CCTC 2017 中国云计算技术大会
Cloud Computing Technology Conference 2017
CSDN

基于Spark的流式处理引擎在Pandora大数据产品中的应用

简单 · 可信赖

赵宏尧
zhaohongyao@qiniu.com

主要内容

- Pandora
- 计算平台架构简介
- 流处理技术服务化需要考虑的两个问题
  - 用户的使用接口
  - 技术细节的屏蔽

主要内容

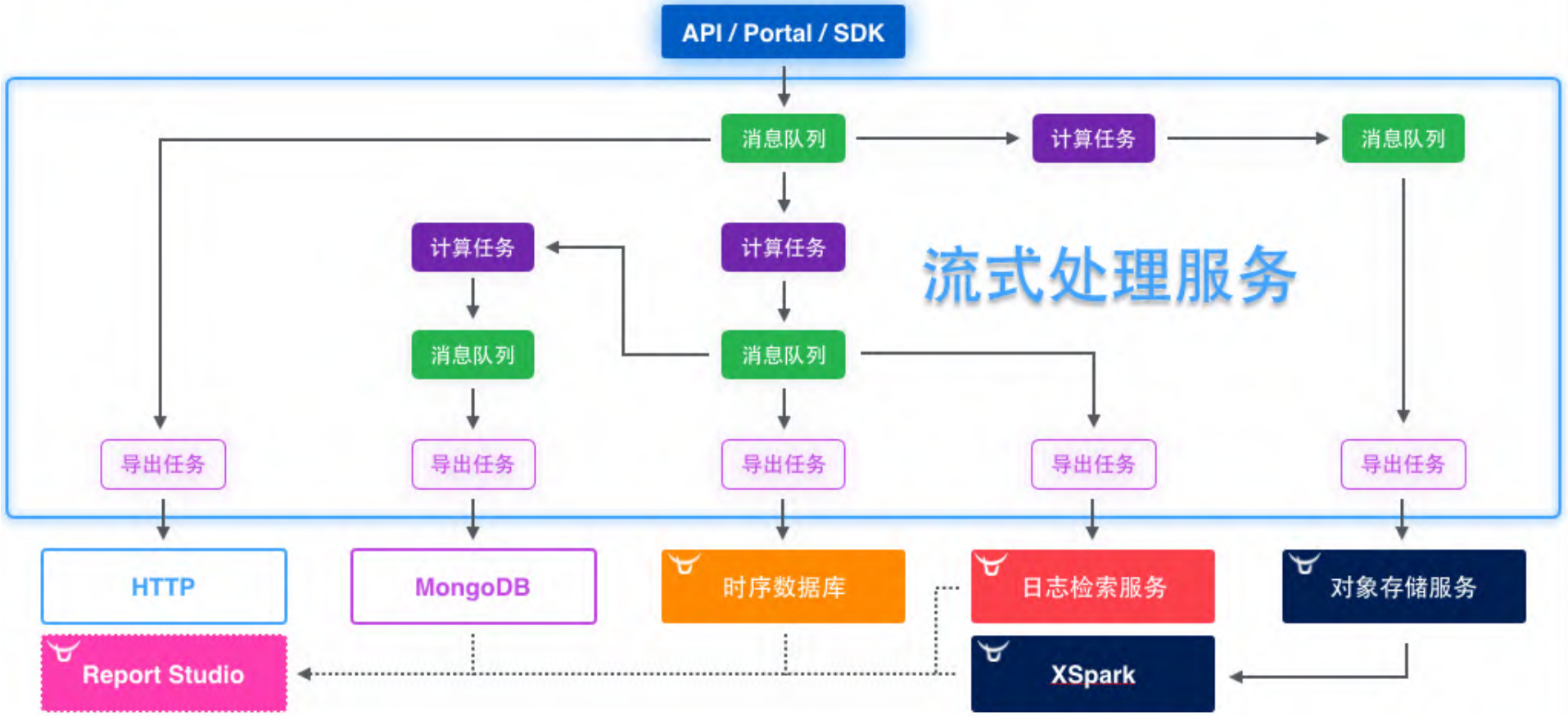
- Pandora
- 计算平台架构简介
- 流处理技术服务化需要考虑的两个问题
  - 用户的使用接口
  - 技术细节的屏蔽

Pandora规模

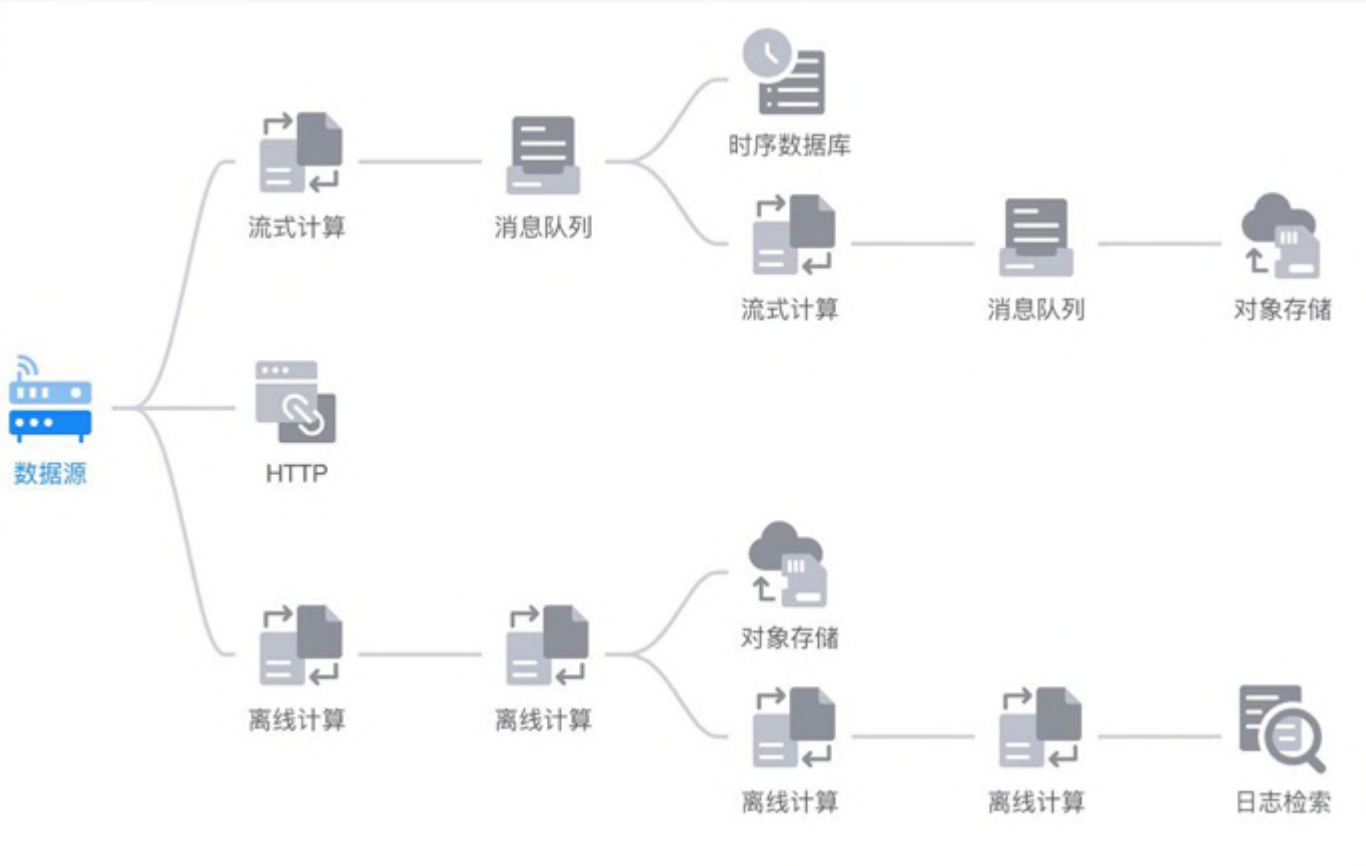
- 每分钟实时写入的数据量达到数百GB
- 每分钟实时写入的数据条目达到数十亿

简单·可信赖

Pandora架构图

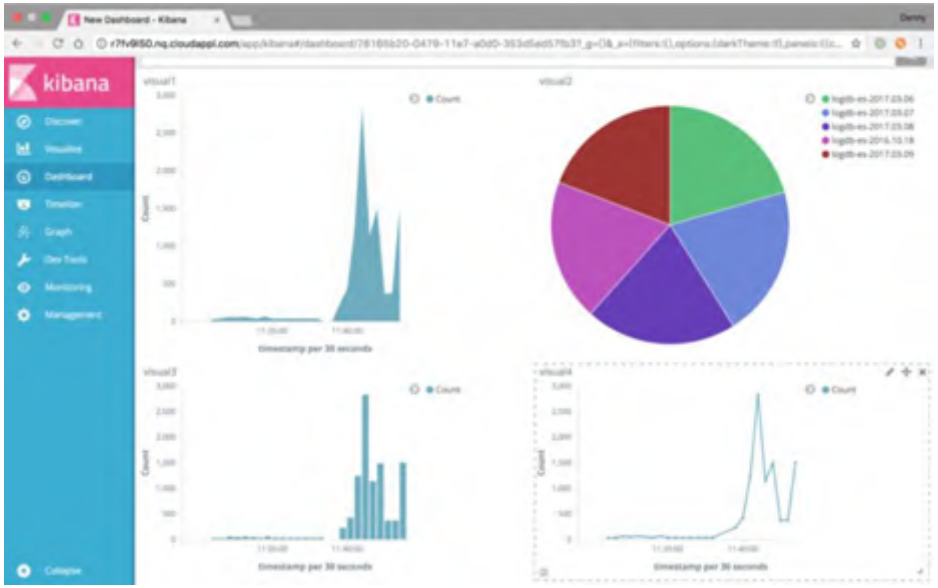


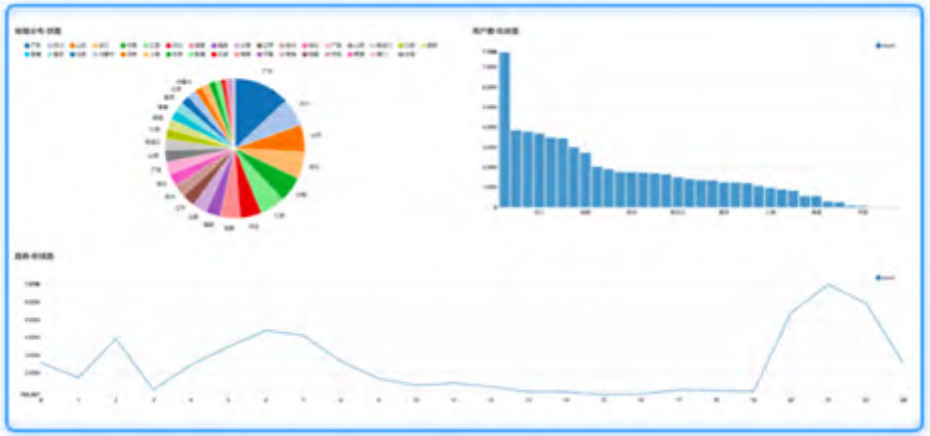
workflow


数据源
流式计算
消息队列
时序数据库
流式计算
消息队列
对象存储
HTTP
离线计算
离线计算
对象存储
离线计算
离线计算
日志检索

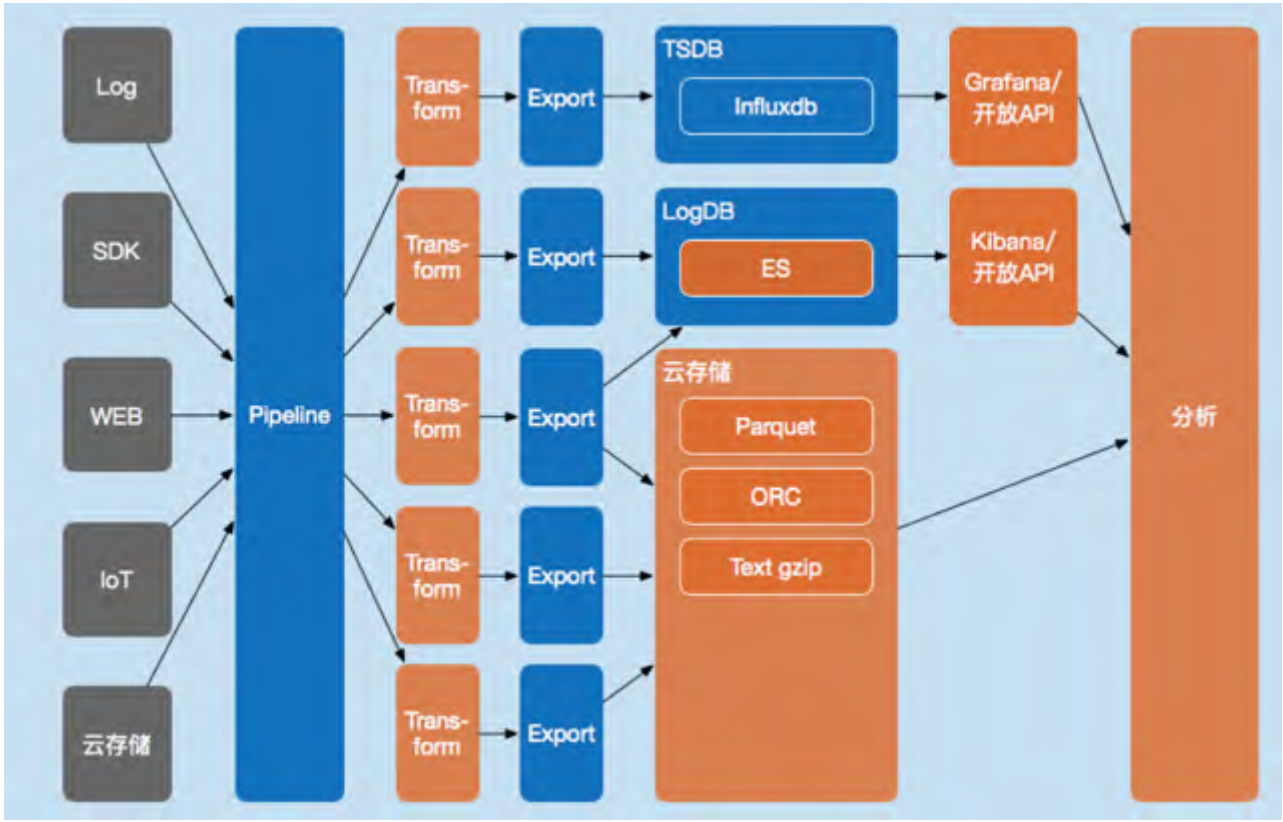
开放生态


New dashboard
cpu idle
disk inode free
memcached connections
mongo netin bytes


kibana



Pandora 整体技术架构



其它一些有意思的事情

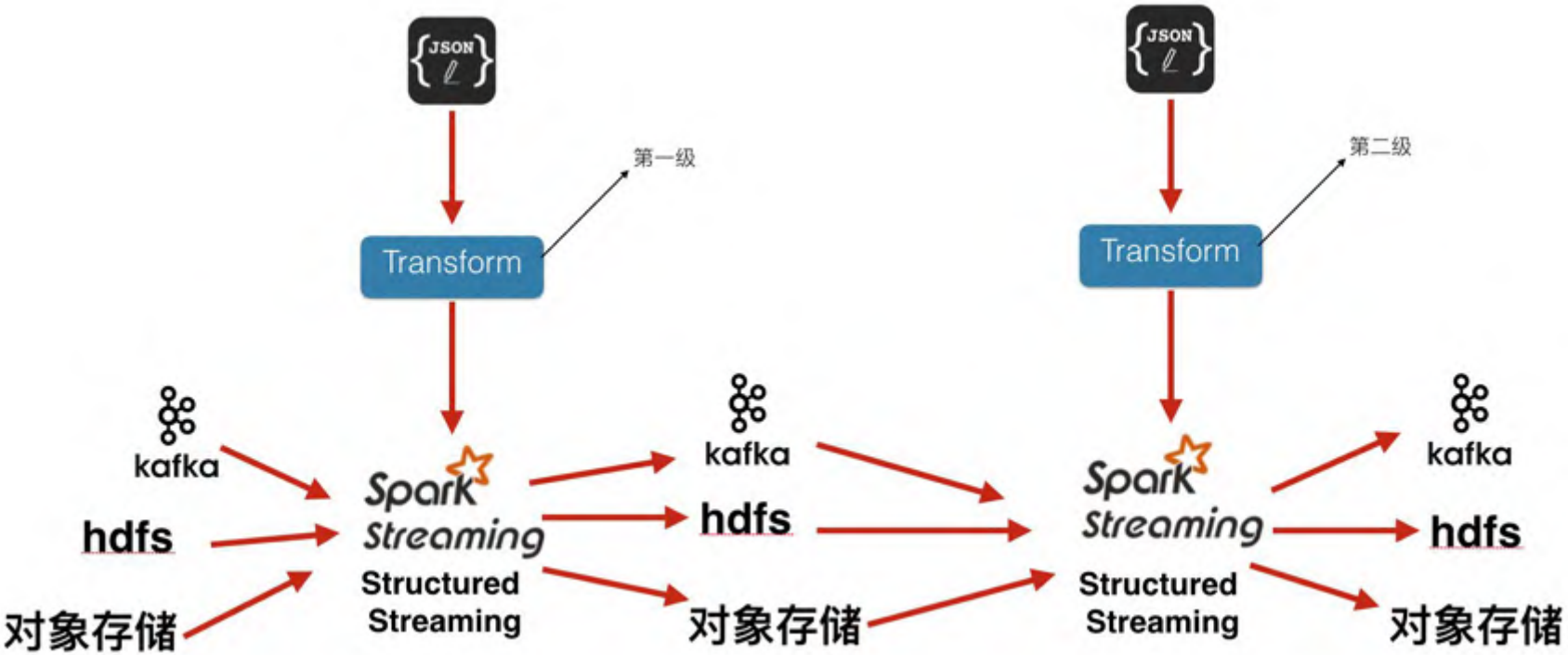
- 开发了一套分布式的goroutine框架
- 基于Golang做了一套轻量级的类Flume的组件
- 基于Golang写了一套分布式计算框架(主要用于TSDB)
- Workflow中整合实时计算和批量计算的调度
- 基于自研容器云做了一套Spark应用
- 基于七牛云存储，实现了HDFS接口
- ……

主要内容

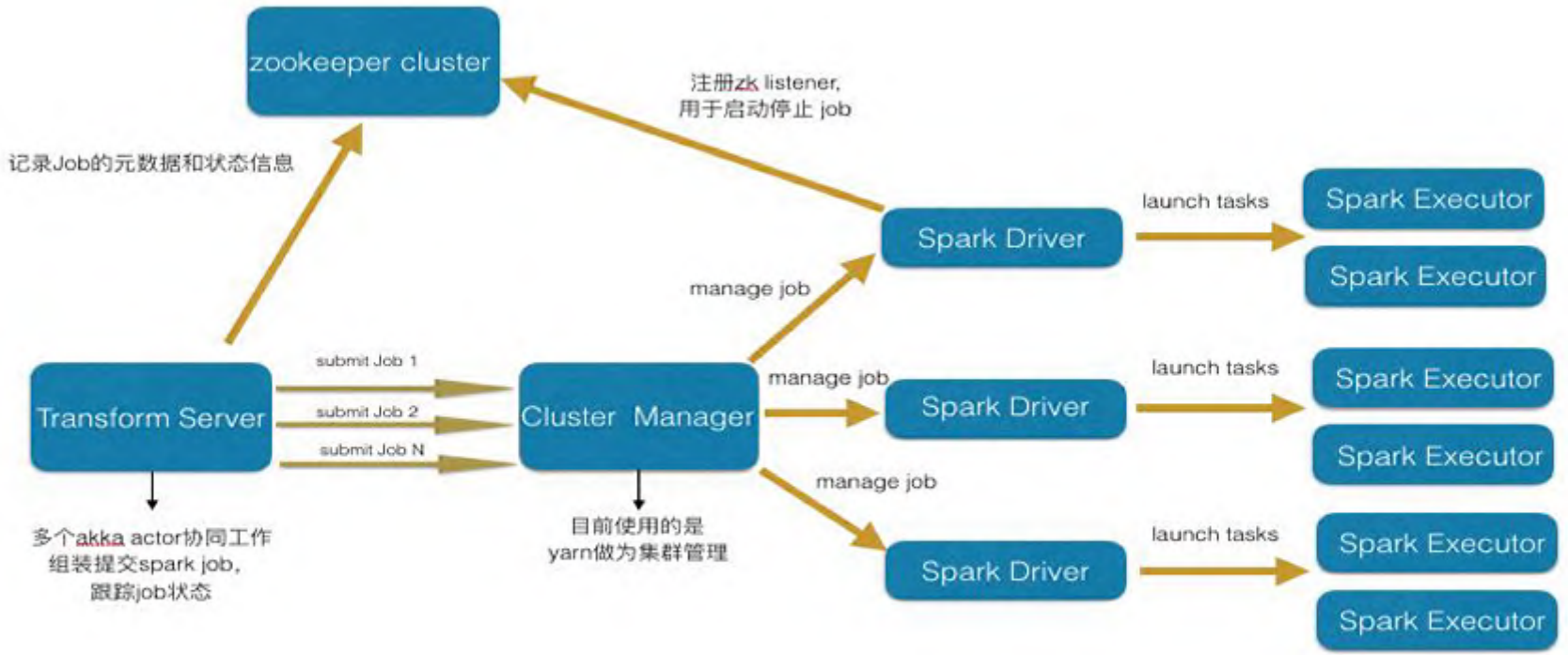
- Pandora
- 计算平台架构简介
- 流处理技术服务化需要考虑的两个问题
  - 用户的使用接口
  - 技术细节的屏蔽

计算平台架构简介

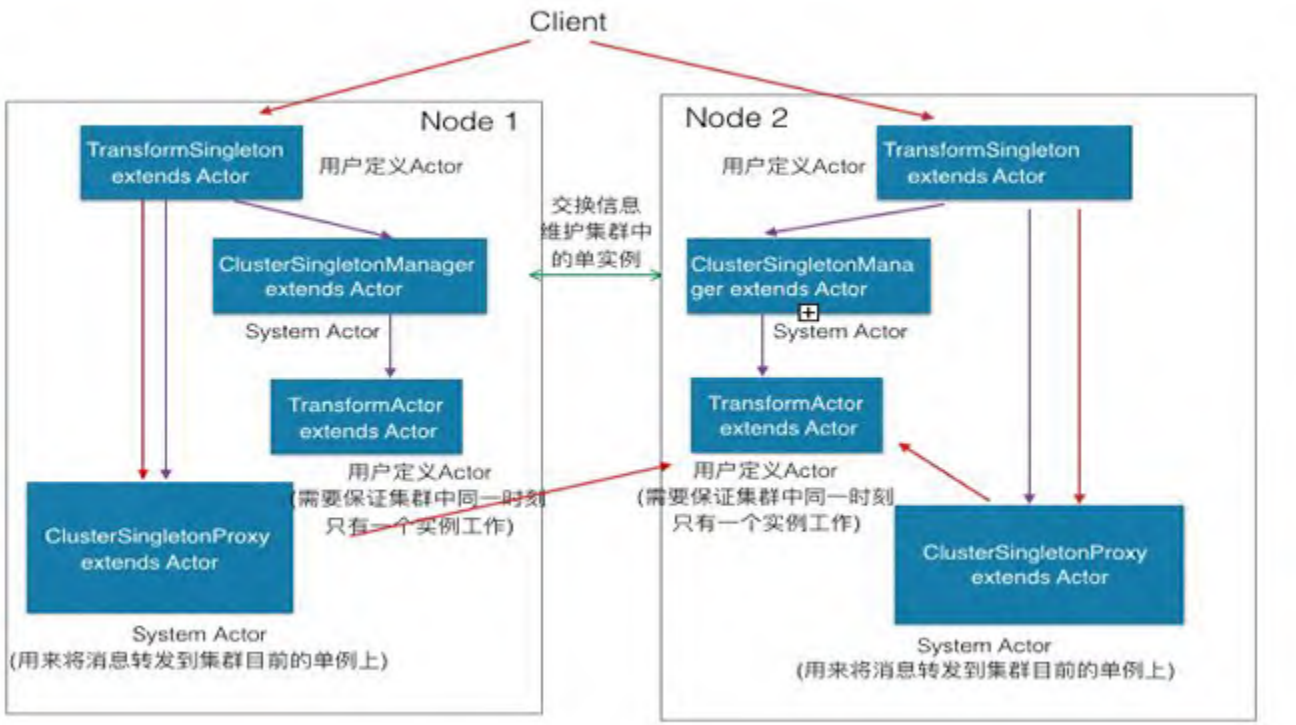
任务管理调度平台示意图



架构实现


zookeeper cluster
注册zk listener,
用于启动停止 job
记录Job的元数据和状态信息
Spark Driver
launch tasks
Spark Executor
Spark Executor
manage job
submit Job 1
submit Job 2
submit Job N
Transform Server
Cluster Manager
manage job
Spark Driver
launch tasks
Spark Executor
Spark Executor
manage job
Spark Driver
launch tasks
Spark Executor
Spark Executor
多个akka actor协同工作
组装提交spark job,
跟踪job状态
目前使用的是
yarn做为集群管理

Transform Server的高可用设计



主要内容

- Pandora
- 计算平台架构简介
- 流处理技术服务化需要考虑的两个问题
  - 用户的使用接口
  - 技术细节的屏蔽

目前实现

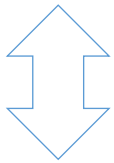
- SQL
- 自定义计算 (称为plugin, 目前支持Java和Scala)

SQL

- 基于spark streaming，是一个单batch的SQL
- 基于structured streaming，是一个跨batch的SQL

Structured Streaming SQL

```
select count(word) from stream
group by word, slide(timeStamp, INTERVAL '5' HOUR,
SLIDE '1' HOUR), watermark(INTERVAL '30' MINUTES)
```

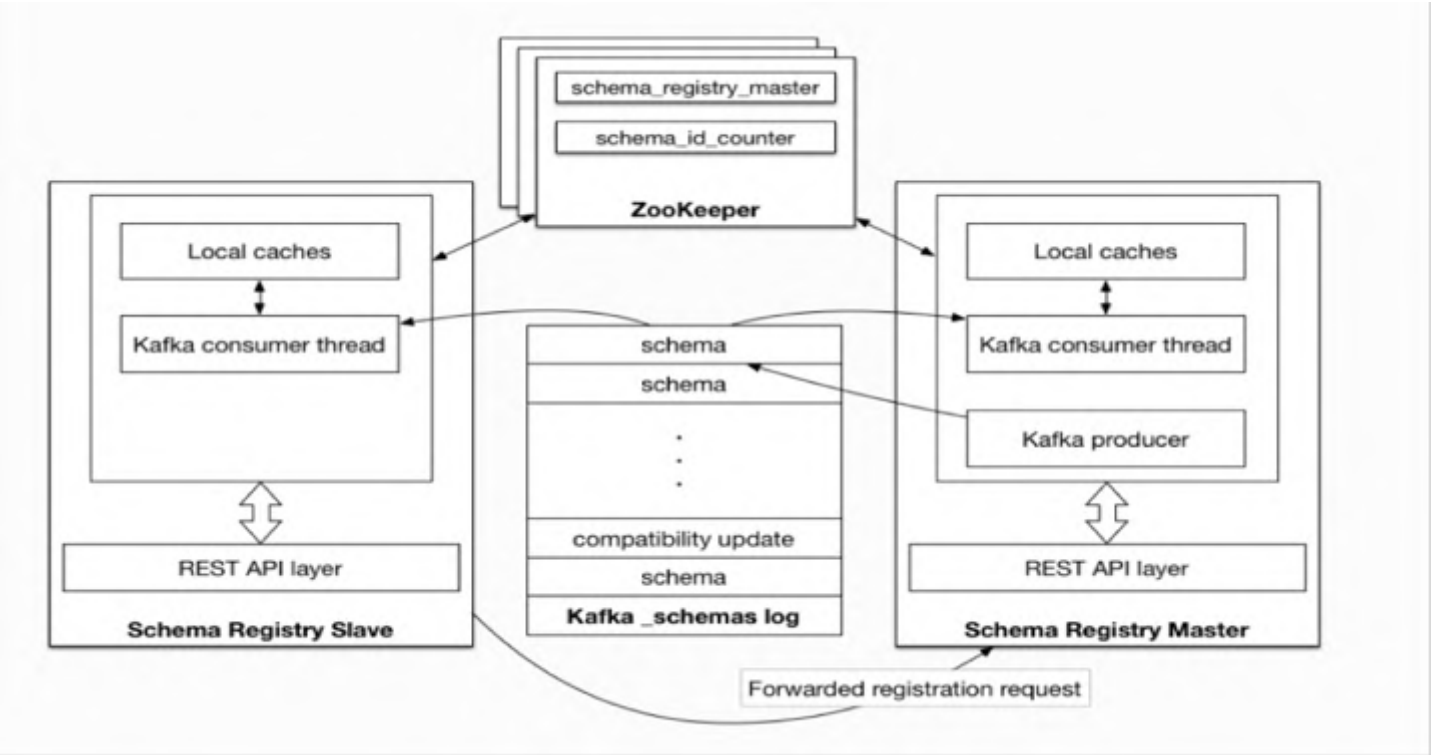


```
stream
    .withWatermark("timestamp", "30 minutes")
        .groupBy(
     window($"timestamp", "5 hours", "1 hours"),
          $"word”)
        .count()
```

使用schema有哪些好处

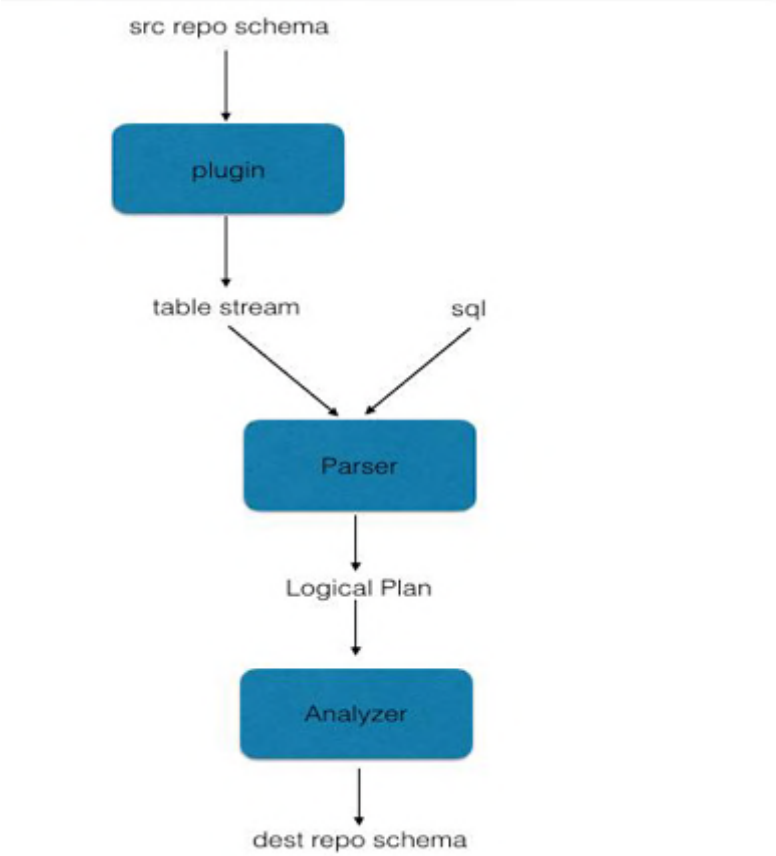
- 保证数据流的健壮性
- 数据格式schema的演进
- 高效存储和计算
- 丰富的数据类型

实时任务的shcema:Confluent avro schema registry



离线任务的shcema: Parquet

目的数据格式schema的推导



目的数据格式schema的推导

```
val sql = "select a, b from stream"
val caseInsensitiveConf = new SimpleCatalystConf(false)
val sparkSqlParser = new SparkSqlParser(new SQLConf)
val sessionCatalog = new SessionCatalog(new InMemoryCatalog, FunctionRegistry.builtin.copy(), caseInsensitiveConf)
val logicalPlan= sparkSqlParser.parsePlan(sql)
sessionCatalog.createTempView("stream", logicalPlan, true)
val analyzer = new Analyzer(sessionCatalog, caseInsensitiveConf)
analyzer.checkAnalysis(logicalPlan)
```

主要内容

- Pandora
- 计算平台架构简介
- 流处理技术服务化需要考虑的两个问题
  - 用户的使用接口
  - 技术细节的屏蔽

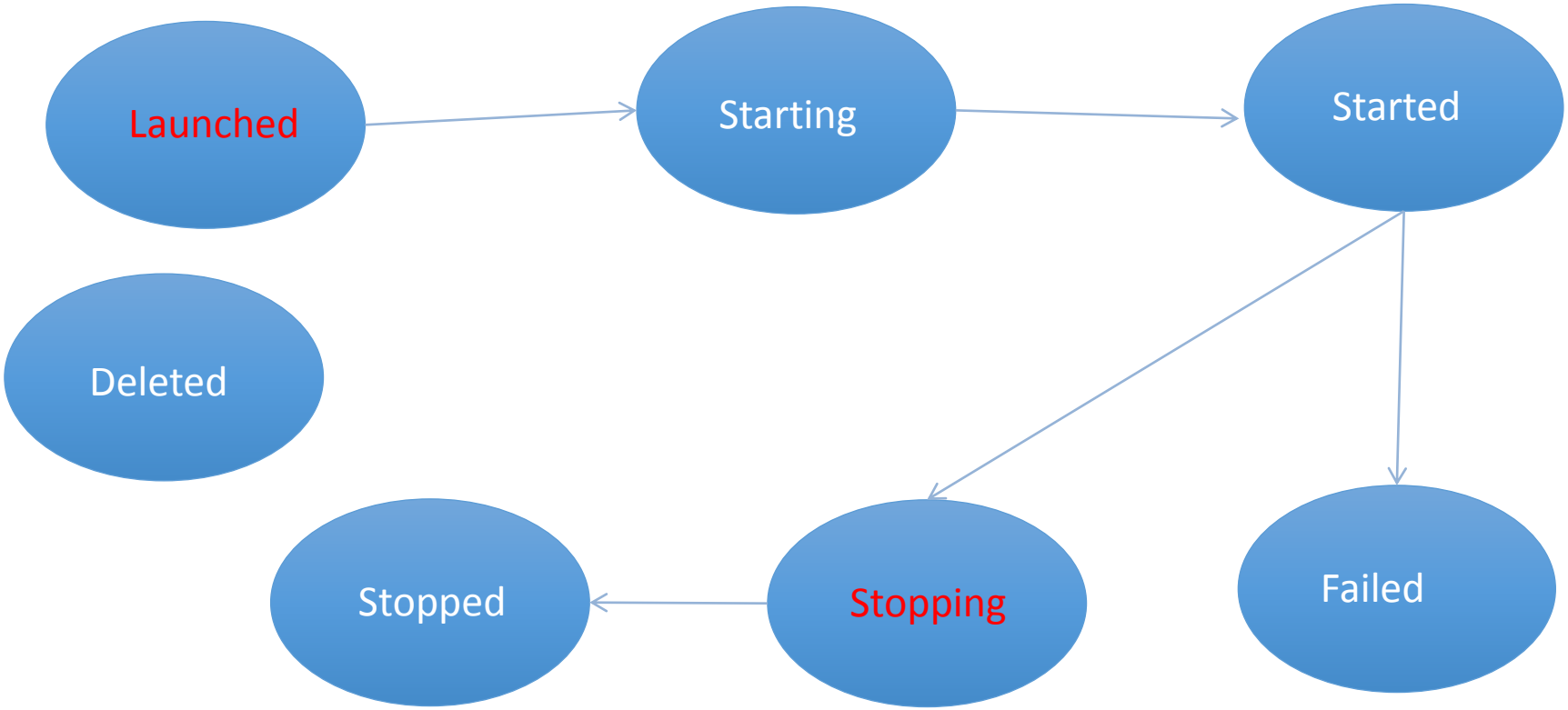
Spark Streaming 在生产上的常见问题

- 运行状态管理和监控
- 数据的丢失问题
- 数据消费的延迟问题
- 如何处理聚合操作
- 其他

Spark Streaming 在生产上的常见问题

- 运行状态管理和监控
- 数据的丢失问题
- 数据消费的延迟问题
- 如何处理聚合操作
- 其他

所有需要考虑的状态


Launched
Starting
Started
Deleted
Stopped
Stopping
Failed

Launch spark streaming application

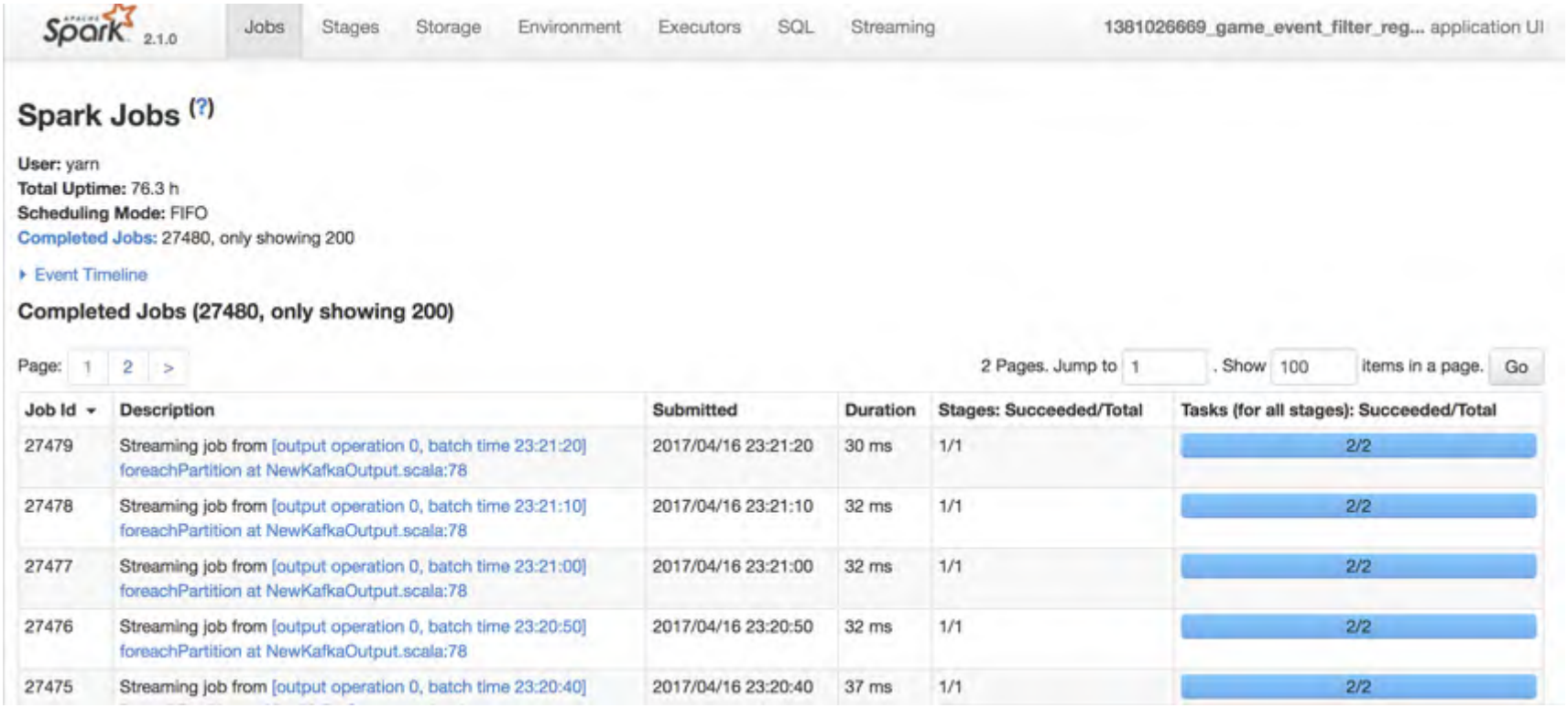
- 选择正确的启动的模式
- 选择正确的启动姿势(SparkLauncher)
- 选择正确的配置选项(spark.yarn.submit.waitAppCompletion)

如何优雅的停止一个正在运行的spark streaming app

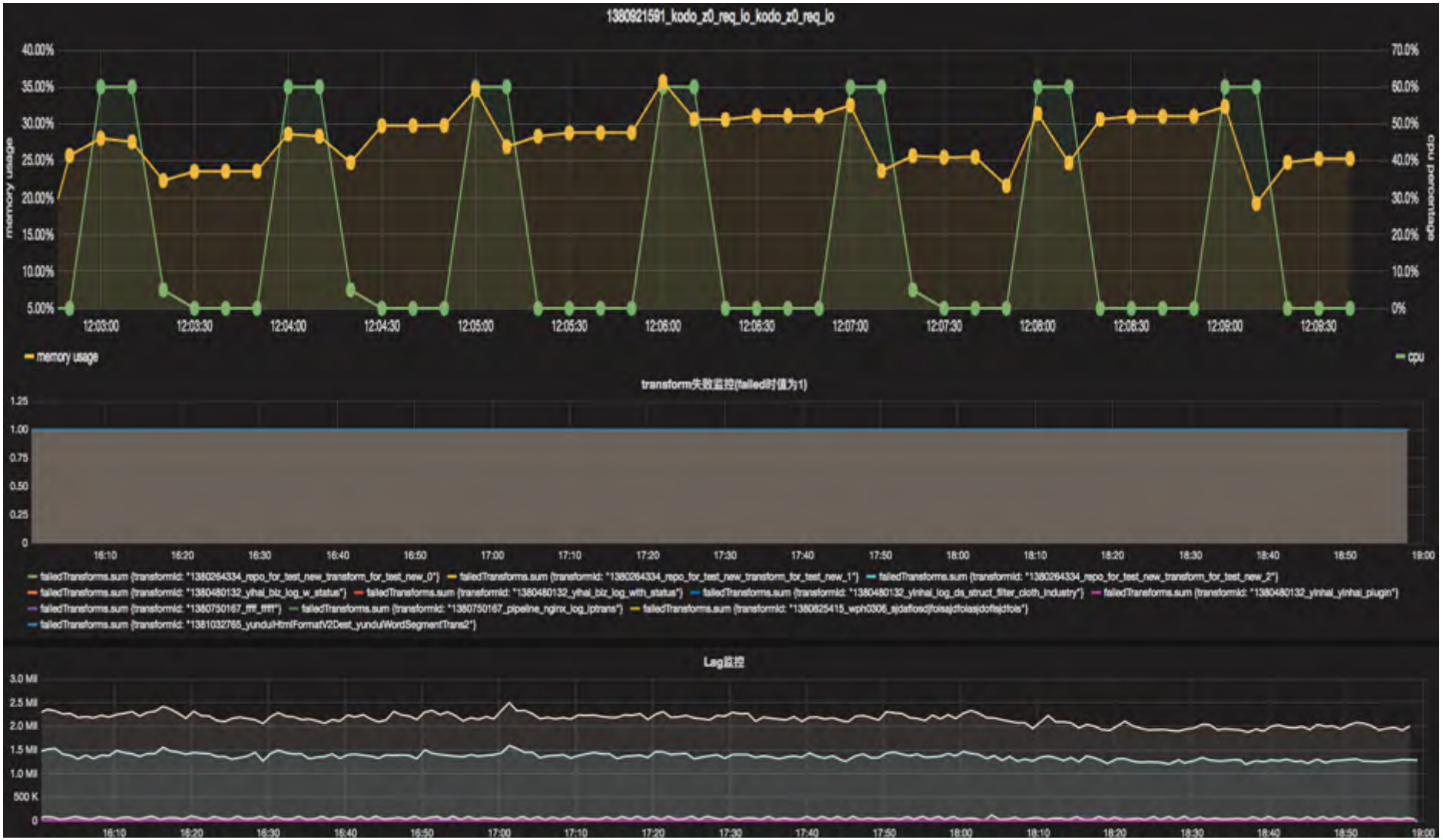
- 什么情况下需要停止？(更新资源或者计算逻辑)
- 如何停止？是否可以直接利用资源管理平台提供的方法？
- 如果让driver自己stop掉，如果通知driver？

监控

Spark UI 是否足够？



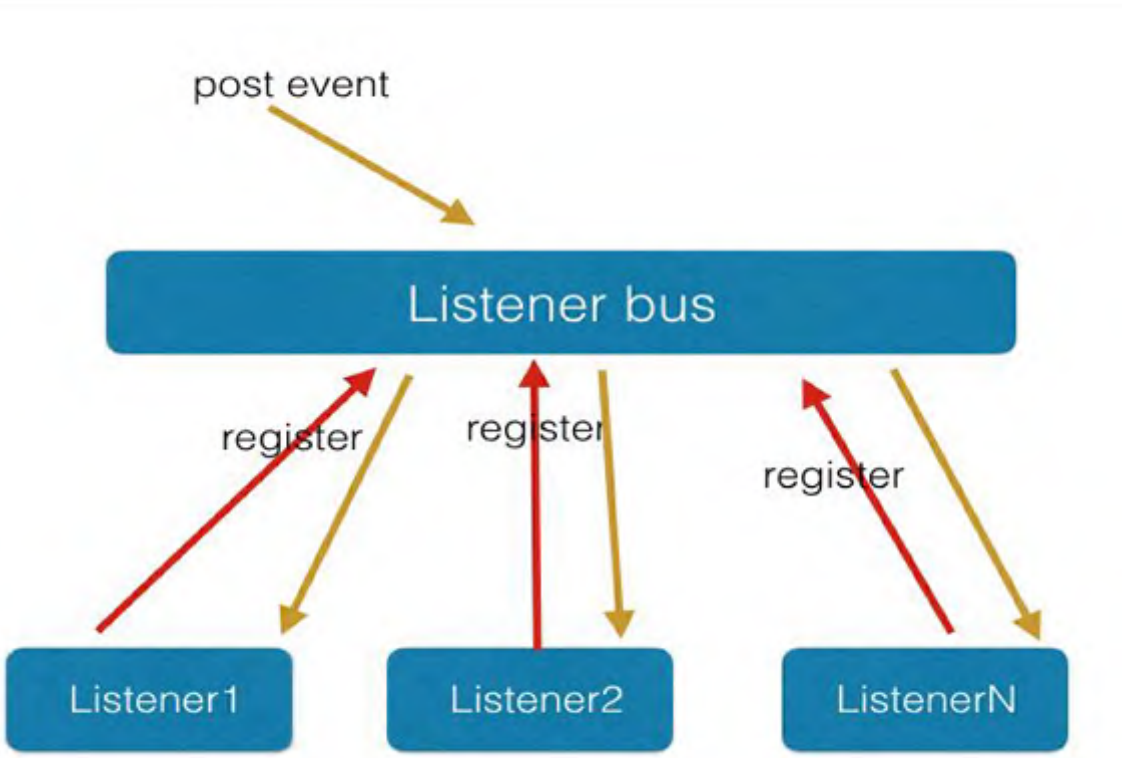
监控


1380921591_kodo_z0_req_lo_kodo_z0_req_lo
memory usage
cpu percentage
memory usage
cpu
transform失败监控(failed时值为1)
Lag监控

常用的获取监控信息的两种手段

- Spark listener(收集Spark component相关信息)
- Spark metrics system(收集low level jvm相关的信息)

Spark listener



Spark metrics system

在metrics信息中添加业务相关信息

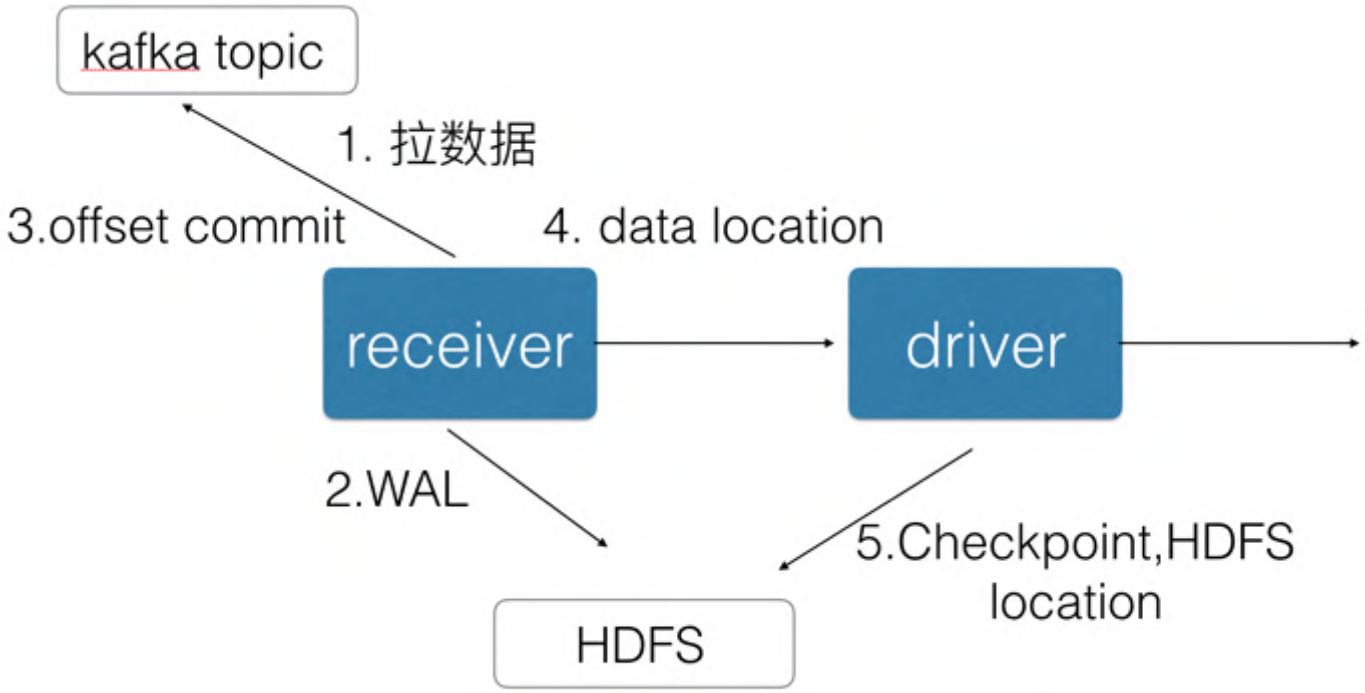
```
1380750167_pipeline_nginx_log_nginxParser3.application_1473412269671_0522.1.jvm.heap.usage 0.04 1494571568
1380750167_pipeline_nginx_log_nginxParser3.application_1473412269671_0522.1.jvm.heap.used 604130664 1494571568
1380750167_pipeline_nginx_log_nginxParser3.application_1473412269671_0522.1.jvm.non-heap.committed 116367360 1494571568
1380750167_pipeline_nginx_log_nginxParser3.application_1473412269671_0522.1.jvm.non-heap.init 2555904 1494571568
1380750167_pipeline_nginx_log_nginxParser3.application_1473412269671_0522.1.jvm.non-heap.max -1 1494571568
1380750167_pipeline_nginx_log_nginxParser3.application_1473412269671_0522.1.jvm.non-heap.usage -89299232.00 1494571568
1380750167_pipeline_nginx_log_nginxParser3.application_1473412269671_0522.1.jvm.non-heap.used 89299232 1494571568
```

```
select sum("used") from 380750167_pipeline_nginx_log_nginxParser3 where "type" = 'heap' group by time(10s),
```

Spark Streaming 在生产上的常见问题

- 运行状态管理和监控
- 数据的丢失问题
- 数据消费的延迟问题
- 如何处理聚合操作
- 其他

Receiver 模式


kafka topic
1. 拉数据
3.offset commit
4. data location
receiver
driver
2.WAL
5.Checkpoint,HDFS location
HDFS

Kafka Direct模式

- 开启checkpoint(不建议)
- 手动保存offset到zk,使用Spark Listener

Spark Streaming 在生产上的常见问题

- 运行状态管理和监控
- 数据的丢失问题
- 数据消费的延迟问题
- 如何处理聚合操作
- 其他

如何处理数据延迟

- 增加executor的个数(如果大于partition的个数) not work
- 增加partition的个数(如果使用kafka 0.8 client会丢失数据)

Spark Streaming 在生产上的常见问题

- 运行状态管理和监控
- 数据的丢失问题
- 数据消费的延迟问题
- 如何处理聚合操作
- 其他

Spark Streaming

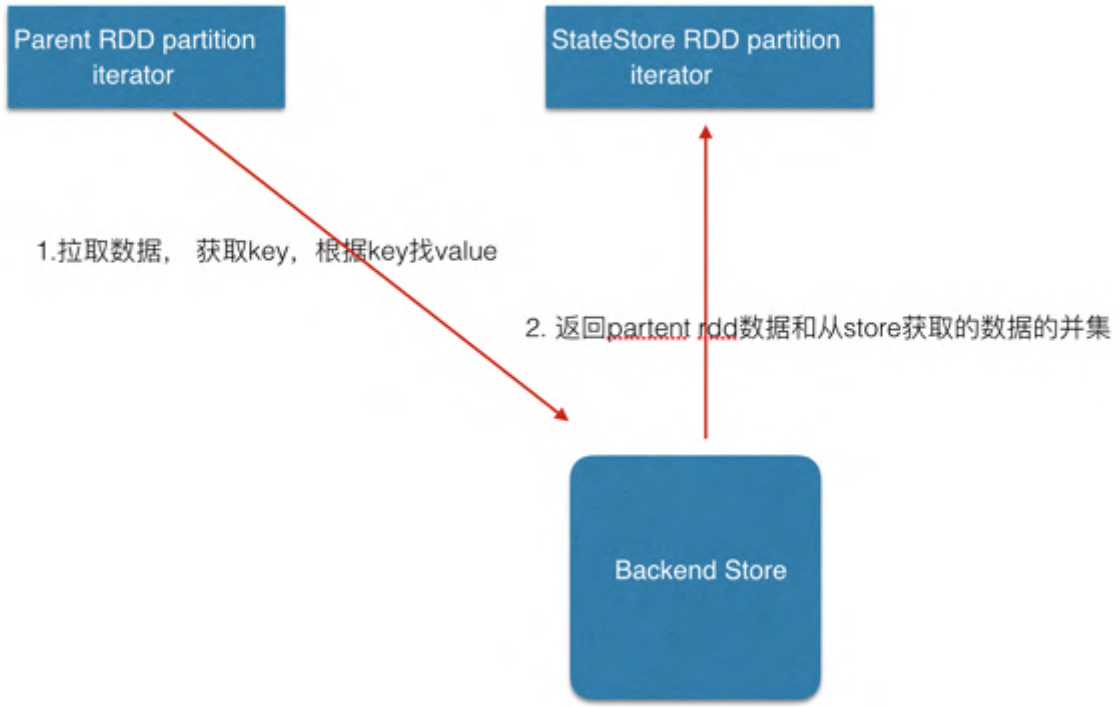
- updatestatebykey(扫描所有的key，效率会有问题)
- mapWithState(缺乏对event time的原生支持)
- 解决方法: 在每个Batch里面做预聚合操作，依靠下游db(tsdb等)进行最终的聚合操作。

Structured Streaming

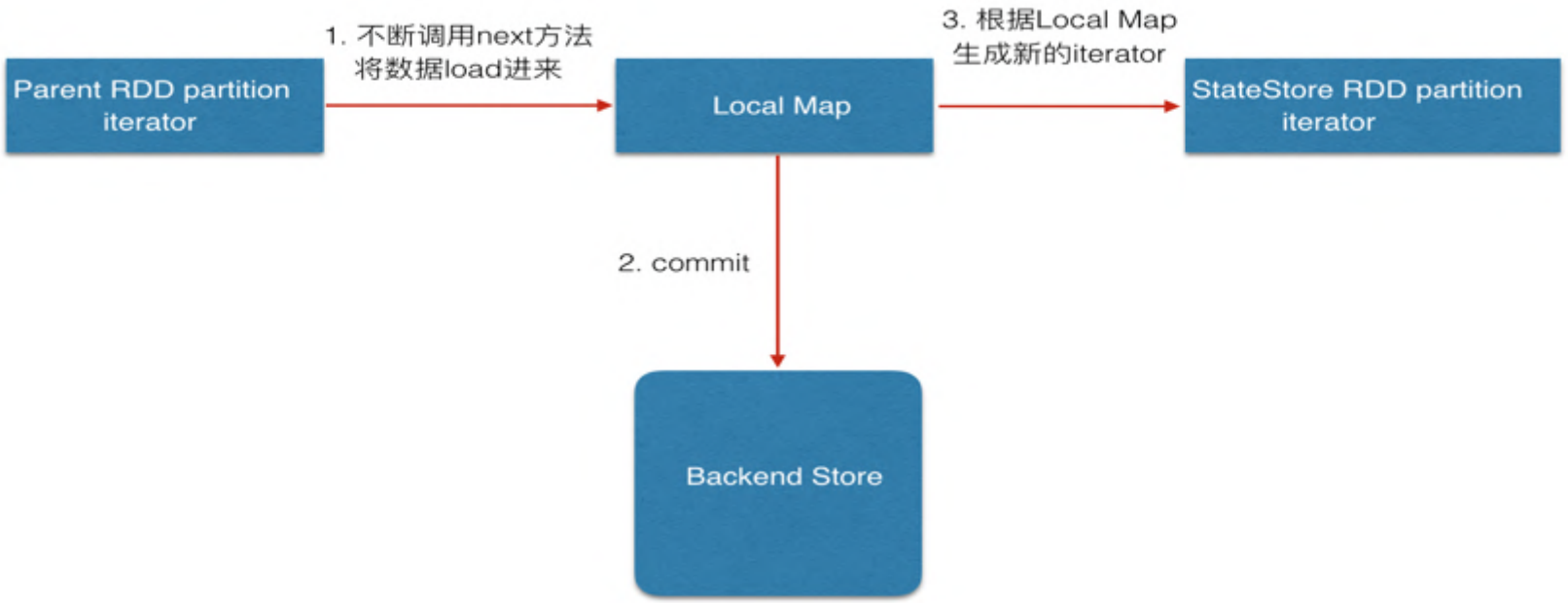
状态的读出与写入插入到了物理计划之中

```
== Physical Plan ==
*HashAggregate(keys=[window#17-T10000ms], functions=[count(1)])
+- StateStoreSave [window#17-T10000ms], OperatorStateId(<unknown>,0,0), Append, 0
   +- *HashAggregate(keys=[window#17-T10000ms], functions=[merge_count(1)])
      +- StateStoreRestore [window#17-T10000ms], OperatorStateId(<unknown>,0,0)
         +- *HashAggregate(keys=[window#17-T10000ms], functions=[merge_count(1)])
            +- Exchange hashpartitioning(window#17-T10000ms, 10)
               +- *HashAggregate(keys=[window#17-T10000ms], functions=[partial_count(1)])
```

Structured Streaming 读取状态

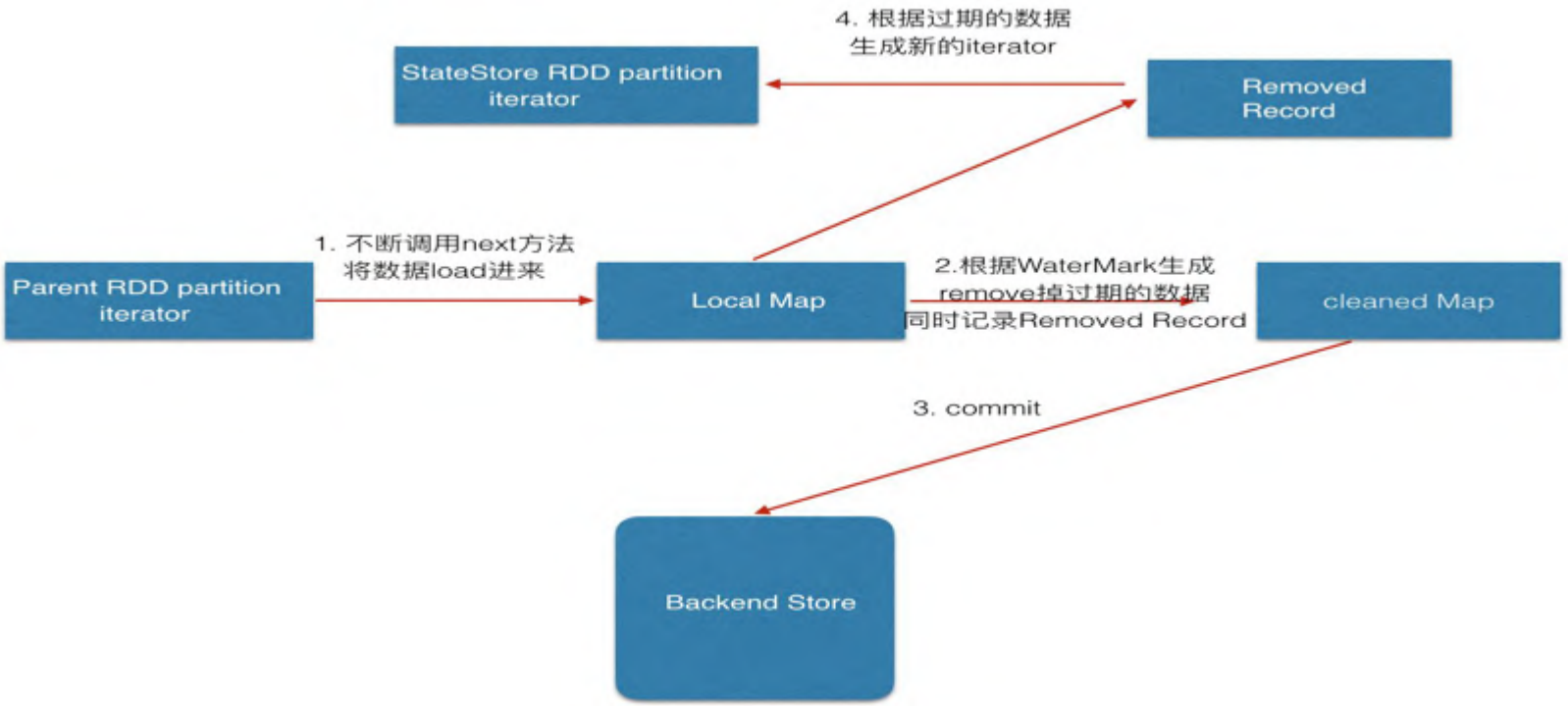


Structured Streaming 写状态(complete mode)

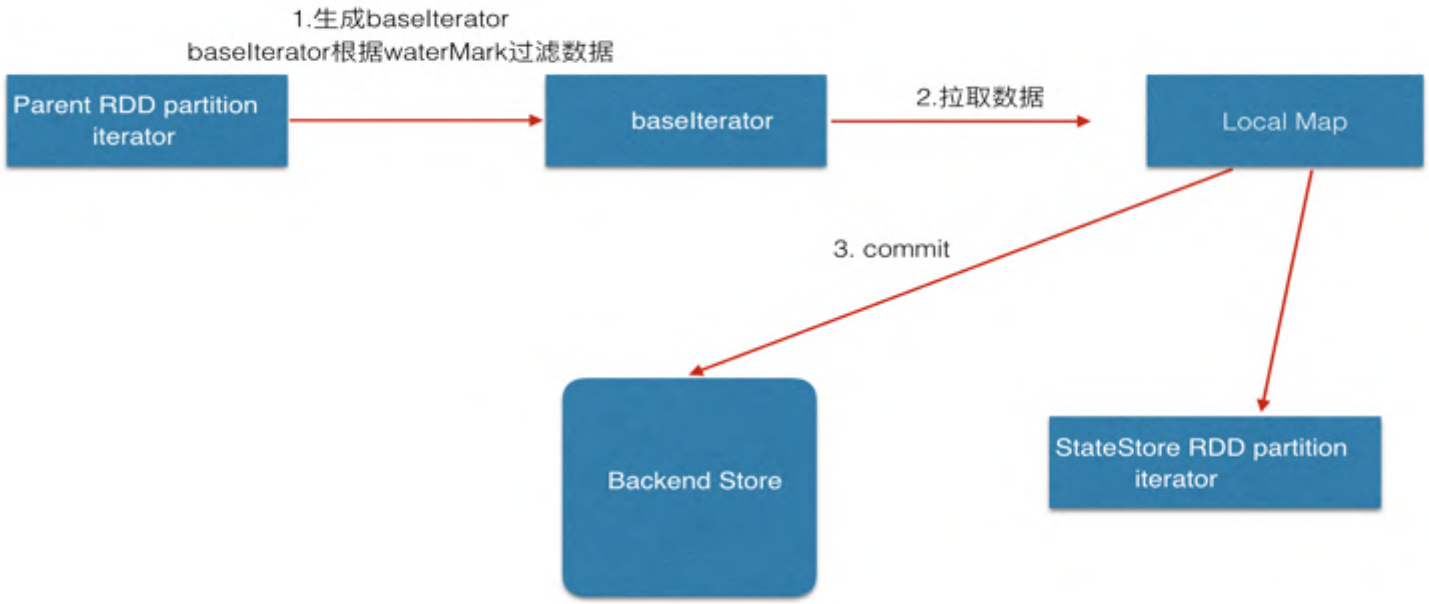


Structured Streaming 写状态(append mode)

Water Mark是必须的，否则不会有数据输出

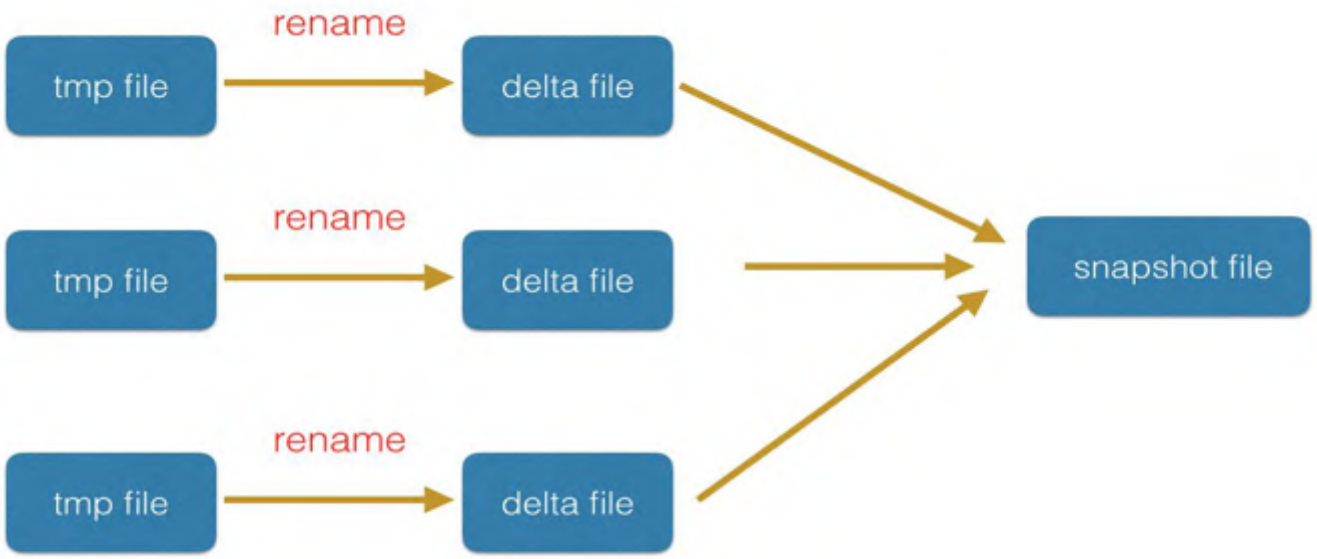


Structured Streaming 写状态(update mode)



关于Backend Store应该注意的

不同的文件系统rename(local, HDFS, Object store)语义并不一样。
对象存储耗时更长。另外可以参考Spark-19677



Spark Streaming 在生产上的常见问题

- 运行状态管理和监控
- 数据的丢失问题
- 数据消费的延迟问题
- 如何处理聚合操作
- 其他

其他各种问题

- Kafka 0.8 clinent是否可以消费Kafka 0.9 broker server
- 各种系统交互(HDFS, Kafka broker)如何集成测试
- 如何给用户展示一个易于理解的日志
- ……

总结

- 提供简介灵活的用户使用范式
- 降低用户的运维成本
- 让用户更多的专注与业务

谢谢！