# PAAS平台紧急问题处理预案

版本：v1.0

目录

[PAAS平台紧急问题处理预案 1](#_Toc18750)

[1. FOOT平台 4](#_Toc7717)

[1.1 目的 4](#_Toc1392)

[1.2 应急预案 4](#_Toc3579)

[1.2.1 镜像仓库异常，无法启动 4](#_Toc10182)

[1.2.2 FOOT平台组件状态异常 5](#_Toc13328)

[1.2.3 pod无法拉起 5](#_Toc26207)

[1.2.4 Heketi服务挂掉，启动失败 6](#_Toc23896)

[1.2.5 镜像仓库空间被占满，无法上传镜像 7](#_Toc21613)

[1.2.6 应用访问失败，日志报unKownHostException 8](#_Toc2593)

[1.2.7 应用一直启动中，但是k8s集群中实际已经启动成功 9](#_Toc19293)

[1.2.8 配置了外部存储的应用，应用一直启动中 9](#_Toc1812)

[1.2.9 gfs主机上挂载点和应用的对应关系 10](#_Toc8797)

[1.2.10 日常维护常用命令 11](#_Toc16169)

[1.3 现场问题处理 12](#_Toc20225)

[2. CSF微服务管理平台 12](#_Toc5685)

[2.1 目的 12](#_Toc8209)

[2.2 CSF 调用讲解 12](#_Toc22788)

[2.2.1 调用流程 12](#_Toc3016)

[2.2.2 服务路由信息同步到zk 15](#_Toc30486)

[2.2.3 单服务超时时间配置 15](#_Toc5388)

[2.2.4 默认超时时间配置 16](#_Toc7035)

[3. IFACE平台 17](#_Toc10087)

[3.1 目的 17](#_Toc9575)

[3.2 应急预案 17](#_Toc1995)

[3.3 现场问题处理 17](#_Toc18025)

[4. LOG4X平台 20](#_Toc10336)

[4.1 目的 20](#_Toc31605)

[4.2 应急预案 20](#_Toc24568)

[4.2.1 log4x突然无法查询到数据 20](#_Toc32541)

[4.2.2 各组件集群状态检查汇总 25](#_Toc18900)

[4.3 现场问题处理 30](#_Toc1707)

[4.3.1 广西B域log4x生产环境业务轨迹办理查询速度过慢问题解决 30](#_Toc30944)

[4.3.2 通过traceid直接从hbase中查询数据，以json格式显示 34](#_Toc26880)

# FOOT平台

## **目的**

为了满足FOOT平台应对突发情况的能力，特整理一些可能遇到的突发情况并针对突发情况整理相应的处理措施，及时高效的保障系统的快速恢复；

## **应急预案**

### 镜像仓库异常，无法启动

场景描述：

镜像仓库registry始终无法启动，尝试解决失败

处理方法：

重新安装镜像仓库，数据目录指定之前备份的镜像仓库数据，启动镜像仓库成功，镜像仓库数据未丢；

具体步骤：

1. 分别登录镜像仓库所有节点
2. 执行命令卸载镜像仓库组件

docker stop b10ac7b77259

docker rm b10ac7b77259

其中b10ac7b77259为容器ID

容器ID查看方法：

docker ps | grep foot-registry

1. 删除镜像仓库数据目录

rm –rf /data/foot/install/app/data/registry

注：删除镜像仓库目录之前一定要确保镜像仓库数据目录已备份；

1. 恢复registry数据
2. 重新安装镜像仓库，镜像仓库数据目录指定为/data/foot/install/app/data/registry

安装文档参考之前提供的《Registry仓库部署指导手册.docx》

1. 确认镜像仓库是否启动成功，镜像数据是否未丢

docker ps | grep foot-registry

查看镜像仓库容器是否运行正常

确认正常后访问http://镜像仓库vip:35000/v2/\_catalog 查看镜像仓库数据返 回正常，与备份的镜像仓库中的镜像数据保持一致，证明镜像仓库恢复成功。

### FOOT平台组件状态异常

场景描述：

1. 通过FOOT平台页面发现某些组件状态异常
2. 通过FOOT日常巡检检测脚本发现某些组件状态异常
3. 接警人收到某些组件异常的告警

以上场景处理方法：

1. 如果发现某个组件状态异常，去对应主机上重启该组件

启动方法参考《FOOT平台维护手册》中相应组件的启动方法

1. 如果依然启动失败，参考《FOOT平台维护手册》中的组件日志查看方法，

查看日志找到具体失败原因，逐个解决；

### pod无法拉起

场景描述：

应用无法启动成功，启动失败或一直处于启动中

应用无法滚动升级成功，滚动升级失败或一直处于滚动升级中

解决方法：

1、通过如下命令查看pod的等待事件，确认具体的异常原因

kubectl describe pod/pod名称 -n 命名空间

也可以查看pod相应的logs，查看相应错误：

kubectl logs pod/pod名称 -n 命名空间

2、如未查明原因，登录对应拉起主机，检查主机的kubelet和docker的日志事件

3、如仍未解决，删除异常pod重新拉起。优先拉起服务。

4、如仍未解决，联系foot相关人员

常见的pod无法拉起的原因:

1. **下载镜像失败**
2. **主机资源不足**
3. **Pod资源规格设置不当**
4. **健康检查失败**

### Heketi服务挂掉，启动失败

场景描述：

Heketi服务挂掉，重新启动，依然启动失败

日志报错：

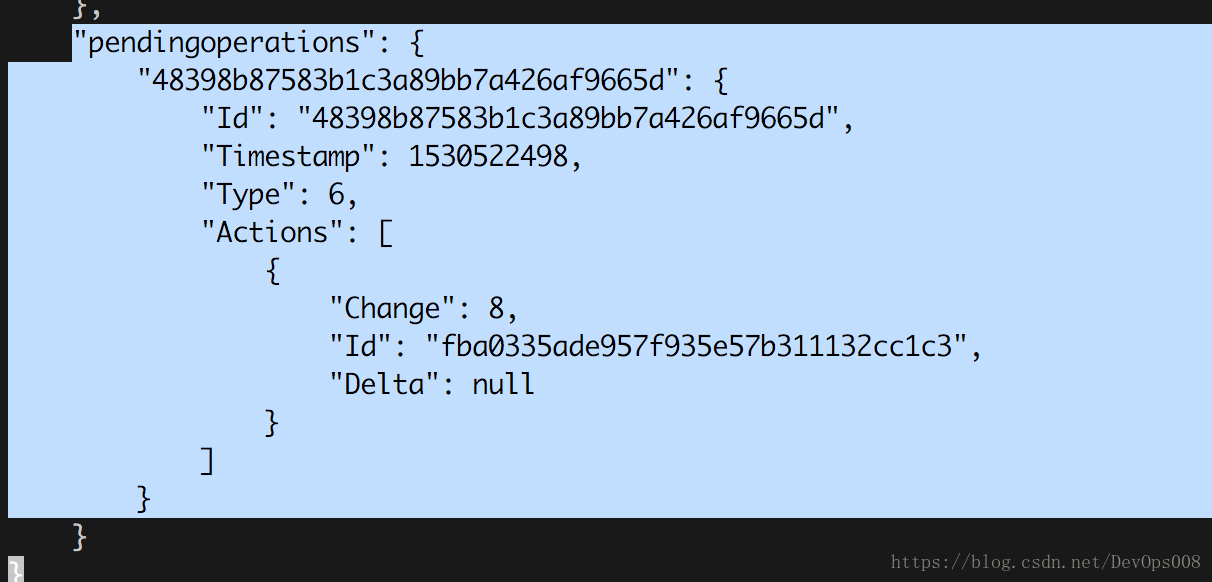
/src/github.com/heketi/heketi/apps/glusterfs/app.go:172: Heketi was terminated while performing one or more operations. Server may refuse to start as long as pending operations are present in the db.

处理方法：

1、导出heketi的heketi.db文件，文件的路径在heketi.json文件里面

heketi db export --dbfile=/var/lib/heketi/heketi.db --jsonfile=/tmp/heketidb.json

2、打开导出的db文件，比如上文的/tmp/heketidb1.json，查找“pendingoperations”选项，找到之后把与它相关的内容删除。



1. 将修改后的文件保存，切记要保存为json后缀。然后将db文件再按照如下命令导入：

heketi db import --jsonfile=/tmp/ heketidb.json --dbfile=/var/lib/heketi/heketi.db

1. 重启heketi 服务

systemctl start heketi

1. 检查heketi是否启动成功

执行命令systemctl status heketi查看heketi状态是否为running；

如果为running执行如下命令检查拓扑信息是否加载正确：

heketi-cli --user admin --secret admin --server http://heketiIP:18080 topology info

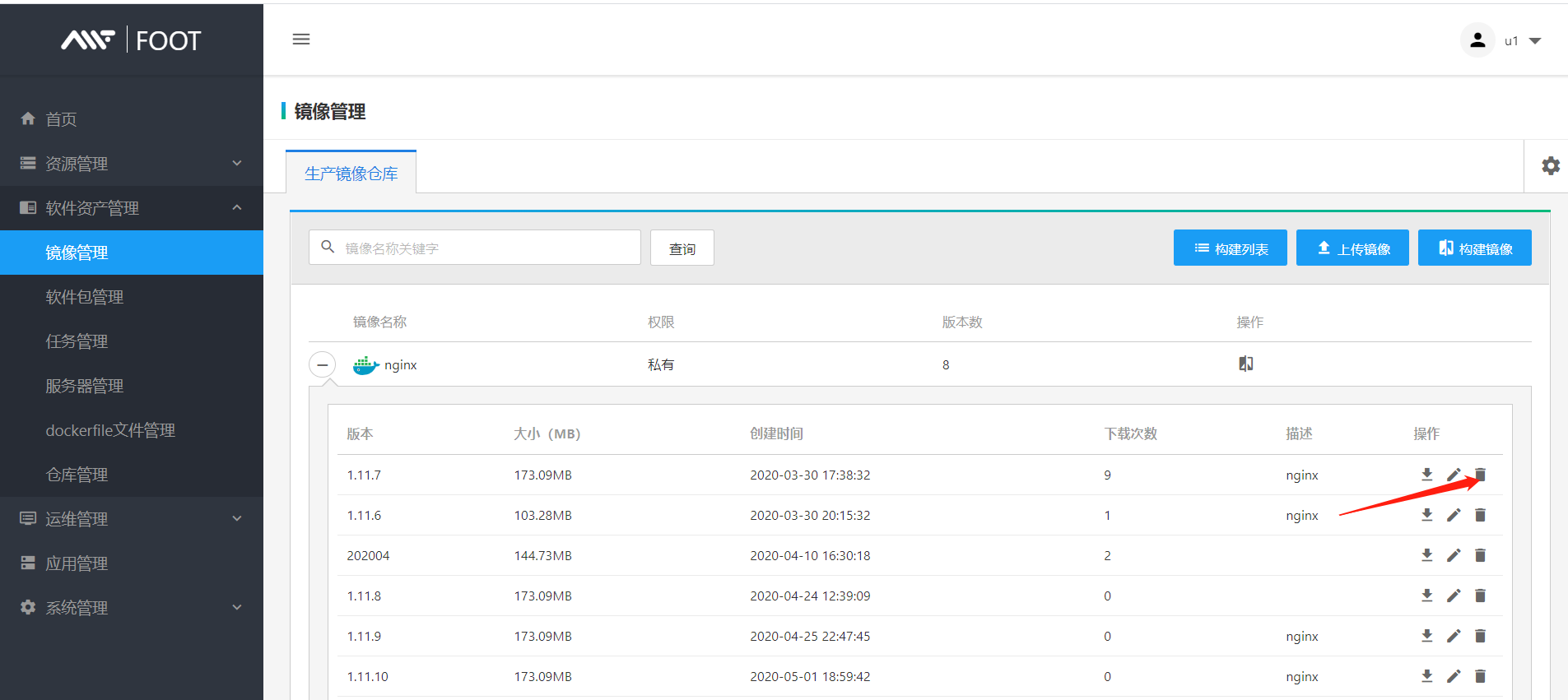
### 镜像仓库空间被占满，无法上传镜像

场景描述：

镜像仓库生产环境规划300G的空间，如果镜像一直不删除，导致规划的300G空间全部被占满，通过页面上传镜像失败。

处理方法：

1. 登录FOOT管理平台
2. 在软件资产管理-》镜像管理页面中，选中不需要的镜像，点击删除；



1. 删除镜像后，可以等到镜像垃圾回收脚本定时执行或手动调用垃圾回收脚本，即可释放镜像占用磁盘空间；

镜像垃圾回收脚本位置：/home/foot/registry\_garbagecollect/registry\_garbage\_collect.sh

镜像垃圾回收脚本每天会定时回收；

手动调用方式：

sh /home/foot/registry\_garbagecollect/registry\_garbage\_collect.sh

1. 登录镜像仓库主机查看镜像磁盘空间是否已经释放

df -h | grep /data/foot/install/data/registry

1. 通过FOOT平台软件资产管理-》镜像管理再次上传镜像，镜像是否上传成功

### 应用访问失败，日志报unKownHostException

场景描述：

应用通过ingress访问失败，应用日志报unKownHostException

处理方法：

1. 登录所有kube node节点；
2. 执行命令systemctl status kube-proxy检查所有节点kube-proxy组件运行状态
3. 在检查失败的节点上启动kube-proxy

systemctl start kube-proxy

### 应用一直启动中，但是k8s集群中实际已经启动成功

场景描述：

通过FOOT管理平台启动应用，应用状态一直启动中，但是通过

Kubectl命令去k8s集群中查看应用已经拉起成功

处理方法：

1. 登录foot-home主机，查看主机当前时间
2. 登录数据库，查询数据库时间

select now();

1. 比较数据库时间和foot-home主机时间是否不一致，

如果不一致，修改为一致即可；

注：数据库时间和主机时间不一致会导致定时刷新应用状态任务处理存在问题；

### 配置了外部存储的应用，应用一直启动中

场景描述：

通过FOOT管理平台创建应用，给应用配置了共享存储，应用状态一直是启动中

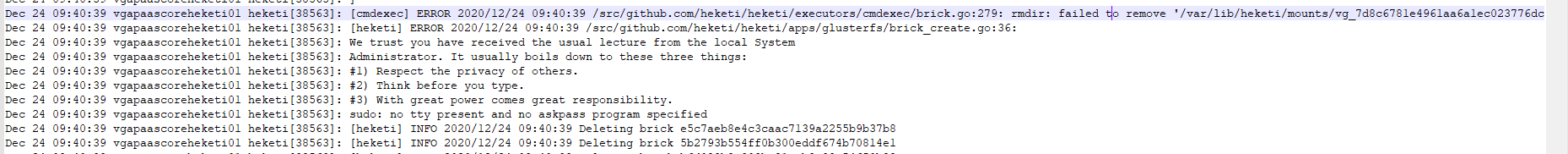
处理方法：

1. 登录kube master主机
2. kubectl get pvc -n \*\*\* 查看pvc的状态

若对应pvc的状态一直是pending，说明heketi出现了故障

1. 检查heketi状态，若heketi运行正常则通过journalctl -u heketi 查看heketi日志

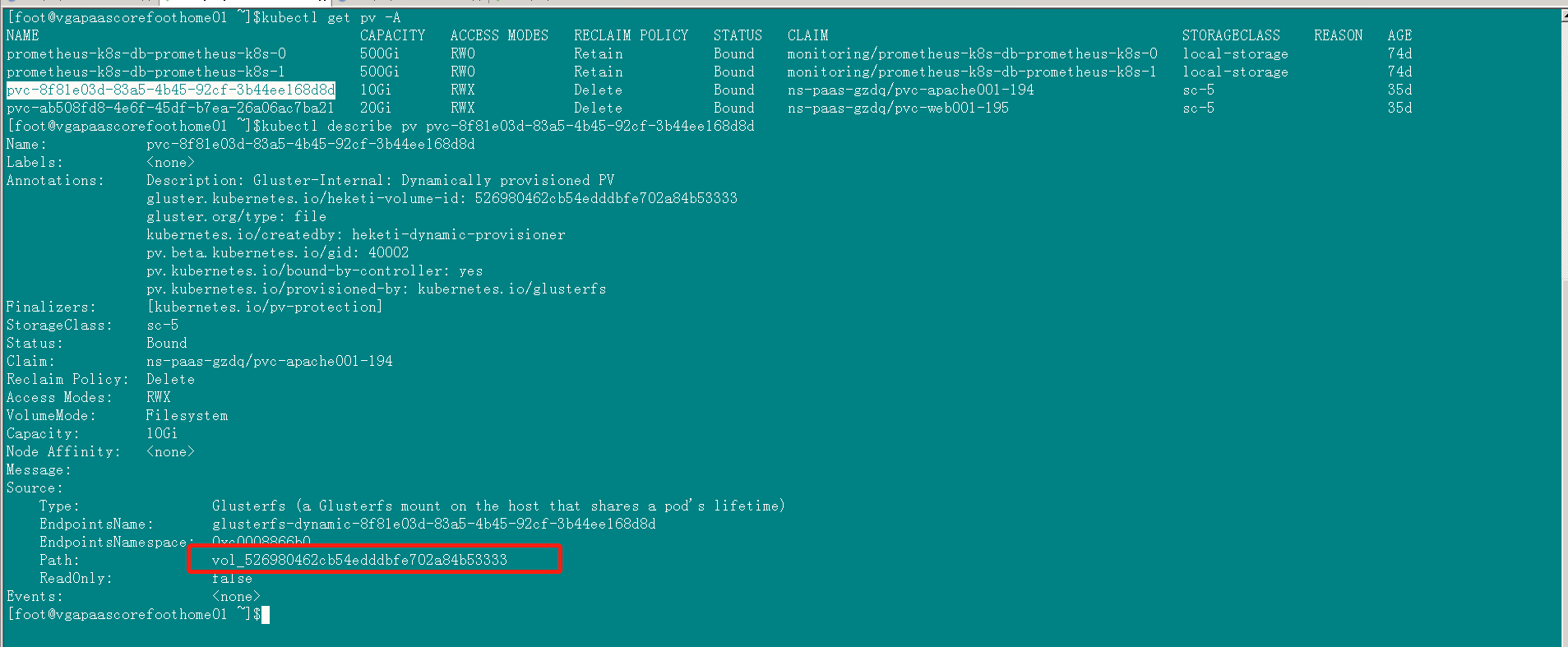
说明：如果heketi是正常的，通常是heketi主机的foot用户不能免密登录到gfs主机上，或者sudo权限缺失，通过heketi的日志就能定位出来。



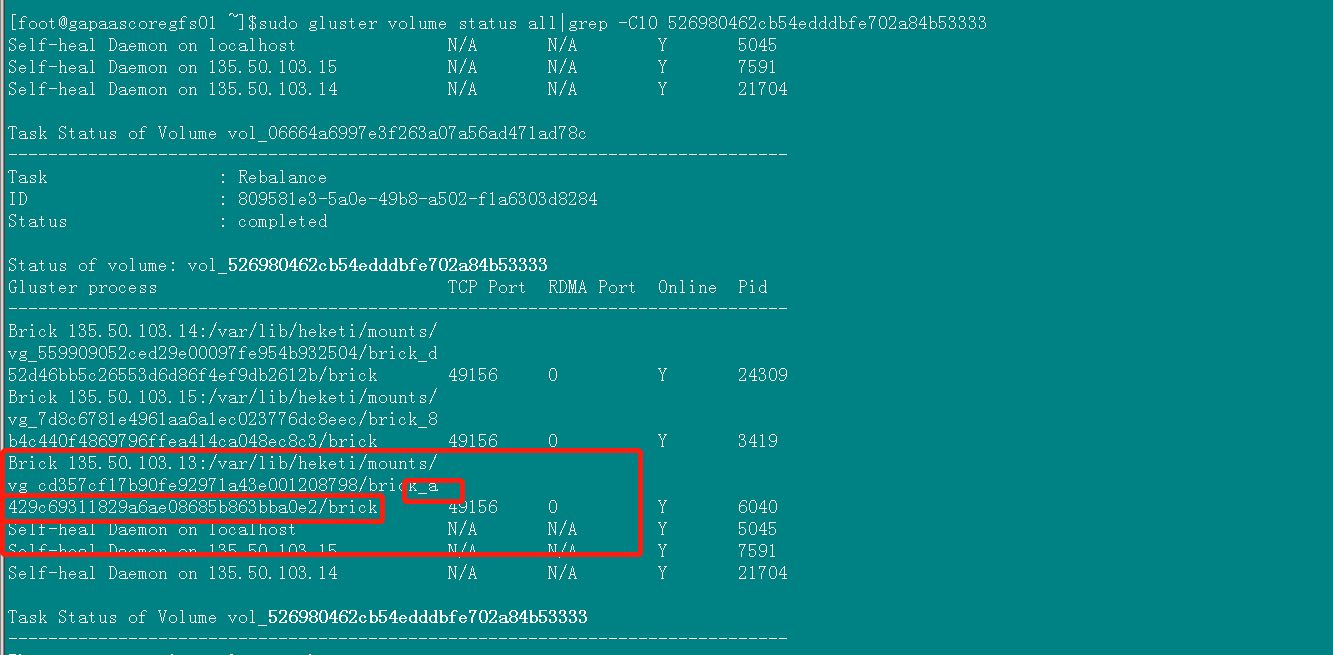
### gfs主机上挂载点和应用的对应关系

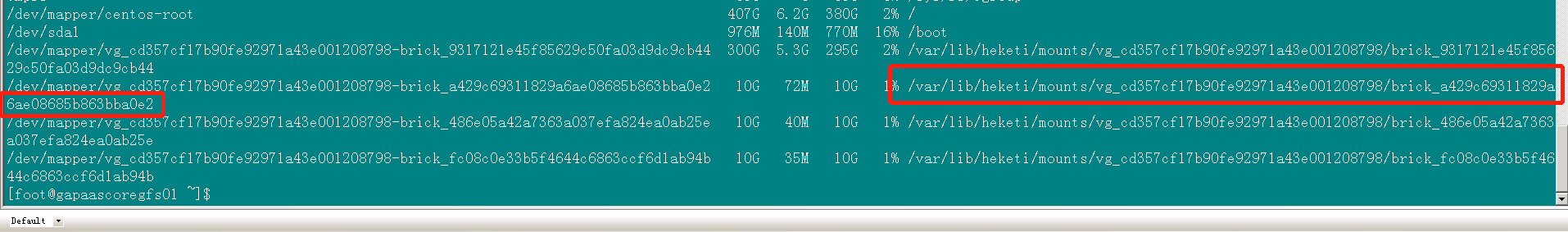
场景：直接在gfs主机上执行df -h，查询结果中的mount点看不出来是哪个应用的

1. kubectl get pvc ，可以看到应用对应的vol卷名



2、gluster volume status all | grep -C10 卷名，可以查看到对应的brick名称，这样就能和df -h 查出来的挂载点对应上了





### 日常维护常用命令

* k8s相关命令

# 查看cboss租户下的应用

kubectl get pod -n ns-cboss -o wide

# 查看pod的具体信息

kubectl get pod/deploy-web-route-soap-5bbdbcff9f-shjz2 -n ns-cboss -o yaml

# 查看pod的具体信息包含事件信息

kubectl describe pod/deploy-web-route-soap-5bbdbcff9f-shjz2 -n ns-cboss

# 查看pod的控制台日志

kubectl logs pod/deploy-web-route-soap-5bbdbcff9f-shjz2 -n ns-cboss

# 查看pod的资源使用

kubectl top pod deploy-web-route-soap-5bbdbcff9f-shjz2 -n ns-cboss

* gfs相关命令

#查看volume列表

gluster volume list

#查看volume信息

gluster volume info

gluster volume info vol\_ece4c072741128dac430956f2620f451

#查看volume状态

gluster volume status all

gluster volume status vol\_ece4c072741128dac430956f2620f451

#停止volume

gluster volume stop vol\_ece4c072741128dac430956f2620f451

#启动volume

gluster volume start vol\_ece4c072741128dac430956f2620f451

* heketi相关命令

heketi-cli --user admin --secret admin --server http://vip:18080 topology info

## **现场问题处理**

# CSF微服务管理平台

## 目的

为了满足微服务平台应对突发情况的能力，特整理一些可能遇到的突发情况并针对突发情况整理相应的处理措施，及时高效的保障系统的快速恢复；

## **CSF 调用讲解**

### **调用流程**

1. 发起请求调用SOA服务。在请求中带有BusiCode字段
2. 通过 BusiCode 去数据库查找其与服务编码（ CSF\_SERVICE\_CODE ）的的映射关系（映射关系在aicache中 Base 中心 有缓存，缓存ID： com.asiainfo.easyframe.sip.access.cache.ServiceCfgCacheImpl）
3. 通过 CSF\_SERVICE\_CODE 去 aicache 的 CSF 中心中查找服务编码对应的服务信息包括出入参版本号等信息。（aicache 中缓存的数据是 csf\_srv\_service\_info，csf\_srv\_service\_param，csf\_srv\_version等信息）
4. SOA 层调用中心层，SOA 通过 \*\*CsfClient.java 类进行调用（该类在编译时通过扫描对应包路径的代码生成）
5. CsfClient.java 类中有调用服务的服务编码信息，并把入参转成了map
6. 根据服务编码信息可以知道调用的是哪个中心服务。 如 ord\_com.asiainfo.crm.center.ord.inter.out.bboss.gbord.service.interfaces.IGbord2CustFSV\_CustomerInfoUp\_94d24ced7f940aed 可知中心为ord
7. 根据SOA启动参数appframe.server.name 去zk的 policies 目录中查找 soa 与 ord 集群的路由关系。（该关系可以去CSF\_CLIENT\_CLUSTER\_MAPPING库中配置，并在CSF服务路由界面同步数据到zk）
8. 集群信息配置可以去csf.xml 中配置。在中心实例启动时会根据 appframe.server.name 中的值去配置文件找到对应集群，将其ip和端口等信息注册到zk的 clusters 目录中。
9. 根据cluster 信息找到zk集群中找到对应的ip和端口，通过其ip和端口和服务编码进行调用服务的调用。
10. 在服务调用过程中会，中心会去CSF 的 aicache 查询该服务编码的出入参信息等，将传递来的数据反射

[PAAS平台紧急问题处理预案 1](#_Toc26374)

[1. FOOT平台 4](#_Toc17079)

[1.1 目的 4](#_Toc21973)

[1.2 应急预案 4](#_Toc15068)

[1.2.1 镜像仓库异常，无法启动 4](#_Toc15826)

[1.2.2 FOOT平台组件状态异常 5](#_Toc14141)

[1.2.3 pod无法拉起 5](#_Toc15044)

[1.2.4 Heketi服务挂掉，启动失败 6](#_Toc15564)

[1.2.5 镜像仓库空间被占满，无法上传镜像 7](#_Toc15628)

[1.2.6 应用访问失败，日志报unKownHostException 8](#_Toc25780)

[1.2.7 应用一直启动中，但是k8s集群中实际已经启动成功 9](#_Toc564)

[1.2.8 配置了外部存储的应用，应用一直启动中 9](#_Toc826)

[1.2.9 gfs主机上挂载点和应用的对应关系 10](#_Toc28361)

[1.2.10 日常维护常用命令 11](#_Toc11946)

[1.3 现场问题处理 12](#_Toc15373)

[2. CSF微服务管理平台 12](#_Toc28350)

[2.1 目的 12](#_Toc16463)

[2.2 CSF 调用讲解 12](#_Toc24604)

[2.2.1 调用流程 12](#_Toc3816)

[2.2.2 服务路由信息同步到zk 15](#_Toc13531)

[2.2.3 单服务超时时间配置 15](#_Toc21802)

[2.2.4 默认超时时间配置 16](#_Toc568)

[3. IFACE平台 17](#_Toc17352)

[3.1 目的 17](#_Toc3294)

[3.2 应急预案 17](#_Toc4221)

[3.3 现场问题处理 17](#_Toc4199)

[4. LOG4X平台 20](#_Toc14852)

[4.1 目的 20](#_Toc9225)

[4.2 应急预案 20](#_Toc18583)

[4.2.1 log4x突然无法查询到数据 20](#_Toc11659)

[4.2.2 各组件集群状态检查汇总 25](#_Toc22408)

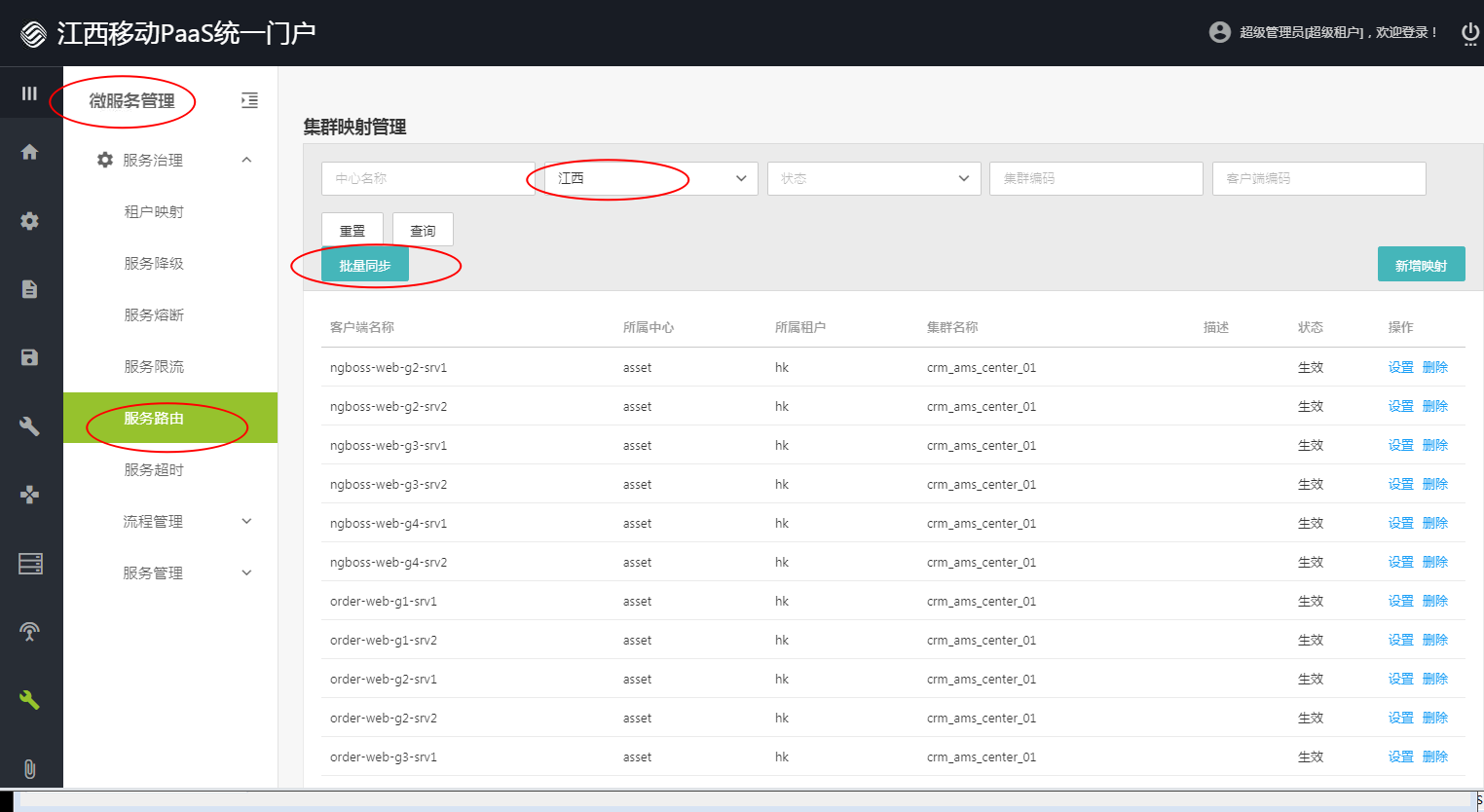
[4.3 现场问题处理 30](#_Toc7409)

[4.3.1 广西B域log4x生产环境业务轨迹办理查询速度过慢问题解决 30](#_Toc22726)

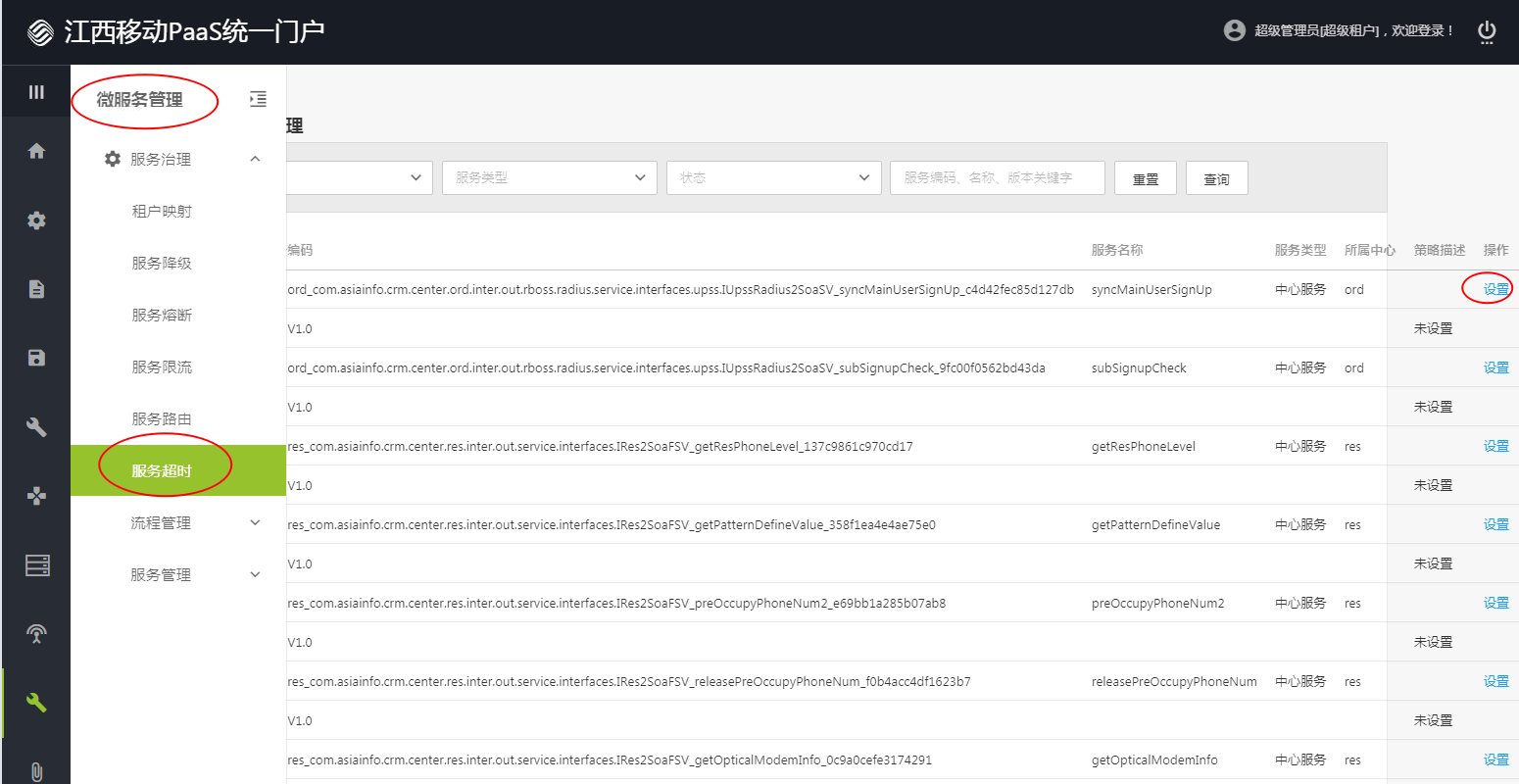
[4.3.2 通过traceid直接从hbase中查询数据，以json格式显示 34](#_Toc16005)

1. 为java类，进行方法调用。

### **服务路由信息同步到zk**

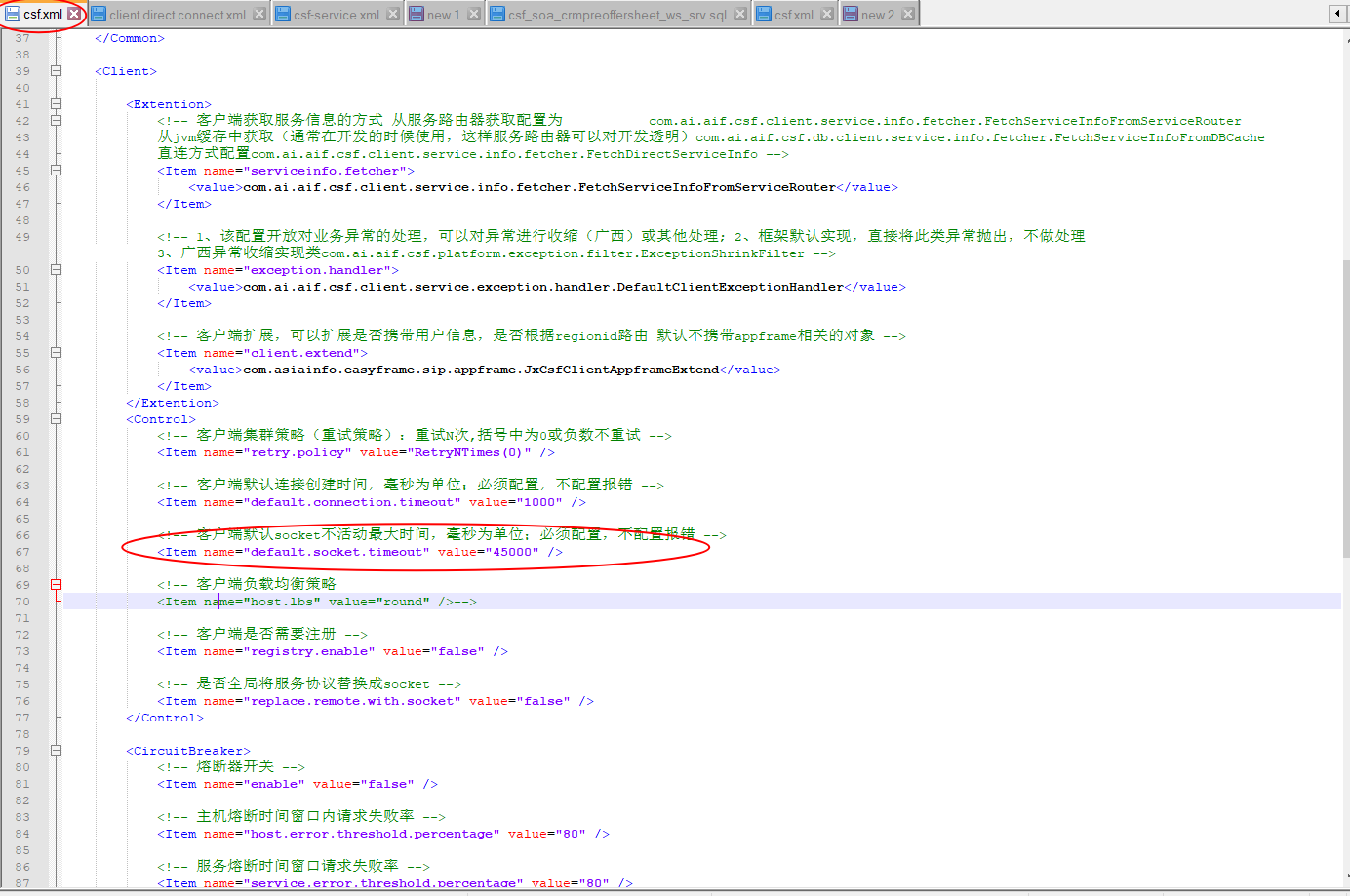


### **单服务超时时间配置**



或者 超时配置到 csf\_srv\_timeout 这个表然后刷新zk

### **默认超时时间配置**



# IFACE平台

## **目的**

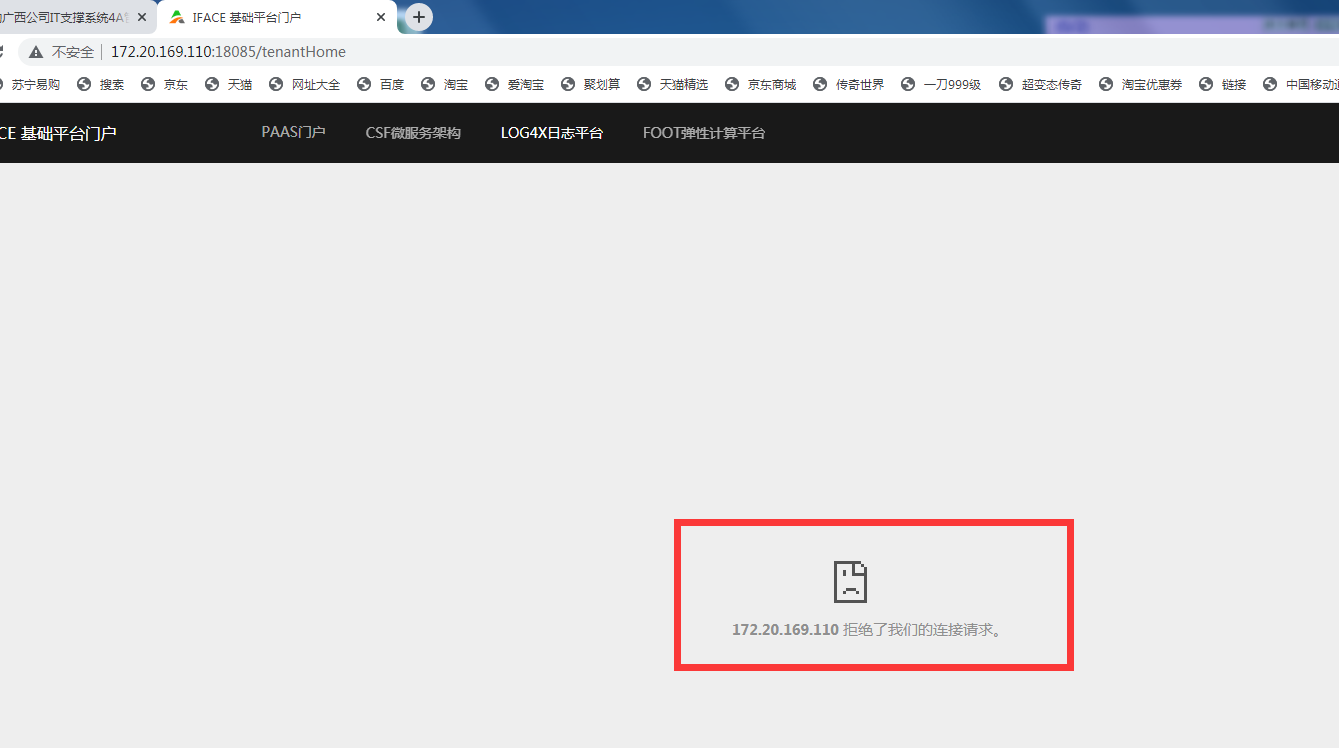
为了满足IFACE平台应对突发情况的能力，特整理一些可能遇到的突发情况并针对突发情况整理相应的处理措施，及时高效的保障系统的快速恢复；

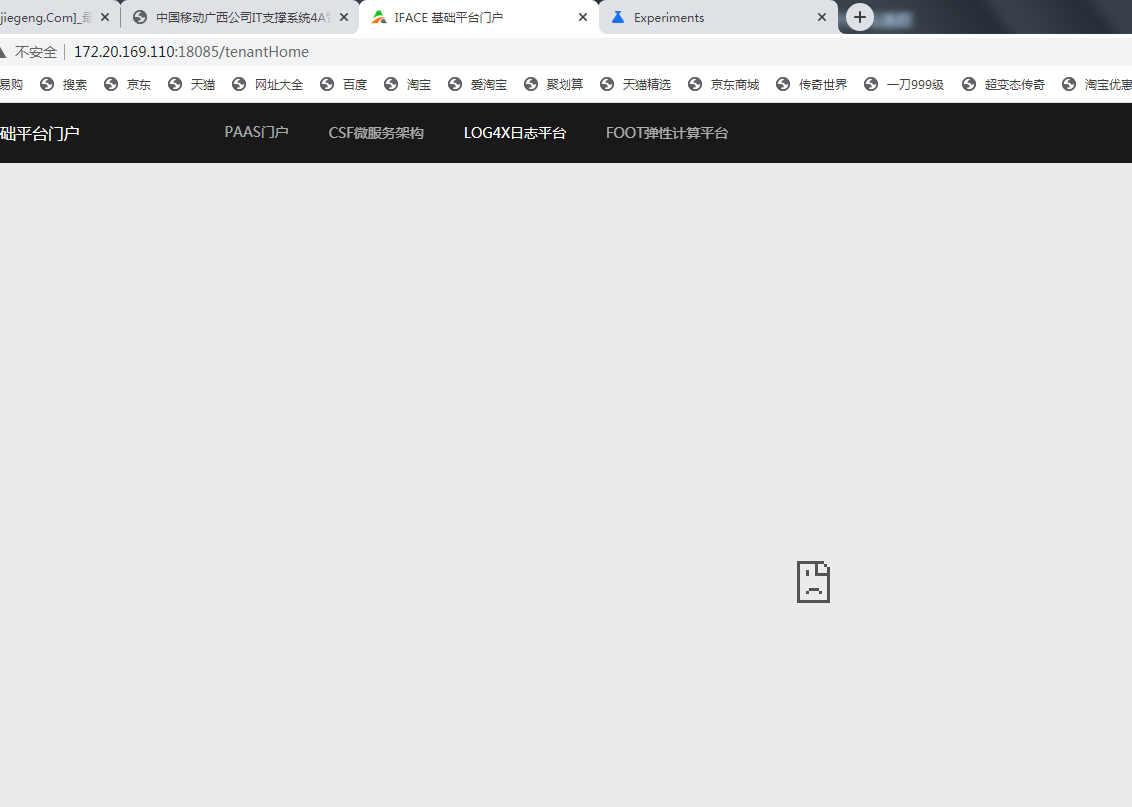
## 应急预案

## 现场问题处理

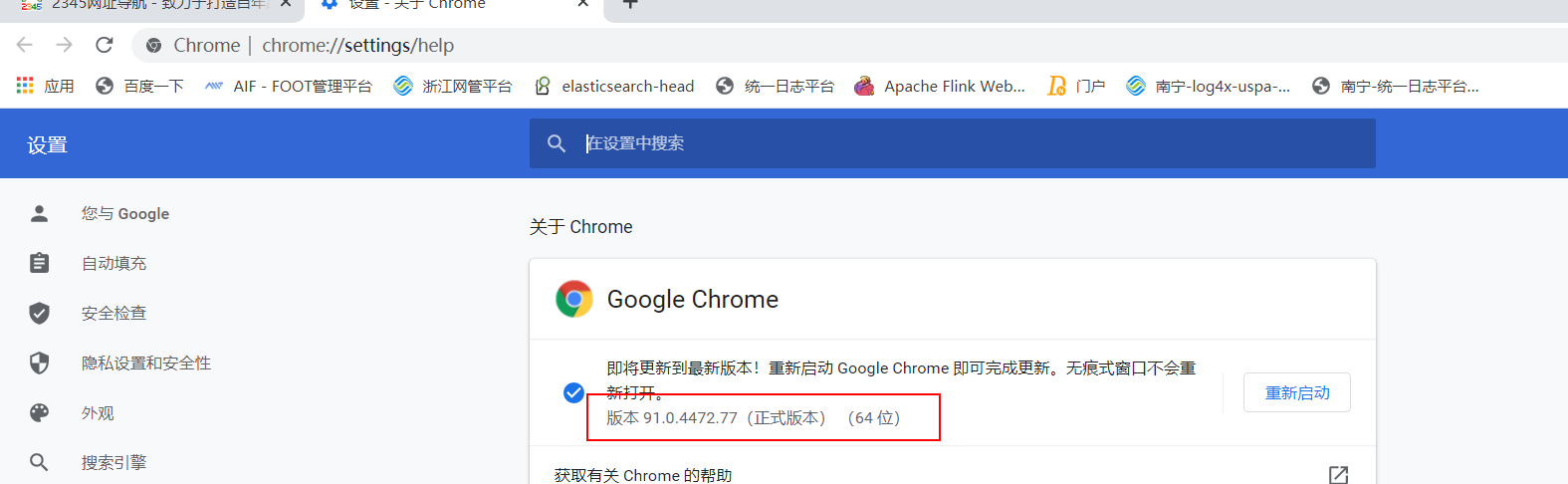
#### 谷歌浏览器无法跳转到其它模块。

广西B域连续两个月出现使用谷歌浏览器后从门户单点登录其他模块（foot,log4x,csf）出现类似拒绝错误：





首先确保打不开的模块前台,nginx是正常运行，日志无报错，首先排除本身环境因素,剩下原因要么是打开主机未开通网络策略或者浏览器本身问题，因同一网段有人能打开，排除网络策略，剩下是浏览器问题，更换火狐浏览器打开正常，谷歌浏览器f12检查后，百度报错，是谷歌浏览器设置问题，解决办法如下：



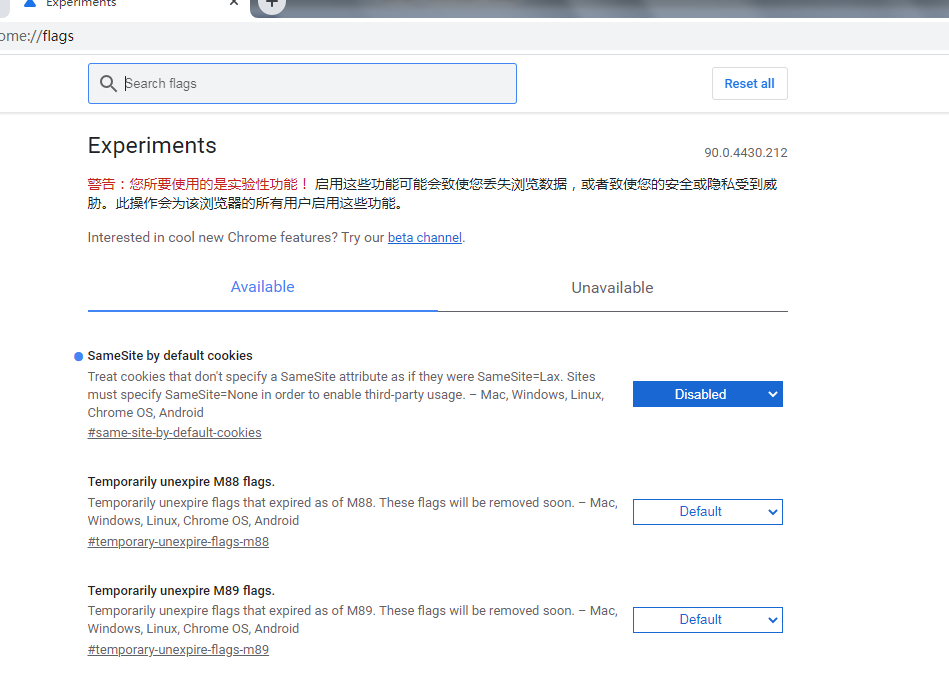
谷歌浏览器版本91开头解决本办法：



--disable-features=SameSiteByDefaultCookies,CookiesWithoutSameSiteMustBeSecure

谷歌浏览器版本91开头解决本办法：

1. 谷歌浏览器打开chrome://flags/  
   2.搜索samesite  
   3.SameSite by default cookies项设置为Disabled即可



# LOG4X平台

## **目的**

为了满足LOG4X平台应对突发情况的能力，特整理一些可能遇到的突发情况并针对突发情况整理相应的处理措施，及时高效的保障系统的快速恢复；

## **应急预案**

### log4x突然无法查询到数据

首先数据的流向 业务侧agent采集数据->写入kafka集群->flink集群读取处理->写入ES集群,hbase(日志数据量大时才用到，正常O域只写入ES，若使用hbase，es中存日志跟链路头，hbase存剩余链路日志)->页面查询

根据数据流向，正常页面查询不报错（页面直接查询es或者hbase），后台server也没报错，即可往上一个环节推：

#### ES

ES常用命令参考：

1、查看集群健康状态

curl -XGET http://192.168.199.133:9207/\_cluster/health?pretty=true

2、查看索引

curl -XGET http://192.168.199.133:9207/\_cat/indices\?v

3、查看某索引的settings

curl -XGET http://192.168.199.133:9207/log4x\_sql\_0/\_settings?pretty=true

4、查看未做分片的原因

curl -XGET http://192.168.199.133:9207/\_cat/shards?h=index,shard,prirep,state,unassigned.reason| grep UNASSIGNED

1）INDEX\_CREATED：由于创建索引的API导致未分配。

2）CLUSTER\_RECOVERED ：由于完全集群恢复导致未分配。

3）INDEX\_REOPENED ：由于打开open或关闭close一个索引导致未分配。

4）DANGLING\_INDEX\_IMPORTED ：由于导入dangling索引的结果导致未分配。

5）NEW\_INDEX\_RESTORED ：由于恢复到新索引导致未分配。

6）EXISTING\_INDEX\_RESTORED ：由于恢复到已关闭的索引导致未分配。

7）REPLICA\_ADDED：由于显式添加副本分片导致未分配。

8）ALLOCATION\_FAILED ：由于分片分配失败导致未分配。

9）NODE\_LEFT ：由于承载该分片的节点离开集群导致未分配。

10）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）。

11）REROUTE\_CANCELLED ：作为显式取消重新路由命令的结果取消分配。

12）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。

curl -XGET http://192.168.199.133:9207/\_cluster/allocation/explain?pretty

5、删除副本分片

curl -H "Content-Type: application/json" -XPUT http://192.168.199.133:9207/log4x\_default\_0/\_settings -d '{"number\_of\_replicas" : "0"}'

6、删除索引

curl -XDELETE -u elastic:changeme http://192.168.199.133:9207/log4x\_default\_0

#### KAFKA

Kafka检查

基本操作是先检查server.log,这边报错基本能百度，如无报错，执行下一步，检查kafka集群topic是否正常读取数据，命令如下：

【读取消息命令】

[log4x@master2 kafka\_2.11-0.10.2.1]$ bin/kafka-console-consumer.sh --bootstrap-server nm-log4x-1:9092, master2:9092, slave1:9092 --topic LOG4X-TRACE-TOPIC（链路）/LOG4X-LOG-TOPIC(普通日志)

若不能读取数据，自己造数据测试下kafka集群是否正常，命令如下：

【发送消息命令】

[log4x@nm-log4x-1 kafka\_2.11-0.10.2.1]$ bin/kafka-console-producer.sh --broker-list nm-log4x-1:9092, master2:9092, slave1:9092 --topic test

在控制台输入一些消息，按ctrl+c退出发送。

【读取消息命令】

[log4x@master2 kafka\_2.11-0.10.2.1]$ bin/kafka-console-consumer.sh --bootstrap-server nm-log4x-1:9092, master2:9092, slave1:9092 --topic test --from-beginning

控制台中看到之前发送的消息则证明kafka配置成功，按ctrl+c退出控制台

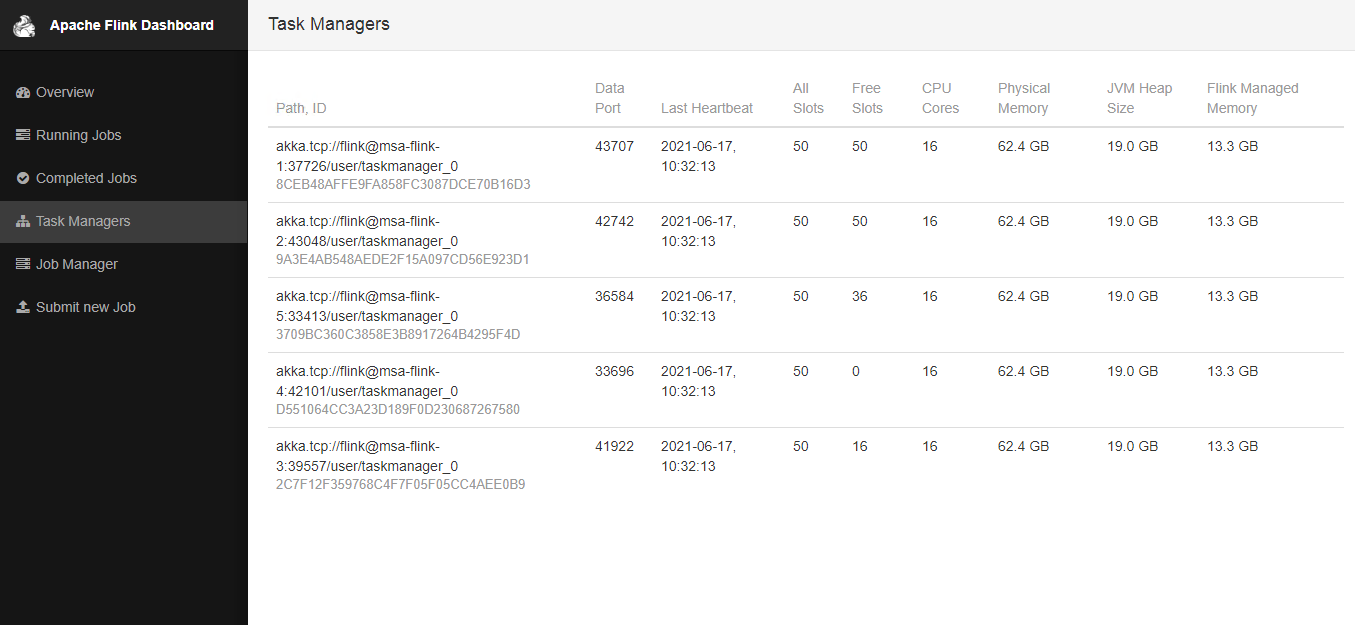
这部若成功，代表kafka集群读写正常，是没有数据写入，直接找业务侧，问他们业务是否正常，是否往kafka写入数据。

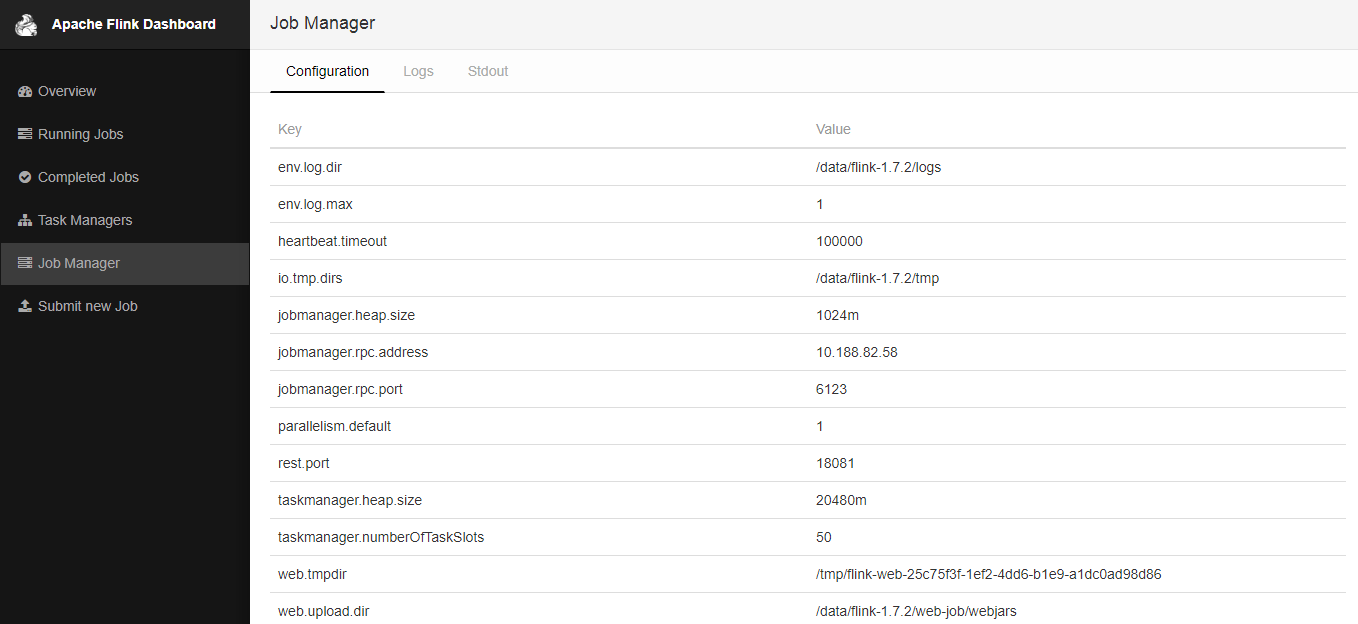
能正常读取数据，下一步检查flink。

#### FLINK

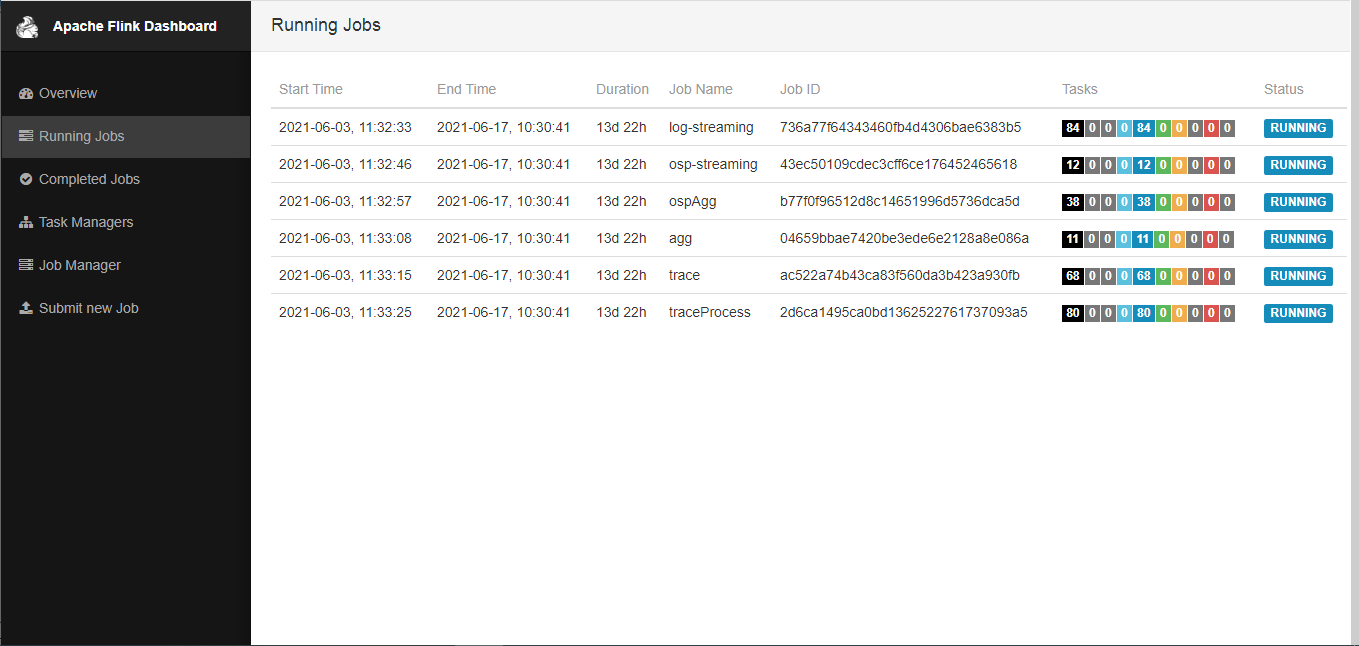
flink检查，检查点如下：

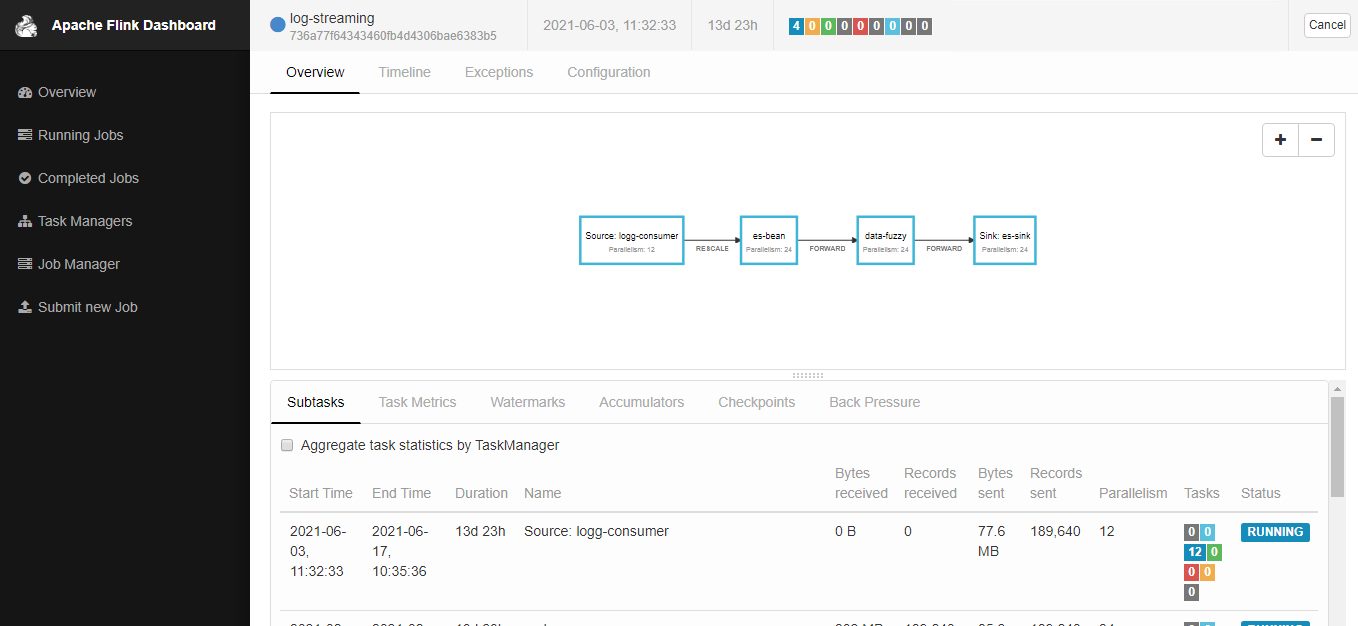
1. 监控页面工作,管理节点是否正常





1. 监控页面job是否启动，是否运行正常





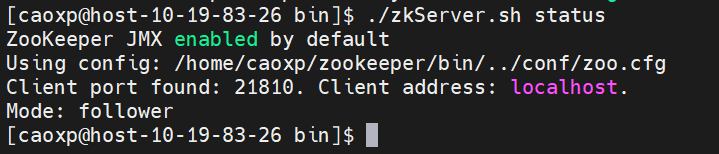
若flink运行不正常，检查后台task日志，查询具体错误，若flink运行正常，job也运行正常，排查下处理业务对应的job是否启动，到这边基本问题都能解决。

### 各组件集群状态检查汇总

#### .ZK集群状态检测

##### **集群状态检查**

对应的zk bin目录下执行./zkServer.sh status可查询当前节点zk的状态及角色信息。



##### **集群日志检查**

ZK日志查询/zookeeper/log下能看到zk启动打印信息。

#### 、Kafka集群状态检测

##### **、创建Topic检查**

对应的kafka bin目录下执行

bin/kafka-topics.sh --zookeeper 10.19.83.25:21810,10.19.83.26:21810,10.19.83.27:21810,10.19.83.28:21810,10.19.83.30:21810 –create topic test --partitions 3 –replication-factor 3

看看能否正常创建topic

##### **、Topic信息检查**

也可以通过下面命令查询当前集群的topic信息。

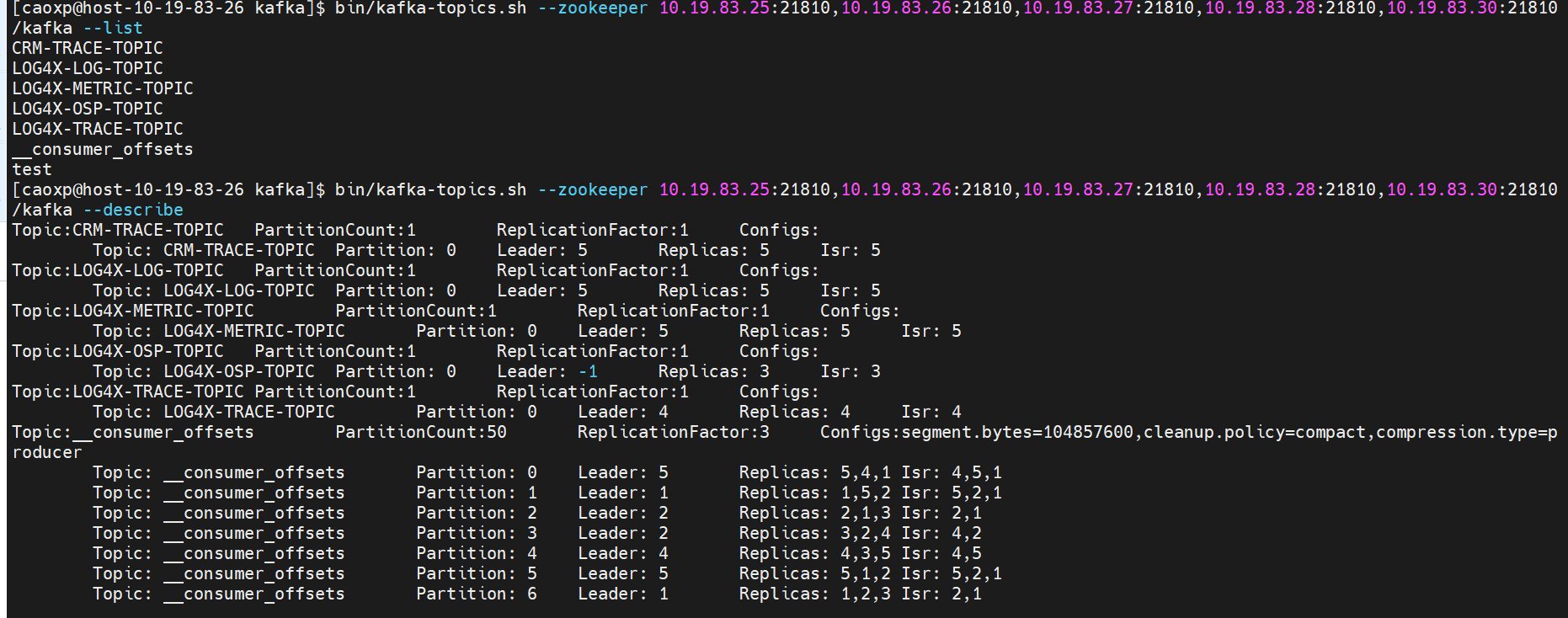
bin/kafka-topics.sh --zookeeper 10.19.83.25:21810,10.19.83.26:21810,10.19.83.27:21810,10.19.83.28:21810,10.19.83.30:21810/kafka –list

或者

bin/kafka-topics.sh --zookeeper 10.19.83.25:21810,10.19.83.26:21810,10.19.83.27:21810,10.19.83.28:21810,10.19.83.30:21810/kafka –describe

##### **、日志检查**

创建失败或者kafka集群启动失败可以查询日志来定位，路径为kafka/log/kafkaServer.out



#### 、ES集群状态查询

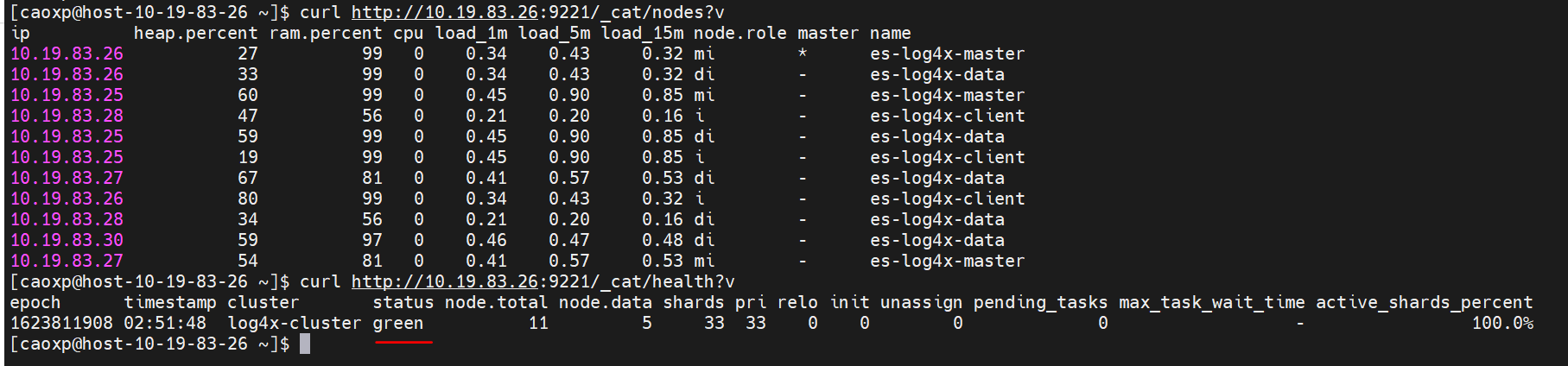
##### **、登录节点执行**

curl <http://10.19.83.26:9221/_cat/nodes?v> 查看各节点信息

curl <http://10.19.83.26:9221/_cat/health?v> 查看集群状态信息

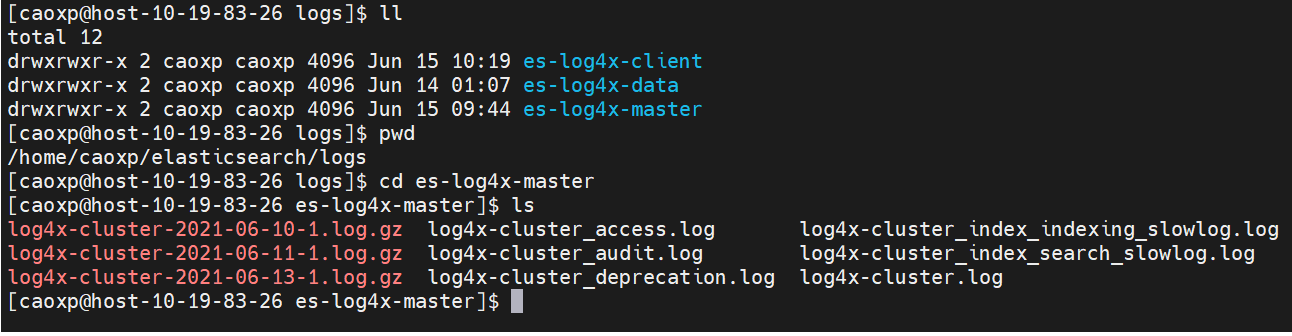
curl -XGET 'http://10.19.83.26:9221/\_cluster/health?pretty'

其中9221为cliet的http端口。



##### **、ES日志日志检查**

(参考配置文件elasticsearch.yml中的path.logs)



#### 、Hadoop集群状态检测

##### **、日志检查**

若日志打印中无eroor

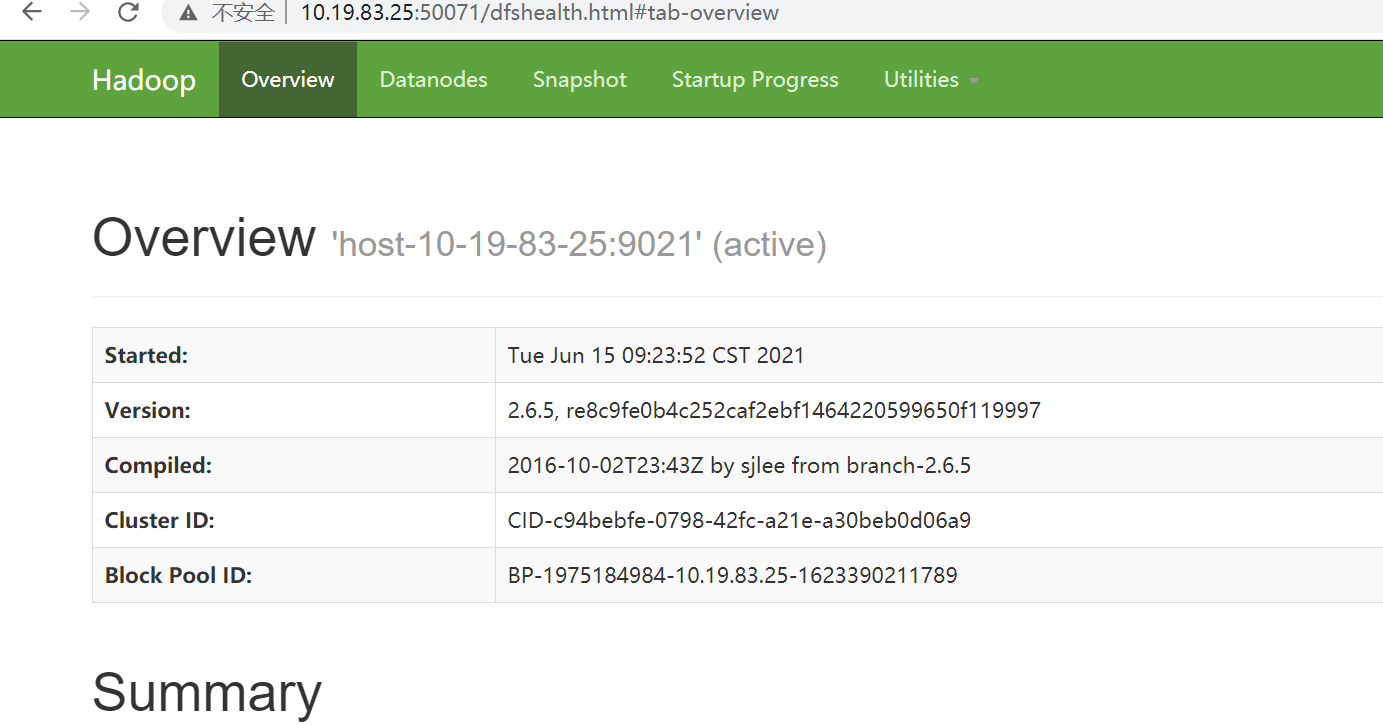
##### **、页面检查hadoop集群状态**

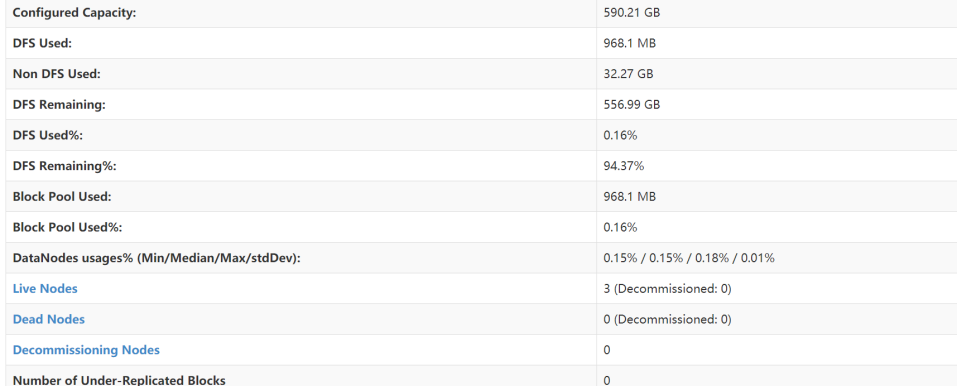
<http://10.19.83.26:50071/dfshealth.html>

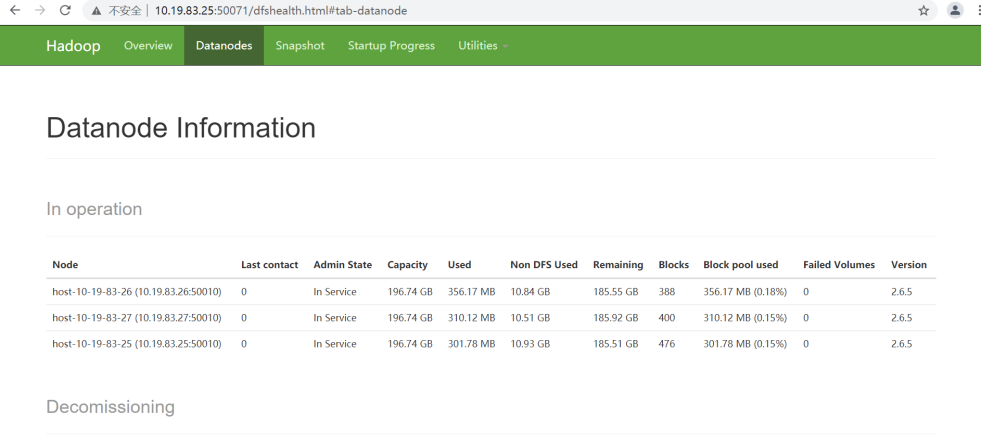
其中50071为namenode的http端口，

overview界面可以看到当前节点状态为active还是standby，以及live nodes个数。Datanode界面能查询到datanode节点信息。

在hadoop-env.sh中定义了HADOOP\_LOG\_DIR，集群启动过程可以查看日志有无异常。







#### 、HBase集群状态检查

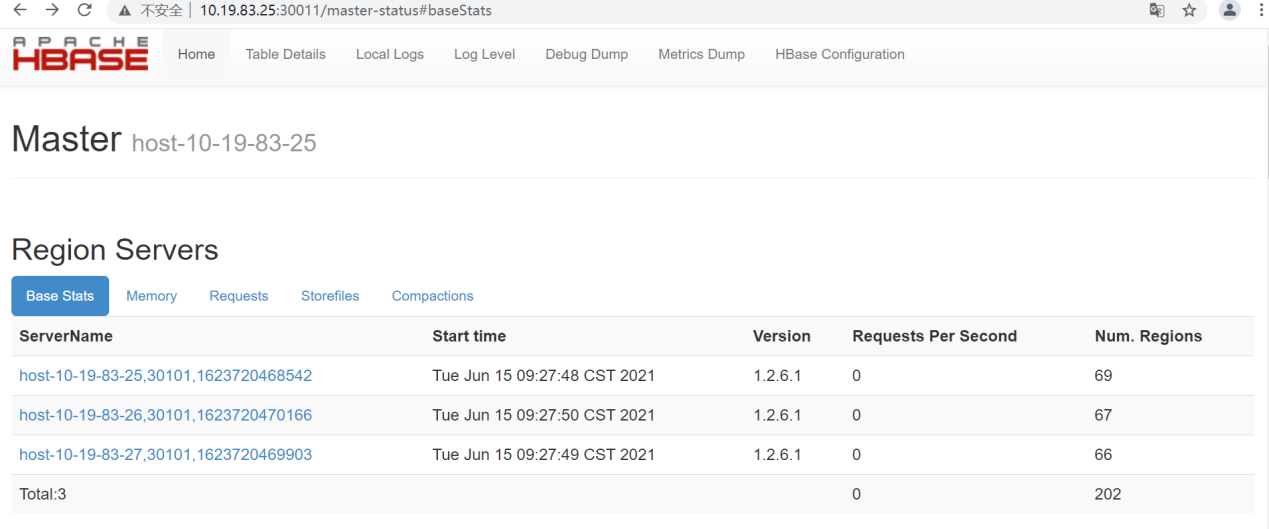
##### **、日志检查**

Hbase集群部署完成后，Hbase日志路径为hbase-env.sh中设置的HBASE\_LOG\_DIR路径，启动过程可以查看日志有无异常。

##### **、页面检查**

浏览器登录master节点<http://10.19.83.25:30011/master-status>

其中30011为配置文件中master.info.port端口。



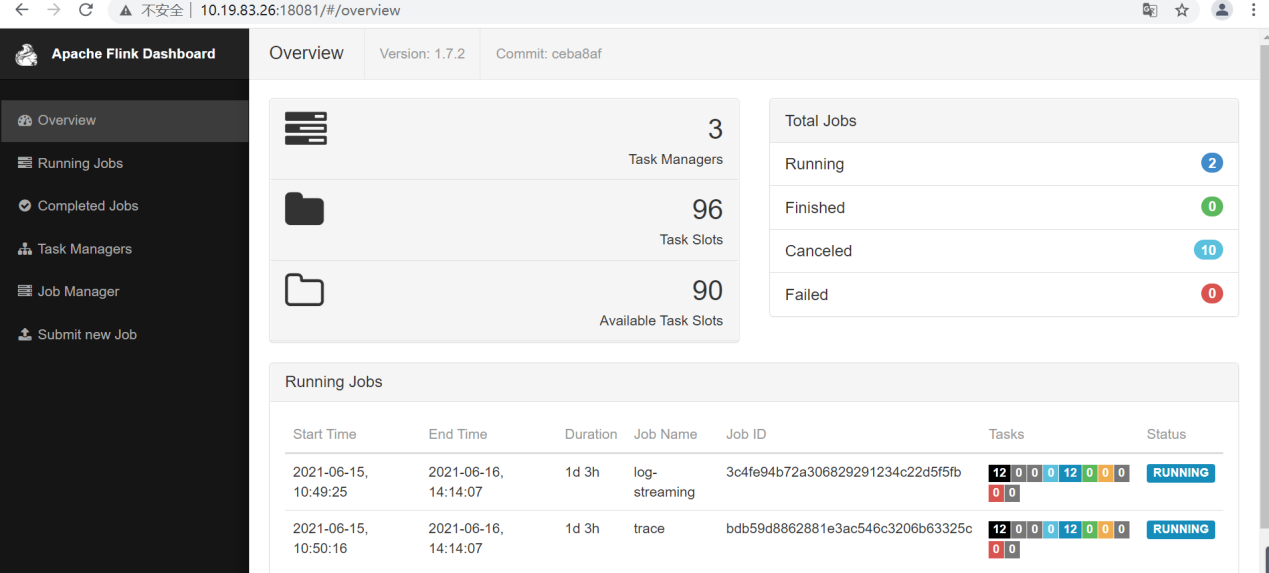
#### 、Flink集群状态检测

##### **、日志检查**

日志路径参考flink-conf.yaml的env.log.dir查询集群启动打印日志。

##### **、页面检查**

Flink集群部署完后可登录web界面查看集群状态 <http://10.19.83.26:18081/#/overview>

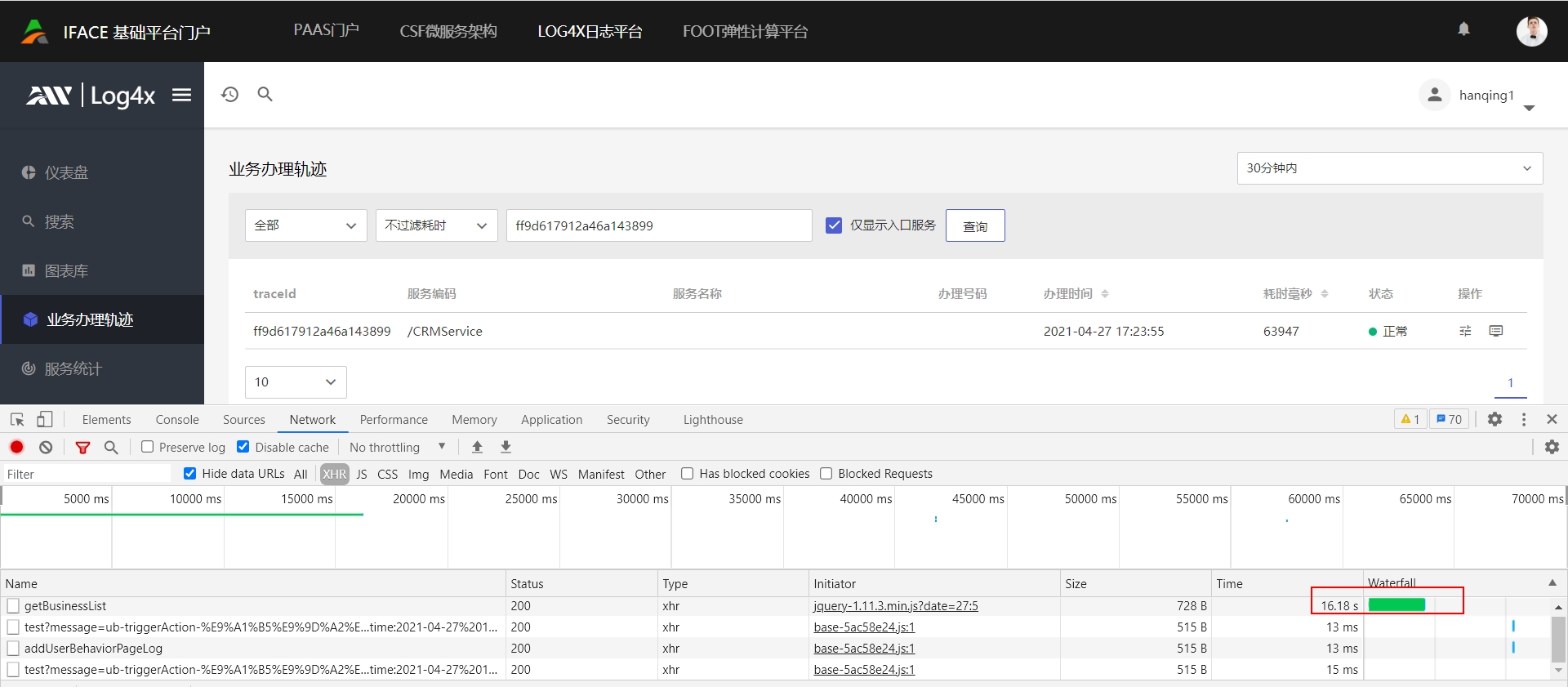


## **现场问题处理**

### 广西B域log4x生产环境业务轨迹办理查询速度过慢问题解决

#### 问题描述：

4.26日客户反馈log4x生产环境业务轨迹办理查询速度根据traceid查询过慢（时间在12S以上），点击打开链路也很慢（7s以上），详情见下图：





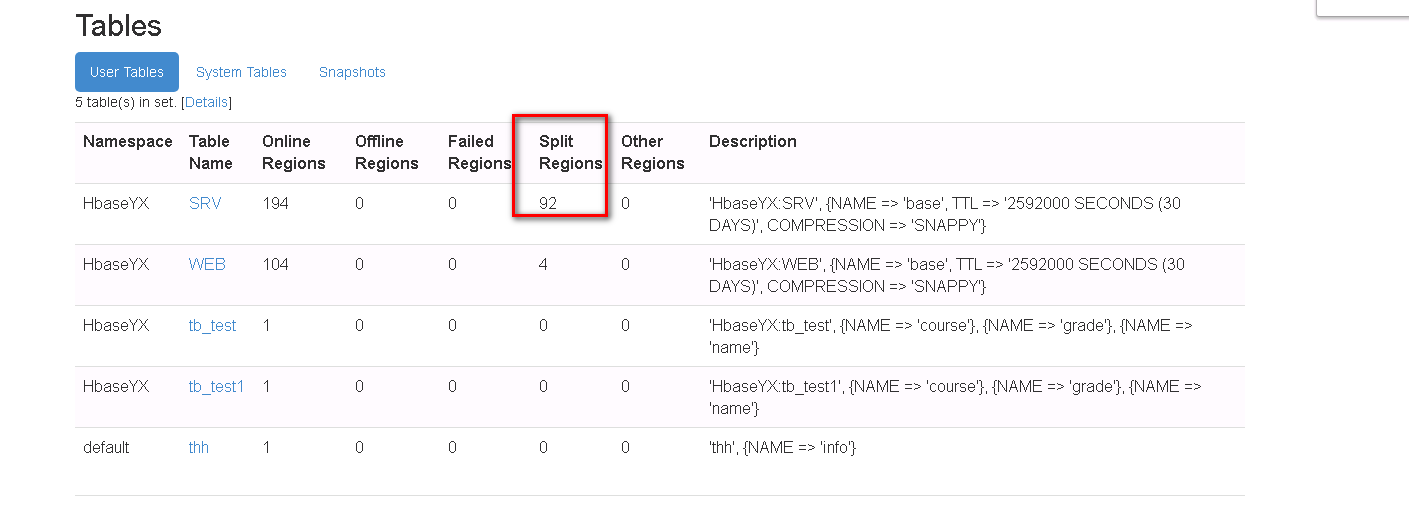
经开发定位存在两个问题：

##### **索引轮转失败，导致单个索引过大（正常轮转单个索引大小在**10**G），数据量高达**900**G，导致查询变慢。**



轮转具体失败原因怀疑跟3月下旬，es集群挂掉有关。

##### **hbase表建表时没有设置region数，数据量上来后导致表频繁的分裂，分裂会影响效能，最终查询链路时非常慢。**



#### 解决方案：

##### **索引轮转失败解决办法**

1. 首先停止log4x-scheduler



1. 登录log4x数据库,清空qrtz表



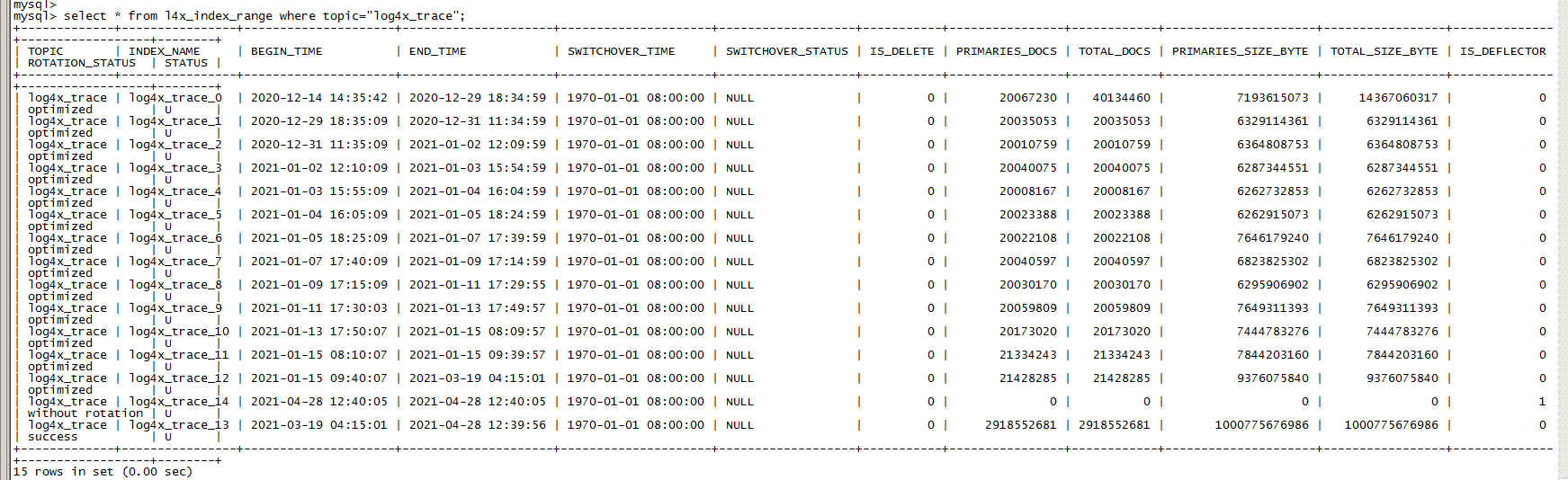
1. 启动log4x-scheduler



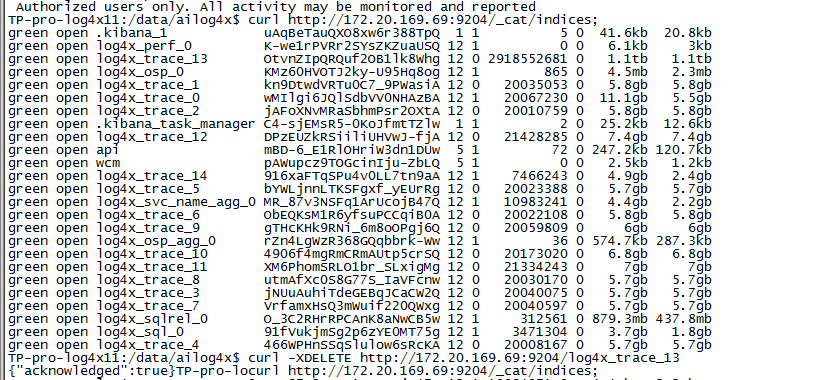
1. 根据轮转策略，查看当前索引是否轮转



1. 删除旧数据，首先删除l4x\_index\_range中对应的过大索引，然后es中删除对应的索引（这步选做，如果需要保留数据可以不操作）。

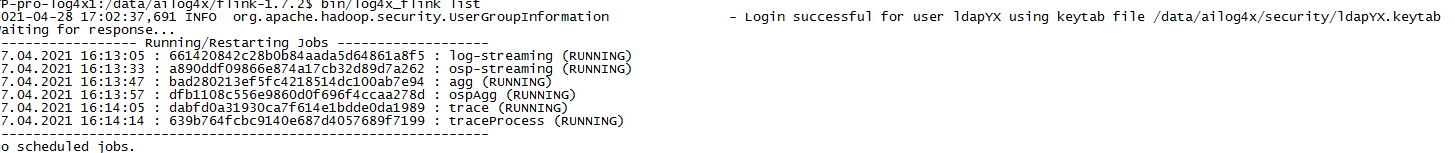






##### **hbase表建表时没有设置region数，数据量上来后导致表频繁的分裂，这个没办法再设置region数，只能删表重新建。**

1. 停掉flinkjob





1. 登录hbase客户端，删除hbase表，

disable ‘HbaseYX:SRV’

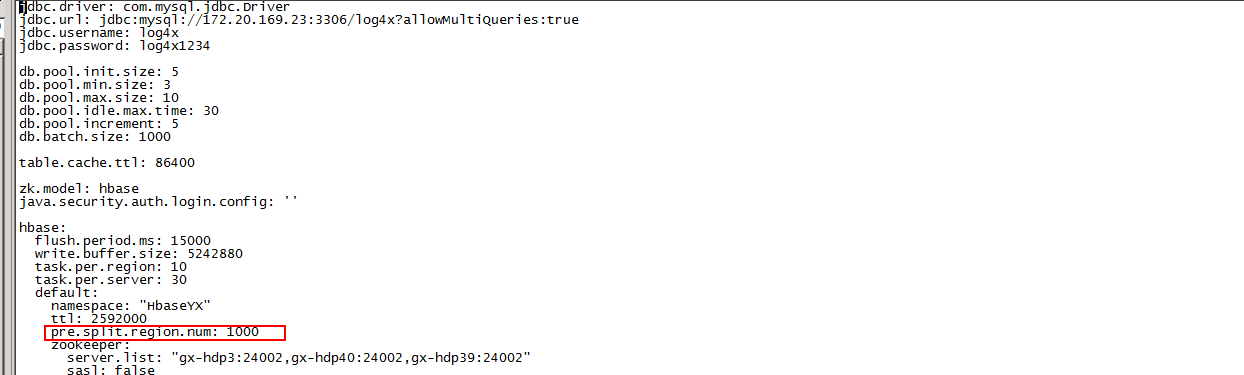
drop ‘HbaseYX:SRV’

disable ‘HbaseYX:WEB’

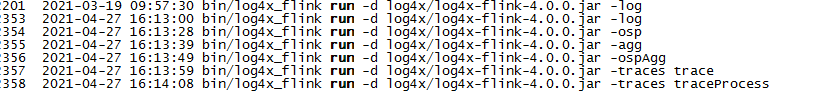
drop ‘HbaseYX:WEB’

1. 修改flink中的job包修改以下参数（正常应该在200以上）：



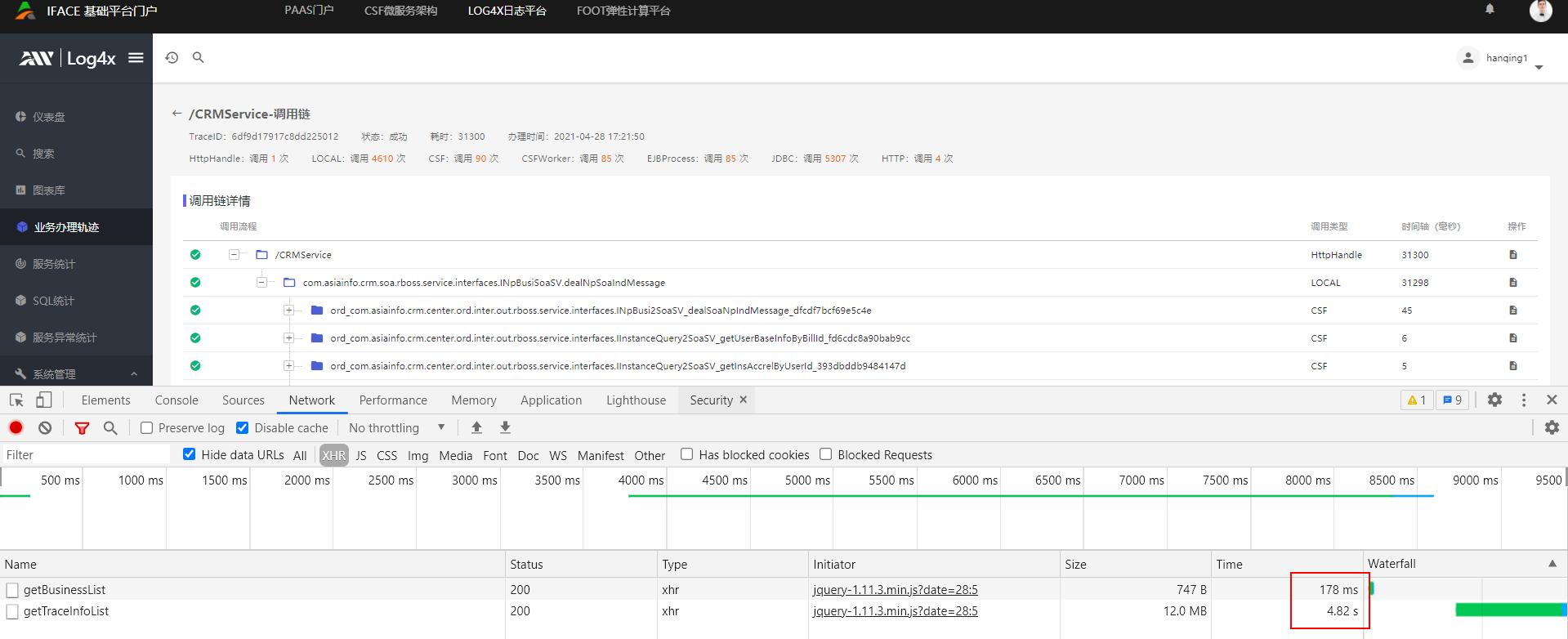


1. 启动flink-job



#### 问题解决

两种查询都不再慢，都在正常查询范围内。



### 通过traceid直接从hbase中查询数据，以json格式显示

#### 问题描述

从页面直接查询较慢，通过脚本的方式，从hbase中直接查询某个特定tarceid的调用信息

#### 问题解决方式01

##### **.log**4**x.sh脚本文件说明：**

文件放至目录hbase的bin目录下：脚本主要查询traceid的调用链数据，以json的格式将数据存放在log4x.json文件中。

##### **.env.ini文件说明：**

输出的结果过滤字段，在想要过滤的字段标记false，文件放至目录hbase的bin目录下

#!/bin/bash

#输入traceid，查询调用链数据；

read -p "输入traceid:" x;

#输入查询语句并格式化输出结果，将结果存放至logx.json文件；

echo "scan 'cx\_log4x:SRV',{STARTROW=>'$x',STOPROW=>'$x,z'}"|./hbase --config

../conf shell|awk -F "value=" '{print $2 "\n"}'|awk -F ',"' '{for

(i=1;i<=NF;i++){if (i!=NF&&i!=1)print "\t""\""$i,",";else if (i==1)print "

{","\n""\t"$i,",";else print "\t""\""$i,"\n""}""\n"}}'|tee log4x.json

q=0;

#根据env.ini配置文件的过滤字段筛选更新json文件，不需要筛选可以注释掉；

for j in `cat env.ini|grep false|awk -F "=" '{print $1}'`;do s[q++]=$j;done;for

p in ${s[\*]};do sed -i "/\"$p\"/d" log4x.json;done

echo "==============================log4x.json======================"

cat log4x.json

regionCode=false

serverIp=true

retCode=false

serverPort=true

channelCode=false

hostName=true

sysCode=true

msgType=false

ps=false

rpc=false

appName=true

center=false

procId=false

channel=false

pid=false

serviceName=true

callType=false

spec=true

caller=false

probeType=false

success=true

dockerId=true

msgTime=true

##### 3**.输出结果log**4**x.json示例（根据env.ini文件已过滤部分字段）**

startTime=true

id=true

sn=false

elapsedTime=true

==============================log4x.json=====================

{

{"traceId":"0b2c317a13b899825362" ,

"serverIp":"10.1.2.3" ,

"serverPort":"6998" ,

"hostName":"10.19.83.30" ,

"sysCode":"NGCRM" ,

"appName":"DEFAULT\_SERVER\_NAME\_1623828438643" ,

"serviceName":"com.ai.common.service.base.impl.BaseSVImpl.getAllBsCenter" ,

"spec":{} ,

"success":true ,

"dockerId":"host-10-19-83-30" ,

"msgTime":1623828502916 ,

"startTime":1623828502915 ,

"id":"0a20f17a13b899835378" ,

"elapsedTime":0}

}

{

{"traceId":"0b2c317a13b899825362" ,

"serverIp":"10.1.2.3" ,

"serverPort":"6998" ,

"hostName":"10.19.83.30" ,

"sysCode":"NGCRM" ,

"appName":"DEFAULT\_SERVER\_NAME\_1623828438643" ,

"serviceName":"com.asiainfo.crm.so.teaminvoke.out.service.impl.CrmQrySVImpl.quer

yNextPeriodInfo" ,

"spec":{} ,

"success":true ,

"dockerId":"host-10-19-83-30" ,

"msgTime":1623828502916 ,

"startTime":1623828502915 ,

"id":"2560c17a13b899835373" ,

"elapsedTime":0}

}

{

{"traceId":"0b2c317a13b899825362" ,

"serverIp":"10.1.2.3" ,

"serverPort":"6998" ,

"hostName":"10.19.83.30" ,

"sysCode":"NGCRM" ,

"appName":"DEFAULT\_SERVER\_NAME\_1623828438643" ,

"serviceName":"com.ai.omframe.instance.service.impl.InstanceQrySVImpl.getInstUse

rByBillId" ,

"spec":{"sql":"select \* from (SELECT A.ACCT\_ID, A.CUST\_ID, A.ACCT\_NAME,

A.ACCT\_TYPE, A.ACCT\_STATUS, A.ACCT\_PASSWORD, A.CREDIT\_LEVEL, A.CREDIT\_VALUE,

A.URGE\_STOP\_FLAG, A.STATE, A.CREATE\_OP\_ID, A.CREATE\_ORG\_ID, A.OP\_ID, A.ORG\_ID,

A.DONE\_CODE, A.CREATE\_DATE, A.DONE\_DATE, A.EFFECTIVE\_DATE, A.EXPIRE\_DATE,

A.REGION\_ID, A.REMARKS, B.ACCT\_SUB\_TYPE, B.MAIN\_PHONE, B.CONT\_ZIPCODE,

B.CONT\_ADDRESS, B.ACCT\_SERVICE\_ID,

B.POST\_TYPE,B.EMAIL\_ADDRESS,\x5Cu0027\x5Cu0027 AS RESYN\_YYYYMM,C.PAYMENT\_ID,

C.PAY\_METHOD, C.BANK\_CONTRACT, C.PAYMENT\_BANK\_CODE, C.PAYMENT\_ACCOUNT,

C.PAYMENT\_ACCOUNT\_NAME, C.PAYMENT\_ACCOUNT\_TYPE, C.PRIORITY,

D.SMALL\_ACCT\_BANK\_NUM, D.PAY\_REMIND, D.PAY\_PHONE, D.STOP\_REMIND,

D.COLLECTION\_REMIND FROM CM\_ACCOUNT A, CMX\_ACCOUNT\_EXT B, CM\_ACCT\_PAYMENT C,

CMX\_ACCT\_PAYMENT\_EXT D WHERE A.ACCT\_ID \x5Cu003d B.ACCT\_ID AND A.ACCT\_ID

\x5Cu003d C.ACCT\_ID(+) AND C.PAYMENT\_ID \x5Cu003d D.PAYMENT\_ID(+) order by

A.CREATE\_DATE desc ) where 1\x5Cu003d1 AND ACCT\_ID \x5Cu003d ?"} ,

"success":true ,

"dockerId":"host-10-19-83-30" ,

"msgTime":1623828502916 ,

"startTime":1623828502915 ,

"id":"6fd1f17a13b899835388" ,

"elapsedTime":0}

}

{

{"traceId":"0b2c317a13b899825362" ,

"serverIp":"10.1.2.3" ,

"serverPort":"6998" ,

"hostName":"10.19.83.30" ,

"sysCode":"NGCRM" ,

"appName":"DEFAULT\_SERVER\_NAME\_1623828438643" ,

"serviceName":"com.ai.common.dao.base.impl.BaseDAOImpl.getAllBsCenter" ,

"spec":{} ,

"success":true ,

"dockerId":"host-10-19-83-30" ,

"msgTime":1623828502916 ,

"startTime":1623828502915 ,

"id":"a897217a13b899835383" ,

"elapsedTime":0}

}

{

{"traceId":"0b2c317a13b899825362" ,

"serverIp":"10.1.2.3" ,

"serverPort":"6998" ,

"hostName":"10.19.83.30" ,

"sysCode":"NGCRM" ,

"appName":"DEFAULT\_SERVER\_NAME\_1623828438643" ,

"serviceName":"com.asiainfo.crm.so.teaminvoke.out.service.impl.CrmFSVImpl.getUse

rInfoByBillId" ,

"spec":{} ,

"success":true ,

"dockerId":"host-10-19-83-30" ,

"msgTime":1623828502916 ,

"startTime":1623828502915 ,

"id":"ab73d17a13b899835368" ,

"elapsedTime":0}

}

#### 问题解决方式02

##### 参数说明

# 1.-DtraceId:调用链id

# 2.-Dtable:hbase中的表名

# 3.-Dhost:hbase挂在的zk地址

# 4.-Dfield:需要展示的列，如果需要全部展示则为""即可 hostName sysCode msgType ps procId channel pid retCode serverPort callType spec regionCode dockerId startTime id sn channelCode traceId appName center serviceName caller probeType success serverIp elapsedTime

# 5.-Dsuccess:查询调用连为false的节点,当需要查询为success为false的时候则可以改参数为-Dsuccess=false 为""时则查询所有节点

# 6.-DelapsedTime:查询时间超过设置的值的调用连节点，为""则不会去按时间过滤

##### 执行命令

java -DtraceId=b315917a0e5722b81000 -Dtable=cx\_log4x:SRV -Dhost=10.19.83.25:21810 -Dfields="" -DelapsedTime="" -Dsuccess="" -jar hbase-0.0.1-SNAPSHOT.jar &

##### 用到的依赖包