用户列表。默认情况下只有开启 Spark job 的用户才有修改列表的权限（例如删除列表）。	-
spark.ui.view.acls	逗号分隔的有权限访问 Spark web ui 的用户列表。默认情况下只有开启 Spark job 的用户才有访问权限。	-

开启 Spark 进程间的认证机制

目前 Spark 进程间支持共享秘钥方式的认证机制，通过配置 spark.authenticate 可以控制 Spark 在通信过程中是否做认证。这种认证方式只是通过简单的握手来确定通信双方享有共同的秘钥。

在 Spark 客户端的“spark-defaults.conf”文件中配置如下参数。

表25-28 参数说明

参数	描述	默认值
spark.authenticate	在 Spark on YARN 模式下，将该参数配置成 true 即可。密钥的生成和分发过程是自动完成的，并且每个应用独占一个密钥。	true

Compression

数据压缩是一个以 CPU 换内存的优化策略，因此当 Spark 内存严重不足的时候（由于内存计算的特质，这种情况非常常见），使用压缩可以大幅提高性能。目前 Spark 支持三种压缩算法：snappy，lz4，lzf。Snappy 为默认压缩算法，并且调用 native 方法进行压缩与解压缩，在 Yarn 模式下需要注意堆外内存对 Container 进程的影响。

表25-29 参数说明

参数	描述	默认值
spark.io.compression.codec	用于压缩内部数据的 codec，例如 RDD 分区、广播变量和 shuffle 输出。默认情况下，Spark 支持三种压缩算法：lz4，lzf 和 snappy。可以使用完全合格的类名称指定算法，例如 org.apache.spark.io.LZ4CompressionCodec、org.apache.spark.io.LZFCompressionCodec 及 org.apache.spark.io.SnappyCompressionCodec。	lz4
spark.io.compression.lz4.block.size	当使用 LZ4 压缩算法时 LZ4 压缩中使用的块大小（字节）。当使用 LZ4 时降低块大小同样也会降低 shuffle 内存使用。	32768
spark.io.compression.snappy.block.size	当使用 Snappy 压缩算法时 Snappy 压缩中使用的块大小（字节）。当使用 Snappy 时降低块大小同样也会降低 shuffle 内存使用。	32768
spark.shuffle.compress	是否压缩 map 任务输出文件。建议压缩。使用 spark.io.compression.codec 进行压缩。	true
spark.shuffle.spill.compress	是否压缩在 shuffle 期间溢出的数据。使用 spark.io.compression.codec 进行压缩。	true
spark.eventLog.compress	设置当 spark.eventLog.enabled 设置为 true 时是否压缩记录的事件。	false
spark.broadcast.compress	在发送之前是否压缩广播变量。建议压缩。	true
spark.rdd.compress	是否压缩序列化的 RDD 分区（例如 StorageLevel.MEMORY_ONLY_SER 的分区）。牺牲部分额外 CPU 的时间可以节省大量空间。	false

在资源不足的情况下，降低客户端运行异常概率

在资源不足的情况下，Application Master 会因等待资源出现超时，导致任务被删除。调整如下参数，降低客户端应用运行异常概率。

在客户端的“spark-defaults.conf”配置文件中调整如下参数。

表25-30 参数说明

参数	说明	默认值
spark.yarn.applicationMaster.waitTries	设置 Application Master 等待 Spark master 的次数，同时也是等待 SparkContext 初始化的次数。增大该参数值，可以防止 AM 任务被删除，降低客户端应用运行异常的概率。	10
spark.yarn.am.memory	调整 AM 的内存。增大该参数值，可以防止 AM 因内存不足而被 RM 删除任务，降低客户端应用运行异常的概率。	1G

25.2.4 SparkOnHBase 概述及基本应用

操作场景

Spark on HBase 为用户提供了在 Spark SQL 中查询 HBase 表，通过 Beeline 工具为 HBase 表进行存数据等操作。通过 HBase 接口可实现创建表、读取表、往表中插入数据等操作。

操作步骤

步骤 1 登录 Manager 界面，选择“集群 > *待操作集群的名称* > 集群属性”查看集群是否为安全模式。

- 是，执行步骤 2。
- 否，执行步骤 5。

步骤 2 选择“集群 > *待操作集群的名称* > 服务 > Spark2x > 配置 > 全部配置 > JDBCServer2x > 默认”，修改以下参数：

表25-31 参数列表 1

参数	默认值	修改结果
spark.yarn.security.credentials.hbase.enabled	false	true

说明

为了保证 Spark2x 可以长期访问 HBase，建议不要修改 HBase 与 HDFS 服务的以下参数：

- dfs.namenode.delegation.token.renew-interval
- dfs.namenode.delegation.token.max-lifetime
- hbase.auth.key.update.interval
- hbase.auth.token.max.lifetime（不可修改，固定值为 604800000 毫秒，即 7 天）

如果必须要修改以上参数，请务必保证 HDFS 参数“dfs.namenode.delegation.token.renew-interval”的值不大于 HBase 参数“hbase.auth.key.update.interval”、“hbase.auth.token.max.lifetime”的值和 HDFS 参数“dfs.namenode.delegation.token.max-lifetime”的值。

步骤 3 选择“SparkResource2x > 默认”，修改以下参数：

表25-32 参数列表 2

参数	默认值	修改结果
spark.yarn.security.credentials.hbase.enabled	false	true

步骤 4 重启 Spark2x 服务，配置生效。

说明

如果需要在 Spark2x 客户端用 Spark on HBase 功能，需要重新下载并安装 Spark2x 客户端。

步骤 5 在 Spark2x 客户端使用 spark-sql 或者 spark-beeline 连接，可以查询由 Hive on HBase 所创建的表，支持通过 SQL 命令创建 HBase 表或创建外表关联 HBase 表。建表前，确认 HBase 中已存在对应 HBase 表，下面以 HBase 表 table1 为例说明。

1. 通过 Beeline 工具创建 HBase 表，命令如下：

```
create table hbaseTable
(
id string,
name string,
age int
)
using org.apache.spark.sql.hbase.HBaseSource
options(
hbaseTableName "table1",
keyCols "id",
colsMapping    "
name=cf1.cq1,
age=cf1.cq2
");
```

说明

- hbaseTable：是创建的 spark 表的表名。
- id string,name string, age int：是 spark 表的字段名和字段类型。
- table1：HBase 表名。
- id：HBase 表的 rowkey 列名。

- name=cf1.cq1, age=cf1.cq2：spark 表的列和 HBase 表的列的映射关系。spark 的 name 列映射 HBase 表的 cf1 列簇的 cq1 列，spark 的 age 列映射 HBase 表的 cf1 列簇的 cq2 列。

2. 通过 csv 文件导入数据到 HBase 表，命令如下：

   hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -Dimporttsv.separator="," -Dimporttsv.columns=HBASE_ROW_KEY,*cf1:cq1,cf1:cq2,cf1:cq3,cf1:cq4,cf1:cq5 table1 /hperson*

   其中：table1 为 HBase 表名，/hperson 为 csv 文件存放的路径。

3. 在 spark-sql 或 spark-beeline 中查询数据，*hbaseTable* 为对应的 spark 表名。命令如下：

   select * from *hbaseTable;*

----结束

25.2.5 SparkOnHBasev2 概述及基本应用

操作场景

Spark on HBaseV2 为用户提供了在 Spark SQL 中查询 HBase 表，通过 Beeline 工具为 HBase 表进行存数据等操作。通过 HBase 接口可实现创建表、读取表、往表中插入数据等操作。

操作步骤

步骤 1 登录 Manager 界面，选择“集群 > *待操作集群的名称* > 集群属性”查看集群是否为安全模式。

- 是，执行步骤 2。
- 否，执行步骤 5。

步骤 2 选择“集群 > *待操作集群的名称* > 服务 > Spark2x > 配置 > 全部配置 > JDBCServer2x > 默认”，修改以下参数：

表25-33 参数列表 1

参数	默认值	修改结果
spark.yarn.security.credentials.hbase.enabled	false	true

说明

为了保证 Spark2x 可以长期访问 HBase，建议不要修改 HBase 与 HDFS 服务的以下参数:

- dfs.namenode.delegation.token.renew-interval
- dfs.namenode.delegation.token.max-lifetime
- hbase.auth.key.update.interval
- hbase.auth.token.max.lifetime（不可修改，固定值为 604800000 毫秒，即 7 天）

如果必须要修改以上参数，请务必保证 HDFS 参数“dfs.namenode.delegation.token.renew-interval”的值不大于 HBase 参数“hbase.auth.key.update.interval”、“hbase.auth.token.max.lifetime”的值和 HDFS 参数“dfs.namenode.delegation.token.max-lifetime”的值。

步骤 3 选择“SparkResource2x > 默认”，修改以下参数：

表25-34 参数列表 2

参数	默认值	修改结果
spark.yarn.security.credentials.hbase.enabled	false	true

步骤 4 重启 Spark2x 服务，配置生效。

说明

如果需要在 Spark2x 客户端用 Spark on HBase 功能，需要重新下载并安装 Spark2x 客户端。

步骤 5 在 Spark2x 客户端使用 spark-sql 或者 spark-beeline 连接，可以查询由 Hive on HBase 所创建的表，支持通过 SQL 命令创建 HBase 表或创建外表关联 HBase 表。具体见下面说明。下面以 HBase 表 table1 为例说明。

1. 通过 spark-beeline 工具创建表的语法命令如下：

   create table *hbaseTable*1
   (*id string, name string, age int***)**
   using org.apache.spark.sql.hbase.HBaseSourceV2
   options(
   hbaseTableName *"table2"***,**
   keyCols *"id"***,**
   colsMapping **"***name=cf1.cq1,age=cf1.cq2***");**

   说明

   - hbaseTable1：是创建的 spark 表的表名。
   - id string,name string, age int：是 spark 表的字段名和字段类型。
   - table2：HBase 表名。
   - id：HBase 表的 rowkey 列名。
   - name=cf1.cq1, age=cf1.cq2：spark 表的列和 HBase 表的列的映射关系。spark 的 name 列映射 HBase 表的 cf1 列簇的 cq1 列，spark 的 age 列映射 HBase 表的 cf1 列簇的 cq2 列。

2. 通过 csv 文件导入数据到 HBase 表，命令如下：

   hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -Dimporttsv.separator="," -Dimporttsv.columns=HBASE_ROW_KEY,*cf1:cq1,cf1:cq2,cf1:cq3,cf1:cq4,cf1:cq5 table2 /hperson*

   其中：table2 为 HBase 表名，/hperson 为 csv 文件存放的路径。

3. 在 spark-sql 或 spark-beeline 中查询数据，*hbaseTable*1 为对应的 spark 表名，命令如下：

   select * from *hbaseTable1;*

----结束

25.2.6 SparkSQL 权限管理（安全模式）

25.2.6.1 SparkSQL 权限介绍

SparkSQL 权限

类似于 Hive，SparkSQL 也是建立在 Hadoop 上的数据仓库框架，提供类似 SQL 的结构化数据。

MRS 提供用户、用户组和角色，集群中的各类权限需要先授予角色，然后将用户或者用户组与角色绑定。用户只有绑定角色或者加入绑定角色的用户组，才能获得权限。

说明

- 如果当前组件使用了 Ranger 进行权限控制，须基于 Ranger 配置相关策略进行权限管理，具体操作可参考添加 Spark2x 的 Ranger 访问权限策略。
- Spark2x 开启或关闭 Ranger 鉴权后，需要重启 Spark2x 服务，并重新下载客户端，或刷新客户端配置文件 spark/conf/spark-defaults.conf：

  开启 Ranger 鉴权：spark.ranger.plugin.authorization.enable=true

  关闭 Ranger 鉴权：spark.ranger.plugin.authorization.enable=false

权限管理介绍

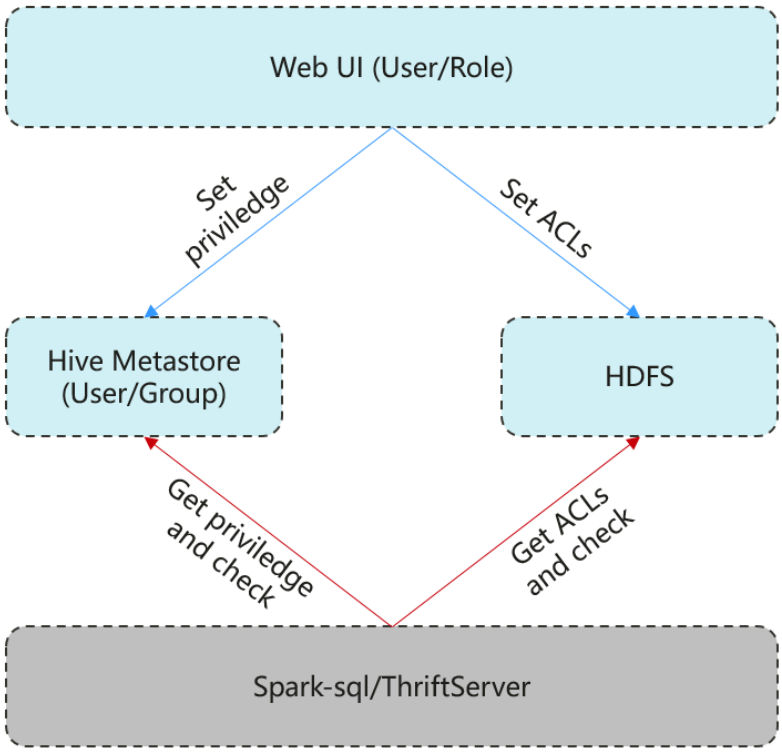
SparkSQL 的权限管理是指 SparkSQL 中管理用户操作数据库的权限系统，以保证不同用户之间操作数据库的独立性和安全性。如果一个用户想操作另一个用户的表、数据库等，需要获取相应的权限才能进行操作，否则会被拒绝。

SparkSQL 权限管理部分集成了 Hive 权限管理的功能。使用 SparkSQL 权限管理功能需要使用 Hive 的 MetaStore 服务和页面上的赋权功能。

图 25-3 展示了 SparkSQL 权限管理的基本架构。主要包含了两部分：页面赋权和服务获权并判断。

- 页面赋权：SparkSQL 仅支持页面赋权的方式。在 FusionInsight Manager 的“系统 > 权限”中，可以进行用户、用户组和角色的添加/删除操作，可以对某个角色进行赋权/撤权。
- 服务获权并判断：当接收到客户端的 DDL、DML 的 SQL 命令时，SparkSQL 服务会向 MetaStore 服务获取客户端用户对数据库信息的已有权限，并检查是否包含了所需的所有权限，如果是则继续执行，否则拒绝该用户的操作。当通过了 MetaStore 的权限检查后，还需进行 HDFS 的 ACLs 权限检查。

图25-3 SparkSQL 权限管理架构图



SparkSQL 还提供了列权限和视图权限，以满足用户不同场景的需求。

- 列权限介绍

  SparkSQL 权限控制由元数据权限控制和 HDFS ACL 权限控制两部分组成。Hive MetaStore 会将表权限自动同步到 HDFS ACL 中时，不会同步列级别的权限。也就是说，当用户对表具有部分列权限或全部列权限时，不能通过 HDFS Client 访问 HDFS 文件。

  - 在 spark-sql 模式下，用户仅具有列级别权限（即列权限用户）将不能访问 HDFS 文件，因此无法访问相应表的列。
  - Beeline/JDBCServer 模式下，用户间赋权，例如将 A 用户创建的表赋权给 B 用户时。
    - “hive.server2.enable.doAs”=true（在 Spark 服务端的“hive-site.xml”文件中配置）

      此时用户 B 不可查询。需在 HDFS 上手动为文件赋读权限。
    - “hive.server2.enable.doAs”=false
      - 用户 A 和 B 均通过 Beeline 连接，用户 B 可查询。
      - A 用户通过 SQL 方式建表，B 用户可在 Beeline 进行查询。

      而其他情况，如 A 用户使用 Beeline 建表，B 用户通过 SQL 查询，或者 A 用户通过 SQL 方式建表，B 用户使用 SQL 方式查询的情况均不支持。需在 HDFS 上手动为文件赋读权限。

说明

由于“spark”用户在 HDFS ACL 的权限控制上为管理员用户权限，Beeline 客户端用户的权限控制仅取决于 Spark 侧的元数据权限。

- 视图权限介绍

  视图权限是指仅对表的视图具有查询、修改等操作的权限，不再依赖于视图所在的表的相应权限。即用户拥有视图的查询权限时，不管是否有表权限都可以进行查询。视图的权限是针对整个表而言的，不支持对其中的部分列创建视图权限。

  视图权限在 SparkSQL 权限上的限制与列权限相似，详细如下：

  - 在 spark-sql 模式下，只有视图权限而没有表权限，且没有 HDFS 的读取权限时，用户不能访问 HDFS 上存储的表的数据，即该情况下不支持对该表的视图进行查询。
  - Beeline/JDBCServer 模式下，用户间赋权，例如将 A 用户创建的视图赋权给 B 用户时。
    - “hive.server2.enable.doAs”=true（在 Spark 服务端的“hive-site.xml”文件中配置）

      此时用户 B 不可查询。需在 HDFS 上手动为文件赋读权限。
    - “hive.server2.enable.doAs”=false
      - 用户 A 和 B 均通过 Beeline 连接，用户 B 可查询。
      - A 用户通过 SQL 方式创建视图，B 用户可在 Beeline 进行查询。

      而其他情况，如 A 用户使用 Beeline 创建视图，B 用户通过 SQL 查询，或者 A 用户通过 SQL 方式创建视图，B 用户使用 SQL 方式查询的情况均不支持。需在 HDFS 上手动为文件赋读权限。

  对表的视图进行相应操作，分别需要具有以下权限。

  - 创建视图时，需要数据库的 CREATE 权限、表的 SELECT、SELECT_of_GRANT 权限。
  - 查询、描述视图时，只需要视图的 SELECT 权限，不需要视图所依赖的表或依赖的视图的 SELECT 权限。若同时查询视图和其他表，则仍然需要其他表的 SELECT 权限，例如：select * from v1 join t1 时，需要有视图 v1 和表 t1 的 SELECT 权限，即使 v1 是基于 t1 的视图，也需要表 t1 的 SELECT 权限。

说明

在 Beeline/JDBCServer 模式下，查询视图只需表的 SELECT 权限；而在 spark-sql 模式下，查询视图需要视图的 SELECT 权限和表的 SELECT 权限。

  - 删除、修改视图时，必须要有视图的 owner 权限。

SparkSQL 权限模型

用户使用 SparkSQL 服务进行 SQL 操作，必须对 SparkSQL 数据库和表（含外表和视图）拥有相应的权限。完整的 SparkSQL 权限模型由元数据权限与 HDFS 文件权限组成。使用数据库或表时所需要的各种权限都是 SparkSQL 权限模型中的一种。

- 元数据权限

元数据权限即在元数据层上进行权限控制，与传统关系型数据库类似，SparkSQL数据库包含“创建”和“查询”权限，表和列包含“查询”、“插入”、“UPDATE”和“删除”权限。SparkSQL中还包含拥有者权限“OWNERSHIP”和管理员权限“管理”。

- 数据文件权限，即HDFS文件权限

  SparkSQL的数据库、表对应的文件保存在HDFS中。默认创建的数据库或表保存在HDFS目录“/user/hive/warehouse”。系统自动以数据库名称和数据库中表的名称创建子目录。访问数据库或者表，需要在HDFS中拥有对应文件的权限，包含“读”、“写”和“执行”权限。

用户对SparkSQL数据库或表执行不同操作时，需要关联不同的元数据权限与HDFS文件权限。例如，对SparkSQL数据表执行查询操作，需要关联元数据权限“查询”，以及HDFS文件权限“读”和“执行”。

使用Manager界面图形化的角色管理功能来管理SparkSQL数据库和表的权限，只需要设置元数据权限，系统会自动关联HDFS文件权限，减少界面操作，提高效率。

SparkSQL使用场景及对应权限

用户通过SparkSQL服务创建数据库需要加入Hive组，不需要角色授权。用户在Hive和HDFS中对自己创建的数据库或表拥有完整权限，可直接创建表、查询数据、删除数据、插入数据、更新数据以及授权他人访问表与对应HDFS目录与文件。

如果用户访问别人创建的表或数据库，需要授予权限。所以根据SparkSQL使用场景的不同，用户需要的权限可能也不相同。

表25-35 SparkSQL使用场景

主要场景	用户需要的权限
使用SparkSQL表、列或数据库	使用其他用户创建的表、列或数据库，不同的场景需要不同的权限，例如： • 创建表，需要“创建”。 • 查询数据，需要“查询”。 • 插入数据，需要“插入”。
关联使用其他组件	部分场景除了SparkSQL权限，还可能需要组件的权限，例如： 使用Spark on HBase，在SparkSQL中查询HBase表数据，需要设置HBase权限。

在一些特殊SparkSQL使用场景下，需要单独设置其他权限。

表25-36 SparkSQL授权注意事项

场景	用户需要的权限
创建SparkSQL数据库、表、外表，或者为已经	• 需要此目录已经存在，客户端用户是目录的属主，且用户对目录拥有“读”、“写”和“执行”权限。同

场景	用户需要的权限
创建的表或外表添加分区，且 Hive 用户指定数据文件保存在“/user/hive/warehouse”以外的 HDFS 目录。	时用户对此目录上层的每一级目录都拥有“读”和“执行”权限。 • 在 Spark2x 中，在创建 HBase 的外表时，需要拥有 Hive 端 database 的“创建”权限。而在 Spark 1.5 中，在创建 HBase 的外表时，需要拥有 Hive 端 database 的“创建”权限，也需要拥有 HBase 端 Namespace 的“创建”权限。
用户使用 load 将指定目录下所有文件或者指定文件，导入数据到表中。	• 数据源为 Linux 本地磁盘，指定目录时需要此目录已经存在，系统用户“omm”对此目录以及此目录上层的每一级目录拥有“r”和“x”的权限。指定文件时需要此文件已经存在，“omm”对此文件拥有“r”的权限，同时对此文件上层的每一级目录拥有“r”和“x”的权限。 • 数据源为 HDFS，指定目录时需要此目录已经存在，SparkSQL 用户是目录属主，且用户对此目录及其子目录拥有“读”、“写”和“执行”权限，并且其上层的每一级目录拥有“读”和“执行”权限。指定文件时需要此文件已经存在，SparkSQL 用户是文件属主，且用户对文件拥有“读”、“写”和“执行”权限，同时对此文件上层的每一级目录拥有“读”和“执行”权限。
创建函数、删除函数或者修改任意数据库。	需要授予“管理”权限。
操作 Hive 中所有的数据库和表。	需加入到 supergroup 用户组，并且授予“管理”权限。
对部分 datasource 表赋予 insert 权限后，执行 insert|analyze 操作前需要单独对 hdfs 上的表目录赋予写权限。	当前对 spark datasource 表赋予 Insert 权限时，若表格式为：text|csv|json|parquet|orc,则不会修改表目录的权限。因此，对以上几种类型的 datasource 表赋予 Insert 权限后，还需要单独对 hdfs 上的表目录赋予写权限，用户才能成功对表执行 insert|analyze 操作。

25.2.6.2 创建 SparkSQL 角色

操作场景

该任务指导系统管理员在 Manager 创建并设置 SparkSQL 的角色。SparkSQL 角色可设置管理员权限以及数据表的数据操作权限。

用户使用 Hive 并创建数据库需要加入 hive 组，不需要角色授权。用户在 Hive 和 HDFS 中对自己创建的数据库或表拥有完整权限，可直接创建表、查询数据、删除数据、插入数据、更新数据以及授权他人访问表与对应 HDFS 目录与文件。默认创建的数据库或表保存在 HDFS 目录“/user/hive/warehouse”。

说明

- 如果当前组件使用了 Ranger 进行权限控制，须基于 Ranger 配置相关策略进行权限管理，具体操作可参考添加 Spark2x 的 Ranger 访问权限策略。
- Spark2x 开启或关闭 Ranger 鉴权后，需要重启 Spark2x 服务，并重新下载客户端，或刷新客户端配置文件 spark/conf/spark-defaults.conf：

  开启 Ranger 鉴权：spark.ranger.plugin.authorization.enable=true

  关闭 Ranger 鉴权：spark.ranger.plugin.authorization.enable=false

操作步骤

1. 登录 Manager 页面，选择“系统 > 权限 > 角色”。
2. 单击“添加角色”，然后“角色名称”和“描述”输入角色名字与描述。
3. 设置角色“配置资源权限”请参见表 25-37。
   - “Hive 管理员权限”：Hive 管理员权限。
   - “Hive 读写权限”：Hive 数据表管理权限，可设置与管理已创建的表的数据操作权限。

说明

- Hive 角色管理支持授予管理员权限、访问表和视图的权限，不支持数据库的授权。
- Hive 管理员权限不支持管理 HDFS 的权限。
- 如果数据库中的表或者表中的文件数量比较多，在授权时可能需要等待一段时间。例如表的文件数量为 1 万时，可能需要等待 2 分钟。

表25-37 设置角色

任务场景	角色授权操作
设置 Hive 管理员权限	在“配置资源权限”的表格中选择“*待操作集群的名称* > Hive”，勾选“Hive 管理权限”。 用户绑定 Hive 管理员角色后，在每个维护操作会话中，还需要执行以下操作： 1. 以客户端安装用户，登录安装 Spark2x 客户端的节点。 2. 执行以下命令配置环境变量。 例如，Spark2x 客户端安装目录为“/opt/client”，执行 **source /opt/client/bigdata_env** **source /opt/client/Spark2x/component_env** 3. 执行以下命令认证用户。 ***kinit*** *Hive 业务用户* 4. 执行以下命令登录客户端工具。 **/opt/client/Spark2x/spark/bin/beeline -u "jdbc:hive2://<zkNode1_IP>:<zkNode1_Port>,<zkNode2_IP>:<zkNode2_Port>,<zkNode3_IP>:<zkNode3_Port>/;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=sparkt**

任务场景	角色授权操作
	hriftserver2x;user.principal=spark2x/hadoop.<*系统域名*>**@**<*系统域名*>**;saslQop=auth-conf;auth=KERBEROS;principal=spark2x/hadoop.**<*系统域名*>**@**<*系统域名*>**;"** 说明 • 其中“<zkNode1_IP>:<zkNode1_Port>,<zkNode2_IP>:<zkNode2_Port>,<zkNode3_IP>:<zkNode3_Port>”是 Zookeeper 的 URL。例如“192.168.81.37:2181,192.168.195.232:2181,192.168.169.84:2181”。 • 其中“sparkthriftserver”是 Zookeeper 上的目录，表示客户端从该目录下随机选择 Triftserver 实例或 proxyThriftServer 进行连接。 • 用户可登录 Manager，选择“系统 > 权限 > 域和互信”，查看“本端域”参数，即为当前系统域名。“spark2x/hadoop.<系统域名>”为用户名，用户的用户名所包含的系统域名所有字母为小写。例如“本端域”参数为“9427068F-6EFA-4833-B43E-60CB641E5B6C.COM”，用户名为“spark2x/hadoo.9427068f-6efa-4833-b43e-60cb641e5b6c.com”。 5. 执行以下命令更新用户的管理员权限。 **set role admin;**
设置在默认数据库中，查询其他用户表的权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > Hive > Hive 读写权限”。 2. 在数据库列表中单击指定的数据库名称，显示数据库中的表。 3. 在指定表的“权限”列，勾选“查询”。
设置在默认数据库中，导入数据到其他用户表的权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > Hive > Hive 读写权限”。 2. 在数据库列表中单击指定的数据库名称，显示数据库中的表。 3. 在指定表的“权限”列，勾选“删除”和“插入”。

4. 单击“确定”完成。

25.2.6.3 配置表、列和数据库的权限

操作场景

使用 SparkSQL 操作表或者数据库时，如果用户访问别人创建的表或数据库，需要授予对应的权限。为了实现更严格权限控制，SparkSQL 也支持列级别的权限控制。如果要访问别人创建的表上某些列，需要授予列权限。以下介绍使用 Manager 角色管理功能在表授权、列授权和数据库授权三个场景下的操作。

操作步骤

SparkSQL 表授权、列授权、数据库授权与 Hive 的操作相同，详情请参见权限管理。

说明

- 在权限管理中，为了方便用户使用，授予数据库下表的任意权限将自动关联该数据库目录的 HDFS 权限。为了避免产生性能问题，取消表的任意权限，系统不会自动取消数据库目录的 HDFS 权限，但对应的用户只能登录数据库和查看表名。
- 若为角色添加或删除数据库的查询权限，数据库中的表也将自动添加或删除查询权限。此机制为 Hive 实现，SparkSQL 与 Hive 保持一致。
- Spark 不支持 struct 数据类型中列名称含有特殊字符（除字母、数字、下划线外的其他字符）。如果 struct 类型中列名称含有特殊字符，在 FusionInsight Manager 的“编辑角色”页面进行授权时，该列将无法正确显示。

相关概念

SparkSQL 的语句在 SparkSQL 中进行处理，权限要求如表 25-38 所示。

表25-38 使用 SparkSQL 表、列或数据库场景权限一览

操作场景	用户需要的权限
CREATE TABLE	“创建”, RWX+ownership（for create external table - the location） 说明 按照指定文件路径创建 datasource 表时，需要 path 后面文件的 RWX+ownership 权限。
DROP TABLE	“Ownership”（of table）
DROP TABLE PROPERTIES	“Ownership”
DESCRIBE TABLE	“查询”
SHOW PARTITIONS	“查询”
ALTER TABLE LOCATION	“Ownership”, RWX+ownership (for new location)
ALTER PARTITION LOCATION	“Ownership”, RWX+ownership (for new partition location)
ALTER TABLE ADD PARTITION	“插入”, RWX+ownership (for partition location)
ALTER TABLE DROP PARTITION	“删除”
ALTER TABLE(all of them except the ones above)	“Update”，“Ownership”

操作场景	用户需要的权限
TRUNCATE TABLE	“Ownership”
CREATE VIEW	“查询”，“Grant Of Select”，“创建”
ALTER VIEW PROPERTIES	“Ownership”
ALTER VIEW RENAME	“Ownership”
ALTER VIEW ADD PARTS	“Ownership”
ALTER VIEW AS	“Ownership”
ALTER VIEW DROPPARTS	“Ownership”
ANALYZE TABLE	“查询”，“插入”
SHOW COLUMNS	“查询”
SHOW TABLE PROPERTIES	“查询”
CREATE TABLE AS SELECT	“查询”，“创建”
SELECT	“查询” 说明 与表一样，对视图进行 SELECT 操作的时候需要有该视图的“查询”权限。
INSERT	“插入”，“删除（for overwrite)”
LOAD	“插入”，“删除”, RWX+ownership(input location)
SHOW CREATE TABLE	“查询”，“Grant Of Select”
CREATE FUNCTION	“管理”
DROP FUNCTION	“管理”
DESC FUNCTION	-
SHOW FUNCTIONS	-
MSCK (metastore check)	“Ownership”
ALTER DATABASE	“管理”
CREATE DATABASE	-
SHOW DATABASES	-
EXPLAIN	“查询”
DROP DATABASE	“Ownership”
DESC DATABASE	-

操作场景	用户需要的权限
CACHE TABLE	“查询”
UNCACHE TABLE	“查询”
CLEAR CACHE TABLE	“管理”
REFRESH TABLE	“查询”
ADD FILE	“管理”
ADD JAR	“管理”
HEALTHCHECK	-

25.2.6.4 配置 SparkSQL 业务使用其他组件的权限

操作场景

SparkSQL 业务还可能需要关联使用其他组件，例如 spark on HBase 需要 HBase 权限。以下介绍 SparkSQL 关联 HBase 服务的操作。

前提条件

- 完成 Spark 客户端的安装，例如安装目录为“/opt/client”。
- 获取一个拥有管理员权限的用户，例如“admin”。

操作步骤

- **Spark on HBase 授权**

  用户如果需要使用类似 SQL 语句的方式来操作 HBase 表，授予权限后可以使用 SparkSQL 访问 HBase 表。以授予用户在 SparkSQL 中查询 HBase 表的权限为例，操作步骤如下：

  说明

  设置“spark.yarn.security.credentials.hbase.enabled”为“true”。

  a. 在 Manager 角色界面创建一个角色，例如“hive_hbase_create”，并授予创建 HBase 表的权限。

     在“配置资源权限”的表格中选择“*待操作集群的名称* > HBase > HBase Scope > global”，勾选命名空间“default”的“创建”，单击“确定”保存。

  说明

  本例中建表是保存在 Hive 的“default”数据库中，默认具有“default”数据库的“建表”权限。如果 Hive 的数据库不是“default”，则还需要执行以下步骤：

  在“配置资源权限”的表格中选择“*待操作集群的名称* > Hive > Hive 读写权限”，勾选所需指定的数据库的“建表”，单击“确定”保存。

b. 在 Manager 角色界面创建一个角色，例如“hive_hbase_submit”，并授予提交任务到 Yarn 的队列的权限。

在“配置资源权限”的表格中选择“*待操作集群的名称* > Yarn > 调度队列 > root”，勾选队列“default”的“提交”，单击“确定”保存。

c. 在 Manager 用户界面创建一个“人机”用户，例如“hbase_creates_user”，加入“hive”组，绑定角色“hive_hbase_create”和“hive_hbase_submit”，用于创建 SparkSQL 表和 HBase 表。

d. 以客户端安装用户登录安装客户端的节点。

e. 执行以下命令，配置环境变量。

source /opt/client/bigdata_env

source /opt/client/Spark2x/component_env

f. 执行以下命令，认证用户。

kinit hbase_creates_user

g. 执行以下命令，进入 Spark JDBCServer 客户端 shell 环境：

/opt/client/Spark2x/spark/bin/beeline -u "jdbc:hive2://<zkNode1_IP>:<zkNode1_Port>,<zkNode2_IP>:<zkNode2_Port>,<zkNode3_IP>:<zkNode3_Port>/;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=sparkthriftserver2x;user.principal=spark2x/hadoop.<*系统域名*>**@**<*系统域名*>**;saslQop=auth-conf;auth=KERBEROS;principal=spark2x/hadoop.**<*系统域名*>**@**<*系统域名*>**;"**

h. 执行以下命令，同时在 SparkSQL 和 HBase 中创建表。例如创建表 hbaseTable。

create table hbaseTable (id string, name string, age int) using org.apache.spark.sql.hbase.HBaseSource options (hbaseTableName "table1", keyCols "id", colsMapping = ", name=cf1.cq1, age=cf1.cq2");

创建好的 SparkSQL 表和 HBase 表分别保存在 Hive 的数据库“default”和 HBase 的命名空间“default”。

i. 在 Manager 角色界面创建一个角色，例如“hive_hbase_select”，并授予查询 SparkSQL on HBase 表 hbaseTable 和 HBase 表 hbaseTable 的权限。

- 在“配置资源权限”的表格中选择“*待操作集群的名称* > HBase > HBase Scope > global > default”，勾选表 hbaseTable 的“读”，单击“确定”保存，授予 HBase 角色查询表的权限。
- 编辑角色，在“配置资源权限”的表格中选择“*待操作集群的名称* > HBase > HBase Scope > global > hbase”，勾选表“hbase:meta”的“执行”，单击“确定”保存。
- 编辑角色，在“配置资源权限”的表格中选择“*待操作集群的名称* > Hive > Hive 读写权限 > default”，勾选表 hbaseTable 的“查询”，单击“确定”保存。

j. 在 Manager 用户界面创建一个“人机”用户，例如“hbase_select_user”，加入“hive”组，绑定角色“hive_hbase_select”，用于查询 SparkSQL 表和 HBase 表。

k. 执行以下命令，配置环境变量。

source /opt/client/bigdata_env

source /opt/client/Spark2x/component_env

l. 执行以下命令，认证用户。

kinit hbase_select_user

m. 执行以下命令，进入 Spark JDBCServer 客户端 shell 环境：

/opt/client/Spark2x/spark/bin/beeline -u "jdbc:hive2://<zkNode1_IP>:<zkNode1_Port>,<zkNode2_IP>:<zkNode2_Port>,<zkNode3_IP>:<zkNode3_Port>/;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=sparkthriftserver2x;user.principal=spark2x/hadoop.<*系统域名*>**@**<*系统域名*>**;saslQop=auth-conf;auth=KERBEROS;principal=spark2x/hadoop.**<*系统域名*>**@**<*系统域名*>**;"**

n. 执行以下命令，使用 SparkSQL 语句查询 HBase 表的数据。

select * from hbaseTable;

25.2.6.5 客户端和服务端配置

SparkSQL 权限管理功能相关的配置如下所示，客户端与服务端的配置相同。要使用表权限功能，需要在服务端和客户端添加如下配置。

- “spark-defaults.conf”配置文件

表25-39 参数说明（1）

参数	描述	默认值
spark.sql.authorization.enabled	是否开启 datasource 语句的权限认证功能。建议将此参数修改为 true，开启权限认证功能。	true

- “hive-site.xml”配置文件

表25-40 参数说明（2）

参数	描述	默认值
hive.metastore.uris	Hive 组件中 MetaStore 服务的地址，如“thrift://10.10.169.84:21088,thrift://10.10.81.37:21088”	-
hive.metastore.sasl.enabled	MetaStore 服务是否使用 SASL 安全加固。表权限功能需要设置为“true”。	true
hive.metastore.kerberos.principal	Hive 组件中 MetaStore 服务的 Principal，如“hive/hadoop.<*系统域名*>@<*系统域名*>”。	hive-metastore/_HOST@EXAMPLE.COM
hive.metastore.thrift.sasl.qop	开启 SparkSQL 权限管理功能后，需将此参数设置为“auth-conf”。	auth-conf
hive.metastore.	MetaStore 服务对应的 token 标识，设为	HiveServer2

参数	描述	默认值
token.signature	“HiveServer2ImpersonationToken”。	ImpersonationToken
hive.security.authenticator.manager	Hive 客户端授权的管理器，需设为“org.apache.hadoop.hive.ql.security.SessionStateUserGroupAuthenticator”。	org.apache.hadoop.hive.ql.security.SessionStateUserMSGroupAuthenticator
hive.security.authorization.enabled	是否开启客户端的授权，需设为“true”。	true
hive.security.authorization.createtable.owner.grants	将哪些权限赋给创建表的 owner，建议设置为“ALL”。	ALL

- MetaStore 服务的 core-site.xml 配置文件

表25-41 参数说明 (3)

参数	描述	默认值
hadoop.proxyuser.spark.hosts	允许 Spark 用户伪装成来自哪些 host 的用户，需设为“*”，代表所有节点。	-
hadoop.proxyuser.spark.groups	允许 Spark 用户伪装成哪些用户组的用户，需设为“*”，代表所有用户组。	-

25.2.7 场景化参数

25.2.7.1 配置多主实例模式

配置场景

集群中支持同时共存多个 ThriftServer 服务，通过客户端可以随机连接其中的任意一个服务进行业务操作。即使集群中一个或多个 ThriftServer 服务停止工作，也不影响用户通过同一个客户端接口连接其他正常的 ThriftServer 服务。

配置描述

登录 Manager，选择“集群 > *待操作集群的名称* > 服务 > Spark2x > 配置”，单击“全部配置”，搜索并修改以下参数。

表25-42 多主实例参数说明

参数	说明	默认值
spark.thriftserver.zookeeper.connection.timeout	Zookeeper 客户端连接超时时间，单位毫秒。	60000
spark.thriftserver.zookeeper.session.timeout	Zookeeper 客户端会话超时时间，单位毫秒。	90000
spark.thriftserver.zookeeper.retry.times	Zookeeper 客户端失联后，重试次数。	3
spark.yarn.queue	JDBCServer 服务所在的 Yarn 队列。	default

25.2.7.2 配置多租户模式

配置场景

多租户模式是将 JDBCServer 和租户绑定，每一个租户对应一个或多个 JDBCServer，一个 JDBCServer 只给一个租户提供服务。不同的租户可以配置不同的 Yarn 队列，从而达到资源隔离。

配置描述

登录 Manager，选择“集群 > *待操作集群的名称* > 服务 > Spark2x > 配置”，单击“全部配置”，搜索并修改以下参数。

表25-43 参数说明

参数	说明	默认值
spark.proxyserver.hash.enabled	是否使用 Hash 算法连接 ProxyServer。 • true 为使用 Hash 算法，使用多租户模式时，该参数需配置为 true。 • false 为使用随机连接，多主实例模式，配置为 false。	true 说明 该参数修改后需要重新下载客户端。
spark.thriftserver.proxy.enabled	是否使用多租户模式。 • false 表示使用多实例模式 • true 表示使用多租户模式	true
spark.thriftserver.proxy.maxThriftServerPerTenancy	多租户模式下，一个租户可启动 JDBCServer 实例的最大个数。	1
spark.thriftserver.proxy.maxSessionPerThriftServer	多租户模式下，单个 JDBCServer 实例的 session 数量超过该值时，如果租户的 JDBCServer 最大实例数量没超过限制，则	50

参数	说明	默认值
	启动新的 JDBCServer，否则输出警告日志。	
spark.thriftserver.proxy.sessionWaitTime	多租户模式下，当 JDBCServer 的 session 连接数为 0 时，停止 JDBCServer 前的等待时间。	180000
spark.thriftserver.proxy.sessionThreshold	多租户模式下，当 JDBCServer 的 session 使用率（公式：当前 session 数 / (spark.thriftserver.proxy.maxSessionPerThriftServer * 当前 JDBCServer 个数)）达到阈值时，自动新增 JDBCServer。	100
spark.thriftserver.proxy.healthcheck.period	多租户模式下，JDBCServer 代理检查 JDBCServer 健康状态周期。	60000
spark.thriftserver.proxy.healthcheck.recheckTimes	多租户模式下，JDBCServer 代理检查 JDBCServer 健康状态失败后重试次数。	3
spark.thriftserver.proxy.healthcheck.waitTime	多租户模式下，JDBCServer 代理发送健康检查，等待 JDBCServer 响应的超时时间。	10000
spark.thriftserver.proxy.session.check.interval	多租户模式下，JDBCServer 代理检查 session 的周期。	6h
spark.thriftserver.proxy.idle.session.timeout	多租户模式下，JDBCServer 代理 session 的空闲超时时间。如果在这段时间内没有做任何操作，session 会被关闭。	7d
spark.thriftserver.proxy.idle.session.check.operation	多租户模式下，JDBCServer 代理 session 的过期是否要判断该 session 上还存在 operation。	true
spark.thriftserver.proxy.idle.operation.timeout	多租户模式下，operation 的超时时间。如果 operation 超时，operation 会被关闭。	5d

25.2.7.3 配置多主实例与多租户模式切换

配置场景

在使用集群中，如果需要在多主实例模式与多租户模式之间切换，则还需要进行如下参数的设置。

- 多租户切换成多主实例模式

  修改 Spark2x 服务的以下参数：

  - spark.thriftserver.proxy.enabled=false
  - spark.scheduler.allocation.file=#{conf_dir}/fairscheduler.xml

  - spark.proxyserver.hash.enabled=false
- 多主实例切换成多租户模式

  修改 Spark2x 服务的以下参数：
  - spark.thriftserver.proxy.enabled=true
  - spark.scheduler.allocation.file=./__spark_conf__/__hadoop_conf__/fairscheduler.xml
  - spark.proxyserver.hash.enabled=true

配置描述

登录 Manager，选择“集群 > *待操作集群的名称* > 服务 > Spark2x > 配置”，单击“全部配置”，搜索并修改以下参数。

表25-44 参数说明

参数	说明	默认值
spark.thriftserver.proxy.enabled	是否使用多租户模式。 • false 表示使用多实例模式 • true 表示使用多租户模式	true
spark.scheduler.allocation.file	公平调度文件路径。 • 多主实例配置为：#{conf_dir}/fairscheduler.xml • 多租户配置为：./__spark_conf__/__hadoop_conf__/fairscheduler.xml	./__spark_conf__/__hadoop_conf__/fairscheduler.xml
spark.proxyserver.hash.enabled	是否使用 Hash 算法连接 ProxyServer。 • true 为使用 Hash 算法，使用多租户模式时，该参数需配置为 true。 • false 为使用随机连接，多主实例模式，配置为 false。	true 说明 该参数修改后需要重新下载客户端。

25.2.7.4 配置事件队列的大小

配置场景

Spark 中见到的 UI、EventLog、动态资源调度等功能都是通过事件传递实现的。事件有 SparkListenerJobStart、SparkListenerJobEnd 等，记录了每个重要的过程。

每个事件在发生后都会保存到一个队列中，Driver 在创建 SparkContext 对象时，会启动一个线程循环的从该队列中依次拿出一个事件，然后发送给各个 Listener，每个 Listener 感知到事件后就会做各自的处理。

因此当队列存放的速度大于获取的速度时，就会导致队列溢出，从而丢失了溢出的事件，影响了 UI、EventLog、动态资源调度等功能。所以为了更灵活的使用，在这边添加一个配置项，用户可以根据 Driver 的内存大小设置合适的值。

配置描述

参数入口：

在执行应用之前，在 Spark 服务配置中修改。在 Manager 系统中，选择“集群 > *待操作集群的名称* > 服务 > Spark2x > 配置”，单击“全部配置”。在搜索框中输入参数名称。

表25-45 参数说明

参数	描述	默认值
spark.scheduler.listenerbus.eventqueue.capacity	事件队列的大小，可以根据 Driver 的内存做适当的配置。	1000000

说明

当 Driver 日志中出现如下的日志时，表示队列溢出了。

1. 普通应用：

```
Dropping SparkListenerEvent because no remaining room in event queue.
This likely means one of the SparkListeners is too slow and cannot keep
up with the rate at which tasks are being started by the scheduler.
```

2. Spark Streaming 应用：

```
Dropping StreamingListenerEvent because no remaining room in event queue.
This likely means one of the StreamingListeners is too slow and cannot keep
up with the rate at which events are being started by the scheduler.
```

25.2.7.5 配置 executor 堆外内存大小

配置场景

当分配的内存太小或者被更高优先级的进程抢占资源时，会出现物理内存超限的情况。调整如下参数，可以防止物理内存超限。

配置描述

参数入口：

在应用提交时通过“--conf”设置这些参数，或者在客户端的“spark-defaults.conf”配置文件中调整如下参数。

表25-46 参数说明

参数	说明	默认值

参数	说明	默认值
spark.executor.memoryOverhead	用于指定每个 executor 的堆外内存大小(MB)，增大该参数值，可以防止物理内存超限。该值是通过max(384,executor-memory*0.1)计算所得，最小值为384。	1024

25.2.7.6 增强有限内存下的稳定性

配置场景

当前 Spark SQL 执行一个查询时需要使用大量的内存，尤其是在做聚合（Aggregate）和关联（Join）操作时，此时如果内存有限的情况下就很容易出现 OutOfMemoryError。有限内存下的稳定性就是确保在有限内存下依然能够正确执行相关的查询，而不出现 OutOfMemoryError。

说明

有限内存并不意味着内存无限小，它只是在内存不足于放下大于内存可用总量几倍的数据时，通过利用磁盘来做辅助从而确保查询依然稳定执行，但依然有一些数据是必须留在内存的，如在做涉及到 Join 的查询时，对于当前用于 Join 的相同 key 的数据还是需要放在内存中，如果该数据量较大而内存较小依然会出现 OutOfMemoryError。

有限内存下的稳定性涉及到 3 个子功能：

1. ExternalSort

   外部排序功能，当执行排序时如果内存不足会将一部分数据溢出到磁盘中。

2. TungstenAggregate

   新 Hash 聚合功能，默认对数据调用外部排序进行排序，然后再进行聚合，因此内存不足时在排序阶段会将数据溢出到磁盘，在聚合阶段因数据有序，在内存中只保留当前 key 的聚合结果，使用的内存较小。

3. SortMergeJoin、SortMergeOuterJoin

   基于有序数据的等值连接。该功能默认对数据调用外部排序进行排序，然后再进行等值连接，因此内存不足时在排序阶段会将数据溢出到磁盘，在连接阶段因数据有序，在内存中只保留当前相同 key 的数据，使用的内存较小。

配置描述

参数入口：

在应用提交时通过“--conf”设置这些参数，或者在客户端的“spark-defaults.conf”配置文件中调整如下参数。

表25-47 参数说明

参数	场景	描述	默认值

参数	场景	描述	默认值
spark.sql.tungsten.enabled	/	类型为 Boolean。 • 当设置的值等于 true 时，表示开启 tungsten 功能，即逻辑计划等同于开启 codegeneration，同时物理计划使用对应的 tungsten 执行计划。 • 当设置的值等于 false 时，表示关闭 tungsten 功能。	true
spark.sql.codegen.wholeStage		类型为 Boolean。 • 当设置的值等于 true 时，表示开启 codegeneration 功能，即运行时对于某些特定的查询将动态生成各逻辑计划代码。 • 当设置的值等于 false 时，表示关闭 codegeneration 功能，运行时使用当前已有静态代码。	true

说明

1. 开启 ExternalSort 除配置 spark.sql.planner.externalSort=true 外，还需配置 spark.sql.unsafe.enabled=false 或者 spark.sql.codegen.wholeStage =false。
2. 如果您需要开启 TungstenAggregate，有如下几种方式：

   将 spark.sql.codegen.wholeStage 和 spark.sql.unsafe.enabled 的值都设置为 true（通过配置文件或命令行方式设置）。

   如果 spark.sql.codegen.wholeStage 和 spark.sql.unsafe.enabled 都不为 true 或者其中一个不为 true，只要 spark.sql.tungsten.enabled 的值设置为 true 时，TungstenAggregate 会开启。

25.2.7.7 配置 WebUI 上查看聚合后的 container 日志

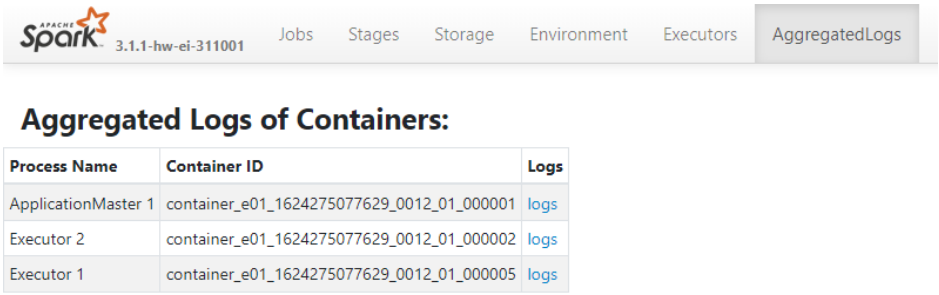
配置场景

当 Yarn 配置“yarn.log-aggregation-enable”为“true”时，就开启了 container 日志聚合功能。日志聚合功能是指：当应用在 Yarn 上执行完成后，NodeManager 将本节点中所有 container 的日志聚合到 HDFS 中，并删除本地日志。详情请参见配置 Container 日志聚合功能。

然而，开启 container 日志聚合功能之后，其日志聚合至 HDFS 目录中，只能通过获取 HDFS 文件来查看日志。开源 Spark 和 Yarn 服务不支持通过 WebUI 查看聚合后的日志。

因此，Spark 在此基础上进行了功能增强。如图 25-4 所示，在 HistoryServer 页面添加“AggregatedLogs”页签，可以通过“logs”链接查看聚合的日志。

图25-4 聚合日志显示页面



配置描述

为了使 WebUI 页面显示日志，需要将聚合日志进行解析和展现。Spark 是通过 Hadoop 的 JobHistoryServer 来解析聚合日志的，所以您可以通过“spark.jobhistory.address”参数，指定 JobHistoryServer 页面地址，即可完成解析和展现。

参数入口：

在应用提交时通过“--conf”设置这些参数，或者在客户端的“spark-defaults.conf”配置文件中调整如下参数。

说明

- 此功能依赖 Hadoop 中的 JobHistoryServer 服务，所以使用聚合日志之前需要保证 JobHistoryServer 服务已经运行正常。
- 如果参数值为空，“AggregatedLogs”页签仍然存在，但是无法通过 logs 链接查看日志。
- 只有当 App 已经 running，HDFS 上已经有该 App 的事件日志文件时才能查看到聚合的 container 日志。
- 正在运行的任务的日志，用户可以通过“Executors”页面的日志链接进行查看，任务结束后日志会汇聚到 HDFS 上，“Executors”页面的日志链接就会失效，此时用户可以通过“AggregatedLogs”页面的 logs 链接查看聚合日志。

表25-48 参数说明

参数	描述	默认值
spark.jobhistory.address	JobHistoryServer 页面的地址，格式：*http(s)://ip:port/jobhistory*。例如，将参数值设置为“https://10.92.115.1:26014/jobhistory”。 默认值为空，表示不能从 WebUI 查看 container 聚合日志。 修改参数后，需重启服务使得配置生效。	-

25.2.7.8 配置 YARN-Client 和 YARN-Cluster 不同模式下的环境变量

配置场景

当前，在 YARN-Client 和 YARN-Cluster 模式下，两种模式的客户端存在冲突的配置，即当客户端为一种模式的配置时，会导致在另一种模式下提交任务失败。

为避免出现如上情况，添加表 25-49 中的配置项，避免两种模式下来回切换参数，提升软件易用性。

- YARN-Cluster 模式下，优先使用新增配置项的值，即服务端路径和参数。
- YARN-Client 模式下，直接使用原有的三个配置项的值。
  原有的三个配置项为：“spark.driver.extraClassPath”、“spark.driver.extraJavaOptions”、“spark.driver.extraLibraryPath”。

说明

不添加表 25-49 中配置项时，使用方式与原有方式一致，程序可正常执行，只是在不同模式下需切换配置。

配置参数

参数入口：

在 Manager 系统中，选择“集群 > *待操作集群的名称* > 服务 > Spark2x > 配置”，单击“全部配置”，在搜索框中输入参数名称。

表25-49 参数介绍

参数	描述	默认值
spark.yarn.cluster.driver.extraClassPath	YARN-Cluster 模式下，Driver 使用的 extraClassPath，配置为服务端的路径和参数。 同时，“spark.driver.extraClassPath”配置成 Spark 客户端路径，可以保证在 YARN-Client 模式下和 YARN-Cluster 模式下不需要切换配置。	${BIGDATA_HOME}/common/runtime/security
spark.yarn.cluster.driver.extraJavaOptions	YARN-Cluster 模式下 Driver 的 extraJavaOptions，配置成服务端的路径和参数。 同时，“spark.driver.extraJavaOptions”配置成 Spark 客户端路径，可以保证 YARN-Client 模式和 YARN-Cluster 模式不需要切换配置。	-Xloggc:<LOG_DIR>/indexserver-%p-gc.log -XX:+PrintGCDetails -XX:-OmitStackTraceInFastThrow -XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=20 -XX:GCLogFileSize=10M -Dlog4j.configuration=./__spark_conf__/__hadoop_conf__/log4j-

参数	描述	默认值
		executor.properties -Dlog4j.configuration.watch=true -Djava.security.auth.login.config=./__spark_conf__/__hadoop_conf__/jaas-zk.conf -Dzookeeper.server.principal=${ZOOKEEPER_SERVER_PRINCIPAL} -Djava.security.krb5.conf=./__spark_conf__/__hadoop_conf__/kdc.conf -Djetty.version=x.y.z -Dorg.xerial.snappy.tempdir=${BIGDATA_HOME}/tmp -Dcarbon.properties.filepath=./__spark_conf__/__hadoop_conf__/carbon.properties -Djdk.tls.ephemeralDHKeySize=2048 -Dspark.ssl.keyStore=./child.keystore #{java_stack_prefer}

25.2.7.9 配置 SparkSQL 的分块个数

配置场景

SparkSQL 在进行 shuffle 操作时默认的分块数为 200。在数据量特别大的场景下，使用默认的分块数就会造成单个数据块过大。如果一个任务产生的单个 shuffle 数据块大于 2G，该数据块在被 fetch 的时候还会报类似错误：

```
Adjusted frame length exceeds 2147483647: 2717729270 - discarded
```

例如，SparkSQL 运行 TPCDS 500G 的测试时，使用默认配置出现错误。所以当数据量较大时需要适当的调整该参数。

配置参数

参数入口：

在 Manager 系统中，选择“集群 > *待操作集群的名称* > 服务 > Spark2x > 配置”，单击“全部配置”。在搜索框中输入参数名称。

表25-50 参数介绍

参数	描述	默认值
spark.sql.shuffle.partitions	SparkSQL 在进行 shuffle 操作时默认的分块数。	200

25.2.7.10 配置 parquet 表的压缩格式

配置场景

当前版本对于 parquet 表的压缩格式分以下两种情况进行配置：

1. 对于分区表，需要通过 parquet 本身的配置项“parquet.compression”设置 parquet 表的数据压缩格式。如在建表语句中设置 tblproperties："parquet.compression"="snappy"。
2. 对于非分区表，需要通过“spark.sql.parquet.compression.codec”配置项来设置 parquet 类型的数据压缩格式。直接设置“parquet.compression”配置项是无效的，因为它会读取“spark.sql.parquet.compression.codec”配置项的值。当“spark.sql.parquet.compression.codec”未做设置时默认值为“snappy”，“parquet.compression”会读取该默认值。

因此，“spark.sql.parquet.compression.codec”配置项只适用于设置非分区表的 parquet 压缩格式。

配置参数

参数入口：

在 Manager 系统中，选择“集群 > *待操作集群的名称* > 服务 > Spark2x > 配置”，单击“全部配置”，在搜索框中输入参数名称。

表25-51 参数介绍

参数	描述	默认值
spark.sql.parquet.compression.codec	对于非分区 parquet 表，设置其存储文件的压缩格式。	snappy

25.2.7.11 配置 WebUI 上显示的 Lost Executor 信息的个数

配置场景

Spark WebUI 中“Executor”页面支持展示 Lost Executor 的信息，对于 JDBCServer 长任务来说，Executor 的动态回收是常态，Lost Executor 个数太多，会撑爆“Executor”页面，因此需要控制页面显示的 Lost Executor 个数。

配置描述

在 Spark 客户端的“spark-defaults.conf”配置文件中进行设置。

表25-52 参数说明

参数	说明	默认值

参数	说明	默认值
spark.ui.retainedDeadExecutors	Spark UI 页面显示的 Lost Executor 的最大个数。	100

25.2.7.12 动态设置日志级别

配置场景

在某些场景下，当任务已经启动后，用户想要修改日志级别以定位问题或者查看想要的信息。

用户可以在进程启动前，在进程的 JVM 参数中增加参数“-Dlog4j.configuration.watch=true”来打开动态设置日志级别的功能。进程启动后，就可以通过修改进程对应的 log4j 配置文件，来调整日志打印级别。

目前支持动态设置日志级别功能的有：Driver 日志、Executor 日志、AM 日志、JobHistory 日志、JDBCServer 日志。

允许设置的日志级别是：FATAL，ERROR，WARN，INFO，DEBUG，TRACE 和 ALL。

配置描述

在进程对应的 JVM 参数配置项中增加以下参数。

表25-53 参数描述

参数	描述	默认值
-Dlog4j.configuration.watch	进程 JVM 参数，设置成“true”用于打开动态设置日志级别功能。	未配置，即为 false。

Driver、Executor、AM 进程的 JVM 参数如表 25-54 所示。在 Spark 客户端的配置文件“spark-defaults.conf”中进行配置。Driver、Executor、AM 进程的日志级别在对应的 JVM 参数中的“-Dlog4j.configuration”参数指定的 log4j 配置文件中设置。

表25-54 进程的 JVM 参数 1

参数	说明	默认日志级别
spark.driver.extraJavaOptions	Driver 的 JVM 参数。	INFO
spark.executor.extraJavaOptions	Executor 的 JVM 参数。	INFO

参数	说明	默认日志级别
spark.yarn.am.extraJavaOptions	AM 的 JVM 参数。	INFO

JobHistory Server 和 JDBCServer 的 JVM 参数如表 25-55 所示。在服务端配置文件“ENV_VARS”中进行配置。JobHistory Server 和 JDBCServer 的日志级别在服务端配置文件“log4j.properties”中设置。

表25-55 进程的 JVM 参数 2

参数	说明	默认日志级别
GC_OPTS	JobHistory Server 的 JVM 参数。	INFO
SPARK_SUBMIT_OPTS	JDBCServer 的 JVM 参数。	INFO

示例：

为了动态修改 Executor 日志级别为 DEBUG，在进程启动之前，修改“spark-defaults.conf”文件中的 Executor 的 JVM 参数“spark.executor.extraJavaOptions”，增加如下配置：

```
-Dlog4j.configuration.watch=true
```

提交用户应用后，修改“spark.executor.extraJavaOptions”中“-Dlog4j.configuration”参数指定的 log4j 日志配置文件（例如：“-Dlog4j.configuration=file:${BIGDATA_HOME}/FusionInsight_Spark2x_8.1.0.1/install/FusionInsight-Spark2x-3.1.1/spark/conf/log4j-executor.properties”）中的日志级别为 DEBUG，如下所示：

```
log4j.rootCategory=DEBUG, sparklog
```

DEBUG 级别生效会有一定的时延。

25.2.7.13 配置 Spark 是否获取 HBase Token

配置场景

使用 Spark 提交任务时，Driver 默认会去 HBase 获取 Token，访问 HBase 则需要配置文件“jaas.conf”进行安全认证。此时若用户未配置“jaas.conf”文件，会导致应用运行失败。

因此，根据应用是否涉及 HBase 进行以下处理：

- 当应用不涉及 HBase 时，即无需获取 HBase Token。此时，将“spark.yarn.security.credentials.hbase.enabled”设置为“false”即可。
- 当应用涉及 HBase 时，将“spark.yarn.security.credentials.hbase.enabled”设置为“true”，且需要在 Driver 端配置“jaas.conf”文件，配置如下：

```
{client}/spark/bin/spark-sql  --master yarn-client --principal {principal} --
keytab {keytab} --driver-java-options "-
Djava.security.auth.login.config={LocalPath}/jaas.conf"
```

在“jaas.conf”中指定 Keytab 和 Prinical，示例如下：

```
Client {
com.sun.security.auth.module.Krb5LoginModule required
useKeyTab=true
keyTab = "{LocalPath}/user.keytab"
principal="super@<系统域名>"
useTicketCache=false
debug=false;
};
```

配置描述

在 Spark 客户端的“spark-defaults.conf”配置文件中进行设置。

表25-56 参数说明

参数	说明	默认值
spark.yarn.security.credentials.hbase.enabled	HBase 是否获取 Token： • true：获取 • false：不获取	false

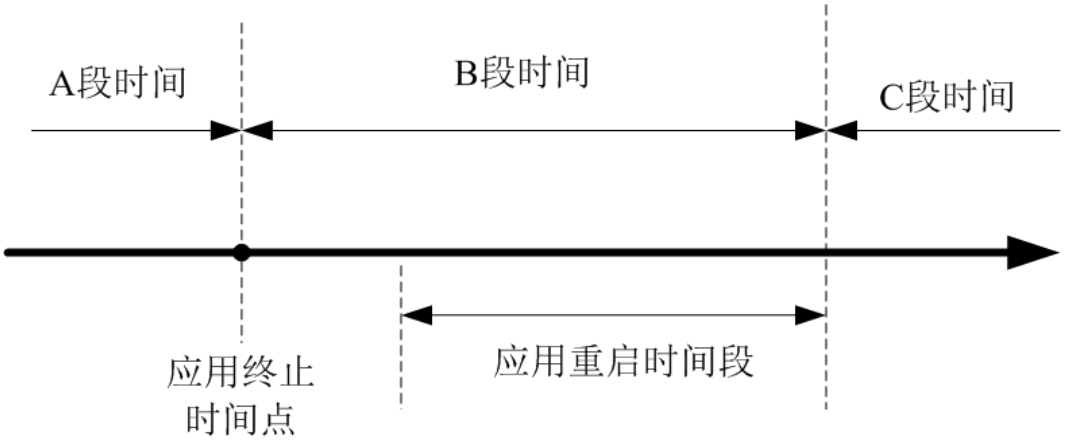
25.2.7.14 配置 Kafka 后进先出

配置场景

当 Spark Streaming 应用与 Kafka 对接，Spark Streaming 应用异常终止并从 checkpoint 恢复重启后，对于进入 Kafka 数据的任务，系统默认优先处理应用终止前（A 段时间）未完成的任务和应用终止到重启完成这段时间内（B 段时间）进入 Kafka 数据生成的任务，最后再处理应用重启完成后（C 段时间）进入 Kafka 数据生成的任务。并且对于 B 段时间进入 Kafka 的数据，Spark 将按照终止时间（batch 时间）生成相应个数的任务，其中第一个任务读取全部数据，其余任务可能不读取数据，造成任务处理压力不均匀。

若 A 段时间的任务和 B 段时间任务处理得较慢，则会影响 C 段时间任务的处理。针对上述场景，Spark 提供 Kafka 后进先出功能。

图25-5 Spark Streaming 应用重启时间轴



开启此功能后，Spark 将优先调度 C 段时间内的任务，若存在多个 C 段任务，则按照任务产生的先后顺序调度执行，再执行 A 段时间和 B 段时间的任务。另外，对于 B 段时间进入 Kafka 的数据，Spark 除了按照终止时间生成相应任务，还将这个期间进入 Kafka 的所有数据均匀分配到各个任务，避免任务处理压力不均匀。

约束条件：

- 目前该功能只适用于 Spark Streaming 中的 Direct 方式，且执行结果与上一个 batch 时间处理结果没有依赖关系（即无 state 操作，如 updatestatebykey）。对多条数据输入流，需要相对独立无依赖的状态，否则可能导致数据切分后结果发生变化。
- Kafka 后进先出功能的开启要求应用只能对接 Kafka 输入源。
- 若提交应用的同时开启 Kafka 后进先出和流控功能，对于 B 段时间进入 Kafka 的数据，将不启动流控功能，以确保读取这些数据的任务调度优先级最低。应用重新启动后 C 段时间的任务启用流控功能。

配置描述

在 Spark Driver 端的“spark-defaults.conf”配置文件中进行设置。

表25-57 参数说明

参数	说明	默认值
spark.streaming.kafka.direct.lifo	配置是否开启 Kafka 后进先出功能。	false
spark.streaming.kafka010.inputstream.class	获取解耦在 FusionInsight 侧的类	org.apache.spark.streaming.kafka010.XXDirectKafkaInputDStream

25.2.7.15 配置对接 Kafka 可靠性

配置场景

Spark Streaming 对接 Kafka 时，当 Spark Streaming 应用重启后，应用根据上一次读取的 topic offset 作为起始位置和当前 topic 最新的 offset 作为结束位置从 Kafka 上读取数据的。

Kafka 服务的 topic 的 leader 异常后，若 Kafka 的 leader 和 follower 的 offset 相差太大，用户重启 Kafka 服务，Kafka 的 follower 和 leader 相互切换，则 Kafka 服务重启后，topic 的 offset 变小。

- 若 Spark Streaming 应用一直在运行，由于 Kafka 上 topic 的 offset 变小，会导致读取 Kafka 数据的起始位置比结束位置大，这样将无法从 Kafka 读取数据，应用报错。
- 若在重启 Kafka 服务前，先停止 Spark Streaming 应用，等 Kafka 重启后，再重启 Spark Streaming 应用使应用从 checkpoint 恢复。此时，Spark Streaming 应用会记录终止前读取到的 offset 位置，以此为基准读取后面的数据，而 Kafka offset 变小（例如从 10 万变成 1 万），Spark Streaming 会等待 Kafka leader 的 offset 增长至 10 万之后才会去消费，导致新发送的 offset 在 1 万至 10 万之间的数据丢失。

针对上述背景，提供配置 Streaming 对接 Kafka 更高级别的可靠性。对接 Kafka 可靠性功能开启后，上述场景处理方式如下。

- 若 Spark Streaming 应用在运行应用时 Kafka 上 topic 的 offset 变小，则会将 Kafka 上 topic 最新的 offset 作为读取 Kafka 数据的起始位置，继续读取后续的数据。

  对于已经生成但未调度处理的任务，若读取的 Kafka offset 区间大于 Kafka 上 topic 的最新 offset，则该任务会运行失败。

  说明

  若任务失败过多，则会将 executor 加入黑名单，从而导致后续的任务无法部署运行。此时用户可以通过配置“spark.blacklist.enabled”参数关闭黑名单功能，黑名单功能默认为开启。

- 若 Kafka 上 topic 的 offset 变小后，Spark Streaming 应用进行重启恢复终止前未处理完的任务若读取的 Kafka offset 区间大于 Kafka 上 topic 的最新 offset，则该任务直接丢弃，不进行处理。

  说明

  若 Streaming 应用中使用了 state 函数，则不允许开启对接 Kafka 可靠性功能。

配置描述

在 Spark 客户端的“spark-defaults.conf”配置文件中进行设置。

表25-58 参数说明

参数	说明	默认值
spark.streaming.Kafka.reliabilit	Spark Streaming 对接 Kafka 是否开启可靠性功能： • true：开启可靠性功能	false

参数	说明	默认值
y	• false：不开启可靠性功能	

25.2.7.16 配置流式读取 driver 执行结果

配置场景

在执行查询语句时，返回结果有可能会很大（10 万数量以上），此时很容易导致 JDBCServer OOM（Out of Memory）。因此，提供数据汇聚功能特性，在基本不牺牲性能的情况下尽力避免 OOM。

配置描述

提供两种不同的数据汇聚功能配置选项，两者在 Spark JDBCServer 服务端的 tunning 选项中进行设置，设置完后需要重启 JDBCServer。

表25-59 参数说明

参数	说明	默认值
spark.sql.bigdata.thriftServer.useHdfsCollect	是否将结果数据保存到 HDFS 中而不是内存中。 优点：由于查询结果保存在 hdfs 端，因此基本不会造成 JDBCServer 的 OOM。 缺点：速度慢。 • true：保存至 HDFS 中 • false：不使用该功能 须知 spark.sql.bigdata.thriftServer.useHdfsCollect 参数设置为 true 时，将结果数据保存到 HDFS 中，但 JobHistory 原生页面上 Job 的描述信息无法正常关联到对应的 SQL 语句，同时 spark-beeline 命令行中回显的 Execution ID 为 null，为解决 JDBCServer OOM 问题，同时显示信息正确，建议选择 spark.sql.userlocalFileCollect 参数进行配置。	false
spark.sql.uselocalFileCollect	是否将结果数据保存在本地磁盘中而不是内存里面。 优点：结果数据小数据量情况下和原生内存的方式相比性能损失可以忽略，大数据情况下（亿级数据）性能远比使用 hdfs，以及原生内存方式好。 缺点：需要调优。大数据情况下建议 JDBCServer driver 端内存 10G，executor 端每个核心分配 3G 内存。 • true：使用该功能 • false: 不使用该功能	false

参数	说明	默认值
spark.sql.collect.Hive	该参数在 spark.sql.uselocalFileCollect 开启的情况下生效。直接序列化的方式，还是间接序列化的方式保存结果数据到磁盘。 优点：针对分区数特别多的表查询结果汇聚性能优于直接使用结果数据保证在磁盘的方式。 缺点：和 spark.sql.uselocalFileCollect 开启时候的缺点一样。 • true：使用该功能 • false：不使用该功能	false
spark.sql.collect.serialize	该参数在 spark.sql.uselocalFileCollect，spark.sql.collect.Hive 同时开启的情况下生效。 作用是进一步提升性能 • java：采用 java 序列化方式收集数据。 • kryo：采用 kryo 序列化方式收集数据，性能要比采用 java 好。	java

说明

参数 spark.sql.bigdata.thriftServer.useHdfsCollect 和 spark.sql.uselocalFileCollect 不能同时设置为 true。

25.2.7.17 配置过滤掉分区表中路径不存在的分区

配置场景

当读取 HIVE 分区表时，如果指定的分区路径在 HDFS 上不存在，则执行 ***select*** 查询时会报 FileNotFoundException 异常。此时可以通过配置“spark.sql.hive.verifyPartitionPath”参数来过滤掉分区路径不存在的分区，来避免读取时报错。

配置描述

可以通过以下两种方式配置是否过滤掉分区表分区路径不存在的分区。

- 在 Spark Driver 端的“spark-defaults.conf”配置文件中进行设置。

表25-60 参数说明

参数	说明	默认值
spark.sql.hive.verifyPartitionPath	配置读取 HIVE 分区表时，是否过滤掉分区表分区路径不存在的分区。 “true”：过滤掉分区路径不存在的分	false

参数	说明	默认值
	区； “false”：不进行过滤。	

- 在 spark-submit 命令提交应用时，通过“--conf”参数配置是否过滤掉分区表分区路径不存在的分区。

  示例：

```
spark-submit --class org.apache.spark.examples.SparkPi  --conf
spark.sql.hive.verifyPartitionPath=true $SPARK_HOME/lib/spark-examples_*.jar
```

25.2.7.18 配置 Spark2x Web UI ACL

配置场景

当 Spark2x Web UI 中有一些不允许其他用户看到的数据时，用户可能想对 UI 进行安全防护。用户一旦登入，Spark2x 可以比较与这个用户相对应的视图 ACLs 来确认是否授权用户访问 UI。

Spark2x 存在两种类型的 Web UI，一种为运行中任务的 Web UI，可以通过 Yarn 原生页面的应用链接或者 REST 接口访问。一种为已结束任务的 Web UI，可以通过 Spark2x JobHistory 服务或者 REST 接口访问。

说明

本章节仅支持安全模式（开启了 Kerberos 认证）集群。

- 运行中任务 Web UI ACL 配置。

  运行中的任务，可通过服务端对如下参数进行配置。

  - “spark.admin.acls”：指定 Web UI 的管理员列表。
  - “spark.admin.acls.groups”：指定管理员组列表。
  - “spark.ui.view.acls”：指定 yarn 界面的访问者列表。
  - “spark.modify.acls.groups”：指定 yarn 界面的访问者组列表。
  - “spark.modify.acls”：指定 Web UI 的修改者列表。
  - “spark.ui.view.acls.groups”：指定 Web UI 的修改者组列表。
- 运行结束后 Web UI ACL 配置。

  运行结束的任务通过客户端的参数“spark.history.ui.acls.enable”控制是否开启 ACL 访问权限。

  如果开启了 ACL 控制，由客户端的“spark.admin.acls”和 s“park.admin.acls.groups”配置指定 Web UI 的管理员列表和管理员组列表，由客户端的“spark.ui.view.acls”和“spark.modify.acls.groups”配置指定查看 Web UI 任务明细的访问者列表和组列表，由客户端的“spark.modify.acls”和“spark.ui.view.acls.groups”配置指定修改 Web UI 任务明细的访问者列表和组列表。

配置描述

登录 FusionInsight Manager 系统，选择“集群 > *待操作集群的名称* > 服务 > Spark2x > 配置”，单击“全部配置”，搜索 acl，在对应的 JobHistory，JDBCServer，SparkResource 和 Spark 界面修改以下参数。

表25-61 参数说明

参数	说明	默认值
spark.history.ui.acls.enable	配置 JobHistory 是否支持单一任务的权限校验。	true
spark.acls.enable	配置是否开启 spark 权限管理。 如果开启，将会检查用户是否有权限访问和修改任务信息。	true
spark.admin.acls	配置 spark 管理员列表，列表中成员有权限管理所有 spark 任务，此处可以配置多个管理员用户，使用“，”分隔。	admin
spark.admin.acls.groups	配置 spark 管理组列表，列表中的组有权限管理所有 spark 任务，此处可以配置多个管理组，使用“，”分隔。	-
spark.modify.acls	配置有权限修改 spark 任务的成员列表。启动任务的用户默认有此权限，此处可以配置多个用户，使用“，”分隔。	-
spark.modify.acls.groups	配置有权限修改 spark 任务的组列表，此处可以配置多个组，使用“，”分隔。	-
spark.ui.view.acls	配置有权限访问 spark 任务的成员列表。启动任务的用户默认有此权限，此处可以配置多个用户，使用“，”分隔。	-
spark.ui.view.acls.groups	配置有权限访问 spark 任务的组列表，此处可以配置多个组，使用“，”分隔。	-

说明

若使用客户端提交任务，“spark.admin.acls”、“spark.admin.acls.groups”、“spark.modify.acls”、“spark.modify.acls.groups”、“spark.ui.view.acls”和“spark.ui.view.acls.groups”参数修改后需要重新下载客户端。

25.2.7.19 配置矢量化读取 ORC 数据

配置场景

ORC 文件格式是一种 Hadoop 生态圈中的列式存储格式，它最初产生自 Apache Hive，用于降低 Hadoop 数据存储空间和加速 Hive 查询速度。和 Parquet 文件格式类似，它并不是一个单纯的列式存储格式，仍然是首先根据行组分割整个表，在每一个行组内按列进行存储，并且文件中的数据尽可能的压缩来降低存储空间的消耗。矢量化读取 ORC 格式的数据能够大幅提升 ORC 数据读取性能。在 Spark2.3 版本中，SparkSQL 支持矢量化读取 ORC 数据（这个特性在 Hive 的历史版本中已经得到支持）。矢量化读取 ORC 格式的数据能够获得比传统读取方式数倍的性能提升。

该特性可以通过下面的配置项开启：

- “spark.sql.orc.enableVectorizedReader”：指定是否支持矢量化方式读取 ORC 格式的数据，默认为 true。
- “spark.sql.codegen.wholeStage”：指定是否需要将多个操作的所有 stage 编译为一个 java 方法，默认为 true。
- “spark.sql.codegen.maxFields”：指定 codegen 的所有 stage 所支持的最大字段数（包括嵌套字段），默认为 100。
- “spark.sql.orc.impl”：指定使用 Hive 还是 Spark SQL native 作为 SQL 执行引擎来读取 ORC 数据，默认为 hive。

配置参数

登录 FusionInsight Manager 系统，选择“集群 > *待操作集群的名称* > 服务 > Spark2x > 配置”，单击“全部配置”，搜索以下参数。

参数	说明	默认值	取值范围
spark.sql.orc.enableVectorizedReader	指定是否支持矢量化方式读取 ORC 格式的数据，默认为 true。	true	[true,false]
spark.sql.codegen.wholeStage	指定是否需要将多个操作的所有 stage 编译为一个 java 方法，默认为 true。	true	[true,false]
spark.sql.codegen.maxFields	指定 codegen 的所有 stage 所支持的最大字段数（包括嵌套字段），默认为 100.	100	大于 0
spark.sql.orc.impl	指定使用 Hive 还是 Spark SQL native 作为 SQL 执行引擎来读取 ORC 数据，默认为 hive。	hive	[hive,native]

说明

1. 使用 SparkSQL 内置的矢量化方式读取 ORC 数据需要满足下面的条件：

- spark.sql.orc.enableVectorizedReader : true，默认是 true，一般不做修改。
- spark.sql.codegen.wholeStage : true，默认为 true，一般不做修改。
- spark.sql.codegen.maxFields 不小于 scheme 的列数。
- 所有的数据类型均为 AtomicType 类型；所谓 Atomic Type 表示非 NULL、UDTs、arrays，maps 类型。如果列中存在这几种类型的任意一种，都无法获得预期的性能。
- spark.sql.orc.impl : native ,默认为 hive。

2. 若使用客户端提交任务，“spark.sql.orc.enableVectorizedReader”、“spark.sql.codegen.wholeStage”、“spark.sql.codegen.maxFields”、“spark.sql.orc.impl”、参数修改后需要重新下载客户端才能生效。

25.2.7.20 Hive 分区修剪的谓词下推增强

配置场景

在旧版本中，对 Hive 表的分区修剪的谓词下推，只支持列名与整数或者字符串的比较表达式的下推，在 2.3 版本中，增加了对 null、in、and、or 表达式的下推支持。

配置参数

登录 FusionInsight Manager 系统，选择“集群 > *待操作集群的名称* > 服务 > Spark2x > 配置”，单击“全部配置”，搜索以下参数。

参数	说明	默认值	取值范围
spark.sql.hive.advancedPartitionPredicatePushdown.enabled	用于配置是否开启 Hive 表的分区谓词下推增强功能。	true	[true,false]

25.2.7.21 支持 Hive 动态分区覆盖语义

配置场景

在旧版本中，使用 insert overwrite 语法覆写分区表时，只支持对指定的分区表达式进行匹配，未指定表达式的分区将被全部删除。在 spark2.3 版本中，增加了对未指定表达式的分区动态匹配的支持，此种语法与 Hive 的动态分区匹配语法行为一致。

配置参数

登录 FusionInsight Manager 系统，选择“集群 > *待操作集群的名称* > 服务 > Spark2x > 配置”，单击“全部配置”，搜索以下参数。

参数	说明	默认值	取值范围

参数	说明	默认值	取值范围
spark.sql.sources.partitionOverwriteMode	当前执行 insert overwrite 命令插入数据到分区表时，支持两种模式：STATIC 模式和 DYNAMIC 模式。STATIC 模式下，Spark 会按照匹配条件删除所有分区。在 DYNAMIC 模式下，Spark 按照匹配条件匹配分区，并动态匹配没有指定匹配条件的分区。	STATIC	[STATIC,DYNAMIC]

25.2.7.22 配置列统计值直方图 Histogram 用以增强 CBO 准确度

配置场景

Spark 优化 sql 的执行，一般的优化规则都是启发式的优化规则，启发式的优化规则，仅仅根据逻辑计划本身的特点给出优化，没有考虑数据本身的特点，也就是未考虑算子本身的执行代价。Spark 在 2.2 中引入了基于代价的优化规则（CBO）。CBO 会收集表和列的统计信息，结合算子的输入数据集来估计每个算子的输出条数以及字节大小，这些就是执行一个算子的代价。

CBO 会调整执行计划，来最小化端到端的查询时间，中心思路 2 点：

- 尽早过滤不相关的数据。
- 最小化每个算子的代价。

CBO 优化过程分为 2 步：

1. 收集统计信息。
2. 根据输入的数据集估算特定算子的输出数据集。

表级别统计信息包括：记录条数；表数据文件的总大小。

列级别统计信息包括：唯一值个数；最大值；最小值；空值个数；平均长度；最大长度；直方图。

有了统计信息后，就可以估计算子的执行代价了。常见的算子包括过滤条件 Filter 算子和 Join 算子。

直方图为列统计值的一种，可以直观的描述列数据的分布情况，将列的数据从最小值到最大值划分为事先指定数量的槽位（bin），计算各个槽位的上下界的值，使得全部数据都确定槽位后，所有槽位中的数据数量相同（等高直方图）。有了数据的详细分布后，各个算子的代价估计能更加准确，优化效果更好。

该特性可以通过下面的配置项开启：

spark.sql.statistics.histogram.enabled：指定是否开启直方图功能，默认为 false。

配置参数

登录 FusionInsight Manager 系统，选择“集群 > *待操作集群的名称* > 服务 > Spark2x > 配置”，单击“全部配置”，搜索以下参数。

参数	说明	默认值	取值范围
spark.sql.cbo.enabled	开启 CBO 来估计执行计划的统计值。	false	[true,false]
spark.sql.cbo.joinReorder.enabled	开启 CBO 连接重排序。	false	[true,false]
spark.sql.cbo.joinReorder.dp.threshold	动态规划算法中允许的最大的 join 节点数量。	12	>=1
spark.sql.cbo.joinReorder.card.weight	在重连接执行计划代价比较中维度（行数）所占的比重：行数 * 比重 + 文件大小 *（1 - 比重）。	0.7	0-1
spark.sql.statistics.size.autoUpdate.enabled	开启当表的数据发生变化时，自动更新表的大小信息。注意如果表的数据文件总数量非常多时，这个操作会非常耗费资源，减慢对数据的操作速度。	false	[true,false]
spark.sql.statistics.histogram.enabled	开启后，当统计列信息时，会生成直方图。直方图可以提高估计准确度，但是收集直方图信息会有额外工作量。	false	[true,false]
spark.sql.statistics.histogram.numBins	生成的直方图的槽位数。	254	>=2
spark.sql.statistics.ndv.maxError	在生成列级别统计信息时，HyperLogLog++算法允许的最大估计误差。	0.05	0-1
spark.sql.statistics.percentile.accuracy	在生成等高直方图时百分位估计的准确率。该值越大意味着越准确。估计错误值可以通过（1.0 / 百分位估计的准确率）来得到。	10000	>=1

说明

- 如果希望直方图可以在 CBO 中生效，需要满足下面的条件：
- spark.sql.statistics.histogram.enabled : true，默认是 false，修改为 true 开启直方图功能。

- spark.sql.cbo.enabled : true，默认为 false，修改为 true 开启 CBO。
- spark.sql.cbo.joinReorder.enabled : true，默认为 false，修改为 true 开启连接重排序。
- 若使用客户端提交任务，“spark.sql.cbo.enabled”、“spark.sql.cbo.joinReorder.enabled”、“spark.sql.cbo.joinReorder.dp.threshold”、“spark.sql.cbo.joinReorder.card.weight”、“spark.sql.statistics.size.autoUpdate.enabled”、“spark.sql.statistics.histogram.enabled”、“spark.sql.statistics.histogram.numBins”、“spark.sql.statistics.ndv.maxError”、“spark.sql.statistics.percentile.accuracy”参数修改后需要重新下载客户端才能生效。

25.2.7.23 配置 JobHistory 本地磁盘缓存

配置场景

JobHistory 可使用本地磁盘缓存 spark 应用的历史数据，以防止 JobHistory 内存中加载大量应用数据，减少内存压力，同时该部分缓存数据可以复用以提高后续对相同应用的访问速度。

配置参数

登录 FusionInsight Manager 系统，选择“集群 > *待操作集群的名称* > 服务 > Spark2x > 配置”，单击“全部配置”，搜索以下参数。

参数	说明	默认值
spark.history.store.path	JobHistory 缓存历史信息的本地目录，如果设置了此配置，则 JobHistory 会将历史应用数据缓存在本地磁盘而不是内存中	${BIGDATA_HOME}/tmp/spark2x_JobHistory
spark.history.store.maxDiskUsage	JobHistory 本地磁盘缓存的最大可用空间	10g

25.2.7.24 配置 Spark SQL 开启 Adaptive Execution 特性

配置场景

Spark SQL Adaptive Execution 特性用于使 Spark SQL 在运行过程中，根据中间结果优化后续执行流程，提高整体执行效率。当前已实现的特性如下：

1. 自动设置 shuffle partition 数

   在启用 Adaptive Execution 特性前，Spark SQL 根据 spark.sql.shuffle.partitions 配置指定 shuffle 时的 partition 个数。此种方法在一个应用中执行多种 SQL 查询时缺乏灵活性，无法保证所有场景下的性能最优。开启 Adaptive Execution 后，Spark SQL 将自动为每个 shuffle 过程动态设置 partition 个数，而不是使用通用配置，使每次 shuffle 过程自动使用最合理的 partition 数。

2. 动态调整执行计划

在启用 Adaptive Execution 特性前，Spark SQL 根据 RBO 和 CBO 的优化结果创建执行计划，此种方法忽略了数据在运行过程中的结果集变化。比如基于某个大表创建的视图，与其他大表 join 时，即便视图的结果集很小，也无法将执行计划调整为 BroadcastJoin。启用 Adaptive Execution 特性后，Spark SQL 能够在运行过程中根据前面 stage 的运行结果动态调整后续的执行计划，从而获得更好的执行性能。

3. 自动处理数据倾斜

   在执行 SQL 语句时，若存在数据倾斜，可能导致单个 executor 内存溢出、任务执行缓慢等问题。启动 Adaptive Execution 特性后，Spark SQL 能自动处理数据倾斜场景，对倾斜的分区，启动多个 task 进行处理，每个 task 读取若干个 shuffle 输出文件，再对这部分任务的 Join 结果进行 Union 操作，以达到消除数据倾斜的效果

配置参数

登录 FusionInsight Manager 系统，选择“集群 > 服务 > Spark2x > 配置”，单击“全部配置”，搜索以下参数。

参数	说明	默认值
spark.sql.adaptive.enabled	配置是否启用自适应执行功能。 注意：AQE 特性与 DPP（动态分区裁剪）特性同时开启时，SparkSQL 任务执行中会优先执行 DPP 特性，从而使得 AQE 特性不生效。	false
spark.sql.optimizer.dynamicPartitionPruning.enabled	动态分区裁剪功能的开关。	true
spark.sql.adaptive.coalescePartitions.enabled	如果配置为 true 并且“spark.sql.adaptive.enabled”为 true，Spark 将根据目标大小（由 spark.sql.adaptive.advisoryPartitionSizeInBytes 指定）合并连续的随机播放分区，以避免执行过多的小任务。	true
spark.sql.adaptive.coalescePartitions.initialPartitionNum	合并之前的 shuffle 分区的初始数量，默认等于 spark.sql.shuffle.partitions。只有当 spark.sql.adaptive.enabled 和 spark.sql.adaptive.coalescePartitions.enabled 都为 true 时，该配置才有效。创建时可选，初始分区数必须为正数。	200
spark.sql.adaptive.coalescePartitions.minPartitionNum	合并后的最小 shuffle 分区数。如果不设置，默认为 Spark 集群的默认并行度。只有当 spark.sql.adaptive.enabled 和 spark.sql.adaptive.coalescePartitions.enable 都为 true 时，该配置才有效。创建时可选，最小分区数必须为正数。	1
spark.sql.adaptive.shuffle.t	shuffle 后单个分区的目标大小，从	64MB

参数	说明	默认值
argetPostShuffleInputSize	Spark3.0 开始不再支持。	
spark.sql.adaptive.advisoryPartitionSizeInBytes	自适应优化时（spark.sql.adaptive.enabled 为 true 时）shuffle 分区的咨询大小（单位：字节），在 Spark 聚合小 shuffle 分区或拆分倾斜的 shuffle 分区时生效。	64MB
spark.sql.adaptive.fetchShuffleBlocksInBatch	是否批量取连续的 shuffle 块。对于同一个 map 任务，批量读取连续的 shuffle 块可以减少 IO，提高性能，而不是逐个读取块。注意，只有当 spark.sql.adaptive.enabled 和 spark.sql.adaptive.coalescePartitions.enabled 都为 true 时，单次读取请求中存在多个连续块。这个特性还依赖于一个可重定位的序列化器，使用的级联支持编解码器和新版本的 shuffle 提取协议。	true
spark.sql.adaptive.localShuffleReader.enabled	当“true”且 spark.sql.adaptive.enabled 为“true”时，Spark 在不需要进行 shuffle 分区时，会尝试使用本地 shuffle reader 读取 shuffle 数据，例如：将 sort-merge join 转换为 broadcast-hash join 后。	true
spark.sql.adaptive.skewJoin.enabled	当此配置为 true 且 spark.sql.adaptive.enabled 设置为 true 时，启用运行时自动处理 join 运算中的数据倾斜功能	true
spark.sql.adaptive.skewJoin.skewedPartitionFactor	此配置为一个倍数因子，用于判定分区是否为数据倾斜分区。单个分区被判定为数据倾斜分区的条件为：当一个分区的数据大小超过除此分区外其他所有分区大小的中值与该配置的乘积，并且大小超过 spark.sql.adaptive.skewJoin.skewedPartitionThresholdInBytes 配置值时，此分区被判定为数据倾斜分区	5
spark.sql.adaptive.skewJoin.skewedPartitionThresholdInBytes	分区大小（单位：字节）大于该阈值且大于 spark.sql.adaptive.skewJoin.skewedPartitionFactor 与分区中值的乘积，则认为该分区存在倾斜。理想情况下，此配置应大于 spark.sql.adaptive.advisoryPartitionSizeInBytes.	256MB
spark.sql.adaptive.nonEmptyPartitionRatioForBroadca	两表进行 join 操作的时候，当非空分区比率低于此配置时，无论其大小如何，	0.2

参数	说明	默认值
stJoin	都不会被视为自适应执行中广播哈希连接的生成端。只有当 spark.sql.adaptive.enabled 为 true 时，此配置才有效。	

25.2.7.25 配置 eventlog 日志回滚

配置场景

当 Spark 开启事件日志模式，即设置“spark.eventLog.enabled”为“true”时，就会往配置的一个日志文件中写事件，记录程序的运行过程。当程序运行很久，job 很多，task 很多时就会造成日志文件很大，如 JDBCServer、Spark Streaming 程序。

而日志回滚功能是指在写事件日志时，将元数据事件（EnviromentUpdate，BlockManagerAdded，BlockManagerRemoved，UnpersistRDD，ExecutorAdded，ExecutorRemoved，MetricsUpdate，ApplicationStart，ApplicationEnd，LogStart）写入日志文件中，Job 事件（StageSubmitted， StageCompleted， TaskResubmit，TaskStart，TaskEnd， TaskGettingResult， JobStart，JobEnd）按文件的大小进行决定是否写入新的日志文件。对于 Spark SQL 的应用，Job 事件还包含 ExecutionStart、ExecutionEnd。

Spark 中有个 HistoryServer 服务，其 UI 页面就是通过读取解析这些日志文件获得的。在启动 HistoryServer 进程时，内存大小就已经定了。因此当日志文件很大时，加载解析这些文件就可能会造成内存不足，driver gc 等问题。

所以为了在小内存模式下能加载较大日志文件，需要对大应用开启日志滚动功能。一般情况下，长时间运行的应用建议打开该功能。

配置参数

登录 FusionInsight Manager 系统，选择“集群 > 服务 > Spark2x > 配置”，单击“全部配置”，搜索以下参数。

参数	说明	默认值
spark.eventLog.rolling.enabled	是否启用滚动 event log 文件。如果设置为 true，则会将每个 event log 文件缩减到配置的大小。	true
spark.eventLog.rolling.maxFileSize	当 spark.eventlog.rolling.enabled=true 时，指定要滚动的 event log 文件的最大大小。	128M
spark.eventLog.compression.codec	用于压缩事件日志的编码解码器。默认情况下，spark 提供四种编码解码器：lz4、lzf、snappy 和 zstd。如果没有给出，将使用 spark.io.compression.codec。	无

参数	说明	默认值
spark.eventLog.logStageExecutorMetrics	是否将 executor metrics 的每个 stage 峰值（针对每个 executor）写入 event log。	false

25.2.8 使用 Ranger 时适配第三方 JDK

配置场景

当使用 Ranger 作为 spark sql 的权限管理服务时，访问 RangerAdmin 需要使用集群中的证书。若用户未使用集群中的 JDK 或者 JRE，而是使用第三方 JDK 时，会出现访问 RangerAdmin 失败，进而 spark 应用程序启动失败的问题。

在这个场景下，需要进行以下操作，将集群中的证书导入第三方 JDK 或者 JRE 中。

配置方法

步骤 1 导出集群中的证书：

1. 安装集群客户端，例如安装路径为“/opt/client”。
2. 执行以下命令，切换到客户端安装目录。

   cd /opt/client
3. 执行以下命令配置环境变量。

   source bigdata_env
4. 生成证书文件

   keytool -export -alias fusioninsightsubroot -storepass changeit -keystore /opt/client/JRE/jre/lib/security/cacerts -file fusioninsightsubroot.crt

步骤 2 将集群中的证书导入第三方 JDK 或者 JRE 中

将步骤 1 中生成的 **fusioninsightsubroot.crt** 文件拷贝到第三方 JRE 节点上，设置好该节点的 JAVA_HOME 环境变量后，执行以下命令导入证书：

keytool -import -trustcacerts -alias fusioninsightsubroot -storepass changeit -file fusioninsightsubroot.crt -keystore MY_JRE/lib/security/cacerts

说明

'MY_JRE'表示第三方 JRE 安装路径，请自行修改。

----结束

25.3 Spark2x 日志介绍

日志描述

日志存储路径：

- Executor 运行日志：“${BIGDATA_DATA_HOME}/hadoop/data${i}/nm/containerlogs/application_${appid}/container_{$contid}”

说明

运行中的任务日志存储在以上路径中，运行结束后会基于 Yarn 的配置确定是否汇聚到 HDFS 目录中，详情请参见 Yarn 常用参数。

- 其他日志：“/var/log/Bigdata/spark2x”

日志归档规则：

- 使用 yarn-client 或 yarn-cluster 模式提交任务时，Executor 日志默认 50MB 滚动存储一次，最多保留 10 个文件，不压缩。
- JobHistory2x 日志默认 100MB 滚动存储一次，最多保留 100 个文件，压缩存储。
- JDBCServer2x 日志默认 100MB 滚动存储一次，最多保留 100 个文件，压缩存储。
- IndexServer2x 日志默认 100MB 滚动存储一次，最多保留 100 个文件，压缩存储。
- JDBCServer2x 审计日志默认 20MB 滚动存储一次，最多保留 20 个文件，压缩存储。
- 日志大小和压缩文件保留个数可以在 FusionInsight Manager 界面中配置。

表25-62 Spark2x 日志列表

日志类型	日志文件名	描述
SparkResource2x 日志	spark.log	Spark2x 服务初始化日志。
	prestart.log	prestart 脚本日志。
	cleanup.log	安装卸载实例时的清理日志。
	spark-availability-check.log	Spark2x 服务健康检查日志。
	spark-service-check.log	Spark2x 服务检查日志
JDBCServer2x 日志	JDBCServer-start.log	JDBCServer2x 启动日志。
	JDBCServer-stop.log	JDBCServer2x 停止日志。
	JDBCServer.log	JDBCServer2x 运行时，Driver 端日志。
	jdbc-state-check.log	JDBCServer2x 健康检查日志。
	jdbcserver-omm-pid***-gc.log.*.current	JDBCServer2x 进程 gc 日志。
	spark-omm-org.apache.spark.sql.hive.thriftserver.HiveThriftProxyServer2-***.out*	JDBCServer2x 进程启动信息日志。若进程停止，会打印 jstack 信息。

日志类型	日志文件名	描述
JobHistory2x 日志	jobHistory-start.log	JobHistory2x 启动日志。
	jobHistory-stop.log	JobHistory2x 停止日志。
	JobHistory.log	JobHistory2x 运行过程日志。
	jobhistory-omm-pid***-gc.log.*.current	JobHistory2x 进程 gc 日志。
	spark-omm-org.apache.spark.deploy.history.HistoryServer-***.out*	JobHistory2x 进程启动信息日志。若进程停止，会打印 jstack 信息。
IndexServer2x 日志	IndexServer-start.log	IndexServer2x 启动日志。
	IndexServer-stop.log	IndexServer2x 停止日志。
	IndexServer.log	IndexServer2x 运行时，Driver 端日志。
	indexserver-state-check.log	IndexServer2x 健康检查日志。
	indexserver-omm-pid***-gc.log.*.current	IndexServer2x 进程 gc 日志。
	spark-omm-org.apache.spark.sql.hive.thriftserver.IndexServerProxy-***.out*	IndexServer2x 进程启动信息日志。若进程停止，会打印 jstack 信息。
审计日志	jdbcserver-audit.log ranger-audit.log	JDBCServer2x 审计日志。

日志级别

Spark2x 中提供了如表 25-63 所示的日志级别。日志级别优先级从高到低分别是 ERROR、WARN、INFO、DEBUG。程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。

表25-63 日志级别

级别	描述
ERROR	ERROR 表示当前时间处理存在错误信息。
WARN	WARN 表示当前事件处理存在异常信息。
INFO	INFO 表示记录系统及各事件正常运行状态信息。
DEBUG	DEBUG 表示记录系统及系统的调试信息。

如果您需要修改日志级别，请执行如下操作：

说明

默认情况下配置 Spark2x 日志级别不需要重启服务。

步骤 1 登录 FusionInsight Manager 系统。

步骤 2 选择“集群 > *待操作集群的名称* > 服务 > Spark2x > 配置”。

步骤 3 单击“全部配置”。

步骤 4 左边菜单栏中选择所需修改的角色所对应的日志菜单。

步骤 5 选择所需修改的日志级别。

步骤 6 单击“保存”，然后单击“确定”，成功后配置生效。

----结束

日志格式

表25-64 日志格式

日志类型	格式	示例
运行日志	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<产生该日志的线程名字>|<log 中的 message>|<日志事件的发生位置>	2014-09-22 11:16:23,980 INFO DAGScheduler: Final stage: Stage 0(reduce at SparkPi.scala:35)

25.4 获取运行中 Spark 应用的 Container 日志

运行中 Spark 应用的 Container 日志分散在多个节点中，本章节用于说明如何快速获取 Container 日志。

场景说明

可以通过 yarn logs 命令获取运行在 Yarn 上的应用的日志，针对不同的场景，我们可以使用以下命令获取需要的日志：

1. 获取 application 的完整日志：**yarn logs --applicationId** <*appId*> **-out** <*outputDir*>
   例如：**yarn logs --applicationId application_1574856994802_0016 -out /opt/test**
   执行结果：
   a. 若该 application 处于运行状态，则无法获取 dead 状态的 container 日志
   b. 若该 application 处于结束状态，则可以获取全部归档的 container 日志

2. 获取指定 Container 日志：**yarn logs -applicationId** <*appId*> **-containerId** <*containerId*>

   例如：**yarn logs -applicationId application_1574856994802_0018 -containerId container_e01_1574856994802_0018_01_000003**

   执行结果：

   a. 若该 application 处于运行状态，则无法获取 dead 状态的 Container 日志

   b. 若该 application 处于结束状态，则可获取任意 Container 的日志

3. 获取任意状态的 Container 日志：**yarn logs -applicationId** <*appId*> **-containerId** <*containerId*> **-nodeAddress** <*nodeAddress*>

   例如：**yarn logs -applicationId application_1574856994802_0019 -containerId container_e01_1574856994802_0019_01_000003 -nodeAddress 192-168-1-1:**8041

   执行结果：可获取任意 Container 的日志

说明

此命令的参数中需要填入 nodeAddress，可通过以下命令获取：

yarn node -list -all

25.5 小文件合并工具

工具介绍

在 Hadoop 大规模生产集群中，由于 HDFS 的元数据都保存在 NameNode 的内存中，集群规模受制于 NameNode 单点的内存限制。如果 HDFS 中有大量的小文件，会消耗 NameNode 大量内存，还会大幅降低读写性能，延长作业运行时间。因此，小文件问题是制约 Hadoop 集群规模扩展的关键问题。

本工具主要有如下两个功能：

1. 扫描表中有多少低于用户设定阈值的小文件，返回该表目录中所有数据文件的平均大小。
2. 对表文件提供合并功能，用户可设置合并后的平均文件大小。

支持的表类型

Spark：Parquet、ORC、CSV、Text、Json。

Hive：Parquet、ORC、CSV、Text、RCFile、Sequence、Bucket。

说明

1. 数据有压缩的表在执行合并后会采用 Spark 默认的压缩格式-Snappy。可以通过在客户端设置“spark.sql.parquet.compression.codec”（可选：uncompressed, gzip, lzo, snappy）和 "spark.sql.orc.compression.codec"（可选：uncompressed, zlib, lzo, snappy）来选择 Parquet 和 Orc 表的压缩格式；由于 Hive 和 Spark 表在可选的压缩格式上有区别，除以上列出的压缩格式外，其他的压缩格式不支持。
2. 合并桶表数据，需要先在 Spark2x 客户端的 hive-site.xml 里加上配置：

```
<property>
<name>hive.enforce.bucketing</name>
<value>false</value>
</property>
<property>
<name>hive.enforce.sorting</name>
<value>false</value>
</property>
```

3. Spark 暂不支持 Hive 的加密列特性。

工具使用

下载安装客户端，例如安装目录为“/opt/client”。进入“/opt/client/Spark2x/spark/bin”，执行 mergetool.sh 脚本。

加载环境变量

source /opt/client/bigdata_env

source /opt/client/Spark2x/component_env

扫描功能

命令形式： **sh mergetool.sh scan <db.table> <filesize>**

db.table 的形式是“数据库名.表名”，filesize 为用户自定义的小文件阈值（单位 MB），返回结果为小于该阈值的文件个数，及整个表目录数据文件的平均大小。

例如：**sh mergetool.sh scan default.table1 128**

合并功能

命令形式： **sh mergetool.sh merge <db.table> <filesize> <shuffle>**

db.table 的形式是“数据库名.表名”，filesize 为用户自定义的合并后平均文件大小（单位 MB），shuffle 是一个 boolean 值，取值 true/false，作用是设置合并过程中是否允许数据进行 shuffle。

例如：**sh mergetool.sh merge default.table1 128 false**

提示如下，则操作成功：

```
SUCCESS: Merge succeeded
```

说明

1. 请确保当前用户对合并的表具有 owner 权限。
2. 合并前请确保 HDFS 上有足够的存储空间，至少需要被合并表大小的一倍以上。
3. 合并表数据的操作需要单独进行，在此过程中读表，可能临时出现找不到文件的问题，合并完成后会恢复正常；另外在合并过程中请注意不要对相应的表进行写操作，否则可能会产生数据一致性问题。
4. 若合并完成后，在一直处于连接状态的 spark-beeline/spark-sql session 中查询分区表的数据，出现文件不存在的问题，根据提示可以执行"refresh table 表名"后再重新查询。
5. 请依据实际情况合理设置 filesize 值，例如可以在 scan 得到表中平均文件大小值 average 后，在 merge 时将 filesize 设置一个比 average 更大的值；否则，执行合并后可能出现文件数变得更多的情况。
6. 合并过程中，会将原表数据放入回收站，再填入已合并的数据。若在此过程中发生异常，根据工具提示，可将 trash 目录中的数据通过 hdfs 的 mv 命令恢复。
7. 在 HDFS router 联邦场景下，如果表的根路径与根路径“/user”的目标 NameService 不同，在二次合并时需要手动清理放入回收站的原表文件，否则会导致合并失败。
8. 此工具应用客户端配置，需要做性能调优可修改客户端配置文件的相关配置。

shuffle 设置

对于合并功能，可粗略估计合并前后分区数的变化：

一般来说，旧分区数>新分区数，可设置 shuffle 为 false；但如果旧分区远大于新分区数，例如高于 100 倍以上，可以考虑设置 shuffle 为 true，增加并行度，提高合并的速度。

须知

- 设置 shuffle 为 true（repartition），会有性能上的提升；但是由于 Parquet 和 Orc 存储方式的特殊性，repartition 会使压缩率变小，直接表现是 hdfs 上表的总大小会增大到 1.3 倍。
- 设置 shuffle 为 false（coalesce），合并后的大小不会非常平均，可能会分布在设置的 filesize 左右。

日志存放位置

默认日志存放位置为/tmp/SmallFilesLog.log4j，如需自定义日志存放位置，可在 /opt/client/Spark2x/spark/tool/log4j.properties 中配置 log4j.appender.logfile.File。

25.6 CarbonData 首查优化工具

工具介绍

CarbonData 的首次查询较慢，对于实时性要求较高的节点可能会造成一定的时延。

本工具主要提供以下功能：

- 对查询时延要求较高的表进行首次查询预热。

工具使用

下载安装客户端，例如安装目录为“/opt/client”。进入 目录“/opt/client/Spark2x/spark/bin”，执行 **start-prequery.sh**。

参考表 25-65，配置 prequeryParams.properties。

表25-65 参数列表

参数	说明	示例
spark.prequery.period.max.minute	预热的最大时长，单位分钟	60
spark.prequery.tables	表名配置 database.table:int，表名支持通配符*，int 代表预热多长时间内有更新的表，单位为天。	default.test*:10
spark.prequery.maxThreads	预热时并发的最大线程数	50
spark.prequery.sslEnable	集群安全模式为 true，非安全模式为 false	true
spark.prequery.driver	JDBCServer 的地址 ip:port，如需要预热多个 Server 则需填写多个 Server 的 IP,多个 IP:port 用逗号隔开。	192.168.0.2:22550
spark.prequery.sql	预热的 sql 语句，不同语句冒号隔开	SELECT COUNT(*) FROM %s;SELECT * FROM %s LIMIT 1
spark.security.url	安全模式下 jdbc 所需 url	;saslQop=auth-conf;auth=KERBEROS;principal=spark2x/hadoop.hadoop.com@HADOOP.COM;

说明

spark.prequery.sql 配置的语句在每个所预热的表中都会执行，表名用%s 代替。

脚本使用

命令形式：**sh start-prequery.sh**

执行此条命令需要：将 user.keytab 或 jaas.conf（二选一），krb5.conf（必须）放入 conf 目录中。

说明

- 此工具暂时只支持 Carbon 表。
- 此工具会初始化 Carbon 环境和预读取表的元数据到 JDBCServer，所以更适合在多主实例、静态分配模式下使用。

25.7 Spark2x 性能调优

25.7.1 Spark Core 调优

25.7.1.1 数据序列化

操作场景

Spark 支持两种方式的序列化 ：

- Java 原生序列化 JavaSerializer
- Kryo 序列化 KryoSerializer

序列化对于 Spark 应用的性能来说，具有很大的影响。在特定的数据格式的情况下，KryoSerializer 的性能可以达到 JavaSerializer 的 10 倍以上，而对于一些 Int 之类的基本类型数据，性能的提升就几乎可以忽略。

KryoSerializer 依赖 Twitter 的 Chill 库来实现，相对于 JavaSerializer，主要的问题在于不是所有的 Java Serializable 对象都能支持，兼容性不好，所以需要手动注册类。

序列化功能用在两个地方：序列化任务和序列化数据。Spark 任务序列化只支持 JavaSerializer，数据序列化支持 JavaSerializer 和 KryoSerializer。

操作步骤

Spark 程序运行时，在 shuffle 和 RDD Cache 等过程中，会有大量的数据需要序列化，默认使用 JavaSerializer，通过配置让 KryoSerializer 作为数据序列化器来提升序列化性能。

在开发应用程序时，添加如下代码来使用 KryoSerializer 作为数据序列化器。

- 实现类注册器并手动注册类。

```
package com.etl.common;

import com.esotericsoftware.kryo.Kryo;
import org.apache.spark.serializer.KryoRegistrator;

public class DemoRegistrator implements KryoRegistrator
{
    @Override
    public void registerClasses(Kryo kryo)
    {
        //以下为示例类，请注册自定义的类
        kryo.register(AggrateKey.class);
        kryo.register(AggrateValue.class);
    }
}
```

您可以在 Spark 客户端对 spark.kryo.registrationRequired 参数进行配置，设置是否需要 Kryo 注册序列化。

当参数设置为 true 时，如果工程中存在未被序列化的类，则会抛出异常。如果设置为 false（默认值），Kryo 会自动将未注册的类名写到对应的对象中。此操作会对系统性能造成影响。设置为 true 时，用户需手动注册类，针对未序列化的类，系统不会自动写入类名，而是抛出异常，相对比 false，其性能较好。

- 配置 KryoSerializer 作为数据序列化器和类注册器。

```
val conf = new SparkConf()
conf.set("spark.serializer", "org.apache.spark.serializer.KryoSerializer")
.set("spark.kryo.registrator", "com.etl.common.DemoRegistrator")
```

25.7.1.2 配置内存

操作场景

Spark 是内存计算框架，计算过程中内存不够对 Spark 的执行效率影响很大。可以通过监控 GC（Garbage Collection），评估内存中 RDD 的大小来判断内存是否变成性能瓶颈，并根据情况优化。

监控节点进程的 GC 情况（在客户端的 conf/spark-default.conf 配置文件中，在 spark.driver.extraJavaOptions 和 spark.executor.extraJavaOptions 配置项中添加参数："-verbose:gc -XX:+PrintGCDetails -XX:+PrintGCTimeStamps"

），如果频繁出现 Full GC，需要优化 GC。把 RDD 做 Cache 操作，通过日志查看 RDD 在内存中的大小，如果数据太大，需要改变 RDD 的存储级别来优化。

操作步骤

- 优化 GC，调整老年代和新生代的大小和比例。在客户端的 conf/spark-default.conf 配置文件中，在 spark.driver.extraJavaOptions 和 spark.executor.extraJavaOptions 配置项中添加参数：-XX:NewRatio。如，" -XX:NewRatio=2"，则新生代占整个堆空间的 1/3，老年代占 2/3。
- 开发 Spark 应用程序时，优化 RDD 的数据结构。
  - 使用原始类型数组替代集合类，如可使用 fastutil 库。
  - 避免嵌套结构。

- Key 尽量不要使用 String。

- 开发 Spark 应用程序时，建议序列化 RDD。

  RDD 做 cache 时默认是不序列化数据的，可以通过设置存储级别来序列化 RDD 减小内存。例如：

```
testRDD.persist(StorageLevel.MEMORY_ONLY_SER)
```

25.7.1.3 设置并行度

操作场景

并行度控制任务的数量，影响 shuffle 操作后数据被切分成的块数。调整并行度让任务的数量和每个任务处理的数据与机器的处理能力达到最优。

查看 CPU 使用情况和内存占用情况，当任务和数据不是平均分布在各节点，而是集中在个别节点时，可以增大并行度使任务和数据更均匀的分布在各个节点。增加任务的并行度，充分利用集群机器的计算能力，一般并行度设置为集群 CPU 总和的 2-3 倍。

操作步骤

并行度可以通过如下三种方式来设置，用户可以根据实际的内存、CPU、数据以及应用程序逻辑的情况调整并行度参数。

- 在会产生 shuffle 的操作函数内设置并行度参数，优先级最高。

```
testRDD.groupByKey(24)
```

- 在代码中配置“spark.default.parallelism”设置并行度，优先级次之。

```
val conf = new SparkConf()
conf.set("spark.default.parallelism", 24)
```

- 在“$SPARK_HOME/conf/spark-defaults.conf”文件中配置“spark.default.parallelism”的值，优先级最低。

```
spark.default.parallelism    24
```

25.7.1.4 使用广播变量

操作场景

Broadcast（广播）可以把数据集合分发到每一个节点上，Spark 任务在执行过程中要使用这个数据集合时，就会在本地查找 Broadcast 过来的数据集合。如果不使用 Broadcast，每次任务需要数据集合时，都会把数据序列化到任务里面，不但耗时，还使任务变得很大。

1. 每个任务分片在执行中都需要同一份数据集合时，就可以把公共数据集 Broadcast 到每个节点，让每个节点在本地都保存一份。
2. 大表和小表做 join 操作时可以把小表 Broadcast 到各个节点，从而就可以把 join 操作转变成普通的操作，减少了 shuffle 操作。

操作步骤

在开发应用程序时，添加如下代码，将“testArr”数据广播到各个节点。

```
def main(args: Array[String]) {
  ...
  val testArr: Array[Long] = new Array[Long](200)
  val testBroadcast: Broadcast[Array[Long]] = sc.broadcast(testArr)
  val resultRdd: RDD[Long] = inpputRdd.map(input => handleData(testBroadcast,
input))
  ...
}

def handleData(broadcast: Broadcast[Array[Long]], input: String) {
  val value = broadcast.value
  ...
}
```

25.7.1.5 使用 External Shuffle Service 提升性能

操作场景

Spark 系统在运行含 shuffle 过程的应用时，Executor 进程除了运行 task，还要负责写 shuffle 数据以及给其他 Executor 提供 shuffle 数据。当 Executor 进程任务过重，导致触发 GC（Garbage Collection）而不能为其他 Executor 提供 shuffle 数据时，会影响任务运行。

External shuffle Service 是长期存在于 NodeManager 进程中的一个辅助服务。通过该服务来抓取 shuffle 数据，减少了 Executor 的压力，在 Executor GC 的时候也不会影响其他 Executor 的任务运行。

操作步骤

步骤 1 登录 FusionInsight Manager 系统。

步骤 2 选择“集群 > *待操作集群的名称* > 服务 > Spark2x > 配置”。单击“全部配置”。

步骤 3 选择“SparkResource2x > 默认”，修改以下参数：

表25-66 参数列表

参数	默认值	修改结果
spark.shuffle.service.enabled	false	true

步骤 4 重启 Spark2x 服务，配置生效。

说明

如果需要在 Spark2x 客户端用 External Shuffle Service 功能，需要重新下载并安装 Spark2x 客户端。

----结束

25.7.1.6 Yarn 模式下动态资源调度

操作场景

对于 Spark 应用来说，资源是影响 Spark 应用执行效率的一个重要因素。当一个长期运行的服务（比如 JDBCServer），若分配给它多个 Executor，可是却没有任何任务分配给它，而此时有其他的应用却资源紧张，这就造成了很大的资源浪费和资源不合理的调度。

动态资源调度就是为了解决这种场景，根据当前应用任务的负载情况，实时的增减 Executor 个数，从而实现动态分配资源，使整个 Spark 系统更加健康。

操作步骤

步骤 1 需要先配置 External shuffle service。

步骤 2 登录 FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > Spark2x > 配置 > 全部配置”。在搜索框中输入“spark.dynamicAllocation.enabled”参数名称，将参数的值设置为“true”，表示开启动态资源调度功能。默认情况下关闭此功能。

----结束

下面是一些可选配置，如表 25-67 所示。

表25-67 动态资源调度参数

配置项	说明	默认值
spark.dynamicAllocation.minExecutors	最小 Executor 个数。	0
spark.dynamicAllocation.initialExecutors	初始 Executor 个数。	0
spark.dynamicAllocation.maxExecutors	最大 Executor 个数。	2048
spark.dynamicAllocation.schedulerBacklogTimeout	调度第一次超时时间。	1s
spark.dynamicAllocation.sustainedSchedulerBacklogTimeout	调度第二次及之后超时时间。	1s
spark.dynamicAllocation.executorIdleTimeout	普通 Executor 空闲超时时间。	60s
spark.dynamicAllocation.cachedExecutorIdleTimeout	含有 cached blocks 的 Executor 空闲超时时间。	• JDBCServer2x：2147483647s • IndexServer2x：2147483647s • SparkResource2x：

配置项	说明	默认值
		120

说明

使用动态资源调度功能，必须配置 External Shuffle Service。

25.7.1.7 配置进程参数

操作场景

Spark on Yarn 模式下，有 Driver、ApplicationMaster、Executor 三种进程。在任务调度和运行的过程中，Driver 和 Executor 承担了很大的责任，而 ApplicationMaster 主要负责 container 的启停。

因而 Driver 和 Executor 的参数配置对 Spark 应用的执行有着很大的影响意义。用户可通过如下操作对 Spark 集群性能做优化。

操作步骤

步骤 1 配置 Driver 内存。

Driver 负责任务的调度，和 Executor、AM 之间的消息通信。当任务数变多，任务平行度增大时，Driver 内存都需要相应增大。

您可以根据实际任务数量的多少，为 Driver 设置一个合适的内存。

- 将“spark-defaults.conf”中的“spark.driver.memory”配置项设置为合适大小。
- 在使用 spark-submit 命令时，添加“--driver-memory MEM”参数设置内存。

步骤 2 配置 Executor 个数。

每个 Executor 每个核同时能跑一个 task，所以增加了 Executor 的个数相当于增大了任务的并发度。在资源充足的情况下，可以相应增加 Executor 的个数，以提高运行效率。

- 将“spark-defaults.conf”中的“spark.executor.instance”配置项或者“spark-env.sh”中的“SPARK_EXECUTOR_INSTANCES”配置项设置为合适大小。
- 在使用 spark-submit 命令时，添加“--num-executors NUM”参数设置 Executor 个数。

步骤 3 配置 Executor 核数。

每个 Executor 多个核同时能跑多个 task，相当于增大了任务的并发度。但是由于所有核共用 Executor 的内存，所以要在内存和核数之间做好平衡。

- 将“spark-defaults.conf”中的“spark.executor.cores”配置项或者“spark-env.sh”中的“SPARK_EXECUTOR_CORES”配置项设置为合适大小。
- 在使用 spark-submit 命令时，添加“--executor-cores NUM”参数设置核数。

步骤 4 配置 Executor 内存。

Executor 的内存主要用于任务执行、通信等。当一个任务很大的时候，可能需要较多资源，因而内存也可以做相应的增加；当一个任务较小运行较快时，就可以增大并发度减少内存。

- 将“spark-defaults.conf”中的“spark.executor.memory”配置项或者“spark-env.sh”中的“SPARK_EXECUTOR_MEMORY”配置项设置为合适大小。
- 在使用 spark-submit 命令时，添加“--executor-memory MEM”参数设置内存。

----结束

示例

- 在执行 spark wordcount 计算中。1.6T 数据，250 个 executor。

  在默认参数下执行失败，出现 Futures timed out 和 OOM 错误。

  因为数据量大，task 数多，而 wordcount 每个 task 都比较小，完成速度快。当 task 数多时 driver 端相应的一些对象就变大了，而且每个 task 完成时 executor 和 driver 都要通信，这就会导致由于内存不足，进程之间通信断连等问题。

  当把 Driver 的内存设置到 4g 时，应用成功跑完。
- 使用 JDBCServer 执行 TPC-DS 测试套，默认参数配置下也报了很多错误：Executor Lost 等。而当配置 Driver 内存为 30g，executor 核数为 2，executor 个数为 125，executor 内存为 6g 时，所有任务才执行成功。

25.7.1.8 设计 DAG

操作场景

合理的设计程序结构，可以优化执行效率。在程序编写过程中要尽量减少 shuffle 操作，合并窄依赖操作。

操作步骤

以“同行车判断”例子讲解 DAG 设计的思路。

- **数据格式：**通过收费站时间、车牌号、收费站编号.....
- **逻辑：**以下两种情况下判定这两辆车是同行车：
  - 如果两辆车都通过相同序列的收费站，
  - 通过同一收费站之间的时间差小于一个特定的值。

该例子有两种实现模式，其中实现 1 的逻辑如图 25-6 所示，实现 2 的逻辑如图 25-7 所示。

图25-6 实现 1 逻辑

实现 1 的逻辑说明 ：

1. 根据车牌号聚合该车通过的所有收费站并排序，处理后数据如下：

   车牌号 1，[（通过时间，收费站 3），（通过时间，收费站 2），（通过时间，收费站 4），（通过时间，收费站 5）]
2. 标识该收费站是这辆车通过的第几个收费站。

   （收费站 3，（车牌号 1，通过时间，通过的第 1 个收费站））

   （收费站 2，（车牌号 1，通过时间，通过的第 2 个收费站））

   （收费站 4，（车牌号 1，通过时间，通过的第 3 个收费站））

   （收费站 5，（车牌号 1，通过时间，通过的第 4 个收费站））
3. 根据收费站聚合数据。

   收费站 1，[（车牌号 1，通过时间，通过的第 1 个收费站），（车牌号 2，通过时间，通过的第 5 个收费站），（车牌号 3，通过时间，通过的第 2 个收费站）]
4. 判断两辆车通过该收费站的时间差是否满足同行车的要求，如果满足则取出这两辆车。

   （车牌号 1，车牌号 2），（通过的第 1 个收费站，通过的第 5 个收费站）

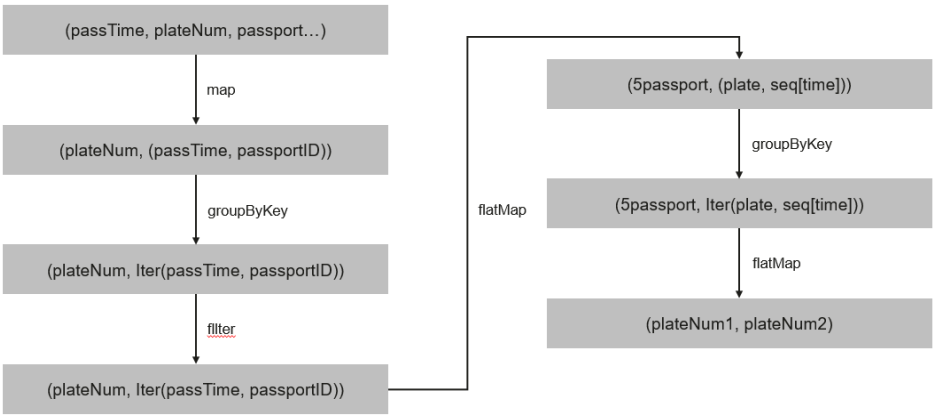
   （车牌号 1，车牌号 3），（通过的第 1 个收费站，通过的第 2 个收费站）
5. 根据通过相同收费站的两辆车的车牌号聚合数据，如下：

   （车牌号 1，车牌号 2），[（通过的第 1 个收费站，通过的第 5 个收费站），（通过的第 2 个收费站，通过的第 6 个收费站），（通过的第 1 个收费站，通过的第 7 个收费站），（通过的第 3 个收费站，通过的第 8 个收费站）]
6. 如果车牌号 1 和车牌号 2 通过相同收费站是顺序排列的（比如收费站 3、4、5 是车牌 1 通过的第 1、2、3 个收费站，是车牌 2 通过的第 6、7、8 个收费站）且数量大于同行车要求的数量则这两辆车是同行车。

实现 1 逻辑的缺点 ：

- 逻辑复杂
- 实现过程中 shuffle 操作过多，对性能影响较大。

图25-7 实现 2 逻辑



实现 2 的逻辑说明 ：

1. 根据车牌号聚合该车通过的所有收费站并排序，处理后数据如下：

   车牌号 1，[（通过时间，收费站 3），（通过时间，收费站 2），（通过时间，收费站 4），（通过时间，收费站 5）]

2. 根据同行车要通过的收费站数量（例子里为 3）分段该车通过的收费站序列，如上面的数据被分解成 ：

   收费站 3->收费站 2->收费站 4， （车牌号 1，[收费站 3 时间，收费站 2 时间，收费站 4 时间]）

   收费站 2->收费站 4->收费站 5， （车牌号 1，[收费站 2 时间，收费站 4 时间，收费站 5 时间]）

3. 把通过相同收费站序列的车辆聚合，如下：

   收费站 3->收费站 2->收费站 4，[（车牌号 1，[收费站 3 时间，收费站 2 时间，收费站 4 时间]），（车牌号 2，[收费站 3 时间，收费站 2 时间，收费站 4 时间]），（车牌号 3，[收费站 3 时间，收费站 2 时间，收费站 4 时间]）]

4. 判断通过相同序列收费站的车辆通过相同收费站的时间差是不是满足同行车的要求，如果满足则说明是同行车。

实现 2 的优点如下：

- 简化了实现逻辑。
- 减少了一个 groupByKey，也就减少了一次 shuffle 操作，提升了性能。

25.7.1.9 经验总结

使用 mapPartitions，按每个分区计算结果

如果每条记录的开销太大，例：

```
rdd.map{x=>conn=getDBConn;conn.write(x.toString);conn.close}
```

则可以使用 MapPartitions，按每个分区计算结果，如

```
rdd.mapPartitions(records => conn.getDBConn;for(item <- records)
write(item.toString); conn.close)
```

使用 mapPartitions 可以更灵活地操作数据，例如对一个很大的数据求 TopN，当 N 不是很大时，可以先使用 mapPartitions 对每个 partition 求 TopN，collect 结果到本地之后再做排序取 TopN。这样相比直接对全量数据做排序取 TopN 效率要高很多。

使用 coalesce 调整分片的数量

coalesce 可以调整分片的数量。coalesce 函数有两个参数：

```
coalesce(numPartitions: Int, shuffle: Boolean = false)
```

当 shuffle 为 true 的时候，函数作用与 repartition(numPartitions: Int)相同，会将数据通过 Shuffle 的方式重新分区；当 shuffle 为 false 的时候，则只是简单的将父 RDD 的多个 partition 合并到同一个 task 进行计算，shuffle 为 false 时，如果 numPartitions 大于父 RDD 的切片数，那么分区不会重新调整。

遇到下列场景，可选择使用 coalesce 算子：

- 当之前的操作有很多 filter 时，使用 coalesce 减少空运行的任务数量。此时使用 coalesce(numPartitions, false)，numPartitions 小于父 RDD 切片数。
- 当输入切片个数太大，导致程序无法正常运行时使用。
- 当任务数过大时候 Shuffle 压力太大导致程序挂住不动，或者出现 linux 资源受限的问题。此时需要对数据重新进行分区，使用 coalesce(numPartitions, true)。

localDir 配置

Spark 的 Shuffle 过程需要写本地磁盘，Shuffle 是 Spark 性能的瓶颈，I/O 是 Shuffle 的瓶颈。配置多个磁盘则可以并行的把数据写入磁盘。如果节点中挂载多个磁盘，则在每个磁盘配置一个 Spark 的 localDir，这将有效分散 Shuffle 文件的存放，提高磁盘 I/O 的效率。如果只有一个磁盘，配置了多个目录，性能提升效果不明显。

Collect 小数据

大数据量不适用 collect 操作。

collect 操作会将 Executor 的数据发送到 Driver 端，因此使用 collect 前需要确保 Driver 端内存足够，以免 Driver 进程发生 OutOfMemory 异常。当不确定数据量大小时，可使用 saveAsTextFile 等操作把数据写入 HDFS 中。只有在能够大致确定数据大小且 driver 内存充足的时候，才能使用 collect。

使用 reduceByKey

reduceByKey 会在 Map 端做本地聚合，使得 Shuffle 过程更加平缓，而 groupByKey 等 Shuffle 操作不会在 Map 端做聚合。因此能使用 reduceByKey 的地方尽量使用该算子，避免出现 groupByKey().map(x=>(x._1,x._2.size))这类实现方式。

广播 map 代替数组

当每条记录需要查表，如果是 Driver 端用广播方式传递的数据，数据结构优先采用 set/map 而不是 Iterator，因为 Set/Map 的查询速率接近 O(1)，而 Iterator 是 O(n)。

数据倾斜

当数据发生倾斜（某一部分数据量特别大），虽然没有 GC（Gabage Collection，垃圾回收），但是 task 执行时间严重不一致。

- 需要重新设计 key，以更小粒度的 key 使得 task 大小合理化。
- 修改并行度。

优化数据结构

- 把数据按列存放，读取数据时就可以只扫描需要的列。
- 使用 Hash Shuffle 时，通过设置 spark.shuffle.consolidateFiles 为 true，来合并 shuffle 中间文件，减少 shuffle 文件的数量，减少文件 IO 操作以提升性能。最终文件数为 reduce tasks 数目。

25.7.2 SQL 和 DataFrame 调优

25.7.2.1 Spark SQL join 优化

操作场景

Spark SQL 中，当对两个表进行 join 操作时，利用 Broadcast 特性（见“使用广播变量”章节），将被广播的表 BroadCast 到各个节点上，从而转变成非 shuffle 操作，提高任务执行性能。

说明

这里 join 操作，只指 inner join。

操作步骤

在 Spark SQL 中进行 Join 操作时，可以按照以下步骤进行优化。为了方便说明，设表 A 和表 B，且 A、B 表都有个名为 name 的列。对 A、B 表进行 join 操作。

1. 估计表的大小。

   根据每次加载数据的大小，来估计表大小。

   也可以在 Hive 的数据库存储路径下直接查看表的大小。首先在 Spark 的配置文件“hive-site.xml”中，查看 Hive 的数据库路径的配置，默认为“/user/hive/warehouse”。Spark 服务多实例默认数据库路径为“/user/*hive*/warehouse”，例如“/user/hive1/warehouse”。

   ```
   <property>
     <name>hive.metastore.warehouse.dir</name>
    <value>${test.warehouse.dir}</value>
    <description></description>
   </property>
   ```

然后通过 hadoop 命令查看对应表的大小。如查看表 A 的大小命令为：

```
hadoop fs -du -s -h ${test.warehouse.dir}/a
```

说明

进行广播操作，需要至少有一个表不是空表。

2. 配置自动广播的阈值。

   Spark 中，判断表是否广播的阈值为 10485760（即 10M）。如果两个表的大小至少有一个小于 10M 时，可以跳过该步骤。

   自动广播阈值的配置参数介绍，见表 25-68。

表25-68 参数介绍

参数	默认值	描述
spark.sql.autoBroadcastJoinThreshold	10485760	当进行 join 操作时，配置广播的最大值。 • 当 SQL 语句中涉及的表中相应字段的大小小于该值时，进行广播。 • 配置为-1 时，将不进行广播。 参见 https://spark.apache.org/docs/3.1.1/sql-programming-guide.html

   配置自动广播阈值的方法：

   - 在 Spark 的配置文件“spark-defaults.conf”中，设置“spark.sql.autoBroadcastJoinThreshold”的值。

     ```
     spark.sql.autoBroadcastJoinThreshold = <size>
     ```

   - 利用 Hive CLI 命令，设置阈值。在运行 Join 操作时，提前运行下面语句：

     ```
     SET spark.sql.autoBroadcastJoinThreshold=<size>;
     ```

3. 进行 join 操作。

   - 两个表的大小都小于阈值。
     - A 表的字节数小于 B 表，则运行 B join A，如

       ```
       SELECT A.name FROM B JOIN A ON A.name = B.name;
       ```

     - 否则运行 A join B。

       ```
       SELECT A.name FROM A JOIN B ON A.name = B.name;
       ```

   - 一个表大于阈值一个表小于阈值。

     将小表进行 BroadCast 操作。

   - 两个表的大小都大于阈值。

     比较查询所涉及的字段大小与阈值的大小。
     - 若某表中涉及字段的大小小于阈值，将该表相应数据进行广播。
     - 若两表中涉及字段的大小都大于阈值，则不进行广播。

4. （可选）如下两种场景，需要执行 Analyze 命令（***ANALYZE TABLE tableName COMPUTE STATISTICS noscan;***）更新表元数据后进行广播。

- 需要广播的表是分区表，新建表且文件类型为非 Parquet 文件类型。
- 需要广播的表是分区表，更新表数据后。

参考信息

被广播的表执行超时，导致任务结束。

默认情况下，BroadCastJoin 只允许被广播的表计算 5 分钟，超过 5 分钟该任务会出现超时异常，而这个时候被广播的表的 broadcast 任务依然在执行，造成资源浪费。

这种情况下，有两种方式处理：

- 调整“spark.sql.broadcastTimeout”的数值，加大超时的时间限制。
- 降低“spark.sql.autoBroadcastJoinThreshold”的数值，不使用 BroadCastJoin 的优化。

25.7.2.2 优化数据倾斜场景下的 Spark SQL 性能

配置场景

在 Spark SQL 多表 Join 的场景下，会存在关联键严重倾斜的情况，导致 Hash 分桶后，部分桶中的数据远高于其它分桶。最终导致部分 Task 过重，跑得很慢；其它 Task 过轻，跑得很快。一方面，数据量大 Task 运行慢，使得计算性能低；另一方面，数据量少的 Task 在运行完成后，导致很多 CPU 空闲，造成 CPU 资源浪费。

通过如下配置项可开启自动进行数据倾斜处理功能，通过将 Hash 分桶后数据量很大的、且超过数据倾斜阈值的分桶拆散，变成多个 task 处理一个桶的数据机制，提高 CPU 资源利用率，提高系统性能。

说明

未产生倾斜的数据，将采用原有方式进行分桶并运行。

使用约束：

- 只支持两表 Join 的场景。
- 不支持 FULL OUTER JOIN 的数据倾斜处理。

  示例：执行下面 SQL 语句，a 表倾斜或 b 表倾斜都无法触发该优化。

  select aid FROM a FULL OUTER JOIN b ON aid=bid;
- 不支持 LEFT OUTER JOIN 的右表倾斜处理。

  示例：执行下面 SQL 语句，b 表倾斜无法触发该优化。

  select aid FROM a LEFT OUTER JOIN b ON aid=bid;
- 不支持 RIGHT OUTER JOIN 的左表倾斜处理。

  示例：执行下面 SQL 语句，a 表倾斜无法触发该优化。

  select aid FROM a RIGHT OUTER JOIN b ON aid=bid;

配置描述

在 Spark Driver 端的“spark-defaults.conf”配置文件中添加如下表格中的参数。

表25-69 参数说明

参数	描述	默认值
spark.sql.adaptive.enabled	自适应执行特性的总开关。 注意：AQE 特性与 DPP（动态分区裁剪）特性同时开启时，SparkSQL 任务执行中会优先执行 DPP 特性，从而使得 AQE 特性不生效。集群中 DPP 特性是默认开启的，因此我们开启 AQE 特性的同时，需要将 DPP 特性关闭。	false
spark.sql.optimizer.dynamicPartitionPruning.enabled	动态分区裁剪功能的开关。	true
spark.sql.adaptive.skewJoin.enabled	当此配置为 true 且 spark.sql.adaptive.enabled 设置为 true 时，启用运行时自动处理 join 运算中的数据倾斜功能。	true
spark.sql.adaptive.skewJoin.skewedPartitionFactor	此配置为一个倍数因子，用于判定分区是否为数据倾斜分区。单个分区被判定为数据倾斜分区的条件为：当一个分区的数据大小超过除此分区外其他所有分区大小的中值与该配置的乘积，并且大小超过 spark.sql.adaptive.skewJoin.skewedPartitionThresholdInBytes 配置值时，此分区被判定为数据倾斜分区	5
spark.sql.adaptive.skewjoin.skewedPartitionThresholdInBytes	分区大小（单位：字节）大于该阈值且大于 spark.sql.adaptive.skewJoin.skewedPartitionFactor 与分区中值的乘积，则认为该分区存在倾斜。理想情况下，此配置应大于 spark.sql.adaptive.advisoryPartitionSizeInBytes.	256MB
spark.sql.adaptive.shuffle.targetPostShuffleInputSize	每个 task 处理的 shuffle 数据的最小数据量。单位：Byte。	67108864

25.7.2.3 优化小文件场景下的 Spark SQL 性能

配置场景

Spark SQL 的表中，经常会存在很多小文件（大小远小于 HDFS 块大小），每个小文件默认对应 Spark 中的一个 Partition，也就是一个 Task。在很多小文件场景下，Spark 会起很多 Task。当 SQL 逻辑中存在 Shuffle 操作时，会大大增加 hash 分桶数，严重影响性能。

在小文件场景下，您可以通过如下配置手动指定每个 Task 的数据量（Split Size），确保不会产生过多的 Task，提高性能。

说明

当 SQL 逻辑中不包含 Shuffle 操作时，设置此配置项，不会有明显的性能提升。

配置描述

要启动小文件优化，在 Spark 客户端的“spark-defaults.conf”配置文件中进行设置。

表25-70 参数说明

参数	描述	默认值
spark.sql.files.maxPartitionBytes	在读取文件时，将单个分区打包的最大字节数。 单位：byte。	134217728（即128M）
spark.files.openCostInBytes	打开文件的预估成本， 按照同一时间能够扫描的字节数来测量。当一个分区写入多个文件时使用。高估更好，这样小文件分区将比大文件分区更先被调度。	4M

25.7.2.4 INSERT...SELECT 操作调优

操作场景

在以下几种情况下，执行 INSERT...SELECT 操作可以进行一定的调优操作。

- 查询的数据是大量的小文件。
- 查询的数据是较多的大文件。
- 在 Beeline/JDBCServer 模式下使用非 Spark 用户操作。

操作步骤

可对 INSERT...SELECT 操作做如下的调优操作。

- 如果建的是 Hive 表，将存储类型设为 Parquet，从而减少执行 INSERT...SELECT 语句的时间。
- 建议使用 spark-sql 或者在 Beeline/JDBCServer 模式下使用 spark 用户来执行 INSERT...SELECT 操作，避免执行更改文件 owner 的操作，从而减少执行 INSERT...SELECT 语句的时间。

说明

在 Beeline/JDBCServer 模式下，executor 的用户跟 driver 是一致的，driver 是 JDBCServer 服务的一部分，是由 spark 用户启动的，因此其用户也是 spark 用户，且当前无法实现在运行时将 Beeline 端的用户透传到 executor，因此使用非 spark 用户时需要对文件进行更改 owner 为 Beeline 端的用户，即实际用户。

- 如果查询的数据是大量的小文件将会产生大量 map 操作，从而导致输出存在大量的小文件，在执行重命名文件操作时将会耗费较多时间，此时可以通过设置

“spark.sql.files.maxPartitionBytes”与“spark.files.openCostInBytes”来设置一个partiton 读取的最大字节，在一个 partition 中合并多个小文件来减少输出文件数及执行重命名文件操作的时间，从而减少执行 INSERT...SELECT 语句的时间。

说明

上述优化操作并不能解决全部的性能问题，对于以下场景仍然需要较多时间：

对于动态分区表，如果其分区数非常多，那么也需要执行较长的时间。

25.7.2.5 多并发 JDBC 客户端连接 JDBCServer

操作场景

JDBCServer 支持多用户多并发接入，但当并发任务数量较高的时候，默认的 JDBCServer 配置将无法支持，因此需要进行优化来支持该场景。

操作步骤

1. 设置 JDBCServer 的公平调度策略。

   Spark 默认使用 FIFO（First In First Out）的调度策略，但对于多并发的场景，使用 FIFO 策略容易导致短任务执行失败。因此在多并发的场景下，需要使用公平调度策略，防止任务执行失败。

   a. 在 Spark 中设置公平调度，具体请参考 http://spark.apache.org/docs/3.1.1/job-scheduling.html#scheduling-within-an-application

   b. 在 JDBC 客户端中设置公平调度。

      i. 在 BeeLine 命令行客户端或者 JDBC 自定义代码中，执行以下语句，其中 PoolName 是公平调度的某一个调度池。

      ```
      SET spark.sql.thriftserver.scheduler.pool=PoolName;
      ```

      ii. 执行相应的 SQL 命令，Spark 任务将会在上面的调度池中运行。

2. 设置 BroadCastHashJoin 的超时时间。

   BroadCastHashJoin 有超时参数，一旦超过预设的时间，该查询任务直接失败，在多并发场景下，由于计算任务抢占资源，可能会导致 BroadCastHashJoin 的 Spark 任务无法执行，导致超时出现。因此需要在 JDBCServer 的“spark-defaults.conf”配置文件中调整超时时间。

表25-71 参数描述

参数	描述	默认值
spark.sql.broadcastTimeout	BroadcastHashJoin 中广播表的超时时间，当任务并发数较高的时候，可以调高该参数值。	-1（数值类型，实际为五分钟）

25.7.2.6 动态分区插入场景内存优化

操作场景

SparkSQL 在往动态分区表中插入数据时，分区数越多，单个 Task 生成的 HDFS 文件越多，则元数据占用的内存也越多。这就导致程序 GC（Gabage Collection）严重，甚至发生 OOM（Out of Memory）。

经测试证明：10240 个 Task，2000 个分区，在执行 HDFS 文件从临时目录 rename 到目标目录动作前，FileStatus 元数据大小约 29G。为避免以上问题，可修改 SQL 语句对数据进行重分区，以减少 HDFS 文件个数。

操作步骤

在动态分区语句中加入 **distribute by**，by 值为分区字段。

示例如下：

insert into table store_returns partition (sr_returned_date_sk) select sr_return_time_sk,sr_item_sk,sr_customer_sk,sr_cdemo_sk,sr_hdemo_sk,sr_addr_sk,sr_store_sk,sr_reason_sk,sr_ticket_number,sr_return_quantity,sr_return_amt,sr_return_tax,sr_return_amt_inc_tax,sr_fee,sr_return_ship_cost,sr_refunded_cash,sr_reversed_charge,sr_store_credit,sr_net_loss,sr_returned_date_sk from ${SOURCE}.store_returns **distribute by sr_returned_date_sk**;

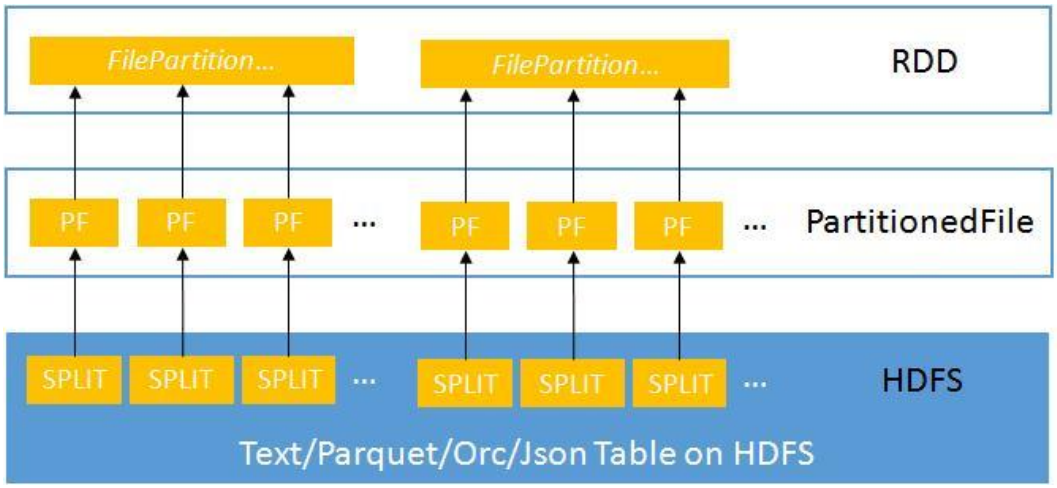
25.7.2.7 小文件优化

操作场景

Spark SQL 表中，经常会存在很多小文件（大小远小于 HDFS 的块大小），每个小文件默认对应 Spark 中的一个 Partition，即一个 Task。在有很多小文件时，Spark 会启动很多 Task，此时当 SQL 逻辑中存在 Shuffle 操作时，会大大增加 hash 分桶数，严重影响系统性能。

针对小文件很多的场景，DataSource 在创建 RDD 时，先将 Table 中的 split 生成 PartitionedFile，再将这些 PartitionedFile 进行合并。即将多个 PartitionedFile 组成一个 partition，从而减少 partition 数量，避免在 Shuffle 操作时生成过多的 hash 分桶，如图 25-8 所示。

图25-8 小文件合并



操作步骤

要启动小文件优化，在 Spark 客户端的“spark-defaults.conf”配置文件中进行设置。

表25-72 参数介绍

参数	描述	默认值
spark.sql.files.maxPartitionBytes	在读取文件时，将单个分区打包的最大字节数。 单位：byte。	134217728（即128M）
spark.files.openCostInBytes	打开文件的预估成本， 按照同一时间能够扫描的字节数来测量。当一个分区写入多个文件时使用。高估更好，这样小文件分区将比大文件分区更先被调度。	4M

25.7.2.8 聚合算法优化

操作场景

在 Spark SQL 中支持基于行的哈希聚合算法，即使用快速聚合 hashmap 作为缓存，以提高聚合性能。hashmap 替代了之前的 ColumnarBatch 支持，从而避免拥有聚合表的宽模式（大量 key 字段或 value 字段）时产生的性能问题。

操作步骤

要启动聚合算法优化，在 Spark 客户端的“spark-defaults.conf”配置文件中进行设置。

表25-73 参数介绍

参数	描述	默认值
spark.sql.codegen.aggregate.map.twolevel.enabled	是否开启聚合算法优化： • true：开启 • false：不开启	true

25.7.2.9 Datasource 表优化

操作场景

将 datasource 表的分区消息存储到 Metastore 中，并在 Metastore 中对分区消息进行处理。

- 优化 datasource 表，支持对表中分区执行增加、删除和修改等语法，从而增加与 Hive 的兼容性。
- 支持在查询语句中，把分区裁剪并下压到 Metastore 上，从而过滤掉不匹配的分区。

  示例如下：

  ```
  select count(*) from table where partCol=1;    //partCol 列为分区列
  ```

  此时，在物理计划中执行 TableScan 操作时，只处理分区(partCol=1)对应的数据。

操作步骤

要启动 Datasource 表优化，在 Spark 客户端的“spark-defaults.conf”配置文件中进行设置。

表25-74 参数介绍

参数	描述	默认值
spark.sql.hive.manageFilesourcePartitions	是否启用 Metastore 分区管理（包括数据源表和转换的 Hive 表）。 • true：启用 Metastore 分区管理，即数据源表存储分区在 Hive 中，并在查询语句中使用 Metastore 修剪分区。 • false：不启用 Metastore 分区管理。	true
spark.sql.hive.metastorePartitionPruning	是否支持将 predicate 下压到 Hive Metastore 中。 • true：支持，目前仅支持 Hive 表的 predicate 下压。 • false：不支持	true
spark.sql.hive.filesourcePartitio	启用内存中分区文件元数据的缓存大小。	250 * 1024 *

参数	描述	默认值
nFileCacheSize	所有表共享一个可以使用指定的 num 字节进行文件元数据的缓存。 只有当“spark.sql.hive.manageFilesourcePartitions”配置为“true”时，该配置项才会生效。	1024
spark.sql.hive.convertMetastoreOrc	设置 ORC 表的处理方式： • false：Spark SQL 使用 Hive SerDe 处理 ORC 表。 • true：Spark SQL 使用 Spark 内置的机制处理 ORC 表。	true

25.7.2.10 合并 CBO 优化

操作场景

Spark SQL 默认支持基于规则的优化，但仅仅基于规则优化不能保证 Spark 选择最优的查询计划。CBO（Cost-Bsed Optimizer）是一种为 SQL 智能选择查询计划的技术。通过配置开启 CBO 后，CBO 优化器可以基于表和列的统计信息，进行一系列的估算，最终选择出最优的查询计划。

操作步骤

要使用 CBO 优化，可以按照以下步骤进行优化。

1. 需要先执行特定的 SQL 语句来收集所需的表和列的统计信息。

   SQL 命令如下（根据具体情况选择需要执行的 SQL 命令）：

   - 生成表级别统计信息（扫表）：

     ANALYZE TABLE src COMPUTE STATISTICS

     生成 sizeInBytes 和 rowCount。

     使用 ANALYZE 语句收集统计信息时，无法计算非 HDFS 数据源的表的文件大小。

   - 生成表级别统计信息（不扫表）：

     ANALYZE TABLE src COMPUTE STATISTICS NOSCAN

     只生成 sizeInBytes，如果原来已经生成过 sizeInBytes 和 rowCount，而本次生成的 sizeInBytes 和原来的大小一样，则保留 rowCount（若存在），否则清除 rowCount。

   - 生成列级别统计信息

     ANALYZE TABLE src COMPUTE STATISTICS FOR COLUMNS a, b, c

     生成列统计信息，为保证一致性，会同步更新表统计信息。目前不支持复杂数据类型（如 Seq, Map 等）和 HiveStringType 的统计信息生成。

- 显示统计信息

  DESC FORMATTED src

  在 Statistics 中会显示“xxx bytes, xxx rows”分别表示表级别的统计信息。也可以通过如下命令显示列统计信息：

  DESC FORMATTED src a

使用限制：当前统计信息收集不支持针对分区表的分区级别的统计信息。

2. 在 Spark 客户端的“spark-defaults.conf”配置文件中进行表 25-75 设置。

表25-75 参数介绍

参数	描述	默认值
spark.sql.cbo.enabled	CBO 总开关。 • true 表示打开， • false 表示关闭。 要使用该功能，需确保相关表和列的统计信息已经生成。	false
spark.sql.cbo.joinReorder.enabled	使用 CBO 来自动调整连续的 inner join 的顺序。 • true：表示打开 • false：表示关闭 要使用该功能，需确保相关表和列的统计信息已经生成，且 CBO 总开关打开。	false
spark.sql.cbo.joinReorder.dp.threshold	使用 CBO 来自动调整连续 inner join 的表的个数阈值。 如果超出该阈值，则不会调整 join 顺序。	12

25.7.2.11 跨源复杂数据的 SQL 查询优化

操作场景

本章节介绍如何打开或关闭跨源复杂数据的 SQL 查询优化功能。

操作步骤

- （可选）连接 MPPDB 数据源的准备

  如果连接的数据源为 MPPDB，由于 MPPDB Driver 文件“gsjdbc4.jar”和 Spark 中的 jar 包“gsjdbc4-VXXXRXXXCXXSPCXXX.jar”包含了相同的类名，存在类名冲突的问题。因此在连接 MPPDB 数据库之前，需要执行以下步骤：

  a. 移除 Spark 中的“gsjdbc4-VXXXRXXXCXXSPCXXX.jar”，由于 Spark 运行不依赖该 jar 包，因此将该 jar 包移动到其他目录（例如，移动到“/tmp”目录，不建议直接删除）不会影响 Spark 正常运行。

i. 登录 Spark 服务端主机，移除“${BIGDATA_HOME}/FusionInsight_Spark2x_8.1.0.1/install/FusionInsight-Spark2x-3.1.1/spark/jars”路径下的“gsjdbc4-VXXXRXXXCXXSPCXXX.jar”。

ii. 登录 Spark 客户端主机，移除“/opt/client/Spark2x/spark/jars”路径下的“gsjdbc4-VXXXRXXXCXXSPCXXX.jar”。

b. 在 MPPDB 的安装包中获取 MPPDB Driver 文件“gsjdbc4.jar”，并将该文件分别上传到以下位置：

说明

“gsjdbc4.jar”在 MPPDB 安装包的目录“FusionInsight_MPPDB\software\components\package\FusionInsight-MPPDB-*xxx*\package\Gauss-MPPDB-ALL-PACKAGES\GaussDB-*xxx*-REDHAT-*xxx*-Jdbc\jdbc”中获取。

- Spark 服务端的“/${BIGDATA_HOME}/FusionInsight_Spark2x_8.1.0.1/install/FusionInsight-Spark2x-3.1.1/spark/jars”路径下。
- Spark 客户端的“/opt/client/Spark2x/spark/jars”路径下。

c. 更新存储在 HDFS 中的“/user/spark2x/jars/8.1.0.1/spark-archive-2x.zip”压缩包。

说明

此处版本号 8.1.0.1 为示例，具体以实际环境的版本号为准。

i. 使用客户端安装用户登录客户端所在节点。执行命令切换到客户端安装目录，例如“/opt/client”。

cd /opt/client

ii. 执行以下命令配置环境变量。

source bigdata_env

iii. 如果集群为安全模式，执行以下命令获得认证。

kinit *组件业务用户*

iv. 新建临时文件./tmp，并从 HDFS 获取“spark-archive-2x.zip”并解压到 tmp 目录，命令如下：

mkdir tmp

hdfs dfs -get /user/spark2x/jars/8.1.0.1/spark-archive-2x.zip **./**

unzip spark-archive-2x.zip -d ./tmp

v. 切换到 tmp 目录，删除“gsjdbc4-VXXXRXXXCXXSPCXXX.jar”文件，并将 MPPDB Driver 文件“gsjdbc4.jar”上传到 tmp 目录中，然后执行以下命令重新打包。

zip -r spark-archive-2x.zip *.jar

vi. 删除 HDFS 上的“spark-archive-2x.zip”，将步骤 c.v 中新生成的压缩包“spark-archive-2x.zip”更新至 HDFS 的“/user/spark2x/jars/8.1.0.1/”路径下。

hdfs dfs -rm /user/spark2x/jars/8.1.0.1/spark-archive-2x.zip

hdfs dfs -put ./spark-archive-2x.zip /user/spark2x/jars/8.1.0.1

d. 重启 Spark 服务，等重启成功后，重新启动 Spark 客户端。

- 打开优化开关

  对所有支持查询下推的模块，可以通过在 spark-beeline 客户端中执行 SET 命令打开跨源查询优化功能，默认均为关闭状态。

  可以从全局、数据源、表这三个维度进行下推开关控制。打开方法如下：

  - 全局（对所有数据源生效）：

    SET spark.sql.datasource.jdbc = project,aggregate,orderby-limit

  - 数据源：

    SET spark.sql.datasource.${url} = project,aggregate,orderby-limit

  - 表：

    SET spark.sql.datasource.${url}.${table} = project,aggregate,orderby-limit

  执行 SET 命令设置上述参数时，允许一次设置多个下推模块，中间以逗号分隔。各个下推模块对应的参数值如下所示：

表25-76 各模块对应的参数值

模块名称	SET 命令的参数值
project	project
aggregate	aggregate
order by，limit over project or aggregate	orderby-limit

  示例：创建一个 MySQL 的外表的语句为：

  create table if not exists pdmysql using org.apache.spark.sql.jdbc options(driver "com.mysql.jdbc.Driver", url "jdbc:mysql://ip2:3306/test", user "hive", password "*xxx*", dbtable "mysqldata");

  则其中：

  - ${url} = jdbc:mysql://ip2:3306/test
  - ${table} = mysqldata

  说明

  - “=”后即设置可以下推打开的算子，以“,”隔开。
  - 优先级：table 开关>数据源开关>全局开关。即若设置了 table 开关，则数据源开关全局开关对该表失效；若配置了数据源开关，则全局开关对该数据源失效。
  - url 中不能包含“=”，若包含，set 时直接删掉“=”。
  - 可多次执行 set，key 不同不会相互覆盖。

- 新增支持查询下推的函数

  除了支持对 abs()、month()、length()等数学、时间、字符串函数进行查询下推外，用户还可以通过 SET 命令新增数据源支持查询下推的函数。在 spark-beeline 客户端中执行如下命令：

  SET spark.sql.datasource.${datasource}.functions = fun1,fun2

- 取消开关设置及取消新增的下推函数

  当前只能通过在 spark-beeline 客户端中执行 ***RESET*** 命令取消所有 SET 的内容。由于执行 ***RESET*** 后所有 SET 的参数值都将被清除，请谨慎使用。

  控制开关的设置仅在客户端当前的会话中生效，当客户端关闭后，SET 内容就失效了。

  或者修改客户端配置文件 spark-defaults.conf 中的 spark.sql.locale.support 参数为 true。

- 新增支持中文和葡萄牙文下推（仅限 Hive 数据源）

  当前 Spark 和 Hive 的表名和字段名支持中文和葡萄牙文，所以用户可以通过 SET 命令新增数据源支持查询中文和葡萄牙文下推。在 spark-sql 客户端中执行如下命令：

  set　spark.sql.locale.support =　true

  或者修改客户端配置文件 spark-defaults.conf 中的 spark.sql.locale.support 参数为 true。

注意事项

数据源只支持 MySQL 和 MPPDB，Hive，oracle， postgresql。

25.7.2.12 多级嵌套子查询以及混合 Join 的 SQL 调优

操作场景

本章节介绍在多级嵌套以及混合 Join SQL 查询的调优建议。

前提条件

例如有一个复杂的查询样例如下：

```
select
s_name,
count(1) as numwait
from (
select s_name from (
select
s_name,
t2.l_orderkey,
l_suppkey,
count_suppkey,
max_suppkey
from
test2 t2 right outer join (
select
s_name,
l_orderkey,
l_suppkey from (
select
s_name,
t1.l_orderkey,
l_suppkey,
```

```
count_suppkey,
max_suppkey
from
test1 t1 join (
select
s_name,
l_orderkey,
l_suppkey
from
orders o join (
select
s_name,
l_orderkey,
l_suppkey
from
nation n join supplier s
on
s.s_nationkey = n.n_nationkey
and n.n_name = 'SAUDI ARABIA'
join lineitem l
on
s.s_suppkey = l.l_suppkey
where
l.l receiptdate > l.l commitdate
and l.l orderkey is not null
) l1 on o.o orderkey = l1.l orderkey and o.o orderstatus = 'F'
) l2 on l2.l orderkey = t1.l orderkey
) a
where
(count_suppkey > 1)
or ((count_suppkey=1)
and (l_suppkey <> max_suppkey))
) l3 on l3.l_orderkey = t2.l_orderkey
) b
where
(count_suppkey is null)
or ((count_suppkey=1)
and (l_suppkey = max_suppkey))
) c
group by
s_name
order by
numwait desc,
s_name
limit 100;
```

操作步骤

步骤 1 分析业务。

从业务入手分析是否可以简化 SQL，例如可以通过合并表去减少嵌套的层级和 Join 的次数。

步骤 2 如果业务需求对应的 SQL 无法简化，则需要配置 DRIVER 内存：

- 使用 **spark-submit** 或者 **spark-sql** 运行 SQL 语句，执行步骤 3。
- 使用 **spark-beeline** 运行 SQL 语句，执行步骤 4。

步骤 3 执行 SQL 语句时，需要添加参数“--driver-memory”，设置内存大小，例如：

/spark-sql --master=local[4] --driver-memory=512M -f /tpch.sql

步骤 4 在执行 SQL 语句前，请使用管理员用户修改内存大小配置。

1. 登录 FusionInsight Manager，选择“集群 > *待操作集群的名称* > 服务 > Spark2x > 配置”。
2. 单击“全部配置”，并搜索“SPARK_DRIVER_MEMORY”。
3. 修改参数值适当增加内存大小。仅支持整数值，且需要输入单位 M 或者 G。例如输入 512M。

----结束

参考信息

DRIVER 内存不足时，查询操作可能遇到以下错误提示信息：

```
2018-02-11 09:13:14,683 | WARN  | Executor task launch worker for task 5 | Calling
spill() on RowBasedKeyValueBatch. Will not spill but return 0. |
org.apache.spark.sql.catalyst.expressions.RowBasedKeyValueBatch.spill(RowBasedKeyVa
lueBatch.java:173)
2018-02-11 09:13:14,682 | WARN  | Executor task launch worker for task 3 | Calling
spill() on RowBasedKeyValueBatch. Will not spill but return 0. |
org.apache.spark.sql.catalyst.expressions.RowBasedKeyValueBatch.spill(RowBasedKeyVa
lueBatch.java:173)
2018-02-11 09:13:14,704 | ERROR | Executor task launch worker for task 2 |
Exception in task 2.0 in stage 1.0 (TID 2) |
org.apache.spark.internal.Logging$class.logError(Logging.scala:91)
java.lang.OutOfMemoryError: Unable to acquire 262144 bytes of memory, got 0
       at
org.apache.spark.memory.MemoryConsumer.allocateArray(MemoryConsumer.java:100)
       at
org.apache.spark.unsafe.map.BytesToBytesMap.allocate(BytesToBytesMap.java:791)
       at
org.apache.spark.unsafe.map.BytesToBytesMap.<init>(BytesToBytesMap.java:208)
       at
org.apache.spark.unsafe.map.BytesToBytesMap.<init>(BytesToBytesMap.java:223)
       at
org.apache.spark.sql.execution.UnsafeFixedWidthAggregationMap.<init>(UnsafeFixedWid
thAggregationMap.java:104)
       at
org.apache.spark.sql.execution.aggregate.HashAggregateExec.createHashMap(HashAggreg
ateExec.scala:307)
       at
org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIterator.agg_doAg
gregateWithKeys$(Unknown Source)
       at
org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIterator.processN
ext(Unknown Source)
       at
```

```
org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java
:43)
       at
org.apache.spark.sql.execution.WholeStageCodegenExec$$anonfun$8$$anon$1.hasNext(Who
leStageCodegenExec.scala:381)
       at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:408)
       at
org.apache.spark.shuffle.sort.BypassMergeSortShuffleWriter.write(BypassMergeSortShu
ffleWriter.java:126)
       at  org.apache.spark.scheduler.ShuffleMapTask.runTask(ShuffleMapTask.scala:96)
       at  org.apache.spark.scheduler.ShuffleMapTask.runTask(ShuffleMapTask.scala:53)
       at org.apache.spark.scheduler.Task.run(Task.scala:99)
       at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:325)
       at
java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
       at
java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
       at java.lang.Thread.run(Thread.java:748)
```

25.7.3 Spark Streaming 调优

操作场景

Streaming 作为一种 mini-batch 方式的流式处理框架，它主要的特点是：秒级时延和高吞吐量。因此 Streaming 调优的目标：在秒级延迟的情景下，提高 Streaming 的吞吐能力，在单位时间处理尽可能多的数据。

说明

本章节适用于输入数据源为 Kafka 的使用场景。

操作步骤

一个简单的流处理系统由以下三部分组件组成：数据源 + 接收器 + 处理器。数据源为 Kafka，接受器为 Streaming 中的 Kafka 数据源接收器，处理器为 Streaming。

对 Streaming 调优，就必须使该三个部件的性能都最优化。

- **数据源调优**

  在实际的应用场景中，数据源为了保证数据的容错性，会将数据保存在本地磁盘中，而 Streaming 的计算结果全部在内存中完成，数据源很有可能成为流式系统的最大瓶颈点。

  对 Kafka 的性能调优，有以下几个点：

  - 使用 Kafka-0.8.2 以后版本，可以使用异步模式的新 Producer 接口。
  - 配置多个 Broker 的目录，设置多个 IO 线程，配置 Topic 合理的 Partition 个数。

  详情请参见 Kafka 开源文档中的“性能调优”部分：
  http://kafka.apache.org/documentation.html

- **接收器调优**

Streaming 中已有多种数据源的接收器，例如 Kafka、Flume、MQTT、ZeroMQ 等，其中 Kafka 的接收器类型最多，也是最成熟一套接收器。

Kafka 包括三种模式的接收器 API：

- KafkaReceiver：直接接收 Kafka 数据，进程异常后，可能出现数据丢失。
- ReliableKafkaReceiver：通过 ZooKeeper 记录接收数据位移。
- DirectKafka：直接通过 RDD 读取 Kafka 每个 Partition 中的数据，数据高可靠。

从实现上来看，DirectKafka 的性能会是最好的，实际测试上来看，DirectKafka 也确实比其他两个 API 性能好了不少。因此推荐使用 DirectKafka 的 API 实现接收器。

数据接收器作为一个 Kafka 的消费者，对于它的配置优化，请参见 Kafka 开源文档：http://kafka.apache.org/documentation.html

- **处理器调优**

  Spark Streaming 的底层由 Spark 执行，因此大部分对于 Spark 的调优措施，都可以应用在 Spark Streaming 之中，例如：

  - 数据序列化
  - 配置内存
  - 设置并行度
  - 使用 External Shuffle Service 提升性能

  说明

  在做 Spark Streaming 的性能优化时需注意一点，越追求性能上的优化，Spark Streaming 整体的可靠性会越差。例如：

  “spark.streaming.receiver.writeAheadLog.enable”配置为“false”的时候，会明显减少磁盘的操作，提高性能，但由于缺少 WAL 机制，会出现异常恢复时，数据丢失。

  因此，在调优 Spark Streaming 的时候，这些保证数据可靠性的配置项，在生产环境中是不能关闭的。

- **日志归档调优**

  参数“spark.eventLog.group.size”用来设置一个应用的 JobHistory 日志按照指定 job 个数分组，每个分组会单独创建一个文件记录日志，从而避免应用长期运行时形成单个过大日志造成 JobHistory 无法读取的问题，设置为“0”时表示不分组。

  大部分 Spark Streaming 任务属于小型 job，而且产生速度较快，会导致频繁的分组，产生大量日志小文件消耗磁盘 I/O。建议增大此值，例如改为“1000”或更大值。

25.8 Spark2x 常见问题

25.8.1 Spark Core

25.8.1.1 日志聚合下，如何查看 Spark 已完成应用日志

问题

当 YARN 开启了日志聚合功能时，如何在页面看到聚合后的 container 日志？

回答

请参考配置 WebUI 上查看聚合后的 container 日志。

25.8.1.2 Driver 返回码和 RM WebUI 上应用状态显示不一致

问题

ApplicationMaster 与 ResourceManager 之间通信发生长时间异常时，为什么 Driver 返回码和 RM WebUI 上应用状态显示不一致？

回答

在 yarn-client 模式下，Spark 的 Driver 和 ApplicationMaster 作为两个独立的进程在运行。当 Driver 完成任务退出时，会通知 ApplicationMaster 向 ResourceManager 注销自身，即调用 unregister 方法。

由于是远程调用，则存在发生网络故障的可能性。当发生网络故障时，ApplicationMaster 会使用 Yarn 客户端的重试机制进行重试。在达到最大重试次数之前网络恢复正常，则 ApplicationMaster 会正常退出。

若超过重试次数和重试时长，则 ApplicationMaster 注销失败，ResourceManager 会认为 ApplicationMaster 异常退出并尝试重新启动 ApplicationMaster。新启动的 ApplicationMaster 在尝试连接已经退出的 Driver 失败后，会在 ResourceManager 页面上标记此次 Application 为 FAILED 状态。

这种情况为小概率事件且不影响 Spark SQL 对外展现的应用完成状态。也可以通过增大 Yarn 客户端连接次数和连接时长的方式减少此事件发生的概率。配置详情请参见：http://hadoop.apache.org/docs/r3.1.1/hadoop-yarn/hadoop-yarn-common/yarn-default.xml

25.8.1.3 为什么 Driver 进程不能退出

问题

运行 Spark Streaming 任务，然后使用 **yarn application -kill applicationID** 命令停止任务，为什么 Driver 进程不能退出？

回答

使用 **yarn application -kill applicationID** 命令后 Spark 只会停掉任务对应的 SparkContext，而不是退出当前进程。如果当前进程中存在其他常驻的线程（类似 spark-shell 需要不断检测命令输入，Spark Streaming 不断在从数据源读取数据），SparkContext 被停止并不会终止整个进程。

如果需要退出 Driver 进程，建议使用 **kill -9 pid** 命令手动退出当前 Driver。

25.8.1.4 网络连接超时导致 FetchFailedException

问题

在 380 节点的大集群上，运行 29T 数据量的 HiBench 测试套中 ScalaSort 测试用例，使用以下关键配置（**--executor-cores 4**）出现如下异常：

```
org.apache.spark.shuffle.FetchFailedException: Failed to connect to
/192.168.114.12:23242
    at
org.apache.spark.storage.ShuffleBlockFetcherIterator.throwFetchFailedException(Shuf
fleBlockFetcherIterator.scala:321)
    at
org.apache.spark.storage.ShuffleBlockFetcherIterator.next(ShuffleBlockFetcherIterat
or.scala:306)
    at
org.apache.spark.storage.ShuffleBlockFetcherIterator.next(ShuffleBlockFetcherIterat
or.scala:51)
    at scala.collection.Iterator$$anon$11.next(Iterator.scala:328)
    at scala.collection.Iterator$$anon$13.hasNext(Iterator.scala:371)
    at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:327)
    at org.apache.spark.util.CompletionIterator.hasNext(CompletionIterator.scala:32)
    at  org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:39)
    at
org.apache.spark.util.collection.ExternalSorter.insertAll(ExternalSorter.scala:217)
    at
org.apache.spark.shuffle.hash.HashShuffleReader.read(HashShuffleReader.scala:102)
    at org.apache.spark.rdd.ShuffledRDD.compute(ShuffledRDD.scala:90)
    at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:301)
    at org.apache.spark.rdd.RDD.iterator(RDD.scala:265)
    at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:38)
    at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:301)
    at org.apache.spark.rdd.RDD.iterator(RDD.scala:265)
    at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:38)
    at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:301)
    at org.apache.spark.rdd.RDD.iterator(RDD.scala:265)
    at org.apache.spark.rdd.UnionRDD.compute(UnionRDD.scala:87)
    at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:301)
    at org.apache.spark.rdd.RDD.iterator(RDD.scala:265)
    at org.apache.spark.scheduler.ShuffleMapTask.runTask(ShuffleMapTask.scala:73)
    at org.apache.spark.scheduler.ShuffleMapTask.runTask(ShuffleMapTask.scala:41)
    at org.apache.spark.scheduler.Task.run(Task.scala:87)
    at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:213)
    at
java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
```

```
    at
java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
    at java.lang.Thread.run(Thread.java:745)
Caused by: java.io.IOException: Failed to connect to /192.168.114.12:23242
    at
org.apache.spark.network.client.TransportClientFactory.createClient(TransportClient
Factory.java:214)
    at
org.apache.spark.network.client.TransportClientFactory.createClient(TransportClient
Factory.java:167)
    at
org.apache.spark.network.netty.NettyBlockTransferService$$anon$1.createAndStart(Net
tyBlockTransferService.scala:91)
    at
org.apache.spark.network.shuffle.RetryingBlockFetcher.fetchAllOutstanding(RetryingB
lockFetcher.java:140)
    at
org.apache.spark.network.shuffle.RetryingBlockFetcher.access$200(RetryingBlockFetch
er.java:43)
    at
org.apache.spark.network.shuffle.RetryingBlockFetcher$1.run(RetryingBlockFetcher.ja
va:170)
    at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:511)
    at java.util.concurrent.FutureTask.run(FutureTask.java:266)
    ... 3 more
Caused by: java.net.ConnectException: Connection timed out: /192.168.114.12:23242
    at sun.nio.ch.SocketChannelImpl.checkConnect(Native Method)
    at sun.nio.ch.SocketChannelImpl.finishConnect(SocketChannelImpl.java:717)
    at
io.netty.channel.socket.nio.NioSocketChannel.doFinishConnect(NioSocketChannel.java:
224)
    at
io.netty.channel.nio.AbstractNioChannel$AbstractNioUnsafe.finishConnect(AbstractNio
Channel.java:289)
    at io.netty.channel.nio.NioEventLoop.processSelectedKey(NioEventLoop.java:528)
    at
io.netty.channel.nio.NioEventLoop.processSelectedKeysOptimized(NioEventLoop.java:46
8)
    at io.netty.channel.nio.NioEventLoop.processSelectedKeys(NioEventLoop.java:382)
    at io.netty.channel.nio.NioEventLoop.run(NioEventLoop.java:354)
    at
io.netty.util.concurrent.SingleThreadEventExecutor$2.run(SingleThreadEventExecutor.
java:111)
    ... 1 more
```

回答

在运行应用程序时，使用 Executor 参数“--executor-cores 4”，单进程中并行度高导致 IO 非常繁忙，以至于任务运行缓慢。

```
16/02/26 10:04:53 INFO TaskSetManager: Finished task 2139.0 in stage 1.0 (TID
151149) in 376455 ms on 10-196-115-2 (694/153378)
```

单个任务运行时间超过 6 分钟，从而导致连接超时问题，最终使得任务失败。

将参数中的核数设置为 1，“--executor-cores 1”，任务正常完成，单个任务处理时间在合理范围之内(15 秒左右)。

```
16/02/29 02:24:46 INFO TaskSetManager: Finished task 59564.0 in stage 1.0 (TID
208574) in 15088 ms on 10-196-115-6 (59515/153378)
```

因此，处理这类网络超时任务，可以减少单个 Executor 的核数来规避该类问题。

25.8.1.5 当事件队列溢出时如何配置事件队列的大小

问题

当 Driver 日志中出现如下的日志时，表示事件队列溢出了。当事件队列溢出时如何配置事件队列的大小？

- 普通应用

  ```
  Dropping SparkListenerEvent because no remaining room in event queue.
  This likely means one of the SparkListeners is too slow and cannot keep
  up with the rate at which tasks are being started by the scheduler.
  ```

- Spark Streaming 应用

  ```
  Dropping StreamingListenerEvent because no remaining room in event queue.
  This likely means one of the StreamingListeners is too slow and cannot keep
  up with the rate at which events are being started by the scheduler.
  ```

回答

1. 停止应用，在 Spark 的配置文件“spark-defaults.conf”中将配置项“spark.event.listener.logEnable”配置为“true”。并把配置项“spark.eventQueue.size”配置为 1000W。如果需要控制打印频率（默认为 1000 毫秒打印 1 条日志），请根据需要修改配置项“spark.event.listener.logRate”，该配置项的单位为毫秒。
2. 启动应用，可以发现如下的日志信息（消费者速率、生产者速率、当前队列中的消息数量和队列中消息数量的最大值）。

   ```
   INFO LiveListenerBus: [SparkListenerBus]:16044 events are consumed in 5000 ms.
   INFO LiveListenerBus: [SparkListenerBus]:51381 events are produced in 5000 ms,
   eventQueue still has 86417 events, MaxSize: 171764.
   ```

3. 用户可以根据日志信息【队列中消息数量的最大值 MaxSize】，在配置文件“spark-defaults.conf”中将配置项“spark.eventQueue.size”配置成合适的队列大小。比如【队列中消息数量的最大值】为 250000，那么配置合适的队列大小为 300000。

25.8.1.6 Spark 应用执行过程中，日志中一直打印 getApplicationReport 异常且应用较长时间不退出

问题

Spark 应用执行过程中，当 driver 连接 RM 失败时，会报下面的错误，且较长时间不退出。

```
16/04/23 15:31:44 INFO RetryInvocationHandler: Exception while invoking
getApplicationReport of class ApplicationClientProtocolPBClientImpl over 37 after 1
fail over attempts. Trying to fail over after sleeping for 44160ms.
java.net.ConnectException: Call From vm1/192.168.39.30 to vm1:8032 failed on
connection exception: java.net.ConnectException: Connection refused; For more
details see:  http://wiki.apache.org/hadoop/ConnectionRefused
```

回答

在 Spark 中有个定期线程，通过连接 RM 监听 AM 的状态。由于连接 RM 超时，就会报上面的错误，且一直重试。RM 中对重试次数有限制，默认是 30 次，每次间隔默认为 30 秒左右，每次重试时都会报上面的错误。超过次数后，driver 才会退出。

RM 中关于重试相关的配置项如表 25-77 所示。

表25-77 参数说明

参数	描述	默认值
yarn.resourcemanager.connect.max-wait.ms	连接 RM 的等待时间最大值。	900000
yarn.resourcemanager.connect.retry-interval.ms	重试连接 RM 的时间频率。	30000

重试次数=yarn.resourcemanager.connect.max-wait.ms/yarn.resourcemanager.connect.retry-interval.ms，即重试次数=连接 RM 的等待时间最大值/重试连接 RM 的时间频率。

在 Spark 客户端机器中，通过修改“conf/yarn-site.xml”文件，添加并配置“yarn.resourcemanager.connect.max-wait.ms”和“yarn.resourcemanager.connect.retry-interval.ms”，这样可以更改重试次数，Spark 应用可以提早退出。

25.8.1.7 Spark 执行应用时上报“Connection to ip:port has been quiet for xxx ms while there are outstanding requests”并导致应用结束

问题

Spark 执行应用时上报如下类似错误并导致应用结束。

```
2016-04-20 10:42:00,557 | ERROR | [shuffle-server-2] | Connection to 10-91-8-
208/10.18.0.115:57959 has been quiet for 180000 ms while there are outstanding
requests. Assuming connection is dead; please adju
st spark.network.timeout if this is wrong. |
org.apache.spark.network.server.TransportChannelHandler.userEventTriggered(Transpor
tChannelHandler.java:128)
2016-04-20 10:42:00,558 | ERROR | [shuffle-server-2] | Still have 1 requests
outstanding when connection from 10-91-8-208/10.18.0.115:57959 is closed |
org.apache.spark.network.client.TransportResponseHandl
er.channelUnregistered(TransportResponseHandler.java:102)
2016-04-20 10:42:00,562 | WARN  | [yarn-scheduler-ask-am-thread-pool-160] | Error
sending message [message = DoShuffleClean(application_1459995017785_0108,319)] in 1
attempts | org.apache.spark.Logging$clas
```

```
s.logWarning(Logging.scala:92)
java.io.IOException: Connection from 10-91-8-208/10.18.0.115:57959 closed
	at
org.apache.spark.network.client.TransportResponseHandler.channelUnregistered(TransportResponseHandler.java:104)
	at
org.apache.spark.network.server.TransportChannelHandler.channelUnregistered(TransportChannelHandler.java:94)
	at
io.netty.channel.AbstractChannelHandlerContext.invokeChannelUnregistered(AbstractChannelHandlerContext.java:158)
	at
io.netty.channel.AbstractChannelHandlerContext.fireChannelUnregistered(AbstractChannelHandlerContext.java:144)
	at
io.netty.channel.ChannelInboundHandlerAdapter.channelUnregistered(ChannelInboundHandlerAdapter.java:53)
	at
io.netty.channel.AbstractChannelHandlerContext.invokeChannelUnregistered(AbstractChannelHandlerContext.java:158)
	at
io.netty.channel.AbstractChannelHandlerContext.fireChannelUnregistered(AbstractChannelHandlerContext.java:144)
	at
io.netty.channel.ChannelInboundHandlerAdapter.channelUnregistered(ChannelInboundHandlerAdapter.java:53)
	at
io.netty.channel.AbstractChannelHandlerContext.invokeChannelUnregistered(AbstractChannelHandlerContext.java:158)
	at
io.netty.channel.AbstractChannelHandlerContext.fireChannelUnregistered(AbstractChannelHandlerContext.java:144)
	at
io.netty.channel.ChannelInboundHandlerAdapter.channelUnregistered(ChannelInboundHandlerAdapter.java:53)
	at
io.netty.channel.AbstractChannelHandlerContext.invokeChannelUnregistered(AbstractChannelHandlerContext.java:158)
	at
io.netty.channel.AbstractChannelHandlerContext.fireChannelUnregistered(AbstractChannelHandlerContext.java:144)
	at
io.netty.channel.DefaultChannelPipeline.fireChannelUnregistered(DefaultChannelPipeline.java:739)
	at
io.netty.channel.AbstractChannel$AbstractUnsafe$8.run(AbstractChannel.java:659)
	at
io.netty.util.concurrent.SingleThreadEventExecutor.runAllTasks(SingleThreadEventExecutor.java:357)
	at io.netty.channel.nio.NioEventLoop.run(NioEventLoop.java:357)
	at
io.netty.util.concurrent.SingleThreadEventExecutor$2.run(SingleThreadEventExecutor.java:111)
	at java.lang.Thread.run(Thread.java:745)
2016-04-20 10:42:00,573 | INFO  | [dispatcher-event-loop-14] | Starting task 177.0
```

```
in stage 1492.0 (TID 1996351, linux-254, PROCESS_LOCAL, 2106 bytes) |
org.apache.spark.Logging$class.logInfo(Logging.scala:
59)
2016-04-20 10:42:00,574 | INFO  | [task-result-getter-0] | Finished task 85.0 in
stage 1492.0 (TID 1996259) in 191336 ms on linux-254 (106/3000) |
org.apache.spark.Logging$class.logInfo(Logging.scala:59)
2016-04-20 10:42:00,811 | ERROR | [Yarn application state monitor] | Yarn
application has already exited with state FINISHED! |
org.apache.spark.Logging$class.logError(Logging.scala:75)
```

回答

当配置 channel 过期时间（spark.rpc.io.connectionTimeout） < RPC 响应超时时间（spark.rpc.askTimeout），在特殊条件下（Full GC，网络延时等）消息响应时间较长，消息还没有反馈，channel 又达到了过期时间，该 channel 就被终止了，AM 端感知到 channel 被终止后认为 driver 失联，然后整个应用停止。

解决办法：在 Spark 客户端的“spark-defaults.conf”文件中或通过 set 命令行进行设置。参数配置时要保证 channel 过期时间（spark.rpc.io.connectionTimeout）大于或等于 RPC 响应超时时间（spark.rpc.askTimeout）。

表25-78 参数说明

参数	描述	默认值
spark.rpc.askTimeout	RPC 响应超时时间，不配置的话默认使用 spark.network.timeout 的值。	120s

25.8.1.8 NodeManager 关闭导致 Executor(s)未移除

问题

在 Executor 动态分配打开的情况下，如果在任务执行过程中，执行 NodeManager 关闭动作，NodeManager 关闭节点上的 Executor(s)在空闲超时之后，在 driver 页面上未被移除。

回答

这是因为 ResourceManager 感知到 NodeManager 关闭时，Executor(s)已经因空闲超时而被 driver 请求 kill 掉，但因 NodeManager 已经关闭，这些 Executor(s)实际上并不能被 kill 掉，因此 driver 不能感知到这些 Executor(s)的 LOST 事件，所以并未从自身的 Executor list 中移除，从而导致在 driver 页面上还能看到这些 Executor(s)，这是 YARN NodeManager 关闭之后的正常现象，NodeManager 再次启动后，这些 Executor(s)会被移除。

25.8.1.9 Password cannot be null if SASL is enabled 异常

问题

运行 Spark 的应用启用了 ExternalShuffle，应用出现了 Task 任务丢失，原因是由于 java.lang.NullPointerException: Password cannot be null if SASL is enabled 异常，部分关键日志如下图所示：

回答

造成该现象的原因是 NodeManager 重启。使用 ExternalShuffle 的时候，Spark 将借用 NodeManager 传输 Shuffle 数据，因此 NodeManager 的内存将成为瓶颈。

在当前版本的 FusionInsight 中，NodeManager 的默认内存只有 1G，在数据量比较大（1T 以上）的 Spark 任务下，内存严重不足，消息响应缓慢，导致 FusionInsight 健康检查认为 NodeManager 进程退出，强制重启 NodeManager，导致上述问题产生。

解决方式：

调整 NodeManager 的内存，数据量比较大（1T 以上）的情况下，NodeManager 的内存至少在 4G 以上。

25.8.1.10 向动态分区表中插入数据时，在重试的 task 中出现"Failed to CREATE_FILE"异常

问题

向动态分区表中插入数据时，shuffle 过程中大面积 shuffle 文件损坏（磁盘掉线、节点故障等）后，为什么会在重试的 task 中出现"Failed to CREATE_FILE"异常？

```
2016-06-25 15:11:31,323 | ERROR | [Executor task launch worker-0] | Exception in
task 15.0 in stage 10.1 (TID 1258) |
org.apache.spark.Logging$class.logError(Logging.scala:96)
org.apache.hadoop.hive.ql.metadata.HiveException:
org.apache.hadoop.ipc.RemoteException(org.apache.hadoop.hdfs.protocol.AlreadyBeingC
reatedException): Failed to CREATE FILE /user/hive/warehouse/testdb.db/we
b sales/.hive-staging hive 2016-06-25 15-09-16 999 8137121701603617850-1/-ext-
10000/ temporary/0/ temporary/attempt 201606251509 0010 m 000015 0/ws sold date=199
9-12-17/part-00015 for DFSClient attempt 2016
06251509 0010 m 000015 0 353134803 151 on 10.1.1.5 because this file lease is
currently owned by DFSClient_attempt_201606251509_0010_m_000015_0_-848353830_156 on
10.1.1.6
```

回答

动态分区表插入数据的最后一步是读取 shuffle 文件的数据，再写入到表对应的分区文件中。

当大面积 shuffle 文件损坏后，会引起大批量 task 失败，然后进行 job 重试。重试前 Spark 会将写表分区文件的句柄关闭，大批量 task 关闭句柄时 HDFS 无法及时处理。在 task 进行下一次重试时，句柄在 NameNode 端未被及时释放，即会抛出"Failed to CREATE_FILE"异常。

这种现象仅会在大面积 shuffle 文件损坏时发生，出现异常后 task 会重试，重试耗时在毫秒级，影响较小，可以忽略不计。

25.8.1.11 使用 Hash shuffle 出现任务失败

问题

使用 Hash shuffle 运行 1000000（map 个数）*100000（reduce 个数）的任务，运行日志中出现大量的消息发送失败和 Executor 心跳超时，从而导致任务失败。

回答

对于 Hash shuffle，在 shuffle 的过程中写数据时不做排序操作，只是将数据根据 Hash 的结果，将各个 reduce 分区的数据写到各自的磁盘文件中。

这样带来的问题是如果 reduce 分区的数量比较大的话，将会产生大量的磁盘文件（比如：该问题中将产生 1000000 * 100000 = 10^11 个 shuffle 文件）。如果磁盘文件数量特别巨大，对文件读写的性能会带来比较大的影响，此外由于同时打开的文件句柄数量多，序列化以及压缩等操作需要占用非常大的临时内存空间，对内存的使用和 GC 带来很大的压力，从而容易造成 Executor 无法响应 Driver。

因此，建议使用 Sort shuffle，而不使用 Hash shuffle。

25.8.1.12 访问 Spark 应用的聚合日志页面报“DNS 查找失败”错误

问题

采用 http(s)://<spark ip>:<spark port>的方式直接访问 Spark JobHistory 页面时，如果当前跳转的 Spark JobHistory 页面不是 FusionInsight 代理的页面（FusionInsight 代理的 URL 地址类似于：https://<oms ip>:20026/Spark2x/JobHistory2x/xx/），单击某个应用，再单击“AggregatedLogs”，然后单击需要查看的其中一个 Executor 的“logs”，此时会报如图 25-9 所示的错误。

图25-9 聚合日志失败页面

回答

原因：弹出的 URL 地址（如 https://<hostname>:20026/Spark2x/JobHistory2x/xx/history/application_xxx/jobs/），其中的<hostname>没有在 Windows 系统的 hosts 文件中添加域名信息，导致 DNS 查找失败无法显示此网页。

解决措施：

- 建议用户使用 FusionInsight 代理去访问 Spark JobHistory 页面，即单击如图 25-10 中蓝框所示的 Spark WebUI 的链接。

  图25-10 FusionInsight Manager 的 Spark2x 页面

- 如果用户需要不通过 FusionInsight Manager 访问 Spark JobHistory 页面，则需要将 URL 地址中的<hostname>更改为 IP 地址进行访问，或者在 Windows 系统的 hosts 文件中添加该域名信息。

25.8.1.13 由于 Timeout waiting for task 异常导致 Shuffle FetchFailed

问题

使用 JDBCServer 模式执行 100T 的 TPCDS 测试套，出现 Timeout waiting for task 异常导致 Shuffle FetchFailed，Stage 一直重试，任务无法正常完成。

回答

JDBCServer 方式使用了 ShuffleService 功能，Reduce 阶段所有的 Executor 会从 NodeManager 中获取数据，当数据量达到一个级别（10T 级别），会出现 NodeManager 单点瓶颈（ShuffleService 服务在 NodeManager 进程中），就会出现某些 Task 获取数据超时，从而出现该问题。

因此，当数据量达到 10T 级别以上的 Spark 任务，建议用户关闭 ShuffleService 功能，即在“Spark-defaults.conf”配置文件中将配置项“spark.shuffle.service.enabled”配置为“false”。

25.8.1.14 Executor 进程 Crash 导致 Stage 重试

问题

在执行大数据量的 Spark 任务（如 100T 的 TPCDS 测试套）过程中，有时会出现 Executor 丢失从而导致 Stage 重试的现象。查看 Executor 的日志，出现“Executor 532 is lost rpc with driver,but is still alive, going to kill it”所示信息，表明 Executor 丢失是由于 JVM Crash 导致的。

JVM 的关键 Crash 错误日志，如下：

```
#
# A fatal error has been detected by the Java Runtime Environment:
#
#  Internal Error (sharedRuntime.cpp:834), pid=241075, tid=140476258551552
#  fatal error: exception happened outside interpreter, nmethods and vtable stubs
at pc 0x00007fcda9eb8eb1
```

回答

上述问题在 Oracle 官网上有类似的情况，该问题现象是 Oracle JVM 的缺陷，并不是平台代码引入的问题，且 Spark 中有对 Executor 的容错机制，Executor Crash 之后，Stage 会进入重试，可以保证任务最终可以执行完成，不会对业务产生影响。

25.8.1.15 执行大数据量的 shuffle 过程时 Executor 注册 shuffle service 失败

问题

执行超过 50T 数据的 shuffle 过程时，出现部分 Executor 注册 shuffle service 超时然后丢失从而导致任务失败的问题。错误日志如下所示：

```
2016-10-19 01:33:34,030 | WARN | ContainersLauncher #14 | Exception from container-launch with container ID: container_e1452_1476801295027_2003_01_004512 and exit code: 1 | LinuxContainerExecutor.java:397
ExitCodeException exitCode=1:
at org.apache.hadoop.util.Shell.runCommand(Shell.java:561)
at org.apache.hadoop.util.Shell.run(Shell.java:472)
at org.apache.hadoop.util.Shell$ShellCommandExecutor.execute(Shell.java:738)
at
org.apache.hadoop.yarn.server.nodemanager.LinuxContainerExecutor.launchContainer(LinuxContainerExecutor.java:381)
at
org.apache.hadoop.yarn.server.nodemanager.containermanager.launcher.ContainerLaunch.call(ContainerLaunch.java:312)
at
org.apache.hadoop.yarn.server.nodemanager.containermanager.launcher.ContainerLaunch.call(ContainerLaunch.java:88)
at java.util.concurrent.FutureTask.run(FutureTask.java:266)
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
at java.lang.Thread.run(Thread.java:745)
2016-10-19 01:33:34,031 | INFO | ContainersLauncher #14 | Exception from container-launch. | ContainerExecutor.java:300
2016-10-19 01:33:34,031 | INFO | ContainersLauncher #14 | Container id: container_e1452_1476801295027_2003_01_004512 | ContainerExecutor.java:300
2016-10-19 01:33:34,031 | INFO | ContainersLauncher #14 | Exit code: 1 | ContainerExecutor.java:300
2016-10-19 01:33:34,031 | INFO | ContainersLauncher #14 | Stack trace: ExitCodeException exitCode=1: | ContainerExecutor.java:300
```

回答

由于当前数据量较大，有 50T 数据导入，超过了 shuffle 的规格，shuffle 负载过高，shuffle service 服务处于过载状态，可能无法及时响应 Executor 的注册请求，从而出现上面的问题。

Executor 注册 shuffle service 的超时时间是 5 秒，最多重试 3 次，该参数目前不可配。

建议适当调大 task retry 次数和 Executor 失败次数。

在客户端的“spark-defaults.conf”配置文件中配置如下参数。
“spark.yarn.max.executor.failures”若不存在，则手动添加该参数项。

表25-79 参数说明

参数	描述	默认值
spark.task.maxFailures	task retry 次数。	4

参数	描述	默认值
spark.yarn.max.executor.failures	Executor 失败次数。 关闭 Executor 个数动态分配功能的场景即“spark.dynamicAllocation.enabled”参数设为“false”时。	numExecutors * 2, with minimum of 3
	Executor 失败次数。 开启 Executor 个数动态分配功能的场景即“spark.dynamicAllocation.enabled”参数设为“true”时。	3

25.8.1.16 在 Spark 应用执行过程中 NodeManager 出现 OOM 异常

问题

当开启 Yarn External Shuffle 服务时，在 Spark 应用执行过程中，如果当前 shuffle 连接过多，Yarn External Shuffle 会出现“java.lang.OutofMemoryError: Direct buffer Memory”的异常，该异常说明内存不足。错误日志如下：

```
2016-12-06 02:01:00,768 | WARN  | shuffle-server-38 | Exception in connection from
/192.168.101.95:53680 | TransportChannelHandler.java:79
io.netty.handler.codec.DecoderException: java.lang.OutOfMemoryError: Direct buffer
memory
       at
io.netty.handler.codec.ByteToMessageDecoder.channelRead(ByteToMessageDecoder.java:1
53)
       at
io.netty.channel.AbstractChannelHandlerContext.invokeChannelRead(AbstractChannelHan
dlerContext.java:333)
       at
io.netty.channel.AbstractChannelHandlerContext.fireChannelRead(AbstractChannelHandl
erContext.java:319)
       at
io.netty.channel.DefaultChannelPipeline.fireChannelRead(DefaultChannelPipeline.java
:787)
       at
io.netty.channel.nio.AbstractNioByteChannel$NioByteUnsafe.read(AbstractNioByteChann
el.java:130)
       at
io.netty.channel.nio.NioEventLoop.processSelectedKey(NioEventLoop.java:511)
       at
io.netty.channel.nio.NioEventLoop.processSelectedKeysOptimized(NioEventLoop.java:46
8)
       at
io.netty.channel.nio.NioEventLoop.processSelectedKeys(NioEventLoop.java:382)
       at io.netty.channel.nio.NioEventLoop.run(NioEventLoop.java:354)
       at
io.netty.util.concurrent.SingleThreadEventExecutor$2.run(SingleThreadEventExecutor.
```

```
java:116)
       at java.lang.Thread.run(Thread.java:745)
Caused by: java.lang.OutOfMemoryError: Direct buffer memory
       at java.nio.Bits.reserveMemory(Bits.java:693)
       at java.nio.DirectByteBuffer.<init>(DirectByteBuffer.java:123)
       at java.nio.ByteBuffer.allocateDirect(ByteBuffer.java:311)
       at io.netty.buffer.PoolArena$DirectArena.newChunk(PoolArena.java:434)
       at io.netty.buffer.PoolArena.allocateNormal(PoolArena.java:179)
       at io.netty.buffer.PoolArena.allocate(PoolArena.java:168)
       at io.netty.buffer.PoolArena.reallocate(PoolArena.java:277)
       at io.netty.buffer.PooledByteBuf.capacity(PooledByteBuf.java:108)
       at io.netty.buffer.AbstractByteBuf.ensureWritable(AbstractByteBuf.java:251)
       at io.netty.buffer.AbstractByteBuf.writeBytes(AbstractByteBuf.java:849)
       at io.netty.buffer.AbstractByteBuf.writeBytes(AbstractByteBuf.java:841)
       at io.netty.buffer.AbstractByteBuf.writeBytes(AbstractByteBuf.java:831)
       at
io.netty.handler.codec.ByteToMessageDecoder.channelRead(ByteToMessageDecoder.java:1
46)
       ... 10 more
```

回答

对于 Yarn 的 Shuffle Service，其启动的线程数为机器可用 CPU 核数的两倍，而默认配置的 Direct buffer Memory 为 128M，因此当有较多 shuffle 同时连接时，平均分配到各线程所能使用的 Direct buffer Memory 将较低（例如，当机器的 CPU 为 40 核，Yarn 的 Shuffle Service 启动的线程数为 80，80 个线程共享进程里的 Direct buffer Memory，这种场景下每个线程分配到的内存将不足 2MB）。

因此建议根据集群中的 NodeManager 节点的 CPU 核数适当调整 Direct buffer Memory，例如在 CPU 核数为 40 时，将 Direct buffer Memory 配置为 512M。即配置集群 NodeManger 的“GC_OPTS”参数，如：

-XX:MaxDirectMemorySize=512M

说明

GC_OPTS 参数中-XX:MaxDirectMemorySize 默认没有配置，如需配置，用户可在 GC_OPTS 参数中自定义添加。

具体的配置方法如下：

用户可登录 FusionInsight Manager，单击“集群 > *待操作集群的名称* > 服务 > Yarn > 配置”，单击“全部配置”，单击“NodeManager > 系统”，在“GC_OPTS”参数中修改配置。

表25-80 参数说明

参数	描述	默认值
GC_OPTS	Yarn NodeManger 的 GC 参数。	128M

25.8.1.17 安全集群使用 HiBench 工具运行 sparkbench 获取不到 realm

问题

运行 HiBench6 的 sparkbench 任务，如 Wordcount，任务执行失败，bench.log 显示 Yarn 任务执行失败，登录 Yarn UI，查看对应 application 的失败信息，显示如下：

```
Exception in thread "main" org.apache.spark.SparkException: Unable to load YARN
support
  at
org.apache.spark.deploy.SparkHadoopUtil$.liftedTree1$1(SparkHadoopUtil.scala:390)
  at
org.apache.spark.deploy.SparkHadoopUtil$.yarn$lzycompute(SparkHadoopUtil.scala:385)
  at org.apache.spark.deploy.SparkHadoopUtil$.yarn(SparkHadoopUtil.scala:385)
  at org.apache.spark.deploy.SparkHadoopUtil$.get(SparkHadoopUtil.scala:410)
  at
org.apache.spark.deploy.yarn.ApplicationMaster$.main(ApplicationMaster.scala:796)
  at
org.apache.spark.deploy.yarn.ExecutorLauncher$.main(ApplicationMaster.scala:821)
  at org.apache.spark.deploy.yarn.ExecutorLauncher.main(ApplicationMaster.scala)
 Caused by: java.lang.IllegalArgumentException: Can't get Kerberos realm
  at
org.apache.hadoop.security.HadoopKerberosName.setConfiguration(HadoopKerberosName.j
ava:65)
  at
org.apache.hadoop.security.UserGroupInformation.initialize(UserGroupInformation.jav
a:288)
  at
org.apache.hadoop.security.UserGroupInformation.setConfiguration(UserGroupInformati
on.java:336)
  at org.apache.spark.deploy.SparkHadoopUtil.<init>(SparkHadoopUtil.scala:51)
  at
org.apache.spark.deploy.yarn.YarnSparkHadoopUtil.<init>(YarnSparkHadoopUtil.scala:4
9)
  at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
  at
sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl
.java:62)
  at
sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAcce
ssorImpl.java:45)
  at java.lang.reflect.Constructor.newInstance(Constructor.java:423)
  at java.lang.Class.newInstance(Class.java:442)
  at
org.apache.spark.deploy.SparkHadoopUtil$.liftedTree1$1(SparkHadoopUtil.scala:387)
  ... 6 more
 Caused by: java.lang.reflect.InvocationTargetException
  at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
  at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
  at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:4
3)
  at java.lang.reflect.Method.invoke(Method.java:498)
  at
```

```
org.apache.hadoop.security.authentication.util.KerberosUtil.getDefaultRealm(KerberosUtil.java:88)
  at
org.apache.hadoop.security.HadoopKerberosName.setConfiguration(HadoopKerberosName.java:63)
  ... 16 more
 Caused by: KrbException: Cannot locate default realm
  at sun.security.krb5.Config.getDefaultRealm(Config.java:1029)
  ... 22 more
```

回答

失败原因是 C80SPC200 版本开始，安装集群不再替换/etc/krb5.conf 文件，改为通过配置参数指定到客户端内 krb5 路径，而 HiBench 并不引用客户端配置文件。解决方案：将客户端/opt/client/KrbClient/kerberos/var/krb5kdc/krb5.conf，copy 覆盖集群内所有节点的/etc/krb5.conf，注意替换前需要备份。

25.8.2 SQL 和 DataFrame

25.8.2.1 Spark SQL ROLLUP 和 CUBE 使用的注意事项

问题

假设有表 src(d1, d2, m)，其数据如下：

```
1 a 1
1 b 1
2 b 2
```

对于语句 select d1, sum(d1) from src group by d1, d2 with rollup 其结果如下：

```
NULL 0
1    2
2    2
1    1
1    1
2    2
```

对于以上结果的第一条为什么是(NULL,0)而不是(NULL,4)。

回答

在进行 rollup 和 cube 操作时，用户通常是基于维度进行分析，需要的是度量的结果，因此不会对维度进行聚合操作。

例如当前有表 src(d1, d2, m)，那么语句 1“select d1, sum(m) from src group by d1, d2 with rollup”就是对维度 d1 和 d2 进行上卷操作计算度量 m 的结果，因此有实际业务意义，而其结果也跟预期是一致的。但语句 2“select d1, sum(d1) from src group by d1, d2 with rollup”则从业务上无法解释。当前对于语句 2 所有聚合（sum/avg/max/min）结果均为 0。

说明

只有在 rollup 和 cube 操作中对出现在 group by 中的字段进行聚合结果才是 0，非 rollup 和 cube 操作其结果跟预期一致。

25.8.2.2 Spark SQL 在不同 DB 都可以显示临时表

问题

切换数据库之后，为什么还能看到之前数据库的临时表？

1. 创建一个 DataSource 的临时表，例如以下建表语句。

```
create temporary table ds_parquet
using org.apache.spark.sql.parquet
options(path '/tmp/users.parquet');
```

2. 切换到另外一个数据库，执行 ***show tables***，依然可以看到上个步骤创建的临时表。

```
0: jdbc:hive2://192.168.169.84:22550/default> show tables;
+-----------------+--------------+--+
|    tableName    | isTemporary  |
+-----------------+--------------+--+
| ds_parquet      | true         |
| cmb_tbl_carbon  | false        |
+-----------------+--------------+--+
2 rows selected (0.109 seconds)
0: jdbc:hive2://192.168.169.84:22550/default>
```

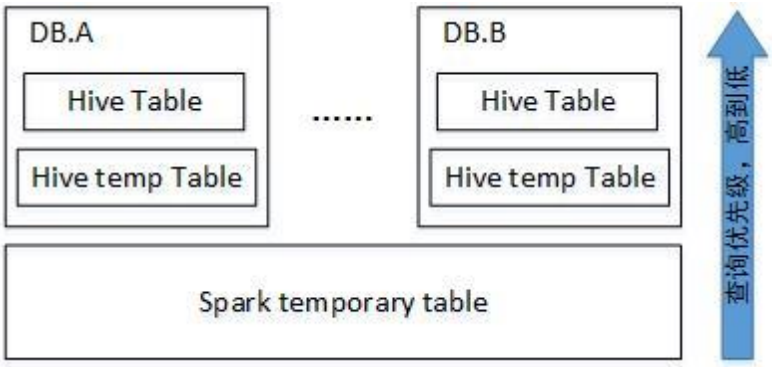
回答

Spark 的表管理层次如图 25-11 所示，最底层是 Spark 的临时表，存储着使用 DataSource 方式的临时表，在这一个层面中没有数据库的概念，因此对于这种类型表，表名在各个数据库中都是可见的。

上层为 Hive 的 MetaStore，该层有了各个 DB 之分。在每个 DB 中，又有 Hive 的临时表与 Hive 的持久化表，因此在 Spark 中允许三个层次的同名数据表。

查询的时候，Spark SQL 优先查看是否有 Spark 的临时表，再查找当前 DB 的 Hive 临时表，最后查找当前 DB 的 Hive 持久化表。

图25-11 Spark 表管理层次



当 Session 退出时，用户操作相关的临时表将自动删除。建议用户不要手动删除临时表。

删除临时表时，其优先级与查询相同，从高到低为 Spark 临时表、Hive 临时表、Hive 持久化表。如果想直接删除 Hive 表，不删除 Spark 临时表，您可以直接使用 ***drop table DbName.TableName*** 命令。

25.8.2.3 如何在 Spark 命令中指定参数值

问题

如果用户不希望在界面上或配置文件设置参数值，如何在 Spark 命令中指定参数值？

回答

Spark 的配置项，不仅可以在配置文件中设置，也可以在命令中指定参数值。

在 Spark 客户端，应用执行命令添加如下内容设置参数值，命令执行完成后立即生效。在--conf 后添加参数名称及其参数值，例如：

--conf spark.eventQueue.size=50000

25.8.2.4 SparkSQL 建表时的目录权限

问题

新建的用户，使用 SparkSQL 建表时出现类似如下错误：

```
0: jdbc:hive2://192.168.169.84:22550/default> create table testACL(c string);
Error: org.apache.spark.sql.execution.QueryExecutionException: FAILED: Execution
Error, return code 1 from
org.apache.hadoop.hive.ql.exec.DDLTask. MetaException(message:Got exception:
org.apache.hadoop.security.AccessControlException
Permission denied: user=testACL, access=EXECUTE,
inode="/user/hive/warehouse/testacl":spark:hadoop:drwxrwx---
   at
org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.checkAccessAcl(FSPermiss
ionChecker.java:403)
```

```
    at
org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.check(FSPermissionChecke
r.java:306)
    at
org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.checkTraverse(FSPermissi
onChecker.java:259)
    at
org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.checkPermission(FSPermis
sionChecker.java:205)
    at
org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.checkPermission(FSPermis
sionChecker.java:190)
    at
org.apache.hadoop.hdfs.server.namenode.FSDirectory.checkPermission(FSDirectory.java
:1710)
    at
org.apache.hadoop.hdfs.server.namenode.FSDirStatAndListingOp.getFileInfo(FSDirStatA
ndListingOp.java:109)
    at
org.apache.hadoop.hdfs.server.namenode.FSNamesystem.getFileInfo(FSNamesystem.java:3
762)
    at
org.apache.hadoop.hdfs.server.namenode.NameNodeRpcServer.getFileInfo(NameNodeRpcSer
ver.java:1014)
    at
org.apache.hadoop.hdfs.protocolPB.ClientNamenodeProtocolServerSideTranslatorPB.getF
ileInfo(ClientNamenodeProtocolServerSideTranslatorPB.java:853)
    at
org.apache.hadoop.hdfs.protocol.proto.ClientNamenodeProtocolProtos$ClientNamenodePr
otocol$2.callBlockingMethod(ClientNamenodeProtocolProtos.java)
    at
org.apache.hadoop.ipc.ProtobufRpcEngine$Server$ProtoBufRpcInvoker.call(ProtobufRpcE
ngine.java:616)
    at org.apache.hadoop.ipc.RPC$Server.call(RPC.java:973)
    at org.apache.hadoop.ipc.Server$Handler$1.run(Server.java:2089)
    at org.apache.hadoop.ipc.Server$Handler$1.run(Server.java:2085)
    at java.security.AccessController.doPrivileged(Native Method)
    at javax.security.auth.Subject.doAs(Subject.java:422)
    at
org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1675)
    at org.apache.hadoop.ipc.Server$Handler.run(Server.java:2083)
) (state=,code=0)
```

回答

Spark SQL 建表底层调用的是 Hive 的接口，其建表时会在“/user/hive/warehouse”目录下新建一个以表名命名的目录，因此要求用户具备“/user/hive/warehouse”目录的读写、执行权限或具有 Hive 的 group 权限。

“/user/hive/warehouse”目录可通过 hive.metastore.warehouse.dir 参数指定。

25.8.2.5 为什么不同服务之间互相删除 UDF 失败

问题

不同服务之间互相删除 UDF 失败，例如，Spark SQL 无法删除 Hive 创建的 UDF。

回答

当前可以通过以下 3 种方式创建 UDF：

1. 在 Hive 端创建 UDF。
2. 通过 JDBCServer 接口创建 UDF。用户可以通过 Spark Beeline 或者 JDBC 客户端代码来连接 JDBCServer，从而执行 SQL 命令，创建 UDF。
3. 通过 spark-sql 创建 UDF。

删除 UDF 失败，存在以下两种场景：

- 在 Spark Beeline 中，对于其他方式创建的 UDF，需要重新启动 Spark 服务端的 JDBCServer 后，才能将此类 UDF 删除成功，否则删除失败。在 spark-sql 中，对于其他方式创建的 UDF，需要重新启动 spark-sql 后，才能将此类 UDF 删除成功，否则删除失败。

  原因：创建 UDF 后，Spark 服务端的 JDBCServer 未重启或者 spark-sql 未重新启动的场景，Spark 所在线程的 FunctionRegistry 对象未保存新创建的 UDF，那么删除 UDF 时就会出现错误。

  解决方法：重启 Spark 服务端的 JDBCServer 和 spark-sql，再删除此类 UDF。
- 在 Hive 端创建 UDF 时未在创建语句中指定 jar 包路径，而是通过 **add jar** 命令添加 UDF 的 jar 包如 **add jar /opt/test/two_udfs.jar**，这种场景下，在其他服务中删除 UDF 时就会出现 ClassNotfound 的错误，从而导致删除失败。

  原因：在删除 UDF 时，会先获取该 UDF，此时会去加载该 UDF 对应的类，由于创建 UDF 时是通过 **add jar** 命令指定 jar 包路径的，其他服务进程的 classpath 不存在这些 jar 包，因此会出现 ClassNotfound 的错误从而导致删除失败。

  解决方法：该方式创建的 UDF 不支持通过其他方式删除，只能通过与创建时一致的方式删除。

25.8.2.6 Spark SQL 无法查询到 Parquet 类型的 Hive 表的新插入数据

问题

为什么通过 Spark SQL 无法查询到存储类型为 Parquet 的 Hive 表的新插入数据？主要有以下两种场景存在这个问题：

1. 对于分区表和非分区表，在 Hive 客户端中执行插入数据的操作后，会出现 Spark SQL 无法查询到最新插入的数据的问题。
2. 对于分区表，在 Spark SQL 中执行插入数据的操作后，如果分区信息未改变，会出现 Spark SQL 无法查询到最新插入的数据的问题。

回答

由于 Spark 存在一个机制，为了提高性能会缓存 Parquet 的元数据信息。当通过 Hive 或其他方式更新了 Parquet 表时，缓存的元数据信息未更新，导致 Spark SQL 查询不到新插入的数据。

对于存储类型为 Parquet 的 Hive 分区表，在执行插入数据操作后，如果分区信息未改变，则缓存的元数据信息未更新，导致 Spark SQL 查询不到新插入的数据。

解决措施：在使用 Spark SQL 查询之前，需执行 Refresh 操作更新元数据信息。

REFRESH TABLE table_name;

table_name 为刷新的表名，该表必须存在，否则会出错。

执行查询语句时，即可获取到最新插入的数据。

Spark 官网提供了此机制的描述，详情请参见：https://spark.apache.org/docs/3.1.1/sql-programming-guide.html#metadata-refreshing

25.8.2.7 cache table 使用指导

问题

cache table 的作用是什么？cache table 时需要注意哪些方面？

回答

Spark SQL 可以将表 cache 到内存中，并且使用压缩存储来尽量减少内存压力。通过将表 cache，查询可以直接从内存中读取数据，从而减少读取磁盘带来的内存开销。

但需要注意的是，被 cache 的表会占用 executor 的内存。尽管在 Spark SQL 采用压缩存储的方式来尽量减少内存开销、缓解 GC 压力，但当缓存的表较大或者缓存表数量较多时，将不可避免的影响 executor 的稳定性。

此时的最佳实践是，当不需要将表 cache 来实现查询加速时，应及时将表进行 uncache 以释放内存。可以执行命令 ***uncache table*** *table_name* 来 uncache 表。

说明

被 cache 的表也可以在 Spark Driver UI 的 Storage 标签里查看。

25.8.2.8 Repartition 时有部分 Partition 没数据

问题

在 repartition 操作时，分块数“spark.sql.shuffle.partitions”设置为 4500，repartition 用到的 key 列中有超过 4000 个的不同 key 值。期望不同 key 对应的数据能分到不同的 partition，实际上却只有 2000 个 partition 里有数据，不同 key 对应的数据也被分到相同的 partition 里。

回答

这是正常现象。

数据分到哪个 partition 是通过对 key 的 hashcode 取模得到的，不同的 hashcode 取模后的结果有可能是一样的，那样数据就会被分到相同的 partition 里面，因此出现有些 partition 没有数据而有些 partition 里面有多个 key 对应的数据。

通过调整“spark.sql.shuffle.partitions”参数值可以调整取模时的基数，改善数据分块不均匀的情况，多次验证发现配置为质数或者奇数效果比较好。

在 Driver 端的“spark-defaults.conf”配置文件中调整如下参数。

表25-81 参数说明

参数	描述	默认值
spark.sql.shuffle.partitions	shuffle 操作时，shuffle 数据的分块数。	200

25.8.2.9 16T 的文本数据转成 4T Parquet 数据失败

问题

使用默认配置时，16T 的文本数据转成 4T Parquet 数据失败，报如下错误信息。

```
Job aborted due to stage failure: Task 2866 in stage 11.0 failed 4 times, most
recent failure: Lost task 2866.6 in stage 11.0 (TID 54863, linux-161, 2):
java.io.IOException: Failed to connect to /10.16.1.11:23124
at
org.apache.spark.network.client.TransportClientFactory.createClient(TransportClient
Factory.java:214)
at
org.apache.spark.network.client.TransportClientFactory.createClient(TransportClient
Factory.java:167)
at
org.apache.spark.network.netty.NettyBlockTransferService$$anon$1.createAndStart(Net
tyBlockTransferService.scala:92)
```

使用的默认配置如表 25-82 所示。

表25-82 参数说明

参数	描述	默认值
spark.sql.shuffle.partitions	shuffle 操作时，shuffle 数据的分块数。	200
spark.shuffle.sasl.timeout	shuffle 操作时 SASL 认证的超时时间。单位：秒。	120s
spark.shuffle.io.connectionTimeout	shuffle 操作时连接远程节点的超时时间。单位：秒。	120s

参数	描述	默认值
spark.network.timeout	所有涉及网络连接操作的超时时间。单位：秒。	360s

回答

由于当前数据量较大，有 16T，而分区数只有 200，造成每个 task 任务过重，才会出现上面的问题。

为了解决上面问题，需要对参数进行调整。

- 增大 partition 数，把任务切分的更小。
- 增大任务执行过程中的超时时间。

在客户端的“spark-defaults.conf”配置文件中配置如下参数。

表25-83 参数说明

参数	描述	建议值
spark.sql.shuffle.partitions	shuffle 操作时，shuffle 数据的分块数。	4501
spark.shuffle.sasl.timeout	shuffle 操作时 SASL 认证的超时时间。单位：秒。	2000s
spark.shuffle.io.connectionTimeout	shuffle 操作时连接远程节点的超时时间。单位：秒。	3000s
spark.network.timeout	所有涉及网络连接操作的超时时间。单位：秒。	360s

25.8.2.10 当表名为 table 时，执行相关操作时出现异常

问题

当创建了表名为 table 的表后，执行 **drop table table** 上报以下错误，或者执行其他操作也会出现类似错误。

```
16/07/12 18:56:29 ERROR SparkSQLDriver: Failed in [drop table table]
java.lang.RuntimeException: [1.1] failure: identifier expected
table
^
at scala.sys.package$.error(package.scala:27)
at
org.apache.spark.sql.catalyst.SqlParserTrait$class.parseTableIdentifier(SqlParser.scala:56)
```

```
at
org.apache.spark.sql.catalyst.SqlParser$.parseTableIdentifier(SqlParser.scala:485)
```

回答

这是因为 table 为 Spark SQL 的关键词，不能做为表名使用。建议用户不要使用 table 作为表的名字。

25.8.2.11 执行 analyze table 语句，因资源不足出现任务卡住

问题

使用 spark-sql 执行 ***analyze table*** 语句，任务一直卡住，打印的信息如下：

```
spark-sql> analyze table hivetable2 compute statistics;
Query ID = root_20160716174218_90f55869-000a-40b4-a908-533f63866fed
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
16/07/20 17:40:56 WARN JobResourceUploader: Hadoop command-line option parsing not
performed. Implement the Tool interface and execute your application with
ToolRunner to remedy this.
Starting Job = job_1468982600676_0002, Tracking URL = http://10-120-175-
107:8088/proxy/application_1468982600676_0002/
Kill Command = /opt/hadoopclient/HDFS/hadoop/bin/hadoop job  -kill
job_1468982600676_0002
```

回答

执行 ***analyze table hivetable2 compute statistics*** 语句时，由于该 sql 语句会启动 MapReduce 任务。从 YARN 的 ResourceManager Web UI 页面看到，该任务由于资源不足导致任务没有被执行，表现出任务卡住的现象。

图25-12 ResourceManager Web UI 页面

建议用户执行 ***analyze table*** 语句时加上 noscan，其功能与 ***analyze table hivetable2 compute statistics*** 语句相同，具体命令如下：

```
spark-sql> analyze table hivetable2 compute statistics noscan
```

该命令不用启动 MapReduce 任务，不会占用 YARN 资源，从而任务可以被执行。

25.8.2.12 为什么有时访问没有权限的 parquet 表时，在上报“Missing Privileges”错误提示之前，会运行一个 Job？

问题

为什么有时访问没有权限的 parquet 表时，在上报“Missing Privileges”错误提示之前，会运行一个 Job？

回答

Spark SQL 对用户 SQL 语句的执行逻辑是：首先解析出语句中包含的表，再获取表的元数据信息，然后对权限进行检查。

当表是 parquet 表时，元数据信息包括文件的 Split 信息。Split 信息需要调用 HDFS 的接口去读取，当表包含的文件数量很多时，串行读取 Split 信息变得缓慢，影响性能。故对此做了优化，当表包含的文件大于一定阈值（即 spark.sql.sources.parallelSplitDiscovery.threshold 参数值）时，会生成一个 Job，利用 Executor 的并行能力去读取，从而提升执行效率。

由于权限检查在获取表元数据之后，因此当读取的 parquet 表包含的文件数量很多时，会在报“Missing Privileges”之前，运行一个 Job 来并行读取元数据信息。

25.8.2.13 执行 Hive 命令修改元数据时失败或不生效

问题

对于 datasource 表和 Spark on HBase 表，执行 Hive 相关命令修改元数据时，出现失败或者不生效情况。

回答

当前版本不支持执行 Hive 修改元数据的相关命令操作 datasource 表和 Spark on HBase 表。

25.8.2.14 spark-sql 退出时打印 RejectedExecutionException 异常栈

问题

执行大数据量的 Spark 任务（如 2T 的 TPCDS 测试套），任务运行成功后，在 spark-sql 退出时概率性出现 RejectedExecutionException 的异常栈信息，相关日志如下所示：

```
16/07/16 10:19:56 ERROR TransportResponseHandler: Still have 2 requests outstanding
when connection from linux-192/10.1.1.5:59250 is closed
java.util.concurrent.RejectedExecutionException: Task
scala.concurrent.impl.CallbackRunnable@5fc1ab rejected from
java.util.concurrent.ThreadPoolExecutor@52fa7e19[Terminated, pool size = 0, active
threads = 0, queued tasks = 0, completed tasks = 3025]
```

回答

出现上述问题的原因是：当 spark-sql 退出时，应用退出关闭消息通道，如果当前还有消息未处理，需要做连接关闭异常的处理，此时，如果 scala 内部的线程池已经关闭，就会打印 RejectEdExecutionException 的异常栈，如果 scala 内部的线程池尚未关闭就不会打印该异常栈。

因为该问题出现在应用退出时，此时任务已经运行成功，所以不会对业务产生影响。

25.8.2.15 健康检查时，误将 JDBCServer Kill

问题

健康检查方案中，在并发执行的语句达到线程池上限后依然会导致健康检查命令无法执行，从而导致健康检查程序超时，然后把 Spark JDBCServer 进程 Kill。

回答

当前 JDBCServer 中存在两个线程池 HiveServer2-Handler-Pool 和 HiveServer2-Background-Pool，其中 HiveServer2-Handler-Pool 用于处理 session 连接，HiveServer2-Background-Pool 用于处理 SQL 语句的执行。

当前的健康检查机制是通过新起一个 session 连接，并在该 session 所在的线程中执行健康检查命令 ***HEALTHCHECK*** 来判断 Spark JDBCServer 的健康状况，因此 HiveServer2-Handler-Pool 必须保留一个线程，用于处理健康检查的 session 连接和健康检查命令执行，否则将导致无法建立健康检查的 session 连接或健康检查命令无法执行，从而认为 Spark JDBCServer 不健康而被 Kill。即如果当前 HiveServer2-Handler-Pool 的线程池数为 100，那么最多支持连接 99 个 session。

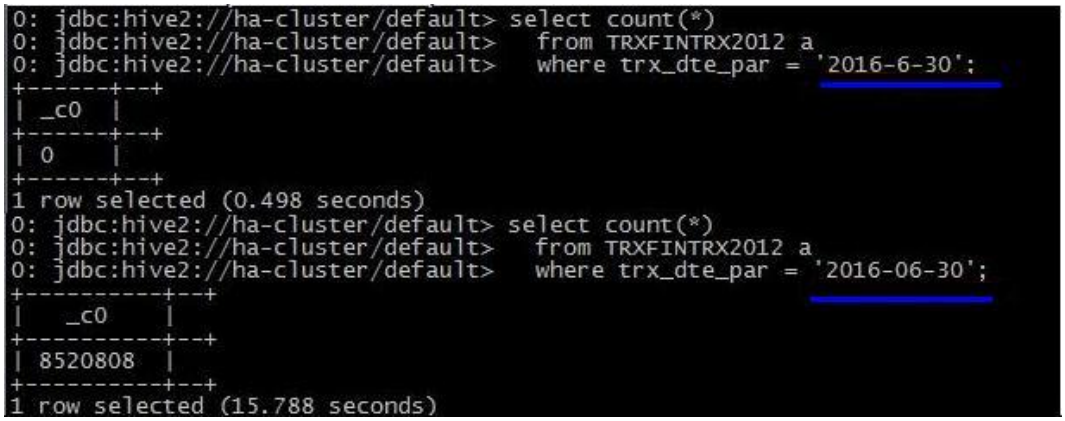
25.8.2.16 日期类型的字段作为过滤条件时匹配'2016-6-30'时没有查询结果

问题

为什么日期类型的字段作为过滤条件时匹配'2016-6-30'时没有查询结果，匹配'2016-06-30'时有查询结果。

如下图所示：“select count(*)from trxfintrx2012 a where trx_dte_par='2016-6-30'”，其中 trx_dte_par 为日期类型的字段，当过滤条件为“where trx_dte_par='2016-6-30'”时没有查询结果，当过滤条件为“where trx_dte_par='2016-06-30'”时有查询结果。

图25-13 示例



回答

在 Spark SQL 查询语句中，当查询条件中含有日期格式的字符串时，Spark SQL 不会对它做日期格式的检查，就是把它当做普通的字符串进行匹配。以上面的例子为例，如果数据格式为"yyyy-mm-dd"，那么字符串'2016-6-30'就是不正确的数据格式。

25.8.2.17 为什么在启动 spark-beeline 的命令中指定“--hivevar”选项无效

问题

为什么在启动 spark-beeline 的命令中指定“--hivevar”选项无效？

从 V100R002C60 版本开始，在启动 spark-beeline 的命令中如果使用了“--hivevar <VAR_NAME>=<var_value>”选项自定义一个变量，在启动 spark-beeline 时不会报错，但在 SQL 语句中用到变量<VAR_NAME>时会报无法解析<VAR_NAME>的错误。

举例说明，场景如下：

1. 执行以下命令启动 spark-beeline：

   spark-beeline --hivevar <VAR_NAME>=<var_value>

2. 启动成功后，在 spark-beeline 中执行 SQL 语句，如“DROP TABLE ${VAR_NAME}”，报无法解析 VAR_NAME 的错误。

回答

从 V100R002C60 版本开始，因新增多 session 管理功能，Hive 的特性“--hivevar <VAR_NAME>=<var_value>”在 Spark 中已不再支持，因此在 spark-beeline 的启动命令中使用“--hivevar”选项无效。

25.8.2.18 在 spark-beeline 中创建临时表/视图时，报 HDFS 目录无权限操作的错误

问题

在普通模式下，用户在 spark-beeline 中创建临时表或创建视图时，报“Permission denied”的错误，这个错误表明 HDFS 目录无权限操作。错误日志信息如下：

```
org.apache.hadoop.security.AccessControlException Permission denied: user=root, access=EXECUTE, inode="/tmp/spark/sparkhive-scratch/omm/e579a76f-43ed-4014-8a54-1072c07ceeff/_tmp_space.db/52db1561-60b0-4e7d-8a25-c2eaa44850a9":omm:hadoop:drwx------
```

回答

在普通模式下，当使用非 omm 用户（例如 root 用户）执行启动 spark-beeline 的命令时，在未指定“-n”时用户为 root，而启动 spark-beeline 后，JDBCServer 会创建一个 HDFS 新目录，由于当前版本启动 JDBCServer 的用户是 omm，而在 DataSightV100R002C30 以前的版本是 root 用户，因此当前该 HDFS 目录的 owner 为 omm、group 为 hadoop。在 spark-beeline 中创建临时表或视图时会使用该 HDFS 目录，此时是 root 用户，但是 root 用户在 HDFS 中是一个普通用户，因此没有权限操作 omm 用户的 HDFS 目录，从而报“Permission denied”的错误。

综上所述，在普通模式下，只有 omm 用户可以创建临时表或视图，如果用户需要创建临时表或视图，可通过在启动 spark-beeline 时带“-n omm”选项指定操作用户为 omm，这样便有权限操作成功。

25.8.2.19 执行复杂 SQL 语句时报“Code of method ... grows beyond 64 KB”的错误

问题

当执行一个很复杂的 SQL 语句时，例如有多层语句嵌套，且单层语句中对字段有大量的逻辑处理（如多层嵌套的 case when 语句），此时执行该语句会报如下所示的错误日志，该错误表明某个方法的代码超出了 64KB。

```
java.util.concurrent.ExecutionException: java.lang.Exception: failed to compile: org.codehaus.janino.JaninoRuntimeException: Code of method "(Lorg/apache/spark/sql/catalyst/expressions/GeneratedClass$SpecificUnsafeProjection;Lorg/apache/spark/sql/catalyst/InternalRow;)V" of class "org.apache.spark.sql.catalyst.expressions.GeneratedClass$SpecificUnsafeProjection" grows beyond 64 KB
```

回答

在开启钨丝计划（即 tungsten 功能）后，Spark 对于部分执行计划会使用 codegen 的方式来生成 Java 代码，但 JDK 编译时要求 Java 代码中的每个函数的长度不能超过 64KB。当执行一个很复杂的 SQL 语句时，例如有多层语句嵌套，且单层语句中对字段有大量的逻辑处理（如多层嵌套的 case when 语句），这种情况下，通过 codegen 生成的 Java 代码中函数的大小就可能会超过 64KB，从而导致编译失败。

规避措施：

当出现上述问题时，用户可以通过关闭钨丝计划，关闭使用 codegen 的方式来生成 Java 代码的功能，从而确保语句的正常执行。即在客户端的“spark-defaults.conf”配置文件中将“spark.sql.codegen.wholeStage”配置为“false”。

25.8.2.20 在 Beeline/JDBCServer 模式下连续运行 10T 的 TPCDS 测试套会出现内存不足的现象

问题

在 Driver 内存配置为 10G 时，Beeline/JDBCServer 模式下连续运行 10T 的 TPCDS 测试套，会出现因为 Driver 内存不足导致 SQL 语句执行失败的现象。

回答

当前在默认配置下，在内存中保留的 Job 和 Stage 的 UI 数据个数为 1000 个。

当前大集群优化已增加将 UI 数据溢出到磁盘的优化，其溢出条件是每个 Stage 中的 UI 数据大小达到最小阈值 5MB。如果每个 Stage 的 task 数较小，那么其 UI 数据大小可能达不到该阈值，从而导致该 Stage 的 UI 数据一直缓存在内存中，直到 UI 数据个数到达保留的上限值（当前默认值为 1000 个），旧的 UI 数据才会在内存中被清除。

因此，在将旧的 UI 数据从内存中清除之前，UI 数据会占用大量内存，从而导致执行 10T 的 TPCDS 测试套时出现 Driver 内存不足的现象。

规避措施：

- 根据业务需要，配置合适的需要保留的 Job 和 Stage 的 UI 数据个数，即配置“spark.ui.retainedJobs”和“spark.ui.retainedStages”参数。详细信息请参考常用参数中的表 25-15。
- 如果需要保留的 Job 和 Stage 的 UI 数据个数较多，可通过配置“spark.driver.memory”参数，适当增大 Driver 的内存。详细信息请参考常用参数中的表 25-12。

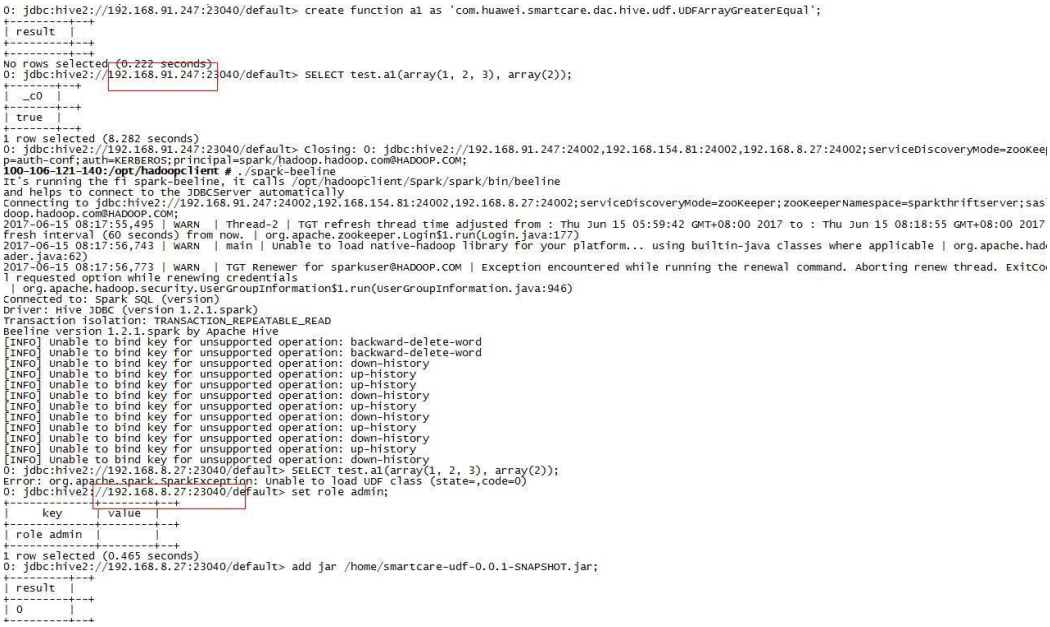
25.8.2.21 连上不同的 JDBCServer，function 不能正常使用

问题

场景一：

通过 add jar 的方式建立永久函数，当 Beeline 连上不同的 JDBCServer 或者 JDBCServer 重启后都需要重新 add jar。

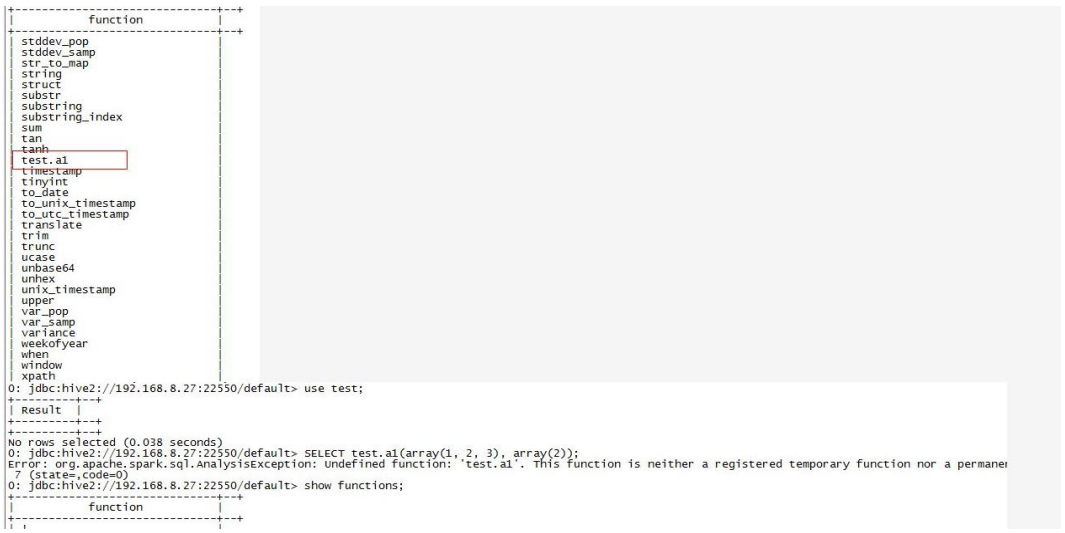
图25-14 场景一异常信息



场景二：

show functions 能够查到相应的函数，但是无法使用，这是由于连接上的 JDBC 节点上没有相应路径的 jar 包，添加上相应的 jar 包能够查询成功。

图25-15 场景二异常信息



回答

场景一：

add jar 语句只会将 jar 加载到当前连接的 JDBCServer 的 jarClassLoader，不同 JDBCServer 不会共用。JDBCServer 重启后会创建新的 jarClassLoader，所以需要重新 add jar。

添加 jar 包有两种方式：可以在启动 spark-sql 的时候添加 jar 包，如 **spark-sql --jars /opt/test/two_udfs.jar**；也可在 spark-sql 启动后再添加 jar 包，如 **add jar /opt/test/two_udfs.jar**。add jar 所指定的路径可以是本地路径也可以是 HDFS 上的路径。

场景二：

show functions 会从外部的 Catalog 获取当前 database 中所有的 function。SQL 中使用 function 时，JDBCServer 会加载该 function 对应的 jar。

若 jar 不存在，则该 function 无法使用，需要重新执行 ***add jar*** 命令。

25.8.2.22 Spark2x 无法访问 Spark1.5 创建的 DataSource 表

问题

在 Spark2x 中访问 Spark1.5 创建的 DataSource 表时，报无法获取 schema 信息，导致无法访问表。

回答

- 原因分析：

  这是由于 Spark2x 与 Spark1.5 存储 DataSoure 表信息的格式不一致导致的。Spark1.5 会将 schema 信息分成多个 part，使用 path.park.0 作为 key 进行存储，读取时再将各个 part 都读取出来，重新拼成完整的信息。而 Spark2x 直接使用相应的 key 获取对应的信息。这样在 Spark2x 中去读取 Spark1.5 创建的 DataSource 表时，就无法成功读取到 key 对应的信息，导致解析 DataSource 表信息失败。

  而在处理 Hive 格式的表时，Spark2x 与 Spark1.5 的存储方式一致，所以 Spark2x 可以直接读取 Spark1.5 创建的表，不存在上述问题。

- 规避措施：

  Spark2x 可以通过创建外表的方式来创建一张指向 Spark1.5 表实际数据的表，这样可以实现在 Spark2x 中读取 Spark1.5 创建的 DataSource 表。同时，Spark1.5 更新过数据后，Spark2x 中访问也能感知到变化 ，反过来一样。这样即可实现 Spark2x 对 Spark1.5 创建的 DataSource 表的访问。

25.8.2.23 为什么 spark-beeline 运行失败报“Failed to create ThriftService instance”的错误

问题

为什么 spark-beeline 运行失败报“Failed to create ThriftService instance”的错误？

Beeline 日志如下所示：

```
Error: Failed to create ThriftService instance (state=,code=0)
Beeline version 1.2.1.spark by Apache Hive
[INFO] Unable to bind key for unsupported operation: backward-delete-word
```

```
[INFO] Unable to bind key for unsupported operation: backward-delete-word
[INFO] Unable to bind key for unsupported operation: down-history
[INFO] Unable to bind key for unsupported operation: up-history
[INFO] Unable to bind key for unsupported operation: up-history
[INFO] Unable to bind key for unsupported operation: down-history
[INFO] Unable to bind key for unsupported operation: up-history
[INFO] Unable to bind key for unsupported operation: down-history
[INFO] Unable to bind key for unsupported operation: up-history
[INFO] Unable to bind key for unsupported operation: down-history
[INFO] Unable to bind key for unsupported operation: up-history
[INFO] Unable to bind key for unsupported operation: down-history
beeline>
```

同时，在 JDBCServer 端出现“Timed out waiting for client to connect”的错误日志，关键日志如下所示：

```
2017-07-12 17:35:11,284 | INFO  | [main] | Will try to open client transport with
JDBC Uri: jdbc:hive2://192.168.101.97:23040/default;principal=spark/hadoop.<系统域名>@<系统域名>;healthcheck=true;saslQop=auth-
conf;auth=KERBEROS;user.principal=spark/hadoop.<系统域名>@<系统域名>;user.keytab=${BIGDATA_HOME}/FusionInsight_HD_8.1.0.1/install/FusionInsight-Spark-3.1.1/keytab/spark/JDBCServer/spark.keytab |
org.apache.hive.jdbc.HiveConnection.openTransport(HiveConnection.java:317)
2017-07-12 17:35:11,326 | INFO  | [HiveServer2-Handler-Pool: Thread-92] | Client
protocol version: HIVE_CLI_SERVICE_PROTOCOL_V8 |
org.apache.proxy.service.ThriftCLIProxyService.OpenSession(ThriftCLIProxyService.java:554)
2017-07-12 17:35:49,790 | ERROR | [HiveServer2-Handler-Pool: Thread-113] | Timed
out waiting for client to connect.
Possible reasons include network issues, errors in remote driver or the cluster has
no available resources, etc.
Please check YARN or Spark driver's logs for further information. |
org.apache.proxy.service.client.SparkClientImpl.<init>(SparkClientImpl.java:90)
java.util.concurrent.ExecutionException: java.util.concurrent.TimeoutException:
Timed out waiting for client connection.
 at io.netty.util.concurrent.AbstractFuture.get(AbstractFuture.java:37)
 at org.apache.proxy.service.client.SparkClientImpl.<init>(SparkClientImpl.java:87)
 at
org.apache.proxy.service.client.SparkClientFactory.createClient(SparkClientFactory.java:79)
 at
org.apache.proxy.service.SparkClientManager.createSparkClient(SparkClientManager.java:145)
 at
org.apache.proxy.service.SparkClientManager.createThriftServerInstance(SparkClientManager.java:160)
 at
org.apache.proxy.service.ThriftServiceManager.getOrCreateThriftServer(ThriftServiceManager.java:182)
 at
org.apache.proxy.service.ThriftCLIProxyService.OpenSession(ThriftCLIProxyService.java:596)
 at
org.apache.hive.service.cli.thrift.TCLIService$Processor$OpenSession.getResult(TCLIService.java:1257)
 at
```

```
org.apache.hive.service.cli.thrift.TCLIService$Processor$OpenSession.getResult(TCLI
Service.java:1242)
 at org.apache.thrift.ProcessFunction.process(ProcessFunction.java:39)
 at org.apache.thrift.TBaseProcessor.process(TBaseProcessor.java:39)
 at
org.apache.hadoop.hive.thrift.HadoopThriftAuthBridge$Server$TUGIAssumingProcessor.p
rocess(HadoopThriftAuthBridge.java:696)
 at
org.apache.thrift.server.TThreadPoolServer$WorkerProcess.run(TThreadPoolServer.java
:286)
 at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
 at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
 at java.lang.Thread.run(Thread.java:748)
Caused by: java.util.concurrent.TimeoutException: Timed out waiting for client
connection.
```

回答

当网络不稳定时，会出现上述问题。当 beeline 出现 timed-out 异常时，Spark 不会尝试重连。相反，用户需要通过重新启动 spark-beeline 进行重连。

25.8.2.24 Spark SQL 无法查询到 ORC 类型的 Hive 表的新插入数据

问题

为什么通过 Spark SQL 无法查询到存储类型为 ORC 的 Hive 表的新插入数据？主要有以下两种场景存在这个问题：

- 对于分区表和非分区表，在 Hive 客户端中执行插入数据的操作后，会出现 Spark SQL 无法查询到最新插入的数据的问题。
- 对于分区表，在 Spark SQL 中执行插入数据的操作后，如果分区信息未改变，会出现 Spark SQL 无法查询到最新插入的数据的问题。

回答

由于 Spark 存在一个机制，为了提高性能会缓存 ORC 的元数据信息。当通过 Hive 或其他方式更新了 ORC 表时，缓存的元数据信息未更新，导致 Spark SQL 查询不到新插入的数据。

对于存储类型为 ORC 的 Hive 分区表，在执行插入数据操作后，如果分区信息未改变，则缓存的元数据信息未更新，导致 Spark SQL 查询不到新插入的数据。

解决措施：

1. 在使用 Spark SQL 查询之前，需执行 Refresh 操作更新元数据信息：

   REFRESH TABLE *table_name***;**

   table_name 为刷新的表名，该表必须存在，否则会出错。

   执行查询语句时，即可获取到最新插入的数据。

2. 使用 sqark 时，执行以下命令禁用 Spark 优化：

   set spark.sql.hive.convertMetastoreOrc=false;

25.8.3 Spark Streaming

25.8.3.1 Spark Streaming 任务一直阻塞

问题

运行一个 Spark Streaming 任务，确认有数据输入后，发现没有任何处理的结果。打开 Web 界面查看 Spark Job 执行情况，发现如下图所示：有两个 Job 一直在等待运行，但一直无法成功运行。

图25-16 Active Jobs

Active Jobs (2)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total
3	print at test2StreamFromKafka.scala:31	2015/05/25 18:28:55	63.7 h	0/3
2	start at test2StreamFromKafka.scala:34	2015/05/25 18:28:55	63.7 h	0/1

继续查看已经完成的 Job，发现也只有两个，说明 Spark Streaming 都没有触发数据计算的任务（Spark Streaming 默认有两个尝试运行的 Job，就是图中两个）

图25-17 Completed Jobs

Completed Jobs (2)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total
1	print at test2StreamFromKafka.scala:31	2015/05/25 18:28:55	0.7 s	2/2 (1 skipped)
0	start at test2StreamFromKafka.scala:34	2015/05/25 18:28:54	1 s	2/2

回答

经过定位发现，导致这个问题的原因是：Spark Streaming 的计算核数少于 Receiver 的个数，导致部分 Receiver 启动以后，系统已经没有资源去运行计算任务，导致第一个任务一直在等待，后续任务一直在排队。从现象上看，就是如问题中的图 25-16 中所示，会有两个任务一直在等待。

因此，当 Web 出现两个任务一直在等待的情况，首先检查 Spark 的核数是否大于 Receiver 的个数。

说明

Receiver 在 Spark Streaming 中是一个常驻的 Spark Job，Receiver 对于 Spark 是一个普通的任务，但它的生命周期和 Spark Streaming 任务相同，并且占用一个核的计算资源。

在调试和测试等经常使用默认配置的场景下，要时刻注意核数与 Receiver 个数的关系。

25.8.3.2 运行 Spark Streaming 任务参数调优的注意事项

问题

运行 Spark Streaming 任务时，随着 executor 个数的增长，数据处理性能没有明显提升，对于参数调优有哪些注意事项？

回答

在 executor 核数等于 1 的情况下，遵循以下规则对调优 Spark Streaming 运行参数有所帮助。

- Spark 任务处理速度和 Kafka 上 partition 个数有关，当 partition 个数小于给定 executor 个数时，实际使用的 executor 个数和 partition 个数相同，其余的将会被空闲。所以应该使得 executor 个数小于或者等于 partition 个数。
- 当 Kafka 上不同 partition 数据有倾斜时，数据较多的 partition 对应的 executor 将成为数据处理的瓶颈，所以在执行 Producer 程序时，数据平均发送到每个 partition 可以提升处理的速度。
- 在 partition 数据均匀分布的情况下，同时提高 partition 和 executor 个数，将会提升 Spark 处理速度（当 partition 个数和 executor 个数保持一致时，处理速度是最快的）。
- 在 partition 数据均匀分布的情况下，尽量保持 partition 个数是 executor 个数的整数倍，这样将会使资源得到合理利用。

25.8.3.3 为什么提交 Spark Streaming 应用超过 token 有效期，应用失败

问题

修改 kerberos 的票据和 HDFS token 过期时间为 5 分钟，设置“dfs.namenode.delegation.token.renew-interval”小于 60 秒，提交 Spark Streaming 应用，超过 token 有效期，提示以下错误，应用失败。

```
token (HDFS_DELEGATION_TOKEN token 17410 for spark2x) is expired
```

回答

- 问题原因：

  ApplicationMaster 进程中有 1 个 Credential Refresh Thread 会根据 *token renew 周期 * 0.75* 的时间比例上传更新后的 Credential 文件到 HDFS 上。

  Executor 进程中有 1 个 Credential Refresh Thread 会根据 *token renew 周期 *0.8* 的时间比例去 HDFS 上获取更新后的 Credential 文件，用来刷新 UserGroupInformation 中的 token，避免 token 失效。

  当 Executor 进程的 Credential Refresh Thread 发现当前时间已经超过 Credential 文件更新时间（即 *token renew 周期 *0.8*）时，会等待 1 分钟再去 HDFS 上面获取最新的 Credential 文件，以确保 AM 端已经将更新后的 Credential 文件放到 HDFS 上。

  当“dfs.namenode.delegation.token.renew-interval”配置值小于 60 秒，Executor 进程起来时发现当前时间已经超过 Credential 文件更新时间，等待 1 分钟再去 HDFS

上面获取最新的 Credential 文件，而此时 token 已经失效，task 运行失败，然后在其他 Executor 上重试，由于重试时间都是在 1 分钟内完成，所以 task 在其他 Executor 上也运行失败，导致运行失败的 Executor 加入到黑名单，没有可用的 Executor，应用退出。

- 修改方案：

  在 Spark 使用场景下，需设置“dfs.namenode.delegation.token.renew-interval”大于 80 秒。“dfs.namenode.delegation.token.renew-interval”参数描述请参表 25-84 考。

表25-84 参数说明

参数	描述	默认值
dfs.namenode.delegation.token.renew-interval	该参数为服务器端参数，设置 token renew 的时间间隔，单位为毫秒。	86400000

25.8.3.4 为什么 Spark Streaming 应用创建输入流，但该输入流无输出逻辑时，应用从 checkpoint 恢复启动失败

问题

Spark Streaming 应用创建 1 个输入流，但该输入流无输出逻辑。应用从 checkpoint 恢复启动失败，报错如下：

```
17/04/24 10:13:57 ERROR Utils: Exception encountered
java.lang.NullPointerException
at
org.apache.spark.streaming.dstream.DStreamCheckpointData$$anonfun$writeObject$1.app
ly$mcV$sp(DStreamCheckpointData.scala:125)
at
org.apache.spark.streaming.dstream.DStreamCheckpointData$$anonfun$writeObject$1.app
ly(DStreamCheckpointData.scala:123)
at
org.apache.spark.streaming.dstream.DStreamCheckpointData$$anonfun$writeObject$1.app
ly(DStreamCheckpointData.scala:123)
at org.apache.spark.util.Utils$.tryOrIOException(Utils.scala:1195)
at
org.apache.spark.streaming.dstream.DStreamCheckpointData.writeObject(DStreamCheckpo
intData.scala:123)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:4
3)
at java.lang.reflect.Method.invoke(Method.java:498)
at java.io.ObjectStreamClass.invokeWriteObject(ObjectStreamClass.java:1028)
at java.io.ObjectOutputStream.writeSerialData(ObjectOutputStream.java:1496)
at java.io.ObjectOutputStream.writeOrdinaryObject(ObjectOutputStream.java:1432
at java.io.ObjectOutputStream.writeObject0(ObjectOutputStream.java:1178)
at java.io.ObjectOutputStream.defaultWriteFields(ObjectOutputStream.java:1548)
```

```
at java.io.ObjectOutputStream.defaultWriteObject(ObjectOutputStream.java:441)
at
org.apache.spark.streaming.dstream.DStream$$anonfun$writeObject$1.apply$mcV$sp(DStr
eam.scala:515)
at
org.apache.spark.streaming.dstream.DStream$$anonfun$writeObject$1.apply(DStream.sca
la:510)
at
org.apache.spark.streaming.dstream.DStream$$anonfun$writeObject$1.apply(DStream.sca
la:510)
at org.apache.spark.util.Utils$.tryOrIOException(Utils.scala:1195)
at org.apache.spark.streaming.dstream.DStream.writeObject(DStream.scala:510)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:4
3)
at java.lang.reflect.Method.invoke(Method.java:498)
at java.io.ObjectStreamClass.invokeWriteObject(ObjectStreamClass.java:1028)
at java.io.ObjectOutputStream.writeSerialData(ObjectOutputStream.java:1496)
at java.io.ObjectOutputStream.writeOrdinaryObject(ObjectOutputStream.java:1432
at java.io.ObjectOutputStream.writeObject0(ObjectOutputStream.java:1178)
at java.io.ObjectOutputStream.writeArray(ObjectOutputStream.java:1378)
at java.io.ObjectOutputStream.writeObject0(ObjectOutputStream.java:1174)
at java.io.ObjectOutputStream.defaultWriteFields(ObjectOutputStream.java:1548)
at java.io.ObjectOutputStream.writeSerialData(ObjectOutputStream.java:1509)
at java.io.ObjectOutputStream.writeOrdinaryObject(ObjectOutputStream.java:1432
at java.io.ObjectOutputStream.writeObject0(ObjectOutputStream.java:1178)
at java.io.ObjectOutputStream.defaultWriteFields(ObjectOutputStream.java:1548)
at java.io.ObjectOutputStream.defaultWriteObject(ObjectOutputStream.java:441)
at
org.apache.spark.streaming.DStreamGraph$$anonfun$writeObject$1.apply$mcV$sp(DStream
Graph.scala:191)
at
org.apache.spark.streaming.DStreamGraph$$anonfun$writeObject$1.apply(DStreamGraph.s
cala:186)
at
org.apache.spark.streaming.DStreamGraph$$anonfun$writeObject$1.apply(DStreamGraph.s
cala:186)
at org.apache.spark.util.Utils$.tryOrIOException(Utils.scala:1195)
at org.apache.spark.streaming.DStreamGraph.writeObject(DStreamGraph.scala:186
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:4
3)
at java.lang.reflect.Method.invoke(Method.java:498)
at java.io.ObjectStreamClass.invokeWriteObject(ObjectStreamClass.java:1028)
at java.io.ObjectOutputStream.writeSerialData(ObjectOutputStream.java:1496)
at java.io.ObjectOutputStream.writeOrdinaryObject(ObjectOutputStream.java:1432
at java.io.ObjectOutputStream.writeObject0(ObjectOutputStream.java:1178)
at java.io.ObjectOutputStream.defaultWriteFields(ObjectOutputStream.java:1548)
at java.io.ObjectOutputStream.writeSerialData(ObjectOutputStream.java:1509)
at java.io.ObjectOutputStream.writeOrdinaryObject(ObjectOutputStream.java:1432
at java.io.ObjectOutputStream.writeObject0(ObjectOutputStream.java:1178)
```

```
at java.io.ObjectOutputStream.writeObject(ObjectOutputStream.java:348)
at
org.apache.spark.streaming.Checkpoint$$anonfun$serialize$1.apply$mcV$sp(Checkpoint.
scala:142)
at
org.apache.spark.streaming.Checkpoint$$anonfun$serialize$1.apply(Checkpoint.scala:1
42)
at
org.apache.spark.streaming.Checkpoint$$anonfun$serialize$1.apply(Checkpoint.scala:1
42)
at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1230)
at org.apache.spark.streaming.Checkpoint$.serialize(Checkpoint.scala:143)
at org.apache.spark.streaming.StreamingContext.validate(StreamingContext.scala:566)
at
org.apache.spark.streaming.StreamingContext.liftedTree1$1(StreamingContext.scala:61
2)
at org.apache.spark.streaming.StreamingContext.start(StreamingContext.scala:611)
at com.spark.test.kafka08LifoTwoInkfk$.main(kafka08LifoTwoInkfk.scala:21)
at com.spark.test.kafka08LifoTwoInkfk.main(kafka08LifoTwoInkfk.scala)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:4
3)
at java.lang.reflect.Method.invoke(Method.java:498)
at
org.apache.spark.deploy.SparkSubmit$.org$apache$spark$deploy$SparkSubmit$$runMain(S
parkSubmit.scala:772)
at org.apache.spark.deploy.SparkSubmit$.doRunMain$1(SparkSubmit.scala:183)
at org.apache.spark.deploy.SparkSubmit$.submit(SparkSubmit.scala:208)
at org.apache.spark.deploy.SparkSubmit$.main(SparkSubmit.scala:123)
at org.apache.spark.deploy.SparkSubmit.main(SparkSubmit.scala)
```

回答

Streaming Context 启动时，若应用设置了 checkpoint，则需要对应用中的 DStream checkpoint 对象进行序列化，序列化时会用到 dstream.context。

dstream.context 是 Streaming Context 启动时从 output Streams 反向查找所依赖的 DStream，逐个设置 context。若 Spark Streaming 应用创建 1 个输入流，但该输入流无输出逻辑时，则不会给它设置 context。所以在序列化时报“NullPointerException”。

解决办法：应用中如果有无输出逻辑的输入流，则在代码中删除该输入流，或添加该输入流的相关输出逻辑。

25.8.3.5 Spark Streaming 应用运行过程中重启 Kafka，Web UI 界面部分 batch time 对应 Input Size 为 0 records

问题

在 Spark Streaming 应用执行过程中重启 Kafka 时，应用无法从 Kafka 获取 topic offset，从而导致生成 Job 失败。如图 25-18 所示，其中 2017/05/11 10:57:00~2017/05/11

10:58:00 为 Kafka 重启时间段。2017/05/11 10:58:00 重启成功后对应的“Input Size”的值显示为“0 records”。

图25-18 Web UI 界面部分 batch time 对应 Input Size 为 0 records

Completed Batches (last 9 out of 9)

Batch Time	Input Size	Scheduling Delay (?)	Processing Time (?)	Total Delay (?)	Output Ops: Succeeded/Total
2017/05/11 10:58:50	18 records	0 ms	0.4 s	0.4 s	1/1
2017/05/11 10:58:40	20 records	4 s	0.3 s	4 s	1/1
2017/05/11 10:58:30	20 records	14 s	0.5 s	14 s	1/1
2017/05/11 10:58:20	20 records	23 s	0.4 s	24 s	1/1
2017/05/11 10:58:10	20 records	33 s	0.5 s	33 s	1/1
2017/05/11 10:58:00	0 records	6 ms	43 s	43 s	1/1
2017/05/11 10:57:00	19 records	1 ms	0.9 s	0.9 s	1/1
2017/05/11 10:56:50	20 records	1 ms	0.6 s	0.6 s	1/1
2017/05/11 10:56:40	28 records	13 ms	5 s	5 s	1/1

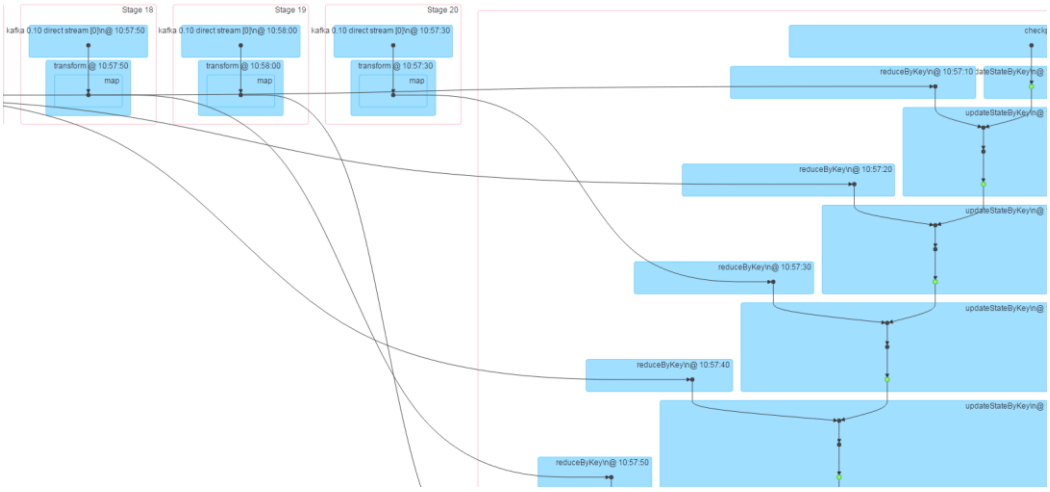
回答

Kafka 重启成功后应用会按照 batch 时间把 2017/05/11 10:57:00~2017/05/11 10:58:00 缺失的 RDD 补上（如图 25-19 所示），尽管 UI 界面上显示读取的数据个数为“0”，但实际上这部分数据在补的 RDD 中进行了处理，因此，不存在数据丢失。

Kafka 重启时间段的数据处理机制如下。

Spark Streaming 应用使用了 state 函数（例如：updateStateByKey），在 Kafka 重启成功后，Spark Streaming 应用生成 2017/05/11 10:58:00 batch 任务时，会按照 batch 时间把 2017/05/11 10:57:00~2017/05/11 10:58:00 缺失的 RDD 补上（Kafka 重启前 Kafka 上未读取完的数据，属于 2017/05/11 10:57:00 之前的 batch），如图 25-19 所示。

图25-19 重启时间段缺失数据处理机制



25.8.4 访问 Spark 应用获取的 restful 接口信息有误

问题

当 Spark 应用结束后，访问该应用的 restful 接口获取 job 信息，发现 job 信息中“numActiveTasks”的值是负数，如图 25-20 所示。

图25-20 job 信息

说明

numActiveTasks 是指当前正在运行 task 的个数。

回答

通过下面两种途径获取上面的 job 信息：

- 配置 **spark.history.briefInfo.gather=true**，查看 JobHistory 的 brief 信息。
- 使用 Spark JobHistory2x 页面访问：https://IP:port/api/v1/<appid>/jobs/。

job 信息中“numActiveTasks”的值是根据 eventlog 文件中 SparkListenerTaskStart 和 SparkListenerTaskEnd 事件的个数的差值计算得到的。如果 eventLog 文件中有事件丢失，就可能出现上面的现象。

25.8.5 为什么从 Yarn Web UI 页面无法跳转到 Spark Web UI 界面

问题

FusionInsight 版本中，在客户端采用 yarn-client 模式运行 Spark 应用，然后从 Yarn 的页面打开该应用的 Web UI 界面，出现下面的错误：

Error Occurred.

Problem accessing /proxy/application_1468986660719_0045/

Powered by Jetty://

从 YARN ResourceManager 的日志看到：

```
2016-07-21 16:35:27,099 | INFO  | Socket Reader #1 for port 8032 | Auth successful
for mapred/hadoop.<系统域名>@<系统域名> (auth:KERBEROS) | Server.java:1388
2016-07-21 16:35:27,105 | INFO  | 1526016381@qtp-1178290888-1015 | admin is
accessing unchecked http://10.120.169.53:23011 which is the app master GUI of
application_1468986660719_0045 owned by spark | WebAppProxyServlet.java:393
2016-07-21 16:36:02,843 | INFO  | Socket Reader #1 for port 8032 | Auth successful
for hive/hadoop.<系统域名>@<系统域名> (auth:KERBEROS) | Server.java:1388
2016-07-21 16:36:02,851 | INFO  | Socket Reader #1 for port 8032 | Auth successful
for hive/hadoop.<系统域名>@<系统域名> (auth:KERBEROS) | Server.java:1388
2016-07-21 16:36:12,163 | WARN  | 1526016381@qtp-1178290888-1015 |
/proxy/application_1468986660719_0045/: java.net.ConnectException: Connection timed
out |
Slf4jLog.java:76
2016-07-21 16:37:03,918 | INFO  | Socket Reader #1 for port 8032 | Auth successful
for hive/hadoop.<系统域名>@<系统域名> (auth:KERBEROS) | Server.java:1388
2016-07-21 16:37:03,926 | INFO  | Socket Reader #1 for port 8032 | Auth successful
for hive/hadoop.<系统域名>@<系统域名> (auth:KERBEROS) | Server.java:1388
2016-07-21 16:37:11,956 | INFO  | AsyncDispatcher event handler | Updating
application attempt appattempt_1468986660719_0045_000001 with final state:
FINISHING,
and exit status: -1000 | RMAppAttemptImpl.java:1253
```

回答

打开 FusionInsight Manager 页面，看到 Yarn 服务的业务 IP 地址为 192 网段。

从 Yarn 的日志看到，Yarn 读取的 Spark Web UI 地址为 http://10.120.169.53:23011，是 10 网段的 IP 地址。由于 192 网段的 IP 和 10 网段的 IP 不能互通，所以导致访问 Spark Web UI 界面失败。

修改方案：

登录 10.120.169.53 客户端机器，修改/etc/hosts 文件，将 10.120.169.53 更改为相对应的 192 网段的 IP 地址。再重新运行 Spark 应用，这时就可以打开 Spark Web UI 界面。

25.8.6 HistoryServer 缓存的应用被回收，导致此类应用页面访问时出错

问题

在 History Server 页面中访问某个 Spark 应用的页面时，发现访问时出错。

查看相应的 HistoryServer 日志后，发现有“FileNotFound”异常，相关日志如下所示：

```
2016-11-22 23:58:03,694 | WARN  | [qtp55429210-232] |
/history/application_1479662594976_0001/stages/stage/ |
org.sparkproject.jetty.servlet.ServletHandler.doHandle(ServletHandler.java:628)
```

```
java.io.FileNotFoundException: ${BIGDATA_HOME}/tmp/spark/jobHistoryTemp/blockmgr-5f1f6aca-2303-4290-9845-88fa94d78480/09/temp_shuffle_11f82aaf-e226-46dc-b1f0-002751557694 (No such file or directory)
```

回答

在 History Server 页面加载 Task 个数较多的 Spark 应用时，由于无法把全部的数据放入内存中，导致数据溢出到磁盘时，会产生前缀为“temp_shuffle”的文件。

HistoryServer 默认会缓存 50 个 Spark 应用（由配置项“spark.history.retainedApplications”决定），当内存中的 Spark 应用个数超过这个数值时，HistoryServer 会回收最先缓存的 Spark 应用，同时会清理掉相应的“temp_shuffle”文件。

当用户正在查看即将被回收的 Spark 应用时，可能会出现找不到“temp_shuffle”文件的错误，从而导致当前页面无法访问。

如果遇到上述问题，可参考以下两种方法解决。

- 重新访问这个 Spark 应用的 HistoryServer 页面，即可查看到正确的页面信息。
- 如果用户场景需要同时访问 50 个以上的 Spark 应用时，需要调大“spark.history.retainedApplications”参数的值。

  请登录 FusionInsight Manager 管理界面，单击“集群 > *待操作集群的名称* > 服务 > Spark2x > 配置”，单击“全部配置”，在左侧的导航列表中，单击“JobHistory2x > 界面”，配置如下参数。

表25-85 参数说明

参数	描述	默认值
spark.history.retainedApplications	HistoryServer 缓存的 Spark 应用数，当需要缓存的应用个数超过此参数值时，HistoryServer 会回收最先缓存的 Spark 应用。	50

25.8.7 加载空的 part 文件时，app 无法显示在 JobHistory 的页面上

问题

在分组模式下执行应用，当 HDFS 上的 part 文件为空时，发现 JobHistory 首页面上不显示该 part 对应的 app。

回答

JobHistory 服务更新页面上的 app 时，会根据 HDFS 上的 part 文件大小变更与否判断是否刷新首页面的 app 显示信息。若文件为第一次查看，则将当前文件大小与 0 作比较，如果大于 0 则读取该文件。

分组的情况下，如果执行的 app 没有 job 处于执行状态，则 part 文件为空，即 JobHistory 服务不会读取该文件，此 app 也不会显示在 JobHistory 页面上。但若 part 文件大小之后有更新，JobHistory 又会显示该 app。

25.8.8 Spark2x 导出带有相同字段名的表，结果导出失败

问题

在 Spark2x 的 spark-shell 上执行如下代码失败：

```
val acctId = List(("49562", "Amal", "Derry"), ("00000", "Fred", "Xanadu"))
val rddLeft = sc.makeRDD(acctId)
val dfLeft = rddLeft.toDF("Id", "Name", "City")
//dfLeft.show
val acctCustId = List(("Amal", "49562", "CO"), ("Dave", "99999", "ZZ"))
val rddRight = sc.makeRDD(acctCustId)
val dfRight = rddRight.toDF("Name", "CustId", "State")
//dfRight.show
val dfJoin = dfLeft.join(dfRight, dfLeft("Id") === dfRight("CustId"), "outer")
dfJoin.show
dfJoin.repartition(1).write.format("com.databricks.spark.csv").option("delimiter",
"\t").option("header", "true").option("treatEmptyValuesAsNulls",
"true").option("nullValue", "").save("/tmp/outputDir")
```

回答

Spark2x 中对 join 语句重名字段做了判断，需要修改代码保证保存的数据中无重复字段。

25.8.9 为什么多次运行 Spark 应用程序会引发致命 JRE 错误

问题

为什么多次运行 Spark 应用程序会引发致命 JRE 错误？

回答

多次运行 Spark 应用程序会引发致命的 JRE 错误，这个错误由 Linux 内核导致。

升级内核版本到 4.13.9-2.ge7d7106-default 来解决这个问题。

25.8.10 IE 浏览器访问 Spark2x 原生 UI 界面失败，无法显示此页或者页面显示错误

问题

通过 IE 9、IE 10 和 IE 11 浏览器访问 Spark2x 的原生 UI 界面，出现访问失败情况或者页面显示错误问题。

现象

访问页面失败，浏览器无法显示此页，如下图所示：

原因

IE 9、IE 10、IE 11 浏览器的某些版本在处理 SSL 握手有问题导致访问失败。

解决方法

推荐使用 Google Chrome 浏览器 71 及其以上版本和 Firefox 浏览器 62 及其以上版本。

25.8.11 Spark2x 如何访问外部集群组件

问题

存在两个集群：cluster1 和 cluster2，如何使用 cluster1 中的 Spark2x 访问 cluster2 中的 HDFS、Hive、HBase 和 Kafka 组件。

回答

1. 可以有条件的实现两个集群间组件互相访问，但是存在以下限制：
   - 仅允许访问一个 Hive MetaStore，不支持同时访问 cluster1 的 Hive MetaStore 和 cluster2 的 Hive MetaStore。
   - 不同集群的用户系统没有同步，因此访问跨集群组件时，用户的权限管理由对端集群的用户配置决定。比如 cluster1 的 userA 没有访问本集群 HBase meta 表权限，但是 cluster2 的 userA 有访问该集群 HBase meta 表权限，则 cluster1 的 userA 可以访问 cluster2 的 HBase meta 表。
   - 跨 Manager 之间的安全集群间组件互相访问，需要先配置系统互信。
2. 以下分别阐述 cluster1 上使用 userA 访问 cluster2 的 Hive、HBase、Kafka 组件。

   说明

   以下操作皆以用户使用 FusionInsight 客户端提交 Spark2x 应用为基础，若用户使用了自己的配置文件目录，则需要修改本应用配置目录中的对应文件，并注意需要将配置文件上传到 executor 端。

   由于 hdfs 和 hbase 客户端访问服务端时，使用 hostname 配置服务端地址，因此，客户端的 /etc/hosts 需要保存有所有需要访问节点的 hosts 配置。用户可预先将对端集群节点的 host 添加到客户端节点的/etc/hosts 文件中。

   - 访问 Hive MetaStore：使用 cluster2 中的 Spark2x 客户端下“conf”目录下的 hive-site.xml 文件，替换到 cluster1 中的 Spark2x 客户端下“conf”目录下的 hive-site.xml 文件。

如上操作后可以用 sparksql 访问 hive MetaStore，如需访问 hive 表数据，需要按照•同时访问两个集群的 HDFS：的操作步骤配置且指定对端集群 nameservice 为 LOCATION 后才能访问表数据。

- 访问对端集群的 HBase：
  i. 先将 cluster2 集群的所有 Zookeeper 节点和 HBase 节点的 IP 和主机名配置到 cluster1 集群的客户端节点的/etc/hosts 文件中。
  ii. 使用 cluster2 中的 Spark2x 客户端下“conf”目录的 hbase-site.xml 文件，替换到 cluster1 中的 Spark2x 客户端下“conf”目录 hbase-site.xml 文件。
- 访问 Kafka，仅需将应用访问的 Kafka Broker 地址设置为 cluster2 中的 Kafka Broker 地址即可。
- 同时访问两个集群的 HDFS：
  - 无法同时获取两个相同 nameservice 的 token，因此两个 HDFS 的 nameservice 必须不同，例如：一个为 hacluster，一个为 test
    1) 从 cluster2 的 hdfs-site.xml 中获取以下配置，添加到 cluster1 的 spark2x 客户端 conf 目录的 hdfs-site.xml 中

       dfs.nameservices.mappings、dfs.nameservices、dfs.namenode.rpc-address.test.*、dfs.ha.namenodes.test、dfs.client.failover.proxy.provider.test

       参考样例如下：

```
<property>
<name>dfs.nameservices.mappings</name>
<value>[{"name":"hacluster","roleInstances":["14","15"]},{"name":
"test","roleInstances":["16","17"]}]</value>
</property>
<property>
<name>dfs.nameservices</name>
<value>hacluster,test</value>
</property>
<property>
<name>dfs.namenode.rpc-address.test.16</name>
<value>192.168.0.1:8020</value>
</property>
<property>
<name>dfs.namenode.rpc-address.test.17</name>
<value>192.168.0.2:8020</value>
</property>
<property>
<name>dfs.ha.namenodes.test</name>
<value>16,17</value>
</property>
<property>
<name>dfs.client.failover.proxy.provider.test</name>
<value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailov
erProxyProvider</value>
</property>
```

    2) 修改 cluster1 的 spark 客户端 conf 目录下的 spark-defaults.conf 配置文件中，修改 spark.yarn.extra.hadoopFileSystems = hdfs://test，spark.hadoop.hdfs.externalToken.enable = true，如下所示：

```
spark.yarn.extra.hadoopFileSystems = hdfs://test
spark.hadoop.hdfs.externalToken.enable = true
```

3) 应用提交命令中，需要添加--keytab 和 --principal 参数，参数配置为 cluster1 中提交任务的用户。

4) 使用 cluster1 的 spark 客户端提交应用，即可同时访问两个 hdfs 服务

- 同时访问两个集群的 HBase：

i. 修改 cluster1 的 spark 客户端 conf 目录下的 spark-defaults.conf 配置文件中，修改 spark.hadoop.hbase.externalToken.enable = true，如下所示：

```
spark.hadoop.hbase.externalToken.enable = true
```

ii. 用户访问 HBase 时，需要使用对应集群的配置文件创建 Configuration 对象，用于创建 Connection 对象。

iii. MRS 集群中支持同时获取多个 HBase 服务的 token，以解决 Executor 中无法访问 HBase 的问题，使用方式如下：

假设需要访问本集群的 HBase 和 cluster2 的 HBase，将 cluster2 的 hbase-site.xml 文件放到一个压缩包内，压缩包命名为 external_hbase_conf***，提交命令时，使用--archives 指定这些压缩包。

25.8.12 对同一目录创建多个外表，可能导致外表查询失败

问题

假设存在数据文件路径“/test_data_path”，用户 userA 对该目录创建外表 tableA，用户 userB 对该目录创建外表 tableB，当 userB 对 tableB 执行 insert 操作后，userA 将查询 tableA 失败，出现 Permission denied 异常。

回答

当 userB 对 tableB 执行 insert 操作后，会在外表数据路径下生成新的数据文件，且文件属组是 userB，当 userA 查询 tableA 时，会读取外表数据目录下的所有的文件，此时会因没有 userB 生成的文件的读取权限而查询失败。

实际上，不只是查询场景，还有其他场景也会出现问题。例如：inset overwrite 操作将会把此目录下的其他表文件也一起复写。

由于 Spark SQL 当前的实现机制，如果对此种场景添加检查限制，会存在一致性问题和性能问题，因此未对此种场景添加限制，但是用户应避免此种用法，以避免此场景带来的各种问题。

25.8.13 访问 Spark2x JobHistory 中某个应用的原生页面时页面显示错误

问题

提交一个 Spark 应用，包含单个 Job 百万个 task。应用结束后，在 JobHistory 中访问该应用的原生页面，浏览器会等待较长时间才跳转到应用原生页面，若 10 分钟内无法跳转，则页面会显示 Proxy Error 信息。

图25-21 错误信息样例

回答

在 JobHistory 界面中跳转到某个应用的原生页面时，JobHistory 需要回放该应用的 Event log，若应用包含的事件日志较大，则回放时间较长，浏览器需要较长时间的等待。

当前浏览器访问 JobHistory 原生页面需经过 httpd 代理，代理的超时时间是 10 分钟，因此，如果 JobHistory 在 10 分钟内无法完成 Event log 的解析并返回，httpd 会主动向浏览器返回 Proxy Error 信息。

解决方法

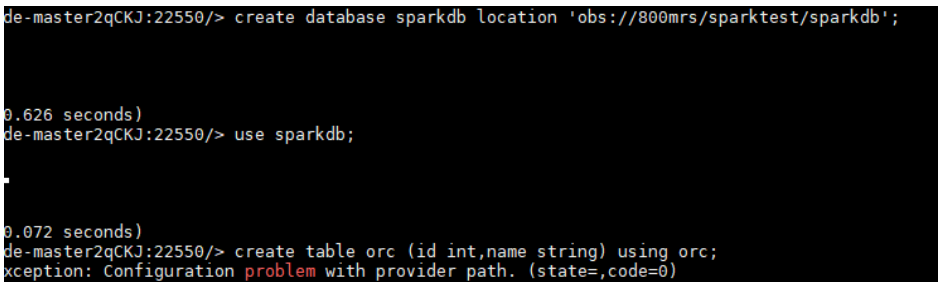
由于当前 JobHistory 开启了本地磁盘缓存功能，访问应用时，会将应用的 Event log 的解析结果缓存到本地磁盘中，第二次访问时，能大大加快响应速度。因此，出现此种情况时，仅需稍作等待，重新访问原来的链接即可，此时不会再出现需要长时间等待的现象。

25.8.14 对接 OBS 场景中，spark-beeline 登录后指定 loaction 到 OBS 建表失败

问题

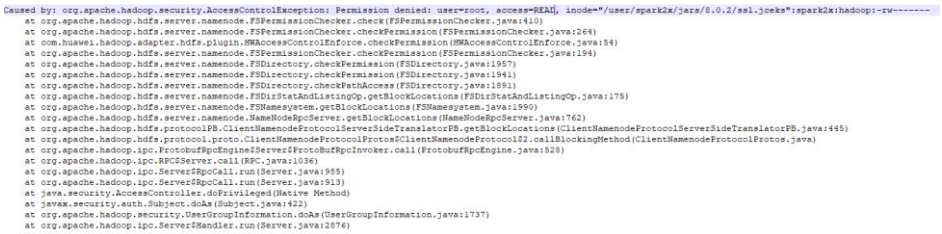
对接 OBS ECS/BMS 集群，spark-beeline 登录后，指定 location 到 OBS 建表报错失败。

图25-22 错误信息



回答

HDFS 上 ssl.jceks 文件权限不足，导致建表失败。



解决方法

1. 使用 **omm** 用户登录 Spark2x 所在节点，执行如下命令：

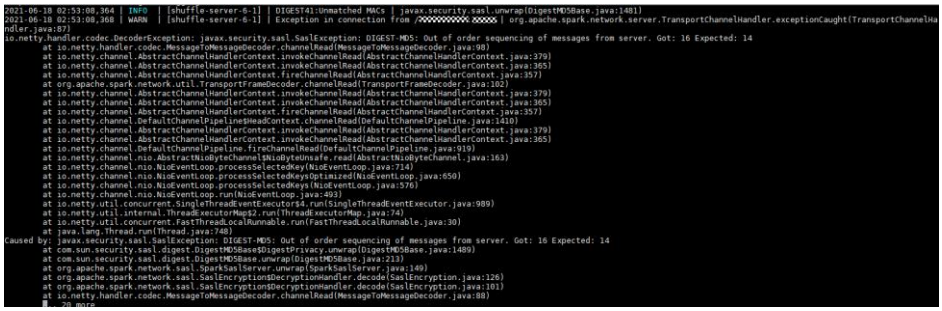
   vi ${BIGDATA_HOME}/FusionInsight_Spark2x_8.1.0.1/install/FusionInsight-Spark2x-3.1.1/spark/sbin/fake_prestart.sh

2. 将“eval "${hdfsCmd}" -chmod 600 "${InnerHdfsDir}"/ssl.jceks >> "${PRESTART_LOG}" 2>&1”修改成“eval "${hdfsCmd}" -chmod 644 "${InnerHdfsDir}"/ssl.jceks >> "${PRESTART_LOG}" 2>&1”。
3. 重启 SparkResource 实例。

25.8.15 Spark shuffle 异常处理

问题

在部分场景 Spark shuffle 阶段会有如下异常



解决方法

JDBC 应该：

登录 FusionInsight Manager 管理界面，修改 JDBCServer 的参数“spark.authenticate.enableSaslEncryption”值为“false”，并重启对应的实例。

客户端作业：

客户端应用在提交应用的时候，修改 spark-defaults.conf 配置文件的“spark.authenticate.enableSaslEncryption”值为“false”。

26 使用 Sqoop

26.1 从零开始使用 Sqoop

Sqoop 是一款开源的工具，主要用于在 Hadoop(Hive)与传统的数据库(MySQL、PostgreSQL...)间进行数据的传递，可以将一个关系型数据库（例如：MySQL、Oracle、PostgreSQL 等）中的数据导进到 Hadoop 的 HDFS 中，也可以将 HDFS 的数据导进到关系型数据库中。

前提条件

- MRS 3.1.0 及之后版本在创建集群时已勾选 Sqoop 组件。
- 安装客户端，具体请参考安装客户端（3.x 及之后版本）。例如安装目录为“/opt/client”，以下操作的客户端目录只是举例，请根据实际安装目录修改。

sqoop export (HDFS 到 MySQL)

步骤 1 登录客户端所在节点。

步骤 2 执行如下命令初始化环境变量。

source /opt/client/bigdata_env

步骤 3 使用 sqoop 命令操作 sqoop 客户端。

sqoop export --connect jdbc:mysql://10.100.231.134:3306/test --username root --password xxxxxx --table component13 -export-dir hdfs://hacluster/user/hive/warehouse/component_test3 --fields-terminated-by ',' -m 1

表26-1 参数说明

参数	说明
-direct	快速模式，利用了数据库的导入工具，如 MySQL 的 mysqlimport，可以比 jdbc 连接的方式更为高效的将数据导入到关系数据库中。
-export-dir <dir>	存放数据的 HDFS 的源目录。
-m 或-num-mappers <n>	启动 n 个 map 来并行导入数据，默认是 4 个，该值请勿高

参数	说明
	于集群的最大 Map 数。
-table <table-name>	要导入的目的关系数据库表。
-update-key <col-name>	后面接条件列名，通过该参数可以将关系数据库中已经存在的数据进行更新操作，类似于关系数据库中的 update 操作。
-update-mode <mode>	更新模式，有两个值 updateonly 和默认的 allowinsert，该参数只能在关系数据表里不存在要导入的记录时才能使用，比如要导入的 hdfs 中有一条 id=1 的记录，如果在表里已经有一条记录 id=2，那么更新会失败。
-input-null-string <null-string>	可选参数，如果没有指定，则字符串 null 将被使用。
-input-null-non-string <null-string>	可选参数，如果没有指定，则字符串 null 将被使用。
-staging-table <staging-table-name>	创建一个与导入目标表同样数据结构的表，将所有数据先存放在该表中，然后由该表通过一次事务将结果写入到目标表中。 该参数是用来保证在数据导入关系数据库表的过程中的事务安全性，因为在导入的过程中可能会有多个事务，那么一个事务失败会影响到其它事务，比如导入的数据会出现错误或出现重复的记录等等情况，那么通过该参数可以避免这种情况。
-clear-staging-table	如果该 staging-table 非空，则通过该参数可以在运行导入前清除 staging-table 里的数据。

----结束

sqoop import（MySQL 到 Hive 表）

步骤 1 登录客户端所在节点。

步骤 2 执行如下命令初始化环境变量。

source /opt/client/bigdata_env

步骤 3 使用 sqoop 命令操作 sqoop 客户端。

sqoop import --connect jdbc:mysql://10.100.231.134:3306/test --username root --password xxxxxx --table component --hive-import --hive-table component_test2 --delete-target-dir --fields-terminated-by "," -m 1 --as-textfile

表26-2 参数说明

参数	说明

参数	说明
-append	将数据追加到 hdfs 中已经存在的 dataset 中。使用该参数，sqoop 将把数据先导入到一个临时目录中，然后重新给文件命名到一个正式的目录中，以避免和该目录中已存在的文件重名。
-as-avrodatafile	将数据导入到一个 Avro 数据文件中。
-as-sequencefile	将数据导入到一个 sequence 文件中。
-as-textfile	将数据导入到一个普通文本文件中，生成该文本文件后，可以在 hive 中通过 sql 语句查询出结果。
-boundary-query <statement>	边界查询，在导入前先通过 SQL 查询得到一个结果集，然后导入的数据就是该结果集内的数据，格式如：**-boundary-query 'select id,creationdate from person where id = 3'**，表示导入的数据为 id=3 的记录，或者 **select min(<split-by>), max(<split-by>) from <table name>**。 注意：查询的字段中不能有数据类型为字符串的字段，否则会报错：java.sql.SQLException: Invalid value for getLong()。
-columns<col,col,col…>	指定要导入的字段值，格式如：-columns id,username
-direct	快速模式，利用了数据库的导入工具，如 MySQL 的 mysqlimport，可以比 jdbc 连接的方式更为高效的将数据导入到关系数据库中。
-direct-split-size	在使用上面 direct 直接导入的基础上，对导入的流按字节数分块，特别是使用直连模式从 PostgreSQL 导入数据时，可以将一个到达设定大小的文件分为几个独立的文件。
-inline-lob-limit	设定大对象数据类型的最大值。
-m 或-num-mappers	启动 n 个 map 来并行导入数据，默认是 4 个，该值请勿高于集群的最大 Map 数。
-query，-e<statement>	从查询结果中导入数据，该参数使用时必须指定-target-dir、-hive-table，在查询语句中一定要有 where 条件且在 where 条件中需要包含$CONDITIONS。 示例：-query 'select * from person where $CONDITIONS ' -target-dir /user/hive/warehouse/person -hive-table person
-split-by<column-name>	表的列名，用来切分工作单元，一般后面跟主键 ID。
-table <table-name>	关系数据库表名，数据从该表中获取。
-target-dir <dir>	指定 hdfs 路径。
-warehouse-dir <dir>	与**-target-dir** 不能同时使用，指定数据导入的存放目录，适用于导入 hdfs，不适合导入 hive 目录。

参数	说明
-where	从关系数据库导入数据时的查询条件，示例：-where 'id = 2'
-z,-compress	压缩参数，默认数据不压缩，通过该参数可以使用 gzip 压缩算法对数据进行压缩，适用于 SequenceFile，text 文本文件，和 Avro 文件。
–compression-codec	Hadoop 压缩编码，默认为 gzip。
–null-string <null-string>	替换 null 字符串，如果没有指定，则字符串 null 将被使用。
–null-non-string<null-string>	替换非 String 的 null 字符串，如果没有指定，则字符串 null 将被使用。
-check-column (col)	增量导入参数，用来作为判断的列名，如 id。
-incremental (mode) append 或 lastmodified	增量导入参数。 append：追加，比如对大于 last-value 指定的值之后的记录进行追加导入。 lastmodified：最后的修改时间，追加 last-value 指定的日期之后的记录。
-last-value (value)	增量导入参数，指定自从上次导入后列的最大值（大于该指定的值），也可以自己设定某一值。

----结束

Sqoop 使用样例

- sqoop import（MySQL 到 HDFS）

  sqoop import --connect jdbc:mysql://10.100.231.134:3306/test --username root --password *xxx* **--query 'SELECT * FROM component where $CONDITIONS and component_id ="MRS 1.0_002"' --target-dir /tmp/component_test --delete-target-dir --fields-terminated-by "," -m 1 --as-textfile**

- sqoop export （obs 到 MySQL）

  sqoop export --connect jdbc:mysql://10.100.231.134:3306/test --username root --password *xxx* **--table component14 -export-dir obs://obs-file-bucket/xx/part-m-00000 --fields-terminated-by ',' -m 1**

- sqoop import（MySQL 到 obs）

  sqoop import --connect jdbc:mysql://10.100.231.134:3306/test --username root --password *xxx* **--table component --target-dir obs://obs-file-bucket/xx --delete-target-dir --fields-terminated-by "," -m 1 --as-textfile**

- sqoop import（MySQL 到 Hive 外 obs 表）

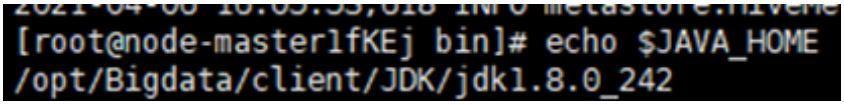
  sqoop import --connect jdbc:mysql://10.100.231.134:3306/test --username root --password *xxx* **--table component --hive-import --hive-table component_test01 --fields-terminated-by "," -m 1 --as-textfile**

26.2 Sqoop1.4.7 适配 MRS 3.x 集群

Sqoop 是专为 Apache Hadoop 和结构化数据库（如关系型数据库）设计的高效传输大量数据的工具。客户需要在 MRS 中使用 sqoop 进行数据迁移，MRS 旧版本中未自带 Sqoop，客户可参考此文档自行安装使用。MRS 3.1.0 及之后版本已支持创建集群时勾选 Sqoop 组件，请创建集群时勾选即可。

前提条件

已安装 MRS 客户端的节点，且已安装 jdk 环境。



Sqoop1.4.7 适配步骤

步骤 1 下载开源 sqoop-1.4.7.bin__hadoop-2.6.0.tar.gz 包（下载地址 http://archive.apache.org/dist/sqoop/1.4.7/）。

步骤 2 将下载好的 sqoop-1.4.7.bin__hadoop-2.6.0.tar.gz 包放入已安装 MRS 客户端的节点的“/opt/Bigdata/client”目录并解压。

tar zxvf sqoop-1.4.7.bin__hadoop-2.6.0.tar.gz

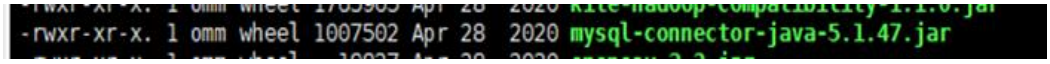
步骤 3 从 MySQL 官网下载 MySQL jdbc 驱动程序“mysql-connector-java-xxx.jar”，具体 MySQL jdbc 驱动程序选择参见下表。

表26-3 版本信息

jdbc 驱动程序版本	MySQL 版本
Connector/J 5.1	MySQL 4.1、MySQL 5.0、MySQL 5.1、MySQL 6.0 alpha
Connector/J 5.0	MySQL 4.1、MySQL 5.0 servers、distributed transaction (XA)
Connector/J 3.1	MySQL 4.1、MySQL 5.0 servers、MySQL 5.0 except distributed transaction (XA)
Connector/J 3.0	MySQL 3.x、MySQL 4.1

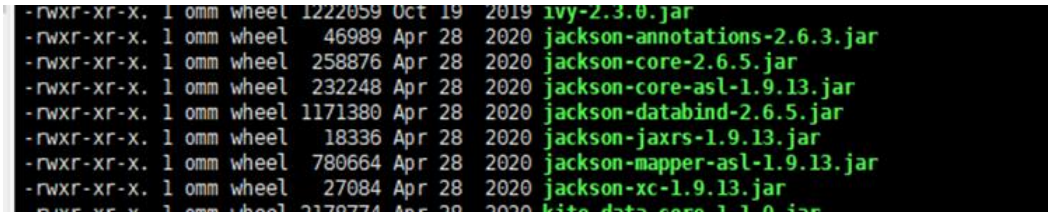
步骤 4 将 MySQL 驱动包放入 Sqoop 的 lib 目录下（/opt/Bigdata/client/sqoop-1.4.7.bin__hadoop-2.6.0/lib）并修改 jar 包的属组和权限，参考图 26-1 的 omm:wheel 和 755 的属组和权限。

图26-1 MySQL 驱动包的属组和权限



步骤 5 使用 MRS 客户端中 Hive 的 lib 目录下（/opt/Bigdata/client/Hive/Beeline/lib）的 jackson 开头的 jar 包替换 Sqoop 的 lib 下的相应 jar 包。

图26-2 jackson 开头的 jar



步骤 6 将 MRS Hive 客户端中（/opt/Bigdata/client/Hive/Beeline/lib）的 jline 的包，拷贝到 Sqoop 的 lib 下。

步骤 7 执行 **vim $JAVA_HOME/jre/lib/security/java.policy** 增加如下配置：

permission javax.management.MBeanTrustPermission "register";

步骤 8 执行如下命令，进入 Sqoop 的 conf 目录并增加配置：

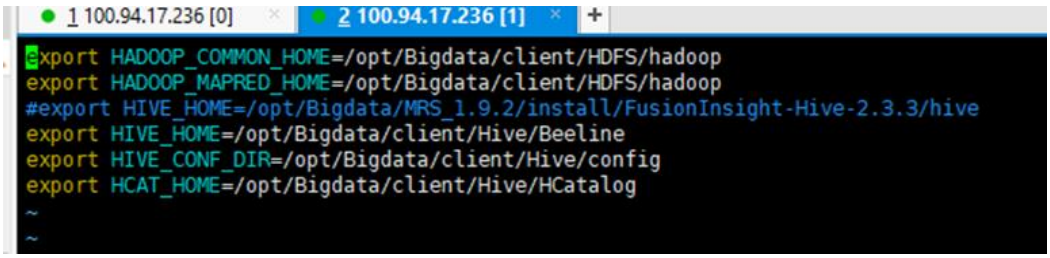
cd /opt/Bigdata/client/sqoop-1.4.7.bin__hadoop-2.6.0/conf

cp sqoop-env-template.sh sqoop-env.sh

步骤 9 执行 **vim sqoop-env.sh** 设置 Sqoop 的环境变量，Hadoop、Hive 的目录根据实际目录修改。

```
export HADOOP_COMMON_HOME=/opt/Bigdata/client/HDFS/hadoop
export HADOOP_MAPRED_HOME=/opt/Bigdata/client/HDFS/hadoop
export HIVE_HOME=/opt/Bigdata/MRS_1.9.X/install/FusionInsight-Hive-3.1.0/hive(请按照实际路径填写)
export HIVE_CONF_DIR=/opt/Bigdata/client/Hive/config
export HCAT_HOME=/opt/Bigdata/client/Hive/HCatalog
```

图26-3 设置 Sqoop 的环境变量



步骤 10 编写 Sqoop 脚本 例如：

```
/opt/Bigdata/FusionInsight_Current/1_19_SqoopClient/install/FusionInsight-Sqoop-
1.4.7/bin/sqoop import
--connect jdbc:mysql://192.168.0.183:3306/test
--driver com.mysql.jdbc.Driver
--username 'root'
--password 'xxx'
--query "SELECT id, name  FROM tbtest WHERE \$CONDITIONS"
--hcatalog-database default
--hcatalog-table test
--num-mappers 1
```

----结束

26.3 Sqoop 常用命令及参数介绍

Sqoop 常用命令介绍

表26-4 Sqoop 常用命令介绍

命令	说明
import	数据导入到集群
export	集群数据导出
codegen	获取数据库中某张表数据生成 Java 并打包 jar
create-hive-table	创建 Hive 表
eval	执行 sql 并查看结果
import-all-tables	导入某个数据库下的所有表到 HDFS 中
job	生成一个 sqoop 任务
list-databases	列举数据库名
list-tables	列举表名
merge	将 HDFS 不同目录下的数据合在一起并存放到指定目录
metastore	启动元数据库，记录 sqoop job 的元数据
help	打印帮助信息
version	打印版本信息

公用参数介绍

表26-5 公用参数介绍

分类	参数	说明
连接数据库	--connect	连接关系型数据库的 url
	--connection-manager	指定连接管理类
	--driver jdbc	连接驱动包
	--help	帮助信息
	--password	连接数据库密码
	--username	连接数据库的用户名
	--verbose	在控制台打印详细信息
import 参数	--fields-terminated-by	设定字段分隔符，和 Hive 表或 hdfs 文件保持一致
	--lines-terminated-by	设定行分隔符，和 hive 表或 hdfs 文件保持一致
	--mysql-delimiters	MySQL 默认分隔符设置
export 参数	--input-fields-terminated-by	字段分隔符
	--input-lines-terminated-by	行分隔符
hive 参数	--hive-delims-replacement	用自定义的字符替换数据中的\r\n 等字符
	--hive-drop-import-delims	在导入数据到 hive 时，去掉\r\n 等字符
	--map-column-hive	生成 hive 表时可以更改字段的数据类型
	--hive-partition-key	创建分区
	--hive-partition-value	导入数据库指定分区
	--hive-home	指定 hive 安装目录
	--hive-import	表示操作是从关系型数据库导入到 hive 中
	--hive-overwrite	覆盖 hive 已有数据
	--create-hive-table	创建 Hive 表，默认 false，如果目标表不存在，则会创建目标表

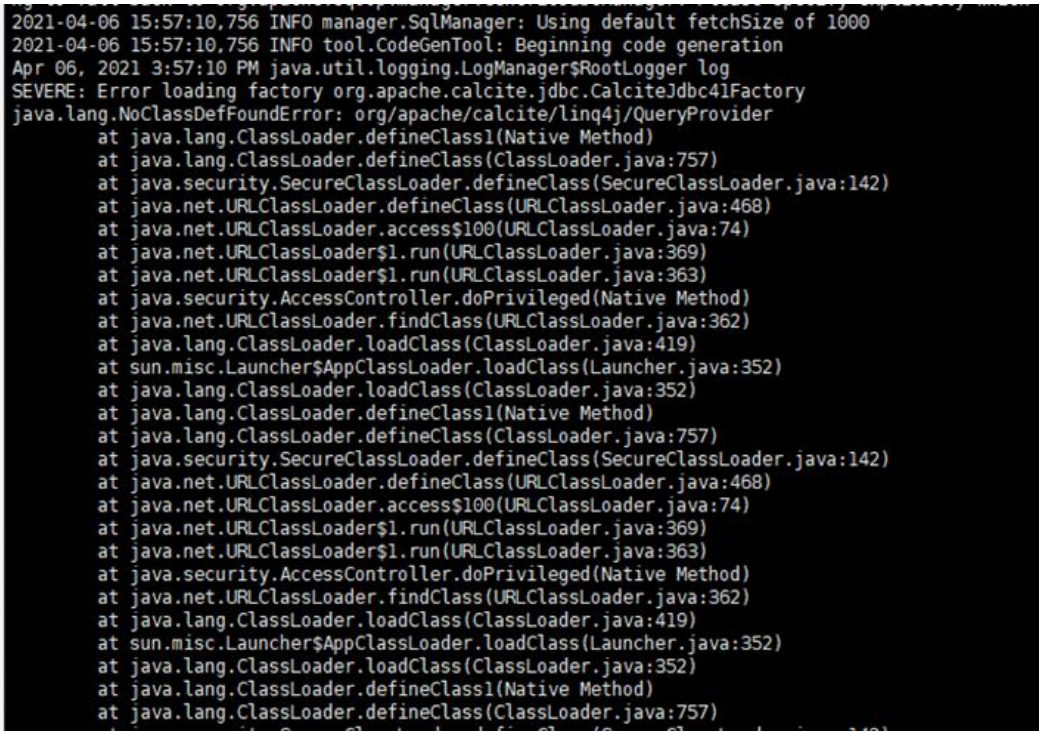
分类	参数	说明
	--hive-table	指定 hive 表
	--table	关系型数据库表名
	--columns	指定需要导入的关系型数据表字段
	--query	指定查询语句，将查询结果导入
hcatalog 参数	--hcatalog-database	指定 hive 库，使用 hcatalog 方式导入 hive 库
	--hcatalog-table	指定 hive 表，使用 hcatalog 方式导入 hive 表
其他参数	-m 或--num-mappers	后跟数字，表示 sqoop 任务的分片数
	--split-by	按照某一字段进行分片，配合-m
	--target-dir	指定 hdfs 临时目录
	--null-string string	类型为 null 时替换字符串
	--null-non-string	非 string 类型为 null 时替换字符串
	--check-column	增量判断的字段
	--incremental append 或 lastmodified	增量导入参数 append：追加，比如对大于 last-value 指定的值之后的记录进行追加导入。 lastmodified：最后的修改时间，追加 last-value 指定的日期之后的记录。
	--last-value	指定一个值，用于标记增量导入
	--input-null-string	替换 null 字符串，如果没有指定，则字符串 null 将被使用。
	--input-null-non-string	替换非 String 的 null 字符串，如果没有指定，则字符串 null 将被使用。

26.4 Sqoop 常见问题

26.4.1 报错找不到 QueryProvider 类

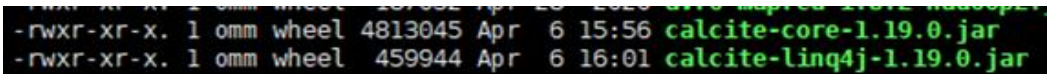
问题

报错找不到 QueryProvider 类。



回答

搜索 mrs 客户端目录，将以下两个 jar 包放入 sqoop 的 lib 目录下。



26.4.2 使用 hcatalog 方式同步数据，报错 getHiveClient 方法不存在

问题

使用 hcatalog 方式同步数据，报错 getHiveClient 方法不存在。

回答

将 https://repo.xxxcloud.com/repository/maven/xxxcloudsdk/org/apache/hive/hcatalog/hive-hcatalog-core/源下的 jar 包替换到 mrs 客户端的 hcatalog 的目录下，并重命名之前的同名 hcatalog 的 jar 包。如图 302002 就是替换后的包，310001-SNAPSHOT.jar.bak 就是加了.bak 后缀的原包。

图26-4 hcatalog 目录

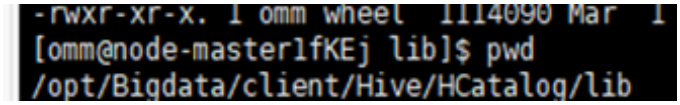
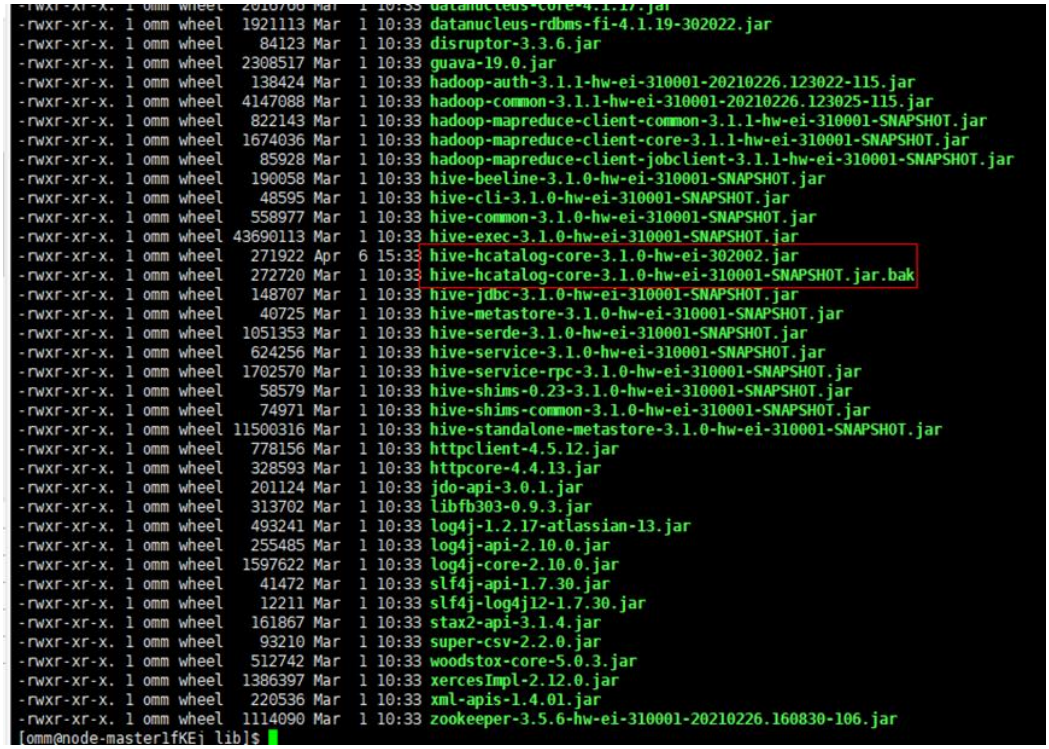


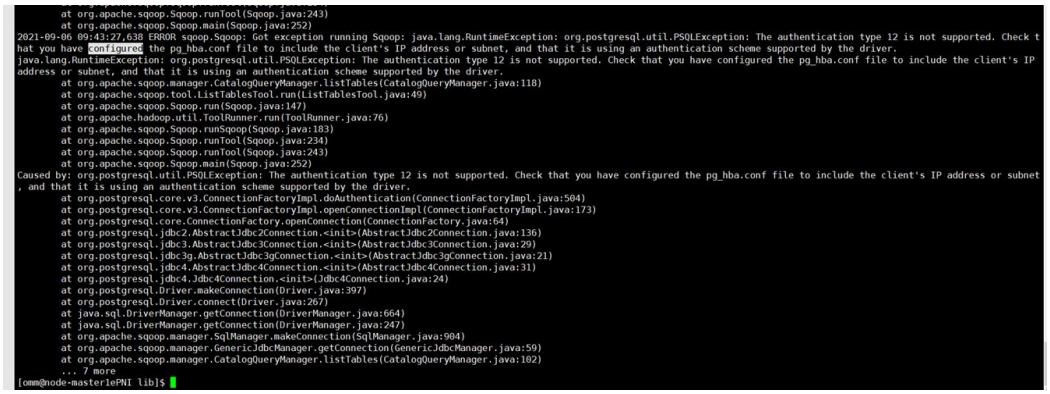
图26-5 替换 jar 包



26.4.3 连接 postgresql 或者 gaussdb 时报错

问题

连接 postgresql 或者 gaussdb 时报错。

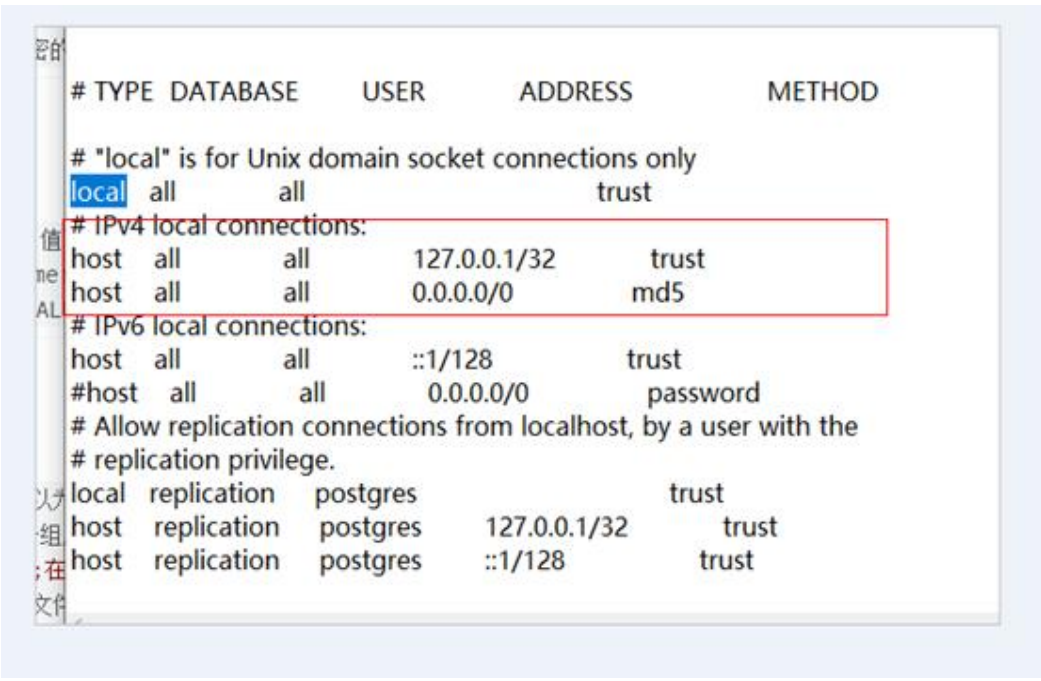


回答

- 场景一：（**import** 场景）使用 **sqoop import** 命令抽取开源 postgre 到 MRS hdfs 或 hive 等。
  - 问题现象：

    使用 sqoop 命令查询 postgre 表可以，但是执行 sqoop import 命令倒数时报错：
    - The authentication type 12 is not supported. Check that you have configured the pg_hba.conf file to include the client's IP address or subnet, and that it
    - The authentication type 5 is not supported. Check that you have configured the pg_hba.conf file to include the client's IP address or subnet, and that it
  - 问题根因：
    - 报错中 type 为 5 时：在执行 sqoop import 命令时，会启动 MapReduce 任务，由于 MRS Hadoop 安装目录（/opt/Bigdata/FusionInsight_HD_*/1_*_DataNode/install/hadoop/share/hadoop/common/lib）下自带了 postgre 驱动包 gsjdbc4-*.jar，与开源 postgre 服务不兼容导致报错。
    - 报错中 type 为 12 时：数据库的 pg_hba.conf 文件配置有误。
  - 解决方案：
    - 报错中 type 为 5 时：在每台 MRS core 节点上移动驱动包 gsjdbc4-*.jar 到 tmp 目录下。

      mv /opt/Bigdata/FusionInsight_HD_*/1_*_DataNode/install/hadoop/share/hadoop/common/lib/gsjdbc4-*.jar /tmp
    - 报错中 type 为 12 时：调整数据库的 pg_hba.conf 文件，将 address 改成 sqoop 所在节点的 ip。

- 场景二：（**export** 场景）使用 **sqoop export** 命令抽取开源 postgre 到 MRS hdfs 或 hive 等。

- 问题现象：

  使用 sqoop 命令查询 postgre 表可以，但是执行 sqoop export 命令倒数时报错：The authentication type 5 is not supported. Check that you have configured the pg_hba.conf file to include the client's IP address or subnet, and that it

- 问题根因：

  在执行 sqoop export 命令时，会启动 MapReduce 任务，由于 MRS Hadoop 安装目录（/opt/Bigdata/FusionInsight_HD_*/1_*_DataNode/install/hadoop/share/hadoop/common/lib）下自带了 postgre 驱动包 gsjdbc4-*.jar，与开源 postgre 服务不兼容导致报错。

- 解决方案：

  1.在每台 MRS core 节点上移动驱动包 gsjdbc4-*.jar 到 tmp 目录下。

  mv /opt/Bigdata/FusionInsight_HD_*/1_*_DataNode/install/hadoop/share/hadoop/common/lib/gsjdbc4-*.jar /tmp

  2. 将/opt/Bigdata/client/Hive/Beeline/lib/gsjdbc4-*.jars 删除 。

26.4.4 使用 hive-table 方式同步数据到 obs 上的 hive 表报错

问题

使用 hive-table 方式同步数据到 obs 上的 hive 表报错。

```
2021-09-03 16:28:11,611 ERROR tools.DistCp: XAttrs not supported on at least one file system:
org.apache.hadoop.tools.CopyListing$XAttrsNotSupportedException: XAttrs not supported for file system:
obs://fdd-fs
        at org.apache.hadoop.tools.util.DistCpUtils.checkFileSystemXAttrSupport(DistCpUtils.java:555)
        at org.apache.hadoop.tools.DistCp.configureOutputFormat(DistCp.java:341)
        at org.apache.hadoop.tools.DistCp.createJob(DistCp.java:308)
        at org.apache.hadoop.tools.DistCp.createAndSubmitJob(DistCp.java:218)
        at org.apache.hadoop.tools.DistCp.execute(DistCp.java:197)
        at org.apache.hadoop.tools.DistCp.run(DistCp.java:155)
```

回答

修改数据同步方式，将**-hive-table** 改成**-hcatalog-table**。

26.4.5 使用 hive-table 方式同步数据到 orc 表或者 parquet 表失败

问题

使用 hive-table 方式同步数据到 orc 表或者 parquet 表失败，报错中会有 kite-sdk 的包名。

回答

修改数据同步方式，将**-hive-table** 改成**-hcatalog-table**。

26.4.6 使用 hive-table 方式同步数据报错

问题

使用 hive-table 方式同步数据报错。

```
        at org.apache.hadoop.hive.ql.metadata.Hive.registerAllFunctionsOnce(Hive.java:230) [hive-exec-3.1.0-hw-ei-312001-SNAPSHOT.jar:3.1.0-hw-ei-312001-SNAPSHOT]
        ... 41 more
14:41:42.891 [bfe1438c-07bb-43fd-91d9-910a348f6e91 main] ERROR org.apache.hadoop.hive.metastore.ObjectStore - Version information not found in metastore. The process will exit.
14:41:42.892 [bfe1438c-07bb-43fd-91d9-910a348f6e91 main] ERROR org.apache.hadoop.hive.metastore.RetryingHMSHandler - ExitSecurityException
        at org.apache.sqoop.util.SubprocessSecurityManager.checkExit(SubprocessSecurityManager.java:83)
        at java.lang.Runtime.exit(Runtime.java:107)
        at java.lang.System.exit(System.java:973)
        at org.apache.hadoop.hive.metastore.ObjectStore.checkSchema(ObjectStore.java:9555)
        at org.apache.hadoop.hive.metastore.ObjectStore.verifySchema(ObjectStore.java:9531)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:498)
        at org.apache.hadoop.hive.metastore.RawStoreProxy.invoke(RawStoreProxy.java:97)
        at com.sun.proxy.$Proxy37.verifySchema(Unknown Source)
        at org.apache.hadoop.hive.metastore.HiveMetaStore$HMSHandler.getMSForConf(HiveMetaStore.java:903)
        at org.apache.hadoop.hive.metastore.HiveMetaStore$HMSHandler.getMS(HiveMetaStore.java:895)
        at org.apache.hadoop.hive.metastore.HiveMetaStore$HMSHandler.createDefaultDB(HiveMetaStore.java:978)
        at org.apache.hadoop.hive.metastore.HiveMetaStore$HMSHandler.init(HiveMetaStore.java:585)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:498)
        at org.apache.hadoop.hive.metastore.RetryingHMSHandler.invokeInternal(RetryingHMSHandler.java:148)
        at org.apache.hadoop.hive.metastore.RetryingHMSHandler.invoke(RetryingHMSHandler.java:109)
        at org.apache.hadoop.hive.metastore.RetryingHMSHandler.<init>(RetryingHMSHandler.java:81)
        at org.apache.hadoop.hive.metastore.RetryingHMSHandler.getProxy(RetryingHMSHandler.java:94)
        at org.apache.hadoop.hive.metastore.HiveMetaStore.newRetryingHMSHandler(HiveMetaStore.java:9683)
        at org.apache.hadoop.hive.metastore.HiveMetaStoreClient.<init>(HiveMetaStoreClient.java:185)
        at org.apache.hadoop.hive.ql.metadata.SessionHiveMetaStoreClient.<init>(SessionHiveMetaStoreClient.java:96)
        at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
        at sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:62)
        at sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.java:45)
        at java.lang.reflect.Constructor.newInstance(Constructor.java:423)
        at org.apache.hadoop.hive.metastore.utils.JavaUtils.newInstance(JavaUtils.java:84)
        at org.apache.hadoop.hive.metastore.RetryingMetaStoreClient.<init>(RetryingMetaStoreClient.java:97)
        at org.apache.hadoop.hive.metastore.RetryingMetaStoreClient.getProxy(RetryingMetaStoreClient.java:155)
```

回答

修改 hive-site.xml，加入如下值。

```
<property>
<name>hive.metastore.schema.verification</name>
<value>false</value>
</property>
```

26.4.7 使用 hcatalog 方式同步 hive parquet 表报错

问题

同步 hive parquet 表，其分区字段为非 string 类型，无法正常使用 hive import 导入，只能考虑使用 hcatalog 方式，但是 hcatalog 方式报错如下：

```
2021-09-28 12:12:17,623 INFO common.HCatUtil: mapreduce.lib.hcatoutput.hive.conf is set. Applying configuration differences.
2021-09-28 12:12:17,629 INFO common.HiveClientCache: Initializing cache: eviction-timeout=120 initial-capacity=50 maximum-capacity=50
2021-09-28 12:12:17,648 INFO metastore.HiveMetaStoreClient: Trying to connect to metastore with URI thrift://node-master4yPDW.a0dbfe45-7b6c-4386-83
68f7765cdd.com:9083
2021-09-28 12:12:17,649 INFO metastore.HiveMetaStoreClient: Opened a connection to metastore, current connections: 2
2021-09-28 12:12:17,651 INFO metastore.HiveMetaStoreClient: Connected to metastore.
2021-09-28 12:12:17,651 INFO metastore.RetryingMetaStoreClient: RetryingMetaStoreClient proxy=class org.apache.hive.hcatalog.common.HiveClientCache
eableHiveMetaStoreClient ugi=poseidon (auth:SIMPLE) retries=1 delay=1 lifetime=0
2021-09-28 12:12:17,875 WARN conf.HiveConf: HiveConf of name hive.http.filter.initializers does not exist
2021-09-28 12:12:17,876 WARN conf.HiveConf: HiveConf of name hive.server2.authentication.ldap.url.port does not exist
2021-09-28 12:12:17,877 INFO conf.HiveConf: current conf hive.parquet.time.zone.isLocal=true
2021-09-28 12:12:18,056 INFO hcat.SqoopHCatUtilities: Setting hCatInputFormat filter to day='20210928'
2021-09-28 12:12:18,072 WARN conf.HiveConf: HiveConf of name hive.http.filter.initializers does not exist
2021-09-28 12:12:18,072 WARN conf.HiveConf: HiveConf of name hive.server2.authentication.ldap.url.port does not exist
2021-09-28 12:12:18,073 INFO conf.HiveConf: current conf hive.parquet.time.zone.isLocal=true
2021-09-28 12:12:18,073 INFO common.HCatUtil: mapreduce.lib.hcatoutput.hive.conf is set. Applying configuration differences.
2021-09-28 12:12:18,108 ERROR tool.ImportTool: Import failed: java.io.IOException: MetaException(message:Filtering is supported only on partition k
f type string)
        at org.apache.hive.hcatalog.mapreduce.HCatInputFormat.setFilter(HCatInputFormat.java:120)
        at org.apache.sqoop.mapreduce.hcat.SqoopHCatUtilities.configureHCat(SqoopHCatUtilities.java:391)
        at org.apache.sqoop.mapreduce.hcat.SqoopHCatUtilities.configureImportOutputFormat(SqoopHCatUtilities.java:850)
        at org.apache.sqoop.mapreduce.ImportJobBase.configureOutputFormat(ImportJobBase.java:102)
        at org.apache.sqoop.mapreduce.ImportJobBase.runImport(ImportJobBase.java:263)
        at org.apache.sqoop.manager.SqlManager.importQuery(SqlManager.java:748)
        at org.apache.sqoop.tool.ImportTool.importTable(ImportTool.java:522)
        at org.apache.sqoop.tool.ImportTool.run(ImportTool.java:628)
        at org.apache.sqoop.Sqoop.run(Sqoop.java:147)
        at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:76)
        at org.apache.sqoop.Sqoop.runSqoop(Sqoop.java:183)
        at org.apache.sqoop.Sqoop.runTool(Sqoop.java:234)
```

回答

1. 修改 sqoop 源码 SqoopHCatUtilities 中的代码，将限制代码去掉。
2. 修改 hive 客户端中的 hive-site.xml 文件，修改 hive.metastore.integral.jdo.pushdown 参数为 true。

26.4.8 使用 Hcatalog 方式同步 Hive 和 MySQL 之间的数据，timestamp 和 data 类型字段会报错

问题

使用 Hcatalog 方式同步 Hive 和 MySQL 之间的数据，timestamp 和 data 类型字段会报错：

回答

- 调整 Sqoop 源码包中的代码，将 timestamp 强制转换类型和 Hive 保持一致。
- 将 Hive 中的字段类型修改为 String。

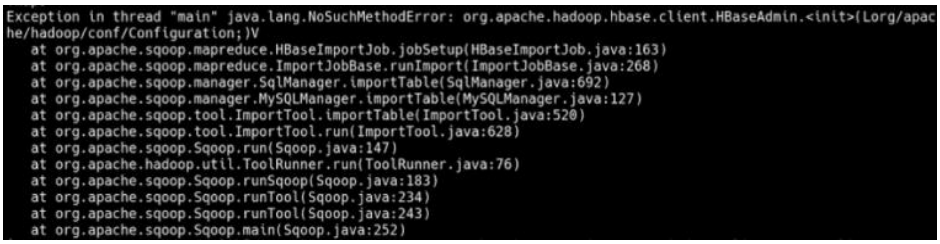
26.4.9 读取 MySQL 中数据到 HBase 报 HBaseAdmin.<init>方法找不到异常

问题

使用 MRS 的 Sqoop 客户端（1.4.7 版本），从 MySQL 数据库中指定表抽取数据，存放到 HBase（2.2.3 版本）指定的表中，报出异常：

```
Trying to load data into HBASE through Sqoop getting below error.
Exception in thread "main" java.lang.NoSuchMethodError:
org.apache.hadoop.hbase.client.HBaseAdmin.<init>(Lorg/apache/hadoop/conf/Configurat
ion;)V
```

完整异常信息如图所示：



执行 Sqoop 抽取数据命令样例：

```
sqoop import \
--connect jdbc:mysql://mysql 服务器地址:端口号/database1 \
--username admin \
--password xxx \
--table table1 \
--hbase-table table2 \
--column-family info \
--hbase-row-key id \
--hbase-create-table --m 1
```

回答

Sqoop 客户端安装完成之后，没有直接引入 HBase 相关的依赖 jar 包，需要通过手动导入指定低版本的 HBase 相关依赖 jar 包。解决方法步骤如下：

步骤 1 确认 Sqoop 客户端和 HBase 客户端是否在同一个路径下。

- 是，执行步骤 2。
- 否，删除原有的 Sqoop 和 HBase 客户端文件，从 FusionInsight Manager 上下载完整的客户端安装在同一路径下。执行步骤 2。

步骤 2 以 **root** 用户登录 Sqoop 客户端安装节点。

步骤 3 下载以下 HBase 1.6.0 版本的 jar 包上传到 Sqoop 客户端的“lib”目录下。

- hbase-client-1.6.0.jar
- hbase-common-1.6.0.jar
- hbase-protocol-1.6.0.jar
- hbase-server-1.6.0.jar

步骤 4 上传包之后，修改包的权限，可以设置为 755，具体执行命令为：

chmod 755 *包名称*

步骤 5 执行以下命令刷新 Sqoop 客户端：

source bigdata_env

----结束

27 使用 Storm

27.1 从零开始使用 Storm

用户可以在 MRS 集群的客户端中提交和删除 Storm 拓扑等基本功能。

前提条件

已安装 MRS 集群客户端，例如安装目录为“/opt/hadoopclient”。以下操作的客户端目录只是举例，请根据实际安装目录修改。

操作步骤

步骤 1 根据业务情况，准备好客户端，登录安装客户端的节点。

请根据客户端所在位置，参考“用户指南 > 使用 MRS 客户端 > 安装客户端”章节，登录安装客户端的节点。

步骤 2 执行以下命令，切换到客户端目录，例如“/opt/hadoopclient”。

cd /opt/hadoopclient

步骤 3 执行以下命令，配置环境变量。

source bigdata_env

步骤 4 启用 Kerberos 认证的集群，执行以下命令认证用户身份。未启用 Kerberos 认证的集群无需执行。

kinit *Storm 用户*

步骤 5 执行以下命令，提交 Storm 拓扑：

storm jar *拓扑包路径 拓扑 Main 方法的类名称 拓扑名称*

界面提示以下信息表示提交成功：

```
Finished submitting topology: topo1
```

步骤 6 执行以下命令，查看 Storm 中的拓扑。启用 Kerberos 认证的集群，只有属于“stormadmin”或“storm”的用户可以查看所有拓扑。

storm list

步骤 7 执行以下命令，删除 Storm 中的拓扑。

storm kill *拓扑名称*

----结束

27.2 使用 Storm 客户端

操作场景

该任务指导用户在运维场景或业务场景中使用 Storm 客户端。

前提条件

- 已安装客户端。例如安装目录为“/opt/hadoopclient”。
- 各组件业务用户由系统管理员根据业务需要创建。安全模式下，“机机”用户需要下载 keytab 文件。“人机”用户第一次登录时需修改密码。（普通模式不涉及）

操作步骤

步骤 1 根据业务情况，准备好客户端，登录安装客户端的节点。

请根据客户端所在位置，参考“用户指南 > 使用 MRS 客户端 > 安装客户端”章节，登录安装客户端的节点。

步骤 2 执行以下命令，切换到客户端安装目录。

cd /opt/hadoopclient

步骤 3 执行以下命令配置环境变量。

source bigdata_env

步骤 4 若安装了 Storm 多实例，在使用 Storm 命令提交拓扑时，请执行以下命令加载具体实例的环境变量，否则请跳过此步骤。例如，Storm-2 实例：

source Storm-2/component_env

步骤 5 执行以下命令，进行用户认证。（普通模式跳过此步骤）

kinit *组件业务用户*

步骤 6 执行命令进行客户端操作。

例如执行以下命令：

- **cql**
- **storm**

说明

同一个 storm 客户端不能同时连接安全和非安全的 ZooKeeper。

----结束

27.3 使用客户端提交 Storm 拓扑

操作场景

用户可以根据业务需要，在集群的客户端中提交 Storm 拓扑，持续处理用户的流数据。启用 Kerberos 认证的集群，需要提交拓扑的用户属于“stormadmin”或“storm”组。

前提条件

已刷新客户端。

操作步骤

步骤 1 根据业务情况，准备好客户端，登录安装客户端的节点。

请根据客户端所在位置，参考“用户指南 > 使用 MRS 客户端 > 安装客户端”章节，登录安装客户端的节点。

步骤 2 执行以下命令，设置拓扑的 jar 包权限。

例如修改“/opt/storm/topology.jar”的权限：

chmod 600 /opt/storm/topology.jar

步骤 3 执行以下命令，切换到客户端目录，例如“/opt/client”。

cd /opt/client

步骤 4 执行以下命令，配置环境变量。

source bigdata_env

步骤 5 若安装了 Storm 多实例，在使用 Storm 命令提交拓扑时，请执行以下命令加载具体实例的环境变量，否则请跳过此步骤。例如，Storm-2 实例：

source Storm-2/component_env

步骤 6 启用 Kerberos 认证的集群，执行以下命令认证用户身份。未启用 Kerberos 认证的集群无需执行。

kinit *Storm 用户*

步骤 7 MRS 3.x 之前版本：执行以下命令，提交 Storm 拓扑。

storm jar *拓扑包路径 拓扑 Main 方法的类名称 拓扑名称*

界面提示以下信息表示提交成功：

```
Finished submitting topology: topo1
```

说明

- 如果需要拓扑支持采样消息，则还需要增加参数“topology.debug”和“topology.eventlogger.executors”。
- 拓扑如何处理数据是拓扑自身行为。样例拓扑随机生成字符并分隔字符串，需要查看处理情况时，请启用采样功能并参见查看 Storm 拓扑日志。

步骤 8 MRS 3.x 及后续版本：执行以下命令，提交拓扑任务。

storm jar *topology-jar-path class 入参列表*

- topology-jar-path：表示拓扑的 jar 包所在路径。
- class：表示拓扑使用的 main 方法所在类名称。
- 入参列表：表示拓扑使用的 main 方法入参。

显示以下信息表示拓扑提交成功：

```
Finished submitting topology: topology1
```

说明

- 登录认证用户必须与所加载环境变量（component_env）一 一对应，否则使用 storm 命令提交拓扑任务出错。
- 加载客户端环境变量且对应用户登录成功后，该用户可以在任意 storm 客户端下执行 storm 命令来提交拓扑任务，但提交拓扑命令执行完成后，提交成功的拓扑仍然在用户所对应的 Storm 集群中，不会出现在其他 Storm 集群中。
- 如果修改了集群域名，需要在提交拓扑前重新设置域名信息，进入 cql 语句执行命令。

步骤 9 执行以下命令，查看 Storm 中的拓扑。启用 Kerberos 认证的集群，只有属于“stormadmin”或“storm”的用户可以查看所有拓扑。

storm list

----结束

27.4 访问 Storm 的 WebUI

操作场景

用户可以通过 Storm 的 WebUI，在图形化界面使用 Storm。

Storm 的 WebUI 支持查看以下信息：

- Storm 集群汇总信息
- Nimbus 汇总信息
- 拓扑汇总信息
- Supervisor 汇总信息
- Nimbus 配置信息

前提条件

- 获取用户“admin”帐号密码。“admin”密码在创建集群时由用户指定。
- 使用其他用户访问 Storm WebUI，需要用户属于“storm”或“stormadmin”用户组。

操作步骤

步骤 1 进入组件管理页面：

- MRS 3.x 之前版本，单击集群名称，登录集群详情页面，选择“组件管理”。

  说明

  若集群详情页面没有“组件管理”页签，请先完成 IAM 用户同步（在集群详情页的“概览”页签，单击“IAM 用户同步”右侧的“同步”进行 IAM 用户同步）。

- MRS 3.x 及后续版本，登录 FusionInsight Manager，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本）。然后选择“集群 > *待操作的集群名称* > 服务 ”。

步骤 2 登录 Storm WebUI：

- MRS 3.x 之前版本：选择“Storm”，在“Storm 概述”的“Storm Web UI”，单击任意一个 UI 链接，打开 Storm 的 WebUI。

  说明

  第一次访问 Storm WebUI，需要在浏览器中添加站点信任以继续打开页面。

- MRS 3.x 及后续版本：选择“Storm > 概览”，在“基本信息”的“Storm Web UI”，单击任意一个 UI 链接，打开 Storm 的 WebUI。

----结束

相关任务

- 单击拓扑名称，可查看指定拓扑的详细信息、拓扑状态、Spouts 信息、Bolts 信息和拓扑配置。
- 在“Topology actions”区域，用户可以对拓扑执行激活、去激活、重部署、删除操作、调试、停止调试和修改日志级别，即“Activate”、“Deactivate”、“Rebalance”、“Kill”、“Debug”、“Stop Debug”、“Change Log Level”。重部署和删除操作需要设置操作执行的等待时间，单位为秒。
- 在“Topology Visualization”区域，用户可以执行拓扑可视化操作，即单击“Show Visualization”。拓扑可视化后，WebUI 将显示拓扑结构图。

27.5 管理 Storm 拓扑

操作场景

用户可以使用 Storm 的 WebUI 管理拓扑。“storm”用户组的用户只能管理由自己提交的拓扑任务，“stormadmin”用户组的用户可以管理所有拓扑任务。

操作步骤

步骤 1 访问 Storm 的 WebUI，请参考访问 Storm 的 WebUI。

步骤 2 在“Topology summary”区域，单击指定的拓扑名称。

步骤 3 通过“Topology actions”管理 Storm 拓扑。

- 激活拓扑

  单击“Activate”，转化当前拓扑为激活状态。

- 去激活拓扑

  单击“Deactivate”，转化当前拓扑为去激活状态。

- 重部署拓扑

  单击“Rebalance”，将当前拓扑重新部署执行，需要输入执行重部署的等待时间，单位为秒。一般在集群中节点数发生变化时进行，以更好利用集群资源。

- 删除拓扑

  单击“Kill”，将当前拓扑删除，需要输入执行操作的等待时间，单位为秒。

- 采样、停止采样拓扑消息

  单击“Debug”，在弹出窗口输入流数据采样消息的数值，单位为百分比，表示从开始采样到停止采样这段时间内所有数据的采集比例。例如输入“10”，则采集比例为 10%。

  如果需要停止采样，则单击“Stop Debug”。

  说明

  只有在提交拓扑时启用采样功能，才支持此功能。查看采样处理数据，请参见查看 Storm 拓扑日志。

- 修改拓扑日志级别

  单击“Change Log Level”，可以为 Storm 日志指定新的日志信息级别。

步骤 4 显示拓扑结构图。

在“Topology Visualization”区域单击“Show Visualization”，执行拓扑可视化操作。

----结束

27.6 查看 Storm 拓扑日志

操作场景

用户需要查看 Storm 拓扑在 worker 进程中的执行情况时，需要查看 worker 中关于拓扑的日志。如果需要查询拓扑在运行时数据处理的日志，提交拓扑并启用“Debug”功能后可以查看日志。仅启用 Kerberos 认证的流集群支持该场景，且用户需要是拓扑的提交者，或者加入“stormadmin”。

前提条件

- 在工作环境完成网络配置。
- 需要查看处理数据的拓扑，提交时已启用采样功能。

查看 worker 进程日志

步骤 1 访问 Storm 的 WebUI，请参考访问 Storm 的 WebUI。

步骤 2 在“Topology Summary”区域单击指定的拓扑名称，打开拓扑的详细信息。

步骤 3 单击要查看日志的“Spouts”或“Bolts”任务，在“Executors (All time)”区域单击“Port”列的端口值，查看详细日志内容。

----结束

查看拓扑处理数据日志

步骤 1 访问 Storm 的 WebUI，请参考访问 Storm 的 WebUI。

步骤 2 在“Topology Summary”区域单击指定的拓扑名称，打开拓扑的详细信息。

步骤 3 单击“Debug”，输入采样数据的百分比数值，并单击“OK”开始采样。

步骤 4 单击拓扑的“Spouts”或“Bolts”任务，在“Component summary”单击“events”打开处理数据日志。

----结束

27.7 Storm 常用参数

本章节内容适用于 MRS 3.x 及后续版本。

参数入口

参数入口，请参考修改集群服务配置参数。

参数说明

表27-1 参数说明

配置参数	说明	默认值
supervisor.slots.ports	supervisor 上能够运行 workers 的端口列表。每个 worker 占用一个端口，且每个端口只运行一个 worker。通过这项配置可以设置每台机器上运行的 worker 数量。端口的取值范围是 1024 到 65535，不同端口使用逗号分隔。	6700,6701,6702,6703
WORKER_GC_OPTS	supervisor 启动 worker 时使用的 jvm 选项。需要根据业务中对内存等的使用来进行设置，例如是简单业务处理，建议 1G，既“-Xmx1G”；如果有窗口缓存，根据窗口大小计算：每条记录大小*周期*2。	-Xms1G -Xmx1G -XX:+UseG1GC -XX:+PrintGCDetails -Xloggc:artifacts/gc.log -XX:+PrintGCDateStamps -XX:+PrintGCTimeStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=1M -XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=artifacts/heapdump
default.schedule.mode	默认调度器的调度模式。目前支持两个值，具体值与含义如下： • “AVERAGE”：使用按空闲 Slot 数目为优先级的调度机制 • “RATE”：使用按空闲 Slot 比率为优先级的调度机制	AVERAGE
nimbus.thrift.threads	设置主用 Nimbus 对外提供服务时的最大连接线程数。当 Storm 集群规模较大，Supervisor 实例数量较多时，需要增加线程数。	512

27.8 配置 Storm 业务用户密码策略

操作场景

本章节内容适用于 MRS 3.x 及后续版本。

使用 Storm 业务用户提交一个拓扑以后，该任务需要使用提交拓扑的用户身份持续运行。在拓扑运行的过程中，worker 进程可能需要正常重启以保持拓扑工作。若业务用户的密码被修改，或密码使用天数超过了默认密码策略指定的最大有效期，则会影响拓扑正常运行。系统管理员需要根据企业安全要求，为 Storm 业务用户配置独立的密码策略。

说明

如果不为 Storm 业务用户配置独立的密码策略，在修改业务用户密码以后，可以删除旧的拓扑并重新提交，使拓扑继续运行。

对系统的影响

- 为 Storm 业务用户配置独立的密码策略后，此用户将不受 Manager 界面上的“密码策略”配置影响。
- 为 Storm 业务用户配置独立的密码策略后，如果配置了跨集群互信，请根据此密码策略，在 Manager 为 Storm 业务用户重置密码。

前提条件

系统管理员已明确业务需求，并创建好“人机”用户，例如“testpol”。

操作步骤

步骤 1 以“omm”用户登录集群内任意节点。

步骤 2 执行以下命令，防止超时退出。

TMOUT=0

说明

执行完本章节操作后，请及时恢复超时退出时间，执行命令 **TMOUT=*超时退出时间*。**例如：**TMOUT=600**，表示用户无操作 600 秒后超时退出。

步骤 3 执行以下命令，导出环境变量。

EXECUTABLE_HOME="${CONTROLLER_HOME}/kerberos_user_specific_binay/kerberos"

LD_LIBRARY_PATH=${EXECUTABLE_HOME}/lib:$LD_LIBRARY_PATH

PATH=${EXECUTABLE_HOME}/bin:$PATH

步骤 4 执行以下命令，并输入 Kerberos 管理员密码，进入 Kerberos 管理控制台。

kadmin -p kadmin/admin

说明

第一次使用“kadmin/admin”用户需要修改“kadmin/admin”密码。

界面显示如下信息，则表示已成功进入 Kerberos 管理控制台。

```
kadmin:
```

步骤 5 执行以下命令，查看创建好的“Human-Machine”用户的具体信息。

getprinc *用户名*

例如，查看“testpol”用户的详细信息：

getprinc testpol

界面显示如下信息，说明指定用户使用了默认的密码策略：

```
Principal: testpol@<系统域名>
……
Policy: default
```

步骤 6 执行以下命令，为 Storm 业务用户创建独立的密码策略，例如“streampol”：

addpol -maxlife 0day -minlife 0sec -history 1 -maxfailure 5 -failurecountinterval 5min -lockoutduration 5min -minlength 8 -minclasses 4 streampol

其中“-maxlife”表示密码最大有效期，“0day”表示永不过期。

步骤 7 执行以下命令，查看新创建的策略“streampol”。

getpol streampol

界面显示如下信息，说明新策略已指定密码不过期：

```
Policy: streampol
 Maximum password life: 0 days 00:00:00
 ……
```

步骤 8 执行以下命令，将新的策略“streampol”应用到 Storm 用户“testpol”。

modprinc -policy streampol testpol

其中“streampol”是策略名称，“testpol”是用户名。

界面显示如下信息，说明指定用户的属性已修改：

```
Principal "testpol@<系统域名>" modified.
```

步骤 9 执行以下命令，查看 Storm 用户“testpol”用户的当前信息。

getprinc testpol

界面显示如下信息，说明指定用户使用了新的密码策略：

```
Principal: testpol@<系统域名>
 ……
 Policy: streampol
```

----结束

27.9 迁移 Storm 业务至 Flink

27.9.1 概述

本章节内容适用于 MRS 3.x 及后续版本。

Flink 从 0.10.0 版本开始提供了一套 API 可以将使用 Storm API 编写的业务平滑迁移到 Flink 平台上，只需要极少的改动即可完成。通过这项转换可以覆盖大部分的业务场景。

Flink 支持两种方式的业务迁移：

1. 完整迁移 Storm 业务：转换并运行完整的由 Storm API 开发的 Storm 拓扑。
2. 嵌入式迁移 Storm 业务：在 Flink 的 DataStream 中嵌入 Storm 的代码，如使用 Storm API 编写的 Spout/Bolt。

Flink 提供了 flink-storm 包用来完成上述转换。

27.9.2 完整迁移 Storm 业务

操作场景

该任务指导用户通过 Storm 业务完整迁移的方式转换并运行完整的由 Storm API 开发的 Storm 拓扑。

操作步骤

步骤 1 打开 Storm 业务工程，修改工程的 pom 文件，增加“flink-storm” 、“flink-core”和“flink-streaming-java_2.11”的引用。如下：

```
<dependency>
   <groupId>org.apache.flink</groupId>
   <artifactId>flink-storm_2.11</artifactId>
   <version>1.4.0</version>
   <exclusions>
      <exclusion>
         <groupId>*</groupId>
         <artifactId>*</artifactId>
      </exclusion>
   </exclusions>
</dependency>
<dependency>
   <groupId>org.apache.flink</groupId>
   <artifactId>flink-core</artifactId>
   <version>1.4.0</version>
   <exclusions>
      <exclusion>
         <groupId>*</groupId>
         <artifactId>*</artifactId>
      </exclusion>
   </exclusions>
</dependency>
<dependency>
   <groupId>org.apache.flink</groupId>
   <artifactId>flink-streaming-java_2.11</artifactId>
   <version>1.4.0</version>
   <exclusions>
      <exclusion>
         <groupId>*</groupId>
```

```
            <artifactId>*</artifactId>
        </exclusion>
    </exclusions>
</dependency>
```

说明

如果是非 maven 工程，则手动收集如上 jar 包，添加到工程的 classpath 中。

步骤 2 修改拓扑提交部分代码，下面以 WordCount 为例：

1. Storm 拓扑的构造部分保持不变，无需修改，包括使用 Storm API 开发的 Spout 和 Bolt 都无需修改。

```
TopologyBuilder builder = new TopologyBuilder();
builder.setSpout("spout", new RandomSentenceSpout(), 5);
builder.setBolt("split", new SplitSentenceBolt(), 8).shuffleGrouping("spout");
builder.setBolt("count", new WordCountBolt(), 12).fieldsGrouping("split", new
Fields("word"));
```

2. 拓扑的提交部分需要修改，Storm 的提交示例如下：

```
Config conf = new Config();
conf.setNumWorkers(3);

StormSubmitter.submitTopology("word-count", conf, builder.createTopology());
```

需要进行如下修改：

```
Config conf = new Config();
 conf.setNumWorkers(3);

 //将 Storm 的 Config 转化为 Flink 的 StormConfig
 StormConfig stormConfig = new StormConfig(conf);

 //使用 Storm 的 TopologBuilder 构造 FlinkTopology
 FlinkTopology topology = FlinkTopology.createTopology(builder);

 //获取 StreamExecutionEnvironment
 StreamExecutionEnvironment env = topology.getExecutionEnvironment();

 //将 StormConfig 设置到 Job 的环境变量中，用于构造 Bolt 和 Spout
 //如果 Bolt 和 Spout 初始化时不需要 config，则不用设置
 env.getConfig().setGlobalJobParameters(stormConfig);
 //执行拓扑提交
 topology.execute();
```

3. 重新打包之后使用 flink 命令行进行提交：

   flink run -class {MainClass} WordCount.jar

----结束

27.9.3 嵌入式迁移 Storm 业务

操作场景

该任务指导用户通过嵌入式迁移的方式在 Flink 的 DataStream 中嵌入 Storm 的代码，如使用 Storm API 编写的 Spout/Bolt。

操作步骤

步骤 1 在 Flink 中，对 Storm 拓扑中的 Spout 和 Bolt 进行嵌入式转换，将之转换为 Flink 的 Operator，代码示例如下：

```
//set up the execution environment
final StreamExecutionEnvironment env =
StreamExecutionEnvironment.getExecutionEnvironment();

//get input data
final DataStream<String> text = getTextDataStream(env);

final DataStream<Tuple2<String, Integer>> counts = text

  //split up the lines in pairs (2-tuples) containing: (word,1)
  //this is done by a bolt that is wrapped accordingly
  .transform("CountBolt",
    TypeExtractor.getForObject(new Tuple2<String, Integer>("", 0)),
    new BoltWrapper<String, Tuple2<String, Integer>>(new CountBolt()))
  //group by the tuple field "0" and sum up tuple field "1"
  .keyBy(0).sum(1);
// execute program
env.execute("Streaming WordCount with bolt tokenizer");
```

步骤 2 修改完成后使用 Flink 命令进行提交。

flink run -class {MainClass} WordCount.jar

----结束

27.9.4 迁移 Storm 对接的外部安全组件业务

迁移 Storm 对接 HDFS 和 HBase 组件的业务

如果 Storm 的业务使用的 storm-hdfs 或者 storm-hbase 插件包进行的对接，那么在按照完整迁移 Storm 业务进行迁移时，需要指定特定安全参数，如下：

```
//初始化 Storm 的 Config
Config conf = new Config();

//初始化安全插件列表
List<String> auto_tgts = new ArrayList<String>();
//添加 AutoTGT 插件
auto_tgts.add("org.apache.storm.security.auth.kerberos.AutoTGT");
//添加 AutoHDFS 插件
//如果对接 HBase，则如下更改为：
```

```
auto_tgts.add("org.apache.storm.hbase.security.AutoHBase");
auto_tgts.add("org.apache.storm.hdfs.common.security.AutoHDFS");

//设置安全参数
conf.put(Config.TOPOLOGY_AUTO_CREDENTIALS, auto_tgts);
//设置 worker 个数
conf.setNumWorkers(3);

//将 Storm 的 Config 转化为 Flink 的 StormConfig
StormConfig stormConfig = new StormConfig(conf);

//使用 Storm 的 TopologBuilder 构造 FlinkTopology
FlinkTopology topology = FlinkTopology.createTopology(builder);

//获取 StreamExecutionEnvironment
StreamExecutionEnvironment env = topology.getExecutionEnvironment();

//将 StormConfig 设置到 Job 的环境变量中，用于构造 Bolt 和 Spout
//如果 Bolt 和 Spout 初始化时不需要 config，则不用设置
env.getConfig().setGlobalJobParameters(stormConfig);

//执行拓扑提交
topology.execute();
```

增加如上的安全插件配置后，可以避免 HDFSBolt 和 HBaseBolt 在初始化过程中的无谓登录，因为 Flink 已经实现准备好了安全上下文，无需再登录。

迁移 Storm 对接其他安全组件的业务

如果 Storm 的业务使用的 storm-kakfa-client 和 storm-solr 等插件包进行的对接时，需要注意，之前所配置的安全插件需要去掉，如下：

```
List<String> auto_tgts = new ArrayList<String>();
//keytab 方式
auto_tgts.add("org.apache.storm.security.auth.kerberos.AutoTGTFromKeytab");

//将客户端配置的 plugin 列表写入 config 指定项中
//安全模式必配
//普通模式不用配置，请注释掉该行
conf.put(Config.TOPOLOGY_AUTO_CREDENTIALS, auto_tgts);
```

如上所配置的 AutoTGTFromKeytab 插件在进行业务迁移时，必须删除，否则会引起相应 Bolt 或 Spout 初始化时登录异常。

27.10 Storm 日志介绍

本章节内容适用于 MRS 3.x 及后续版本。

日志描述

日志路径：Storm 相关日志的默认存储路径为“/var/log/Bigdata/storm/*角色名*”（运行日志），“/var/log/Bigdata/audit/storm/*角色名*”（审计日志）。

- Nimbus：“/var/log/Bigdata/storm/nimbus”（运行日志），“/var/log/Bigdata/audit/storm/nimbus”（审计日志）
- Supervisor：“/var/log/Bigdata/storm/supervisor”（运行日志），“/var/log/Bigdata/audit/storm/supervisor”（审计日志）
- UI：“/var/log/Bigdata/storm/ui”（运行日志），“/var/log/Bigdata/audit/storm/ui”（审计日志）
- Logviewer：“/var/log/Bigdata/storm/logviewer”（运行日志），“/var/log/Bigdata/audit/storm/logviewer”（审计日志）

日志归档规则：Storm 的日志启动了自动压缩归档功能，缺省情况下，当日志大小超过 10MB 的时候会自动压缩，压缩后的日志文件名规则为：“<原有日志名>.log.[编号].gz”。默认最多保留最近的 20 个压缩文件，压缩文件保留个数和压缩文件阈值可以配置。

审计日志压缩后的日志文件名规则为：“audit.log.[yyyy-MM-dd].[编号].zip”。该文件永远都不会删除。

表27-2 Storm 日志列表

日志类型	日志文件名	描述
运行日志	nimbus/access.log	Nimbus 用户访问日志。
	nimbus/nimbus-<PID>-gc.log	Nimbus 进程的 GC 日志。
	nimbus/checkavailable.log	Nimbus 可用性检查日志。
	nimbus/checkService.log	Nimbus 可服务性检查日志。
	nimbus/metrics.log	Nimbus 监控统计的日志。
	nimbus/nimbus.log	Nimbus 进程运行日志。
	nimbus/postinstall.log	Nimbus 安装后的工作日志。
	nimbus/prestart.log	Nimbus 启动前的工作日志。
	nimbus/start.log	Nimbus 启动的工作日志。
	nimbus/stop.log	Nimbus 停止的工作日志。
	supervisor/access.log	Supervisor 用户访问日志。
	supervisor/metrics.log	Supervisor 监控统计的日志。
	supervisor/postinstall.log	Supervisor 安装后的工作日志。
	supervisor/prestart.log	Supervisor 启动前的工作日

日志类型	日志文件名	描述
		志。
	supervisor/start.log	Supervisor 启动的工作日志。
	supervisor/stop.log	Supervisor 停止的工作日志。
	supervisor/supervisor.log	Supervisor 进程运行日志。
	supervisor/supervisor-<PID>-gc.log	Supervisor 进程的 GC 日志。
	ui/access.log	UI 用户访问日志。
	ui/metric.log	UI 监控统计的日志。
	ui/ui-<PID>-gc.log	UI 进程的 GC 日志。
	ui/postinstall.log	UI 安装后的工作日志。
	ui/prestart.log	UI 启动前的工作日志。
	ui/start.log	UI 启动的工作日志。
	ui/stop.log	UI 停止的工作日志。
	ui/ui.log	UI 进程运行日志。
	logviewer/access.log	Logviewer 用户访问日志。
	logviewer/metric.log	Logviewer 监控统计的日志。
	logviewer/logviewer-<PID>-gc.log	Logviewer 进程的 GC 日志。
	logviewer/logviewer.log	logviewer 运行日志。
	logviewer/postinstall.log	logviewer 安装后的工作日志。
	logviewer/prestart.log	logviewer 启动前的工作日志。
	logviewer/start.log	logviewer 启动的工作日志。
	logviewer/stop.log	logviewer 停止的工作日志。
	supervisor/[topologyId]-worker-[端口号].log	Worker 进程运行日志，一个端口占用一个日志文件，系统默认包含

日志类型	日志文件名	描述
		29100,29101,29102,29103,29304 五个端口。
	supervisor/metadata/[topologyid]-worker-[端口号].yaml	worker 日志元数据文件，logviewer 在清理日志的时候会以该文件来作为清理依据。该文件会被 logviewer 日志清理线程根据一定条件自动删除。
	nimbus/cleanup.log	Nimbus 卸载的清理日志。
	logviewer/cleanup.log	logviewer 卸载的清理日志。
	ui/cleanup.log	UI 卸载的清理日志。
	supervisor/cleanup.log	Supervisor 卸载的清理日志。
	leader_switch.log	Storm 主备倒换运行日志。
审计日志	nimbus/audit.log	Nimbus 审计日志。
	ui/audit.log	UI 审计日志。
	supervisor/audit.log	Supervisor 审计日志。
	logviewer/audit	Logviewer 审计日志。

日志级别

Storm 提供了如表 27-3 所示的日志级别。

运行日志和审计日志的级别优先级从高到低分别是 ERROR、WARN、INFO、DEBUG，程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。

表27-3 日志级别

级别	描述
ERROR	ERROR 表示系统运行的错误信息。
WARN	WARN 表示当前事件处理存在异常信息。
INFO	INFO 表示记录系统及各事件正常运行状态信息。

级别	描述
DEBUG	DEBUG 表示记录系统及系统的调试信息。

如果您需要修改日志级别，请执行如下操作：

步骤 1 请参考修改集群服务配置参数，进入 Storm 的“全部配置”页面。

步骤 2 左边菜单栏中选择所需修改的角色所对应的日志菜单。

步骤 3 选择所需修改的日志级别。

步骤 4 保存配置，在弹出窗口中单击“确定”使配置生效。

----结束

日志格式

Storm 的日志格式如下所示：

表27-4 日志格式

日志类型	格式	示例
运行日志	%d{yyyy-MM-dd HH:mm:ss,SSS} | %-5p | [%t] | %m | %logger (%F:%L) %n	2015-03-11 23:04:00,241 | INFO | [RMI TCP Connection(2646)-10.0.0.2] | The baseSleepTimeMs [1000] the maxSleepTimeMs [1000] the maxRetries [1] | backtype.storm.utils.StormBoundedExponentialBackoffRetry (StormBoundedExponentialBackoffRetry.java:46)
	<yyyy-MM-dd HH:mm:ss,SSS><HostName><RoleName><logLevel><Message>	2017-03-28 02:57:52 493 10-5-146-1 storm- INFO Nimbus start normally
审计日志	<用户名><用户 IP><时间><操作><操作对象><操作结果>	UserName=storm/hadoop, UserIP=10.10.0.2, Time=Tue Mar 10 01:15:35 CST 2015, Operation=Kill, Resource=test, Result=Success

27.11 性能调优

27.11.1 Storm 性能调优

操作场景

通过调整 Storm 参数设置，可以提升特定业务场景下 Storm 的性能。

本章节适用于 MRS 3.x 及后续版本。

修改服务配置参数，请参考修改集群服务配置参数。

拓扑调优

当需要提升 Storm 数据量处理性能时，可以通过拓扑调优的操作提高效率。建议在可靠性要求不高的场景下进行优化。

表27-5 调优参数

配置参数	默认值	调优场景
topology.acker.executors	null	Acker 的执行器数量。当业务应用对可靠性要求较低，允许不处理部分数据，可设置参数值为“null”或“0”，以关闭 Acker 的执行器，减少流控制，不统计消息时延，提高性能。
topology.max.spout.pending	null	Spout 消息缓存数，仅在 Acker 不为 0 或者不为 null 的情况下生效。Spout 将发送到下游 Bolt 的每条消息加入到 pending 队列，待下游 Bolt 处理完成并确认后，再从 pending 队列移除，当 pending 队列占满时 Spout 暂停消息发送。增加 pending 值可提高 Spout 的每秒消息吞吐量，提高性能，但延时同步增加。
topology.transfer.buffer.size	32	每个 worker 进程 Distuptor 消息队列大小，建议在 4 到 32 之间，增大消息队列可以提升吞吐量，但延时可能会增加。
RES_CPUSET_PERCENTAGE	80	设置各个节点上的 Supervisor 角色实例（包含其启动并管理的 Worker 进程）所使用的物理 CPU 百分比。根据 Supervisor 所在节点业务量需求，适当调整参数值，优化 CPU 使用率。

JVM 调优

当应用程序需要处理大量数据从而占用更多的内存时，存在 worker 内存大于 2GB 的情况，推荐使用 G1 垃圾回收算法。

表27-6 调优参数

配置参数	缺省值	调优场景
WORKER_GC_OPTS	-Xms1G -Xmx1G -XX:+UseG1GC -XX:+PrintGCDetails -Xloggc:artifacts/gc.log -XX:+PrintGCDateStamps -XX:+PrintGCTimeStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=1M -XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=artifacts/heapdump	应用程序内存中需要保存大量数据，worker 进程使用的内存大于 2G，那么建议使用 G1 垃圾回收算法，可修改参数值为“-Xms2G -Xmx5G -XX:+UseG1GC”。

28 使用 Tez

28.1 使用前须知

本章节适用于 MRS 3.x 及后续版本。

28.2 Tez 常用参数

参数入口

在 Manager 系统中，选择“集群 > 服务 > Tez > 配置”，选择“全部配置”。在搜索框中输入参数名称。

参数说明

表28-1 参数说明

配置参数	说明	缺省值
property.tez.log.dir	Tez 日志目录。	/var/log/Bigdata/tez/tezui
property.tez.log.level	Tez 的日志级别。	INFO

28.3 访问 TezUI

Tez 提供 Tez 任务执行过程图形化展示功能，使用户可以通过界面的方式查看 Tez 任务执行细节。

前提条件

已安装 Yarn 服务的 TimelineServer 实例。

使用介绍

登录 Manager 系统，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本），在 Manager 界面选择“集群 > 服务 > Tez”，在“基本信息”中单击“Tez WebUI”右侧的链接，打开 Tez WebUI。可查看执行的 Tez 任务执行细节。

28.4 日志介绍

日志描述

日志路径：Tez 相关日志的默认存储路径为“/var/log/Bigdata/tez/*角色名*”。

TezUI：“/var/log/Bigdata/tez/tezui”（运行日志），“/var/log/Bigdata/audit/tez/tezui”（审计日志）。

日志归档规则：Tez 的日志启动了自动压缩归档功能，缺省情况下，当日志大小超过 20MB 的时候（此日志文件大小可进行配置），会自动压缩，压缩后的日志文件名规则为：“<原有日志名>-<yyyy-mm-dd_hh-mm-ss>.[编号].log.zip”。最多保留最近的 20 个压缩文件，压缩文件保留个数和压缩文件阈值可以配置。

表28-2 Tez 日志列表

日志类型	日志文件名	描述
运行日志	tezui.out	TezUI 运行环境信息日志
	tezui.log	TezUI 进程的运行日志
	tezui-omm-<日期>-gc.log.<编号>	TezUI 进程的 GC 日志
	prestartDetail.log	TezUI 启动前的工作日志
	check-serviceDetail.log	TezUI 服务启动是否成功的检查日志
	postinstallDetail.log	TezUI 安装后的工作日志
	startDetail.log	TezUI 进程启动日志
	stopDetail.log	TezUI 进程停止日志
审计日志	tezui-audit.log	TezUI 审计日志

日志级别

TezUI 提供了如表 28-3 所示的日志级别。

运行日志的级别优先级从高到低分别是 ERROR、WARN、INFO、DEBUG，程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。

表28-3 日志级别

级别	描述
ERROR	ERROR 表示系统运行的错误信息。
WARN	WARN 表示当前事件处理存在异常信息。
INFO	INFO 表示记录系统及各事件正常运行状态信息。
DEBUG	DEBUG 表示记录系统及系统的调试信息。

如果您需要修改日志级别，请执行如下操作：

步骤 1 登录 Manager。

步骤 2 选择“集群 > 服务 > Tez > 配置”。

步骤 3 选择“全部配置”。

步骤 4 左边菜单栏中选择“TezUI > 日志”。

步骤 5 选择所需修改的日志级别。

步骤 6 单击“保存”，在弹出窗口中单击“确定”保存配置。

步骤 7 单击“实例”，勾选“TezUI”角色，选择“更多 > 重启实例”，输入用户密码后，在弹出窗口单击“确定”。

步骤 8 等待实例重启完成，配置生效。

----结束

日志格式

Tez 的日志格式如下所示：

表28-4 日志格式

日志类型	格式	示例
运行日志	<yyyy-MM-dd HH:mm:ss,SSS>|<LogLevel>|<产生该日志的线程名字>|<log 中的 message>|<日志事件的发生位置>	2020-07-31 11:44:21,378 | INFO | TezUI-health-check | Start health check | com.XXX.tez.HealthCheck.run(HealthCheck.java:30)
审计日志	<yyyy-MM-dd HH:mm:ss,SSS>|<LogLevel>|<产生该日志的线程名字>|<User Name><User IP><Time><Operation><Resource><Result><Detail >|<日志事件的发生位置>	2018-12-24 12:16:25,319 | INFO | HiveServer2-Handler-Pool: Thread-185 | UserName=hive UserIP=10.153.2.204 Time=2018/12/24 12:16:25 Operation=CloseSession Result=SUCCESS Detail= |

日志类型	格式	示例
		org.apache.hive.service.cli.thrift.ThriftCLIService.logAuditEvent(ThriftCLIService.java:434)

28.5 常见问题

28.5.1 TezUI 无法展示 Tez 任务执行细节

问题

登录 Manager 界面，跳转 Tez WebUI 界面，已经提交的 Tez 任务未展示，如何解决。

回答

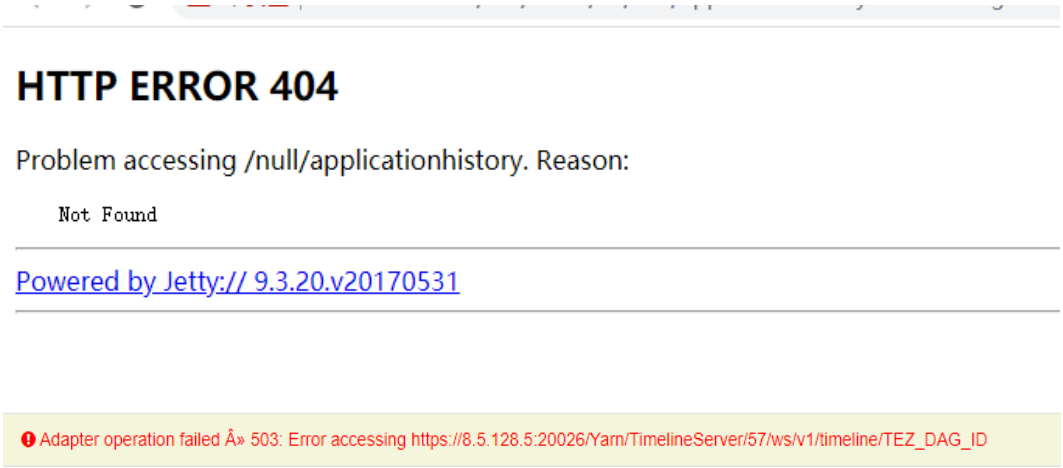
Tez WebUI 展示的 Tez 任务数据，需要 Yarn 的 TimelineServer 支持，确认提交任务之前 TimelineServer 已经开启且正常运行。

在设置 Hive 执行引擎为 Tez 的同时，需要设置参数“yarn.timeline-service.enabled”为“true”，详情请参考切换 Hive 执行引擎为 Tez。

28.5.2 进入 Tez 原生界面显示异常

问题

登录 Manager 界面，跳转 Tez WebUI 界面，显示 404 异常或 503 异常。



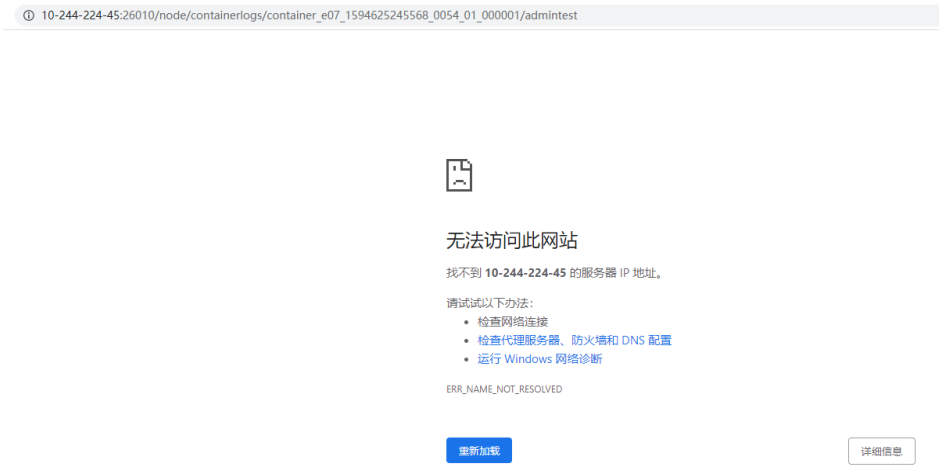
回答

Tez WebUI 依赖 Yarn 的 TimelineServer 实例，需要预先安装 TimelineServer，且处于良好状态。

28.5.3 TezUI 界面无法查看 yarn 日志

问题

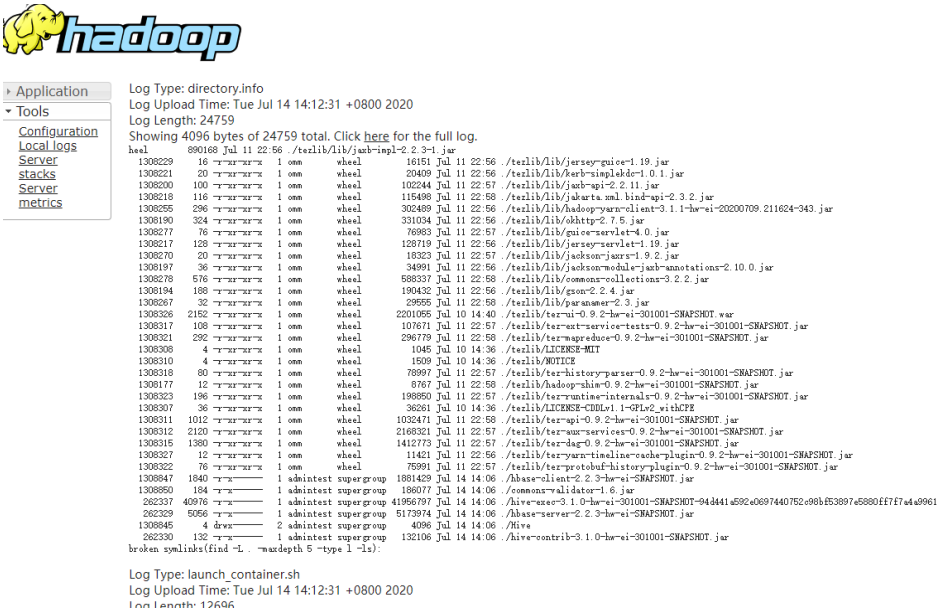
登录 Tez WebUI 界面，单击 Logs 跳转 yarn 日志界面失败，无法加载数据。



回答

Tez WebUI 跳转 Yarn Logs 界面时，目前是通过 hostname 进行访问，需要在 windows 机器，配置 hostname 到 ip 的映射。具体方法为：

修改 windows 机器 C:\Windows\System32\drivers\etc\hosts 文件，增加一行 hostname 到 ip 的映射，例：10.244.224.45 10-044-224-45，保存后重新访问正常。



28.5.4 TezUI HiveQueries 界面表格数据为空

问题

登录 Manager 界面，跳转 Tez WebUI 界面，已经提交的任务，Hive Queries 界面未展示数据，如何解决。

回答

Tez WebUI 展示的 Hive Queries 任务数据，需要设置以下 3 个参数：

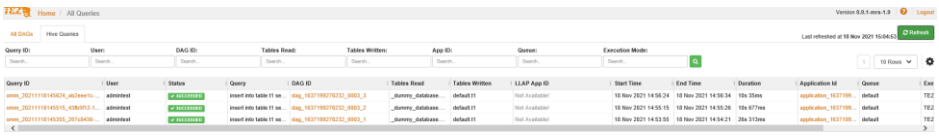
在 FusionInsight Manager 页面，选择“集群 > 服务 > Hive > 配置 > 全部配置 > HiveServer > 自定义”，在 hive-site.xml 中增加以下配置：

属性名	属性值
hive.exec.pre.hooks	org.apache.hadoop.hive.ql.hooks.ATSHook
hive.exec.post.hooks	org.apache.hadoop.hive.ql.hooks.ATSHook
hive.exec.failure.hooks	org.apache.hadoop.hive.ql.hooks.ATSHook

说明

TezUI 数据展示依赖于 Yarn 组件的 TimelineServer 实例，如果 TimelineServer 实例故障或未启动，需设置 hive 自定义参数 yarn-site.xml 中 **yarn.timeline-service.enabled**=false，否则 hive 任务会执行失败。

参数设置完成后，Hive Queries 界面即可展示数据，但无法展示历史数据，展示效果如下:



29 使用 Yarn

29.1 Yarn 常用参数

队列资源分配

Yarn 服务提供队列给用户使用，用户分配对应的系统资源给各队列使用。完成配置后，您可以单击“刷新队列”按钮或者重启 Yarn 服务使配置生效。

参数入口：

MRS 3.x 之前的版本集群执行以下操作：

用户在 MRS 控制台中，选择“租户管理 > 资源分布策略”。

参数说明以 default 为例，其他队列的配置类似，单击“修改”编辑。

表29-1 参数说明

配置参数	说明	默认值
资源容量	队列的资源容量（百分比）。当系统非常繁忙时，应保证每个队列的容量得到满足，而如果每个队列应用程序较少，可将剩余资源共享给其他队列。注意，所有队列的容量之和应小于100。	20
最大资源容量	队列的资源使用上限（百分比）。由于存在资源共享，因此一个队列使用的资源量可能超过其容量，而最多使用资源量可通过该参数限制。	100

MRS 3.x 及后续版本集群执行以下操作：

用户可在 Manager 系统中，选择“租户资源 > 动态资源计划 > 队列配置”。

参数说明以修改 Superior 调度器的 default 租户为例，其他队列的配置类似，单击“修改”编辑。

表29-2 队列配置参数

参数名	描述
AM 最多占有资源（%）	表示当前队列内所有 Application Master 所占的最大资源百分比。
每个 YARN 容器最多分配核数	表示当前队列内单个 YARN 容器可分配的最多核数，默认为-1，表示取值范围内不限制。
每个 YARN 容器最大分配内存（MB）	表示当前队列内单个 YARN 容器可分配的最大内存，默认为-1，表示取值范围内不限制。
最多运行任务数	表示当前队列最多同时可执行任务的数目，默认为-1，表示取值范围内不限制（为空意义相同），为 0 表示不可执行任务。取值范围为-1～2147483647。
每个用户最多运行任务数	表示每个用户在当前队列中最多同时可执行任务的数目，默认为-1，表示取值范围内不限制（为空意义相同），为 0 表示不可执行任务。取值范围为-1～2147483647。
最多挂起任务数	表示当前队列最多同时可挂起任务的数目，默认为-1，表示取值范围内不限制（为空意义相同），为 0 表示不可挂起任务。取值范围为-1～2147483647。
资源分配规则	表示单个用户任务间的资源分配规则，包括 FIFO 和 FAIR。 一个用户若在当前队列上提交了多个任务，FIFO 规则代表一个任务完成后再执行其他任务，按顺序执行。FAIR 规则代表各个任务同时获取到资源并平均分配资源。
默认资源标签	表示在指定资源标签（Label）的节点上执行任务。
Active 状态	• ACTIVE 表示当前队列可接受并执行任务。 • INACTIVE 表示当前队列可接受但不执行任务，若提交任务，任务将处于挂起状态。
Open 状态	• OPEN 表示当前队列处于打开状态。 • CLOSED 表示当前队列处于关闭状态，若提交任务，任务直接会被拒绝。

在 UI 显示 container 日志

默认情况下，系统会将 container 日志收集到 HDFS 中。如果您不需要将 container 日志收集到 HDFS 中，可以配置参数见表 29-3。具体配置操作请参考修改集群服务配置参数。

表29-3 参数说明

配置参数	说明	默认值

配置参数	说明	默认值
yarn.log-aggregation-enable	设置是否将 container 日志收集到 HDFS 中。 • 设置为 true，表示日志会被收集到 HDFS 目录中。默认目录为“{yarn.nodemanager.remote-app-log-dir}/${user}/{thisParam}”，该路径可通过界面上的“yarn.nodemanager.remote-app-log-dir-suffix”参数进行配置。 • 设置为 false，表示日志不会收集到 HDFS 中。 修改参数值后，需重启 Yarn 服务使其生效。 说明 在修改值为 false 并生效后，生效前的日志无法在 UI 中获取。您可以在“yarn.nodemanager.remote-app-log-dir-suffix”参数指定的路径中获取到生效前的日志。 如果需要在 UI 上查看之前产生的日志，建议将此参数设置为 true。	true

在 WebUI 显示更多历史作业

默认情况下，Yarn WebUI 界面支持任务列表分页功能，每个分页最多显示 5000 条历史作业，总共最多保留 10000 条历史作业。如果您需要在 WebUI 上查看更多的作业，可以配置参数如表 29-4。具体配置操作请参考修改集群服务配置参数。

表29-4 参数说明

配置参数	说明	默认值
yarn.resourcemanager.max-completed-applications	设置在 WebUI 总共显示的历史作业数量。	10000
yarn.resourcemanager.webapp.pagination.enable	是否开启 Yarn WebUI 的任务列表后台分页功能。	true
yarn.resourcemanager.webapp.pagination.threshold	开启 Yarn WebUI 的任务列表后台分页功能后，每个分页显示的最大作业数量。	5000

说明

- 显示更多的历史作业，会影响性能，增加打开 Yarn WebUI 的时间，建议开启后台分页功能，并根据实际硬件性能修改“yarn.resourcemanager.max-completed-applications”参数。

- 修改参数值后，需重启 Yarn 服务使其生效。

29.2 创建 Yarn 角色

操作场景

该任务指导系统管理员创建并设置 Yarn 的角色。Yarn 角色可设置 Yarn 管理员权限以及 Yarn 队列资源管理。

说明

如果当前组件使用了 Ranger 进行权限控制，须基于 Ranger 配置相关策略进行权限管理。对于 MRS 3.x 及后续版本集群，具体操作可参考添加 Yarn 的 Ranger 访问权限策略。

前提条件

- 系统管理员已明确业务需求。
- 登录 Manager。

操作步骤

MRS 3.x 以前版本集群执行以下操作：

步骤 1 选择“系统设置 > 角色管理 > 添加角色”。

步骤 2 在“角色名称”和“描述”输入角色名字与描述。

步骤 3 设置角色“权限”请参见表 29-5。

Yarn 权限：

- “Cluster Admin Operations”：Yarn 管理员权限。
- “Scheduler Queue”：队列资源管理。

表29-5 设置角色

任务场景	角色授权操作
设置 Yarn 管理员权限	在“权限”的表格中选择“Yarn”，勾选“Cluster Admin Operations”。 说明 设置 Yarn 管理员权限需要重启 Yarn 服务，才能使保存的角色配置生效。
设置用户在指定 Yarn 队列提交任务的权限	1. 在“权限”的表格中选择“Yarn > Scheduler Queue”。 2. 在指定队列的“权限”列，勾选“Submit”。
设置用户在指定 Yarn 队列管理任务	1. 在“权限”的表格中选择“Yarn > Scheduler Queue”。

任务场景	角色授权操作
的权限	2. 在指定队列的“权限”列，勾选“Admin”。

如果 Yarn 角色包含了某个父队列的“提交”或“管理”权限，则角色默认子队列也继承此权限，将自动添加子队列的“提交”或“管理”权限。子队列继承的权限不在“配置资源权限”表格显示被选中。

如果设置 Yarn 角色时仅勾选到某个父队列的“提交”权限，使用拥有该角色权限的用户提交任务时，注意需要手动指定队列名称，否则当父队列下有多个子队列时，系统并不会自动判断，从而将任务提交到了“default”队列。

步骤 4 单击“确定”完成。

----结束

MRS 3.x 及以后版本集群执行以下操作：

步骤 1 选择“系统 > 权限 > 角色”。

步骤 2 单击“添加角色”，然后“角色名称”和“描述”输入角色名字与描述。

步骤 3 设置角色“配置资源权限”请参见表 29-6。

Yarn 权限：

- “集群管理操作权限”：Yarn 管理员权限。
- “调度队列”：队列资源管理。

表29-6 设置角色

任务场景	角色授权操作
设置 Yarn 管理员权限	在“配置资源权限”的表格中选择“*待操作集群的名称* > Yarn”，勾选“集群管理操作权限”。 **说明** **设置 Yarn 管理员权限需要重启 Yarn 服务，才能使保存的角色配置生效。**
设置用户在指定 Yarn 队列提交任务的权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > Yarn > 调度队列 > root”。 2. 在指定队列的“权限”列，勾选“提交”。
设置用户在指定 Yarn 队列管理任务的权限	1. 在“配置资源权限”的表格中选择“*待操作集群的名称* > Yarn > 调度队列 > root”。 2. 在指定队列的“权限”列，勾选“管理”。

如果 Yarn 角色包含了某个父队列的“提交”或“管理”权限，则角色默认子队列也继承此权限，将自动添加子队列的“提交”或“管理”权限。子队列继承的权限不在“配置资源权限”表格显示被选中。

如果设置 Yarn 角色时仅勾选到某个父队列的“提交”权限，使用拥有该角色权限的用户提交任务时，注意需要手动指定队列名称，否则当父队列下有多个子队列时，系统并不会自动判断，从而将任务提交到了“default”队列。

步骤 4 单击“确定”完成。

----结束

29.3 使用 Yarn 客户端

操作场景

该任务指导用户在运维场景或业务场景中使用 Yarn 客户端。

前提条件

- 已安装客户端。

  例如安装目录为“/opt/hadoopclient”，以下操作的客户端目录只是举例，请根据实际安装目录修改。
- 各组件业务用户由系统管理员根据业务需要创建。安全模式下，“机机”用户需要下载 keytab 文件。“人机”用户第一次登录时需修改密码。普通模式不需要下载 keytab 文件及修改密码操作。

使用 Yarn 客户端

步骤 1 以客户端安装用户，登录安装客户端的节点。

步骤 2 执行以下命令，切换到客户端安装目录。

cd /opt/hadoopclient

步骤 3 执行以下命令配置环境变量。

source bigdata_env

步骤 4 如果集群为安全模式，执行以下命令进行用户认证。普通模式集群无需执行用户认证。

kinit *组件业务用户*

步骤 5 直接执行 Yarn 命令。例如：

yarn application -list

----结束

客户端常见使用问题

1. 当执行 Yarn 客户端命令时，客户端程序异常退出，报“java.lang.OutOfMemoryError”的错误。

   这个问题是由于 Yarn 客户端运行时的所需的内存超过了 Yarn 客户端设置的内存上限（默认为 128MB）。对于 MRS 3.x 后续版本集群，可以通过修改“<客户端安装路径>/HDFS/component_env”中的“CLIENT_GC_OPTS”来修改 Yarn 客户端的内存上限。例如，需要设置该内存上限为 1GB，则设置：

   ```
   export CLIENT_GC_OPTS="-Xmx1G"
   ```

   对于 MRS 3.x 之前版本集群，可以通过修改“<客户端安装路径>/HDFS/component_env”中的“GC_OPTS_YARN”来修改 Yarn 客户端的内存上限。例如，需要设置该内存上限为 1GB，则设置：

   ```
   export GC_OPTS_YARN="-Xmx1G"
   ```

   在修改完后，使用如下命令刷新客户端配置，使之生效：

   source <*客户端安装路径*>**/bigdata_env**

2. 如何设置 Yarn 客户端运行时的日志级别？

   Yarn 客户端运行时的日志是默认输出到 Console 控制台的，其级别默认是 INFO 级别。有的时候为了定位问题，需要开启 DEBUG 级别日志，可以通过导出一个环境变量来设置，命令如下：

   export YARN_ROOT_LOGGER=DEBUG,console

   在执行完上面命令后，再执行 Yarn Shell 命令时，即可打印出 DEBUG 级别日志。

   如果想恢复 INFO 级别日志，可执行如下命令：

   export YARN_ROOT_LOGGER=INFO,console

29.4 配置 NodeManager 角色实例使用的资源

操作场景

如果部署 NodeManager 的各个节点硬件资源（如 CPU 核数、内存总量）不一样，而 NodeManager 可用硬件资源设置为相同的值，可能造成性能浪费或状态异常，需要修改各个 NodeManager 角色实例的配置，使硬件资源得到充分利用。

对系统的影响

保存新的配置需要重启 NodeManager 角色实例，此时对应的角色实例不可用。

前提条件

- MRS 3.x 之前的版本集群：已登录 MRS 控制台。
- MRS 3.x 及后续版本集群：已登录 Manager。

操作步骤

MRS 3.x 之前的版本集群执行以下操作：

步骤 1 选择“集群列表 > 现有集群”，单击集群名称。选择“组件管理 > Yarn > 实例”。

步骤 2 单击“角色”列“NodeManager”角色实例名称，并切换到“实例配置”。单击“基础配置”下拉菜单，选择“全部配置”，在搜索框中输入以下参数。

步骤 3 “yarn.nodemanager.resource.cpu-vcores”设置当前节点上 NodeManager 可使用的虚拟 CPU 核数，建议按节点实际逻辑核数的 1.5 到 2 倍配置。
“yarn.nodemanager.resource.memory-mb”设置当前节点上 NodeManager 可使用的物理内存大小，建议按节点实际物理内存大小的 75%~90%配置。

说明

“yarn.scheduler.maximum-allocation-vcores”可配置单个 Container 最多 CPU 可用核数，
“yarn.scheduler.maximum-allocation-mb”可配置单个 Container 最大内存可用值。不支持实例级别的修改，需要在 Yarn 服务的配置中修改参数值，并重启 Yarn 服务。

步骤 4 单击“保存配置”，勾选“重新启动受影响的服务或实例”，单击“确定”。重启 NodeManager 角色实例。

界面提示“操作成功。”，单击“完成”，NodeManager 角色实例成功启动。

----结束

MRS 3.x 及后续版本集群也可执行以下操作：

步骤 1 选择“集群 > *待操作集群的名称* > 服务 > Yarn > 实例”。

步骤 2 单击部署 NodeManager 节点对应角色实例名称，并切换到“实例配置”，选择“全部配置”。

步骤 3 “yarn.nodemanager.resource.cpu-vcores”设置当前节点上 NodeManager 可使用的虚拟 CPU 核数，建议按节点实际逻辑核数的 1.5 到 2 倍配置。
“yarn.nodemanager.resource.memory-mb”设置当前节点上 NodeManager 可使用的物理内存大小，建议按节点实际物理内存大小的 75%配置。

说明

“yarn.scheduler.maximum-allocation-vcores”可配置单个 Container 最多 CPU 可用核数，
“yarn.scheduler.maximum-allocation-mb”可配置单个 Container 最大内存可用值。不支持实例级别的修改，需要在 Yarn 服务的配置中修改参数值，并重启 Yarn 服务。

步骤 4 单击“保存”，单击“确定”。重启 NodeManager 角色实例。

界面提示“操作成功”，单击“完成”，NodeManager 角色实例成功启动。

----结束

29.5 更改 NodeManager 的存储目录

操作场景

Yarn NodeManager 定义的存储目录不正确或 Yarn 的存储规划变化时，系统管理员需要在 Manager 中修改 NodeManager 的存储目录，以保证 Yarn 正常工作。NodeManager 的存储目录包含本地存放目录“yarn.nodemanager.local-dirs”和日志目录“yarn.nodemanager.log-dirs”。适用于以下场景：

- 更改 NodeManager 角色的存储目录，所有 NodeManager 实例的存储目录将同步修改。
- 更改 NodeManager 单个实例的存储目录，只对单个实例生效，其他节点 NodeManager 实例存储目录不变。

对系统的影响

- 更改 NodeManager 角色的存储目录需要停止并重新启动集群，集群未启动前无法提供服务。
- 更改 NodeManager 单个实例的存储目录需要停止并重新启动实例，该节点 NodeManager 实例未启动前无法提供服务。
- 服务参数配置如果使用旧的存储目录，需要更新为新目录。
- 更改 NodeManager 的存储目录以后，需要重新下载并安装客户端。

前提条件

- 在各个数据节点准备并安装好新磁盘，并格式化磁盘。
- 规划好新的目录路径，用于保存旧目录中的数据。
- 准备好系统管理员用户 **admin**。

操作步骤

MRS 3.x 之前的版本集群执行以下操作：

步骤 1 检查环境。

1. 登录 MRS 控制台，在左侧导航栏选择“集群列表 > 现有集群”，单击集群名称。选择“组件管理”，查看 Yarn 的“健康状态”是否为“良好”。
   - 是，执行步骤 1.3。
   - 否，Yarn 状态不健康，执行步骤 1.2。
2. 请先修复 Yarn 异常，任务结束。
3. 确定修改 NodeManager 的存储目录场景。
   - 更改 NodeManager 角色的存储目录，执行步骤 2。
   - 更改 NodeManager 单个实例的存储目录，执行步骤 3。

步骤 2 更改 NodeManager 角色的存储目录。

1. 选择“集群列表 > 现有集群”，单击集群名称。选择“组件管理 > Yarn > 停止”，停止 Yarn 服务。
2. 登录弹性云主机，以 **root** 用户登录到安装 Yarn 服务的各个节点中，执行如下操作。
   a. 创建目标目录。

      例如目标目录为“${BIGDATA_DATA_HOME}/data2”：

      执行 **mkdir ${BIGDATA_DATA_HOME}/data2**

   b. 挂载目标目录到新磁盘。

      例如挂载“${BIGDATA_DATA_HOME}/data2”到新磁盘。

   c. 修改新目录的权限。

      例如新目录路径为“${BIGDATA_DATA_HOME}/data2”：

      执行 **chmod 750 ${BIGDATA_DATA_HOME}/data2 -R** 和 **chown omm:wheel ${BIGDATA_DATA_HOME}/data2 -R**
3. 在 MRS 控制台界面，选择“集群列表 > 现有集群”，单击集群名称。选择“组件管理 > Yarn > 实例”，选择对应主机的 NodeManager 实例，单击“实例配置”，“选择”“全部配置”。

   将配置项“yarn.nodemanager.local-dirs”或“yarn.nodemanager.log-dirs”修改为新的目标目录。

   例如：如果修改“yarn.nodemanager.local-dirs”参数，则将其值修改为“/srv/BigData/data2/nm/localdir”。如果修改“yarn.nodemanager.log-dirs”参数，则将其值修改为“/srv/BigData/data2/nm/containerlogs”。
4. 单击“保存配置”，勾选“重新启动受影响的服务或实例”，单击“确定”。重启 Yarn 服务。

   界面提示“操作成功”，单击“完成”，Yarn 成功启动，任务结束。

步骤 3 更改 NodeManager 单个实例的存储目录。

1. 选择“集群列表 > 现有集群”，单击集群名称。选择“组件管理 > Yarn > 实例”，勾选需要修改存储目录的 NodeManager 单个实例，选择“更多 > 停止实例”。
2. 登录弹性云主机，以 **root** 用户登录到这个 NodeManager 节点，执行如下操作。
   a. 创建目标目录。

      例如目标目录为“${BIGDATA_DATA_HOME}/data2”：

      执行 **mkdir ${BIGDATA_DATA_HOME}/data2**。

   b. 挂载目标目录到新磁盘。

      例如挂载“${BIGDATA_DATA_HOME}/data2”到新磁盘。

   c. 修改新目录的权限。

      例如新目录路径为“${BIGDATA_DATA_HOME}/data2”：

      执行 **chmod 750 ${BIGDATA_DATA_HOME}/data2 -R** 和 **chown omm:wheel ${BIGDATA_DATA_HOME}/data2 -R**。
3. 在 MRS 控制台，单击指定的 NodeManager 实例并切换到“实例配置”。

将配置项“yarn.nodemanager.local-dirs”或“yarn.nodemanager.log-dirs”修改为新的目标目录。

例如：如果修改“yarn.nodemanager.local-dirs”参数，则将其值修改为“/srv/BigData/data2/nm/localdir”。如果修改“yarn.nodemanager.log-dirs”参数，则将其值修改为“/srv/BigData/data2/nm/containerlogs”。

4. 单击“保存配置”，勾选“重新启动受影响的服务或实例”。单击“确定”。重启 NodeManager 实例。

   界面提示“操作成功”，单击“完成”，NodeManager 实例启动成功。

----结束

MRS 3.x 及后续版本集群也可执行以下操作：

步骤 1 检查环境。

1. 登录 Manager，选择“集群 > *待操作集群的名称* > 服务”查看 Yarn 的状态“运行状态”是否为“良好”。
   - 是，执行 1.c。
   - 否，Yarn 状态不健康，执行 1.b。
2. 修复 Yarn 异常，任务结束。
3. 确定修改 NodeManager 的存储目录场景。
   - 更改 NodeManager 角色的存储目录，执行 2。
   - 更改 NodeManager 单个实例的存储目录，执行 3。

步骤 2 更改 NodeManager 角色的存储目录。

1. 选择“集群 > *待操作集群的名称* > 服务 > Yarn > 停止服务”，停止 Yarn 服务。
2. 以 **root** 用户登录到安装 Yarn 服务的各个节点中，执行如下操作。
   a. 创建目标目录。

      例如目标目录为“${BIGDATA_DATA_HOME}/data2”：

      执行 **mkdir ${BIGDATA_DATA_HOME}/data2**

   b. 挂载目标目录到新磁盘。

      例如挂载“${BIGDATA_DATA_HOME}/data2”到新磁盘。

   c. 修改新目录的权限。

      例如新目录路径为“${BIGDATA_DATA_HOME}/data2”：

      执行 **chmod 750 ${BIGDATA_DATA_HOME}/data2 -R** 和 **chown omm:wheel ${BIGDATA_DATA_HOME}/data2 -R**
3. 在 Manager 管理界面，选择“集群 > *待操作集群的名称* > 服务 > Yarn > 实例”，选择对应主机的 NodeManager 实例，单击“实例配置”，选择“全部配置”。

   将配置项“yarn.nodemanager.local-dirs”或“yarn.nodemanager.log-dirs”修改为新的目标目录。

例如：如果修改“yarn.nodemanager.local-dirs”参数，则将其值修改为“/srv/BigData/data2/nm/localdir”。如果修改“yarn.nodemanager.log-dirs”参数，则将其值修改为“/srv/BigData/data2/nm/containerlogs”。

4. 单击“保存”，单击“确定”。重启 Yarn 服务。

   界面提示“操作成功”，单击“完成”，Yarn 成功启动，任务结束。

步骤 3 更改 NodeManager 单个实例的存储目录。

1. 选择“集群 > *待操作集群的名称* > 服务 > Yarn > 实例”，勾选需要修改存储目录的 NodeManager 单个实例，选择“更多 > 停止实例”。
2. 以 **root** 用户登录到这个 NodeManager 节点，执行如下操作。
   a. 创建目标目录。

      例如目标目录为“${BIGDATA_DATA_HOME}/data2”：

      执行 **mkdir ${BIGDATA_DATA_HOME}/data2**。

   b. 挂载目标目录到新磁盘。

      例如挂载“${BIGDATA_DATA_HOME}/data2”到新磁盘。

   c. 修改新目录的权限。

      例如新目录路径为“${BIGDATA_DATA_HOME}/data2”：

      执行 **chmod 750 ${BIGDATA_DATA_HOME}/data2 -R** 和 **chown omm:wheel ${BIGDATA_DATA_HOME}/data2 -R**。
3. 在 Manager 管理界面，单击指定的 NodeManager 实例并切换到“实例配置”。

   将配置项“yarn.nodemanager.local-dirs”或“yarn.nodemanager.log-dirs”修改为新的目标目录。

   例如：如果修改“yarn.nodemanager.local-dirs”参数，则将其值修改为“/srv/BigData/data2/nm/localdir”。如果修改“yarn.nodemanager.log-dirs”参数，则将其值修改为“/srv/BigData/data2/nm/containerlogs”。
4. 单击“保存”，单击“确定”。重启 NodeManager 实例。

   界面提示“操作成功”，单击“完成”，NodeManager 实例启动成功。

----结束

29.6 配置 YARN 严格权限控制

配置场景

在安全模式的多租户场景下，一个集群可以支持多个用户使用以及支持多个用户任务提交、运行，用户之间是不可见，需要有一个权限控制机制，使用户的任务信息不被其他用户获取。

例如，用户 A 提交的应用正在运行，此时用户 B 登录系统并查看应用列表，用户 B 不应该访问到 A 用户的应用信息。

配置描述

- 查看 Yarn 服务配置参数

  参考修改集群服务配置参数进入 Yarn 服务参数“全部配置”界面，在搜索框中输入表 29-7 中参数名称。

表29-7 参数描述

参数	描述	默认值
yarn.acl.enable	Yarn 权限控制启用开关。	true
yarn.webapp.filter-entity-list-by-user	严格视图启用开关，开启后，登录用户只能查看该用户有权限查看的内容。当要开启该功能时，同时需要设置参数“yarn.acl.enable”为 true。 说明 此参数适用于 MRS 3.x 及后续版本集群。	true

- 查看 Mapreduce 服务配置参数

  参考修改集群服务配置参数进入 Mapreduce 服务参数“全部配置”界面，在搜索框中输入表 29-8 中参数名称。

表29-8 参数描述

参数	描述	默认值
mapreduce.cluster.acls.enabled	MR JobHistoryServer 权限控制启用开关。该参数为客户端参数，当 JobHistoryServer 服务端开启权限控制之后该参数生效。	true
yarn.webapp.filter-entity-list-by-user	MR JobHistoryServer 严格视图启用开关，开启后，登录用户只能查看该用户有权限查看的内容。该参数为 JobHistoryServer 的服务端参数，表示 JHS 开启了权限控制，但是否要对某一个特定的 Application 进行控制，是由客户端参数：“mapreduce.cluster.acls.enabled”决定。 说明 此参数适用于 MRS 3.x 及后续版本集群。	true

须知

以上配置会影响 restful API 和 shell 命令结果，即以上配置开启后，restful API 调用和 shell 命令运行所返回的内容只包含调用用户有权查看的信息。

当 yarn.acl.enable 或 mapreduce.cluster.acls.enabled 设置为 false 时，即关闭 Yarn 或 Mapreduce 的权限校验功能。此时任何用户都可以在 Yarn 或 MapReduce 上提交任务和查看任务信息，存在安全风险，请谨慎使用。

29.7 配置 Container 日志聚合功能

配置场景

YARN 提供了 Container 日志聚合功能，可以将各节点 Container 产生的日志收集到 HDFS，释放本地磁盘空间。日志收集的方式有两种：

- 应用完成后将 Container 日志一次性收集到 HDFS。
- 应用运行过程中周期性收集 Container 输出的日志片段到 HDFS。

配置描述

参数入口：

参考修改集群服务配置参数进入 Yarn 服务参数“全部配置”界面，在搜索框中输入表 29-9 中参数名称，修改并保存配置。然后在 Yarn 服务“概览”页面选择“更多 > 同步配置”。同步完成后重启 Yarn 服务。

其中“yarn.nodemanager.remote-app-log-dir-suffix”参数还需要在 YARN 的客户端进行配置，且在 ResourceManager、NodeManager 和 JobHistory 节点的配置与在 YARN 的客户端的配置必须一致。

周期性收集日志功能目前仅支持 MapReduce 应用，且 MapReduce 应用必须进行相应的日志文件滚动输出配置，需要在 MapReduce 客户端节点的“mapred-site.xml”配置文件中进行如表 29-11 所示的配置。

表29-9 参数说明

参数	描述	默认值
yarn.log-aggregation-enable	设置是否打开 Container 日志聚合功能。 • 设置为“true”，表示打开该功能，日志会被收集到 HDFS 目录中。 • 设置为“false”，表示关闭该功能，表示日志不会收集到 HDFS 中。 修改参数值后，需重启 YARN 服务使其生效。 说明	true

参数	描述	默认值
	• 在修改值为“false”并生效后，生效前的日志无法在 WebUI 中获取。 • 如果需要在 WebUI 界面上查看之前产生的日志，建议将此参数设置为“true”。	
yarn.nodemanager.log-aggregation.roll-monitoring-interval-seconds	NodeManager 周期性日志收集的时间间隔。 • 设置为-1 或 0 时，表示周期性收集日志功能关闭，日志在应用运行完成后一次性收集。 • 收集周期最小可设定为 3600 秒。当设置为大于 0 秒且小于 3600 秒时，收集周期将使用 3600 秒。 定义 NodeManager 唤醒并上传日志的间隔周期。设置为-1 或 0 表示禁用滚动监控，应用任务结束后日志汇聚。取值范围大于等于-1。	-1
yarn.nodemanager.disk-health-checker.log-dirs.max-disk-utilization-per-disk-percentage	配置 Container 日志目录可以占用每块磁盘上 YARN 的磁盘配额的最大百分比。当日志目录占用空间超过此设定值时，将触发周期性日志收集服务启动一次周期外的日志收集活动，以释放本地磁盘空间。每个磁盘上可提供给 Container logs 的最大可使用率。当 Container logs 使用超过这个限制，会触发滚动汇聚。 • 对于 MRS 3.x 之前的版本集群，磁盘配额最大百分比的有效取值范围为 0～100，如果配置小于等于 0，会被强制重置为 25；如果配置大于 100，则被强制重置为 25。 • 对于 MRS 3.x 及后续版本集群，磁盘配额最大百分比的有效取值范围为-1～100，如果配置小于-1，会被强制重置为 25；如果配置大于 100，则被强制重置为 25。而配置为-1 时则关闭 Container 日志目录的磁盘容量检测功能。 说明 • Container 日志目录实际可用磁盘百分比=YARN 磁盘可用百分比（“yarn.nodemanager.disk-health-checker.max-disk-utilization-per-disk-percentage”） * 日志目录可用百分比（“yarn.nodemanager.disk-health-checker.log-dirs.max-disk-utilization-per-disk-percentage”）。 • 只有启用了周期性收集日志功能的应用才会在日志目录磁盘配额超过设定阈值时被触发启动日志收集。	25

参数	描述	默认值
yarn.nodemanager.remote-app-log-dir-suffix	设置 HDFS 用于存放 Container 日志的文件夹名称。该配置加上“yarn.nodemanager.remote-app-log-dir”，构成了 Container 日志的完整存放目录。目录为：“{yarn.nodemanager.remote-app-log-dir}/${user}/{yarn.nodemanager.remote-app-log-dir-suffix}”。 说明 *{user}*为运行任务时的用户名。	logs
yarn.nodemanager.log-aggregator.on-fail.remain-log-in-sec	设置 Container 日志归集失败后日志在本地保留的时间。单位：秒。 • 设置为 0 时，本地日志将马上删除。 • 设置为正数时，表示本地日志将保留这段时间。	604800

参考修改集群服务配置参数进入 Mapreduce 服务参数“全部配置”界面，在搜索框中输入表 29-10 中参数名称。

表29-10 参数说明

参数	描述	默认值
yarn.log-aggregation.retain-seconds	汇聚日志的保存时间。单位：秒。 • 设置为-1 时，表示 HDFS 上面的 Container 聚合日志将永久保留。 • 设置为 0 或正数时，表示 HDFS 上面的 Container 聚合日志将保留这段时间，超时将被删除。 说明 当时间设置太短时，有可能会增加 NameNode 的负担，建议根据实际情况设置一个合理的时间值。	1296000
yarn.log-aggregation.retain-check-interval-seconds	设置扫描 HDFS 保存的 Container 聚合日志的间隔时间。单位：秒。 • 设置为-1 或 0 时，间隔时间将为“yarn.log-aggregation.retain-seconds”该配置时间的十分之一。 说明 当该配置设置为-1 或 0 时，“yarn.log-aggregation.retain-seconds”不能设置为 0。 • 设置为正数时，将周期性的间隔这段时间以后对 HDFS 上的 container 聚合日志进行扫描。	86400

参数	描述	默认值
	说明 当时间设置太短时，有可能会增加 NameNode 的负担，建议根据实际情况设置一个合理的时间。	

参考修改集群服务配置参数进入 Yarn 服务参数“全部配置”界面，在搜索框中输入表29-11 中参数名称。

表29-11 MapReduce 应用日志文件滚动输出配置

参数	描述	默认值
mapreduce.task.userlog.limit.kb	MR 应用程序单个 task 日志文件大小限制。当日志文件达到该限制时，会新建一个日志文件进行输出。设置为“0”表示不限制日志文件大小。	51200
yarn.app.mapreduce.task.container.log.backups	MR 应用程序 task 日志保留的最大个数。 设置为“0”表示不滚动输出。 使用 CRLA（ContainerRollingLogAppender）时任务日志备份文件的数量。默认使用 CLA（ContainerLogAppender）且 container 日志不回滚。 当 mapreduce.task.userlog.limit.kb 和 yarn.app.mapreduce.task.container.log.backups 都大于 0 时，任务启用 CRLA。取值范围 0~999。	10
yarn.app.mapreduce.am.container.log.limit.kb	MR 应用程序单个 AM 日志文件大小限制。单位：KB，当日志文件达到该限制时，会新建一个日志文件进行输出。设置为“0”表示不限制单个 AM 日志文件大小。	51200
yarn.app.mapreduce.am.container.log.backups	MR 应用程序 AM 日志保留的最大个数。设置为“0”表示不滚动输出。使用 CRLA（ContainerRollingLogAppender）时 ApplicationMaster 日志备份文件的数量。默认使用 CLA（ContainerLogAppender）且容器日志不回滚。 当 yarn.app.mapreduce.am.container.log.limit.kb 和 yarn.app.mapreduce.am.container.log.backups 都大于 0 时，ApplicationMaster 启用 CRLA。取值范围 0~999。	20
yarn.app.mapreduce.shuffle.log.backups	MR 应用程序 shuffle 日志保留的最大个数。设置为“0”表示不滚动输出。 当 yarn.app.mapreduce.shuffle.log.limit.kb 和	10

参数	描述	默认值
	yarn.app.mapreduce.shuffle.log.backups 都大于 0 时，syslog.shuffle 将采用 CRLA。取值范围 0~999。	
yarn.app.mapreduce.shuffle.log.limit.kb	MR 应用程序单个 shuffle 日志文件大小限制，单位 KB。当日志文件达到该限制时，会新建一个日志文件进行输出。设置为“0”不限制单个 shuffle 日志文件大小。取值范围大于等于 0。	51200

29.8 启用 CGroups 功能

本章节适用于 MRS 3.x 及后续版本集群。

配置场景

CGroups 是一个 Linux 内核特性。它可以将任务集及其子集聚合或分离成具备特定行为的分层组。在 YARN 中，CGroups 特性对容器（container）使用的资源（例如 CPU 使用率）进行限制。本特性大大降低了限制容器 CPU 使用的难度。

说明

当前 CGroups 仅用于限制 CPU 使用率。

配置描述

有关如何配置 CPU 隔离与安全的 CGroups 功能的详细信息，请参见 Hadoop 官网：http://hadoop.apache.org/docs/r3.1.1/hadoop-yarn/hadoop-yarn-site/NodeManagerCgroups.html

由于 CGroups 为 Linux 内核特性，是通过 **LinuxContainerExecutor** 进行开放。请参考官网资料对 **LinuxContainerExecutor** 进行安全配置。您可通过官网资料了解系统用户和用户组配置对应的文件系统权限。详情请参见：http://hadoop.apache.org/docs/r3.1.1/hadoop-project-dist/hadoop-common/SecureMode.html#LinuxContainerExecutor

说明

- 请勿修改对应文件系统中各路径所属的用户、用户组及对应的权限，否则可能导致本功能异常。
- 当参数“yarn.nodemanager.resource.percentage-physical-cpu-limit”配置过小，导致可使用的核不足 1 个时，例如 4 核节点，将此参数设置为 20%，不足 1 个核，那么将会使用系统全部的核。Linux 的一些版本不支持 Quota 模式，例如 Cent OS。在这种情况下，我们可以使用 CPUset 模式。

配置 cpuset 模式，即 YARN 只能使用配置的 CPU，需要添加以下配置。

表29-12 cpuset 配置

参数	描述	默认值
yarn.nodemanager.linux-container-executor.cgroups.cpu-set-usage	设置为“true”时，应用以cpuset 模式运行。	false

配置 strictcpuset 模式，即 container 只能使用配置的 CPU，需要添加以下配置。

表29-13 CPU 硬隔离参数配置

参数	描述	默认值
yarn.nodemanager.linux-container-executor.cgroups.cpu-set-usage	设置为“true”时，应用以cpuset 模式运行。	false
yarn.nodemanager.linux-container-executor.cgroups.cpuset.strict.enabled	设置为 true 时，container只能使用配置的 CPU。	false

要从 cpuset 模式切换到 Quota 模式，必须遵循以下条件：

- 配置“yarn.nodemanager.linux-container-executor.cgroups.cpu-set-usage”=“false”。
- 删除 container 文件夹（如果存在）。
- 删除 cpuset.cpus 文件中设置的所有 CPU。

操作步骤

步骤 1 登录 Manager 系统。选择“集群 > *待操作集群的名称* > 服务 > Yarn > 配置”，选择“全部配置”。

步骤 2 在左侧导航栏选择“NodeManager > 自定义”，找到 yarn-site.xml 文件。

步骤 3 添加表 29-12 和表 29-13 中的参数为自定义参数。

根据配置文件与参数作用，在“yarn-site.xml”所在行“名称”列输入参数名，在“值”列输入此参数的参数值。

单击“＋”增加自定义参数。

步骤 4 单击“保存”，在弹出的“保存配置”窗口中确认修改参数，单击“确定”。界面提示“操作成功”，单击“完成”，配置保存成功。

保存完成后请重新启动配置过期的 Yarn 服务以使配置生效。

----结束

29.9 配置 AM 失败重试次数

配置场景

在资源不足导致 ApplicationMaster 启动失败的情况下，调整如下参数值，提高容错性，保证客户端应用的正常运行。

配置描述

参考修改集群服务配置参数进入 Yarn 服务参数“全部配置”界面，在搜索框中输入表29-14 中参数名称。

表29-14 参数说明

参数	描述	默认值
yarn.resourcemanager.am.max-attempts	ApplicationMaster 重试次数，增加重试次数，可以防止资源不足导致的 AM 启动失败问题。适用于所有 ApplicationMaster 的全局设置。每个 ApplicationMaster 都可以使用 API 设置一个单独的最大尝试次数，但这个次数不能大于全局的最大次数。如果大于了，那 ResourceManager 将会覆写这个单独的最大尝试次数。以允许至少一次重试。取值范围大于等于 1。	5

29.10 配置 AM 自动调整分配内存

本章节适用于 MRS 3.x 及后续版本集群。

配置场景

启动该配置的过程中，ApplicationMaster 在创建 container 时，分配的内存会根据任务总数的浮动自动调整，资源利用更加灵活，提高了客户端应用运行的容错性。

配置描述

参数入口：

在 Manager 系统中，选择“集群 > *待操作集群的名称* > 服务 > Yarn > 配置”，选择“全部配置”，在搜索框中输入参数名称“mapreduce.job.am.memory.policy”。

配置说明：

配置项的默认值为空，此时不会启动自动调整的策略，ApplicationMaster 的内存仍受“yarn.app.mapreduce.am.resource.mb”配置项的影响。

配置参数的值由 5 个数值组成，中间使用“:”与“,”分隔，格式为：**baseTaskCount:taskStep:memoryStep,minMemory:maxMemory**，在键入时会严格校验格式。

表29-15 配置数值说明

数值名称	描述	设定要求
baseTaskCount	任务总量基数，只有当应用的 task 总数（map 端与 reduce 端之和）不小于该值时配置才会起作用	不能为空且大于零
taskStep	任务增量步进，与 memoryStep 共同决定内存调整量	不能为空且大于零
memoryStep	内存增量步进，在"yarn.app.mapreduce.am.resource.mb"配置的基础上对内存向上调整	不能为空且大于零，单位：MB
minMemory	内存自动调整下限，若调整后的内存不大于该值，仍保持"yarn.app.mapreduce.am.resource.mb"的配置	不能为空且大于零，且不大于 maxMemory 的设定值 单位：MB
maxMemory	内存自动调整上限，若调整后的内存超过该值，则使用该值作为最终调整值	不能为空且大于零，且不小于 minMemory 的设定值 单位：MB

配置示例

配置情况：

- yarn.app.mapreduce.am.resource.mb=1536
- mapreduce.job.am.memory.policy=100:10:50,1200:2000
- 某应用 task 总数=120

计算过程：

调整后内存=1536+[（120-100）/10]*50=1636，满足 1200<1636 且 2000>1636，最终 ApplicationMaster 内存会设定为 1636MB。

若 memStep 修改为 250，调整后内存=1536+[（120-100）/10]*250=2136，超过 maxMemory=2000 的限制，最终 ApplicationMaster 内存会设定为 2000MB。

说明

对于计算后的调整值低于设定的“minMemory”值的情形，虽然此时配置不会生效但后台仍然会打印出这个调整值，用于为用户提供“minMemory”参数调整的依据，保证配置可以生效。

29.11 配置访问通道协议

配置场景

服务端配置了 web 访问为 https 通道，如果客户端没有配置，默认使用 http 访问，客户端和服务端的配置不同，就会导致访问结果显示乱码。在客户端和服务端配置相同的“yarn.http.policy”参数，可以防止客户端访问结果显示乱码。

操作步骤

步骤 1 在 Manager 系统中，选择“集群 > 服务 > Yarn > 配置”，选择“全部配置”，在搜索框中输入参数名称“yarn.http.policy”。

- 安全模式下配置为“HTTPS_ONLY”。
- 普通模式下配置为“HTTP_ONLY”。

步骤 2 以客户端安装用户，登录安装客户端的节点。

步骤 3 执行以下命令，进入客户端安装路径。

cd /opt/client

步骤 4 执行以下命令编辑“yarn-site.xml”文件。

vi Yarn/config/yarn-site.xml

修改“yarn.http.policy”的参数值。

安全模式下，“yarn.http.policy”配置成“HTTPS_ONLY”。

普通模式下，“yarn.http.policy”配置成“HTTP_ONLY”。

步骤 5 执行**:wq** 命令保存。

步骤 6 重启客户端使配置生效。

----结束

29.12 检测内存使用情况

配置场景

针对所提交应用的内存使用无法预估的情况，可以通过修改服务端的配置项控制是否对内存使用进行检测。

若不检测内存使用，Container 会占用内存直到内存溢出；若检测内存使用，当内存使用超过配置的内存大小时，相应的 Container 会被 kill 掉。

配置描述

参考修改集群服务配置参数进入 Yarn 服务参数“全部配置”界面，在搜索框中输入参数名称。

表29-16 参数说明

参数	描述	默认值
yarn.nodemanager.vmem-check-enabled	是否进行虚拟内存检测的开关。如果任务使用的内存量超出分配值，则直接将任务强制终止。 • 设置为 true 时，进行虚拟内存检测； • 设置为 false 时，不进行虚拟内存检测。	MRS 3.x 之前的版本集群:false MRS 3.x 及后续版本集群:true
yarn.nodemanager.pmem-check-enabled	是否进行物理内存检测的开关。如果任务使用的内存量超出分配值，则直接将任务强制终止。 • 设置为 true 时，进行物理内存检测； • 设置为 false 时，不进行物理内存检测。	true

29.13 配置自定义调度器的 WebUI

配置场景

如果用户在 ResourceManager 中配置了自定义的调度器，可以通过以下配置项为其配置相应的 Web 展示页面及其他 Web 应用。

配置描述

参考修改集群服务配置参数进入 Yarn 服务参数“全部配置”界面，在搜索框中输入参数名称。

表29-17 配置自定义调度器的 WebUI

参数	描述	默认值
hadoop.http.rmwebapp.scheduler.page.classes	在 RM WebUI 中为自定义调度器加载相应的 web 页面。仅当“yarn.resourcemanager.scheduler.class”配置为自定义调度器时此配置项生效。	-
yarn.http.rmwebapp.external.classes	在 RM 的 Web 服务中加载用户自定义的 web 应用。	-

29.14 配置 YARN Restart 特性

配置场景

YARN Restart 特性包含两部分内容：ResourceManager Restart 和 NodeManager Restart。

- 当启用 ResourceManager Restart 时，升主后的 ResourceManager 就可以通过加载之前的主 ResourceManager 的状态信息，并通过接收所有 NodeManager 上 container 的状态信息，重构运行状态继续执行。这样应用程序通过定期执行检查点操作保存当前状态信息，就可以避免工作内容的丢失。
- 当启用 NodeManager Restart 时，NodeManager 在本地保存当前节点上运行的 container 信息，重启 NodeManager 服务后通过恢复此前保存的状态信息，就不会丢失在此节点上运行的 container 进度。

配置描述

参考修改集群服务配置参数进入 Yarn 服务参数“全部配置”界面，在搜索框中输入参数名称。

ResourceManager Restart 特性配置如下。

表29-18 ResourceManager Restart 参数配置

参数	描述	默认值
yarn.resourcemanager.recovery.enabled	设置是否让 ResourceManager 在启动后恢复状态。如果设置为 true，那 yarn.resourcemanager.store.class 也必须设置。	true
yarn.resourcemanager.store.class	指定用于保存应用程序和任务状态以及证书内容的 state-store 类。	MRS 3.x 之前的版本集群：org.apache.hadoop.yarn.server.resourcemanager.recovery.ZKRMStateStore MRS 3.x 及后续版本集群： org.apache.hadoop.yarn.server.resourcemanager.recovery.AsyncZKRMStateStore
yarn.resourcemanager.zk-state-store.parent-path	ZKRMStateStore 在 ZooKeeper 上的保存目录。	/rmstore
yarn.resourcemanager.work-preserving-recovery.enabled	启用 ResourceManager Work preserving 功能。该配置仅用于 YARN 特性验证。	true

参数	描述	默认值
yarn.resourcemanager.state-store.async.load	对已完成的 application 采用 ResourceManager 异步恢复方式。	MRS 3.x 之前的版本集群：false MRS 3.x 及后续版本集群：true
yarn.resourcemanager.zk-state-store.num-fetch-threads	启用异步恢复功能，增加工作线程的数量可以加快恢复 ZK 中保存的任务信息的速度，取值范围大于 0。	MRS 3.x 之前的版本集群：1 MRS 3.x 及后续版本集群：20

NodeManager Restart 特性配置如下。

表29-19 NodeManager Restart 参数配置

参数	描述	默认值
yarn.nodemanager.recovery.enabled	当 Nodemanager 重启时是否启用日志失败收集功能，是否恢复未完成的 Application。	true
yarn.nodemanager.recovery.dir	NodeManager 用于保存 container 状态的本地目录。适用于 MRS 3.x 及后续版本集群。	${SRV_HOME}/tmp/yarn-nm-recovery
yarn.nodemanager.recovery.supervised	NodeManager 是否在监控下运行。开启此特性后 NodeManager 在退出后不会清理 containers，NodeManager 会假设自己会立即重启和恢复 containers。	true

29.15 配置 AM 作业保留

本章节适用于 MRS 3.x 及后续版本集群。

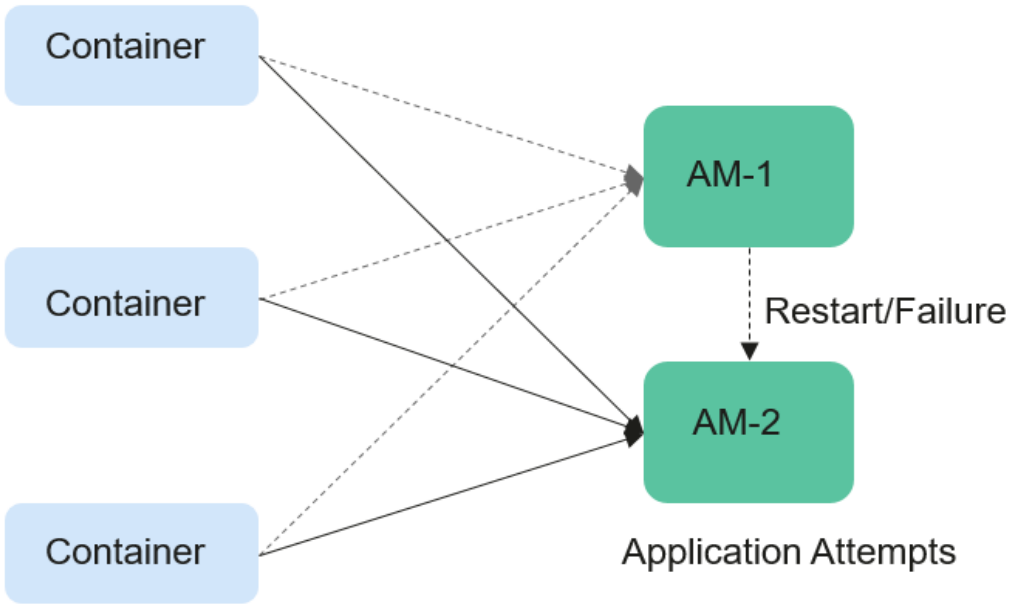
配置场景

在 YARN 中，ApplicationMaster(AM)与 Container 类似，都运行在 NodeManager(NM)上（本文中忽略未管理的 AM）。AM 可能由于多种原因崩溃、退出或关闭。如果 AM 停止运行，ResourceManager(RM)会关闭 ApplicationAttempt 中管理的所有 Container，其中包括当前在 NM 上运行的所有 Container。RM 会在另一计算节点上启动新的 ApplicationAttempt。

对于不同类型的应用，我们希望以不同方式处理 AM 重启的事件。MapReduce 类应用的目标是不丢失任务，但允许丢失当前运行的 Container。但是对于长周期的 YARN 服务而言，用户可能并不希望由于 AM 的故障而导致整个服务停止运行。

YARN 支持在新的 ApplicationAttempt 启动时，保留之前 Container 的状态，因此运行中的作业可以继续无故障的运行。

图29-1 AM 作业保留



配置描述

参考修改集群服务配置参数进入 Yarn 服务参数“全部配置”界面，在搜索框中输入参数名称。

根据表 29-20，对如下参数进行设置。

表29-20 AM 作业保留相关参数

参数	说明	默认值
yarn.app.mapreduce.am.work-preserve	是否开启 AM 作业保留特性。	false
yarn.app.mapreduce.am.umbilical.max.retries	AM 作业保留特性中，运行的容器尝试恢复的最大次数。	5
yarn.app.mapreduce.am.umbilical.retry.interval	AM 作业保留特性中，运行的容器尝试恢复的时间间隔。单位：毫秒。	10000
yarn.resourcemanager.	ApplicationMaster 的重试次数。增加重试次数	2

参数	说明	默认值
am.max-attempts	可以避免当资源不足时造成 AM 启动失败。 适用于所有 ApplicationMaster 的全局设置。每个 ApplicationMaster 都可以使用 API 设置一个单独的最大尝试次数，但这个次数不能大于全局的最大次数。如果大于了，那 ResourceManager 将会覆写这个单独的最大尝试次数。取值范围大于等于 1。	

29.16 配置本地化日志级别

本章节适用于 MRS 3.x 及后续版本集群。

配置场景

container 本地化默认的日志级别是 INFO。用户可以通过配置“yarn.nodemanager.container-localizer.java.opts”来改变日志级别。

配置描述

在 Manager 系统中，选择“集群 > *待操作集群的名称* > 服务 > Yarn > 配置”，选择“全部配置”，在 NodeManager 的配置文件“yarn-site.xml”中配置下面的参数来更改日志级别。

表29-21 参数描述

参数	描述	默认值
yarn.nodemanager.container-localizer.java.opts	附加的 jvm 参数是提供给本地化 container 进程使用的。	-Xmx256m -Djava.security.krb5.conf=${KRB5_CONFIG}

默认值-Xmx256m -Djava.security.krb5.conf=${KRB5_CONFIG}和默认日志级别是 INFO。为了更改 container 本地化的日志级别，添加下面的内容。

```
-Dhadoop.root.logger=<LOG_LEVEL>,localizationCLA
```

示例：

为了更改本地化日志级别为 DEBUG，参数值应该为

```
-Xmx256m -Dhadoop.root.logger=DEBUG,localizationCLA
```

说明

允许的日志级别是：FATAL，ERROR，WARN，INFO，DEBUG，TRACE 和 ALL。

29.17 配置运行任务的用户

本章节适用于 MRS 3.x 及后续版本集群。

配置场景

目前 YARN 支持启动 NodeManager 的用户运行所有用户提交的任务，也支持以提交任务的用户运行任务。

配置描述

在 Manager 系统中，选择“集群 > *待操作集群的名称* > 服务 > Yarn > 配置”，选择“全部配置”。在搜索框中输入参数名称。

表29-22 参数描述

参数	描述	默认值
yarn.nodemanager.linux-container-executor.user	运行任务的用户。	默认为空。 说明 默认为空，实际以提交任务的用户来运行任务。
yarn.nodemanager.container-executor.class	启动任务的 executor。	org.apache.hadoop.yarn.server.nodemanager.EnhancedLinuxContainerExecutor

说明

- “yarn.nodemanager.linux-container-executor.user”配置运行 container 的用户。默认空表示运行 container 的用户就是提交任务的用户。该参数仅在“yarn.nodemanager.container-executor.class”配置为“org.apache.hadoop.yarn.server.nodemanager.EnhancedLinuxContainerExecutor”时有效。
- 非安全模式下，当“yarn.nodemanager.linux-container-executor.user”设置为 omm 时，也需设置“yarn.nodemanager.linux-container-executor.nonsecure-mode.local-user”为 omm。
- 建议“yarn.nodemanager.linux-container-executor.user”和“yarn.nodemanager.container-executor.class”这两个参数都采用默认值，这样安全性更高。

29.18 Yarn 日志介绍

日志描述

Yarn 相关日志的默认存储路径如下：

- ResouceManager：“/var/log/Bigdata/yarn/rm”（运行日志），“/var/log/Bigdata/audit/yarn/rm”（审计日志）
- NodeManager：“/var/log/Bigdata/yarn/nm”（运行日志），“/var/log/Bigdata/audit/yarn/nm”（审计日志）

日志归档规则：Yarn 的日志启动了自动压缩归档功能，缺省情况下，当日志大小超过 50MB 的时候，会自动压缩，压缩后的日志文件名规则为：“<原有日志名>-<yyyy-mm-dd_hh-mm-ss>.[编号].log.zip”。最多保留最近的 100 个压缩文件，压缩文件保留个数可以在 Manager 界面中配置。

日志归档规则：

表29-23 Yarn 日志列表

日志类型	日志文件名	描述
运行日志	hadoop-<SSH_USER>-<process_name>-<hostname>.log	Yarn 组件日志，记录 Yarn 组件运行时候所产生的大部分日志。
	hadoop-<SSH_USER>-<process_name>-<hostname>.out	Yarn 运行环境信息日志。
	<process_name>-<SSH_USER>-<DATE>-<PID>-gc.log	垃圾回收日志。
	yarn-haCheck.log	ResourceManager 主备状态检测日志。
	yarn-service-check.log	Yarn 服务健康状态检查日志。
	yarn-start-stop.log	Yarn 服务启停操作日志。
	yarn-prestart.log	Yarn 服务启动前集群操作的记录日志。
	yarn-postinstall.log	Yarn 服务安装后启动前的工作日志。
	hadoop-commission.log	Yarn 入服日志。
	yarn-cleanup.log	Yarn 服务卸载时候的清理日志。
	yarn-refreshqueue.log	Yarn 刷新队列日志。
	upgradeDetail.log	升级日志记录。
	stderr/stdin/syslog	Yarn 服务上运行的应用所对应的 container 日志。
	yarn-application-check.log	Yarn 服务上运行的应用检查日志。

日志类型	日志文件名	描述
	yarn-appsummary.log	Yarn 服务上运行的应用的运行结果日志。
	yarn-switch-resourcemanager.log	Yarn 主备倒换运行日志。
	ranger-yarn-plugin-enable.log	Yarn 启用 Ranger 鉴权的日志
	yarn-nodemanager-period-check.log	Yarn nodemanager 的周期检查日志
	yarn-resourcemanager-period-check.log	Yarn resourcemanager 的周期检查日志
	hadoop.log	Hadoop 的客户端日志
	env.log	实例启停前的环境信息日志。
审计日志	yarn-audit-<process_name>.log ranger-plugin-audit.log	Yarn 操作审计日志。
	SecurityAuth.audit	Yarn 安全审计日志。

日志级别

Yarn 中提供了如表 29-24 所示的日志级别。其中日志级别优先级从高到低分别是 OFF、FATAL、ERROR、WARN、INFO、DEBUG。程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。

表29-24 日志级别

级别	描述
FATAL	FATAL 表示当前事件处理存在严重错误信息。
ERROR	ERROR 表示当前事件处理存在错误信息。
WARN	WARN 表示当前事件处理存在异常告警信息。
INFO	INFO 表示记录系统及各事件正常运行状态信息
DEBUG	DEBUG 表示记录系统及系统的调试信息

如果您需要修改日志级别，请执行如下操作：

步骤 1 参考修改集群服务配置参数，进入 Yarn 服务“全部配置”页面。

步骤 2 在左边菜单栏中选择所需修改的角色所对应的日志菜单。

步骤 3 选择所需修改的日志级别。

步骤 4 单击“保存配置”，在弹出窗口中单击“确定”使配置生效。

说明

配置完成后立即生效，不需要重启服务。

----结束

日志格式

Yarn 的日志格式如下所示：

表29-25 日志格式

日志类型	格式	示例
运行日志	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<产生该日志的线程名字>|<log 中的 message>|<日志事件的发生位置>	2014-09-26 14:18:59,109	INFO	main	Client environment:java.compiler=<NA>	org.apache.zookeeper.Environment.logEnv(Environment.java:100)
审计日志	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<产生该日志的线程名字>|<log 中的 message>|<日志事件的发生位置>	2014-09-26 14:24:43,605	INFO	main-EventThread	USER=omm OPERATION=refreshAdminAcls TARGET=AdminService RESULT=SUCCESS	org.apache.hadoop.yarn.server.resourcemanager.RMAuditLogger$LogLevel$6.printLog(RMAuditLogger.java:91)

29.19 Yarn 性能调优

29.19.1 抢占任务

操作场景

抢占任务可精简队列中的 job 运行并提高资源利用率，由 ResourceManager 的 capacity scheduler 实现，其简易流程如下：

1. 假设存在两个队列 A 和 B。其中队列 A 的 capacity 为 25%，队列 B 的 capacity 为 75%。

2. 初始状态下，任务 1 发送给队列 A，此任务需要 75%的集群资源。之后任务 2 发送到了队列 B，此任务需要 50%的集群资源。
3. 任务 1 将会使用队列 A 提供的 25%的集群资源，并从队列 B 获取的 50%的集群资源。队列 B 保留 25%的集群资源。
4. 启用抢占任务特性，则任务 1 使用的资源将会被抢占。队列 B 会从队列 A 中获取 25%的集群资源以满足任务 2 的执行。
5. 当任务 2 完成后，集群中存在足够的资源时，任务 1 将重新开始执行。

操作步骤

参数入口：

参考修改集群服务配置参数进入 Yarn 服务参数“全部配置”界面，在搜索框中输入参数名称。

表29-26 Preemption 配置

参数	描述	默认值
yarn.resourcemanager.scheduler.monitor.enable	根据“yarn.resourcemanager.scheduler.monitor.policies”中的策略，启用新的 scheduler 监控。设置为“true”表示启用监控，并根据 scheduler 的信息，启动抢占的功能。设置为“false”表示不启用。	false
yarn.resourcemanager.scheduler.monitor.policies	设置与 scheduler 配合的“SchedulingEditPolicy”的类的清单。	org.apache.hadoop.yarn.server.resourcemanager.monitor.capacity.ProportionalCapacityPreemptionPolicy
yarn.resourcemanager.monitor.capacity.preemption.observe_only	• 设置为“true”，则执行策略，但是不对集群资源进程抢占操作。 • 设置为“false”，则执行策略，且根据策略启用集群资源抢占的功能。	false
yarn.resourcemanager.monitor.capacity.preemption.monitoring_interval	根据策略监控的时间间隔，单位为毫秒。如果将该参数设置为更大的值，容量检测将不那么频繁地运行。	3000
yarn.resourcemanager.monitor.capacity.preemption.max_wait_before_kill	应用发送抢占需求到停止 container（释放资源）的时间间隔，单位为毫秒。取值范围大于等于 0。 默认情况下，若 ApplicationMaster15 秒内没有终止 container，ResourceManager 等待 15 秒后会强制终止。	15000

参数	描述	默认值
yarn.resourcemanager.monitor.capacity.preemption.total_preemption_per_round	在一个周期内能够抢占资源的最大的比例。可使用这个值来限制从集群回收容器的速度。计算出了期望的总抢占值之后，策略会伸缩回这个限制。	0.1
yarn.resourcemanager.monitor.capacity.preemption.max_ignored_over_capacity	集群中资源总量乘以此配置项的值加上某个队列（例如队列 A）原有的资源量为资源抢占盲区。当队列 A 中的任务实际使用的资源超过该抢占盲区时，超过部分的资源将会被抢占。取值范围：0~1。 说明 设置的值越小越有利于资源抢占。	0
yarn.resourcemanager.monitor.capacity.preemption.natural_termination_factor	设置抢占目标，Container 只会抢占所配置比例的资源。 示例，如果设置为 0.5，则在 5*“yarn.resourcemanager.monitor.capacity.preemption.max_wait_before_kill”的时间内，任务会回收所抢占资源的近 95%。即接连抢占 5 次，每次抢占待抢占资源的 0.5，呈几何收敛，每次的时间间隔为“yarn.resourcemanager.monitor.capacity.preemption.max_wait_before_kill”。取值范围：0~1。	1

29.19.2 任务优先级

操作场景

集群的资源竞争场景如下：

1. 提交两个低优先级的应用 Job 1 和 Job 2。
2. 正在运行中的 Job 1 和 Job 2 有部分 task 处于 running 状态，但由于集群或队列资源容量有限，仍有部分 task 未得到资源而处于 pending 状态。
3. 提交一个较高优先级的应用 Job 3，此时会出现如下资源分配情况：当 Job 1 和 Job 2 中 running 状态的 task 运行结束并释放资源后，Job 3 中处于 pending 状态的 task 将优先得到这部分新释放的资源。
4. Job 3 完成后，资源释放给 Job 1、Job 2 继续执行。

用户可以在 YARN 中配置任务的优先级。任务优先级是通过 ResourceManager 的调度器实现的。

操作步骤

设置参数“mapreduce.job.priority”，使用命令行接口或 API 接口设置任务优先级。

- 命令行接口。

  提交任务时，添加“**-Dmapreduce.job.priority=**<*priority*>”参数。

  <priority>可以设置为：

  - VERY_HIGH
  - HIGH
  - NORMAL
  - LOW
  - VERY_LOW
- API 接口。

  用户也可以使用 API 配置对象的优先级。

  设置优先级，可通过 **Configuration.set("mapreduce.job.priority", <priority>)**或**Job.setPriority(JobPriority priority)**设置。

29.19.3 节点配置调优

操作场景

合理配置大数据集群的调度器后，还可通过调节每个节点的可用内存、CPU 资源及本地磁盘的配置进行性能调优。

具体包括以下配置项：

- 可用内存
- CPU 虚拟核数
- 物理 CPU 使用百分比
- 内存和 CPU 资源的协调
- 本地磁盘

操作步骤

若您需要对参数配置进行调整，具体操作请参考修改集群服务配置参数。

- **可用内存**

  除了分配给操作系统、其他服务的内存外，剩余的资源应尽量分配给 YARN。通过如下配置参数进行调整。

  例如，如果一个 container 默认使用 512M，则内存使用的计算公式为：512M*container 数。

  默认情况下，Map 或 Reduce container 会使用 1 个虚拟 CPU 内核和 1024MB 内存，ApplicationMaster 使用 1536MB 内存。

参数	描述	默认值
yarn.nodemanager.resource.memory-mb	设置可分配给容器的物理内存数量。单位：MB，取值范围大于 0。 建议配置成节点物理内存总量的 75%~90%。若该节点有其他业务的常	MRS 3.x 及之后：16384 MRS 3.x 之

参数	描述	默认值
	驻进程，请降低此参数值给该进程预留足够运行资源。	前：8192

- **CPU 虚拟核数**

  建议将此配置设定在逻辑核数的 1.5～2 倍之间。如果上层计算应用对 CPU 的计算能力要求不高，可以配置为 2 倍的逻辑 CPU。

参数	描述	默认值
yarn.nodemanager.resource.cpu-vcores	表示该节点上 YARN 可使用的虚拟 CPU 个数，默认是 8。 目前推荐将该值设值为逻辑 CPU 核数的 1.5～2 倍之间。	8

- **物理 CPU 使用百分比**

  建议预留适量的 CPU 给操作系统和其他进程（数据库、HBase 等）外，剩余的 CPU 核都分配给 YARN。可以通过如下配置参数进行调整。

参数	描述	默认值
yarn.nodemanager.resource.percentage-physical-cpu-limit	表示该节点上 YARN 可使用的物理 CPU 百分比。默认是 90，即不进行 CPU 控制，YARN 可以使用节点全部 CPU。该参数只支持查看，可通过调整 YARN 的 RES_CPUSET_PERCENTAGE 参数来修改本参数值。注意，目前推荐将该值设为可供 YARN 集群使用的 CPU 百分数。 例如：当前节点除了 YARN 服务外的其他服务（如 HBase、HDFS、Hive 等）及系统进程使用 CPU 为 20%左右，则可以供 YARN 调度的 CPU 为 1-20%=80%，即配置此参数为 80。	90

- **本地磁盘**

  由于本地磁盘会提供给 MapReduce 写 job 执行的中间结果，数据量大。因此配置的原则是磁盘尽量多，且磁盘空间尽量大，单个达到百 GB 以上规模最好。简单的做法是配置和 data node 相同的磁盘，只在最下一级目录上不同即可。

说明

多个磁盘之间使用逗号隔开。

参数	描述	默认值
yarn.nodemanager.log-dirs	日志存放地址（可配置多个目录）。 容器日志的存储位置。默认值为%{@auto.detect.datapart.nm.logs}。如果有数据分区，基于该数据分区生成一个类似/srv/BigData/hadoop/data1/nm/containerlogs,/srv/BigData/hadoop/data2/nm/containerlogs 的路径清单。如果没有数据分区，生成默认路径/srv/BigData/yarn/data1/nm/containerlogs。除了使用表达式以外，还可以输入完整的路径清单，比如/srv/BigData/yarn/data1/nm/containerlogs 或/srv/BigData/yarn/data1/nm/containerlogs,/srv/BigData/yarn/data2/nm/containerlogs。这样数据就会存储在所有设置的目录中，一般会是在不同的设备中。为保证磁盘 IO 负载均衡，最好提供几个路径且每个路径都对应一个单独的磁盘。应用程序的本地化后的日志目录存在于相对路径/application_%{appid}中。单独容器的日志目录，即 container_{$contid}，是该路径下的子目录。每个容器目录都含容器生成的 stderr、stdin 及 syslog 文件。要新增目录，比如新增/srv/BigData/yarn/data2/nm/containerlogs 目录，应首先删除/srv/BigData/yarn/data2/nm/containerlogs 下的文件。之后，为/srv/BigData/yarn/data2/nm/containerlogs 赋予跟/srv/BigData/yarn/data1/nm/containerlogs 一样的读写权限，再将/srv/BigData/yarn/data1/nm/containerlogs 修改为/srv/BigData/yarn/data1/nm/containerlogs,/srv/BigData/yarn/data2/nm/containerlogs。可以新增目录，但不要修改或删除现有目录。否则，NodeManager 的数据将丢失，且服务将不可用。 【默认值】%{@auto.detect.datapart.nm.logs} 【注意】请谨慎修改该项。如果配置不当，将造成服务不可用。当角色级别的该配置项修改后，所有实例级别的	%{@auto.detect.datapart.nm.logs}

参数	描述	默认值
	该配置项都将被修改。如果实例级别的配置项修改后，其他实例的该配置项的值保持不变。	
yarn.nodemanager.local-dirs	本地化后的文件的存储位置。默认值为%{@auto.detect.datapart.nm.localdir}。如果有数据分区，基于该数据分区生成一个类似/srv/BigData/hadoop/data1/nm/localdir,/srv/BigData/hadoop/data2/nm/localdir 的路径清单。如果没有数据分区，生成默认路径/srv/BigData/yarn/data1/nm/localdir。除了使用表达式以外，还可以输入完整的路径清单，比如/srv/BigData/yarn/data1/nm/localdir 或/srv/BigData/yarn/data1/nm/localdir,/srv/BigData/yarn/data2/nm/localdir。这样数据就会存储在所有设置的目录中，一般会是在不同的设备中。为保证磁盘 IO 负载均衡，最好提供几个路径且每个路径都对应一个单独的磁盘。应用程序的本地化后的文件目录存在于相对路径/usercache/%{user}/appcache/application_%{appid}中。单独容器的工作目录，即 container_%{contid}，是该路径下的子目录。要新增目录，比如新增/srv/BigData/yarn/data2/nm/localdir 目录，应首先删除/srv/BigData/yarn/data2/nm/localdir 下的文件。之后，为/srv/BigData/hadoop/data2/nm/localdir 赋予跟/srv/BigData/hadoop/data1/nm/localdir 一样的读写权限，再将/srv/BigData/yarn/data1/nm/localdir 修改为/srv/BigData/yarn/data1/nm/localdir,/srv/BigData/yarn/data2/nm/localdir。可以新增目录，但不要修改或删除现有目录。否则，NodeManager 的数据将丢失，且服务将不可用。 【默认值】%{@auto.detect.datapart.nm.localdir} 【注意】请谨慎修改该项。如果配置不	%{@auto.detect.datapart.nm.localdir}

参数	描述	默认值
	当，将造成服务不可用。当角色级别的该配置项修改后，所有实例级别的该配置项都将被修改。如果实例级别的配置项修改后，其他实例的该配置项的值保持不变。	

29.20 Yarn 常见问题

29.20.1 任务完成后 Container 挂载的文件目录未清除

问题

使用了 CGroups 功能的场景下，任务完成后 Container 挂载的文件目录未清除。

回答

即使任务失败，Container 挂载的目录也应该被清除。

上述问题是由于删除动作超时导致的。完成某些任务所使用的时间已远超过删除时间。

为避免出现这种场景，您可以参考修改集群服务配置参数，进入 Yarn“全部配置”页面。在搜索框搜索“yarn.nodemanager.linux-container-executor.cgroups.delete-timeout-ms”配置项来修改删除时间的时长。参数值的单位为毫秒。

29.20.2 作业执行失败时会抛出 HDFS_DELEGATION_TOKEN 到期的异常

问题

安全模式下，为什么作业执行失败时会抛出 HDFS_DELEGATION_TOKEN 到期的异常？

回答

HDFS_DELEGATION_TOKEN 到期的异常是由于 token 没有更新或者超出了最大生命周期。

在 token 的最大生命周期内确保下面的参数值大于作业的运行时间。

“dfs.namenode.delegation.token.max-lifetime”=“604800000”（默认是一星期）

参考修改集群服务配置参数，进入 HDFS“全部配置”页面，在搜索框搜索该参数。

说明

建议在 token 的最大生命周期内参数值为多倍小时数。

29.20.3 重启 YARN，本地日志不被删除

问题

在以下两种情况下重启 YARN，本地日志不会被定时删除，将被永久保留。

- 在任务运行过程中，重启 YARN，本地日志不被删除。
- 在任务完成，日志归集失败后定时清除日志前，重启 YARN，本地日志不被删除。

回答

NodeManager 有个重启恢复机制（详情请参见 https://hadoop.apache.org/docs/r3.1.1/hadoop-yarn/hadoop-yarn-site/NodeManager.html#NodeManager_Restart），参考修改集群服务配置参数，进入 Yarn“全部配置”页面。需将 NodeManager 的“yarn.nodemanager.recovery.enabled”配置项为“true”后才生效，默认为“true”，这样在 YARN 重启的异常场景时会定时删除多余的本地日志，避免问题的出现。

29.20.4 为什么执行任务时 AppAttempts 重试次数超过 2 次还没有运行失败

问题

系统默认的 AppAttempts 运行失败的次数为 2，为什么在执行任务时，AppAttempts 重试次数超过 2 次还没有运行失败？

回答

在执行任务过程中，若 ContainerExitStatus 的返回值为 ABORTED、PREEMPTED、DISKS_FAILED、KILLED_BY_RESOURCEMANAGER 这四种状态之一时，系统不会将其计入 failed attempts 中，因此出现上面的问题，只有当真正失败尝试 2 次之后才会运行失败。

29.20.5 为什么在 ResourceManager 重启后，应用程序会移回原来的队列

问题

将应用程序从一个队列移到另一个队列时，为什么在 RM（ResourceManager）重启后，应用程序会被移回原来的队列？

回答

这是 RM 的使用限制，应用程序运行过程中移动到别的队列，此时 RM 重启，RM 并不会在状态存储中存储新队列的信息。

假设用户提交一个 MR 任务到叶子队列 test11 上。当任务运行时，删除叶子队列 test11，这时提交队列自动变为 lost_and_found 队列（找不到队列的任务会被放入 lost_and_found 队列中），任务暂停运行。要启动该任务，用户将任务移动到叶子队列 test21 上。在将任务移动到叶子队列 test21 后，任务继续运行，此时 RM 重启，重启后显示提交队列为 lost_and_found 队列，而不是 test21 队列。

发生上述情况的原因是，任务未完成时，RM 状态存储中存储的还是应用程序移动前的队列状态。唯一的解决办法就是等 RM 重启后，再次移动应用程序，将新的队列状态信息写入状态存储中。

29.20.6 为什么 YARN 资源池的所有节点都被加入黑名单，而 YARN 却没有释放黑名单，导致任务一直处于运行状态

问题

为什么 YARN 资源池的所有节点都被加入黑名单，而 YARN 却没有释放黑名单，导致任务一直处于运行状态？

回答

在 YARN 中，当一个 APP 的节点被 AM（ApplicationMaster）加入黑名单的数量达到一定比例（默认值为节点总数的 33%）时，该 AM 会自动释放黑名单，从而不会出现由于所有可用节点都被加入黑名单而任务无法获取节点资源的现象。

在资源池场景下，假设该集群上有 8 个节点，通过 NodeLabel 特性将集群划分为两个资源池，pool A 和 pool B，其中 pool B 包含两个节点。用户提交了一个任务 App1 到 pool B，由于 HDFS 空间不足，App1 运行失败，导致 pool B 的两个节点都被 App1 的 AM 加入了黑名单，根据上述原则，2 个节点小于 8 个节点的 33%，所以 YARN 不会释放黑名单，使得 App1 一直无法得到资源而保持运行状态，后续即使被加入黑名单的节点恢复，App1 也无法得到资源。

由于上述原则不适用于资源池场景，所以目前可通过调整客户端参数“yarn.resourcemanager.am-scheduling.node-blacklisting-disable-threshold”为：(nodes number of pool / total nodes)* 33%解决该问题。

29.20.7 ResourceManager 持续主备倒换

问题

RM（ResourceManager）在多个任务（比如 2000 个任务）正常并发运行时出现持续的主备倒换，导致 YARN 服务不可用。

回答

产生上述问题的原因是，full GC（GabageCollection）时间过长，超出了 RM 与 ZK（ZooKeeper）之间定期交互时长的阈值，导致 RM 与 ZK 失联，从而造成 RM 主备倒换。

在多任务情况下，RM 需要保存多个任务的鉴权信息，并通过心跳传递给各个 NM（NodeManager），即心跳 Response。心跳 Response 的生命周期短，默认值为 1s，一般可以在 JVM minor GC 时被回收，但在多任务的情况下，集群规模较大，比如 5000 节点，多个节点的心跳 Response 会占用大量内存，导致 JVM 在 minor GC 时无法完全回收，无法回收的内存持续累积，最终触发 JVM 的 full GC。JVM 的 GC 都是阻塞式的，即在 GC 过程中不执行任何作业，所以若 full GC 的时间过长，超出了 RM 与 ZK 之间定期交互时长的阈值，就会出现主备倒换。

登录 FusionInsight Manager，选择“集群 > 服务 > Yarn > 配置 > 全部配置”，在左侧选择“Yarn > 自定义”，在“yarn.yarn-site.customized.configs”中添加“yarn.resourcemanager.zk-timeout-ms”参数来增大 RM 与 ZK 之间定期交互时长的阈值（参数值的范围为小于等于 90000 毫秒），可以解决 RM 持续主备倒换的问题。

29.20.8 当一个 NodeManager 处于 unhealthy 的状态 10 分钟时，新应用程序失败

问题

当一个 NM（NodeManager）处于 unhealthy 的状态 10 分钟时，新应用程序失败。

回答

当 nodeSelectPolicy 为 SEQUENCE，且第一个连接到 RM 的 NM 不可用时，RM 会在“yarn.nm.liveness-monitor.expiry-interval-ms”属性中指定的周期内，一直尝试为同一个 NM 分配任务。

可以通过两种方式来避免上述问题：

- 使用其他的 nodeSelectPolicy，如 RANDOM。
- 参考修改集群服务配置参数，进入 Yarn“全部配置”页面。在搜索框搜索以下参数，通过“yarn-site.xml”文件更改以下属性：

  “yarn.resourcemanager.am-scheduling.node-blacklisting-enabled” = “true”；

  “yarn.resourcemanager.am-scheduling.node-blacklisting-disable-threshold” = “0.5”。

29.20.9 Superior 通过 REST 接口查看已结束或不存在的 applicationID，返回的页面提示 Error Occurred

问题

Superior 通过 REST 接口查看已结束或不存在的 applicationID，返回的页面提示 Error Occurred。

回答

用户提交查看 applicationID 的请求，访问 REST 接口“https://<SS_REST_SERVER>/ws/v1/sscheduler/applications/{application_id}”。

由于 Superior Scheduler 只存储正在运行的 applicationID，所以当查看的是已结束或不存在的 applicationID，服务器会响应给浏览器“404”的状态码。但是由于 chrome 浏览器访问该 REST 接口时，优先以“application/xml”的格式响应，该行为会导致服务器端处理出现异常，所以返回的页面会提示“Error Occurred”。而 IE 浏览器访问该 REST 接口时，优先以“application/json”的格式响应，服务器会正确响应给浏览器“404”的状态码。

29.20.10 Superior 调度模式下，单个 NodeManager 故障可能导致 MapReduce 任务失败

问题

在 Superior 调度模式下，如果出现单个 NodeManager 故障，可能会导致 Mapreduce 任务失败。

回答

正常情况下 ，当一个 application 的单个 task 的 attempt 连续在一个节点上失败 3 次，那么该 application 的 AppMaster 就会将该节点加入黑名单，之后 AppMaster 就会通知调度器不要继续调度 task 到该节点，从而避免任务失败。

但是默认情况下，当集群中有 33%的节点都被加入黑名单时，调度器会忽略黑名单节点。因此，该黑名单特性在小集群场景下容易失效。比如，集群只有 3 个节点，当 1 个节点出现故障，黑名单机制失效，不管 task 的 attempt 在同一个节点失败多少次，调度器仍然会将 task 继续调度到该节点，从而导致 application 因为 task 失败达到最大 attempt 次数（MapReduce 默认 4 次）而失败。

规避手段：

“yarn.resourcemanager.am-scheduling.node-blacklisting-disable-threshold”参数以百分比的形式配置忽略黑名单节点的阈值。建议根据集群规模，适当增大该参数的值，如 3 个节点的集群， 建议增大到 50%。

说明

Superior 调度器的框架设计是基于时间的异步调度，当 NodeManager 故障后，ResourceManager 无法快速的感知到 NodeManager 已经出了问题(默认 10mins)，因此在此期间，Superior 调度器仍然会向该节点调度 task，从而导致任务失败。

29.20.11 当应用程序从 lost_and_found 队列移动到其他队列时，应用程序不能继续执行

问题

当删除一个有部分应用程序正在运行的队列，这些应用程序会被移动到“lost_and_found”队列上。当这些应用程序移回运行正常的队列时，某些任务会被挂起，不能正常运行。

回答

如果应用程序没有设置标签表达式，那么该应用程序上新增的 container/resource 将使用其所在队列默认的标签表达式。如果队列没有默认的标签表达式，则将其标签表达设置为“default label”。

当应用程序（app1）提交到队列（Q1）上时，应用程序上新增的 container/resource 使用队列默认的标签表达式（“label1”）。若 app1 正在运行时 Q1 被删除，则 app1 被移动到“lost_and_found”队列上。由于“lost_and_found”队列没有标签表达式，其标签表达式设置为“default label”，此时 app1 上新增的 container/resource 也将其标签表达式设置为“default label”。当 app1 被移回正常运行的队列（例如，Q2）时，如果 Q2 支持调用 app1 中的所有标签表达式（包含“label1”和“default label”），则 app1 能正常运行直到结束；如果 Q2 仅支持调用 app1 中的部分标签表达式（例如，仅支持调用“default label”），那么 app1 在运行时，拥有“label1”标签表达式的部分任务的资源请求将无法获得资源，从而被挂起，不能正常运行。

因此当把应用程序从“lost_and_found”队列移动到其他运行正常的队列上时，需要保证目标队列能够调用该应用程序的所有标签表达式。

建议不要删除正在运行应用程序的队列。

29.20.12 如何限制存储在 ZKstore 中的应用程序诊断消息的大小

问题

如何限制存储在 ZKstore 中的应用程序诊断消息的大小？

回答

在某些情况下，已经观察到诊断消息可能无限增长。由于诊断消息存储在状态存储中，不建议允许诊断消息无限增长。因此，需要有一个属性参数用于设置诊断消息的最大大小。

若您需要设置“yarn.app.attempt.diagnostics.limit.kc”参数值，具体操作参考修改集群服务配置参数，进入 Yarn“全部配置”页面，在搜素框搜索以下参数。

表29-27 参数描述

参数	描述	默认值
yarn.app.attempt.dia	定义每次应用连接的诊断消息的数据大小，以千字	64

参数	描述	默认值
gnostics.limit.kc	节为单位（字符数*1024）。当使用 ZooKeeper 来存储应用程序的行为状态时，需要限制诊断消息的大小，以防止 YARN 拖垮 ZooKeeper。如果将“yarn.resourcemanager.state-store.max-completed-applications”设置为一个较大的数值，则需要减小该属性参数的值以限制存储的总数据大小。	

29.20.13 为什么将非 ViewFS 文件系统配置为 ViewFS 时 MapReduce 作业运行失败

问题

为什么将非 ViewFS 文件系统配置为 ViewFS 时 MR 作业运行失败？

回答

通过集群将非 ViewFS 文件系统配置为 ViewFS 时，ViewFS 中的文件夹的用户权限与默认 NameService 中的非 ViewFS 不同。因为目录权限不匹配，所以已提交的 MR 作业运行失败。

在集群中配置 ViewFS 的用户，需要检查并校验目录权限。在提交作业之前，应按照默认的 NameService 文件夹权限更改 ViewFS 文件夹权限。

下表列出了 ViewFS 中配置的目录的默认权限结构。如果配置的目录权限与下表不匹配，则必须相应地更改目录权限。

表29-28 ViewFS 中配置的目录的默认权限结构

参数	描述	默认值	默认值及其父目录的默认权限
yarn.nodemanager.remote-app-log-dir	在默认文件系统上（通常是 HDFS），指定 NM 应将日志聚合到哪个目录。	logs	777
yarn.nodemanager.remote-app-log-archive-dir	将日志归档的目录。	-	777
yarn.app.mapreduce.am.staging-dir	提交作业时使用的 staging 目录。	/tmp/hadoop-yarn/staging	777
mapreduce.jobhistory.intermediate-done-dir	MapReduce 作业记录历史文件的目录。	${yarn.app.mapreduce.am.staging-dir}/history/done_intermediate	777

参数	描述	默认值	默认值及其父目录的默认权限
mapreduce.jobhistory.done-dir	由 MR JobHistory Server 管理的历史文件的目录。	${yarn.app.mapreduce.am.staging-dir}/history/done	777

29.20.14 开启 Native Task 特性后，Reduce 任务在部分操作系统运行失败

问题

开启 Native Task 特性后，Reduce 任务在部分操作系统运行失败。

回答

运行包含 Reduce 的 Mapreduce 任务时，通过-Dmapreduce.job.map.output.collector.class=org.apache.hadoop.mapred.nativetask.NativeMapOutputCollectorDelegator 命令开启 Native Task 特性，任务在部分操作系统运行失败，日志中提示错误“version 'GLIBCXX_3.4.20' not found”。该问题原因是操作系统的 GLIBCXX 版本较低，导致该特性依赖的 libnativetask.so.1.0.0 库无法加载，进而导致任务失败。

规避手段：

设置配置项 mapreduce.job.map.output.collector.class 的值为 org.apache.hadoop.mapred.MapTask$MapOutputBuffer。

30 使用 ZooKeeper

30.1 从零开始使用 Zookeeper

Zookeeper 是一个开源的，高可靠的，分布式一致性协调服务。Zookeeper 设计目标是用来解决那些复杂，易出错的分布式系统难以保证数据一致性的。不必开发专门的协同应用，十分适合高可用服务保持数据一致性。

背景信息

在使用客户端前，除主管理节点以外的客户端，需要下载并更新客户端配置文件。

操作步骤

MRS 2.x 及以前版本集群执行以下操作：

步骤 1 下载客户端配置文件。

1. 登录 MRS 控制台，在左侧导航栏选择“集群列表 > 现有集群”，单击待操作集群的名称。该集群为“用户指南 > 配置集群 > 创建自定义集群”中创建的集群。
2. 选择“组件管理”。

步骤 2 登录 MRS Manager 的主管理节点。

1. 在 MRS 控制台，选择“集群列表 > 现有集群”，单击集群名称，在“节点管理”页签中查看节点名称，名称中包含“master1”的节点为 Master1 节点，名称中包含“master2”的节点为 Master2 节点。

   MRS Manager 的主备管理节点默认安装在集群 Master 节点上。在主备模式下，由于 Master1 和 Master2 之间会切换，Master1 节点不一定是 MRS Manager 的主管理节点，需要在 Master1 节点中执行命令，确认 MRS Manager 的主管理节点。命令请参考步骤 2.4。
2. 以 **root** 用户使用密码方式登录 Master1 节点。操作方法，请参见章节。
3. 切换至 **omm** 用户。

   sudo su - root

   su - omm

4. 执行以下命令确认 MRS Manager 的主管理节点。

   sh ${BIGDATA_HOME}/om-0.0.1/sbin/status-oms.sh

   回显信息中“HAActive”参数值为“active”的节点为主管理节点（如下例中“mgtomsdat-sh-3-01-1”为主管理节点），参数值为“standby”的节点为备管理节点（如下例中“mgtomsdat-sh-3-01-2”为备管理节点）。

```
Ha mode
double
NodeName            HostName                    HAVersion         StartTime
HAActive            HAAllResOK          HARunPhase
192-168-0-30         mgtomsdat-sh-3-01-1          V100R001C01       2014-11-18
23:43:02     active             normal              Actived
192-168-0-24         mgtomsdat-sh-3-01-2          V100R001C01       2014-11-21
07:14:02     standby            normal              Deactived
```

5. 使用 **root** 用户登录 MRS Manager 的主管理节点，例如“192-168-0-30”节点，并执行以下命令切换到 **omm** 用户。

   sudo su - omm

步骤 3 执行以下命令切换到客户端安装目录。例如“/opt/client”。

cd /opt/client

步骤 4 执行以下命令，更新主管理节点的客户端配置。

sh refreshConfig.sh /opt/client *客户端配置文件压缩包完整路径*

例如，执行命令：

sh refreshConfig.sh /opt/client/tmp/MRS-client/MRS_Services_Client.tar

界面显示以下信息表示配置刷新更新成功：

```
ReFresh components client config is complete.
Succeed to refresh components client config.
```

说明

步骤步骤 1~步骤 4 的操作也可以参考页面的方法二操作。

步骤 5 在 Master 节点使用客户端。

1. 在已更新客户端的主管理节点，例如“192-168-0-30”节点，执行以下命令切换到客户端目录。

   cd /opt/client

2. 执行以下命令配置环境变量。

   source bigdata_env

3. 如果当前集群已启用 Kerberos 认证，执行以下命令认证当前用户。如果当前集群未启用 Kerberos 认证，则无需执行此命令。

   kinit MRS 集群用户

   例如，kinit zookeeperuser。

4. 直接执行 Zookeeper 组件的客户端命令。

   zkCli.sh -server <zookeeper 安装节点 ip>:<port>

例如：***zkCli.sh -server node-master1DGhZ:2181***

步骤 6 运行 Zookeeper 客户端命令。

1. 创建 ZNode。

```
create /test
```

2. 查看 ZNode 信息。

```
ls /
```

3. 向 ZNode 中写入数据。

```
set /test "zookeeper test"
```

4. 查看写入 ZNode 中的数据。

```
get /test
```

5. 删除创建的 ZNode。

```
delete /test
```

----结束

MRS 3.x 及以后版本集群执行以下操作：

步骤 1 下载客户端配置文件。

1. 登录 FusionInsight Manager 页面，具体请参见访问 FusionInsight Manager（MRS 3.x 及之后版本）。
2. 选择“集群 > *待操作集群的名称* > 概览 > 更多 > 下载客户端”。
3. 下载集群客户端。

   “选择客户端类型”选择“仅配置文件”，选择平台类型，单击“确定”开始生成客户端配置文件，文件生成后默认保存在主管理节点“/tmp/FusionInsight-Client/”。

步骤 2 登录 Manager 的主管理节点。

1. 以 **root** 用户登录任意部署 Manager 的节点。
2. 执行以下命令确认主备管理节点。

   sh ${BIGDATA_HOME}/om-server/om/sbin/status-oms.sh

   界面打印信息中“HAActive”参数值为“active”的节点为主管理节点（如下例中“node-master1”为主管理节点），参数值为“standby”的节点为备管理节点（如下例中“node-master2”为备管理节点）。

```
HAMode
double
NodeName            HostName        HAVersion         StartTime
HAActive            HAAllResOK          HARunPhase
192-168-0-30         node-master1    V100R001C01       2020-05-01 23:43:02
active              normal              Actived
192-168-0-24         node-master2    V100R001C01       2020-05-01 07:14:02
standby             normal              Deactived
```

3. 以 **root** 用户登录主管理节点，并执行以下命令切换到 **omm** 用户。

   sudo su - omm

步骤 3 执行以下命令切换到客户端安装目录。例如“/opt/client”。

cd /opt/client

步骤 4 执行以下命令，更新主管理节点的客户端配置。

sh refreshConfig.sh /opt/client *客户端配置文件压缩包完整路径*

例如，执行命令：

sh refreshConfig.sh /opt/client /tmp/FusionInsight-Client/FusionInsight_Cluster_1_Services_Client.tar

界面显示以下信息表示配置刷新更新成功：

```
ReFresh components client config is complete.
Succeed to refresh components client config.
```

步骤 5 在 Master 节点使用客户端。

1. 在已更新客户端的主管理节点，例如“192-168-0-30”节点，执行以下命令切换到客户端目录。

   cd /opt/client

2. 执行以下命令配置环境变量。

   source bigdata_env

3. 如果当前集群已启用 Kerberos 认证，执行以下命令认证当前用户，具体请参见配置拥有对应权限的角色，参考为用户绑定对应角色。如果当前集群未启用 Kerberos 认证，则无需执行此命令。

   kinit MRS 集群用户

   例如，**kinit zookeeperuser**。

4. 直接执行 Zookeeper 组件的客户端命令。

   zkCli.sh -server <zookeeper 安装节点 ip>:<port>

   例如：**zkCli.sh -server node-master1DGhZ:2181**

步骤 6 运行 Zookeeper 客户端命令。

1. 创建 ZNode。

   ```
   create /test
   ```

2. 查看 ZNode 信息。

   ```
   ls /
   ```

3. 向 ZNode 中写入数据。

   ```
   set /test "zookeeper test"
   ```

4. 查看写入 ZNode 中的数据。

   ```
   get /test
   ```

5. 删除创建的 ZNode。

   ```
   delete /test
   ```

----结束

30.2 ZooKeeper 常用参数

参数入口：

请参考修改集群服务配置参数，进入 ZooKeeper“全部配置”页面。在搜索框中输入参数名称。

表30-1 参数说明

配置参数	说明	默认值
skipACL	是否跳过 ZooKeeper 节点的权限检查。	no
maxClientCnxns	ZooKeeper 的最大连接数，在连接数多的情况下，建议增加。	2000
LOG_LEVEL	日志级别，在调试的时候，可以改为 DEBUG。	INFO
acl.compare.shortName	当 Znode 的 ACL 权限认证类型为 SASL 时，是否仅使用 principal 的用户名部分进行 ACL 权限认证。	true
synclimit	Follower 与 leader 进行同步的时间间隔（单位为 tick）。如果在指定的时间内 leader 没响应，连接将不能被建立。	15
tickTime	一次 tick 的时间（毫秒），它是 ZooKeeper 使用的基本时间单位，心跳、超时的时间都由它来规定。	4000

说明

ZooKeeper 内部时间由参数 ticktime 和参数 synclimit 控制，如需调大 ZooKeeper 内部超时时间，需要调大客户端连接 ZooKeeper 的超时时间。

30.3 使用 ZooKeeper 客户端

操作场景

该任务指导用户在运维场景或业务场景中使用 ZooKeeper 客户端。

前提条件

已安装客户端。例如安装目录为“/opt/client”，以下操作的客户端目录只是举例，请根据实际安装目录修改。

操作步骤

步骤 1 以客户端安装用户，登录安装客户端的节点。

步骤 2 执行以下命令，切换到客户端安装目录。

cd /opt/client

步骤 3 执行以下命令配置环境变量。

source bigdata_env

步骤 4 执行以下命令进行用户认证。(普通模式跳过此步骤)

kinit *组件业务用户*

步骤 5 执行以下命令登录客户端工具。

zkCli.sh -server *ZooKeeper 角色实例所在节点业务 IP:* ***clientPort***

----结束

30.4 ZooKeeper 权限设置指南

操作场景

该操作指导用户对 ZooKeeper 的 znode 设置权限。

ZooKeeper 通过访问控制列表（ACL）来对 znode 进行访问控制。ZooKeeper 客户端为 znode 指定 ACL，ZooKeeper 服务器根据 ACL 列表判定某个请求 znode 的客户端是否有对应操作的权限。ACL 设置涉及如下四个方面。

- 查看 ZooKeeper 中 znode 的 ACL。
- 增加 ZooKeeper 中 znode 的 ACL。
- 修改 ZooKeeper 中 znode 的 ACL。
- 删除 ZooKeeper 中 znode 的 ACL。

  ZooKeeper 的 ACL 权限说明：

  ZooKeeper 目前支持 create，delete，read，write，admin 五种权限，且 ZooKeeper 对权限的控制是 znode 级别的，而且不继承，即对父 znode 设置权限，其子 znode 不继承父 znode 的权限。ZooKeeper 中 znode 的默认权限为 **world:anyone:cdrwa**，即任何用户都有所有权限。

说明

ACL 有三部分：

第一部分是认证类型，如 world 指所有认证类型，sasl 是 kerberos 认证类型；

第二部分是帐号，如 anyone 指的是任何人；

第三部分是权限，如 cdrwa 指的是拥有所有权限。

特别的，由于普通模式启动客户端不需要认证，sasl 认证类型的 ACL 在普通模式下将不能使用。本文所有涉及 sasl 方式的鉴权操作均是在安全集群中进行。

表30-2 Zookeeper 的五种 ACL

权限说明	权限简称	权限详情
创建权限	create(c)	可以在当前 znode 下创建子 znode
删除权限	delete(d)	删除当前的 znode
读权限	read(r)	获取当前 znode 的数据，可以列出当前 znode 所有的子 znodes
写权限	write(w)	向当前 znode 写数据，写入子 znode
管理权限	admin(a)	设置当前 znode 的权限

对系统的影响

须知

修改 ZooKeeper 的 ACL 是高危操作。修改 ZooKeeper 中 znode 的权限，可能会导致其他用户无权限访问该 znode，导致系统功能异常。另外在 3.5.6 及以后版本，用户对于 getAcl 操作需要有读权限。

前提条件

- 已安装 ZooKeeper 客户端。例如安装目录为“/opt/client”。
- 已获取系统管理员用户和密码。

操作步骤

启动 ZooKeeper 客户端

步骤 1 以 **root** 用户登录安装了 ZooKeeper 客户端的服务器。

步骤 2 进入客户端安装目录。

cd /opt/client

步骤 3 执行以下命令配置环境变量。

source bigdata_env

步骤 4 执行以下命令认证用户身份，并输入用户密码（任意有权限的用户，这里以 admin 为例，普通模式不涉及）。

kinit admin

步骤 5 在 ZooKeeper 客户端执行以下命令，进入 ZooKeeper 命令行。

sh zkCli.sh -server *ZooKeeper 任意实例所在节点业务平面 IP:clientPort*

默认的“clientPort”为“2181”

例如：**sh zkCli.sh -server 192.168.0.151:2181**

步骤 6 登录 ZooKeeper 客户端后，使用 **ls** 命令，可以查看 ZooKeeper 中的 znode 列表。例如，可以查看根目录 znode 列表。

ls /

```
[zk: 192.168.0.151:2181(CONNECTED) 1] ls /
[hadoop-flag, hadoop-ha, test, test2, test3, test4, test5, test6, zookeeper]
```

查看 ZooKeeper znode ACL 信息

步骤 7 启动 ZooKeeper 客户端。

步骤 8 使用 **getAcl** 命令，可以查看 znode。如下命令，可以查看到之前创建的名为 test 的 znode 的 ACL 权限。

getAcl */znode 名称*

```
[zk: 192.168.0.151:2181(CONNECTED) 2] getAcl /test
'world,'anyone
: cdrwa
```

增加 ZooKeeper znode ACL 信息

步骤 9 启动 ZooKeeper 客户端。

步骤 10 查看旧的 ACL 信息，查看当前帐号是否有权限修改该 znode 的 ACL 信息的权限（a 权限），如果没有权限，需要 kinit 登录有权限的用户，并重新启动 ZooKeeper 客户端。

getAcl */znode 名称*

```
 [zk: 192.168.0.151:2181(CONNECTED) 3] getAcl /test
'world,'anyone
: cdrwa
```

步骤 11 使用 **setAcl** 命令增加权限。设置新权限命令如下：

setAcl /test world:anyone:cdrwa,sasl:*用户名@<系统域名>:权限值*

例如对 test 的 znode，需要增加 admin 用户的权限：

setAcl /test world:anyone:cdrwa,sasl:admin@HADOOP.COM:cdrwa

说明

增加权限时，需要保留已有权限。新增加权限和旧的权限用英文逗号隔开，新增加权限有三个部分：

第一部分是认证类型，如 sasl 指使用 kerberos 认证；

第二部分是帐号，如 admin@HADOOP.COM 指的是 admin 用户；

第三部分是权限，如 cdrwa 指的是拥有所有权限。

步骤 12 setAcl 后，可以使用 **getAcl** 命令查看增加权限是否成功：

getAcl */znode 名称*

```
[zk: 192.168.0.151:2181(CONNECTED) 4] getAcl /test
'world,'anyone
: cdrwa
'sasl,'admin@<系统域名>
: cdrwa
```

修改 ZooKeeper znode ACL 信息

步骤 13 启动 ZooKeeper 客户端。

步骤 14 查看旧的 ACL 信息，查看当前帐号是否有权限修改该 znode 的 ACL 信息的权限（a 权限），如果没有权限，需要 kinit 登录有权限的用户，并重新启动 ZooKeeper 客户端。

getAcl */znode 名称*

```
[zk: 192.168.0.151:2181(CONNECTED) 5] getAcl /test
'world,'anyone
: cdrwa
'sasl,'admin@<系统域名>
: cdrwa
```

步骤 15 使用 **setAcl** 命令修改权限。设置新权限命令如下：

setAcl /test sasl:*用户名@<系统域名>:权限值*

例如仅保留 admin 用户的所有权限，删除 anyone 用户的 rw 权限。

setAcl /test sasl:admin@HADOOP.COM:cdrwa

步骤 16 setAcl 后，可以使用 **getAcl** 命令查看修改权限是否成功：

getAcl */znode 名称*

```
[zk: 192.168.0.151:2181(CONNECTED) 6] getAcl /test
'sasl,'admin@<系统域名>
: cdrwa
```

删除 ZooKeeper znode ACL 信息

步骤 17 启动 ZooKeeper 客户端。

步骤 18 查看旧的 ACL 信息，查看当前帐号是否有权限修改该 znode 的 ACL 信息的权限（a 权限），如果没有权限，需要 kinit 登录有权限的用户，并重新启动 ZooKeeper 客户端。

getAcl */znode 名称*

```
[zk: 192.168.0.151:2181(CONNECTED) 5] getAcl /test
'world,'anyone
: rw
```

```
'sasl,'admin@<系统域名>
: cdrwa
```

步骤 19 使用 **setAcl** 命令增加权限。设置新权限命令如下：

setAcl /test sasl:*用户名@<系统域名>:权限值*

例如，仅保留 admin 用户是所有权限，取消 anyone 用户的 rw 权限。

setAcl /test sasl:admin@HADOOP.COM:cdrwa

步骤 20 setAcl 后，可以使用 **getAcl** 命令查看修改权限是否成功

getAcl */znode 名称*

```
[zk: 192.168.0.151:2181(CONNECTED) 6] getAcl /test
'sasl,'admin@<系统域名>
: cdrwa
```

----结束

30.5 ZooKeeper 日志介绍

日志描述

日志存储路径：“/var/log/Bigdata/zookeeper/quorumpeer”（运行日志），“/var/log/Bigdata/audit/zookeeper/quorumpeer”（审计日志）

日志归档规则：ZooKeeper 的日志启动了自动压缩归档功能，缺省情况下，当日志大小超过 30MB 的时候，会自动压缩。最多保留 20 个压缩文件，压缩文件保留个数可以在 Manager 界面中配置。

表30-3 ZooKeeper 日志列表

日志类型	日志文件名	描述
运行日志	zookeeper-<SSH_USER>-<process_name>-<hostname>.log	ZooKeeper 系统日志，记录 ZooKeeper 系统运行时候所产生的大部分日志。
	check-serviceDetail.log	ZooKeeper 服务启动是否成功的检查日志。
	zookeeper-<SSH_USER>-<DATA>-<PID>-gc.log	ZooKeeper 垃圾回收日志。
	instanceHealthDetail.log	ZooKeeper 实例健康状态检查日志
	zookeeper-omm-server-<hostname>.out	ZooKeeper 运行异常退出日志。
	zk-err-<zkpid>.log	ZooKeeper 致命错误日志。

日志类型	日志文件名	描述
	java_pid<zkpid>.hprof	ZooKeeper 内存溢出日志。
	funcDetail.log	ZooKeeper 实例启动日志。
	zookeeper-period-check.log	ZooKeeper 实例健康检查日志。
	zookeeper-period-check-java.log	ZooKeeper 配额监控周期检查日志。
审计日志	zk-audit-quorumpeer.log	ZooKeeper 操作审计日志。

日志级别

ZooKeeper 中提供了如表 30-4 所示的日志级别。日志级别优先级从高到低分别是 FATAL、ERROR、WARN、INFO、DEBUG。程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。

表30-4 日志级别

级别	描述
FATAL	FATAL 表示当前事件处理出现严重错误信息，可能导致系统崩溃。
ERROR	ERROR 表示当前事件处理出现错误信息，系统运行出错。
WARN	WARN 表示当前事件处理存在异常信息，但认为是正常范围，不会导致系统出错。
INFO	INFO 表示系统及各事件正常运行状态信息。
DEBUG	DEBUG 表示系统及系统的调试信息。

如果您需要修改日志级别，请执行如下操作：

步骤 1 参考修改集群服务配置参数章节，进入 ZooKeeper 服务“全部配置”页面。

步骤 2 左边菜单栏中选择所需修改的角色所对应的日志菜单。

步骤 3 选择所需修改的日志级别。

步骤 4 单击“保存”，在弹出窗口中单击“确定”使配置生效。

说明

配置完成后立即生效，不需要重启服务。

----结束

日志格式

ZooKeeper 的日志格式如下所示：

表30-5 日志格式

日志类型	组件	格式	示例
运行日志	zookeeper quorumpeer	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<产生该日志的线程名字>|<log 中的 message>|<日志事件的发生位置>	2020-01-20 16:33:43,816 | INFO | main | Defaulting to majority quorums | org.apache.zookeeper.server.quorum.QuorumPeerConfig.parseProperties(QuorumPeerConfig.java:335)
审计日志	zookeeper quorumpeer	<yyyy-MM-dd HH:mm:ss,SSS>|<Log Level>|<产生该日志的线程名字>|<log 中的 message>|<日志事件的发生位置>	2020-01-20 16:33:54,313 | INFO | CommitProcessor:13 | session=0xd4b0679daea0000 ip=10.177.112.145 operation=create znode target=ZooKeeperServer znode=/zk-write-test-2 result=success | org.apache.zookeeper.ZKAuditLogger$LogLevel$5.printLog(ZKAuditLogger.java:70)

30.6 ZooKeeper 常见问题

30.6.1 创建大量 znode 后，ZooKeeper Sever 启动失败

问题

创建大量 znode 后，ZooKeeper 集群处于故障状态不能自动恢复，尝试重启失败，ZooKeeper server 日志显示如下内容：

follower：

```
2016-06-23 08:00:18,763 | WARN  |
QuorumPeer[myid=26](plain=/10.16.9.138:2181)(secure=disabled) | Exception when
following the leader |
org.apache.zookeeper.server.quorum.Follower.followLeader(Follower.java:93)
java.net.SocketTimeoutException: Read timed out
    at java.net.SocketInputStream.socketRead0(Native Method)
    at java.net.SocketInputStream.socketRead(SocketInputStream.java:116)
    at java.net.SocketInputStream.read(SocketInputStream.java:170)
    at java.net.SocketInputStream.read(SocketInputStream.java:141)
    at java.io.BufferedInputStream.fill(BufferedInputStream.java:246)
    at java.io.BufferedInputStream.read(BufferedInputStream.java:265)
    at java.io.DataInputStream.readInt(DataInputStream.java:387)
    at org.apache.jute.BinaryInputArchive.readInt(BinaryInputArchive.java:63)
    at
org.apache.zookeeper.server.quorum.QuorumPacket.deserialize(QuorumPacket.java:83)
    at org.apache.jute.BinaryInputArchive.readRecord(BinaryInputArchive.java:99)
    at org.apache.zookeeper.server.quorum.Learner.readPacket(Learner.java:156)
    at
org.apache.zookeeper.server.quorum.Learner.registerWithLeader(Learner.java:276)
    at org.apache.zookeeper.server.quorum.Follower.followLeader(Follower.java:75)
    at org.apache.zookeeper.server.quorum.QuorumPeer.run(QuorumPeer.java:1094)
2016-06-23 08:00:18,764 | INFO  |
QuorumPeer[myid=26](plain=/10.16.9.138:2181)(secure=disabled) | shutdown called |
org.apache.zookeeper.server.quorum.Follower.shutdown(Follower.java:198)
java.lang.Exception: shutdown Follower
    at org.apache.zookeeper.server.quorum.Follower.shutdown(Follower.java:198)
    at
org.apache.zookeeper.server.quorum.QuorumPeer.stopFollower(QuorumPeer.java:1141)
    at org.apache.zookeeper.server.quorum.QuorumPeer.run(QuorumPeer.java:1098)
```

leader：

```
2016-06-23 07:30:57,481 | WARN  |
QuorumPeer[myid=25](plain=/10.16.9.136:2181)(secure=disabled) | Unexpected
exception | org.apache.zookeeper.server.quorum.QuorumPeer.run(QuorumPeer.java:1108)
java.lang.InterruptedException: Timeout while waiting for epoch to be acked by
quorum
    at org.apache.zookeeper.server.quorum.Leader.waitForEpochAck(Leader.java:1221)
    at org.apache.zookeeper.server.quorum.Leader.lead(Leader.java:487)
    at org.apache.zookeeper.server.quorum.QuorumPeer.run(QuorumPeer.java:1105)
2016-06-23 07:30:57,482 | INFO  |
QuorumPeer[myid=25](plain=/10.16.9.136:2181)(secure=disabled) | Shutdown called |
org.apache.zookeeper.server.quorum.Leader.shutdown(Leader.java:623)
java.lang.Exception: shutdown Leader! reason: Forcing shutdown
    at org.apache.zookeeper.server.quorum.Leader.shutdown(Leader.java:623)
    at  org.apache.zookeeper.server.quorum.QuorumPeer.stopLeader(QuorumPeer.java:1149)
    at org.apache.zookeeper.server.quorum.QuorumPeer.run(QuorumPeer.java:1110)
```

回答

创建大量节点后，follower 与 leader 同步时数据量大，在集群数据同步限定时间内不能完成同步过程，导致超时，各个 ZooKeeper server 启动失败。

参考修改集群服务配置参数章节，进入 ZooKeeper 服务“全部配置”页面。不断尝试调大 ZooKeeper 配置文件“zoo.cfg”中的“syncLimit”和“initLimit”两参数值，直到 ZooKeeperServer 正常。

表30-6 参数说明

参数	描述	默认值
syncLimit	follower 与 leader 进行同步的时间间隔（时长为 ticket 时长的倍数）。如果在该时间范围内 leader 没响应，连接将不能被建立。	15
initLimit	follower 连接到 leader 并与 leader 同步的时间（时长为 ticket 时长的倍数）。	15

如果将参数“initLimit”和“syncLimit”的参数值均配置为“300”之后，ZooKeeper server 仍然无法恢复，则需确认没有其他应用程序正在 kill ZooKeeper。例如，参数值为“300”，ticket 时长为 2000 毫秒，即同步限定时间为 300*2000ms=600s。

可能存在以下场景，在 ZooKeeper 中创建的数据过大，需要大量时间与 leader 同步，并保存到硬盘。在这个过程中，如果 ZooKeeper 需要运行很长时间，则需确保没有其他监控应用程序 kill ZooKeeper 而判断其服务停止。

30.6.2 为什么 ZooKeeper Server 出现 java.io.IOException: Len 的错误日志

问题

在父目录中创建大量的 znode 之后，当 ZooKeeper 客户端尝试在单个请求中获取该父目录中的所有子节点时，将返回失败。

客户端日志，如下所示：

```
2017-07-11 13:17:19,610 [myid:] - WARN  [New I/O worker
#3:ClientCnxnSocketNetty$ZKClientHandler@468] - Exception caught: [id: 0xb66cbb85,
/10.18.97.97:49192 :> 10.18.97.97/10.18.97.97:2181] EXCEPTION:
java.nio.channels.ClosedChannelException
java.nio.channels.ClosedChannelException
at org.jboss.netty.handler.ssl.SslHandler$6.run(SslHandler.java:1580)
at
org.jboss.netty.channel.socket.ChannelRunnableWrapper.run(ChannelRunnableWrapper.ja
va:40)
at
org.jboss.netty.channel.socket.nio.AbstractNioWorker.executeInIoThread(AbstractNioW
orker.java:71)
at  org.jboss.netty.channel.socket.nio.NioWorker.executeInIoThread(NioWorker.java:36)
at
org.jboss.netty.channel.socket.nio.AbstractNioWorker.executeInIoThread(AbstractNioW
orker.java:57)
at  org.jboss.netty.channel.socket.nio.NioWorker.executeInIoThread(NioWorker.java:36)
at
```

```
org.jboss.netty.channel.socket.nio.AbstractNioChannelSink.execute(AbstractNioChanne
lSink.java:34)
at org.jboss.netty.handler.ssl.SslHandler.channelClosed(SslHandler.java:1566)
at org.jboss.netty.channel.Channels.fireChannelClosed(Channels.java:468
at
org.jboss.netty.channel.socket.nio.AbstractNioWorker.close(AbstractNioWorker.java:3
76)
at org.jboss.netty.channel.socket.nio.NioWorker.read(NioWorker.java:93)
at
org.jboss.netty.channel.socket.nio.AbstractNioWorker.process(AbstractNioWorker.java
:109)
at
org.jboss.netty.channel.socket.nio.AbstractNioSelector.run(AbstractNioSelector.java
:312)
at
org.jboss.netty.channel.socket.nio.AbstractNioWorker.run(AbstractNioWorker.java:90)
at org.jboss.netty.channel.socket.nio.NioWorker.run(NioWorker.java:178)
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
at java.lang.Thread.run(Thread.java:745)
```

Leader 节点的日志，如下所示：

```
2017-07-11 13:17:33,043 [myid:1] - WARN  [New I/O worker #7:NettyServerCnxn@445] -
Closing connection to /10.18.101.110:39856
java.io.IOException: Len error 45
at
org.apache.zookeeper.server.NettyServerCnxn.receiveMessage(NettyServerCnxn.java:438)
at
org.apache.zookeeper.server.NettyServerCnxnFactory$CnxnChannelHandler.processMessag
e(NettyServerCnxnFactory.java:267)
at
org.apache.zookeeper.server.NettyServerCnxnFactory$CnxnChannelHandler.messageReceiv
ed(NettyServerCnxnFactory.java:187)
at
org.jboss.netty.channel.SimpleChannelHandler.handleUpstream(SimpleChannelHandler.ja
va:88)
at
org.jboss.netty.channel.DefaultChannelPipeline.sendUpstream(DefaultChannelPipeline.
java:564)
at
org.jboss.netty.channel.DefaultChannelPipeline.sendUpstream(DefaultChannelPipeline.
java:559)
at org.jboss.netty.channel.Channels.fireMessageReceived(Channels.java:268)
at org.jboss.netty.channel.Channels.fireMessageReceived(Channels.java:255)
at org.jboss.netty.channel.socket.nio.NioWorker.read(NioWorker.java:88)
at
org.jboss.netty.channel.socket.nio.AbstractNioWorker.process(AbstractNioWorker.java
:109)
at
org.jboss.netty.channel.socket.nio.AbstractNioSelector.run(AbstractNioSelector.java
:312)
at
org.jboss.netty.channel.socket.nio.AbstractNioWorker.run(AbstractNioWorker.java:90)
at org.jboss.netty.channel.socket.nio.NioWorker.run(NioWorker.java:178)
at org.jboss.netty.util.ThreadRenamingRunnable.run(ThreadRenamingRunnable.java:108)
```

```
at
org.jboss.netty.util.internal.DeadLockProofWorker$1.run(DeadLockProofWorker.java:42)
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
at java.lang.Thread.run(Thread.java:745)
```

回答

在单个父目录中创建大量的 znode 后，当客户端尝试在单个请求中获取所有子节点时，服务端将无法返回，因为结果将超出可存储在 znode 上的数据的最大长度。

为了避免这个问题，应该根据客户端应用的实际情况将“jute.maxbuffer”参数配置为一个更高的值。

“jute.maxbuffer”只能设置为 Java 系统属性，且没有 zookeeper 前缀。如果要将“jute.maxbuffer”的值设为 X，在 ZooKeeper 客户端或服务端启动时传入以下系统属性：-Djute.maxbuffer=X。

例如，将参数值设置为 4MB：-Djute.maxbuffer=0x400000。

表30-7 配置参数

参数	描述	默认值
jute.maxbuffer	指定可以存储在 znode 中的数据的最大长度。单位是 Byte。默认值为 0xfffff，即低于 1MB。 说明 如果更改此选项，则必须在所有服务器和客户端上设置该系统属性，否则将出现问题。	0xfffff

30.6.3 为什么在 Zookeeper 服务器上启用安全的 netty 配置时，四个字母的命令不能与 linux 的 netcat 命令一起使用

问题

为什么在 Zookeeper 服务器上启用安全的 netty 配置时，四个字母的命令不能与 linux 的 ***netcat*** 命令一起使用？

例如：

echo stat |netcat host port

回答

Linux 的 ***netcat*** 命令没有与 Zookeeper 服务器安全通信的选项，所以当启用安全的 netty 配置时，它不能支持 Zookeeper 四个字母的命令。

为了避免这个问题，用户可以使用下面的 Java API 来执行四个字母的命令。

```
org.apache.zookeeper.client.FourLetterWordMain
```

例如：

```
String[] args = new String[]{host, port, "stat"};
org.apache.zookeeper.client.FourLetterWordMain.main(args);
```

说明

netcat 命令只能用于非安全的 netty 配置。

30.6.4 如何查看哪个 ZooKeeper 实例是 leader

问题

如何查看 ZooKeeper 实例的角色是 leader 还是 follower？

回答

登录 Manager，选择“集群 > *待操作集群的名称* > 服务 > ZooKeeper > 实例”，单击相应的 quorumpeer 实例名称，进入对应实例的详情页面，即可查看到该实例的“服务器状态”。

30.6.5 使用 IBM JDK 时客户端无法连接 ZooKeeper

问题

使用 IBM 的 JDK 的情况下客户端连接 ZooKeeper 失败。

回答

可能原因为 IBM 的 JDK 和普通 JDK 的 jaas.conf 文件格式不一样。

在使用 IBM JDK 时，建议使用如下 jaas.conf 文件模板，其中“useKeytab”中的文件路径必须以“file://”开头，后面为绝对路径。

```
Client {
com.ibm.security.auth.module.Krb5LoginModule required
useKeytab="file://D:/install/HbaseClientSample/conf/user.keytab"
principal="hbaseuser1"
credsType="both";
};
```

30.6.6 ZooKeeper 客户端刷新 TGT 失败

问题

ZooKeeper 客户端刷新 TGT 失败，无法连接 ZooKeeper。报错内容如下：

```
Login: Could not renew TGT due to problem running shell command: '***/kinit -R';
exception was:org.apache.zookeeper.Shell$ExitCodeException: kinit: Ticket expired
while renewing credentials
```

回答

ZooKeeper 使用系统命令 **kinit -R** 对票据进行刷新，当前 MRS 版本已经取消了该命令的功能，如需运行长任务，建议使用 keytab 方式完成鉴权功能。

在“jaas.conf”配置文件中设置属性“useTicketCache=false”，设置“useKeyTab=true”，并指明 keytab 路径。

30.6.7 使用 deleteall 命令，删除大量 znode 时，偶现报错“Node does not exist”错误

问题

客户端连接非 leader 实例，使用 deleteall 命令删除大量 znode 时，报错 Node does not exist，但是 stat 命令能够获取到 node 状态。

回答

由于网络问题或者数据量大导致 leader 和 follower 数据不同步。解决方法是客户端连接到 leader 实例进行删除操作。具体过程是首先根据如何查看哪个 ZooKeeper 实例是 leader 查看 leader 所在节点 IP，使用连接客户端命令 zkCli.sh -server leader 节点 IP:2181 成功后进行 deleteall 命令删除操作，具体操作请参见使用 ZooKeeper 客户端。

31 附录

31.1 修改集群服务配置参数

- MRS 3.x 之前版本，用户可直接通过 MRS 管理控制台的集群管理页面修改各服务配置参数：
  a. 登录 MRS 控制台，在左侧导航栏选择“集群列表 > 现有集群”，单击集群名称。
  b. 选择“组件管理 > *服务名称* > 服务配置”。
     默认显示“基础配置”，如果需要修改更多参数，请选择“全部配置”，界面上将显示该服务的全部配置参数导航树，导航树从上到下的一级节点分别为服务名称和角色名称。展开一级节点后显示参数分类。
  c. 在导航树选择指定的参数分类，并在右侧修改参数值。
     不确定参数的具体位置时，支持在右上角输入参数名，系统将实时进行搜索并显示结果。
  d. 单击“保存配置”，并在确认对话框中单击“是”。
  e. 等待界面提示“操作成功”，单击“完成”，配置已修改。
     查看集群是否存在配置过期的服务，如果存在，需重启对应服务或角色实例使配置生效。也可在保存配置时直接勾选“重新启动受影响的服务或实例。”。
- MRS 3.x 之前的版本，服务配置参数均支持登录 MRS Manager 进行修改:
  a. 登录 MRS Manager。
  b. 单击“服务管理”。
  c. 单击服务视图中指定的服务名称。
  d. 单击“服务配置”。
     默认显示“基础配置”，如果需要修改更多参数，请选择“全部配置”，界面上将显示该服务的全部配置参数导航树，导航树从上到下的一级节点分别为服务名称和角色名称。展开一级节点后显示参数分类。
  e. 在导航树选择指定的参数分类，并在右侧修改参数值。
     不确定参数的具体位置时，支持在右上角输入参数名，Manager 将实时进行搜索并显示结果。

f. 单击“保存配置”，并在确认对话框中单击“是”。

g. 等待界面提示“操作成功”，单击“完成”，配置已修改。

查看集群是否存在配置过期的服务，如果存在，需重启对应服务或角色实例使配置生效。也可在保存配置时直接勾选“重新启动受影响的服务或实例。”

- MRS 3.x 及后续版本，服务配置参数均支持登录 FusionInsight Manager 进行修改：

a. 登录 FusionInsight Manager。

b. 选择“集群 > 服务”。

c. 单击服务视图中指定的服务名称。

d. 单击“配置”。

默认显示“基础配置”，如果需要修改更多参数，请选择“全部配置”，界面上将显示该服务的全部配置参数导航树，导航树从上到下的一级节点分别为服务名称和角色名称。展开一级节点后显示参数分类。

e. 在导航树选择指定的参数分类，并在右侧修改参数值。

不确定参数的具体位置时，支持在右上角输入参数名，Manager 将实时进行搜索并显示结果。

f. 单击“保存”，并在确认对话框中单击“确定”。

g. 等待界面提示“操作成功”，单击“完成”，配置已修改。

查看集群是否存在配置过期的服务，如果存在，需重启对应服务或角色实例使配置生效。

31.2 访问集群 Manager

31.2.1 访问 MRS Manager（MRS 3.x 之前版本）

操作场景

MRS 3.x 之前版本集群使用 MRS Manager 对集群进行监控、配置和管理，用户可以在 MRS 控制台页面打开 Manager 管理页面。

访问 MRS Manager

当集群开启弹性公网 IP 功能时，执行如下步骤：

步骤 1 登录 MRS 管理控制台页面。

步骤 2 单击“集群列表 > 现有集群”，在集群列表中单击指定的集群名称，进入集群信息页面。

步骤 3 单击“前往 Manager”，打开“访问 MRS Manager 页面”。

- 若用户创建集群时已经绑定弹性公网 IP：

a. 选择待添加的安全组规则所在安全组，该安全组在创建群时配置。

b. 添加安全组规则，默认填充的是用户访问公网 IP 地址 9022 端口的规则，如需开放多个 IP 段为可信范围用于访问 MRS Manager 页面，请参考步骤 6～步

骤 9。如需对安全组规则进行查看，修改和删除操作，请单击“管理安全组规则”。

说明

- 自动获取的访问公网 IP 与用户本机 IP 不一致，属于正常现象，无需处理。
- 9022 端口为 knox 的端口，需要开启访问 knox 的 9022 端口权限，才能访问 MRS Manager 服务。

c. 勾选“我确认 xx.xx.xx.xx 为可信任的公网访问 IP，并允许从该 IP 访问 MRS Manager 页面。”

- 若用户创建集群时暂未绑定弹性公网 IP：

a. 在弹性公网 IP 下拉框中选择可用的弹性公网 IP 或单击“管理弹性公网 IP”创建弹性公网 IP。

b. 选择待添加的安全组规则所在安全组，该安全组在创建群时配置。

c. 添加安全组规则，默认填充的是用户访问公网 IP 地址 9022 端口的规则，如需开放多个 IP 段为可信范围用于访问 MRS Manager 页面，请参考步骤 6～步骤 9。如需对安全组规则进行查看，修改和删除操作，请点击“管理安全组规则”。

说明

- 自动获取的访问公网 IP 与用户本机 IP 不一致，属于正常现象，无需处理。
- 9022 端口为 knox 的端口，需要开启访问 knox 的 9022 端口权限，才能访问 MRS Manager 服务。

d. 勾选“我确认 xx.xx.xx.xx 为可信任的公网访问 IP，并允许从该 IP 访问 MRS Manager 页面。”

步骤 4 单击“确定”，进入 MRS Manager 登录页面。

说明

访问 Manager 页面需要确保弹性公网 IP 能够 Ping 通，如果 Ping 不通请联系运维处理。

步骤 5 输入默认用户名“admin”及创建集群时设置的密码，单击“登录”进入 MRS Manager 页面。

步骤 6 在 MRS 管理控制台，在“现有集群”列表，单击指定的集群名称，进入集群信息页面。

说明

如需给其他用户开通访问 MRS Manager 的权限，请执行步骤 6-步骤 9，添加对应用户访问公网的 IP 地址为可信范围。

步骤 7 单击弹性公网 IP 后边的“添加安全组规则”。

步骤 8 进入“添加安全组规则”页面，添加需要开放权限用户访问公网的 IP 地址段并勾选“我确认这里设置的授权对象是可信任的公网访问 IP 范围，禁止使用 0.0.0.0/0,否则会有安全风险。”

默认填充的是用户访问公网的 IP 地址，用户可根据需要修改 IP 地址段，如需开放多个 IP 段为可信范围，请重复执行步骤 6-步骤 9。如需对安全组规则进行查看，修改和删除操作，请点击“管理安全组规则”。

步骤 9 单击“确定”完成安全组规则添加。

----结束

为其他用户开通访问 MRS Manager 的权限

步骤 1 在 MRS 管理控制台，在“现有集群”列表，单击指定的集群名称，进入集群信息页面。

步骤 2 单击弹性公网 IP 后边的“添加安全组规则”。

步骤 3 进入“添加安全组规则”页面，添加需要开放权限用户访问公网的 IP 地址段并勾选“我确认这里设置的授权对象是可信任的公网访问 IP 范围，禁止使用 0.0.0.0/0,否则会有安全风险。”

默认填充的是用户访问公网的 IP 地址，用户可根据需要修改 IP 地址段，如需开放多个 IP 段为可信范围，请重复执行步骤 1-步骤 4。如需对安全组规则进行查看，修改和删除操作，请点击“管理安全组规则”。

步骤 4 单击“确定”完成安全组规则添加。

----结束

31.2.2 访问 FusionInsight Manager（MRS 3.x 及之后版本）

操作场景

MRS 3.x 及之后版本的集群使用 FusionInsight Manager 对集群进行监控、配置和管理。用户在集群安装后可使用帐号登录 FusionInsight Manager。

说明

如果不能正常登录组件的 WebUI 页面，请参考通过 ECS 访问 FusionInsight Manager 方式访问 FusionInsight Manager。

通过弹性 IP 访问 FusionInsight Manager

步骤 1 登录 MRS 管理控制台页面。

步骤 2 单击“集群列表 > 现有集群”，在集群列表中单击指定的集群名称，进入集群信息页面。

步骤 3 单击“集群管理页面 ”后的“前往 Manager”，在弹出的窗口中配置弹性 IP 信息。

1. 若创建 MRS 集群时暂未绑定弹性公网 IP，在“弹性公网 IP”下拉框中选择可用的弹性公网 IP。若用户创建集群时已经绑定弹性公网 IP，直接执行步骤 3.2

说明

如果没有弹性公网 IP，可先单击“管理弹性公网 IP”创建弹性公网 IP 后，然后在弹性公网 IP 下拉框中选择创建的弹性公网 IP。

2. 在“安全组”中选择待添加的安全组规则所在安全组，该安全组在创建群时配置。
3. 添加安全组规则，默认填充的是用户访问弹性 IP 地址的规则，如需开放多个 IP 段为可信范围用于访问 Manager 页面，请参考步骤 6～步骤 9。如需对安全组规则进行查看，修改和删除操作，请单击“管理安全组规则”。
4. 勾选确认信息后，单击“确定”。

步骤 4 单击“确定”，进入 Manager 登录页面。

步骤 5 输入默认用户名“admin”及创建集群时设置的密码，单击“登录”进入 Manager 页面。

步骤 6 在 MRS 管理控制台，在“现有集群” 列表，单击指定的集群名称，进入集群信息页面。

说明

如需给其他用户开通访问 Manager 的权限，请执行步骤 6～步骤 9，添加对应用户访问公网的 IP 地址为可信范围。

步骤 7 单击弹性公网 IP 后边的“添加安全组规则”。

步骤 8 进入“添加安全组规则”页面，添加需要开放权限用户访问公网的 IP 地址段并勾选“我确认这里设置的*公网 IP/端口号*是可信任的公网访问 IP 范围，我了解使用 0.0.0.0/0 会带来安全风险”

默认填充的是用户访问公网的 IP 地址，用户可根据需要修改 IP 地址段，如需开放多个 IP 段为可信范围，请重复执行步骤 6-步骤 9。如需对安全组规则进行查看，修改和删除操作，请单击“管理安全组规则”。

步骤 9 单击“确定”完成安全组规则添加。

----结束

通过 ECS 访问 FusionInsight Manager

步骤 1 在 MRS 管理控制台，单击“集群列表”。

步骤 2 在“现有集群”列表中，单击指定的集群名称。

记录集群的“可用区”、“虚拟私有云”、“集群管理页面”、“安全组”。

步骤 3 在管理控制台首页服务列表中选择“弹性云主机”，进入 ECS 管理控制台，创建一个新的弹性云主机。

- 弹性云主机的“可用区”、“虚拟私有云”、“安全组”，需要和待访问集群的配置相同。
- 选择一个 Windows 系统的公共镜像。例如，选择一个标准镜像“Windows Server 2012 R2 Standard 64bit(40GB)”。

- 其他配置参数详细信息，请参见“弹性云主机 > 用户指南 > 快速入门 > 创建并登录 Windows 弹性云主机”。

说明

如果 ECS 的安全组和 Master 节点的“默认安全组”不同，用户可以选择以下任一种方法修改配置：

- 将 ECS 的安全组修改为 Master 节点的默认安全组，请参见“弹性云主机 > 用户指南 > 安全组 > 更改安全组”。
- 在集群 Master 节点和 Core 节点的安全组添加两条安全组规则使 ECS 可以访问集群，“协议”需选择为“TCP”，“端口”需分别选择“28443”和“20009”。请参见“虚拟私有云 > 用户指南 > 安全性 > 安全组 > 添加安全组规则”。

步骤 4 在 VPC 管理控制台，申请一个弹性 IP 地址，并与 ECS 绑定。

具体请参见“虚拟私有云 > 用户指南 > 弹性公网 IP > 为弹性云主机申请和绑定弹性公网 IP”。

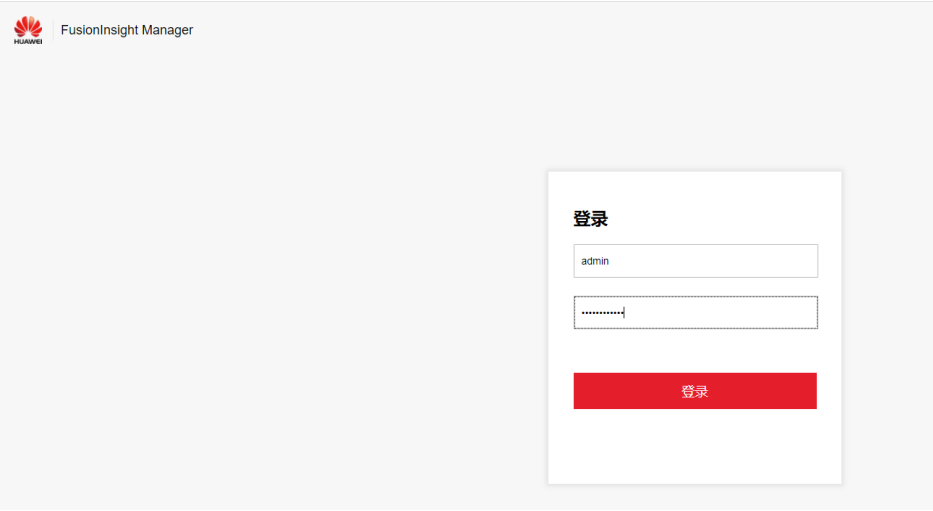
步骤 5 登录弹性云主机。

登录 ECS 需要 Windows 系统的帐号、密码，弹性 IP 地址以及配置安全组规则。具体请参见“弹性云主机 > 用户指南 > 实例 > 登录弹性云主机 > 登录 Windows 弹性云主机”。

步骤 6 在 Windows 的远程桌面中，打开浏览器访问 Manager。

例如 Windows 2012 操作系统可以使用 Internet Explorer 11。

Manager 访问地址为“集群管理页面”地址。访问时需要输入集群的用户名和密码，例如“admin”用户。



说明

- 如果使用其他集群用户访问 Manager，第一次访问时需要修改密码。新密码需要满足集群当前的用户密码复杂度策略。请咨询管理员。
- 默认情况下，在登录时输入 5 次错误密码将锁定用户，需等待 5 分钟自动解锁。

步骤 7 注销用户退出 Manager 时移动鼠标到右上角  ，然后单击“注销”。

----结束

31.3 使用 MRS 客户端

31.3.1 安装客户端（3.x 及之后版本）

操作场景

该操作指导安装工程师安装 MRS 集群所有服务（不包含 Flume）的客户端。Flume 客户端安装请参见“组件操作指南 > 使用 Flume > 安装 Flume 客户端”。

客户端可以安装集群内节点，也可以安装在集群外节点，本章节以安装目录“/opt/client”为例进行介绍，请以实际集群版本为准。

在集群外节点安装客户端前提条件

- 已准备一个 Linux 弹性云主机，主机操作系统及版本建议参见表 31-1。

表31-1 参考列表

CPU 架构	操作系统	支持的版本号
x86 计算	Euler	EulerOS 2.5
	SUSE	SUSE Linux Enterprise Server 12 SP4（SUSE 12.4）
	RedHat	RedHat-7.5-x86_64（RedHat 7.5）
	CentOS	CentOS-7.6 版本（CentOS 7.6）
鲲鹏计算 (ARM)	Euler	EulerOS 2.8
	CentOS	CentOS-7.6 版本（CentOS 7.6）

  同时为弹性云服务分配足够的磁盘空间，例如“40GB”。
- 弹性云主机的 VPC 需要与 MRS 集群在同一个 VPC 中。
- 弹性云主机的安全组需要和 MRS 集群 Master 节点的安全组相同。
- 弹性云主机操作系统已安装 NTP 服务，且 NTP 服务运行正常。
  若未安装，在配置了 **yum** 源的情况下，可执行 **yum install ntp -y** 命令自行安装。
- 需要允许用户使用密码方式登录 Linux 弹性云主机（SSH 方式）。

集群内节点安装客户端

1. 获取软件包。

访问 FusionInsight Manager（MRS 3.x 及之后版本），在“集群”下拉列表中单击需要操作的集群名称。

选择“更多 > 下载客户端”，弹出“下载集群客户端”信息提示框。

图31-1 下载客户端

说明

在只安装单个服务的客户端的场景中，选择“集群 > 服务 > *服务名称* > 更多 > 下载客户端”，弹出“下载客户端”信息提示框。

2. “选择客户端类型”中选择“完整客户端”。

   “仅配置文件”下载的客户端配置文件，适用于应用开发任务中，完整客户端已下载并安装后，管理员通过 Manager 界面修改了服务端配置，开发人员需要更新客户端配置文件的场景。

   平台类型包括 x86_64 和 aarch64 两种：

   - x86_64：可以部署在 X86 平台的客户端软件包。
   - aarch64：可以部署在 TaiShan 服务器的客户端软件包。

说明

集群支持下载 x86_64 和 aarch64 两种类型客户端，但是客户端类型必须与待安装节点的架构匹配，否则客户端会安装失败。

3. 勾选“仅保存到如下路径”，单击“确定”开始生成客户端文件。

   文件生成后默认保存在主管理节点“/tmp/FusionInsight-Client”。支持自定义其他目录且 **omm** 用户拥有目录的读、写与执行权限。单击“确定”，等待下载完成后，使用 **omm** 用户或 **root** 用户将获取的软件包复制到将要安装客户端的服务器文件目录。

   客户端软件包名称格式为：“FusionInsight_Cluster_<*集群ID*>_Services_Client.tar”。

后续步骤及章节以 FusionInsight_Cluster_1_Services_Client.tar 进行举例。

说明

当用户无法获取 **root** 用户权限，需要用 **omm** 用户操作。

如需安装客户端至集群内其他节点，则执行以下命令复制客户端到待安装客户端的节点：

scp -p */tmp/FusionInsight-Client/***FusionInsight_Cluster_1_Services_Client.tar** *待安装客户端节点的 IP 地址:/opt/Bigdata/client*

4. 以 **user_client** 用户登录将要安装客户端的服务器。
5. 解压软件包。

   进入安装包所在目录，例如“/tmp/FusionInsight-Client”。执行如下命令解压安装包到本地目录。

   tar -xvf FusionInsight_Cluster_1_Services_Client.tar
6. 校验软件包。

   执行 **sha256sum** 命令校验解压得到的文件，检查回显信息与 sha256 文件里面的内容是否一致，例如：

   sha256sum -c FusionInsight_Cluster_1_Services_ClientConfig.tar.sha256

   ```
   FusionInsight_Cluster_1_Services_ClientConfig.tar: OK
   ```
7. 解压获取的安装文件。

   tar -xvf FusionInsight_Cluster_1_Services_ClientConfig.tar
8. 进入安装包所在目录，执行如下命令安装客户端到指定目录（绝对路径），例如安装到“/opt/client”目录。

   cd /tmp/FusionInsight-Client/FusionInsight_Cluster_1_Services_ClientConfig

   执行**./install.sh /opt/client** 命令，等待客户端安装完成（以下只显示部分屏显结果）。

   ```
   The component client is installed successfully
   ```

说明

- 如果已经安装的全部服务或某个服务的客户端使用了“/opt/client”目录，再安装其他服务的客户端时，需要使用不同的目录。
- 卸载客户端请删除客户端安装目录。
- 如果要求安装后的客户端仅能被该安装用户（如“user_client”）使用，请在安装时加“-o”参数，即执行**./install.sh /opt/client -o** 命令安装客户端。
- 由于 HBase 使用的 Ruby 语法限制，如果安装的客户端中包含了 HBase 客户端，建议客户端安装目录路径只包含大写字母、小写字母、数字以及_-?.@+=字符。

使用客户端

1. 在已安装客户端的节点，执行 **sudo su - omm** 命令切换用户。执行以下命令切换到客户端目录：

   cd /opt/client
2. 执行以下命令配置环境变量：

   source bigdata_env

3. 如果当前集群已启用 Kerberos 认证，执行以下命令认证当前用户。如果当前集群未启用 Kerberos 认证，则无需执行此命令。

   kinit*MRS 集群用户*

   例如，**kinit admin**。

   说明

   启用 Kerberos 认证的 MRS 集群默认创建“admin”用户帐号，用于集群管理员维护集群。

4. 直接执行组件的客户端命令。

   例如：使用 HDFS 客户端命令查看 HDFS 根目录文件，执行 **hdfs dfs -ls /**。

集群外节点安装客户端

1. 根据在集群外节点安装客户端前提条件，创建一个满足要求的弹性云主机。
2. 执行 ntp 时间同步，使集群外节点的时间与 MRS 集群时间同步。
   a. 执行 **vi /etc/ntp.conf** 命令编辑 NTP 客户端配置文件，并增加 MRS 集群中 Master 节点的 IP 并注释掉其他 **server** 的地址。

```
server master1_ip prefer
server master2_ip
```

图31-2 增加 Master 节点的 IP

```
# For more information about this file, see the man pages
# ntp.conf(5), ntp_acc(5), ntp_auth(5), ntp_clock(5), ntp_misc(5), ntp_mon(5).

driftfile /var/lib/ntp/drift

# Permit time synchronization with our time source, but do not
# permit the source to query or modify the service on this system.
restrict default nomodify notrap nopeer noquery

# Permit all access over the loopback interface.  This could
# be tightened as well, but to do so would effect some of
# the administrative functions.
restrict 127.0.0.1
restrict ::1

# Hosts on local network are less restricted.
#restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap

# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).
#server 0.centos.pool.ntp.org iburst
#server 1.centos.pool.ntp.org iburst
#server 2.centos.pool.ntp.org iburst
#server 3.centos.pool.ntp.org iburst
#[redacted]
server 10.9.2.38 prefer
server 10.9.2.39
#broadcast 192.168.1.255 autokey        # broadcast server
#broadcastclient                        # broadcast client
#broadcast 224.0.1.1 autokey            # multicast server
#multicastclient 224.0.1.1              # multicast client
#manycastserver 239.255.254.254         # manycast server
#manycastclient 239.255.254.254 autokey # manycast client

# Enable public key cryptography.
#crypto
```

b. 执行 **service ntpd stop** 命令关闭 NTP 服务。

c. 执行如下命令，手动同步一次时间：

/usr/sbin/ntpdate *192.168.10.8*

说明

192.168.10.8 为主 Master 节点的 IP 地址。

d. 执行 **service ntpd start** 或 **systemctl restart ntpd** 命令启动 NTP 服务。

e. 执行 **ntpstat** 命令查看时间同步结果。

3. 参考以下步骤，从 FusionInsight Manager 下载集群客户端软件包并复制到 ECS 节点后安装客户端。

a. 访问 FusionInsight Manager（MRS 3.x 及之后版本），参考集群内节点安装客户端下载集群客户端到主管理节点的指定目录。

b. 使用 **root** 用户登录主管理节点，执行以下命令复制客户端安装包到待安装客户端的节点：

scp -p /tmp/FusionInsight-Client/FusionInsight_Cluster_1_Services_Client.tar *待安装客户端节点的 IP 地址*:**/tmp**

c. 使用待安装客户端的用户登录待安装客户端节点。

执行以下命令安装客户端，如果当前用户无客户端软件包以及客户端安装目录的操作权限，需使用 **root** 用户进行赋权：

cd /tmp

tar -xvf FusionInsight_Cluster_1_Services_Client.tar

tar -xvf FusionInsight_Cluster_1_Services_ClientConfig.tar

cd FusionInsight_Cluster_1_Services_ClientConfig

./install.sh /opt/client

d. 执行以下命令，切换到客户端目录并配置环境变量：

cd /opt/client

source bigdata_env

e. 如果当前集群已启用 Kerberos 认证，执行以下命令认证当前用户。如果当前集群未启用 Kerberos 认证，则无需执行此命令。

kinit *MRS 集群用户*

例如，**kinit admin**。

f. 直接执行组件的客户端命令。

例如使用 HDFS 客户端命令查看 HDFS 根目录文件，执行 **hdfs dfs -ls /**。

31.3.2 安装客户端（3.x 之前版本）

操作场景

用户需要使用 MRS 客户端。MRS 集群客户端可以安装在集群内的 Master 节点或者 Core 节点，也可以安装在集群外节点上。

MRS 3.x 之前版本集群在集群创建后，在主 Master 节点默认安装有客户端，可以直接使用，安装目录为“/opt/client”。

MRS 3.x 及之后版本客户端的安装请参考安装客户端（3.x 及之后版本）。

说明

如果集群外的节点已安装客户端且只需要更新客户端，请使用安装客户端的用户例如“root”。

在集群外节点安装客户端前提条件

- 已准备一个弹性云主机，主机操作系统及版本请参见表 31-2。

表31-2 参考列表

操作系统	支持的版本号
Euler	• 可用：Euler OS 2.2 • 可用：Euler OS 2.3 • 可用：Euler OS 2.5

例如，用户可以选择操作系统为 **Euler** 的弹性云主机准备操作。

同时为弹性云服务分配足够的磁盘空间，例如“40GB”。

- 弹性云主机的 VPC 需要与 MRS 集群在同一个 VPC 中。
- 弹性云主机的安全组需要和 MRS 集群 Master 节点的安全组相同。

  如果不同，请修改弹性云主机安全组或配置弹性云主机安全组的出入规则允许 MRS 集群所有安全组的访问。
- 需要允许用户使用密码方式登录 Linux 弹性云主机（SSH 方式），请参见弹性云主机《用户指南》中“实例> 登录 Linux 弹性云主机 >SSH 密码方式登录”。

在 Core 节点安装客户端

1. 登录 MRS Manager 页面，选择“服务管理 > 下载客户端”下载客户端安装包至主管理节点。

   说明

   如仅需更新客户端配置文件，请参考更新客户端（3.x 之前版本）页面的方法二操作。
2. 使用 IP 地址搜索主管理节点并使用 VNC 登录主管理节点。
3. 在主管理节点，执行以下命令切换用户。

   sudo su - omm
4. 在 MRS 管理控制台，查看指定集群“节点管理”页面的“IP”地址。

   记录需使用客户端的 Core 节点 IP 地址。
5. 在主管理节点，执行以下命令，将客户端安装包从主管理节点文件拷贝到当前 Core 节点：

   scp -p /tmp/MRS-client/MRS_Services_Client.tar *Core 节点的 IP 地址*:**/opt/client**
6. 使用“root”登录 Core 节点。

Master 节点支持 Cloud-Init 特性，Cloud-init 预配置的用户名“root”，密码为创建集群时设置的密码。

7. 执行以下命令，安装客户端：

   cd /opt/client

   tar -xvf MRS_Services_Client.tar

   tar -xvf MRS_Services_ClientConfig.tar

   cd /opt/client/MRS_Services_ClientConfig

   ./install.sh *客户端安装目录*

   例如，执行命令：

   ./install.sh /opt/client

8. 客户端的使用请参见使用 MRS 客户端。

使用 MRS 客户端

1. 在已安装客户端的节点，执行 **sudo su - omm** 命令切换用户。执行以下命令切换到客户端目录：

   cd /opt/client

2. 执行以下命令配置环境变量：

   source bigdata_env

3. 如果当前集群已启用 Kerberos 认证，执行以下命令认证当前用户。如果当前集群未启用 Kerberos 认证，则无需执行此命令。

   kinit *MRS 集群用户*

   例如，**kinit admin**。

   说明

   启用 Kerberos 认证的 MRS 集群默认创建“admin”用户帐号，用于集群管理员维护集群。

4. 直接执行组件的客户端命令。

   例如：使用 HDFS 客户端命令查看 HDFS 根目录文件，执行 **hdfs dfs -ls /**。

在集群外节点上安装客户端

步骤 1 根据前提条件，创建一个满足要求的弹性云主机。

步骤 2 登录 MRS Manager 页面，具体请参见访问 MRS Manager（MRS 3.x 之前版本），然后选择“服务管理”。

步骤 3 单击“下载客户端”。

步骤 4 在“客户端类型”选择“完整客户端”。

步骤 5 在“下载路径”选择“远端主机”。

步骤 6 将“主机 IP”设置为 ECS 的 IP 地址，设置“主机端口”为“22”，并将“存放路径”设置为“/tmp”。

- 如果使用 SSH 登录 ECS 的默认端口“22”被修改，请将“主机端口”设置为新端口。

- “存放路径”最多可以包含 256 个字符。

步骤 7 “登录用户”设置为“root”。

如果使用其他用户，请确保该用户对保存目录拥有读取、写入和执行权限。

步骤 8 在“登录方式”选择“密码”或“SSH 私钥”。

- 密码：输入创建集群时设置的 root 用户密码。
- SSH 私钥：选择并上传创建集群时使用的密钥文件。

步骤 9 单击“确定”开始生成客户端文件。

若界面显示以下提示信息表示客户端包已经成功保存。单击“关闭”。客户端文件请到下载客户端时设置的远端主机的“存放路径”中获取。

```
下载客户端文件到远端主机成功。
```

若界面显示以下提示信息，请检查用户名密码及远端主机的安全组配置，确保用户名密码正确，及远端主机的安全组已增加 SSH(22)端口的入方向规则。然后从步骤 2 执行重新开始下载客户端。

```
连接到服务器失败，请检查网络连接或参数设置。
```

说明

生成客户端会占用大量的磁盘 IO，不建议在集群处于安装中、启动中、打补丁中等非稳态场景下载客户端。

步骤 10 使用 VNC 方式，登录弹性云主机。参见弹性云主机《用户指南》的**远程登录（VNC 方式）**章节（“实例 > 登录 Linux 弹性云主机 > 远程登录（VNC 方式）”）。

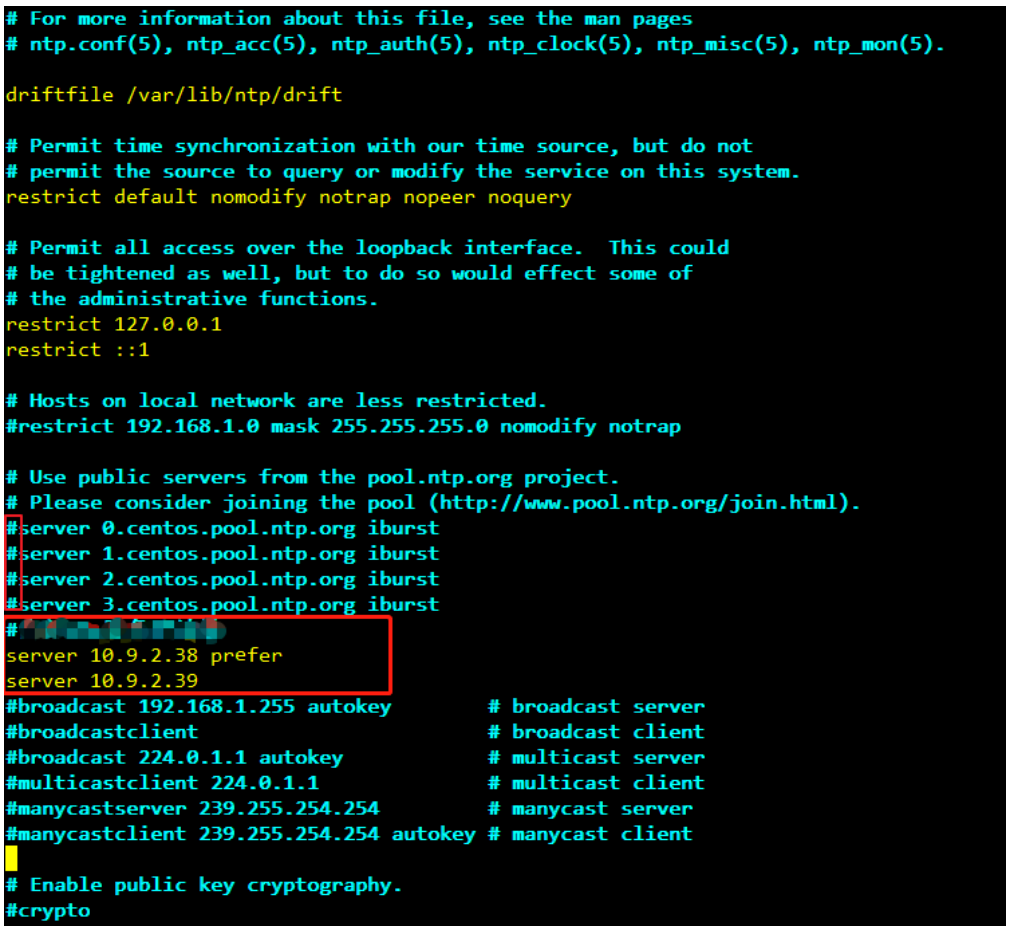
所有镜像均支持 Cloud-init 特性。Cloud-init 预配置的用户名“root”，密码为创建集群时设置的密码。首次登录建议修改。

步骤 11 执行 ntp 时间同步，使集群外节点的时间与 MRS 集群时间同步。

1. 检查安装 NTP 服务有没有安装，未安装请执行 **yum install ntp -y** 命令自行安装。
2. 执行 **vim /etc/ntp.conf** 命令编辑 NTP 客户端配置文件，并增加 MRS 集群中 Master 节点的 IP 并注释掉其他 **server** 的地址。

```
server master1_ip prefer
server master2_ip
```

图31-3 增加 Master 节点的 IP



3. 执行 **service ntpd stop** 命令关闭 NTP 服务。
4. 执行如下命令，手动同步一次时间：

   /usr/sbin/ntpdate *192.168.10.8*

   说明

   192.168.10.8 为主 Master 节点的 IP 地址。
5. 执行 **service ntpd start** 或 **systemctl restart ntpd** 命令启动 NTP 服务。
6. 执行 **ntpstat** 命令查看时间同步结果。

步骤 12 在弹性云主机，切换到 **root** 用户，并将步骤 6 中“存放路径”中的安装包复制到目录“/opt”，例如“存放路径”设置为“/tmp”时命令如下。

sudo su - root

cp /tmp/MRS_Services_Client.tar /opt

步骤 13 在“/opt”目录执行以下命令，解压压缩包获取校验文件与客户端配置包。

tar -xvf MRS_Services_Client.tar

步骤 14 执行以下命令，校验文件包。

sha256sum -c MRS_Services_ClientConfig.tar.sha256

界面显示如下：

```
MRS_Services_ClientConfig.tar: OK
```

步骤 15 执行以下命令，解压“MRS_Services_ClientConfig.tar”。

tar -xvf MRS_Services_ClientConfig.tar

步骤 16 执行以下命令，安装客户端到新的目录，例如“/opt/Bigdata/client”。安装时自动生成目录。

sh /opt/MRS_Services_ClientConfig/install.sh /opt/Bigdata/client

查看安装输出信息，如有以下结果表示客户端安装成功：

```
Components client installation is complete.
```

步骤 17 验证弹性云主机节点是否与集群 Master 节点的 IP 是否连通？

例如，执行以下命令：**ping** *Master 节点 IP 地址*

- 是，执行步骤 18。
- 否，检查 VPC、安全组是否正确，是否与 MRS 集群在相同 VPC 和安全组，然后执行步骤 18。

步骤 18 执行以下命令配置环境变量：

source /opt/Bigdata/client/bigdata_env

步骤 19 如果当前集群已启用 Kerberos 认证，执行以下命令认证当前用户。如果当前集群未启用 Kerberos 认证，则无需执行此命令。

kinit*MRS 集群用户*

例如, **kinit admin**

步骤 20 执行组件的客户端命令。

例如，执行以下命令查看 HDFS 目录：

hdfs dfs -ls /

----结束

31.3.3 更新客户端（3.x 及之后版本）

集群提供了客户端，可以在连接服务端、查看任务结果或管理数据的场景中使用。用户如果在 Manager 修改了服务配置参数并重启了服务，已安装的客户端需要重新下载并安装，或者使用配置文件更新客户端。

更新客户端配置

方法一：

步骤 1 访问 FusionInsight Manager（MRS 3.x 及之后版本），在“集群”下拉列表中单击需要操作的集群名称。

步骤 2 选择“更多 > 下载客户端 > 仅配置文件”。

此时生成的压缩文件包含所有服务的配置文件。

步骤 3 是否在集群的节点中生成配置文件？

- 是，勾选“仅保存到如下路径”，单击“确定”开始生成客户端文件，文件生成后默认保存在主管理节点“/tmp/FusionInsight-Client”。支持自定义其他目录且 **omm** 用户拥有目录的读、写与执行权限。然后执行步骤 4。
- 否，单击“确定”指定本地的保存位置，开始下载完整客户端，等待下载完成，然后执行步骤 4。

步骤 4 使用 WinSCP 工具，以客户端安装用户将压缩文件保存到客户端安装的目录，例如“/opt/hadoopclient”。

步骤 5 解压软件包。

例如下载的客户端文件为“FusionInsight_Cluster_1_Services_Client.tar”执行如下命令进入客户端所在目录，解压文件到本地目录。

cd /opt/hadoopclient

tar -xvf FusionInsight_Cluster_1_Services_Client.tar

步骤 6 校验软件包。

执行 **sha256sum** 命令校验解压得到的文件，检查回显信息与 sha256 文件里面的内容是否一致，例如：

sha256sum -c FusionInsight_Cluster_1_Services_ClientConfig_ConfigFiles.tar.sha256

```
FusionInsight_Cluster_1_Services_ClientConfig_ConfigFiles.tar: OK
```

步骤 7 解压获取配置文件。

tar -xvf FusionInsight_Cluster_1_Services_ClientConfig_ConfigFiles.tar

步骤 8 在客户端安装目录下执行如下命令，使用配置文件更新客户端。

sh refreshConfig.sh *客户端安装目录 配置文件所在目录*

例如，执行以下命令：

sh refreshConfig.sh /opt/hadoopclient /opt/hadoopclient/FusionInsight_Cluster_1_Services_ClientConfig_ConfigFiles

界面显示以下信息表示配置刷新更新成功：

```
Succeed to refresh components client config.
```

----结束

方法二：

步骤 1 以 **root** 用户登录客户端安装节点。

步骤 2 进入客户端安装的目录，例如“/opt/hadoopclient”，执行以下命令更新配置文件：

cd /opt/hadoopclient

sh autoRefreshConfig.sh

步骤 3 按照提示输入 FusionInsight Manager 管理员用户名，密码以及 FusionInsight Manager 界面浮动 IP。

步骤 4 输入需要更新配置的组件名，组件名之间使用“,”分隔。如需更新所有组件配置，可直接单击回车键。

界面显示以下信息表示配置刷新更新成功：

```
Succeed to refresh components client config.
```

----结束

31.3.4 更新客户端（3.x 之前版本）

说明

本章节适用于 MRS 3.x 之前版本的集群。MRS 3.x 及之后版本，请参考更新客户端（3.x 及之后版本）。

更新客户端配置文件

操作场景

MRS 集群提供了客户端，可以在连接服务端、查看任务结果或管理数据的场景中使用。用户使用 MRS 的客户端时，如果在 MRS Manager 修改了服务配置参数并重启了服务或者重启了服务，需要先下载并更新客户端配置文件。

用户创建集群时，默认在集群所有节点的“/opt/client”目录安装保存了原始客户端。集群创建完成后，仅 Master 节点的客户端可以直接使用，Core 节点客户端在使用前需要更新客户端配置文件。

操作步骤

方法一：

步骤 1 登录 MRS Manager 页面，具体请参见访问 MRS Manager（MRS 3.x 之前版本），然后选择“服务管理”。

步骤 2 单击“下载客户端”。

“客户端类型”选择“仅配置文件”，“下载路径”选择“服务器端”，单击“确定”开始生成客户端配置文件，文件生成后默认保存在主管理节点“/tmp/MRS-client”。文件保存路径支持自定义。

步骤 3 查询并登录主 Master 节点。

步骤 4 若在集群内使用客户端，执行以下命令切换到 omm 用户，若在集群外使用客户端，请切换到 root 用户：

sudo su - omm

步骤 5 执行以下命令切换客户端目录，例如“**/opt/Bigdata/client**”：

cd /opt/Bigdata/client

步骤 6 执行以下命令，更新客户端配置：

sh refreshConfig.sh *客户端安装目录客户端配置文件压缩包完整路径*

例如，执行命令：

sh refreshConfig.sh /opt/Bigdata/client /tmp/MRS-client/MRS_Services_Client.tar

界面显示以下信息表示配置刷新更新成功：

```
ReFresh components client config is complete.
Succeed to refresh components client config.
```

----结束

方法二：

步骤 1 集群安装完成之后，执行以下命令切换到 omm 用户，若在集群外使用客户端，请切换到 root 用户。

sudo su - omm

步骤 2 执行以下命令切换客户端目录，例如“**/opt/Bigdata/client**”。

cd /opt/Bigdata/client

步骤 3 执行以下命令并按照提示输入 MRS Manager 有下载权限的用户名和密码（例如，用户名为 admin，密码为创建集群时设置的密码），更新客户端配置。

sh autoRefreshConfig.sh

步骤 4 命令执行后显示如下信息，其中 XXX 表示集群安装的组件名称，如需更新全部组件配置，单击“Enter”键，如需更新部分组件配置，请输入需要更新的组件名称，多个组件名称以逗号相隔。

```
Components "xxx" have been installed in the cluster. Please input the comma-
separated names of the components for which you want to update client
configurations. If you press Enter without inputting any component name, the client
configurations of all components will be updated:
```

界面显示以下信息表示配置更新成功：

```
Succeed to refresh components client config.
```

界面显示以下信息表示用户名或者密码错误：

```
login manager failed,Incorrect username or password.
```

说明

- 该脚本会自动连接到集群并调用 refreshConfig.sh 脚本下载并刷新客户端配置文件。
- 客户端默认使用安装目录下文件 Version 中的“wsom=xxx”所配置的浮动 IP 刷新客户端配置，如需刷新为其他集群的配置文件，请执行本步骤前修改 Version 文件中“wsom=xxx”的值为对应集群的浮动 IP 地址。

----结束

全量更新主 Master 节点的原始客户端

场景描述

用户创建集群时，默认在集群所有节点的“/opt/client”目录安装保存了原始客户端。以下操作以“/opt/Bigdata/client”为例进行说明。

- MRS 普通集群，在 console 页面提交作业时，会使用 master 节点上预置安装的客户端进行作业提交。
- 用户也可使用 master 节点上预置安装的客户端来连接服务端、查看任务结果或管理数据等

对集群安装补丁后，用户需要重新更新 master 节点上的客户端，才能保证继续使用内置客户端功能。

操作步骤

步骤 1 登录 MRS Manager 页面，具体请参见访问 MRS Manager（MRS 3.x 之前版本），然后选择“服务管理”。

步骤 2 单击“下载客户端”。

“客户端类型”选择“完整客户端”，“下载路径”选择“服务器端”，单击“确定”开始生成客户端配置文件，文件生成后默认保存在主管理节点“/tmp/MRS-client”。文件保存路径支持自定义。

步骤 3 查询并登录主 Master 节点。

步骤 4 在弹性云主机，切换到 **root** 用户，并将安装包复制到目录“/opt”。

sudo su - root

cp /tmp/MRS-client/MRS_Services_Client.tar /opt

步骤 5 在“/opt”目录执行以下命令，解压压缩包获取校验文件与客户端配置包。

tar -xvf MRS_Services_Client.tar

步骤 6 执行以下命令，校验文件包。

sha256sum -c MRS_Services_ClientConfig.tar.sha256

界面显示如下：

```
MRS_Services_ClientConfig.tar: OK
```

步骤 7 执行以下命令，解压“MRS_Services_ClientConfig.tar”。

tar -xvf MRS_Services_ClientConfig.tar

步骤 8 执行以下命令，移走原来老的客户端到/opt/Bigdata/client_bak 目录下

mv /opt/Bigdata/client /opt/Bigdata/client_bak

步骤 9 执行以下命令，安装客户端到新的目录，客户端路径必须为“/opt/Bigdata/client”。

sh /opt/MRS_Services_ClientConfig/install.sh /opt/Bigdata/client

查看安装输出信息，如有以下结果表示客户端安装成功：

```
Components client installation is complete.
```

步骤 10 执行以下命令，修改/opt/Bigdata/client 目录的所属用户和用户组。

chown omm:wheel /opt/Bigdata/client -R

步骤 11 执行以下命令配置环境变量：

source /opt/Bigdata/client/bigdata_env

步骤 12 如果当前集群已启用 Kerberos 认证，执行以下命令认证当前用户。如果当前集群未启用 Kerberos 认证，则无需执行此命令。

kinit*MRS 集群用户*

例如, **kinit admin**

步骤 13 执行组件的客户端命令。

例如，执行以下命令查看 HDFS 目录：

hdfs dfs -ls /

----结束

全量更新备 Master 节点的原始客户端

步骤 1 参见步骤 1~步骤 3 登录备 Master 节点，执行如下命令切换到 omm 用户。

sudo su - omm

步骤 2 在备 master 节点上执行如下命令，从主 master 节点拷贝下载的客户端包。

scp omm@*master1 节点 IP 地址***:/tmp/MRS-client/MRS_Services_Client.tar /tmp/MRS-client/**

说明

- 该命令以 master1 节点为主 master 节点为例。
- 目的路径以备 master 节点的/tmp/MRS-client/目录为例，请根据实际路径修改。

步骤 3 参见步骤 4~步骤 13，更新备 Master 节点的客户端。

----结束