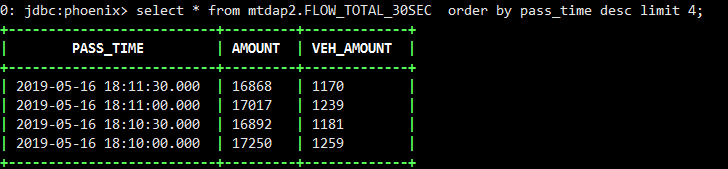
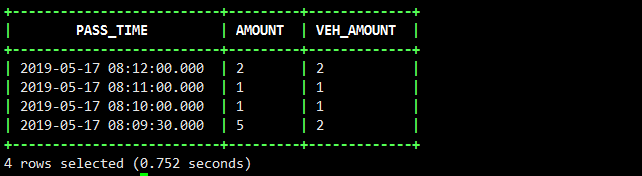
# Flink 常见异常整理

## 1:实时计算结果运行一段时间不正确

刚发布时是正常的:

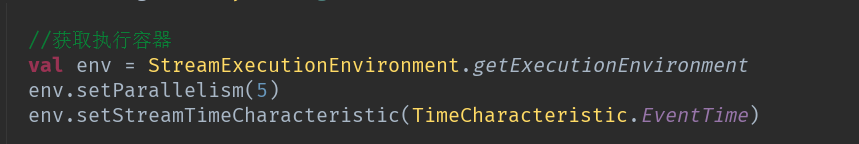


运行一段时间出现:



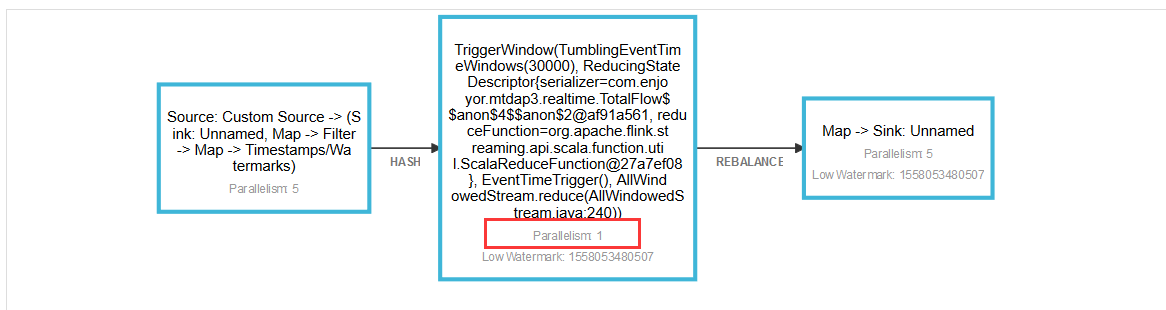
### 思路:资源不够

代码中设置并发度5



效果立竿见影,但第二天发现 问题仍然存在,只会异常出现时间延长了

查看运行状况

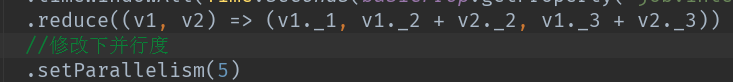


接收和输出的都是5个slot ~~但是主要用作计算的算子还是使用1个并行度~~

(主要进行计算的是第一个步骤,需要12个slot)

Api的优先级: 设置parallelism有多中方式，优先级为api>env>p>file

尝试在代码中给Reduce算子设置并行度5



重新发布下,抛:

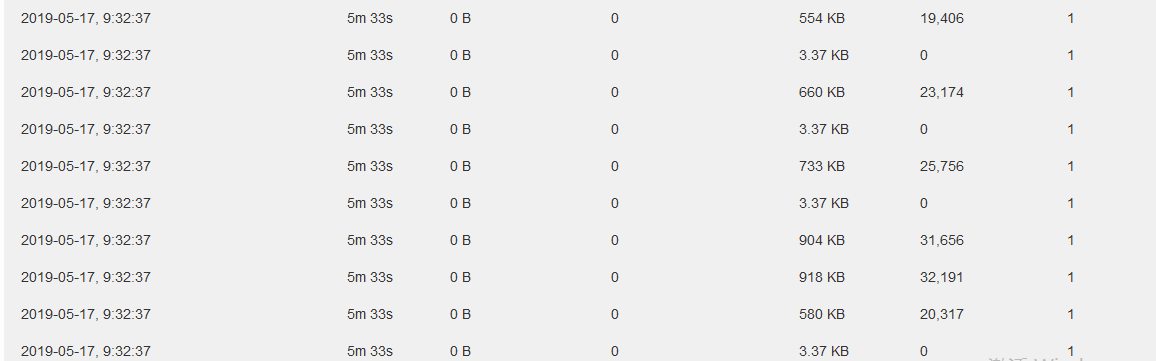


原因是flink的~~reduce~~ timeWindowAll算子为非并行度算子 只能设置为1 (有误, timeWindowAll是non-keyed window 非并行的,只能给定一个算子)

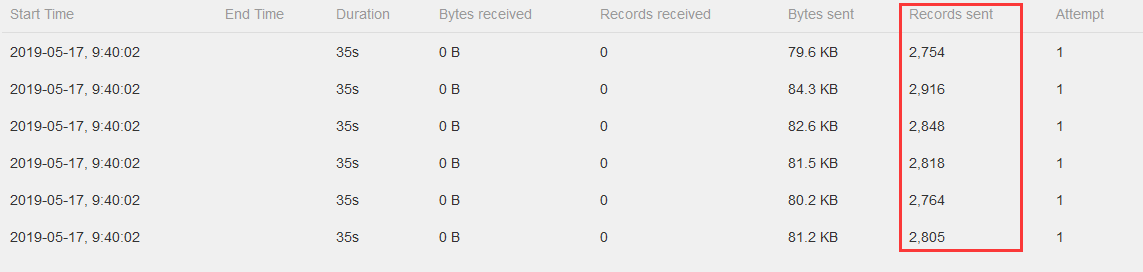
加大资源,设置并行度为10 ,指定keyBy(0) 按照过车时间分流

发布后查看Subtasks,发现数据倾斜,可以通过DataStream.rebalance解决

~~(可以规避,引入新的问题)~~

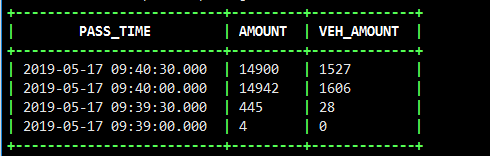


并行度设置为6,重新发布下,没有数据倾斜的现象(跟CPU内核有关)



重新发布的时候,这些计算量小的是怎么产生的?

上面两个是最新发布计算产生的,下面两个是关闭的时候?或者那时候已经计算异常了



原因: 程序采用事件时间 即事件发生的时间,这样避免网络延迟对数据的影响

但是存在设备采集硬件的故障问题,传过来过车时间有过去某一个点的,这些时间进入到窗口中,水位线被触发输出(解决方式:1过车时间不符合规则的过滤掉 2:判断过车时间是程序发布的N分钟范围外的不进行输出 8:00程序启动,pass\_time字段时间戳跟程序启动的时间戳对比,<的过滤掉)

### 代码优化

代码中将设置并行度去掉 同时在程序发布的时候使用-p 的方式指定并行度

同时使用keyBy算子按照过车时间对过车数据数据流进行分流操作

这种方式实现的缺陷:

1:当数据量继续增加或者是计算周期中逻辑复杂会导致该周期计算数据偏小,长时间运行 会导致 数据滞后 几天需要重启一下 参见21 37

2:正常运行的突然数据滞后或统计结果偏低是资源被其他程序争夺导致的

后续尝试:

1:使用rebalance 解决数据倾斜的问题,但是引入反压问题

2:先将过车时间取余,再keyBy(0),然后将程序的单并行窗口改为并行窗口 增加资源 (30slot)

多种方式对比结论: 步骤越少 问题越少,去掉rebalance 多个keyBy map(时间取余) timeWindows 🡪 timeWindowAll

采用60s+12slot+180s(调参) +下图(注释掉的是尝试的步骤)



## 2: Timeout expired while fetching topic metadata

这种bug出现的原因:连接kafka/redis异常,多见于认证失败/连接错误

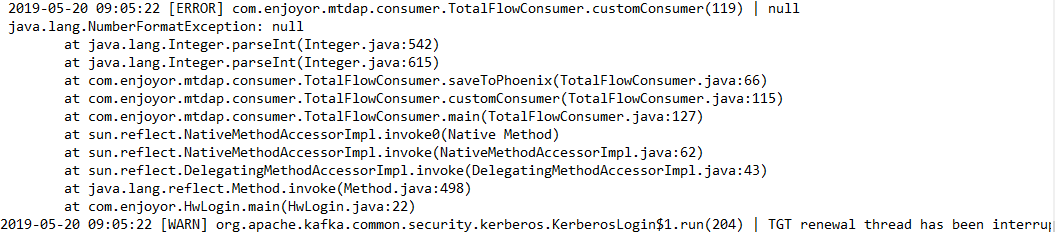
## 3. 30033 ms has passed since batch creation plus linger time

现象:多个实时计算程序运行一天后 直接挂掉



原因是:kafka磁盘满了,一共500G 实时计算一天写160G数据,塞满了

## 4.全流量消费者程序 发布报错



原因是全路段拥堵 的sink跟全点位流量sink到一个topic

导致流量的topic结果数据 字段不一样, 程序出错

重新发布后,发现问题仍然存在,

原因是消费组仍然接着消费之前offset的数据,从最新地方消费就好了

6.车辆轨迹计算 2小时后作业重启,不影响计算

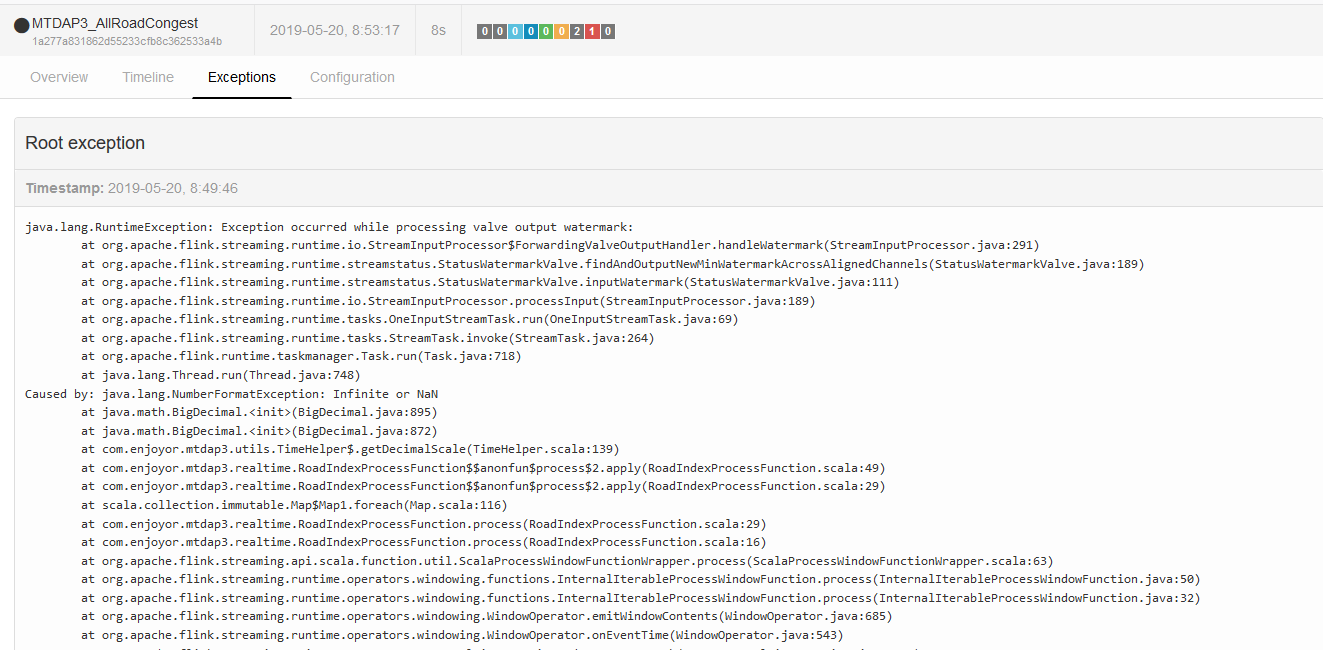
抛; Lost connection to task manager 'NCJJ06/192.169.101.6:32391'. This indicates that the remote task manager was lost.

原因:给定slot太少

增加slot的数量,之前是5个slot 现在给定了18个,已经平稳运行13小时,继续跟踪下

## 5.全路段拥堵 短时间不断重启

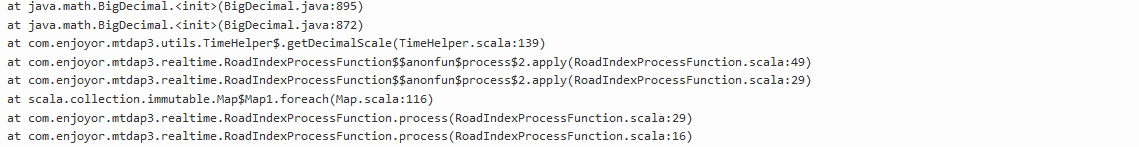
发布后程序一直在重启,依赖的key (23w条路段)



抛:处理输出水印时发生异常

java.lang.RuntimeException: Exception occurred while processing valve output watermark:

Caused by: java.lang.NumberFormatException: Infinite or NaN

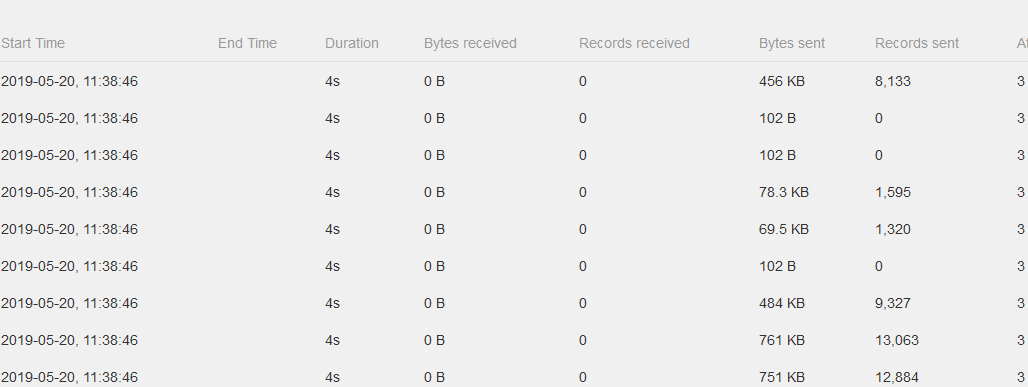
计算出的数据严重滞后

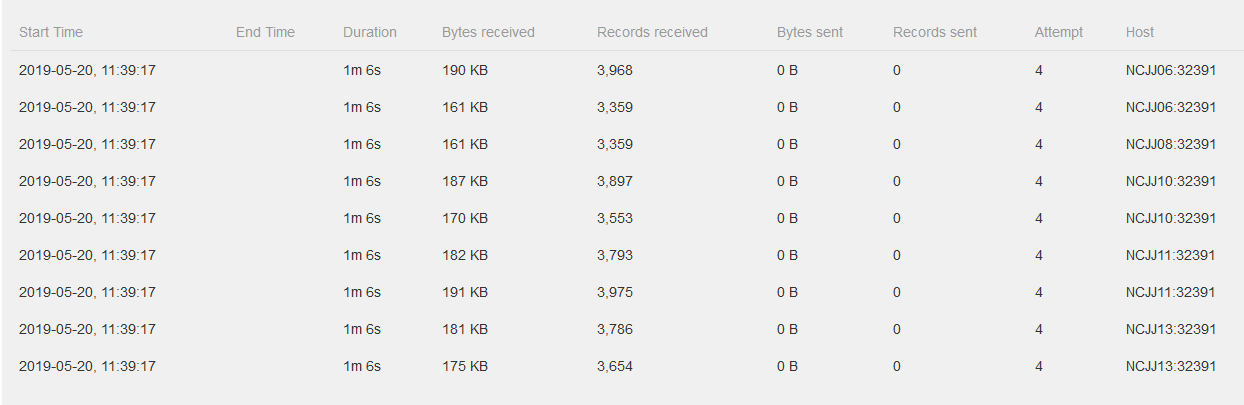
Infinite or NaN:无法格式化的数字，此数字要么不是个数字，要么是无穷大的数字，从而导致 BigDecimal 无法对其解析。

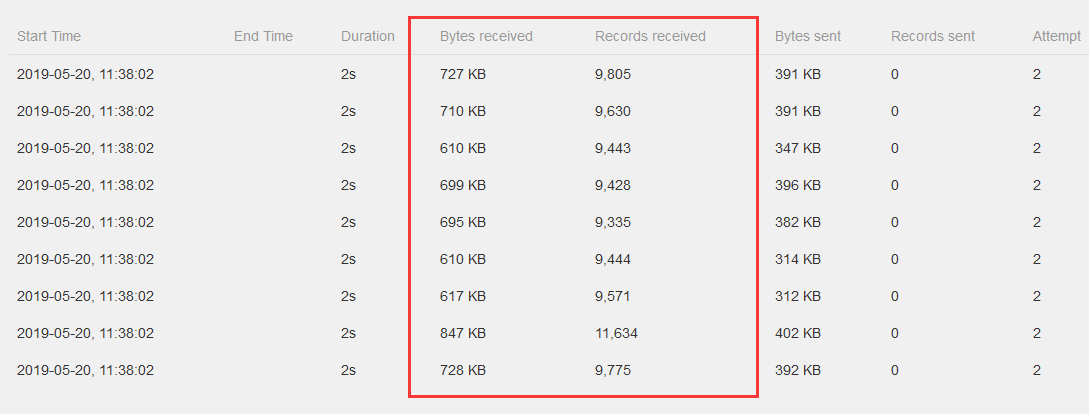
打印输出看下,发现频繁连接redis

**下面记录一次错误的排bug思路…**

路段23w 加点资源看看:





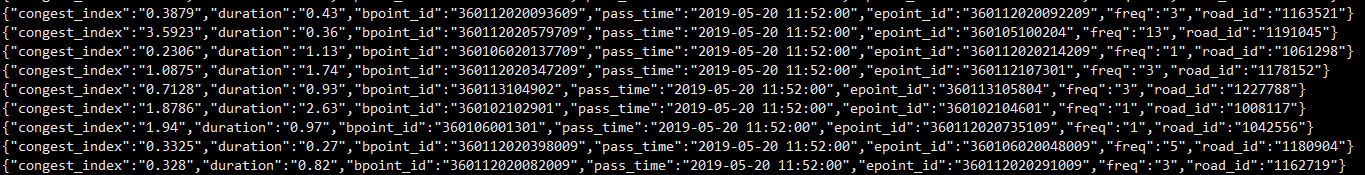


不是资源的问题,代码中出现异常的脏数据,导致程序出错 能多执行一会,但是还是重启

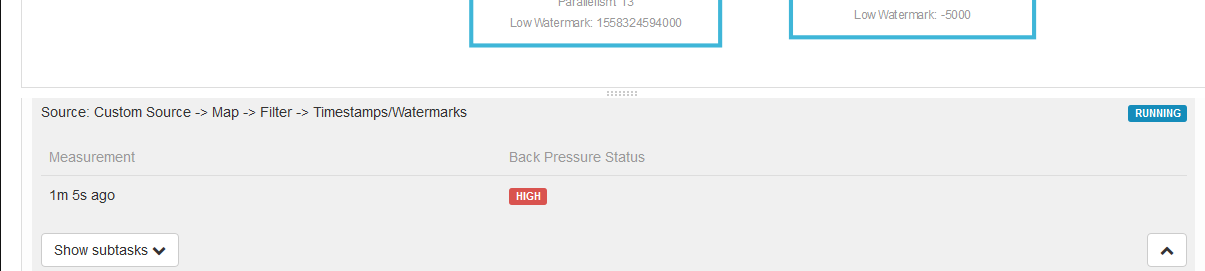
给定13个;

页面上可以出来了,但是程序一直重启???(跟踪下)

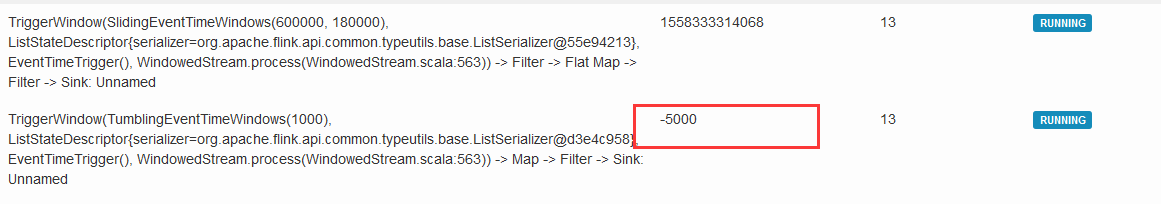
下午查看消费者,仍然在算1点多的数据:



查看监控,背压太高:



水印输出有错:

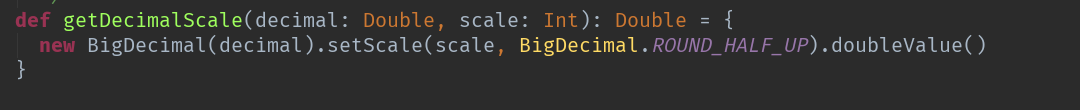


推测原因:

计算的路段太多,九江2w+ 南昌23w+,导致计算量太大

一直重启导致实时计算每次消费少量数据就挂掉,重启从checkpoint恢复后接着消费,出错挂掉重启 导致数据越积越多导致背压高

定位出错程序:



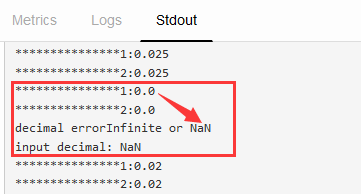
这段抛Infinite or NaN:

无法格式化的数字，此数字要么不是个数字，要么是无穷大的数字，从而导致 BigDecimal 无法对其解析。

跟CPU 没有关系:是代码中有问题

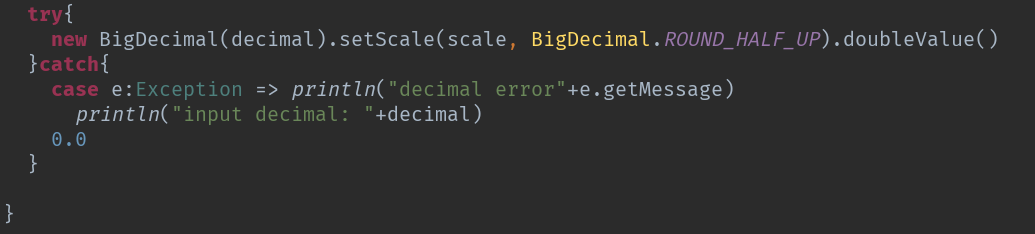
一个cpu内核分为3个slot 计算分配 50个内核 150个slot 每个程序10个

打印日志:



### 代码中更改:

捕获异常,NaN:not a number 0.0/0.0会出现NaN



### **问题原因**:

离线计算的全路段逻辑没处理好,出现路段bpoint为null, 之前程序没有这方面处理,直接调用工具类计算出现异常 挂掉,挂掉后触发Flink程序的未保证实时计算的稳定性的重启机制 不断重启(每次重启会连接redis,最终导致redis的最大连接数满了)

同时,挂掉后 导致下一次连接接着从kafka的topic消费 数据积压太多,产生反压现象,将解决问题的方向带偏,没有注意到导致程序出错到的核心问题

## 6: 连接redis抛: ERR max number of clients reached

JedisDataException: ERR max number of clients reached

原因是Flink程序报错 重启 每一次都得连接 redis 达到最大连接数

端口不会断掉吗?之前的连接没有释放吗 ?

### 解决:重启redis服务

实时计算不断去redis拿数据的任务由于访问不了redis,重启 一次性拿到内存的没事

重启后查看redis的最大连接数:10000

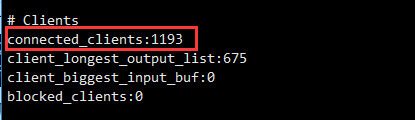


更改最大连接数,但是fusioninsight上不能直接修改



查看当前redis的连接数:

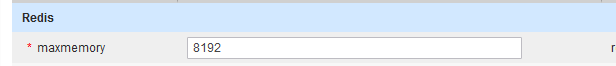
Info 🡪查看下面显示的:



**2;**出现 OOM command not allowed when used memory > 'maxmemory'

需要进入到fusioninsight 配置页面,将最大内存参数1024m调大,8192m

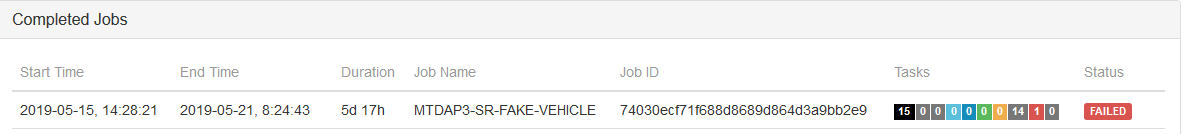
然后重启,实时计算程序会自动起来



**3**;几个月后出现的新问题:凌晨redis出现的达到最大连接数,早上来redis服务重启后 flink 程序 没有像之前那样恢复,而是一直重试抛空指针,需要将程序重新发布,才能解决(应该是状态恢复出现异常)

## 7: Connection reset by peer

Lost connection to task manager 'host-redis01/\*.\*.\*.\*:port'. This indicates that the remote task manager was lost.

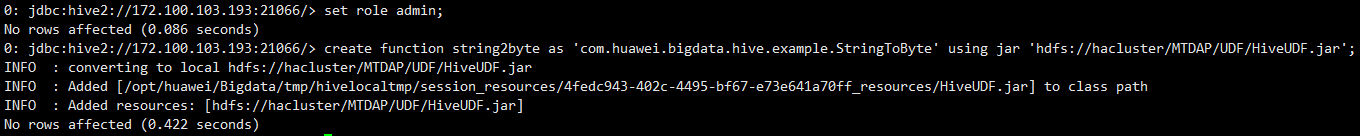


出现的场景1:全点位流量数据window计算周期过长时出现,新的检查点还没生成,旧的挂掉

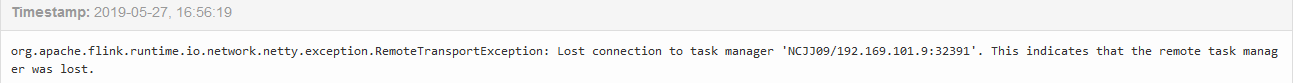
场景2:二次识别套牌车运行5天后挂掉

原因是代码中没有设置检查点自己就挂掉了,无法拉起(低级错误)

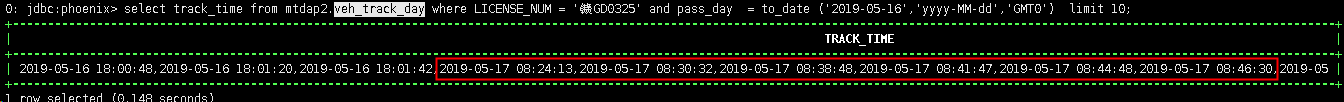
当时注释掉是因为1.3版本不兼容,1.4 版本试一下



全路段拥堵,给定30个slot 还是会重启()

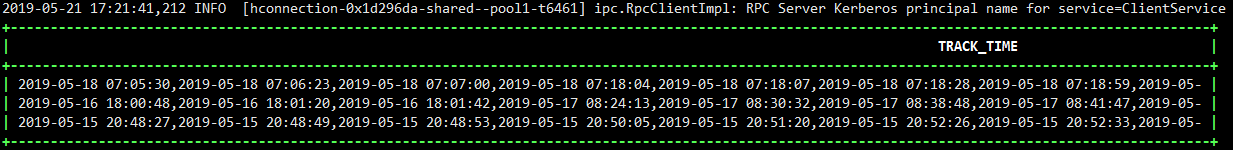


## 8:车辆轨迹 过车时间异常



每天一辆车一条轨迹,但是查询16号的轨迹,在过车时间出现17号

查看表:



16号跟17号的轨迹是连在一起的

推测原因:中间的结果缓存是存到redis的 17号 redis 中还是16号的 没过期,但是设置的是有过期时间的 ,但是可能没过期(之前也是出现过这种情况的)

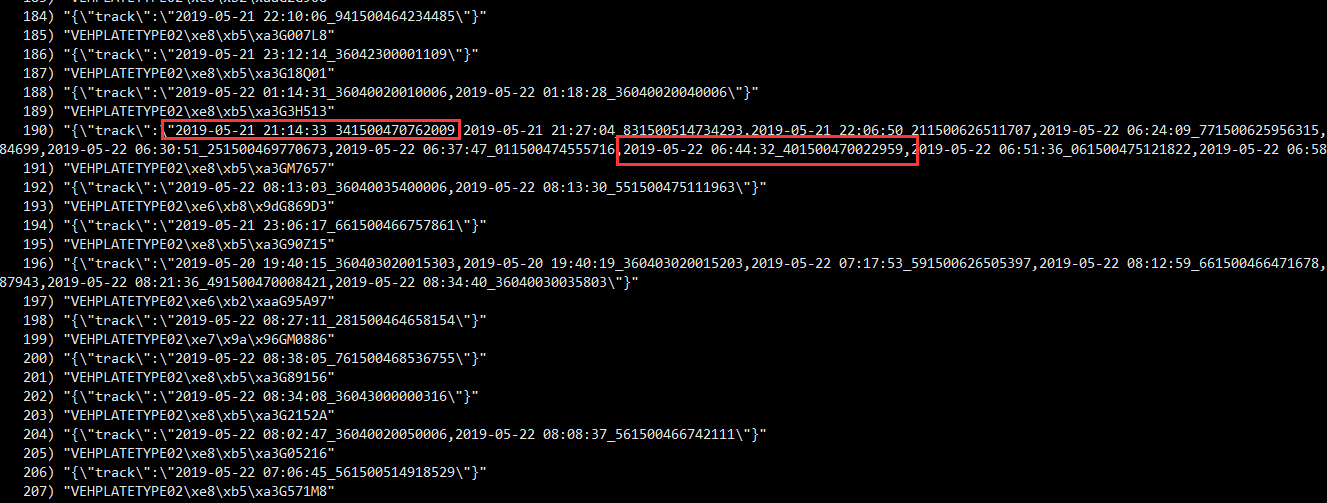
错误判断:

redis中作为临时存储的key: mtdap3\_vehicle\_track 设置的是午夜过期

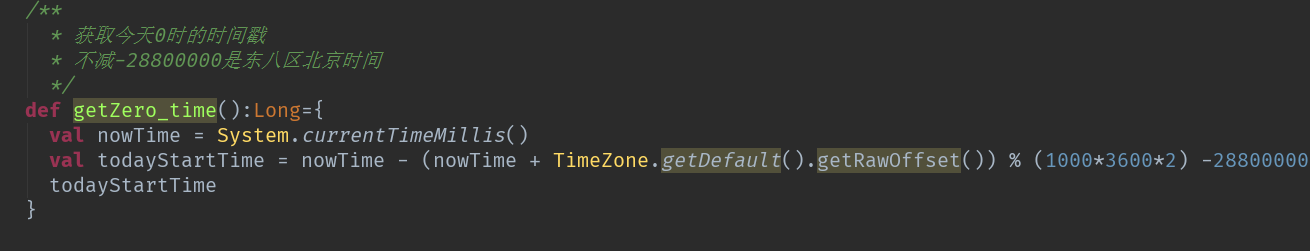
实时计算16号挂掉 redis中保存的还是16号的过车数据,17号实时计算执行 没有删掉过车轨迹的key 所以消费者跑完16号的跑17号,将过车轨迹追加到16号的后面 ,18号没有重复是因为key设置的是午夜自动过期,18号计算新产生一个过车轨迹key

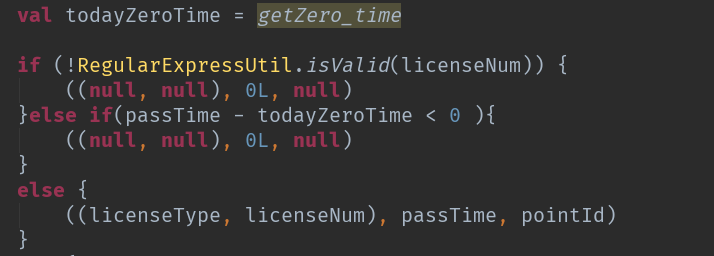
真正原因:**数据质量原因**

每天的key是正常过期的,但是传入的过车时间有可能是昨天的数据或前几天的数据,导致轨迹排序时,将这些加到过车轨迹的前面



解决:在代码中将不是今天的过车数据过滤掉

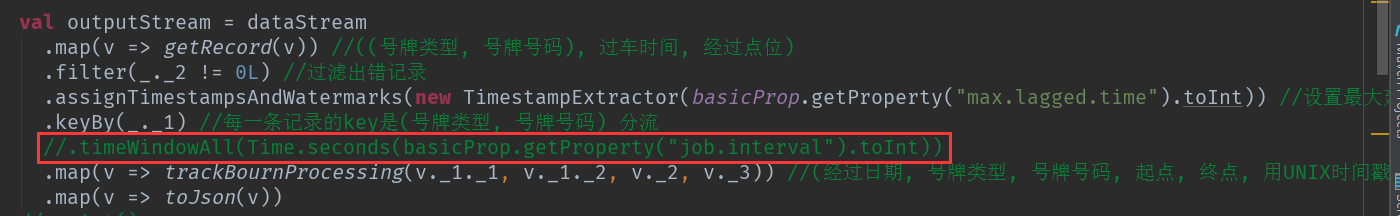




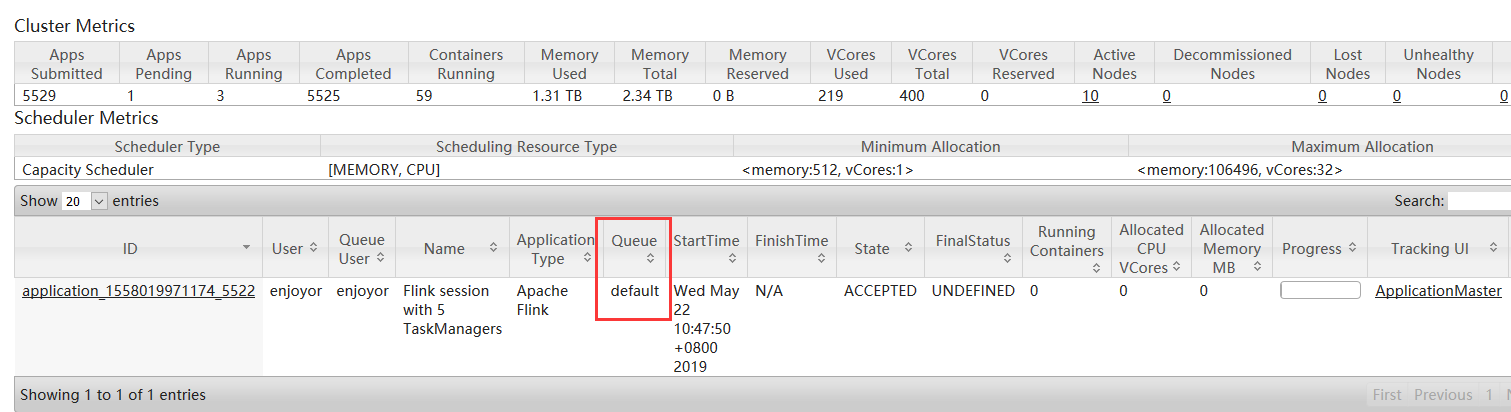
重新发布,数据正常

疑问: 当实时数据乱序到达时,水印应该会将事件时间延迟太久的这部分抛弃 不应该计算进去的???

水印没生效的原因:车辆轨迹没有使用窗口(原因:车辆轨迹开窗口24小时)

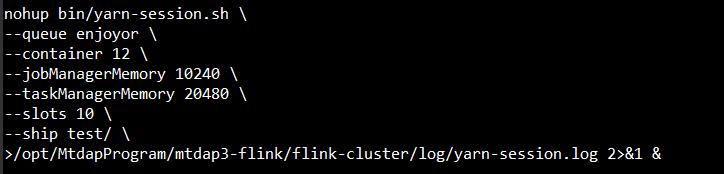


## 9:租户调整 yarn发布失败



原因:default租户的资源已经用完了 租户分成default和enjoyor 如果不进行修改 就会使用默认的default资源队列,从页面看 default已经用了一半了 所以发布的任务一直等待

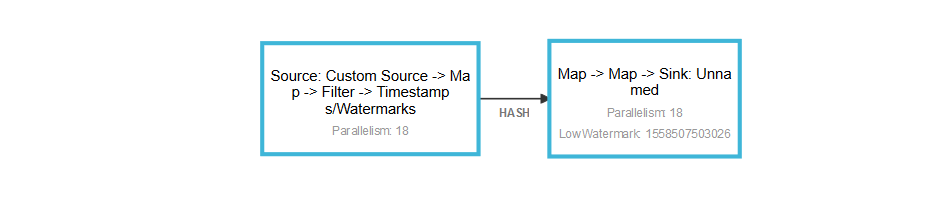
发布命令使用enjoyor租户



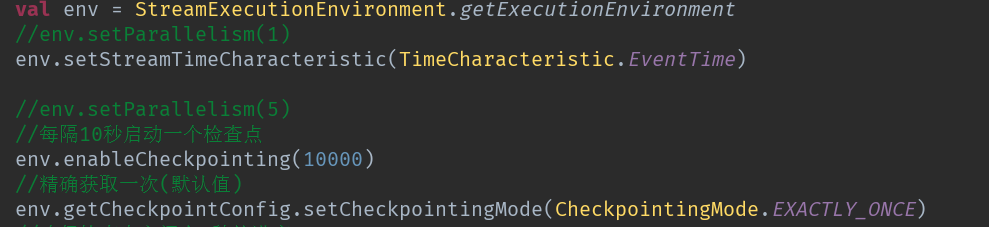
## 10.Flink车辆轨迹计算无窗口(未明白)

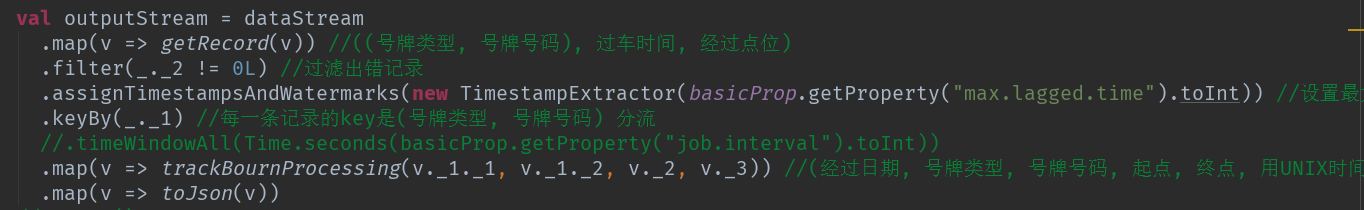
窗口是实时计算统计间隔的 程序中并没有使用窗口 底层是怎样算的 ?

Flink页面监控:并没有使用窗口



代码中





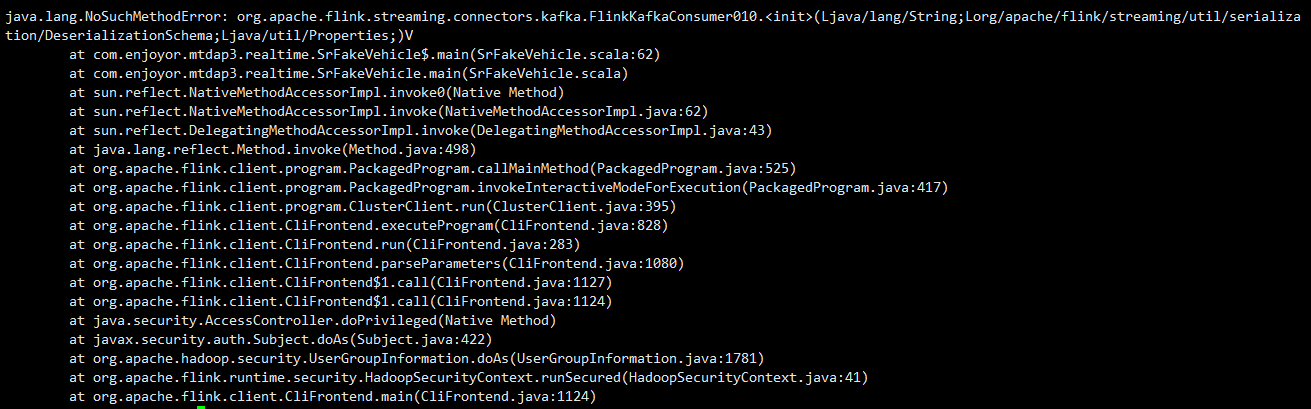
页面上先搞懂吧

## 11:双流Join跟checkpoint 1.3.0不兼容Bug

没有添加checkpoint是,可以正常运行,但是运行一段时间挂掉不能自己起来

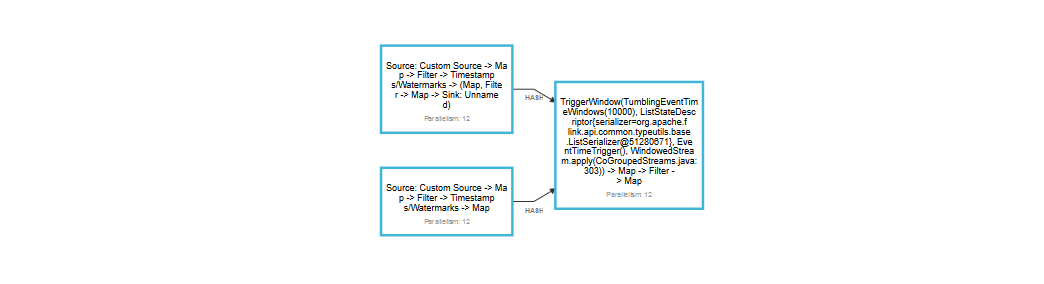


加上checkpoint时,发布会出现这个问题?



原因:pom文件没有替换成1.4的

重新发布:



跟踪一下,看是否会挂掉重启 稳定运行

## 12: This indicates that the remote task manager was lost.

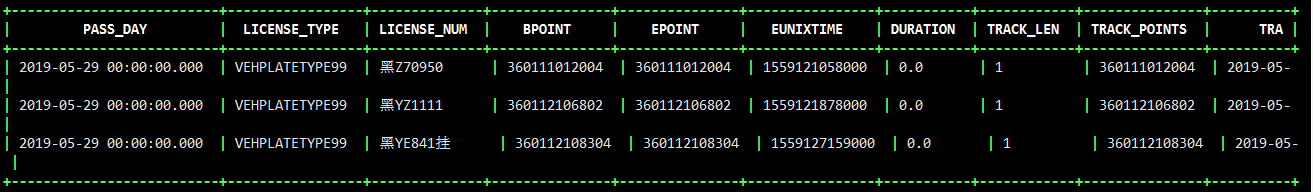
现象:NC全路段实时计算 会不断重启 且重启间隔会越来越短,最后3分钟就会重启一次

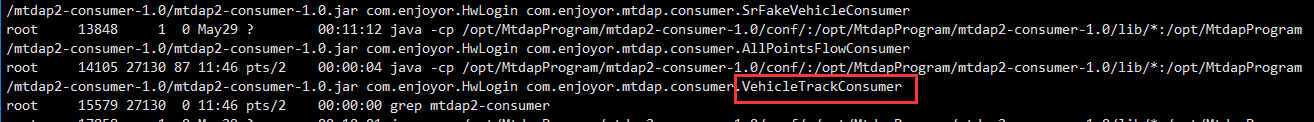


得益于Flink的检查点机制,重启接着计算 ,暂时先解决其他异常

## 13:车辆轨迹计算数据量太大 消费不过来

现象:车辆轨迹实时计算 数据量太大 消费者消费不及时(表里今天没数据)





日志打印下,是消费者的问题 还在消费昨天的数据 kafka消费数据堆积

消费滞后,每条的数据量太大,消费时候2秒poll一次导致阻塞,改成10s

原因:3分钟算一次

车辆轨迹一天往kafka发 160G 消费者入库吃不消,几分钟算好一批 将这些算好的入库更新掉

车辆轨迹数据太大 往kafka里发造成压力太大160G/天(现场kafka磁盘容量才1T)

kafka存号牌类型\_号牌号码,同步时候去redis根据号牌去取,

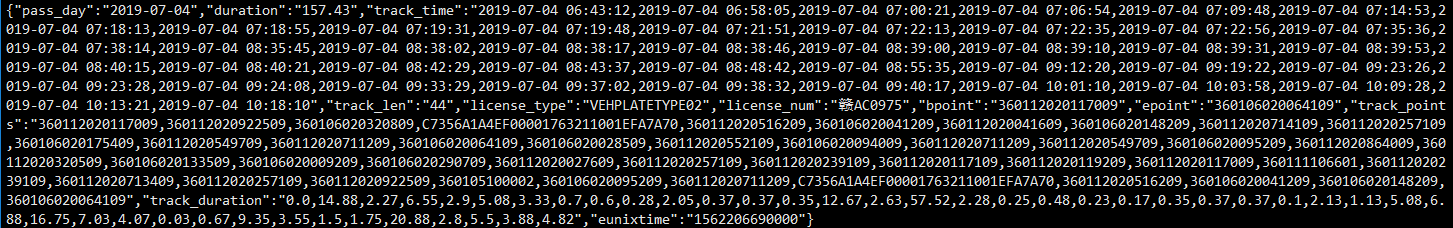
问题:数据量太大了,3000w次/每天

redis中数据结构:

1) "VEHPLATETYPE02\xe8\xb5\xa3A5A210"

2) "{\"track\":\"2019-06-05

06:33:32\_360112020800609,2019-06-05 07:37:05\_360112020800709\"}"

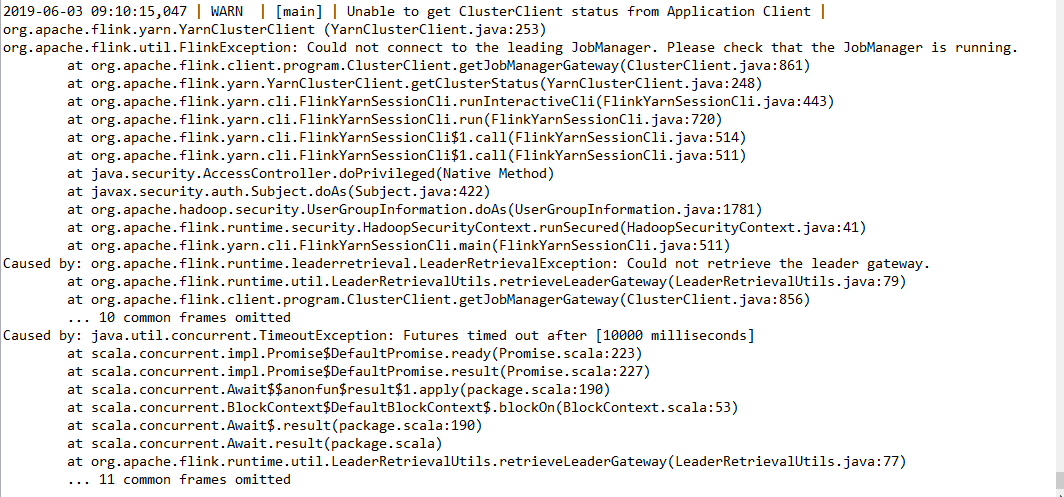


解决:kafka扩容 或者程序改成kafka存号牌类型+号牌号码,同步拿到kafka的数据从redis中取 在同步到phoenix(但性能无法保证).

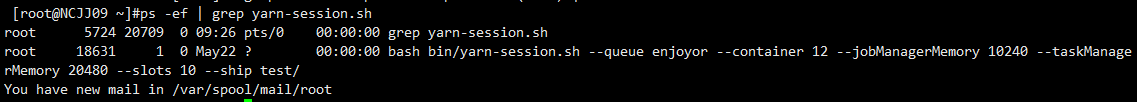
暂时解决:先发布多个程序,消费同一个消费组,代替多线程消费,增加消费能力

## 14:资源被争夺 导致程序挂掉

离线计算伴随车一发布,实时计算的程序就被干掉了



程序挂掉了 但是日志还是在一直的打印

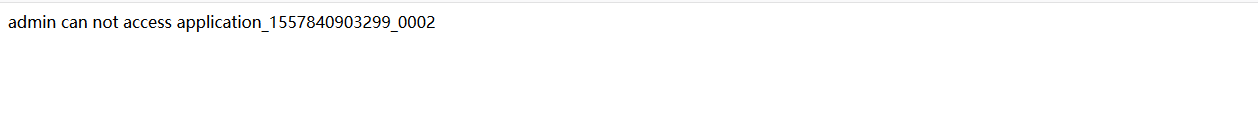
最终解决:由于伴随车算法和实现存在问题,NC市点位密集

每日计算需要大量CPU和内存,否则算不完

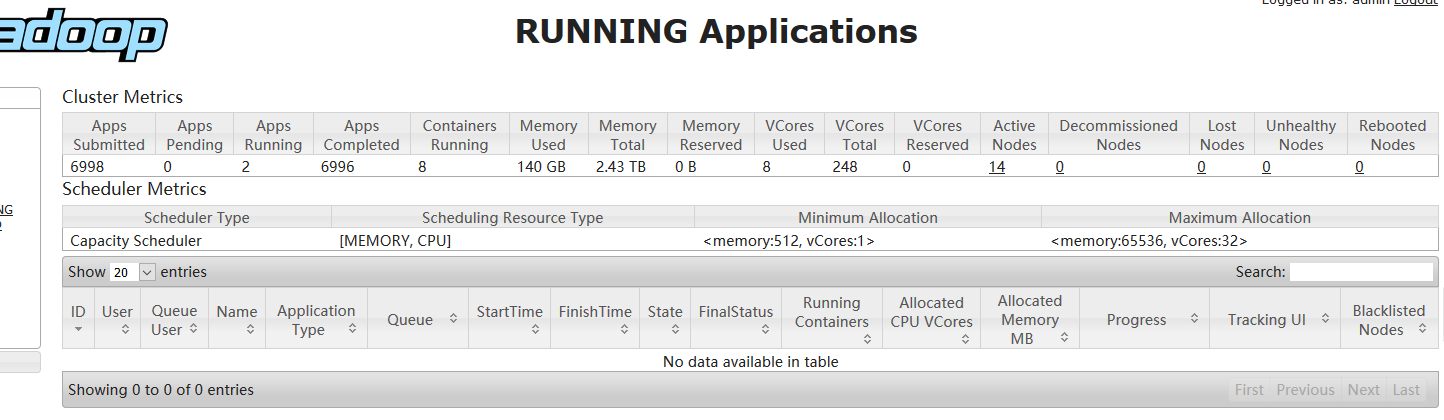
伴随车程序优化,暂时下线

## 15: YARN RM展示异常问题

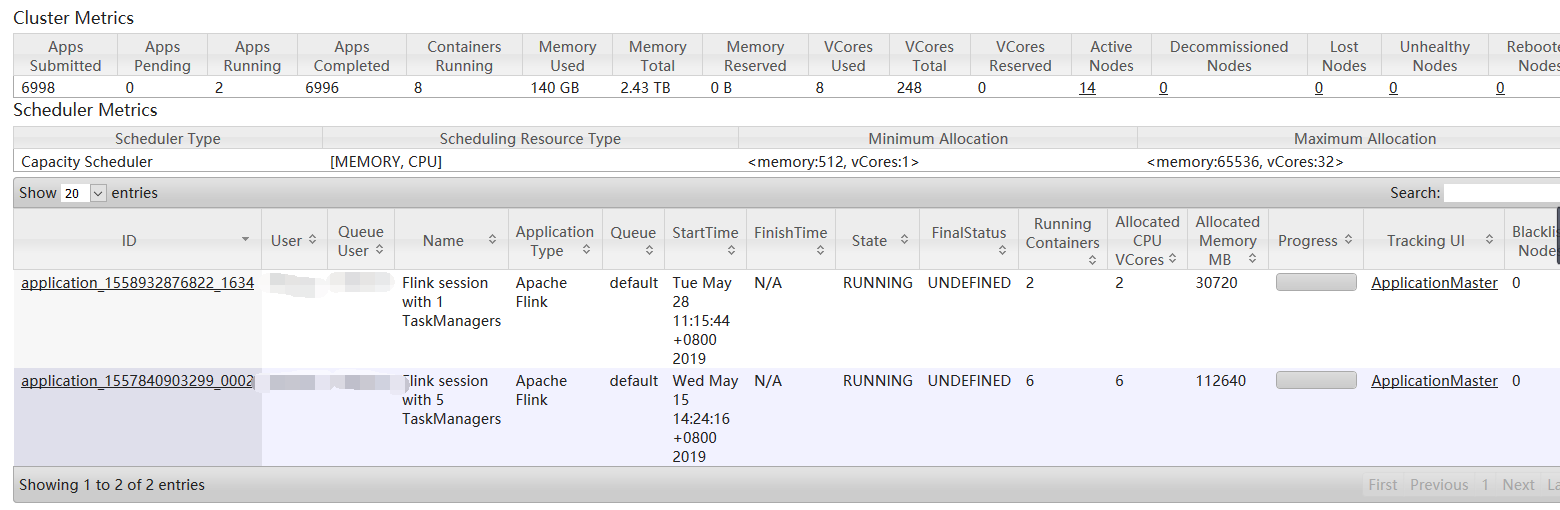
Yaen:ResourceManager



Yarn:

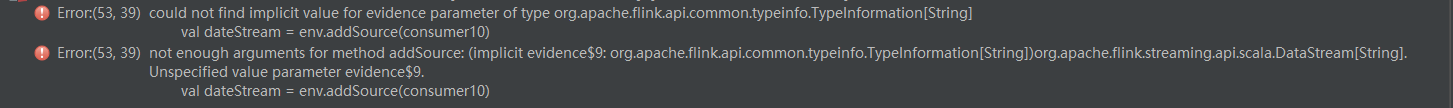


过一会再看 页面显示异常(yarn资源释放,告警恢复)



## 16:号牌类型维度进出城流量分析

### 1:程序打包bug



Error:(53, 39) could not find implicit value for evidence parameter of type org.apache.flink.api.common.typeinfo.TypeInformation[String]

val dateStream = env.addSource(consumer10)

最后定位到:



应该是

import org.apache.flink.streaming.util.serialization.SimpleStringSchema

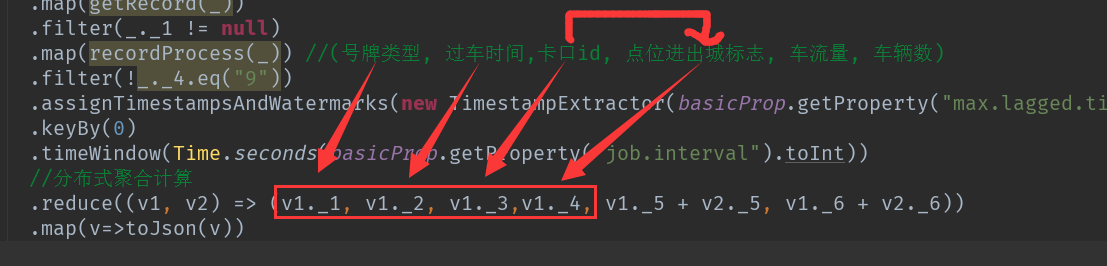
添加:

import org.apache.flink.streaming.api.scala.\_

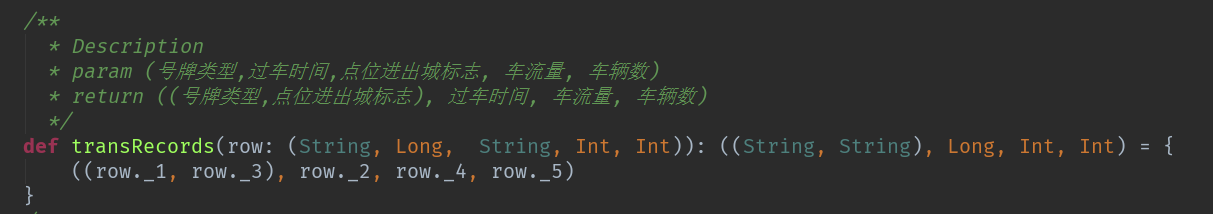
编译出错: Error:(66, 16) forward reference extends over definition of value outputStream .map(toJson(\_))

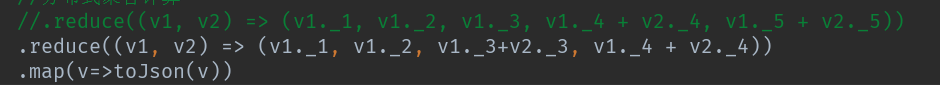
**添加lazy 懒加载** 问题解决

### 2:reduce聚合异常 && 时间异常



上面reduce时没有区分进出城,计算的是车辆类型进出城累计结果





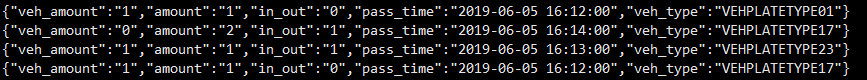
按照(车辆类型,进出城点位标识)分组 ,就行了

**新问题:**

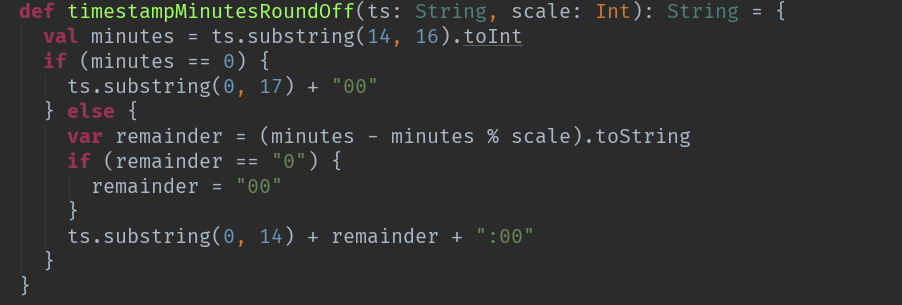
N分钟统计一次,过车时间秒部分按照传入的offset取整,有问题

过车时间是按照:统计窗口中出现的车辆类型维度下,第一条过车数据的过车时间,如果是15:59 到 16:01三分钟计算的 VEHPLATETYPE02出现的首次时间是16:00:23 最后统计结果的时间就是16:00:00,VEHPLATETYPE23出现的首次时间是15:59:12 最后统计结果的时间就是16:59:00.

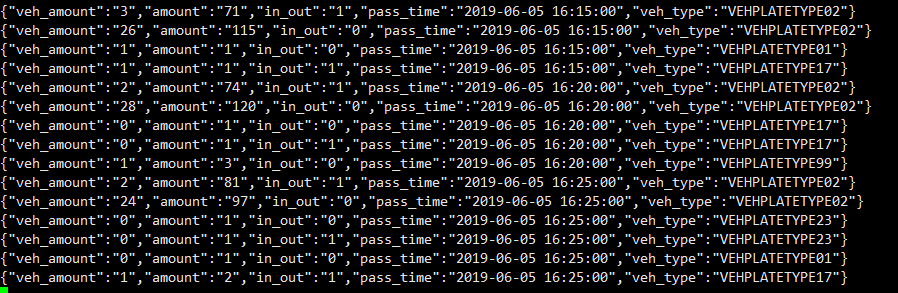
三分钟窗口计算后,统一输出就出现: 时间不一致(#**WARN** 可能会导致,趋势统计每小时,会将跨整点的数据分成两个小时区间累计展示,数据错误)



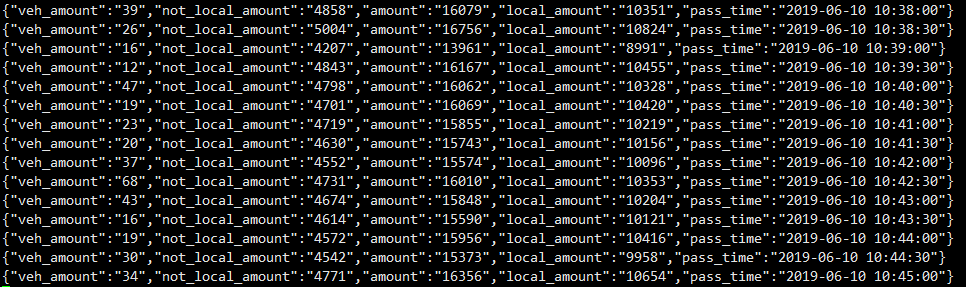
**解决方式:**时间窗口加大,改成**5分钟**算一次,同时按照分钟按照窗口时间取整

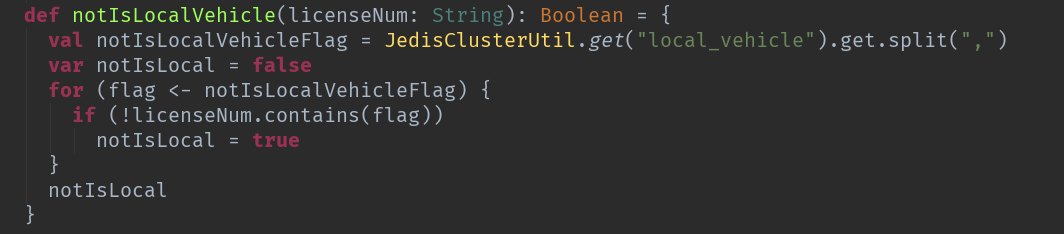


最终结果:

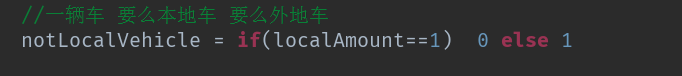


## 17:数据异常(总流量!=本地车流量+外地车流量)



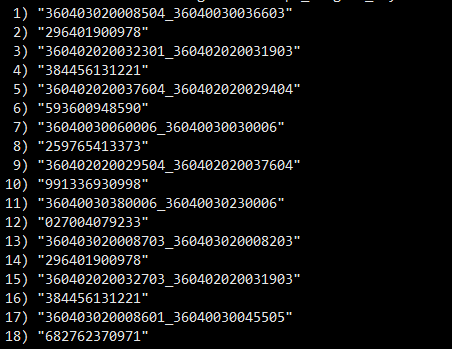


先改掉了,数据质量问题:数据不一致是卡口识别错误的 摩托车/电动车

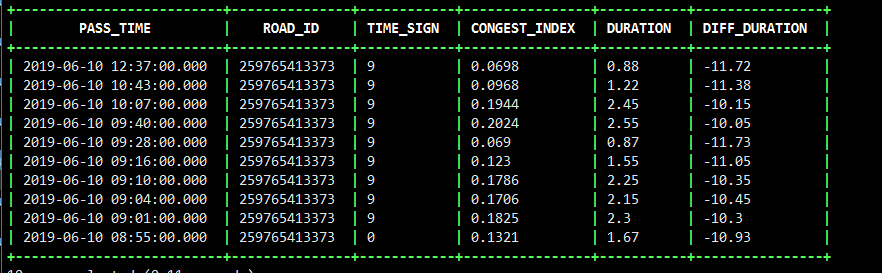


## 18:重点路段拥堵

大部分时间只有一条数据,偶尔2条



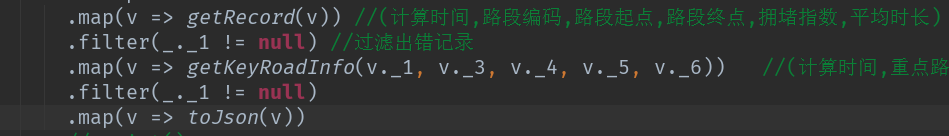
表里:



实时计算:没有使用计算窗口



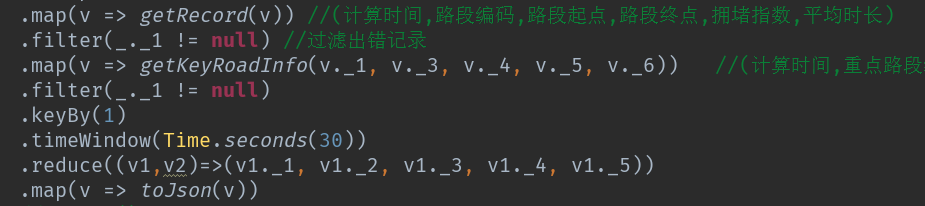
看了下南昌 也是一条,初步判断是程序问题,查看程序没有使用窗口



数据接入:全路段拥堵里拿的

1:资源不够 2:计算逻辑有问题, 没有加计算窗口 3:页面配置问题

先加资源 还是一条 🡪计算逻辑有问题?



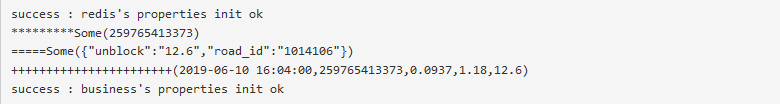
加计算窗口 加资源 还是一条

打印输出;

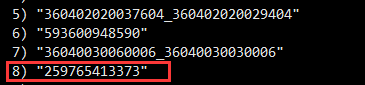




重点路段展一小部分,只打印重点路段的



问题出在这个key(mtdap3\_congest\_keyroad)上 ,Key是同步配置中配的重点路段



使用process算子 实现

可以参考issue50

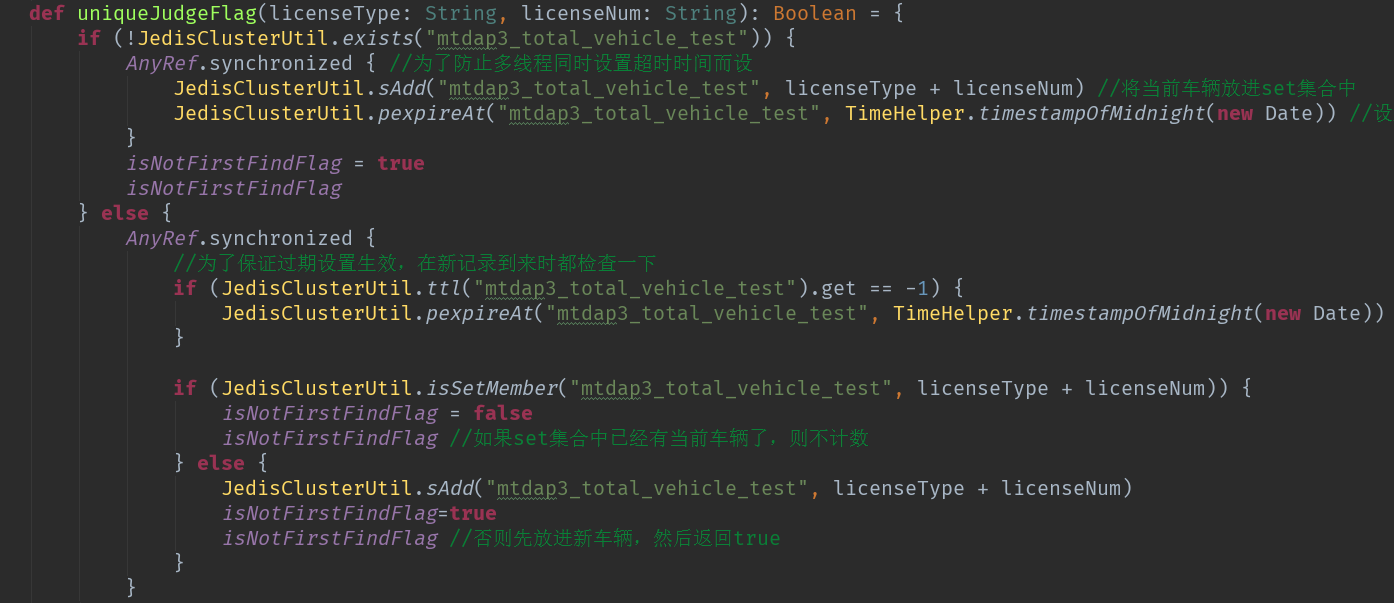
## 19:外地车分析

数据异常

{"veh\_amount":"0","amount":"85","pass\_time":"2019-06-11 13:36:30","not\_local\_veh\_amount":"0","out\_longtime\_veh\_amount":"0","not\_local\_amount":"24","province\_out\_veh\_amount":"0","out\_province\_veh\_amount":"0","out\_longtime\_amount":"0","local\_amount":"61","local\_veh\_amount":"0","out\_province\_amount":"8","province\_out\_amount":"16"}

1:外地车 应该= 省外车量和本省外市车辆 (有异常数据的)

2:外地车 车辆数都是0 复用同一个车辆去重函数导致,只有第一个有车辆数,然后添加上redis的key 导致后面判断该辆车都是出现过的 返回0 将返回true/false改为变量,后面的用第一次判断的变量做该辆车的标识



异常2:

{"veh\_amount":"10","amount":"89","pass\_time":"2019-06-11 15:36:00","not\_local\_veh\_amount":"4","out\_longtime\_veh\_amount":"0","not\_local\_amount":"20","province\_out\_veh\_amount":"7","out\_province\_veh\_amount":"3","out\_longtime\_amount":"0","local\_amount":"69","local\_veh\_amount":"6","out\_province\_amount":"8","province\_out\_amount":"81"}

{"veh\_amount":"1087","amount":"15270","pass\_time":"2019-06-11 15:39:00","not\_local\_veh\_amount":"658","out\_longtime\_veh\_amount":"0","not\_local\_amount":"5346","province\_out\_veh\_amount":"626","out\_province\_veh\_amount":"461","out\_longtime\_amount":"0","local\_amount":"9924","local\_veh\_amount":"429","out\_province\_amount":"1496","province\_out\_amount":"12834"}

1;数据少的多 30s 流量就89条 是kafka的问题? 有时会达到

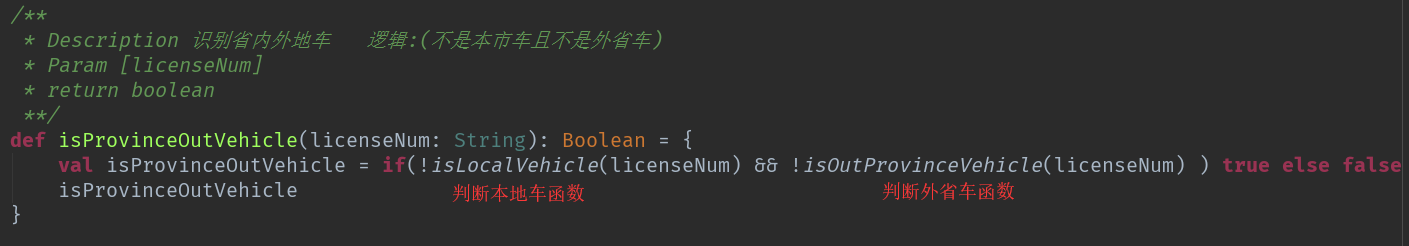
2:省内外市车跟本地车流量有问题

本省外地车+本地车 >总流量 判断逻辑改成非本地车-省外车辆

问题:

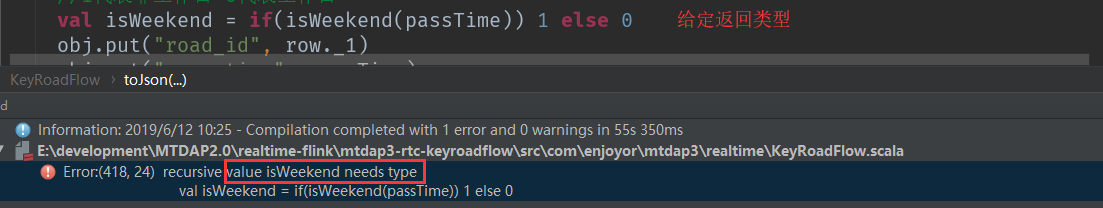


修改成:将循环赋值改成调用工具类 数据正常

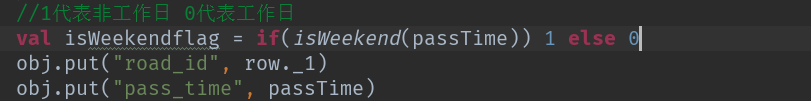


{"veh\_amount":"1054","amount":"16637","pass\_time":"2019-06-11 16:51:00","not\_local\_veh\_amount":"674","out\_longtime\_veh\_amount":"0","not\_local\_amount":"5914","province\_out\_veh\_amount":"184","out\_province\_veh\_amount":"490","out\_longtime\_amount":"0","local\_amount":"10723","local\_veh\_amount":"380","out\_province\_amount":"1585","province\_out\_amount":"3199"}

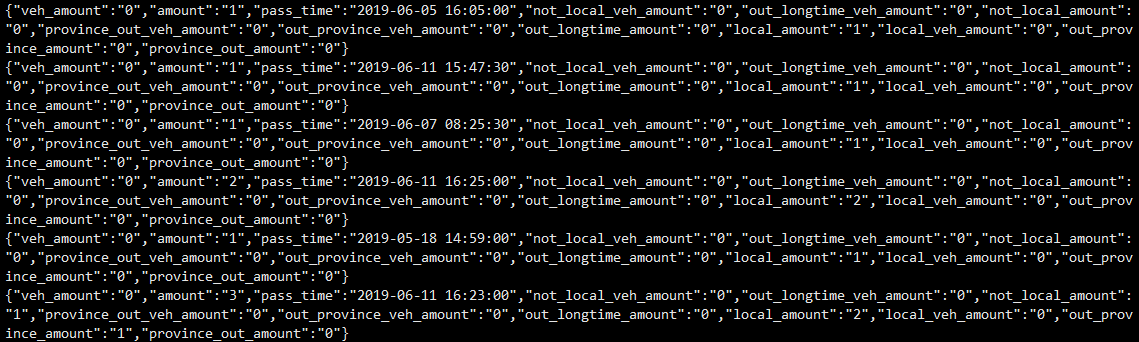
编译错误



修改为 变量名跟函输名重名导致的!



## 20:程序kill后,会产生一些脏数据

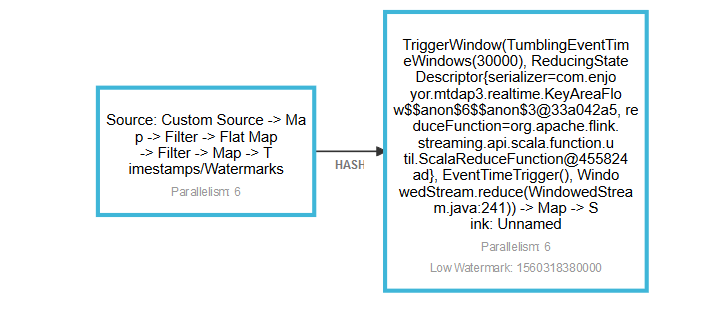


什么原因?

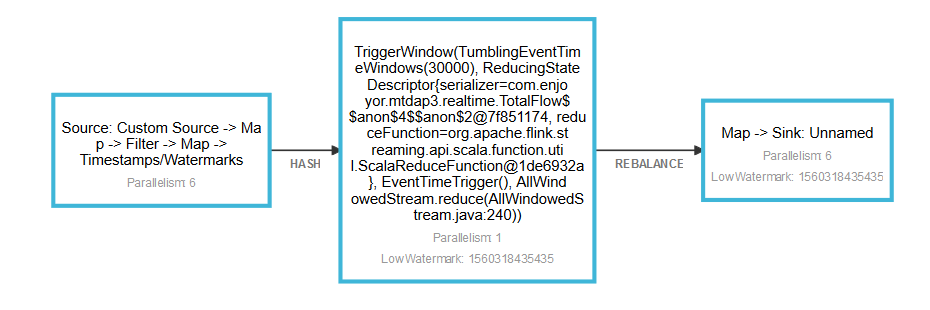
推测是程序关闭后,Flink在keyBy -> Reduce的数据 直接输出kafka了

## 21:算子并行度 问题(全流量外地车)

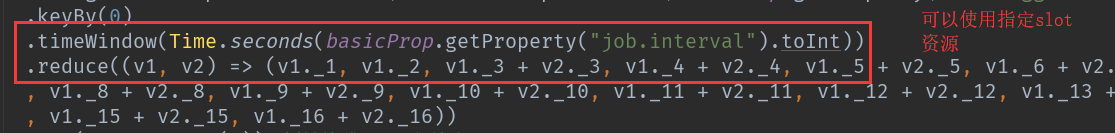
新发布的测试程序的reduce 的并行度竟然是 6 !

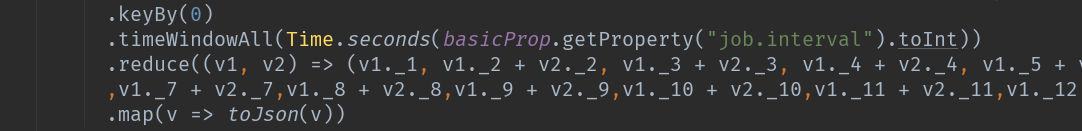


全流量计算的中间计算slot是1 ?



代码中的区别:





原因:

dataStream :

* 对于非KeyedStream，有timeWindowAll、countWindowAll、windowAll操作，其中最主要的是windowAll操作，它的parallelism为1，它需要一个WindowAssigner参数，返回的是AllWindowedStream
* TimeWindowAll 不是并行的 所以slot只能用1 个,滚动窗口

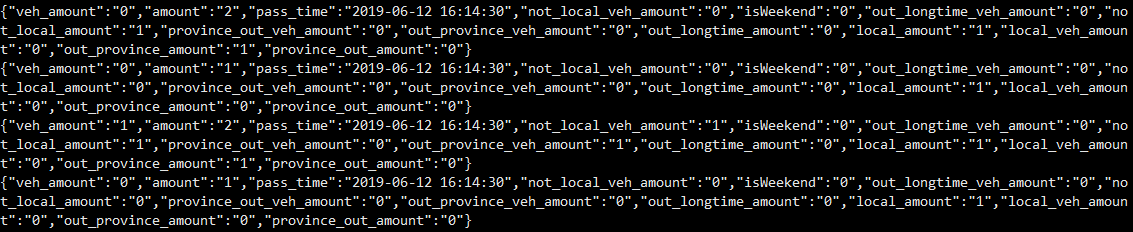
KeyedStream:

* 对于KeyedStream除了继承了DataStream的window相关操作，它主要用的是timeWindow、countWindow、window操作，其中最主要的是window操作，它也需要一个WindowAssigner参数，返回的是WindowedStream

需要看一下DataStream 源码

将 TimeWindowAll 改成timeWindow,统计的是每个卡口的数据,就要reduce

将代码改成keyBy(0) + reduce 输出的是一条一条的没有汇总

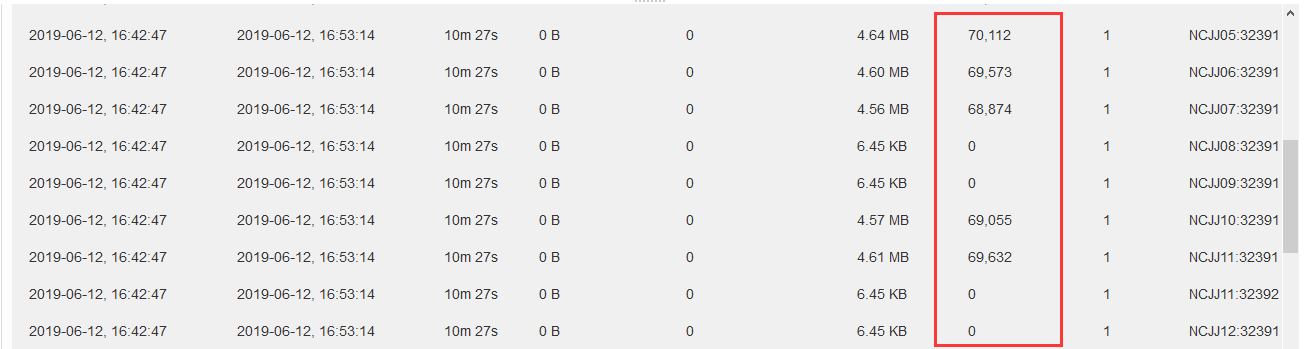


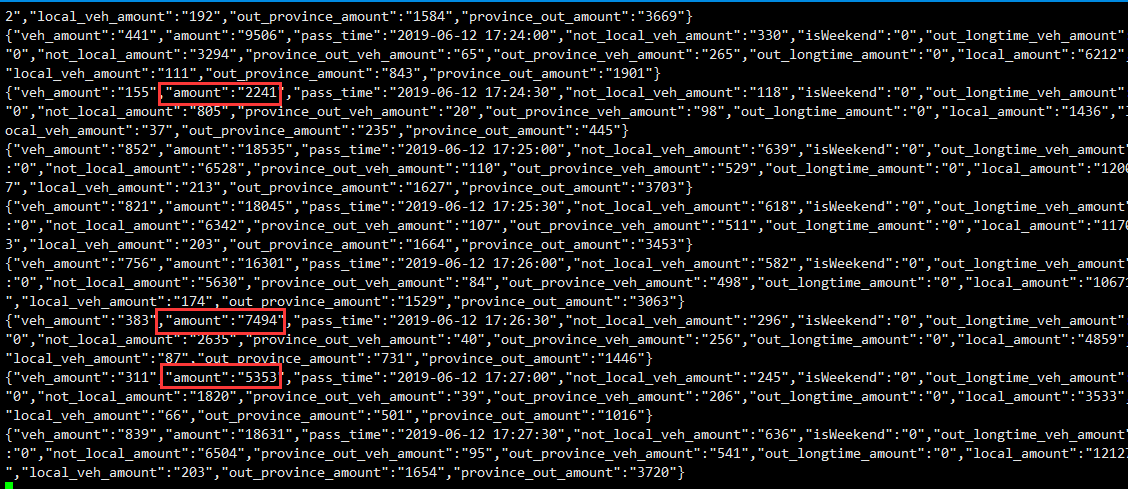
原因,keyBy(过车时间) 过车时间不一样,不一样的过车时间都出来了

参考区域跟路段的实现,keyBy(0) 就决定了 输出的条数,reduce只是将各个keyBy进行汇总

之前使用TimeWindowAll实现,是为了减少复杂度,但是没有增加资源,所以我在这里改掉.

但是是做数据汇总的! 还要使用TimeWindowAll,在slot上注意点

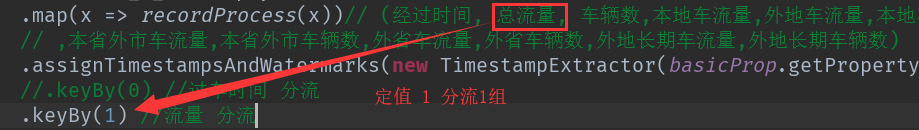
上面这种资源浪费,性能反而没有最优6个的好



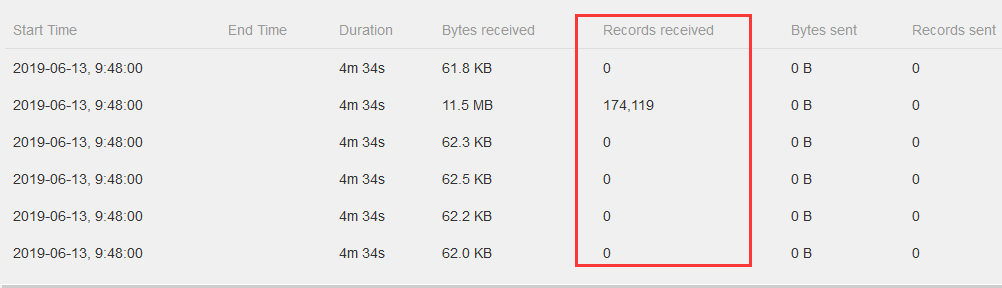
跟踪发现在流量低的时候是稳定的,流量高就算不过来了

第3次测试:既然30s统计有问题,将时间窗口拉大改成1分钟计算一次,问题仍在存在

测试4:将使用TimeWindowAll的逻辑改掉,使用keyby+分流窗口timeWindow,在调试资源+更改keyBy的参数



页面监控:

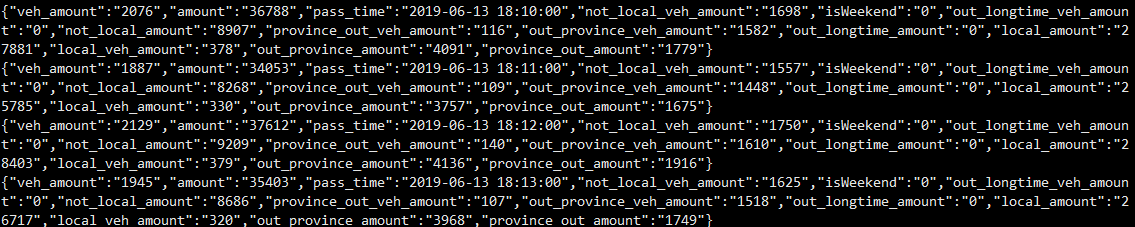


使用车辆数分流试试: 不行

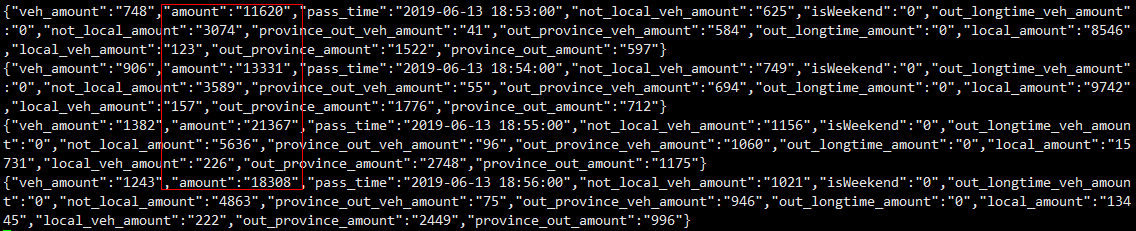
先将这些过车时间都转成按照30s 取整的过车时间,然后进行keyBy会怎么样 ,数据量还是那么多 分流还是1 , reduce计算的话 还是算不完 ,推测跟keyBy(流量结果一样)

测试: 不行

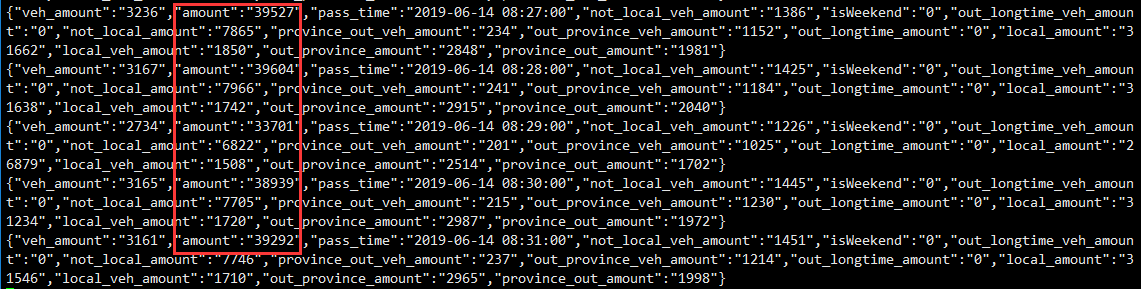
加一个水印时长试一下,一分钟计算一次,延迟30s (数据跟正常差不多了)



But过一段时间后,又出问题!!只是将数据延迟,过一段时间还是会出现这个问题



运行一晚上后:正常(1分钟计算一次,水印时间60s,数据产生->展现总时延2min)



## 22:QA

Q:Flink统计周期数据算不完,是怎么处理的 ?

Q:Flink是什么时候触发计算的?A:最后一个窗口结束的时间戳>窗口时间+延迟时间时,开始计算

Q:Flink自定义触发器

## 23:打包编译错误

forward reference extends over definition of variable 函数名

在前面加lazy

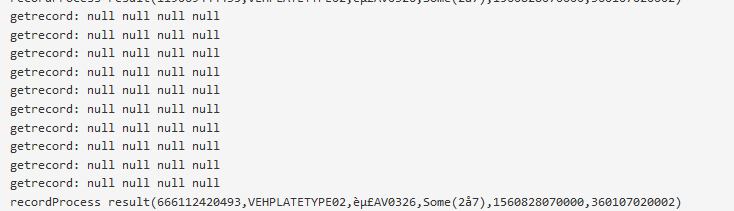
## 24:尾号限行

实时对数据进行清洗,从过车中按照规则获取到违反尾号限行的数据,存到phoenix,数据供离线计算使用

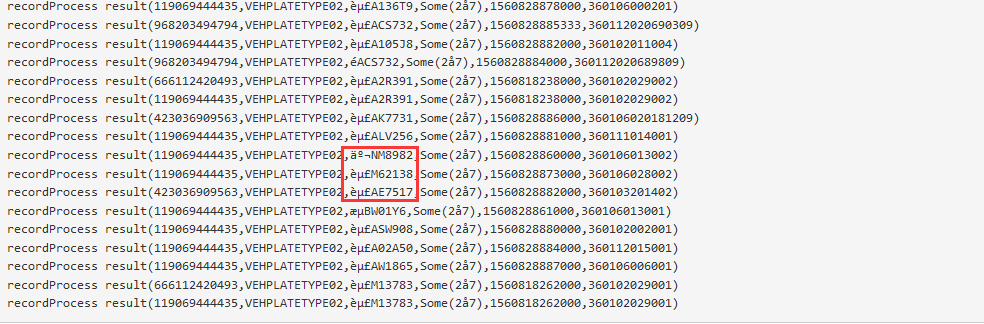
发布异常 一直重启 查看源码出现主体类调用类名调用跟自己同名的工具类 导致的

数据发布一晚上没有抓到违反尾号限行车辆,程序正常 跟踪发现问题出在isTailNumberLimitTime类上

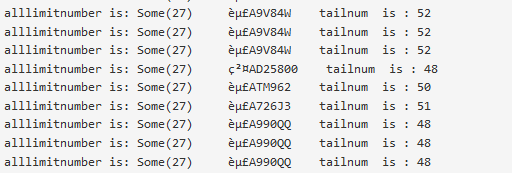
2019-06-18 00:30-23:30:00 中00:30-23:30 在数据中没拆分 导致异常捕获 输出null



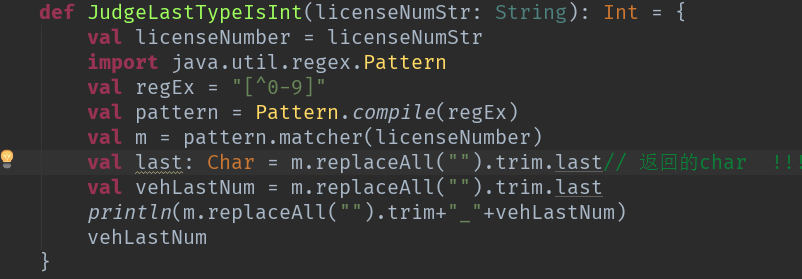
过滤后打印:



继续打印;



方法类有问题!



返回的是char!!!!

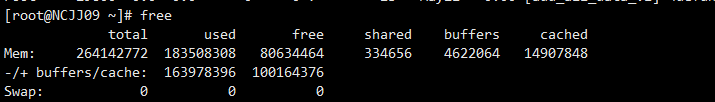
## 25:客户机9内存占用明显

平台查看

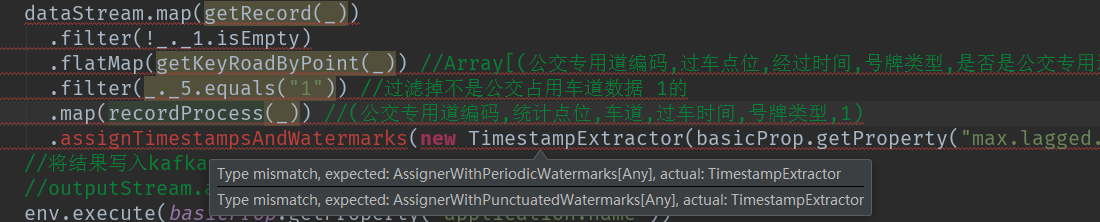


ps aux --sort –rss 查看耗内存的程序

消费者程序X2 跑了20天 占用服务器30%内存 关掉重新发布:



## 26:使用水印异常



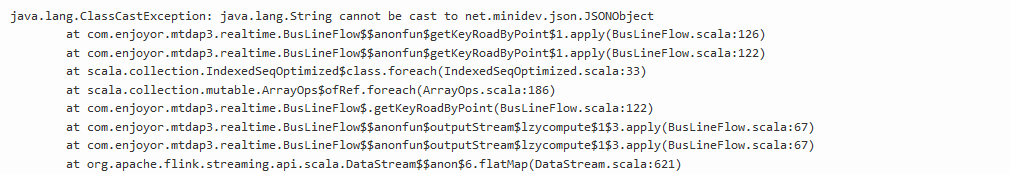
问题是在:



应该是:



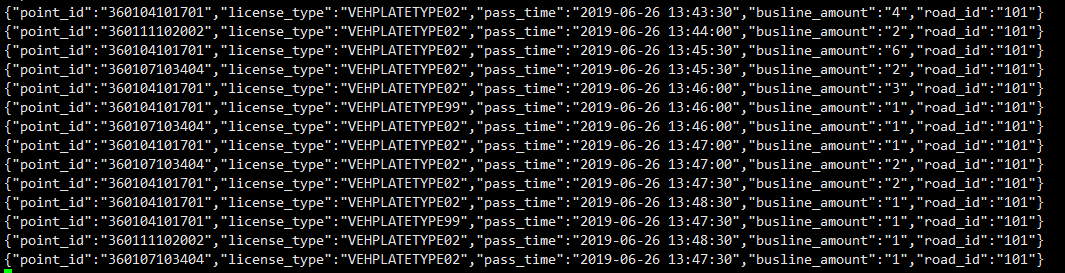
Bug2: java.lang.ClassCastException: java.lang.String cannot be cast to net.minidev.json.JSONObject



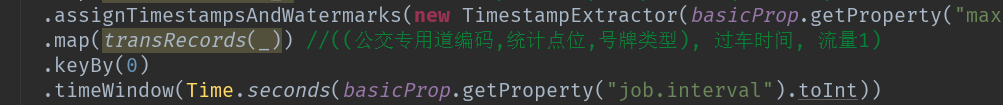
从redis中读取的数据 不是json 而是string 导致转换错误

A:使用程序 同步到redis,以json格式存储

公交车专用道流量计算:



数据延迟,过车时间顺序紊乱(可能是数据清洗放在水印后面导致的)

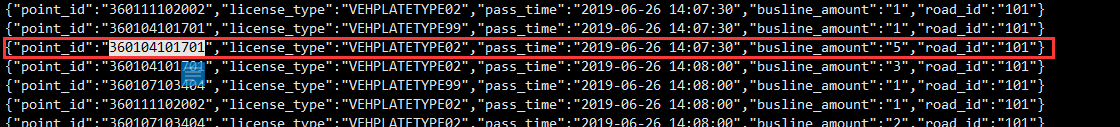


将数据清洗放在水印之前后,时间乱序解决

验证:拿到一条数据跟过车数据做对比,

passing\_time:[2019-06-26T14:07:00Z TO 2019-06-26T14:10:59Z] point\_id:360104101701 中

lane\_numbe:3做对比 数据是对上的,都是5



## 27:常用函数

ProcessWindowFunction :一次性迭代整个窗口中的所有元素,比较重要的是里面的context对象,可以获取到事件和状态信息 ,缺点:不是增量计算,需要缓存整个窗口的数据再去处理,需要设计好内存,有点浪费内存

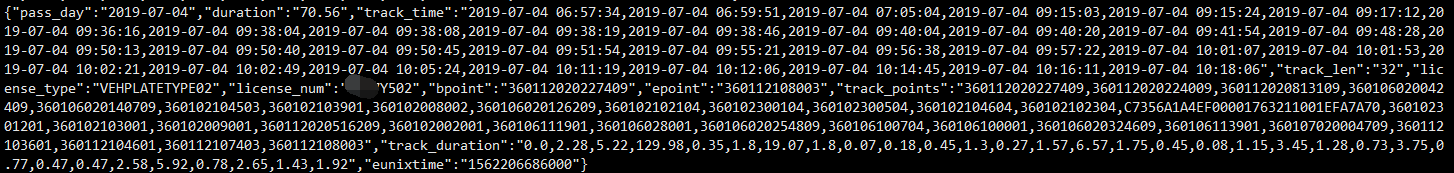
但是: ProcessWindowFunction可以结合ReduceFunction, AggregateFunction, FoldFunction来做增量计算.

使用:

自定义函数继承ProcessWindowFunction 重写process方法

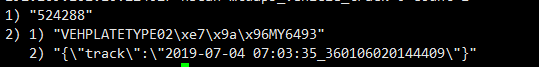
## 28:车辆轨迹计算优化(issue:13)

现状:查看issue:13



思路:

整体逻辑不变 ,往kafka发送号牌类型+号牌号码 ,在消费的时候根据拿到的号牌类型号牌号码去redis中获取对应的轨迹



这样改:如果redis挂掉了 ,车辆轨迹就没了

如果是在kafka上,挂掉之后数据还在

或者是直接入库,对hbase进行更新 但是会对phoenix的压力较大 不知道会不会像spark那样发生写不及时,导致背压高 影响计算

## 29:突发拥堵

一开始是一条一条的展示 ,预警时间是结束时间 体验不好

改成持续拥堵时,只保留一条

修改:

突发拥堵 改表 主键(开始时间\_路段id) ok

程序中 最后的结果 转成开始时间 (窗口结束时间-持续拥堵时间 ==> 拥堵开始时间)

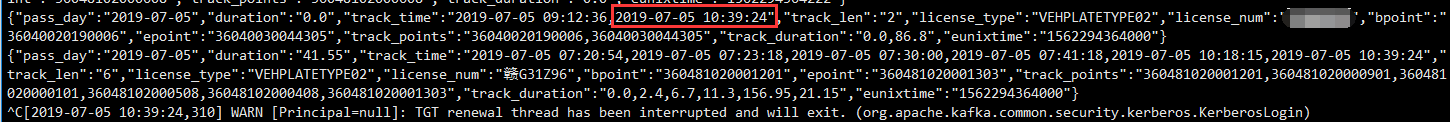


## 30:车辆轨迹 bug

套牌车的两辆车 找不到过车图片 原因是有的过车时间对不上

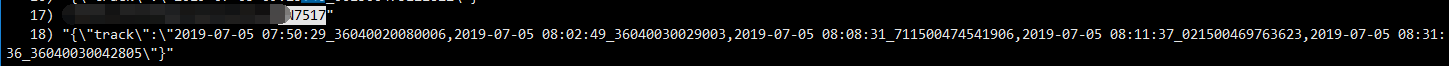
由于套牌车是根据轨迹时间计算的,所以先看一下轨迹计算结果,果然





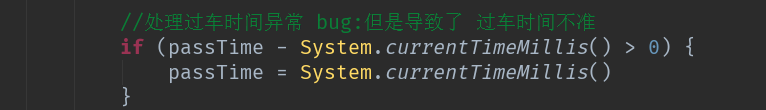
查看轨迹计算中间结果redis 缓存 跟solr对比: 5条2条对不上





数据源不同?

是不是一次识别跟二次识别导致的,不会,九江没有二次识别 用的同一个数据源



修改:

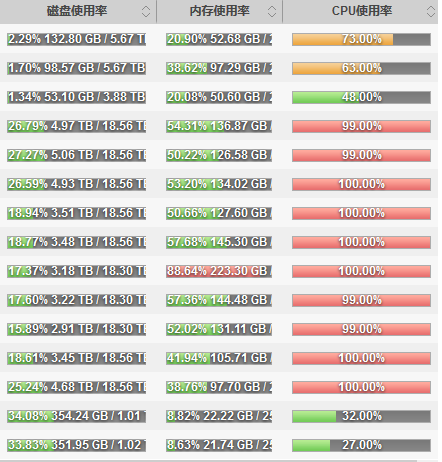
加一个字段记录最新时间 ,缓存时再替换掉

涉及范围:其他的不影响

## 31:流量实时计算出现数据延迟

早上发现数据延迟一小时

查看集群状态:资源不足 CPU等待导致

离线计算MR跑起来,CPU 直接满,导致我实时程序出问题

重点区域 ,新增加一万多个区域,算不完延迟

全流量计算 今天才算到昨天的时间 页面上没有数据

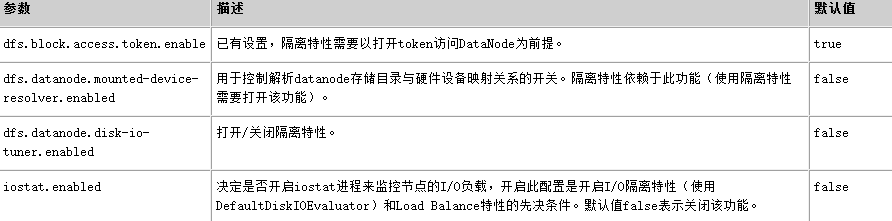
Redis连接数满 导致 全路段拥堵出问题

问题:配置一环二环给资源占掉 导致 这几个出问题

原因是资源不足,离线计算 跟实时计算争夺CPU

租户划分后 仍然是出现CPU占用100%的问题

应该是没有配置IO 隔离;



解决是计算延迟问题:

实时计算需要计算的指标较多并且窗口聚合使用的是timeWindowAll()窗口,并行度为1,所以不能通过加并行度解决. 由于1分钟内计算不完会导致上游流控,kafka接收数据控制速度,这样不会造成数据挂掉,但是计算结果一直在延迟.

之前的做法是增加水印的时间60s 问题可以解决,但是多增加一个指标后,满足不了,水印时间改成120s,监控下:晚高峰最高延迟45min 一次计算6个

## 32:指定程序发布在哪一个flink集群上

不指定job发布,会默认发布到最近的集群

使用 –yid 参数

bin/flink run \

-yid application\_1562200407137\_6605 \

--class com.enjoyor.mtdap3.realtime.BurstCongest \

/opt/MtdapProgram/mtdap3-flink/mtdap3-rtc-burstcongest-1.0/mtdap3-rtc-burstcongest-1.0.jar

## 33:flink on yarn模式发布 flink-cluster失败

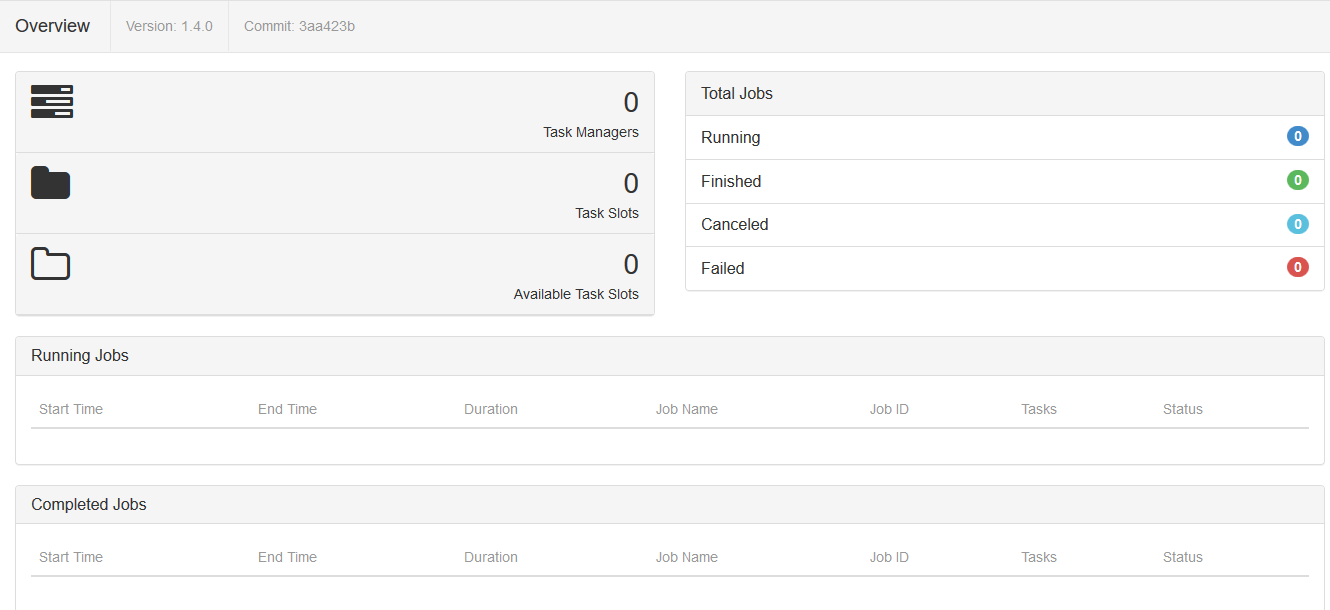
现象: Available Task Slots 0 Task Slots 0 Task Managers 0 过一会失败

原来以为原因是客户端 磁盘空间不足,需要申请内存的2.1倍磁盘

解决方式:换一台客户机,重建配置安全认证

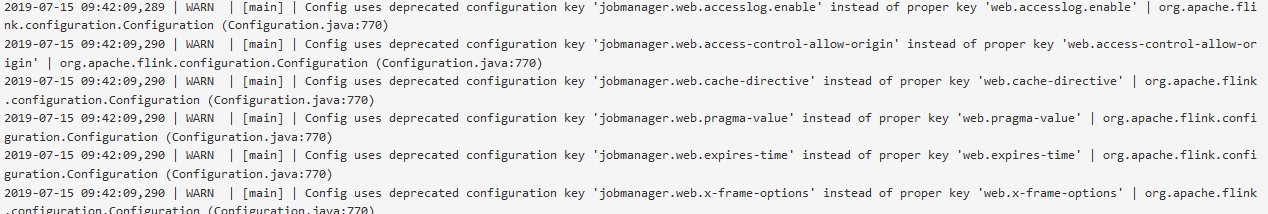
调试步骤:

新发布flink集群出现,资源分配不上的



在WEB-UI页面查看Job Manager 🡪 Logs 和 在程序发布的log中找:

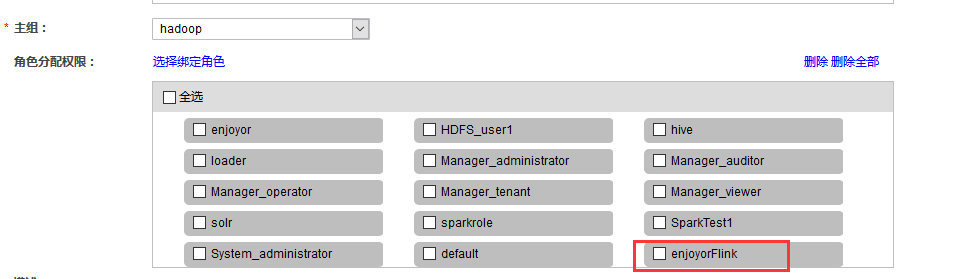
Config uses deprecated configuration key 'jobmanager.web.expires-time' instead of proper key 'web.expires-time' | org.apache.flink.configuration.Configuration (Configuration.java:770)



(上面不是主要原因,可忽略)

**A2:是不是划分租户导致的**

查看enjoyorFlink队列的权限 没有enjoyor用户, 增加后无效.



切换使用enjoyor队列执行仍然不行 ,修改成最小的1024 2048 发布仍然不行,加上上周已经使用enjoyorFlink部署成功过,所以排除这种可能

**A3: 资源占用问题:FlinkException: Could not connect to the leading JobManager**

参照网上博客 https://cloud.tencent.com/developer/article/1398884

这种错误一般是 hadoop 集群资源（内存、磁盘、虚拟内存等等）不足造成的

<property>

<name>yarn.nodemanager.vmem-check-enabled</name>

<value>false</value>

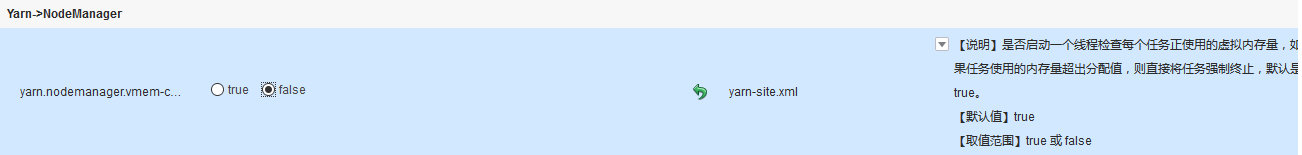
</property>

</configuration>

物理内存 充足 虚拟内存?? 虚拟内存:磁盘上划一块空间

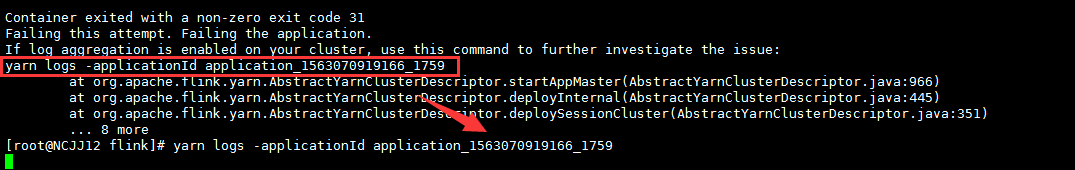
使用15G的spark程序yarn上运行时可以的,所以不一定是虚拟内存不够的原因

修改虚拟内存检查机制:

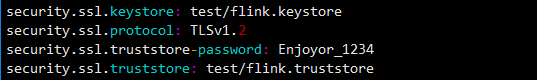


由于不能动线上,所以排除这种操作.

**A4:尝试迁移到12客户端测试,需要配置一遍kerberos和ssl安全认证;**

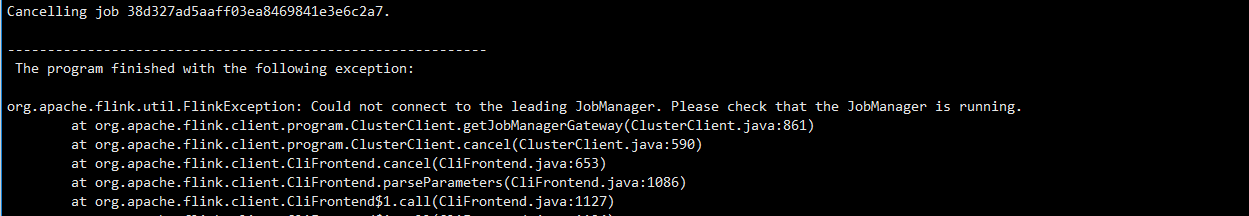


需要在12客户端生成认证,赋权限flink-conf.yaml文件 修改:

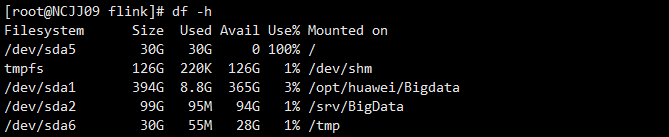


~~发布成功 ! 可以确定是磁盘空间不足导致的flink-cluster发布失败~~

南昌9客户机上flink cancel job 异常 ,可以kill



查看磁盘空间



删除数据 空出来1% 取消任务 仍然失败,抛

org.apache.flink.util.FlinkException: Could not connect to the leading JobManager. Please check that the JobManager is running.

解决: 将程序迁到 12客户机

九江194也出现这种情况,**同时磁盘程序充足**(排除上述问题原因)

需要对原来程序进行修改,使用yarn取消flink job 出现上述问题

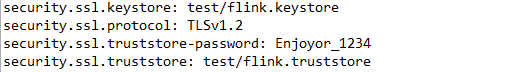
**怀疑是是证书过期了**



1:重新生成一下 sh generate\_keystore.sh Enjoyor\_1234

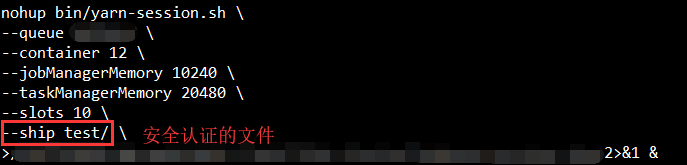
2:将这些文件放到flink目录下的test文件夹 赋权限

3:修改flink-conf.yaml文件 将修改为test/



4:上述做法原因:证书的有效期3个月

Flink-session发布的时候需要指定安全认证文件的路径,为了方便

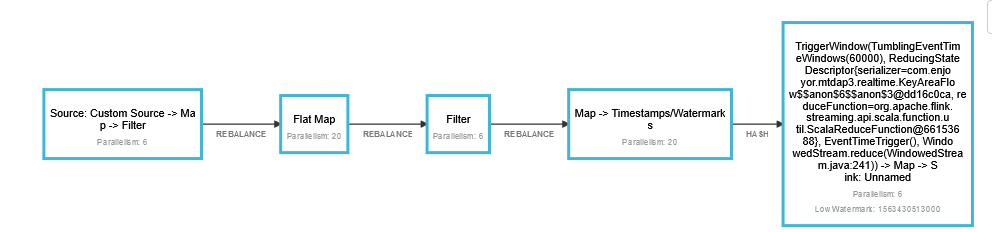


## 34:重点区域(一环二环)+外地车 调试

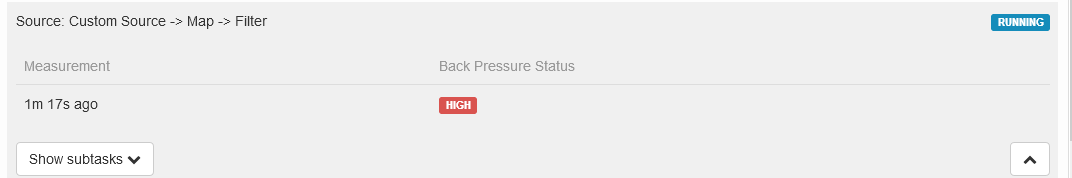
现在数据算不出来

使用6slot +60s+120s 算出来 但数据有滞后

给每个算子增加资源全局7 (根据不同的计算量给定不同的slot并行度)



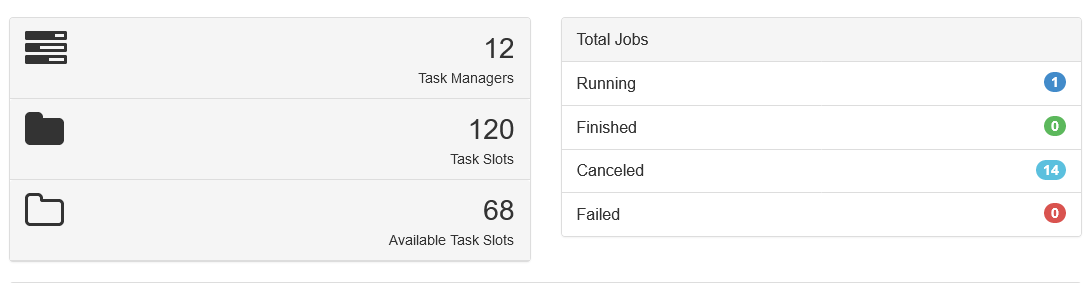
Rebalance重分区

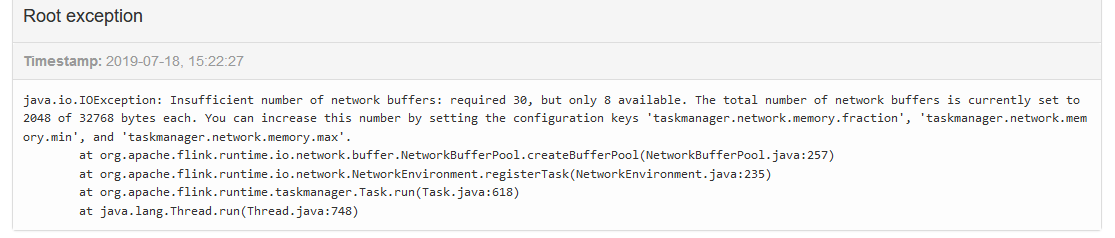


增加slot

背压太高导致数据延迟

使用这种小号的资源是累加的

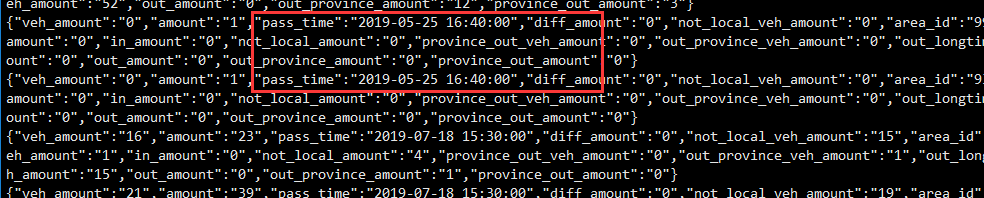




调大slot解决slot>1 slot<40 ,

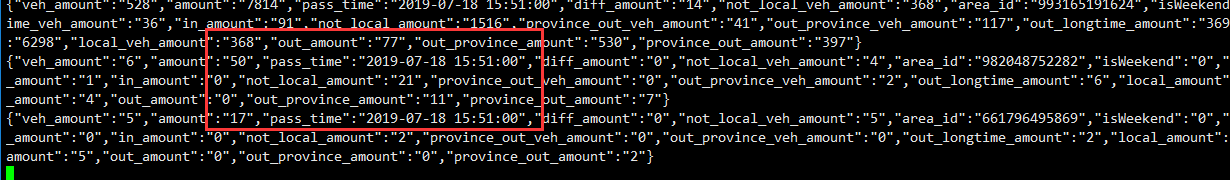
结果:

数据可以正常出来,但是页面检测背压高 过一会数据算不出来且数据值变个位数



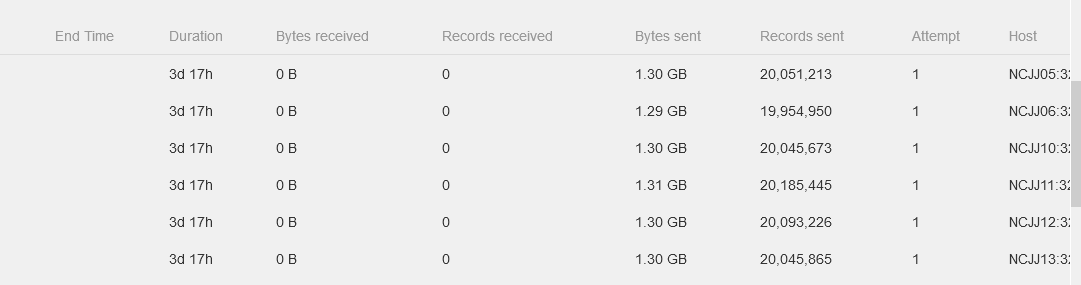
全部20 30

数据重发了



11

运行后会影响全流量计算 ,7G的数据不会自动清理吗?



## 35:Flink面试题

1.flink是怎样保证exact-once的保证；

为甚是要插入barrier来做checkpoint，而不是在不同算子的定时器来做checkpoint？

2.flink的checkpoint机制与恢复？

3.flink与storm/spark对比怎么样？为什么要从storm/spark迁移到flink上？

4.你在使用中怎样使用flink的？

5.源码是否已经阅读,举例？

6.flink中其中一个节点宕机之后，怎样恢复的？？

7. flink中流和批是怎样统一的？改动量是否很大？？

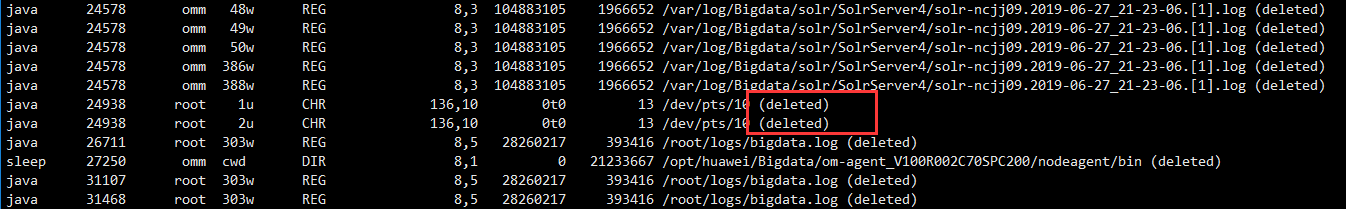
8: akka的actor system 在flink中作用

9:flink on yarn 提交流程

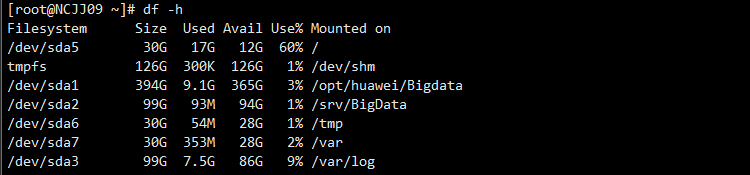
## 36: 处理磁盘已满但找不到对应的大文件的问题

lsof |grep delete

进程打开了一个大文件,被删除后,并没有释放.重启进程后， 才释放



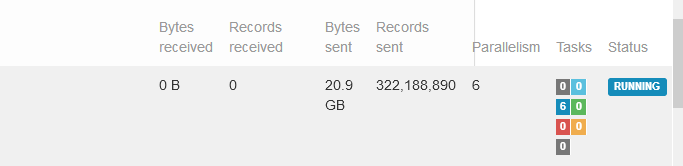
Kill -9 pid



## 37:长时间运行,未自动重启,出现数据滞后问题?

全流量计算,统计全市2w多卡口30s内过车流量,过车数据,等十几个计算指标

使用 keyBy(过车时间) + no-keyed window(并行度始终为1) 运行9天20h后出现数据滞后,且逐渐增加



Bytes send 达到20.9G 是不是这里导致的? 否

将job取消,重新发布,正常

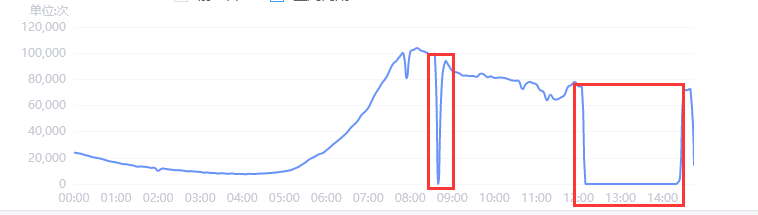


看下效果:



临时解决方式:手动重启(9天左右) 弊端 滞后期间的数据会丢失

根本原因:离线计算程序占用cpu 导致滞后



Flink 重启策略:默认的重启策略 是job失败只有启动checkpoint,才会重发重启策略

默认采用的fixed-delay(固定延迟),重试Integer.MAX\_VALUE次

怎样在代码中给不同job指定不同的重启策略??

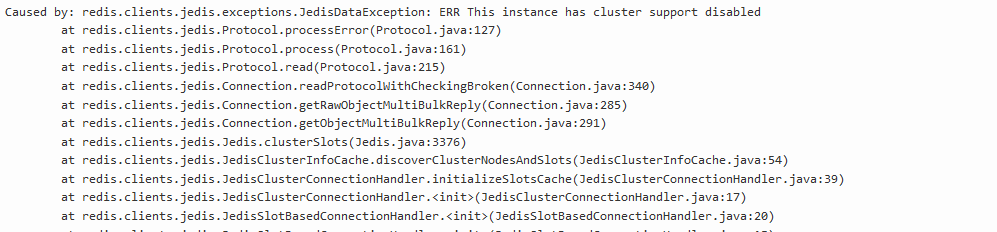
env.setRestartStrategy(RestartStrategies.fixedDelayRestart(3, // 尝试重启次数 Time.of(10, TimeUnit.SECONDS) // 延迟时间间隔10s));

程序优化:中间丢失的数据需要使用sparksql 离线计算写到phoenix中 ,补上(TODO)

## 38: redis 导致的问题

ERR This instance has cluster support disabled 原因:redis连接有问题

java.util.NoSuchElementException: None.get 原因 redis中为null



## 39:HUE 监控yarn执行任务



## 40:流量突发预警

业务规则:

实时路段流量跟该路段30天内当前时段流量均值作对比,涨跌幅>20%的发出预警

开发时遇到:

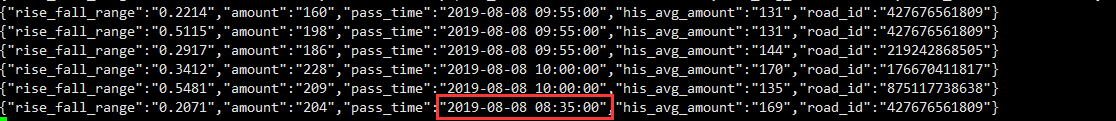
代码调试阶段,接着kafka消费组消费,会出现数值很小的结果集,导致指标结果跟历史比对

相差较大,触发预警 需要将这部分数据拦截下,大概有两种思路

1:每次发布使用新的消费组

2:程序发布第一批结果集计算的指标不进行预警

问题2:出现这种



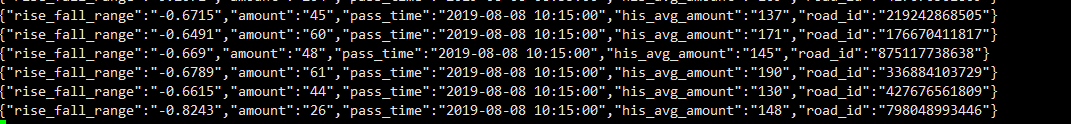
巡检代码:使用的不是事件时间, 处理时间?



上述问题解决:修改为事件时间



修改消费组发布后: 得到的结果集仍然是少的(是程序停止时成圣的)



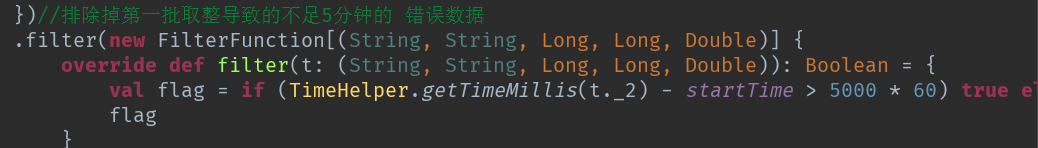
程序开启时间10:18- 10:20区间这么多过车,由于5分钟取整🡪10:15上,导致这个原因



这部分数据需要 剔除掉

容器外获取执行时间,在输出是判断时间是否是第一次窗口的,是就排除掉





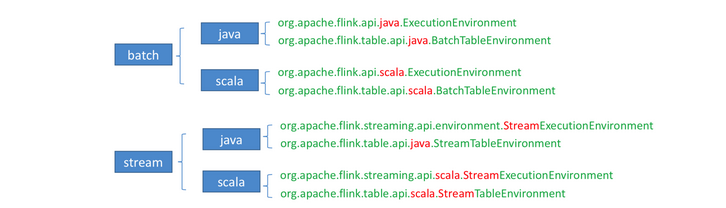
## 41:flink sql && Table api

Sql支持不完善,很多算子都是高版本才加进来,先使用flink1.7.0进行研究测试



Jar 包需要都替换成1.7.0的 ,线上开发1.4.0 研究使用1.7.0

## 42:flink environment归纳



## 43:并行度太多,导致生成barrier对齐太久,checkpoint超时

30个并行度barrier对齐时间超过50s

原因在于Flink在生成checkpoint的过车中,会在source的插入barrier与正常消息一起往下游发送,算子在等到指定的barrier会触发checkpoint,这是一个流

如果多个流同时进入一个算子处理会些,flink在生成checkpoint时,发现多个流进入到一个算子,先进入的这个算子的barrier对应的那段消息就会buffer到算子中等待另外的流对应的barrier也到达时才会触发checkpoint,如果其他的算子没有到,就会一直等待,这个buffer等到的时间就是barrier对齐

调优策略:

**1,尽量减少并行度**

**2:使用at least once 语义代替exactly once**,这样checkpoint时就不会做barrier对齐,数据到了算子马上做checkpoint并发送到下游

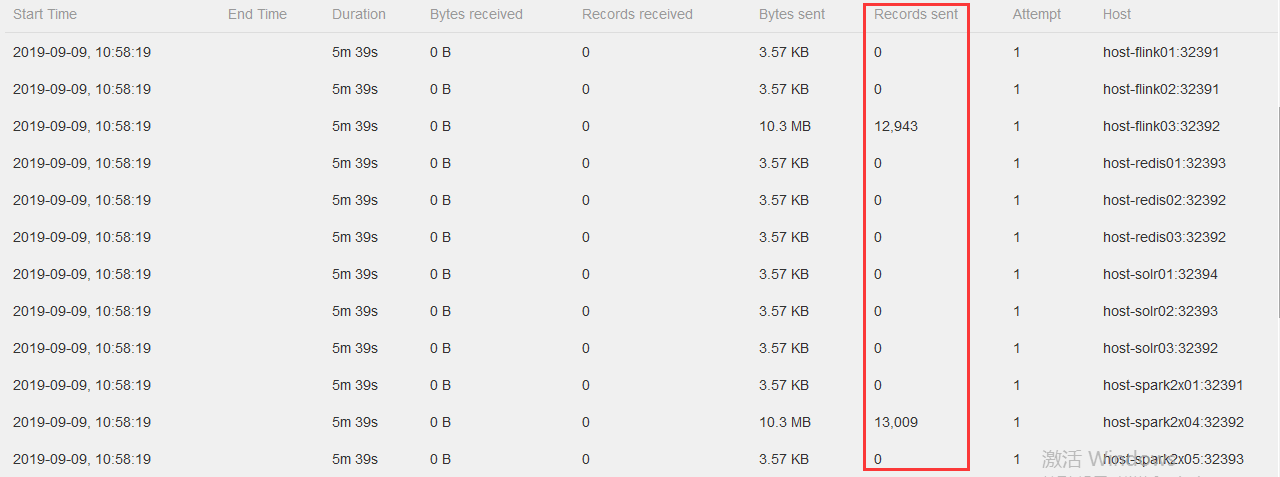
使用哪种按照业务场景,

如果使用at least once数据保证就能满足业务需求的尽量用at least once语义。

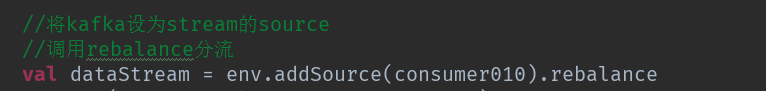
如果不支持的，就减少并行度以此减少barrier对齐的数据量和时间

## 44:数据倾斜 解决

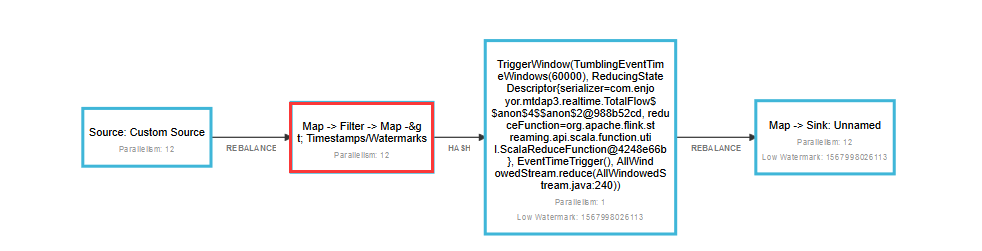
修改前,使用12个slot 出现

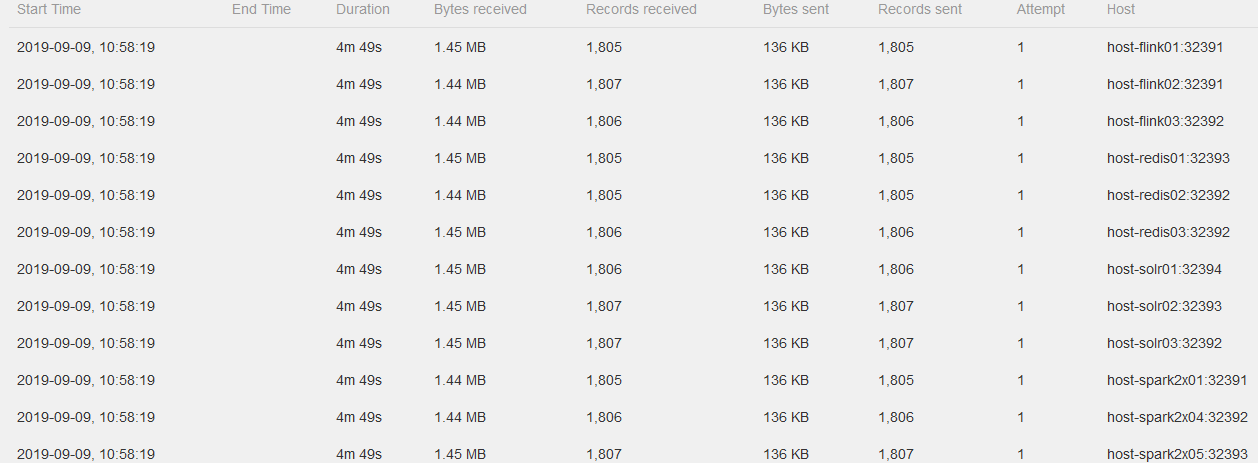


数据倾斜,资源浪费,且需要网络IO同步,效果还不如6个slot



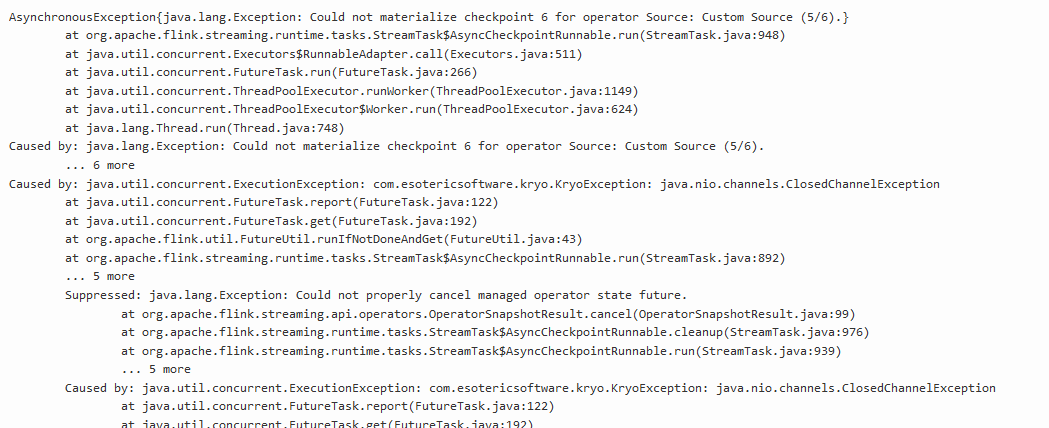
页面效果:





## 45: Could not materialize checkpoint 6 for operator Source: Custom Source (5/6)

全流量计算 使用 6slot + rebalance 发布 ,运行44s抛:

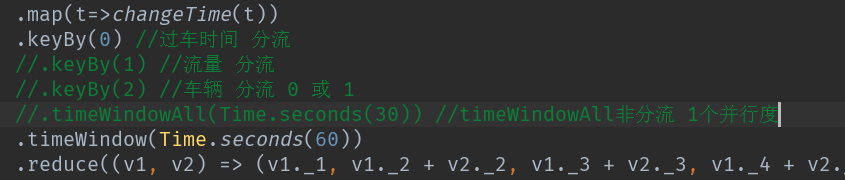


## 46: 全点位流量计算 timeWindowAll –> timeWindow

如果白天离线程序重新跑数据会对实时计算争夺资源 就会出现反压现象

keyBy(0)+ timeWindow 会出现多个过车时间分流的结果

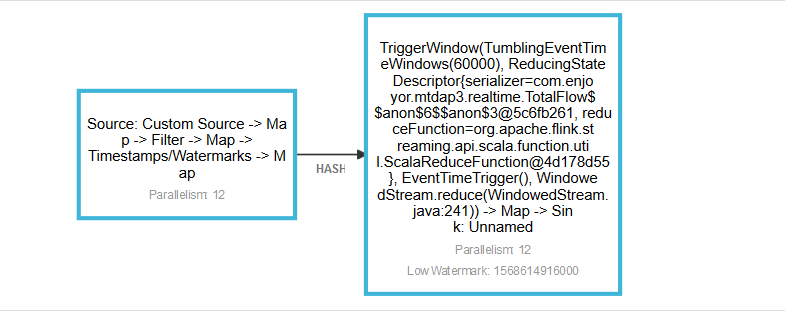
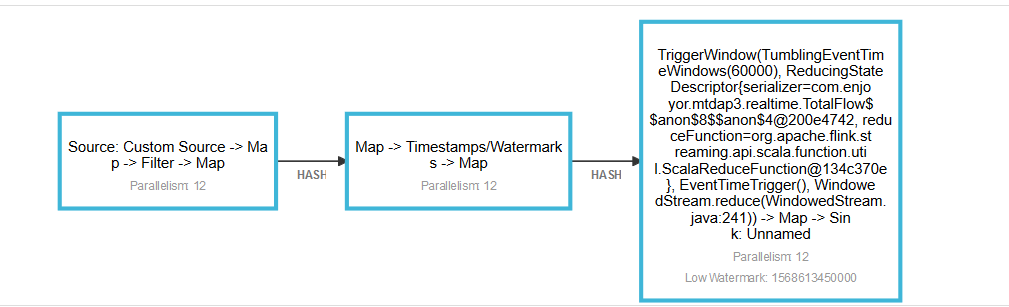
changeTime(\_)先将最近1分钟的过车时间取余 可以实现，但是相当于没有分流



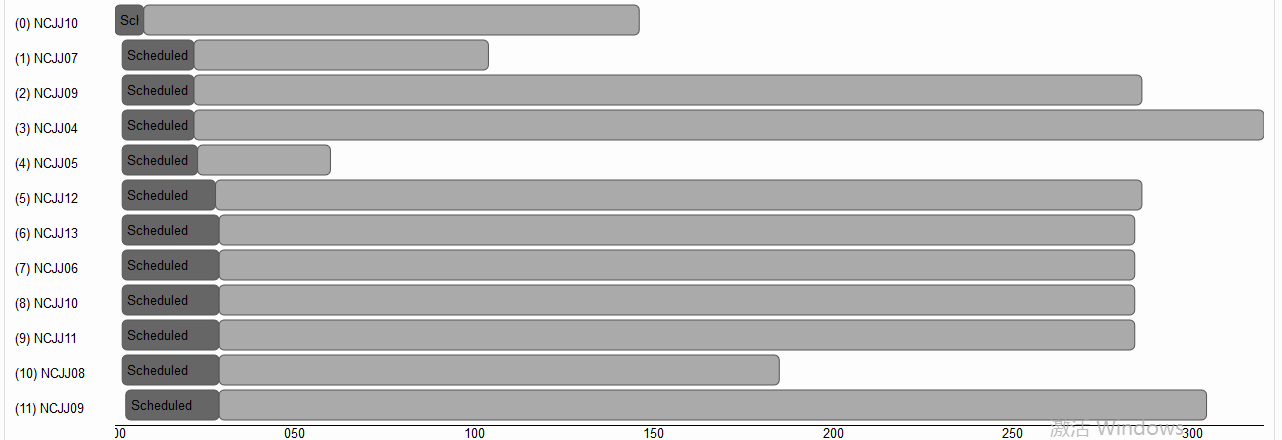
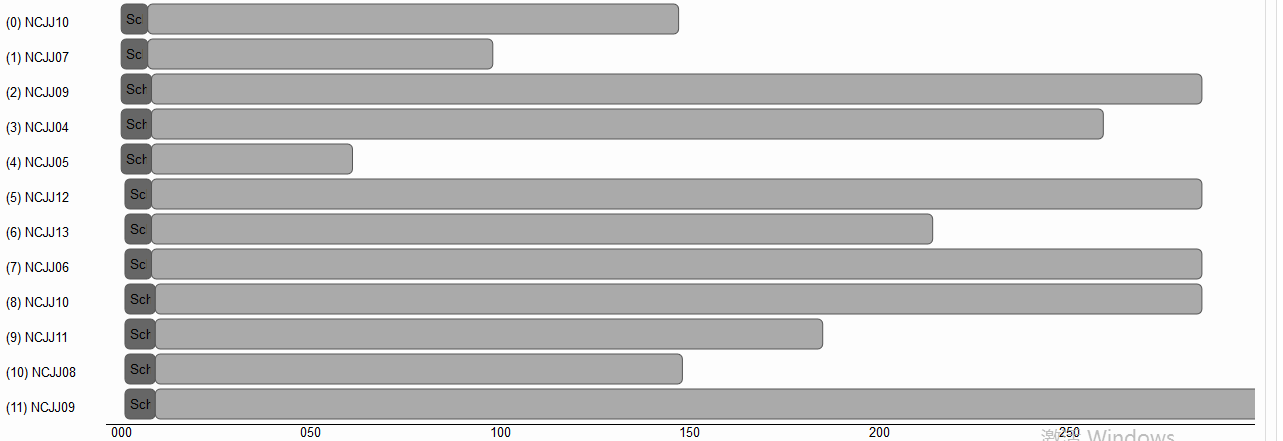
V2.0思路:先对程序进行10秒钟进行一次分流，再进行map计算,再汇总

DataStream –> KeyedStream –> DataStream

流通过keyBy分流,去掉rebalance (反压低)



右边的就是比左边的节约资源!timeline查看分别两个步骤:



参考43对比

## 47:flink-session突然挂掉 yarn页面找不到记录

New JobManager elected. Connecting to null

org.apache.flink.client.program.ProgramInvocationException: The program execution failed: Couldn't retrieve the JobExecutionResult from the JobManager.

Caused by: org.apache.flink.runtime.client.JobExecutionException: Couldn't retrieve the JobExecutionResult from the JobManager.

Caused by: org.apache.flink.runtime.client.JobClientActorConnectionTimeoutException: Lost connection to the JobManager.

之前发生过一次,相同的多jobsession,可能是程序部署的过多了,在yarn发生主备置换时 出异常

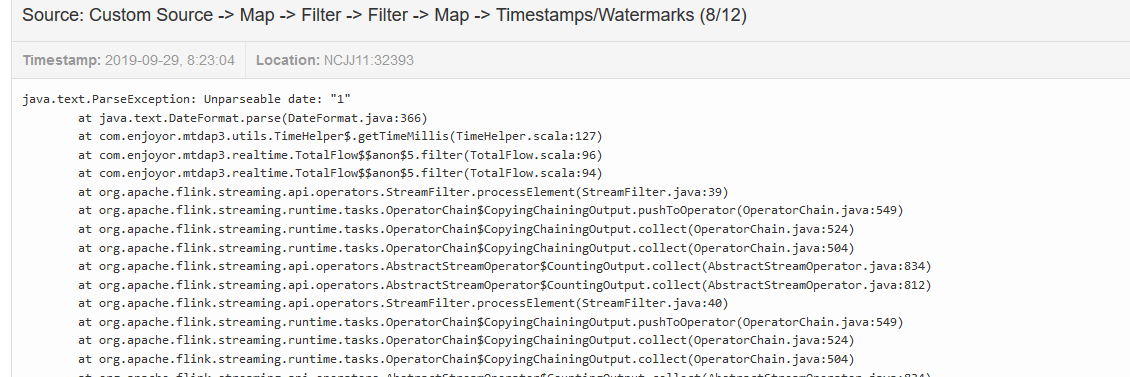
解决方式:每个yarn-session中job拆分,job数不超过container数

48: yarn-session 中任务滞后 争夺CPU 挂掉了



分成2个yarn-session

全点位流量异常:



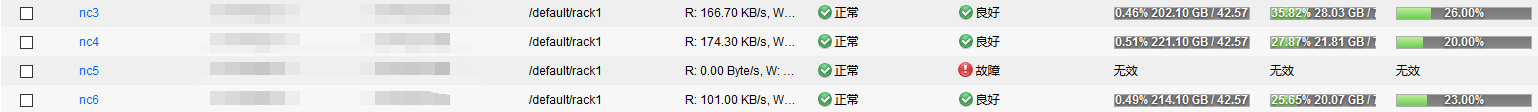
## 48:Flink程序全点位流量计统计 出现超前的bug(窗口被提前触发)

Kafka 发送的数据过车时间超前(是卡口设备时间错乱导致)

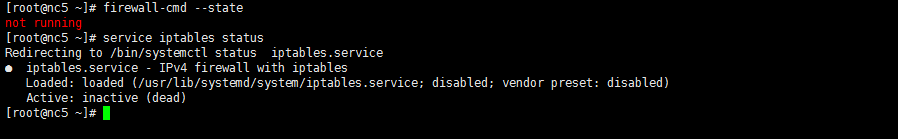
最终导致 窗口提前被触发(16:00 时 16:02 16:03 16:04 16:05数据都出来 )

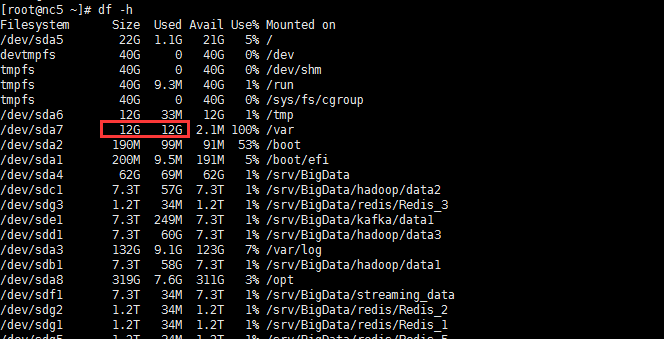
解决: 只能把这部分数据过滤掉,避免影响到程序正常运行

## 49:集群主机故障 导致多个服务包括HDFS故障 和 计算结果出问题



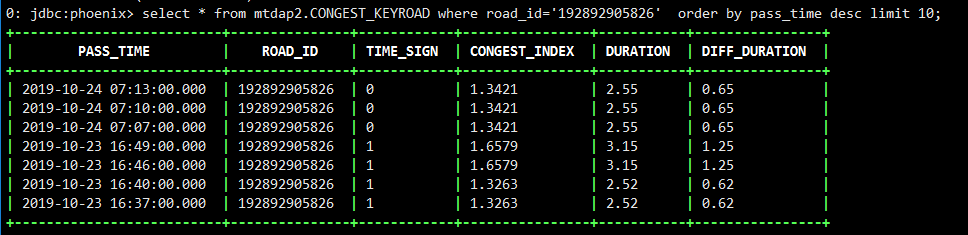
查看服务器nc5,防火墙未关闭

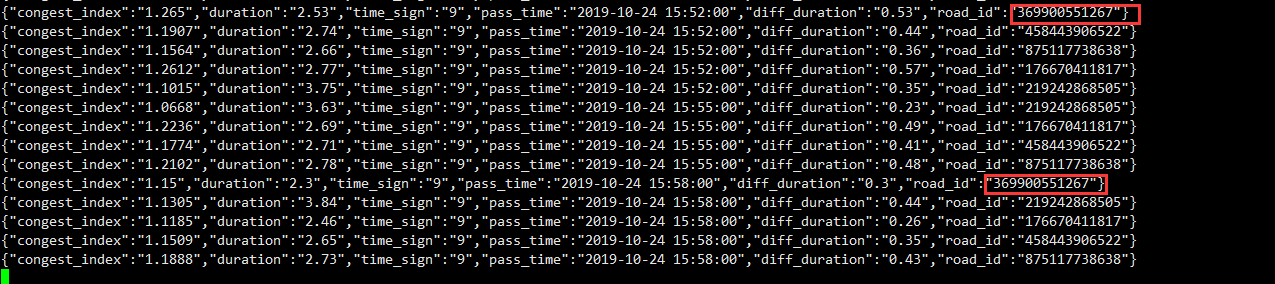




删除日志 Reboot重启 故障仍在,最终原因是该台服务器硬件故障

## 50:重点路段拥堵 配置路段偶尔有数据





实现流程:

实时:10分钟窗口,3分钟滑动,一次获取最近十分钟数据,

处理后形成轨迹 取相邻两点位id,来关联配置好的重点路段

离线:预计算有轨迹两点位 闲时时长(凌晨畅通状态下的过车时间s)

配置:页面配置要计算的路段(开始点位-结束点位)

拥堵指数:需要满足

条件1:dws.congest\_road\_info中开始结束点位 对上(spark预计算)

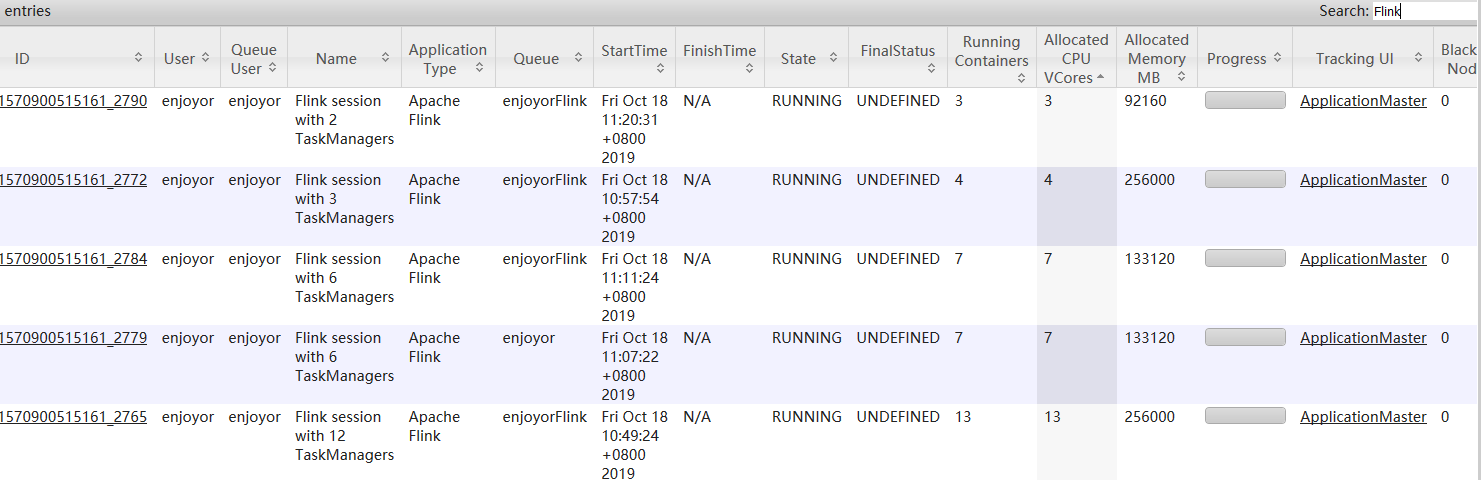
条件2:最近十分钟,有车从这两个点位经过(直达,中间没经过中间卡口)

且卡口数据传输不延迟(kafka数据质量)

由于数据质量不高,会出现很多配置的路段某个时段没有拥堵指数

最终解决方式:如果最近路段没有拥堵结果,输出默认值1.0

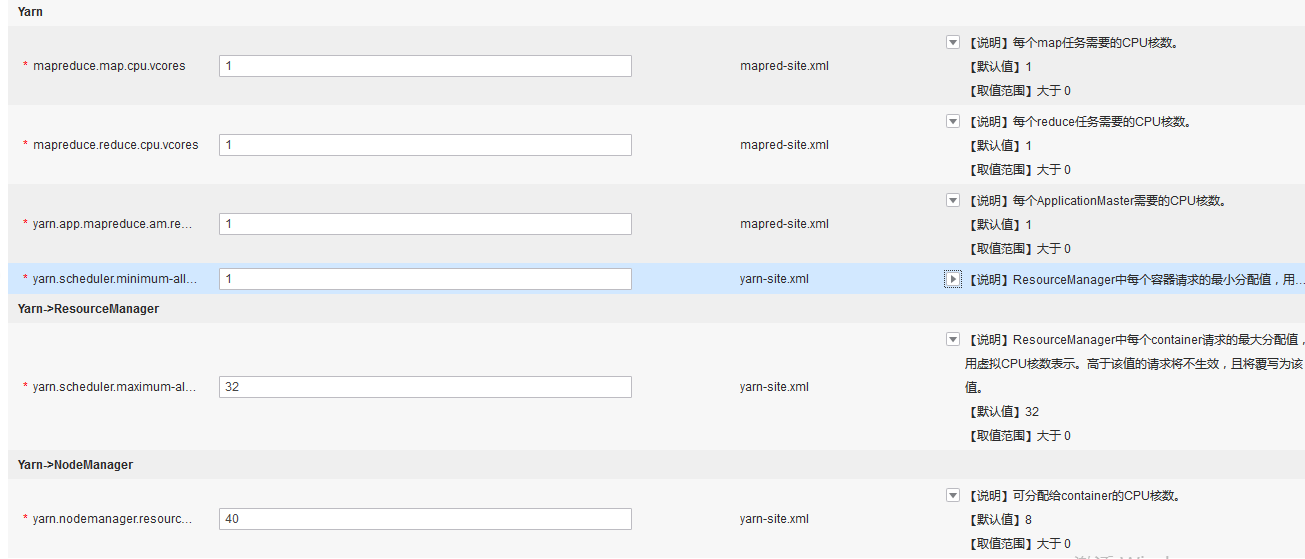
## 51:flink-conf.yaml配置问题:



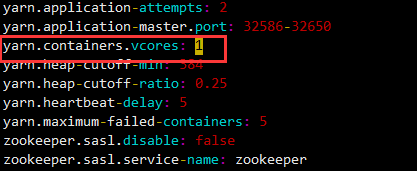
截图:running container 7 (实际上设置Task Managers为6,多出来一个是作为ApplicationMaster和JobManager的)

疑问?每个节点的虚拟核心数配置为3，但Yarn只有1个虚拟核可用,导致CPU紧张

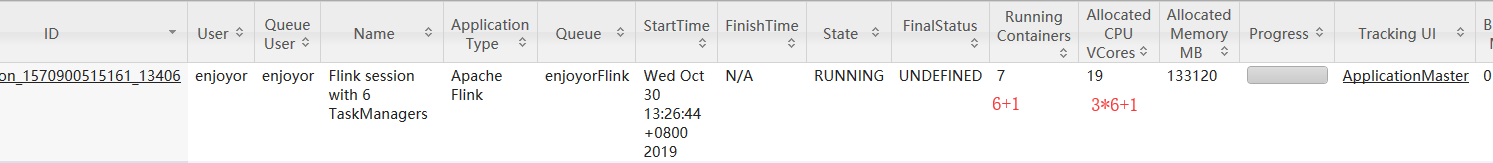
先查看yarn配置



再查看客户端yarn-conf.yaml



修改下: yarn.containers.vcores: 1 -> 3 重新启动yarn-session结果如下图:成功



网上其他解决方式: (不适用我们,集群上这个参数配置过,是40)



## 52:ResourceManager默认内存引发问题

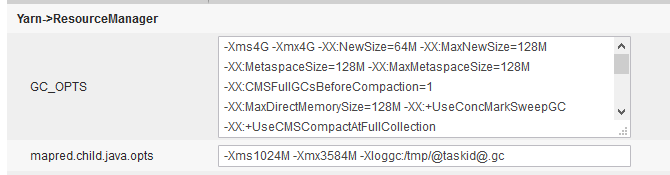
现象:JK区可视化部署后,早高峰时数据服务经常响应慢

(页面5分钟刷新请求一次)

查看集群告警:发现往往伴随yarn的RM GC



这种GC一般都是给定堆内存太小,查看yarn配置,果然默认给定的是512m 1G 修改🡪



先同步配置,晚上重启yarn

## 53：Could not connect to the leading JobManager.

常见的出现原因：

1：kerberos 导致的

Flink生成的认证文件默认90天，之后取消job会抛这个异常

解决：需要重新生成证书，认证一下

2：未指定yarn appId导致

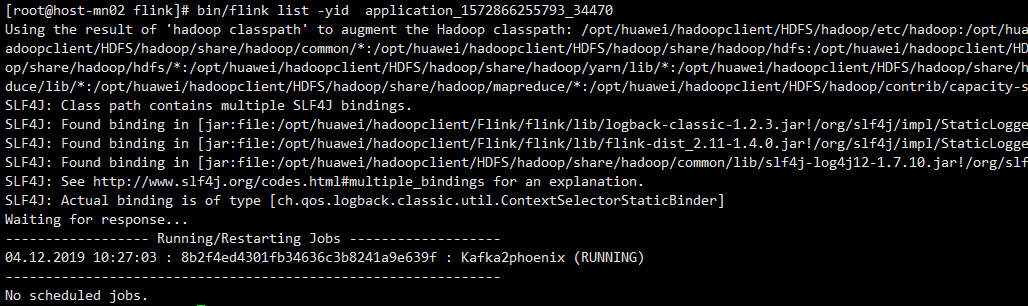
比如：bin/flink list 和 bin/flink savepoint -yid application\_1572866255793\_17760 会抛

需要指定到具体的id

如：bin/flink list -yid application\_1572866255793\_34470

bin/flink savepoint -yid application\_1572866255793\_17760

8b08d32f2f0be5c5b6d064576768f20c



## 54：savepoint（保存点）

flink savepoint:

Savepoints在持久化存储中保存某个checkpoint,以便用户可以暂停自己的应用进行升级,

并将状态设置为savepoint的状态,并继续运行.

该机制利用了Flink的checkpoint机制创建流应用的快照,并将快照的元数据(meta-data)写入到一个额外的持久化文件系统中

1:每个算子需要id手动设置UID

程序自动生成(跟应用结构算子有关)

建议手动给每个算子通过UID(String)分配一个固定的id

2:Savepoint产生的数据将被保

存到配置的文件系统中,如FsStateBackend或者RocksDBStateBackend

触发：

bin/flink savepoint jobId 路径[选,不写读flink-conf.yaml文件]

停止并触发：

bin/flink cancel -s [路径] jobId

3:从savepoint恢复作业:

bin/flink run -s savepointPath [runArgs]

允许不恢复某个算子的状态：

bin/flink run -s savepointPath -n [runArgs]

#默认情况下，系统会尝试恢复savepoint的状态全部映射到用户的流应用

如果代码有修改(删除某个算子)，可以通过--allowNonRestoredState(简写-n)恢复状态

4:清除savepoint

bin/flink savepoint -d savepointPath

---------------------------------------------------------------------------------------------------------------------------------

生成savepoint： ok

bin/flink savepoint -yid application\_1572866255793\_17760

8b08d32f2f0be5c5b6d064576768f20c

查看生成的savepoint：ok

hdfs dfs -ls /flink/savepoint/

取消job，并生成保存点,ok

bin/flink cancel -yid application\_1572866255793\_17760 8b08d32f2f0be5c5b6d064576768f20c -savepoint hdfs:///flink/savepoint

启动job：接着保存点运行 ok

bin/flink run -s hdfs:///flink/savepoint/savepoint-8b08d3-c077f4bd59b0 -m yarn-cluster -yn 2 -yjm 1024 -ytm 2048 -yt test/ --class com.xxx.xxx.xxx.Kafka2Phoenix /opt/xxx/xxx/xxx/xxx-1.0.jar

后台运行：ok

nohup bin/flink run -s hdfs:///flink/savepoint/savepoint-8b08d3-c077f4bd59b0 -m yarn-cluster -yn 2 -yjm 1024 -ytm 2048 -yt test/ --class com.xxx.xxx.xxx.Kafka2Phoenix /opt/xxx/xxx/xxx/xxx-1.0.jar >/opt/xxx/xxx/xxx/xxx-1.0.log &