



云搜索服务 (ES)

故障排除

天翼云科技有限公司

目 录

1 访问集群类	4
1.1 无法正常打开 Kibana.....	4
1.2 Elasticsearch 针对 filebeat 配置调优.....	5
1.3 TransportClient 客户端连接 css 报错.....	6
1.4 Spring Boot 使用 Elasticsearch 出现 Connection reset by peer 问题.....	6
1.5 为什么集群创建失败.....	7
1.6 Elasticsearch 集群出现写入拒绝“Bulk Reject”，如何解决？.....	8
1.7 Elasticsearch 集群创建 index pattern 卡住.....	10
1.8 云搜索控制台页面提示系统繁忙.....	11
1.9 Elasticsearch 集群报错：unassigned shards all indices.....	11
1.10 es-head 插件连接 Elasticsearch 集群报跨域错误.....	11
1.11 单节点集群打开 Cerebro 界面显示告警.....	12
2 集群不可用	13
2.1 集群不可用排查指导.....	13
2.2 集群冻结状态导致集群不可用.....	17
2.3 X-pack 参数配置导致集群不可用.....	17
2.4 安全组策略设置不合理导致集群不可用.....	18
2.5 插件不兼容导致集群不可用.....	20
2.6 分片未正常分配导致集群不可用.....	21
2.7 数据类型不兼容导致集群不可用.....	32
2.8 集群负载过高导致集群不可用.....	34
3 数据导入导出类	37
3.1 Elasticsearch 显示 CPU 使用率高，导致日志无法写入.....	37
3.2 ECS 服务器部署 logstash 推送数据到云搜索服务报错.....	38
3.3 ES-Hadoop 导出数据时报“Could not write all entries”异常.....	38
4 功能使用类	40
4.1 无法备份索引.....	40
4.2 无法使用自定义词库功能.....	41
4.3 快照仓库找不到.....	41

4.4 集群一直处于快照中	42
4.5 数据量很大，如何进行快照备份?.....	43
4.6 集群突现 load 高的故障排查.....	44
4.7 使用 Elasticsearch 的 HLRC (High Level Rest Client) 时，报出 I/O Reactor STOPPED.....	45
4.8 Elasticsearch 集群最大堆内存持续过高 (超过 90%)	47
4.9 Elasticsearch 集群更改规格失败	48
4.10 安全集群索引只读状态修改报错.....	49
4.11 Elasticsearch 集群某一节点分配不到 shard	50
4.12 集群索引插入数据失败	50
4.13 CSS 创建索引报错 maximum shards open	51
4.14 删除索引报错 “403 Forbidden” 是什么原因?	51
4.15 Kibana 中删除 index pattern 报错 Forbidden.....	52
4.16 执行命令 update-by-query 报错 “Trying to create too many scroll contexts”	52
4.17 Elasticsearch 集群无法创建 pattern	53
5 端口访问类.....	54
5.1 9200 端口访问失败	54
6 修订记录.....	56

1 访问集群类

1.1 无法正常打开 Kibana

问题现象

Es-event 集群单击进入 kibana 后，会出现一直卡在加载页面中，不能进入 Kibana 控制台。

原因分析

浏览器缓存导致，清理缓存。

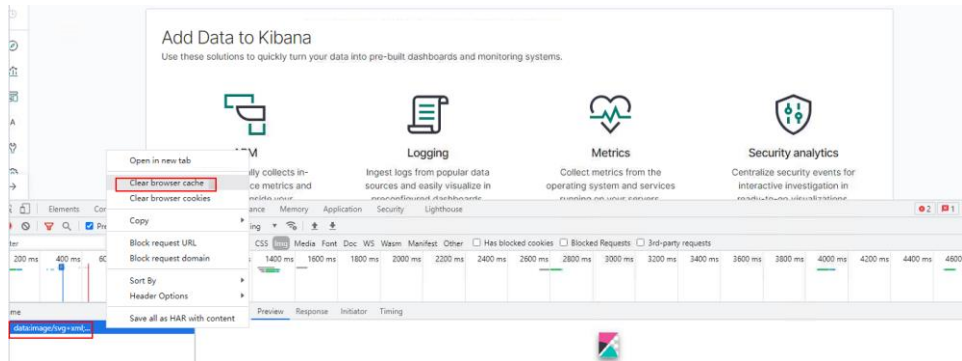
处理步骤

1. 登录云搜索服务管理控制台。
2. 在左侧导航栏，单击“集群管理”。
3. 在集群对应的“操作”列，单击“Kibana”，打开 Kibana 界面。

说明

- 如果您开启了安全模式，登录时候需要输入用户名和密码。用户名默认为 admin，密码为创建集群时设置的密码。
 - 如果忘记密码，可以在集群详情页面的“配置信息”区域，单击“重置密码”后的“重置”，设置并确认新的管理员密码。
4. 在 kibana 页面按 F12。
 5. 单击“Network”，选中“data:image”，右键选择“clear browser cache”，弹出对话框，单击确定，关闭 Kibana 界面。

图1-1 关闭 Kibana 界面



6. 在集群对应的“操作”列，单击“Kibana”即可访问。

1.2 Elasticsearch 针对 filebeat 配置调优

问题现象

filebeat 是性能非常出色的文件采集工具，绝大多数的业务日志可以很容易的在 1 秒内收集至 elasticsearch 内，但是个别日志量大的业务日志无法及时收集，按照官方的默认配置通常 1 核 CPU 分配给 filebeat 时，写 ES 的速率低于 1M/S，这里可以针对 filebeat.yml 配置文件做优化，提高写入 ES 的性能。

原因分析

filebeat.yml 的默认配置比较保守，在日志量很大的业务场景，需要修改 filebeat.yml 参数进行调优。

处理步骤

1. 针对 filebeat.yml 配置文件做参数优化，调整 input 端配置：
#根据实际情况调大 harvester_buffer_size 参数（该参数是指每个 harvester 监控文件时，使用的 buffer 大小）。
harvester_buffer_size:40960000
#根据实际情况调大 filebeat.spool_size 参数（该参数是指 spooler 的大小，一次 Publish，上传多少行的日志量）。
filebeat.spool_size:250000
#根据实际情况调整 filebeat.idle_timeout 参数（该参数是指 spooler 的超时时间，如果到了超时时间，不论是否到达容量阈值，spooler 会清空发送出去）。
filebeat.idle_timeout:1s
2. 针对 filebeat.yml 配置文件做参数优化，调整 output.elasticsearch 端配置：
#根据实际情况将 worker 参数调整为跟 ES 个数一致（该参数是指对应 ES 的个数，默认 1）。
worker:1

#根据实际情况调大 `bulk_max_size` 参数（该参数是指单个 `elasticsearch` 批量 API 索引请求的最大事件数，默认是 50）。

```
bulk_max_size:15000
```

#根据实际情况调整 `flush_interval` 参数（该参数是指新事件两个批量 API 索引请求之间需要等待的秒数，如果 `bulk_max_size` 在该值之前到达，额外的批量索引请求生效）。

```
flush_interval:1s
```

1.3 TransportClient 客户端连接 css 报错

用户问题

采用 `spring data elasticsearch` 方式去连接 `css` 不通，报错：`None of the configured nodes are available`。

问题现象

采用 `spring data elasticsearch` 方式去连接 `css` 报错。

原因分析

通常 `TransportClient` 连接集群时，需要配置 `cluster.name` 信息。出错原因可能是使用 `Settings.EMPTY` 选项，或者使用了错误的 `Setting` 配置，导致连接集群失败。

处理步骤

客户端接入集群，请参见。

1.4 Spring Boot 使用 Elasticsearch 出现 Connection reset by peer 问题

问题现象

`Spring Boot` 服务使用 `Elasticsearch RestHighLevelClient` 链接 `Elasticsearch` 运行一段时间就会出现 `Connection reset by peer`，`TCP` 连接中断，业务数据写入失败。

原因分析

连接关闭有很多原因，是 `Elasticsearch` 服务器端不能完全控制的。例如，有可能关闭了连接，有可能有防火墙，交换机，`VPN` 等，也有可能是 `keepalive` 设置问题，更换了连接服务器节点，网络不稳定等。

处理步骤

- 方法一
修改 RestHighLevelClient 连接请求的超时时间，默认 1000ms 可以尝试增加到 10000ms。

```
RestClientBuilder builder = RestClient.builder(new HttpHost(endpoint, port))
    .setHttpClientConfigCallback(httpClientBuilder->
httpClientBuilder.setDefaultCredentialsProvider(credentialsProvider))
    .setRequestConfigCallback(requestConfigBuilder ->
requestConfigBuilder.setConnectTimeout(10000).setSocketTimeout(60000));
return new RestHighLevelClient(builder);
```

单个请求修改：`request.timeout(TimeValue.timeValueSeconds(60))`。

- 方法二
设置 RestHighLevelClient keepalive 时间，建议设置到 15 分钟。
- 方法三
代码层面捕获异常，重试请求。

参考

- TCP 长连接和短连接
TCP 协议中有长连接和短连接之分。短连接在数据包发送完成后会自己断开，长连接在发包完成后，会在一定的时间内保持连接，即通常所说的 Keepalive（存活定时器）功能。
- TCP 保活机制
保活机制是由一个保活计时器实现的。当计时器被激发，连接一段将发送一个保活探测报文，另一端接收报文的会发送一个 ACK 作为响应。如果客户端无响应，服务器将中断连接，否则会重置保活计时器。
服务器端 Keepalive 设置时间 30m，Linux 有三个参数可以控制保活时间：
`tcp_keepalive_time`（开启 Keepalive 的闲置时长）、`tcp_keepalive_intvl`（Keepalive 探测包的发送间隔）、`tcp_keepalive_probes`（如果对方不予应答，探测包的发送次数）。
- http-keepalive
`http-keepalive` 是保证一个 TCP 连接尽可能传递多的报文，每次交互一个报文后就会更新 `http-keepalive` 时间。如果 `http-keepalive` 时间超时，意味这个这段时间 `client` 和 `server` 没有报文交互，本端会主动关闭释放连接。
`tcp-keepalive` 是一种探测 TCP 连接状态的保活机制，TCP 连接建立后如果不主动关闭，`client` 和 `server` 没有发生异常，这个连接理论上是一直存在的，`http-keepalive` 是保证一个 TCP 连接上尽可能传递更多的报文，如果 `http-keepalive` 时间内没有报文交互则会主动关闭连接。

1.5 为什么集群创建失败

集群创建失败原因有如下 3 种：

- 资源配额不足，无法创建集群。建议申请足够的资源配额。

- 如果集群配置信息中，“安全组”的“端口范围/ICMP 类型”不包含“9200”端口，导致集群创建失败。请修改安全组信息或选择其他可用安全组。
- 7.6.2 以及 7.6.2 之后的版本，集群内通信端口 9300 默认开放在用户 VPC 的子网上面。创建集群时需要确认所选安全组是否放通子网内的 9300 通信端口，如果未放通，请修改安全组信息或选择其他可用安全组。
- 权限不足导致集群创建失败。建议参考[权限管理](#)获取权限，然后创建集群。

1.6 Elasticsearch 集群出现写入拒绝 “Bulk Reject”，如何解决？

问题现象

集群在某些情况下会出现写入拒绝率增大“bulk reject”的现象，具体表现为 bulk 写入时，会有类似以下报错：

```
[2019-03-01 10:09:58][ERROR]rspItemError: {
  "reason": "rejected execution of
org.elasticsearch.transport.TransportService$7@5436e129 on
EsThreadPoolExecutor[bulk, queue capacity = 1024,
org.elasticsearch.common.util.concurrent.EsThreadPoolExecutor@6bd77359[Running,
pool size = 12, active threads = 12, queued tasks = 2390, completed tasks =
20018208656]]",
  "type": "es_rejected_execution_exception"
}
```

问题分析

引起 bulk reject 的大多原因是 shard 容量过大或 shard 分配不均，具体可通过以下方法进行定位分析。

1. 检查分片（shard）数据量是否过大。

单个分片数据量过大，可能引起 Bulk Reject，建议单个分片大小控制在 20GB - 50GB 左右。可在 kibana 控制台，通过如下命令查看索引各个分片的大小。

```
GET _cat/shards?index=index_name&v
```

2. 检查分片数是否分布不均匀。

提供如下两种方式查看：

- a. 通过 CSS 控制台集群详情页的“集群监控”中的“节点状态”查看，具体操作可参见[查看监控指标](#)。
- b. 通过 CURL 客户端，查看集群各个节点的分片个数。

```
curl "$p:$port/_cat/shards?index={index_name}&s=node,store:desc" | awk
'{{print $8}}' | sort | uniq -c | sort
```

结果如下图所示：


```
100 5763 100 5763 0 0 26084 0 --:--:
1 1528894127000000711
1 1536026850017169311
1 1536026850017169611
2 1528894127000000511
2 1532573921106167111
2 1532573921106167511
2 1532573921106167611
3 1532573921106167211
4 1528894127000000611
4 1532573921106167311
5 1536026850017169211
5 1536026850017169511
8 1536026850017169111
8 1536026850017169411
```

说明

第一列为分片个数，第二列为节点 ID，有的节点分片为 1，有的为 8，分布极不均匀。

解决方案

- 如果问题是由分片数据量过大导致。
分片大小可以通过 index 模版下的 “number_of_shards” 参数进行配置。

说明

模板创建完成后，再次新创建索引时生效，旧的索引不能调整。

- 如果问题是由分片数分布不均匀导致。
临时解决方案：
 - 可以通过如下命令设置 “routing.allocation.total_shards_per_node” 参数，动态调整某个 index 解决。

```
PUT <index_name>/_settings
{
  "settings": {
    "index": {
      "routing": {
        "allocation": {
          "total_shards_per_node": "3"
        }
      }
    }
  }
}
```

说明

“total_shards_per_node” 要留有一定的 buffer，防止机器故障导致分片无法分配（例如 10 台机器，索引有 20 个分片，则 total_shards_per_node 设置要大于 2，可以取 3）。

- 索引产生前设置。
通过索引模板，设置其在每个节点上的分片个数。

```
PUT _template/<template_name>
{
  "order": 0,
```

```
"template": " {index_prefix@} *", //要调整的 index 前缀
"settings": {
  "index": {
    "number_of_shards": "30", //指定 index 分配的 shard 数, 可以根据一个
    shard 30GB 左右的空间来分配
    "routing.allocation.total_shards_per_node": 3 //指定一个节点最多容纳
    的 shards 数
  }
},
"aliases": {}
}
```

1.7 Elasticsearch 集群创建 index pattern 卡住

问题现象

在 Kibana 的“Dev Tools”页面，执行 `GET .kibana/_settings` 命令查询 kibana 索引的状态，如果状态为“true”，说明集群磁盘过高。

```
1 {
2   ".kibana_1" : {
3     "settings" : {
4       "index" : {
5         "number_of_shards" : "1",
6         "auto_expand_replicas" : "0-1",
7         "blocks" : {
8           "read_only_allow_delete" : "true"
9         },
10        "provided_name" : ".kibana_1",
11        "max_result_window" : "100000",
12        "creation_date" : "1602490470096",
13        "number_of_replicas" : "0",
14        "uuid" : "_I3td16IRt6605J1tAbKQ",
15        "version" : {
16          "created" : "7060299"
17        }
18      }
19    }
20  }
21 }
22 }
```

问题原因

索引置为只读状态导致。

解决方案

1. 将 kibana 索引只读状态改为“false”后再执行创建 **index pattern**。
2. 在 kibana 的“Dev Tools”页面，执行如下命令：

```
PUT .kibana/_settings
{
  "index": {
```

```
"blocks": {  
  "read_only_allow_delete": false  
}
```

1.8 云搜索控制台页面提示系统繁忙

问题描述

登录云搜索控制台，单击集群列表，显示“系统繁忙，请稍后重试或拨打客服电话4000-955-988”和“当前策略不允许 css:cluster:list”。

问题原因

该问题是由于此帐号没有云搜索服务的读权限导致的，需要主帐号赋予此帐号需要使用功能的 IAM 权限。

解决方案

为 IAM 帐号进行权限授权，更多信息请参见。

1.9 Elasticsearch 集群报错：unassigned shards all indices

问题描述

Elasticsearch 集群报错 unassigned shards all indices，集群状态为 red。

原因分析

当前集群存在未分配的 shard。

解决方案

在 Kibana 的“Dev Tools”页面，执行如下命令：

```
POST /_cluster/reroute?retry_failed=true
```

1.10 es-head 插件连接 Elasticsearch 集群报跨域错误

解决方案

1. 在安装 es-head 的云主机上测试网络是否连通。
2. 网络连通后，登录云搜索服务管理控制台。

3. 在“集群管理”页面，单击需要修改参数配置的集群名称，进入集群基本信息页面。
4. 选择“参数配置”，单击“编辑”，将“http.cors.allow-origin”参数更改为“*”。

1.11 单节点集群打开 Cerebro 界面显示告警

原因分析

单节点集群索引默认有副本，但是副本无法下发请求，所以显示告警。

解决方案

在 Kibana 的“Dev Tools”页面，执行以下命令将索引副本数量修改为“0”。

```
PUT _all/_settings
{
  "index" :
  {
    "number_of_replicas" : 0
  }
}
```

2 集群不可用

2.1 集群不可用排查指导

问题现象

云搜索服务的集群列表中，“集群状态”出现“不可用”。

图2-1 集群不可用

名称/ID	集群状态	任务状态
CSS- fb8d95ca-d38b-...	 不可用	--

原因分析及处理方法

- 若集群列表的任务状态显示“冻结”，可能是 2.2 集群冻结状态导致集群不可用。
- 若集群列表的任务状态显示“配置错误，重启失败”，可能是 2.3 X-pack 参数配置导致集群不可用。
- 若集群节点的日志内容存在警告“master not discovered or elected yet, an election requires at least 2 nodes with ids [xxx, xxx, xxx, ...], have discovered [xxx...] which is not a quorum”，可能是 2.4 安全组策略设置不合理导致集群不可用。
- 若集群节点的日志内容存在明显的关于插件的报错“fatal error in thread [main], exitingjava.lang. NoClassDefFoundError: xxx/xxx/.../xxxPlugin at ...”，可能是 2.5 插件不兼容导致集群不可用。
- 若集群的健康状态为红色和且“unassigned shards”不为 0，表示集群存在无法分配的索引分片，是 2.6 分片未正常分配导致集群不可用。
- 若集群进行备份恢复或集群迁移操作后，出现的不可用现象，可能是 2.7 数据类型不兼容导致集群不可用。

- 若集群节点的日志内容存在报错“OutOfMemoryError”和警告“[gc][xxxxx] overhead spent [x.xs] collecting in the last [x.xs]”，可能是 2.8 集群负载过高导致集群不可用。

原因分析

云搜索服务界面上的集群不可用由后端主动上报，可能出现的原因：

- 集群异常、发生故障
- 集群状态为 Red

处理步骤

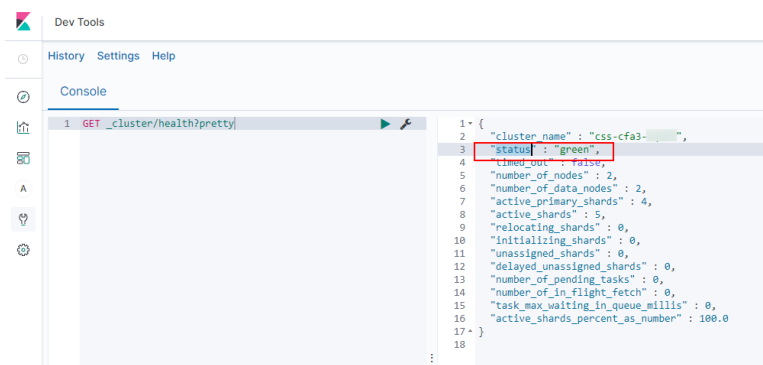
当集群状态显示“不可用”时，根据 Kibana 能否正常登录选择不同的解决方案。

能够正常登录 Kibana

1. 选择“不可用”集群，单击操作列“Kibana”，登录 Kibana 界面。
2. 单击左侧导航栏的“Dev Tools”，
3. 在 Dev Tools 中执行如下命令，查看集群的后台状态。

```
GET _cluster/health?pretty
```

图2-2 查看 status 值



Elasticsearch 集群有如下三种后台状态：

- 若“status”的值为“Green”，表示集群状态正常。
集群状态是一分钟检测一次，当后台查询到的集群状态正常，前台界面上显示“不可用”时，可能由于时间误差导致前台状态还未更新，请等待几分钟再查看界面上的集群状态是否正常。如若还是显示“不可用”请联系技术支持处理。
- 若“status”的值为“Yellow”，表示集群的副分片存在异常。
确认分片信息：“initializing_shards”表示初始化中的分片有多少；
“unassigned_shards”表示未分配的分片有多少。
- 若“status”的值为“Red”，表示主分片存在异常。

确认分片信息：“initializing_shards”表示初始化中的分片有多少；
“unassigned_shards”表示未分配的分片有多少。

图2-3 查看分片信息

```
1 {
2   "cluster_name" : "css-cfa3-cqian",
3   "status" : "green",
4   "timed_out" : false,
5   "number_of_nodes" : 2,
6   "number_of_data_nodes" : 2,
7   "active_primary_shards" : 4,
8   "active_shards" : 5,
9   "relocating_shards" : 0,
10  "initializing_shards" : 0,
11  "unassigned_shards" : 0,
12  "delayed_unassigned_shards" : 0,
13  "number_of_pending_tasks" : 0,
14  "number_of_in_flight_fetch" : 0,
15  "task_max_waiting_in_queue_millis" : 0,
16  "active_shards_percent_as_number" : 100.0
17 }
```

4. 存在初始化中的分片，可能是 translog 太大导致的。当主分片启动的时候，会主动加载文件夹下的 translog，如果 translog 文件很大，这个过程会持续很久。请耐心等待 10 分钟左右，查看集群状态是否恢复。如若还是显示“不可用”请联系技术支持处理。
5. 当存在未分配的分片时，执行如下步骤尝试恢复。
 - a. 执行如下命令，查看未分配原因。

```
GET /_cluster/allocation/explain?pretty
```

未分配的常见原因：

- **INDEX_CREATED**：由于创建索引的 API 导致未分配。这种情况下，查看磁盘是否已经达到 85% 以上使用率，如果超过这个值，ES 不会再往节点分配新的 shard。建议清理无用的数据释放磁盘空间。
- **CLUSTER_RECOVERED**：由于完全集群恢复导致未分配。
- **INDEX_REOPENED**：由于打开 open 或关闭 close 一个索引导致未分配。
- **DANGLING_INDEX_IMPORTED**：由于导入 dangling 索引的结果导致未分配。
- **NEW_INDEX_RESTORED**：由于恢复到新索引导致未分配。
- **EXISTING_INDEX_RESTORED**：由于恢复到已关闭的索引导致未分配。
- **REPLICA_ADDED**：由于显式添加副本分片导致未分配。
- **ALLOCATION_FAILED**：由于分片分配失败导致未分配。
- **NODE_LEFT**：由于承载该分片的节点离开集群导致未分配。
- **REINITIALIZED**：由于当分片从开始移动到初始化时导致未分配（例如，使用影子 shadow 副本分片）。

- **REROUTE_CANCELLED** : 作为显式取消重新路由命令的结果取消分配。
 - **REALLOCATED_REPLICA**: 确定更好的副本位置被标定使用, 导致现有的副本分配被取消, 出现未分配。
- b. 执行如下命令, 重新分配分片。

```
POST /_cluster/reroute?retry_failed=true
```

等待 15 分钟, 如果集群状态恢复正常, 则故障修复完成; 如果集群状态依旧“不可用”, 执行下一步。

- c. 重新分配分片无效, 可能是分片已损坏, 导致无法启动。可以执行如下命令, 给分片分配一个空的 **Shard**。

```
POST _cluster/reroute
{
  "commands": [
    {
      "allocate_empty_primary": {
        "index": "index-test", //索引名称
        "shard": 13, //索引编号
        "node": "css-test -ess-esn-11-1", //节点名称
        "accept_data_loss": true
      }
    }
  ]
}
```

等待 15 分钟, 如果集群状态恢复正常, 则故障修复完成; 如果集群状态依旧“不可用”, 则联系技术支持处理。

无法正常登录 Kibana

云搜索服务在节点故障后, 会启动节点的守护进程尝试修复, 在修复失败后才上报不可用。

引起修复失败的主要是以下几类故障:

1. 节点之间的网络故障, 相互 ping 不通 (eth1 和 eth1, eth2 和 eth2)。
请检查确认节点间的网络。
2. 集群压力大, 节点掉线频繁。
单击集群操作列的“监控信息”查看集群当前以及之前的 cpu、内存、load 的变化趋势, 是否有突增变化或者长期处于高位。突增的变化可能是由于突增流量进入集群导致; 流量的变化趋势可以查看 http 连接数的监控。处于高位的 load, cpu, 内存均有可能导致节点掉线。
3. 集群 shard 数太多 (超过 5w+), 导致集群始终无法启动。shard 启动的时候, 需要把 shard 相关的元数据加载到内存, shard 数太多会占用较高的内存。同时, 如果有节点掉线或者新索引创建, master 在计算 shard 分配上面也会因为 shard 数过多耗费更多的资源, 增大集群的压力。

2.2 集群冻结状态导致集群不可用

问题现象

“集群状态”为“不可用”，集群的“任务状态”为“冻结”。

图2-4 集群冻结状态

名称/ID	集群状态	任务状态
css-08c	不可用	冻结
css-2a5	不可用	冻结

原因分析

集群出现冻结状态的原因是账户欠费。

处理步骤

- **按需计费集群**
 - a. 在控制台上方单击“费用与成本”进入费用中心。
 - b. 在“总览”页面查看账户的欠费情况。
 - 如果是 IAM 用户，请登录 IAM 帐号对应的帐号进行费用充值。
 - 如果是帐号用户，请直接单击“充值”前往费用充值页面。
 - c. 确认账户有现金预算后，等待集群自动恢复可用状态。

2.3 X-pack 参数配置导致集群不可用

问题现象

“集群状态”为“不可用”，集群的“任务状态”为“配置错误，重启失败”。

图2-5 集群配置错误

名称/ID	集群状态	任务状态
css-95f3-2a57e9	不可用	配置错误, 重启失败

原因分析

集群可能配置了 X-pack 相关的自定义参数导致集群不可用。CSS 服务不支持 X-pack 功能。

处理步骤

1. 在集群管理页面，单击不可用的集群名称，进入集群基本信息页面。
2. 选择“参数配置”，展开“自定义”模块，查看是否存在 X-pack 相关的自定义参数。

说明

X-pack 相关的自定义参数示例：

- “xpack.security.enabled” : “true”
- “xpack.security.http.ssl.enabled” : “true”
- “xpack.security.http.ssl.keystore.path” : “http.p12”

- 是，则执行下一步。
- 否，联系技术支持定位参数配置问题。

3. 删除 X-pack 相关的自定义参数。
 - a. 单击“编辑”，在需要删除的参数右侧单击“删除”。
 - b. 单击“提交”，在“提交配置”弹窗中，勾选“参数修改后需要手动重启才能生效”，单击“确定”。

当下方的参数修改列表显示“作业状态”为“成功”时，表示修改保存成功。
 - c. 返回集群列表，单击集群操作列的“更多 > 重启”，重启集群使修改的配置生效。
 - d. 集群重启成功后，集群恢复可用。

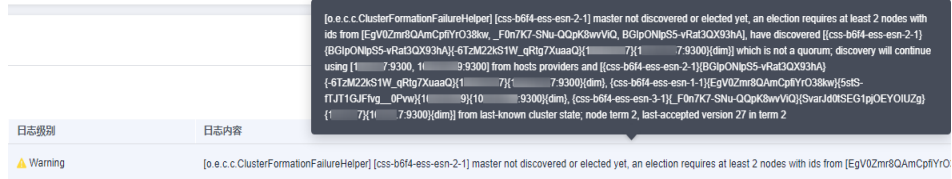
2.4 安全组策略设置不合理导致集群不可用

问题现象

“集群状态”为“不可用”。

单击集群名称进入集群基本信息页面，选择“日志管理”，单击“日志查询”页签，可见日志内容存在警告“master not discovered or elected yet, an election requires at least 2 nodes with ids [xxx, xxx, xxx, ...], have discovered [xxx...] which is not a quorum”。

图2-6 节点报错日志示例



原因分析

出现以上报错日志表示集群各节点之间无法通信，导致集群无法进行选主，可能原因是集群当前所选安全组未放通 9300 端口。

说明

云搜索服务在 7.6.2 及以上的版本，集群内通信端口 9300 默认开放在用户 VPC 的子网上。集群所选安全组需要放通子网内的 9300 通信端口才能保证节点之间通信。

处理步骤

1. 在集群管理页面，单击不可用的集群名称，进入集群基本信息页面。
2. 单击“配置信息”中的安全组名称，进入当前集群所选安全组的基本信息页面。
3. 分别查看“入方向规则”和“出方向规则”页签下，是否存在“策略”为“允许”，“协议端口”为“TCP:9300”，“类型”为“IPv4”的安全组规则。
 - 是，联系技术支持定位集群不可用问题。
 - 否，执行下一步。
4. 修改集群当前所选安全组信息，放通 9300 通信端口。
 - a. 在当前集群所选安全组基本信息界面，选择“入方向规则”页签。
 - b. 单击“添加规则”，在添加入方向规则对话框设置“优先级”为“100”，“策略”选择“允许”，“协议端口”选择“基本协议/自定义 TCP”，端口填写“9300”，“类型”选择“IPv4”，“源地址”选择“安全组”下的集群当前安全组名称，即同安全组内放通。

图2-7 添加安全组规则

添加方向规则 [教我设置](#)

安全组规则对不同规格的云服务器生效情况不同，为了避免您的安全组规则不生效，请查看安全组规则限制。

安全组 default

如您要添加多条规则，建议单击 [导入规则](#) 以进行批量导入。

优先级	策略	协议端口	类型	源地址	描述	操作
100	允许	基本协议/自定义TCP 9300	IPv4	安全组 default(b1038649-1f77-4ae9-b1		复制 删除

[+ 增加1条规则](#)

[确定](#) [取消](#)

- c. 单击“确定”即可完成放通 9300 端口的设置。
 - d. 同样的步骤，在“出方向规则”页签添加放通 9300 端口的设置。
5. 安全组放通 9300 端口后，等待集群自动恢复可用状态。

2.5 插件不兼容导致集群不可用

问题现象

安装自定义插件后重启集群，“集群状态”变为“不可用”。

单击集群名称进入集群基本信息页面，选择“日志管理”，单击“日志查询”页签，可见日志内容存在明显的关于插件的报错“fatal error in thread [main], exitingjava.lang.NoClassDefFoundError: xxx/xxx/.../xxxPlugin at ...”。

图2-8 节点报错日志示例



说明

CSS 服务已下线自定义插件功能，但历史版本的集群可能还装有自定义插件，只有这类集群可能出现该故障。

原因分析

可能是安装的自定义插件与 CSS 集群版本不兼容，导致 Elasticsearch 进程无法正常启动。

处理步骤

1. 在集群管理页面，单击不可用的集群名称，进入集群基本信息页面。
2. 选择“插件管理”，单击“自定义插件列表”页签。确认是否继续使用该自定义插件。
 - 是，删除插件后重新安装。
 - i. 在自定义插件操作列表，卸载并删除插件。
 - ii. 根据节点日志的报错信息解决插件存在的问题，无法自行解决时可联系技术支持。
 - iii. 插件问题解决后，在自定义插件操作列表，上传并安装插件。当“插件状态”为“已安装，待重启集群后生效”时，表示插件安装成功。
 - 否，删除插件。

在自定义插件操作列表，卸载并删除插件。
3. 返回集群列表，单击集群操作列的“更多 > 重启”，集群重启成功后，集群恢复可用。

2.6 分片未正常分配导致集群不可用

问题现象

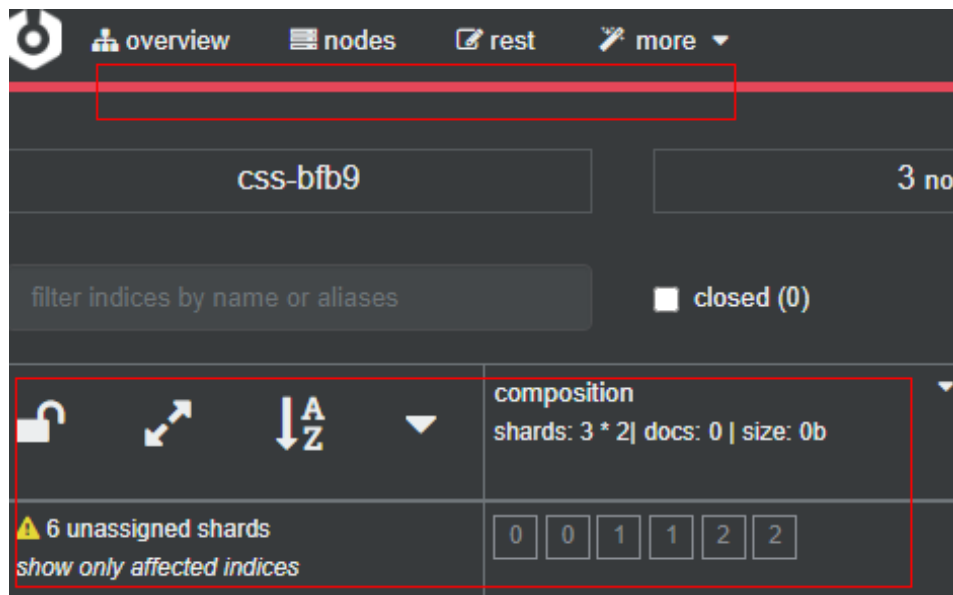
“集群状态”为“不可用”。

在 Kibana 的“Dev Tools”页面，执行命令 `GET _cluster/health` 查看集群健康状态，结果中“status”为“red”，“unassigned_shards”不为 0。或者在“Cerebro”可视化页面，单击“overview”查看索引分片在各数据节点的分配情况，可见集群状态为红色和“unassigned shards”不为 0，表示集群存在无法分配的索引分片。

图2-9 集群健康状态

```
1 {
2   "cluster_name" : "css-bfb9",
3   "status" : "red",
4   "timed_out" : false,
5   "number_of_nodes" : 3,
6   "number_of_data_nodes" : 3,
7   "active_primary_shards" : 19,
8   "active_shards" : 38,
9   "relocating_shards" : 0,
10  "initializing_shards" : 0,
11  "unassigned_shards" : 6,
12  "delayed_unassigned_shards" : 0,
13  "number_of_pending_tasks" : 0,
14  "number_of_in_flight_fetch" : 0,
15  "task_max_waiting_in_queue_millis" : 0,
16  "active_shards_percent_as_number" : 86.36363636363636
17 }
```

图2-10 cerebro 可视化界面



原因分析

集群出现不可用状态的原因是集群有索引分片未正常分配。

处理步骤

步骤一：确认集群不可用原因

1. 通过 Kibana 接入故障集群，在 Kibana 的“Dev Tools”页面，执行命令 **GET /_recovery?active_only=true** 查看集群是否在进行副本恢复：
 - 返回 “{“index_name”:{“shards”:[{“id”:25,“type”：“...””，代表集群存在正在进行副本恢复的索引。等待副本恢复完毕，如果集群状态仍为“不可用”，则执行下一步。
 - 返回 “{ }”：代表集群未进行副本恢复，则执行下一步。
2. 执行命令 **GET _cluster/allocation/explain?pretty** 查看索引分片未分配的原因，根据返回信息进行筛选。

表2-1 参数说明

参数	描述
index	索引名称
shard	分片标号
current_state	分片当前状态
allocate_explanation	分片分配解释
explanation	解释说明

表2-2 不同故障说明

现象	原因	处理步骤
“explanation”中存在“no allocations are allowed due to cluster setting [cluster.routing.allocation.enable=none]”	集群当前设置的 allocation 策略禁止所有分片分配。	参考“shard allocation 策略配置错误”之 cluster.routing.allocat...
“explanation”中存在“too many shards [3] allocated to this node for index [write08]index setting [index.routing.allocation.total_shards_per_node=3]”	集群当前设置的单个索引的分片允许分配给每个数据节点的分片数值过小，不满足索引分片的分配要求。	参考“shard allocation 策略配置错误”之 index.routing.allocatio...
“explanation”中存在“too many shards [31] allocated to this node, cluster setting [cluster.routing.allocation.total_shards_per_node=30]”	集群当前设置的集群所有索引分片允许分配给每个数据节点的分片数值太小。	参考“shard allocation 策略配置错误”之 cluster.routing.allocat...
“explanation”中存在“node does not match index setting [index.routing.allocation.include] filters [box_type:“hot”]”	索引分片需要下发到标记为“hot”的数据节点，而集群中所有数据节点都没有打这个标记时，分片	参考“shard allocation 策略配置错误”之 index.routing.allocatio...

现象	原因	处理步骤
	无法下发。	
“explanation”中存在“node does not match index setting [index.routing.allocation.require] filters [box_type:"xl"]”	索引分片需要下发到特定标记的数据节点，而集群中所有节点都没有打这个标记时，分片便无法下发。	参考“shard allocation 策略配置错误”之 index.routing.allocatio...
“explanation”中存在“[failed to obtain in-memory shard lock]”	这种情况一般出现在有节点短暂离开集群，然后马上重新加入，并且有线程正在对某个 shard 做 bulk 或者 scroll 等长时间的写入操作，等节点重新加入集群的时候，由于 shard lock 没有释放，master 无法 allocate 这个 shard。	参考 shardlock 错误
“explanation”中存在“node does not match index setting [index.routing.allocation.include] filters [_tier_preference:"data_hot OR data_warm OR data_cold"]”	集群的某个索引设置的参数与版本不匹配。	参考 索引参数版本不匹配
“explanation”中存在“cannot allocate because all found copies of the shard are either stale or corrupt”	集群的索引分片数据被损坏。	参考“主分片数据损坏” 主分片数据损坏
“explanation”中存在“the node is above the high watermark cluster setting [cluster.routing.allocation.disk.watermark.high=90%], using more disk space than the maximum allowed [90.0%], actual free: [6.976380997419324%]”	节点的磁盘使用率已超过磁盘空间允许的最大值。	参考“磁盘使用率过高” 磁盘使用率过高

步骤二：根据不同的问题现象处理故障

- **shard allocation 策略配置错误**

- cluster.routing.allocation.enable 参数

- i. 返回结果中“explanation”如下，表示集群当前设置的 allocation 策略禁止所有分片分配导致。

图2-11 allocation.enable 参数配置错误

```
"deciders" : [
  {
    "decider" : "enable",
    "decision" : "NO",
    "explanation" : "no allocations are allowed due to cluster setting
    [cluster.routing.allocation.enable=none]"
  }
]
```

- ii. 在 Kibana 的“Dev Tools”页面，执行命令将“enable”设置为“all”，允许所有分片进行分配。

```
PUT _cluster/settings
{
  "persistent": {
    "cluster": {
      "routing": {
        "allocation.enable": "all"
      }
    }
  }
}
```

📖 说明

index 级别会覆盖 cluster 级别配置，参数设置含义如下：

- **all** - (默认) 所有类型均允许 allocation。
- **primaries** - 只允许 allocation 主分片。
- **new_primaries** - 只允许 allocation 新创建 index 的主分片。
- **none** - 所有的分片都不允许 allocation。

- iii. 再执行命令 **POST _cluster/reroute?retry_failed=true** 手动进行分片分配，等待索引分片分配完成，集群状态变为可用。
- **index.routing.allocation.total_shards_per_node** 参数。
- i. 返回结果中“explanation”如下，表示设置的“index.routing.allocation.total_shards_per_node”值过小，不满足索引的分片分配要求。

图2-12 index total_shards_per_node 设置错误

```
"deciders" : [
  {
    "decider" : "shards_limit",
    "decision" : "NO",
    "explanation" : "too many shards [3] allocated to this node for index [write08],
    index setting [index.routing.allocation.total_shards_per_node=3]"
  }
]
```

- ii. 在 Kibana 的“Dev Tools”页面，执行命令修改索引在每个节点允许分配的分片数。

```
PUT index_name/_settings
{
  "index": {
    "routing": {
      "allocation.total_shards_per_node": 3
    }
  }
}
```

说明

“index.routing.allocation.total_shards_per_node” 的值 = index_name 索引分片数 / (数据节点个数 - 1)

参数值应设置稍微大一些，假设集群有 10 个节点，其中 5 个数据节点，2 个 client 节点，3 个 master 节点，有个索引的分片数为 30，如果将 total_shards_per_node 值设为 4，能分配的 shard 总数只有 4*5=20，分片无法完全分配。5 个数据节点，需要分配 30 个分片，每个节点应最少分配 6 个分片，防止某数据节点故障脱离，那最少应设置每个节点允许分配 8 个分片。

- iii. 再执行命令 **POST _cluster/reroute?retry_failed=true** 手动进行分片分配，等待索引分片分配完成，集群状态变为可用。
- cluster.routing.allocation.total_shards_per_node 参数。
 - i. 返回结果中“explanation”如下，表示集群允许分配给每个数据节点的分片数设置太小。

图2-13 cluster total_shards_per_node 设置错误

```
"deciders" : [
  {
    "decider" : "shards_limit",
    "decision" : "NO",
    "explanation" : "too many shards [31] allocated to this node, cluster setting [cluster.routing.allocation.total_shards_per_node=30]"
  }
]
```

- ii. “cluster.routing.allocation.total_shards_per_node” 参数为限制集群每个数据节点可分配的分片数量，此参数默认设置为“1000”，在 Kibana 的“Dev Tools”页面执行如下命令设置“cluster.routing.allocation.total_shards_per_node”参数。

```
PUT _cluster/settings
{
  "persistent": {
    "cluster": {
      "routing": {
        "allocation.total_shards_per_node": 1000
      }
    }
  }
}
```

- iii. 出现此场景大多数是参数使用错误，误将“index.routing.allocation.total_shards_per_node”参数设置为“cluster.routing.allocation.total_shards_per_node”参数。执行如下命令可以设置“index.routing.allocation.total_shards_per_node”参数：

```
PUT index_name/_settings
{
  "index": {
    "routing": {
      "allocation.total_shards_per_node": 30
    }
  }
}
```

说明

两个参数都是限制单个数据节点所能分配的最大分片数。

- “cluster.routing.allocation.total_shards_per_node” 是**集群**级别的分片限制。
- “index.routing.allocation.total_shards_per_node” 是**索引**级别的分片限制。
 - iv. 再执行命令 **POST _cluster/reroute?retry_failed=true** 手动进行分片分配，等待索引分片分配完成，集群状态变为可用。
- index.routing.allocation.include 参数。
 - i. 返回结果中“explanation”如下，表示是将索引分片下发到标记为“hot”的数据节点，而集群中所有数据节点都没有打这个标记时，分片无法下发。

图2-14 include 参数配置错误

```
"deciders" : [
  {
    "decider" : "filter",
    "decision" : "NO",
    "explanation" : ""node does not match index setting [index.routing.allocation.include] filters [box_type:"hot"]""
  }
]
```

- ii. 在 Kibana 的“Dev Tools”页面执行命令取消该配置：

```
PUT index_name/_settings
{
  "index.routing.allocation.include.box_type": null
}
```

- iii. 再执行命令 **POST _cluster/reroute?retry_failed=true** 手动进行分片分配，等待索引分片分配完成，集群状态变为可用。
- index.routing.allocation.require 参数。
 - i. 返回结果中“explanation”如下，表示是将分片下发到特定标记的数据节点，而集群中所有节点都没有打这个标记时，分片便无法下发。

图2-15 require 参数配置错误

```
"deciders" : [
  {
    "decider" : "filter",
    "decision" : "NO",
    "explanation" : ""node does not match index setting [index.routing.allocation.require]
filters [box_type:"xl"]""
  },
]
```

- ii. 在 Kibana 的 “Dev Tools” 页面执行命令取消该配置:

```
PUT index_name/_settings
{
  "index.routing.allocation.require.box_type": null
}
```

- iii. 再执行命令 **POST _cluster/reroute?retry_failed=true** 手动进行分片分配，等待索引分片分配完成，集群状态变为可用。

- **shard lock 错误**

- a. 返回结果中 “explanation” 存在 “[failed to obtain in-memory shard lock]”，这种情况一般出现在有节点短暂离开集群，然后马上重新加入，并且有线程正在对某个 shard 做 bulk 或者 scroll 等长时间的写入操作，等节点重新加入集群的时候，由于 shard lock 没有释放，master 无法 allocate 这个 shard。
- b. 此现象不会造成分片数据丢失，只需要重新触发一下分配即可。在 Kibana 的 “Dev Tools” 页面执行命令 **POST /_cluster/reroute?retry_failed=true** 手动对未分配分片进行分配，等待索引分片分配完成，集群状态变为可用。

- **索引参数版本不匹配**

- a. 返回信息中的索引名称 “index” 和索引未分配解释 “explanation ”：“node does not match index setting [index.routing.allocation.include] filters [_tier_preference:"data_hot OR data_warm OR data_cold"]”，表示集群的某个索引设置的参数与节点不匹配。

图2-16 索引参数不匹配

```
{
  "index" : "write710",
  "shard" : 4,
  "primary" : true,
  "current_state" : "unassigned",
  "unassigned_info" : {
    "reason" : "INDEX_CREATED",
    "at" : "2022-12-12T08:38:13.832Z",
    "last_allocation_status" : "no"
  },
  "can_allocate" : "no",
  "allocate_explanation" : "cannot allocate because allocation is not permitted to any of the nodes",
  "node_allocation_decisions" : [
    {
      "node_id" : "LFtxfyAbQCaRZcNaGtKv-g",
      "node_name" : "css-9755-...",
      "transport_address" : "1...",
      "node_decision" : "no",
      "weight_ranking" : 1,
      "deciders" : [
        {
          "decider" : "filter",
          "decision" : "NO",
          "explanation" : ""node does not match index setting [index.routing.allocation.include] filters [_tier_preference:'data_hot OR data_warm OR data_cold']""
        }
      ]
    }
  ]
}
```

- b. 执行命令 **GET index_name/_settings** 查看索引配置，返回结果中是否存在不符合自身版本的索引特性。

图2-17 索引设置

```
{
  "write710" : {
    "settings" : {
      "index" : {
        "routing" : {
          "allocation" : {
            "include" : {
              "_tier_preference" : "data_hot,data_warm,data_cold"
            }
          }
        }
      },
      "number_of_shards" : "6",
      "provided_name" : "write710",
      "creation_date" : "1670834293827",
      "number_of_replicas" : "2",
    }
  }
}
```

以 `index.routing.allocation.include_tier_preference` 特性为例，当前集群是 7.9.3 版本，这个索引特性是在 7.10 版本之后才支持的，低版本集群使用该特性将无法分配索引的分片，导致集群不可用。

- c. 确定集群是否必须使用该不匹配的特性。
- 是，创建与所需索引特性相匹配的版本集群，然后将老集群的数据通过备份恢复至新集群。
 - 否，执行下一步。

- d. 执行命令去除索引中不符合集群版本的特性。

```
PUT /index_name/_settings
{
  "index.routing.allocation.include._tier_preference": null
}
```

- e. 执行命令 **POST /_cluster/reroute?retry_failed=true** 手动对未分配分片进行分配，等待索引分片分配完成，集群状态变为可用。

• 主分片数据损坏

- a. 返回信息中的索引名称"index"、分片标号"shard"、分配解释"allocate_explanation" 和"store_exception":{"type":"corrupt index exception"}，表示集群的某个索引的某个分片数据被损坏。

图2-18 索引数据损坏

```
{
  "index": "write02",
  "shard": 2,
  "primary": true,
  "current_state": "unassigned",
  "unassigned_info": {
    "can_allocate": "no_valid_shard_copy",
    "allocate_explanation": "cannot allocate because all found copies of the shard are either stale or corrupt",
    "node_allocation_decisions": [
      {
        "node_id": "Ys7fdtHHSYa7W8Xhtz4-NA",
        "node_name": "css-9755",
        "transport_address": "10.10.10.10",
        "node_decision": "no",
        "store": {
          "in_sync": true,
          "allocation_id": "XNp_o6v0SieJyiMOC4eSig",
          "store_exception": {
            "type": "corrupt index exception",
            "reason": "Unexpected file read error while reading index. (resource=BufferedChecksumIndexInput"
          }
        }
      }
    ]
  }
}
```

- b. 当索引数据被损坏或者某个分片的主副本都丢失时，为了能使集群恢复 green 状态，解决方法是划分一个空 shard，执行以下命令划分空分片并指定分配的节点。

```
POST /_cluster/reroute
{
  "commands": [
    {
      "allocate_empty_primary": {
        "index": "index_name",
        "shard": 2,
        "node": "node_name",
        "accept_data_loss": true
      }
    }
  ]
}
```

须知

一定要谨慎该操作，会导致对应分片的数据完全清空。

- c. 索引分片重新分配后，集群状态恢复可用。
- 磁盘使用率过高
 - a. 返回结果如下，其中“allocate_explanation”表示该索引的分片无法分配给任何数据节点，“explanation”表示节点磁盘使用率已超过磁盘空间允许的最大值。

图2-19 explain 查询结果

```

1 {
2   "index": "composition",
3   "shard": 1,
4   "primary": true,
5   "current_state": "unassigned",
6   "unassigned_info": {
7     "reason": "INDEX_CREATED",
8     "at": "2022-12-08T02:12:10.768Z",
9     "last_allocation_status": "no"
10  },
11  "can_allocate": "no",
12  "allocate_explanation": "cannot allocate because allocation is not permitted to any of the nodes",
13  "node_allocation_decisions": [
14    {
15      "node_id": "npRZlfjsT4NZdHctIxtcGg",
16      "node_name": "css-bfb9",
17      "transport_address": "10.10.10.10",
18      "node_decision": "no",
19      "weight_ranking": 1,
20      "deciders": [
21        {
22          "decider": "disk_threshold",
23          "decision": "NO",
24          "explanation": "the node is above the high watermark cluster setting [cluster.routing.allocation.disk.watermark.high=90%], using more disk space than the maximum allowed [90.0%], actual free: [6.976380997419324%]"
25        }
26      ]
27    }
28  ]
29 }

```

📖 说明

- 磁盘使用率超过 85%：会导致新的分片无法分配。
- 磁盘使用率超过 90%：集群会尝试将对应节点中的分片迁移到其他磁盘使用率比较低的数据节点中。无法迁移时系统会对集群每个索引强制设置“read_only_allow_delete”属性，此时索引将无法写入数据，只能读取和删除对应索引。
- 磁盘使用率过高时可能会发生节点脱离，后续节点自动恢复后也可能会因为集群压力过大，监控调 ES 接口查询集群状态时无响应，无法及时更新集群状态导致集群状态为不可用。

b. 增加集群可用磁盘容量。

- 在 Kibana 的“Dev Tools”页面，执行命令 **DELETE index_name** 清理集群的无效数据释放磁盘空间。
- 临时降低索引副本数，待扩容磁盘容量或扩容节点完成后改回索引副本数。

1) 在 Kibana 的“Dev Tools”页面，执行命令临时降低索引副本数。

```

PUT index_name/_settings
{
  "number_of_replicas": 1
}

```

如果返回结果如下：

图2-20 索引 read-only-allow-delete 状态

```
{
  "type" : "cluster_block_exception",
  "reason" : "index [log07] blocked by: [TOO_MANY_REQUESTS/12/disk usage
    exceeded flood-stage watermark, index has read-only-allow-delete block];"
}
```

则是因为磁盘使用率已超过磁盘空间允许的最大值，集群所有索引被强制设置“read_only_allow_delete”属性，先执行命令将该属性值置为“null”，再执行 b.1) 的命令降低索引副本数。

```
PUT /_settings
{
  "index.blocks.read_only_allow_delete": null
}
```

- 2) 参考[扩容](#)对集群进行节点数量或节点存储容量进行扩容。
- 3) 待扩容完成后再执行 b.1) 改回索引副本数，待索引分片完全分配后，集群状态变为可用。

2.7 数据类型不兼容导致集群不可用

问题现象

集群进行备份恢复或集群迁移操作后，“集群状态”变为“不可用”。

原因分析

集群出现此场景的原因可能是目标集群不支持被恢复的数据中某些数据类型，比如旧集群有安装一些插件或者定义 settings，新集群没有，导致的索引分片无法分配。

处理步骤

1. 在 Kibana 的“Dev Tools”页面，执行命令 `GET _cluster/allocation/explain?pretty` 查看索引分片未分配的原因。
2. 返回结果中显示索引名称“index”和未分配解释“explanation”：“primary shard for this replica is not yet active”，表示分片副本未激活。

图2-21 索引分片未分配

```
{
  "index": "write24",
  "shard": 3,
  "primary": false,
  "current_state": "unassigned",
  "unassigned_info": {
    "reason": "NEW_INDEX_RESTORED",
    "at": "2022-12-12T07:12:58.652Z",
    "details": "restore_source[restore_repo_auto/snapshot-8033]",
    "last_allocation_status": "no_attempt"
  },
  "can_allocate": "no",
  "allocate_explanation": "cannot allocate because allocation is not permitted to any of the nodes",
  "node_allocation_decisions": [
    {
      "node_id": "1xQ9pmVqSPqVTT1IRAB-wA",
      "node_name": "css-bfb",
      "transport_address": "10.10.10.10",
      "node_decision": "no",
      "deciders": [
        {
          "decider": "replica_after_primary_active",
          "decision": "NO",
          "explanation": "primary shard for this replica is not yet active"
        }
      ]
    }
  ]
}
```

3. 尝试修改该索引的配置，执行命令将其副本数置为 0。

```
PUT /index name/ settings
{
  "number_of_replicas": 0
}
```

返回信息“reason”中表示在恢复的数据中存在 CSS 集群不支持的数据类型。

图2-22 数据不兼容

```
"error": {
  "root_cause": [
    {
      "type": "mapper_parsing_exception",
      "reason": "Failed to parse mapping [_doc]: analyzer [ik_max_word] has not been configured in mappings"
    }
  ],
  "type": "mapper_parsing_exception",
  "reason": "Failed to parse mapping [_doc]: analyzer [ik_max_word] has not been configured in mappings",
  "caused_by": {
    "type": "illegal_argument_exception",
    "reason": "analyzer [ik_max_word] has not been configured in mappings"
  }
},
"status": 400
```

4. 根据问题根因，将数据中 CSS 集群不支持的数据类型删除或选择支持该数据类型的 CSS 集群版本，再进行备份恢复或数据迁移。

2.8 集群负载过高导致集群不可用

问题现象

“集群状态”为“不可用”，单击集群名称进入集群基本信息页面，选择“日志管理”，单击“日志查询”页签，可见日志内容存在报错“OutOfMemoryError”和警告“[gc][xxxxx] overhead spent [x.xs] collecting in the last [x.xs]”。

图2-23 频繁 GC 导致 OOM

日志时间	日志级别	日志内容
2023-02-21T06:25:17.424	Warning	[o.e.t.OutboundHandler] [css-HighLoad-ess-esn-1-1] send message failed [channel: Netty4TcpChannel[localAddress=0.0.0.0:0.0.0.0:14991, remoteAddress=192.168.96.51:192.168.96.51:9300][java...
2023-02-21T06:25:09.874	Warning	[i.suppressed] [css-HighLoad-ess-esn-1-1] path: /_cat/master, params: {org.elasticsearch.discovery.MasterNotDiscoveredException: NodeDisconnectedException[css-HighLoad-ess-esn-5-1] 192.16...
2023-02-21T06:25:09.873	Warning	[i.suppressed] [css-HighLoad-ess-esn-1-1] path: /_cat/master, params: {org.elasticsearch.discovery.MasterNotDiscoveredException: NodeDisconnectedException[css-HighLoad-ess-esn-5-1] 192.16...
2023-02-21T06:24:49.579	Error	[o.e.b.ElasticsearchUncaughtExceptionHandler] [css-HighLoad-ess-esn-1-1] fatal error in thread [elasticsearch[css-HighLoad-ess-esn-1-1][management][T45], exiting java.lang.OutOfMemoryError: Ja...
2023-02-21T06:23:40.154	Warning	[o.e.m.j.AvmOclMonitorService] [css-HighLoad-ess-esn-1-1] [gc[354945] overhead: spent [1.2m] collecting in the last [9.7s]
2023-02-21T06:22:24.317	Warning	[o.e.m.j.AvmOclMonitorService] [css-HighLoad-ess-esn-1-1] [gc[354844] overhead: spent [7.9s] collecting in the last [1.6s]
2023-02-21T06:22:16.306	Warning	[o.e.m.j.AvmOclMonitorService] [css-HighLoad-ess-esn-1-1] [gc[354843] overhead: spent [8.7s] collecting in the last [8.7s]
2023-02-21T06:22:08.535	Information	[o.e.m.j.AvmOclMonitorService] [css-HighLoad-ess-esn-1-1] [gc[354842] overhead: spent [1.6s] collecting in the last [5.6s]
2023-02-21T06:22:07.853	Warning	[o.e.m.j.AvmOclMonitorService] [css-HighLoad-ess-esn-1-1] [gc[354841] overhead: spent [8.2s] collecting in the last [4.3s]
2023-02-21T06:21:58.478	Warning	[o.e.m.j.AvmOclMonitorService] [css-HighLoad-ess-esn-1-1] [gc[354840] overhead: spent [1.8s] collecting in the last [1.8s]

原因分析

集群负载过高，可能是有大量查询或写入任务堆积。当堆内存不足时，任务无法分配，将频繁触发 GC，导致 Elasticsearch 进程异常退出。

处理步骤

说明

如果集群长期处于高负载状态，则集群会存在写入、查询缓慢等情形，建议根据业务需要升级节点规格或者对集群节点的数量和存储容量进行扩容，使集群更好的满足业务需求。升级节点规模的指导请参见[变更规格](#)，扩容节点数量和节点存储容量的指导请参见[扩容](#)。

1. 查询集群是否存在任务堆积。

- 方式一：在 Kibana 的“Dev Tools”页面，分别执行以下命令查询是否存在任务堆积。

```
GET /_cat/thread_pool/write?v
GET /_cat/thread_pool/search?v
```

如下所示“queue”的值为非 0，表示存在任务堆积。

node_name	name	active	queue	rejected
css-0323-ess-esn-2-1	write	2	200	7662
css-0323-ess-esn-1-1	write	2	188	7660
css-0323-ess-esn-5-1	write	2	200	7350
css-0323-ess-esn-3-1	write	2	196	8000
css-0323-ess-esn-4-1	write	2	189	7753

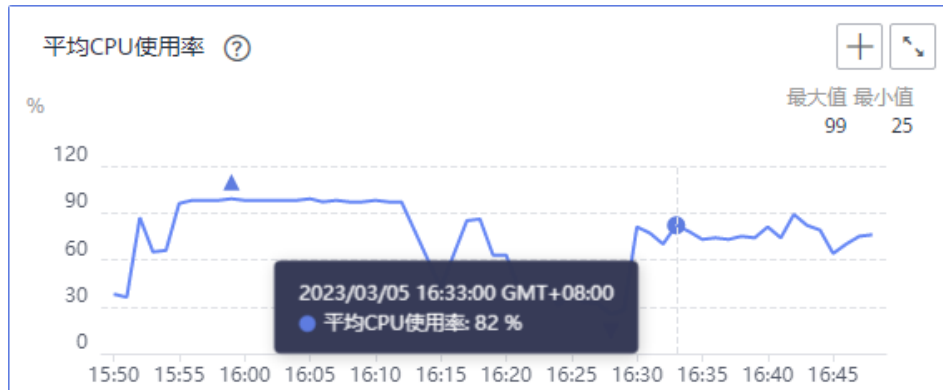
- 方式二：在集群管理列表，单击集群操作列的“监控信息”查看监控指标，在集群监控信息页面查看集群的“Search 队列中总排队任务数”和“Write 队列中总排队任务数”，若排队任务数值非 0 表示存在任务堆积。

图2-24 Write 队列中总排队任务数示例



- 如果集群存在大量的任务堆积，则参考如下步骤优化集群。
 - 在集群的“日志管理”页面查看节点日志，查看节点在 OOM 前是否存在大量慢查询日志记录，分析查询是否会对节点造成压力导致节点内存不足，如果存在则根据业务实际情况优化查询语句。
 - 在集群的“日志管理”页面查看节点日志，查看节点日志是否有“**Inflight circuit break**”或“**segment can't keep up**”的报错信息，如果存在则可能是写入压力过大，对集群造成较大的压力导致熔断。需要查看监控信息，排查近期数据写入量（写入速率）是否存在激增，如果存在则根据业务实际情况合理安排写入高峰时间窗。
 - 如果集群不存在任务堆积或者集群优化完依旧不可用，则执行下一步，查看集群是否压力过大。
2. 查看集群是否压力过大。
- 在集群管理列表，单击集群操作列的“监控信息”查看监控指标，在监控信息页面查看 CPU 和堆内存相关指标，如“平均 CPU 使用率”和“平均 JVM 堆使用率”。如“平均 CPU 使用率”超过 80%或“平均 JVM 堆使用率”高于 70%，则说明集群当前压力较大。

图2-25 “平均 CPU 使用率” 示例



- 如果集群压力过大，请降低客户端的请求发送速率或扩容集群。
- 如果集群压力正常或降低发送请求速率后集群依旧不可用，则执行下一步，查看集群是否存在大量缓存。

3. 在 Kibana 的 “Dev Tools” 页面，执行以下命令查询集群是否存在大量缓存。

```
GET /_cat/nodes?v&h=name,queryCacheMemory,fielddataMemory,requestCacheMemory
```

- 如果返回结果中 `queryCacheMemory`、`fielddataMemory` 或 `requestCacheMemory` 的数值超过堆内存的 20%，则表示缓存过大，可执行命令 **POST _cache/clear** 清除缓存。这些缓存数据是在数据查询时生成的，目的是为了加快查询速度，当缓存清除则可能使查询时延增加。

name	queryCacheMemory	fielddataMemory	requestCacheMemory
css-0323-ess-esn-1-1	200mb	1.6gb	200mb

📖 说明

每个节点的最大堆内存可以执行如下命令查询：

```
GET _cat/nodes?v&h=name,ip,heapMax
```

其中，`name` 为节点名称，`ip` 为节点的 IP 地址。

- 如果排查优化后，集群依旧负载过高，则联系技术支持。

3 数据导入导出类

3.1 Elasticsearch 显示 CPU 使用率高，导致日志无法写入

用户问题

Elasticsearch 在 12 月 31 日 21:30 左右 CPU 比较高，logstash 在该时刻报错 Elasticsearch Unreachable，导致日志无法写入到 elasticsearch 里。

问题现象

日志无法写入到 Elasticsearch 里。

原因分析

客户 index 是单 shard，压力承载于单个节点，负载过高，造成队列满后，作业被拒绝。

处理步骤

1. 登录云搜索服务控制台。
2. 选择“集群管理”进入集群管理列表。
3. 选择对应集群操作列“更多”>“Cerebro”。
4. 在 Cerebro 中查看集群的分片数、各节点的 cpu、load、head、dis 等数据指标。
5. 根据指标分析可能出现的原因，针对性优化。

a. 增加队列数，减少拒绝作业，修改参数 `write.queue_size` 取值。

i. 单击需要修改参数的集群名称，进入集群基本信息页面。

ii. 选择“参数配置”，查找 `write.queue_size` 并修改取值。

如果没有此参数，可以在自定义参数列进行添加。详情请参考[参数配置](#)章节。

b. 重建索引，使分片数大于集群节点数。

6. 如果分片数和队列大小都满足条件，但是 cpu 和负载依然比较高，建议扩容节点。

3.2 ECS 服务器部署 logstash 推送数据到云搜索服务报错

问题现象

ECS 服务器部署 logstash，然后推送数据到云搜索服务。，错误信息如下：

```
LogStash::Outputs::ElasticSearch::HttpClient::Pool::BadResponseCodeError: Got response code '500' contacting Elasticsearch at URL 'https://192.168.xx.xx:9200/_xpack'。
```

原因分析

目前云搜索服务没有集成 x-pack 插件，自行搭建 logstash 连接 es 服务的时候，会检查 es 是否启用了 x-pack。

处理步骤

1. 删除 logstash 中的 x-pack 目录。
2. 修改 logstash 配置文件 output 标签下 elasticsearch 内添加配置项 ilm_enabled => false。
3. 重新尝试推送数据到 es。

3.3 ES-Hadoop 导出数据时报"Could not write all entries"异常

问题分析

Elasticsearch 后台的 bulk 的线程池最大只支持接受 200 请求数队列，超过的请求会被 rejected。

解决方案

1. 建议根据实际情况调整客户端的并发写入请求数（调整到一个合适的阈值），另外被 rejected 的 http 请求 ES-Hadoop 是有重试机制的，可修改以下参数：
 - “es.batch.write.retry.count”：默认重试 3 次。
 - “es.batch.write.retry.wait”：每次重试等待时间 10s。
2. 如果对查询的实时性级别要求不高的话，可以调整下分片刷新的时间（默认是每秒刷新一次），提高写入速度。

```
PUT /my_logs
{
  "settings": {
    "refresh_interval": "30s"
  }
}
```

```
}  
}
```

4 功能使用类

4.1 无法备份索引

索引的备份是通过创建集群快照实现的。遇到无法备份索引问题，请按照如下操作步骤排查解决。

排查集群的创建时间

1. 登录云搜索服务管理控制台。
2. 在左侧导航栏，选择“集群管理”。
3. 在集群管理列表页，查看待备份索引的集群的“创建时间”。
 - 如果创建时间早于 2018 年 3 月 10 日，则创建该集群时备份与恢复索引功能尚未上线，当前无法为该集群备份索引。
 - 如果创建时间晚于 2018 年 3 月 10 日，则需要排查当前登录所用的帐号或 IAM 用户是否具有备份索引的权限，具体操作请参见[排查是否有权限](#)。

排查是否有权限

1. 登录统一身份认证服务管理控制台。
2. 查看当前登录所用的帐号或 IAM 用户所属的用户组。

具体操作请参见《统一身份认证服务用户指南》中的“如何查看或修改用户信息”章节。
3. 查看用户组的权限中是否包含：“全局服务”中“对象存储服务”项目的“OBS Administrator”权限、当前所属区域的“Elasticsearch Administrator”权限。

具体操作请参见《统一身份认证服务用户指南》中的“如何查看或修改用户组”章节。

 - 如果用户组的权限中不包含以上两个权限，请执行步骤 4。
 - 如果用户组的权限中包含以上两个权限，请联系技术支持协助解决。
4. 为用户组添加：“全局服务”中“对象存储服务”项目的“OBS Administrator”权限、当前所属区域的“Elasticsearch Administrator”权限。

具体操作请参见《统一身份认证服务用户指南》中的“如何查看或修改用户组”章节。

4.2 无法使用自定义词库功能

遇到该问题，请按照如下操作步骤排查解决。

排查集群的创建时间

步骤 1 登录云搜索服务管理控制台。

步骤 2 在左侧导航栏，选择“集群管理”。

步骤 3 在集群管理列表页，查看待配置自定义词库的集群的“创建时间”。

- 如果创建时间早于 2018 年 3 月 10 日，则创建该集群时自定义词库功能尚未上线，当前无法为该集群配置自定义词库。
- 如果创建时间晚于 2018 年 3 月 10 日，则需要排查当前登录所用的帐号或 IAM 用户是否具有使用自定义词库功能的权限，具体操作请参见[排查是否有权限](#)。


----结束

排查是否有权限

1. 登录统一身份认证服务管理控制台。
2. 查看当前登录所用的帐号或 IAM 用户所属的用户组。
具体操作请参见《统一身份认证服务用户指南》中的[查看或编辑用户信息](#)章节。
3. 查看用户组的权限中是否包含：“全局服务”中“对象存储服务”项目的“Tenant Administrator”权限、当前所属区域的“Elasticsearch Administrator”权限。
具体操作请参见《统一身份认证服务用户指南》中的[查看或修改用户组](#)章节。
 - 如果用户组的权限中不包含以上两个权限，请执行 4。
 - 如果用户组的权限中包含以上两个权限，请联系人工客服协助解决。
4. 为用户组添加：“全局服务”中“对象存储服务”项目的“Tenant Administrator”权限、当前所属区域的“Elasticsearch Administrator”权限。
具体操作请参见《统一身份认证服务用户指南》中的[查看或修改用户组](#)章节。

4.3 快照仓库找不到


1. 在云搜索服务的“集群管理”页面上，单击集群“操作”列的“Kibana”访问集群。
2. 在 Kibana 的左侧导航中选择“Dev Tools”，单击“Get to work”，进入 Console 界面。

Console 左侧区域为输入框，右侧为结果输出区域， 为执行命令按钮。

3. 执行命令 `GET _snapshot/_all` 返回为空，或者执行命令 `GET _snapshot/repo_auto/_all` 返回如图 4-1 提示错误，这个表示没有设置快照。需要重新修改快照配置信息。

图4-1 返回信息

```
1 {
2   "error": {
3     "root_cause": [
4       {
5         "type": "repository_missing_exception",
6         "reason": "[repo_auto] missing"
7       }
8     ],
9     "type": "repository_missing_exception",
10    "reason": "[repo_auto] missing"
11  },
12  "status": 404
13 }
```

4. 单击集群名称，进入集群基本信息页面，选择“集群快照”，进入集群快照页面。
5. 单击“基础配置”后面的  ，修改基础配置。
6. 修改完成后，单击“确定”。

如果保存后还不能生效，可以尝试修改下备份路径，换一个不一样的值保存然后再修改回来。

4.4 集群一直处于快照中

集群一直处于快照中，有三个比较常见的原因：

- 集群数据量大或者集群压力大，备份快照耗时长。

单个节点的快照速度默认是 40MB/s，同时，快照的性能还受集群情况影响，如果此时集群负载较高，耗时将会更久。可以通过上述章节的查询单个快照信息查询正在执行的快照情况。

执行 `GET _snapshot/repo_auto/snapshot-name`，可以看到剩余还需要完成的 shard 个数，也可以通过删除快照接口提前终止。

解决方法：等待或者提前终止。

- 快照信息更新失败。

Elasticsearch 将进行中的快照信息保存在 `cluster state` 中，快照完成后需要更新快照状态，由于 ES 更新快照状态的接口没有加入重试或者容错机制，比如由于当时集群内存压力大，更新快照动作被熔断，那么这个快照将会一直处于快照中。

解决方法：调用快照删除接口。

- 临时 AK、SK 过期。

CSS 通过委托将 es 中的数据写入到用户的 OBS 中，快照仓库创建的时候，需要去使用委托获取临时的 AK、SK 设置到仓库中。由于临时的 AK、SK 是有效性的（24 小时过期），如果一个快照超过 24 小时还未完成，那么这个快照将会失败。这种情况会有一个比较大的风险，因为此时仓库的 AK、SK 过期，无法对这

个仓库进行更新、查询、删除操作，这种情况下 `cluster state` 信息将会无法清除，只能通过普通重启（滚动重启无法生效）集群来清除 `cluster state` 里面残留的快照信息。

解决方法：暂时只能通过普通重启集群来消除，后期 CSS 会提供终止接口，可以来解决这种无法消除状态的现象。

4.5 数据量很大，如何进行快照备份？

如果快照数据量极大，快照备份要超过一天时，可参考如下方法进行优化。

1. 快照备份的时候指定索引，比如先分批，默认是*，将会备份所有的索引。
2. 使用自定义快照仓库。
 - a. 创建自定义仓库。

除了使用云搜索服务提供的 `repo_auto` 之外，客户也可以自己创建一个仓库，接口见如下：

```
PUT _snapshot/my_backup
{
  "type" : "obs",
  "settings" : {
    "bucket" : "css-backup-name", //桶名
    "base_path" : "css_backup/711/", //备份路径
    "chunk_size" : "2g",
    "endpoint" : "obs.xxx.com:443", //OBS 域名地址
    "region" : "xxx", //Region 名称
    "compress" : "true",
    "access_key" : "xxxxxx", //AK
    "secret_key" : "xxxxxxxxxxxxxxxxxxxx" //SK
    "max_restore_bytes_per_sec" : "100mb", //OBS 速度，默认是 40MB,可以根据实际性能调大
    "max_snapshot_bytes_per_sec" : "100mb"
  }
}
```

- b. 使用自定义仓库创建快照。

```
PUT _snapshot/my_backup/snapshot_name (快照名称)
{
  "indices" : "*", //备份的索引，*表示索引，逗号分隔
  "ignore_unavailable" : true, //是否忽略单个 index 是否可用,true 表示忽略
  "include_global_state" : false //默认 false 表示 cluster state 和其他的一些 state 不会保存下来
}
```

- c. 查询快照状态。

```
GET _snapshot/my_backup/snapshot_name/_status
```

- d. 恢复自定义仓库中的索引。

```
POST /_snapshot/my_backup/snapshot_name/_restore
{
  "indices" : "test-000000000000",
  "ignore_unavailable" : true,
  "include_global_state" : false,
}
```

```
"rename_pattern": "(.+)",  
"rename_replacement": "$1"  
}
```

4.6 集群突现 load 高的故障排查

问题现象

集群任务被长时间拒绝，且大量任务出现卡死的情况，在 Cerebro 界面可以看到集群的 load 数值突然飙升。

原因分析

集群出现 load 升高的可能原因如下：

- 查询请求命中的数据较多导致查询线程执行缓慢。
- 写入压力过大导致很多线程出现卡死现象。

排查步骤

方法 1：Cerebro 工具

1. 登录云搜索服务管理控制台。
2. 左侧导航栏，选择“集群管理 > Elasticsearch”，进入集群列表页面。
3. 找到 load 飙升的集群，单击集群操作列的“Cerebro”进入可视化页面。
4. 查看 cpu 和 heap 指标，若这两个指标过高则说明集群当前压力较大，客户端可以适当减少大请求发送，等待集群压力下降。
5. 查看 shards 是否合理，官方建议单个 shard 大小为 20-40GB，建议不要超过 50GB；单个节点上的同一索引 shard 数不要超过 5 个。

方法 2：Kibana 工具

1. 登录云搜索服务管理控制台。
2. 左侧导航栏，选择“集群管理 > Elasticsearch”，进入集群列表页面。
3. 找到 load 飙升的集群，单击集群操作列的“Kibana”进入页面，单击可以执行命令的 Dev Tools 工具。
4. 执行 GET _cat/thread_pool?v 查看哪些线程任务堆积，定位是什么原因导致的集群压力倍增。
5. 执行 GET /_nodes/hot_threads 可以查看当前占用大量 CPU 且执行时间很长的线程，定位何处导致任务积压。

4.7 使用 Elasticsearch 的 HLRC (High Level Rest Client) 时，报出 I/O Reactor STOPPED

问题现象

使用 Elasticsearch 的 HLRC (High Level Rest Client) 时，偶现报出 I/O Reactor STOPPED。排查 Elasticsearch 日志，未有报错。

```
java.lang.RuntimeException: Request cannot be executed; I/O reactor status: STOPPED
    at org.elasticsearch.client.RestClient.extractAndWrapCause(RestClient.java:796)
    at org.elasticsearch.client.RestClient.performRequest(RestClient.java:218)
    at org.elasticsearch.client.RestClient.performRequest(RestClient.java:205)
    at org.elasticsearch.client.RestHighLevelClient.internalPerformRequest(RestHighLevelClient.java:1454)
    at org.elasticsearch.client.RestHighLevelClient.performRequest(RestHighLevelClient.java:1424)
    at org.elasticsearch.client.RestHighLevelClient.performRequestAndParseEntity(RestHighLevelClient.java:1394)
    at org.elasticsearch.client.RestHighLevelClient.search(RestHighLevelClient.java:930)
    at
    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)
Caused by: java.lang.IllegalStateException: Request cannot be executed; I/O reactor status: STOPPED
    at org.apache.http.util.Asserts.check(Asserts.java:46)
    at org.apache.http.impl.nio.client.CloseableHttpAsyncClientBase.ensureRunning(CloseableHttpAsyncClientBase.java:90)
    at org.apache.http.impl.nio.client.InternalHttpAsyncClient.execute(InternalHttpAsyncClient.java:123)
    at org.elasticsearch.client.RestClient.performRequest(RestClient.java:214)
    ... 9 common frames omitted
```

I/O Reactor STOPPED 是什么问题？

首先根据调用栈可以定位到报错来自 `CloseableHttpAsyncClientBase` 中的 90 行，如下图所示：

```
88     protected void ensureRunning() {
89         final Status currentStatus = this.status.get();
90         Asserts.check( expression: currentStatus == Status.ACTIVE, message: "Request cannot be executed; " +
91             "I/O reactor status: %s", currentStatus);
92     }
```

`ensureRunning()` 方法是在每次请求执行开始的时候调用，用来确认 `client` 状态是否为 `ACTIVE`，如果不是 `ACTIVE`，则会报错。然后观察 `CloseableHttpAsyncClientBase` 中的 `status` 何时会变成 `STOPPED`，发现有且仅有两处会 `set` 为 `STOPPED`，如下图所示：

图4-2 第一处 STOPPED

```
public CloseableHttpAsyncClientBase(
    final NHttpClientConnectionManager connmgr,
    final ThreadFactory threadFactory,
    final NHttpClientEventHandler handler) {
    super();
    this.connmgr = connmgr;
    if (threadFactory != null && handler != null) {
        this.reactorThread = threadFactory.newThread(() -> {
            try {
                final IOEventDispatch ioEventDispatch = new InternalIODispatch(handler);
                connmgr.execute(ioEventDispatch);
            } catch (final Exception ex) {
                log.error( "I/O reactor terminated abnormally", ex);
            } finally {
                status.set(Status.STOPPED);
            }
        });
    } else {
        this.reactorThread = null;
    }
    this.status = new AtomicReference<Status>(Status.INACTIVE);
}
```

图4-3 第二处 STOPPED

```
@Override
public void close() {
    if (this.status.compareAndSet(Status.ACTIVE, Status.STOPPED)) {
        if (this.reactorThread != null) {
            try {
                this.connmgr.shutdown();
            } catch (final IOException ex) {
                this.log.error("I/O error shutting down connection manager", ex);
            }
        }
        try {
            this.reactorThread.join();
        } catch (final InterruptedException ex) {
            Thread.currentThread().interrupt();
        }
    }
}
```

- 由于客户不会手动关闭客户端之后再调用接口，所以有且仅有第一处会导致 status 切换为 STOPPED 状态。
- 对于第一处 STOPPED，reactorThread 线程是用来在 io events 发生时调度 io events，当内部抛出异常时，最终会将 status 改为 STOPPED 状态。然后在 bulkAsync 请求的回调中抛出异常验证 status 的状态切换，如下图所示：

```
client.bulkAsync(request, RequestOptions.DEFAULT, new ActionListener<BulkResponse>() {
    @Override
    public void onResponse(BulkResponse bulkResponse) {}

    @Override
    public void onFailure(Exception e) {
        e.printStackTrace();
        throw new RuntimeException("bulk failed!");
    }
});
```

- 当请求失败，status 会切换为 STOPPED 且 I/O Reactor 将关闭，并使 HLRC 实例卡住。后续使用该 HLRC 实例调用任何请求都会失败。此处手动抛出异常是为了复现问题，生产环境中很难分辨是什么原因导致 I/O Reactor 关闭。

原因分析

导致出现 I/O Reactor STOPPED 的原因，大致可以分为以下 3 类：

1. 回调中抛出异常导致。
2. 客户端并发太高导致。

在日志中发现异常后，查看 Elasticsearch 集群监控指标，如 CPU 使用率、网络连接数等。

当用户集群配置为 5 台 16U128G 的 i3.4xlarge.8 节点时，且每天上午 5 点左右会做大量 bulk 操作，写入大概 100G-200G 的数据，根据集群监控指标的 CPU 使用率、网络流入流出速率来看对 Elasticsearch 节点造成不了压力，网络连接数较高，其它节点情况也相同。但是，有的节点网络连接数高达近 9000，5 个节点瞬间有将近 5 万连接数，用户的代码大致是用同一个 Rest Client 多个线程并发且调用 HLRC 的 bulkAsync 接口。客户端一个节点，单是 ES 的连接就消耗了 4-5 万个连接数，这种情况很容易造成客户端节点句柄数耗尽，或者连接数耗尽。

3. Elasticsearch 的 Rest client 导致。建议 Elasticsearch 完善 Rest client，添加 exception handler。

Apache (HLRC 和 LLRC 都使用了 Apache HTTPComponents Async Client) 手册中提到，在与会话通道交互过程中有些 I/O 异常是可以预料的，这些异常可能会导致单个 session 终止，但不会影响 I/O Reactor 和其他 session。但某些情况下，当 I/O Reactor 本身遇到内部问题是，例如底层 NIO 的一些类中的 I/O 异常或者没被 handle 的一些 Runtime Exception。这些异常是致命的，会使 I/O Reactor 关闭。

Apache 官方建议重写 IOReactorExceptionHandler 接口。

```
DefaultConnectingIOReactor ioreactor = <...>
ioreactor.setExceptionHandler(new IOReactorExceptionHandler() {
    public boolean handle(IOException ex) {
        if (ex instanceof BindException) {
            // bind failures considered OK to ignore
            return true;
        }
        return false;
    }
    public boolean handle(RuntimeException ex) {
        if (ex instanceof UnsupportedOperationException) {
            // Unsupported operations considered OK to ignore
            return true;
        }
        return false;
    }
});
```

但是尝试了重写 ExceptionHandler 并放到 HLRC 的配置中，并通过在回调中抛异常来模拟异常场景，最后发现这种回调的异常并不会被 IOReactorExceptionHandler 捕获，运行脚本后也没有异常抛出，所以此处 IOReactorExceptionHandler 的实现并不好验证。通过 Elasticsearch 社区 issue 中其他开发者的验证，添加了 IOReactorExceptionHandler 后跑很久也不会有问题。所以建议添加 IOReactorExceptionHandler，但是注意不要忽略所有异常。

Elasticsearch Rest Client 需要有一个 exception handler，而不是让用户通过设置 IOReactorExceptionHandler 来处理异常，且这种方式也不会解决所有的异常。

处理建议

1. 客户端连接池大小建议根据业务实际情况调整。
2. 客户端并发数建议根据业务实际情况调整，或者配置多个节点、分摊压力。
3. 建议配置 IOReactorExceptionHandler，可以用来处理一些异常。
4. 调用 HLRC 时 catch exception，并判断 cause 是否为 I/O Reactor STOPPED，然后通过重启客户端恢复连接。

4.8 Elasticsearch 集群最大堆内存持续过高（超过 90%）

问题描述

关于 Elasticsearch 集群的最大堆内存持续超过 90% 的问题。其中若节点在 90% 堆内存上下波动，有增有减，则无异常；持续高内存时，集群存在一定的风险。

原因分析

- 排查集群的写入和查询队列，查看是否有大量任务堆积。

```
GET /_cat/thread_pool/write?v
GET /_cat/thread_pool/search?v
```

- 查看集群监控，排查集群的写入和查询任务相关指标。

- 如果集群长期处于高堆内存占用状态，查看集群节点个数、节点规模，确认是否需要扩容。

解决方案

- 根据任务堆积现象优化客户端写入或查询程序。
- 根据业务情况，若集群长期处于高负载状态，则集群会存在写入、查询缓慢，节点频繁掉线等情况。需要根据需要扩容节点规模，或分业务重新规划集群规模。
- 若日常使用有 95%堆内存波动和节点掉线情况，可根据需要使用流量控制功能。

4.9 Elasticsearch 集群更改规格失败

问题描述

执行集群更改规格操作失败，console 界面报错详情如下图所示。

图4-4 CSS.0073 错误

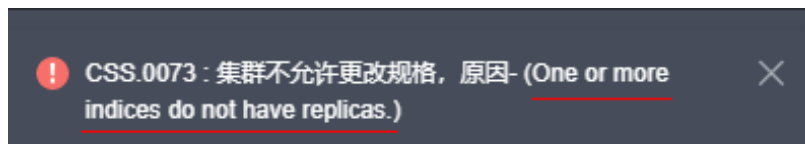
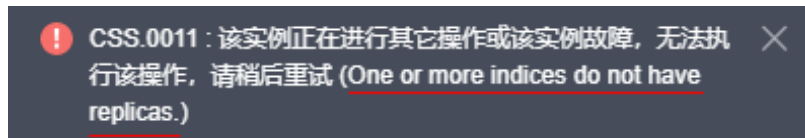


图4-5 CSS.0011 错误



原因分析

当前集群未设置副本数，后台拦截了更改规格的请求。需要设置好副本数，再进行更改规格操作，否则会有分片丢失的风险。

解决方案

设置副本参数：

```
PUT /index/_settings
{
  "number_of_replicas" : 1 //表示需要设置的副本数
}
```


4.10 安全集群索引只读状态修改报错

问题描述

安全集群空间存满之后，索引会全部变为只读模式"**read_only_allow_delete**": "true"，导致无法再写入，需手动修改只读模式为“false”，执行如下命令：

```
PUT _settings
{
  "index": {
    "blocks": {
      "read_only_allow_delete": "false"
    }
  }
}
```

报错为：

```
{
  "error": {
    "root_cause": [
      {
        "type": "security_exception",
        "reason": "no permissions for [] and User [name=admin, roles=[admin], requestedTenant=null]"
      }
    ],
    "type": "security_exception",
    "reason": "no permissions for [] and User [name=admin, roles=[admin], requestedTenant=null]"
  },
  "status": 403
}
```

原因分析

安全集群，默认有一个`.opendistro_security`索引，不可执行写操作，修改索引读写模式时要忽略掉这个索引。

解决方案

使用通配符进行匹配，将 `indexname` 用通配符代替。

```
PUT indexname/_settings
{
  "index": {
    "blocks": {
      "read_only_allow_delete": "false"
    }
  }
}
```

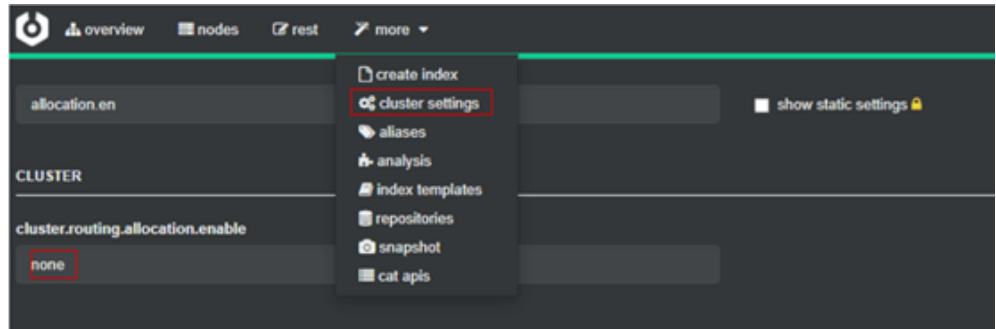
4.11 Elasticsearch 集群某一节点分配不到 shard

解决方案

1. 使用 `GET _cluster/allocation/explain?pretty` 查看未分配 shard。

```
"can_allocate": "no",
"allocate_explanation": "cannot allocate because allocation is not permitted to any of the nodes",
"node_allocation_decisions": [ -
  ( -
    "node_id": "VzWv0uP9SksrDID0frUtTg",
    "node_name": "prod-log-ess-esn-2-1",
    "transport_address": "192.168.100.91:9300",
    "node_decision": "no",
    "weight_ranking": 1,
    "deciders": [ -
      ( -
        "decider": "enable",
        "decision": "NO",
        "explanation": "no allocations are allowed due to cluster setting [cluster.routing.allocation.enable=none]"
      )
    ]
  )
],
```

2. 在 console 上，选择“cerebro > more > cluster settings”，在左上角输入“allocation.enable”，将“none”改为“all”。



3. 如果没有 cerebro，请执行如下命令：

```
curl -X PUT "http://内网 ip:9200/_cluster/settings" -H 'Content-Type: application/json' -d '{"persistent":{"cluster.routing.allocation.enable": "all"}}'
```

4.12 集群索引插入数据失败

问题现象

向 CSS 集群索引中插入数据失败，报错如下：

```
"reason": "blocked by: [FORBIDDEN/12/index read-only / allow delete (api)];")",
```

问题分析

当磁盘的使用率超过 95%时，Elasticsearch 为了防止节点耗尽磁盘空间，自动将索引设置为只读模式。

解决方案

- 新版本（7.10.2 之后）集群磁盘使用率下降后会自动关闭只读模式，只需清理或扩容磁盘。

- 旧版本需要手动操作，kibana 执行如下命令：

```
PUT /_all/_settings { "index.blocks.read_only_allow_delete": null }
```

4.13 CSS 创建索引报错 maximum shards open

问题描述

创建索引时，报错显示 this action would add [2] total shards, but this cluster currently has [1000]/[1000] maximum shards open。

问题原因

由于节点 shard 数量超过最大值限制，ES 默认每个节点的 shard 数量是 1000，如果 shard 数量超过这个值创建索引会报错。

解决方案

1. 关闭或者删除不用的索引，减少 shard 数量。
2. 修改节点的 shard 数量的限制。

```
PUT _cluster/settings
{
  "persistent": {
    "cluster": {
      "max_shards_per_node": 2000
    }
  }
}
```

4.14 删除索引报错“403 Forbidden”是什么原因？

问题描述

执行命令 `curl -i -u admin:password -XDELETE https://ip:9200/_all` (“password” 为 admin 帐号的密码，“ip” 为集群的内网访问地址) 删除所有索引时，报错“403 Forbidden”。

解决方案

安全模式的集群中索引“.opendistro_security”无法被删除，所以删除所有索引的命令对安全集群无效。建议使用索引名称或者通配符进行删除，不要使用全删除命令。

4.15 Kibana 中删除 index pattern 报错 Forbidden

问题描述

在 Kibana 界面删除索引模式，单击删除按钮报错 Forbidden。

原因分析

之前创建的索引模式无法删除索引模式是因为 kibana 索引只读导致的，磁盘使用率超过一定阈值会自动转为只读，所以报错没有权限。

解决方案

在 Kibana 的“Dev Tools”页面，执行以下命令解除 Kibana 索引只读状态。

```
PUT .kibana*/_settings
{
  "index.blocks.read_only_allow_delete":null
}
```

4.16 执行命令 update-by-query 报错 “Trying to create too many scroll contexts”

问题现象

云搜索服务的 Elasticsearch 集群执行命令 **update-by-query**，出现报错 “Trying to create too many scroll contexts. ”，具体报错信息如下：

```
{"error":{"root_cause":[{"type":"exception","reason":"Trying to create too many scroll contexts. Must be less than or equal to: [100000]. This limit can be set by changing the [search.max_open_scroll_context] setting."}],"type":"exception","reason":"Trying to create too many scroll contexts. Must be less than or equal to: [100000]. ....."}}
```

原因分析

当集群每调用一次 scroll 的创建接口，都会新建一个 scroll 使用的 context，当 scroll contexts 的数目达到预定值时，将无法继续创建 scroll。

处理步骤

执行如下命令，查看“open_contexts”的参数值即为当前 scroll contexts 的数目。

```
GET /_nodes/stats/indices/search
```

当数目达到最大值时，可以通过以下两种方式解决。

- 方式一：删除无用的 scroll，释放 scroll contexts 的存储空间。

```
DELETE /_search/scroll
{ "scroll_id" :
  "DXF1ZXJ5QW5kRmV0Y2gBAAAAAAAAAD4WYm91aVYtZndUQ1NsdDcwakFMNjU1QQ==" }
```

- 方式二：调整“search.max_open_scroll_context”的值，增加 scroll contexts 的存储空间。

```
PUT /_cluster/settings
{
  "persistent" : {
    "search.max_open_scroll_context": 2012345678
  },
  "transient": {
    "search.max open scroll context": 2012345678
  }
}
```

4.17 Elasticsearch 集群无法创建 pattern

问题描述

单击创建 pattern 无反应，无法创建 pattern。

问题原因

1. 检查磁盘是否太满，导致 Kibana 索引为只读状态。
2. 检查是否有多个 Kibana 索引。

解决方案

执行以下命令删除不必要的索引数据，释放磁盘空间。

```
PUT .kibana/_settings
{
  "index": {
    "blocks": {
      "read_only_allow_delete": "null"
    }
  }
}
```

5 端口访问类

5.1 9200 端口访问失败

问题现象

通过 VPN 专线或 VPC 的对等连接访问 CSS 集群的场景下，使用 curl 命令接入 Elasticsearch 集群时，无返回结果。

例如，执行如下命令接入集群，无返回结果。

```
curl -s 'http://<节点内网访问地址>:9200'
```

原因分析

在“使用 VPN 专线访问 CSS 集群”或“通过 VPC 的对等连接访问 CSS 集群”场景下，其所在的客户端与 CSS 不在同一 VPC 下。因此，要求 CSS 集群的子网与其 VPC 具有不同的网段。

例如，某一 CSS 集群，选用的 VPC 为 vpc-8e28，其网络配置为 192.168.0.0/16。选用了此 VPC 下的子网 subnet-4a81，subnet-4a81 子网的网段与 vpc-8e28 一致，均为 192.168.0.0/16。此时，如果使用 VPN 专线访问 CSS 集群或通过 VPC 的对等连接访问 CSS 集群，会导致此子网创建的机器内没有该 VPC 对应的网关，从而影响 CSS 服务的默认路由的设置，最终导致 9200 端口访问失败。

处理步骤

当出现 9200 端口访问失败错误时，且 CSS 集群状态为可用状态。执行步骤如下所示：

1. 进入 CSS 服务管理控制台，在集群列表中，单击集群名称进入集群详情页面，查看此集群使用的 VPC 和子网。
2. 进入 VPC 服务管理控制台，在虚拟私有云列表中，单击 CSS 集群使用的 VPC 名称，进入 VPC 详情页面。查看 VPC 和子网的网段信息。

如图 5-1 所示，VPC 的网段信息，与子网的网段信息一致。在使用 VPN 专线访问或使用 VPC 对等连接访问时，会导致 9200 端口访问失败。

图5-1 查看网段信息



3. 如果出现上述错误，请重新创建集群，并选择一个网段与 VPC 不同的子网，如不存在这样的子网，请在 VPC 管理控制台重新创建一个子网。

创建新的 CSS 集群后，将旧集群的数据迁移至新集群中，然后再通过 VPN 专线访问或使用 VPC 对等连接访问使用。

📖 说明

如果需要 VPN 专线访问或使用 VPC 对等连接访问 CSS 集群时，请务必保证，新创建的 CSS 集群，其 VPC 与子网，具备不同的网段信息。

6 修订记录

发布日期	修改说明
2023-07-24	第一次正式发布。