Akka Stream实践

OneAPM-陈友国

Akka Stream 在我司应用场景

1.时序数据聚合

2.数据采集查询中间件

3.报警检测

重点谈谈时序数据聚合

面临的问题

1.源源不断的数据

2.对用户的查询友好

Reactive Stream！

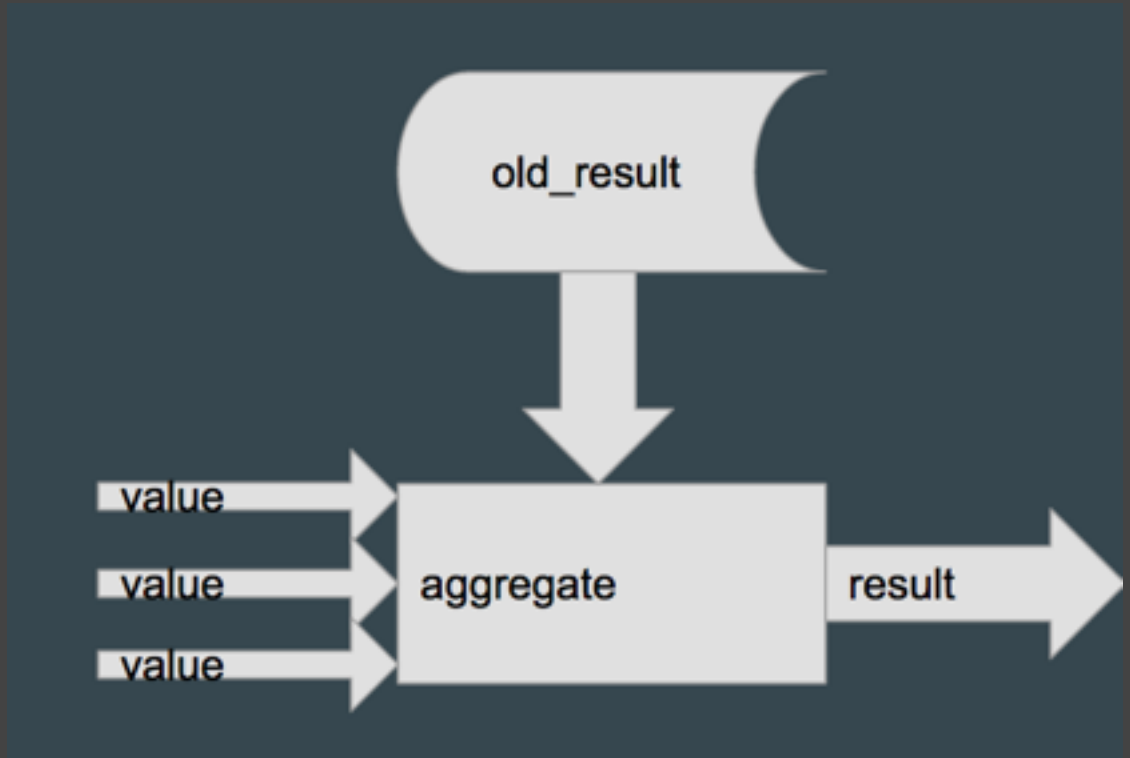
选择Akka Stream

1.实现Reactive Streams宣言

2. 采用Akka实现，容易集成Akka

3.BackPressure

聚合



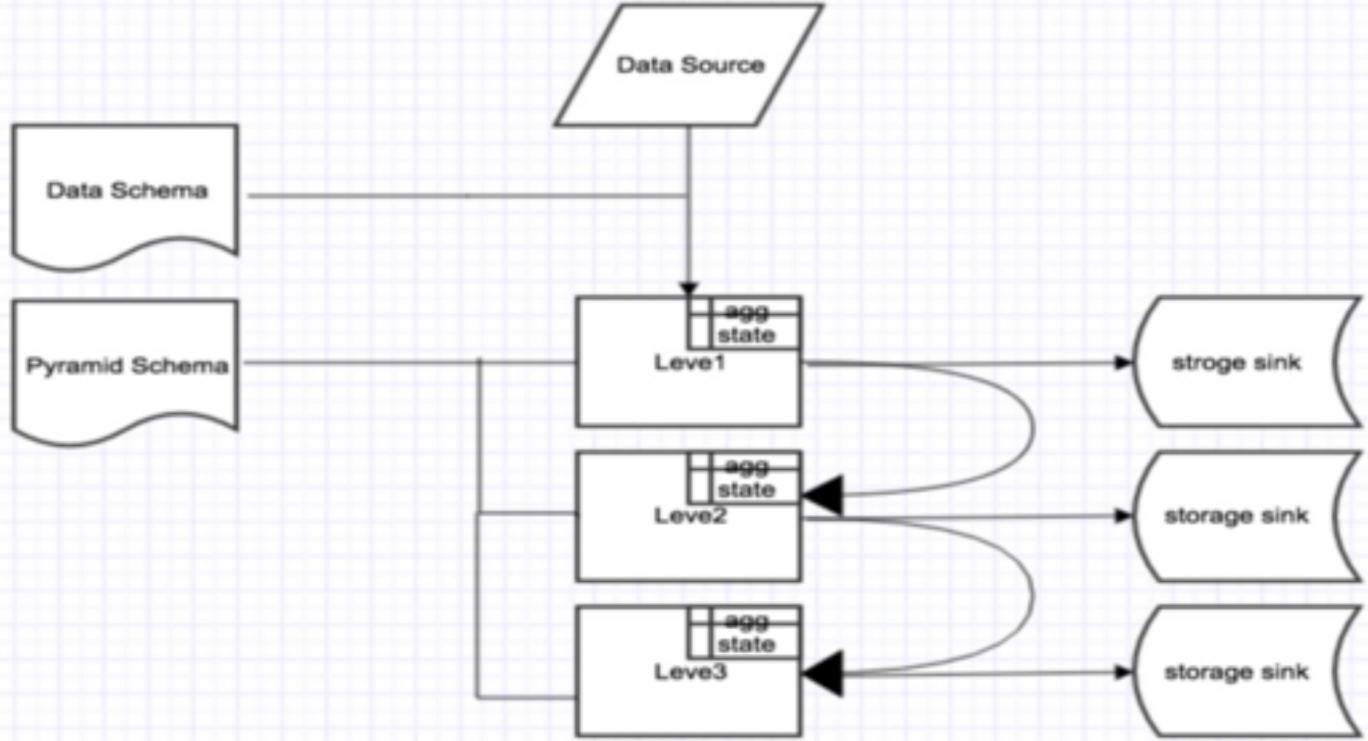
时序数据聚合

举个🌰统计方法被调用次数，指标有：

1分钟内，某个方法被调用次数

5分钟内，某个方法被调用次数

1小时内，某个方法被调用次数


Data Source
Data Schema
Pyramid Schema
agg
state
Leve1
agg
state
Leve2
agg
state
Leve3
stroge sink
storage sink
storage sink

局部状态采用HashMap：处理速度快 －20万/s

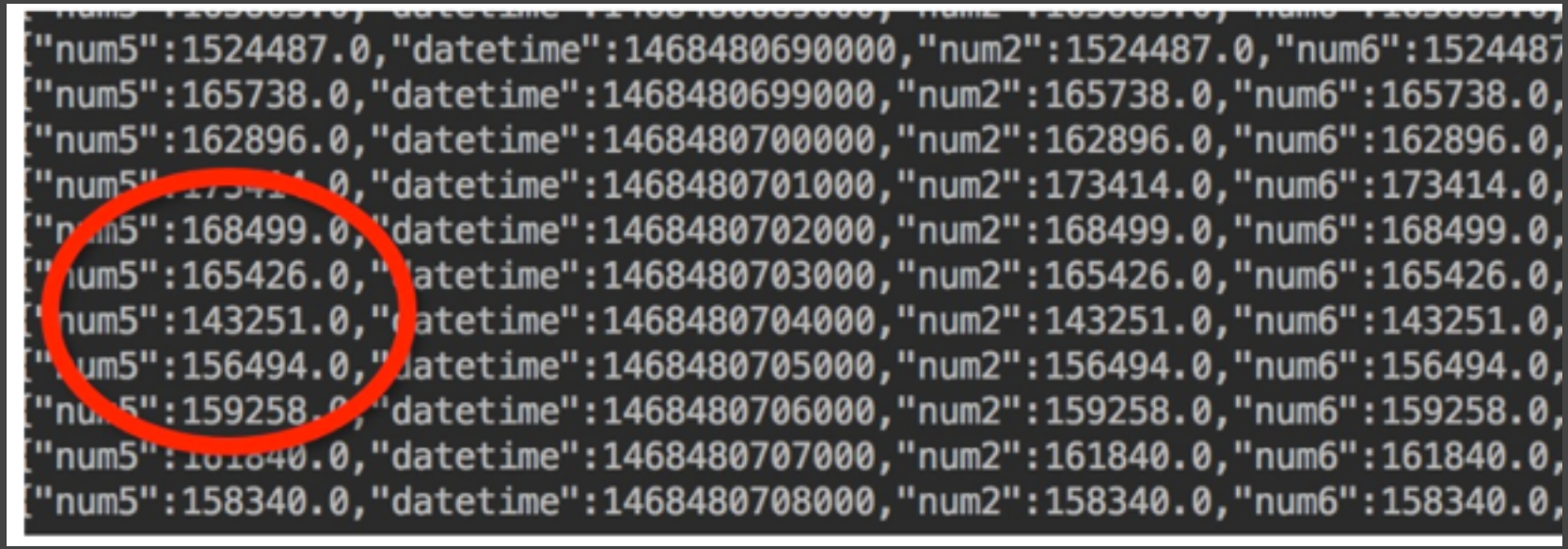
聚合速度不稳定

原因：创建临时小对象多，JVM频繁GC

解决方案：

采用ByteBuffer表示数据

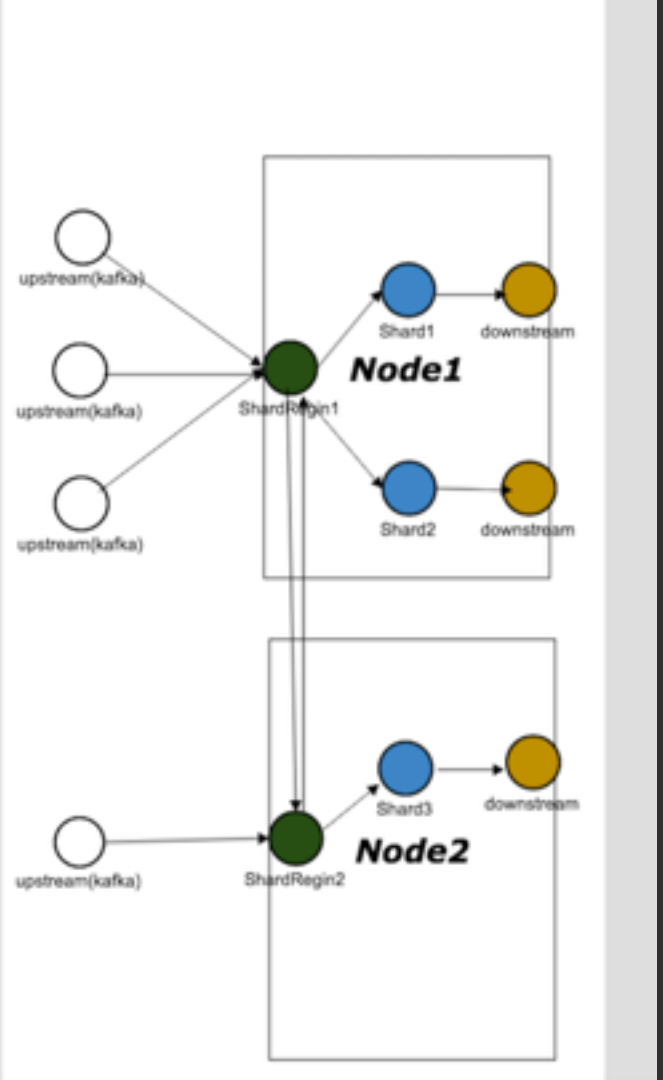
聚合中间状态采用堆外存储



聚合速度有所下降但是稳定 －15万/s

Scalability?

尝试集成Akka Cluster



目的

1.利用Akka Cluster对数据动态分区

2.打算利用Akka Cluster水平伸缩提高聚合能力

理想很美好，现实很骨感

问题

1.节点间失去了BackPressure特性

2.当集群发生rebalance时，Shard的缓冲大小不可控，会出现数据丢失。

当有新node加入到cluster时，cluster中的shard会发生rebalance（重新决定shard分布在某个node上）
在rebalance过程中，是否需要暂停

优势

1.提供基本的算子，业务逻辑实现简洁
2.提供自定义算子，编程足够灵活
3.BackPressure，不容易导致机器过载

劣势

1.跨机器可伸缩性不足
2.有状态业务，状态存储没有对应的API，需要自己处理

小结

Akka Stream 暂时是一套流式编程库，而不是流式系统！

代码中的经验教训

GraphStage之殇

内部处理逻辑状态琐碎

考虑上有有没有数据

上游是否关闭

下游是不是在请求拉取数据

下游是否关闭

是不是该向上游传递下游的压力

…

解决方案：测试驱动开发

GraphStage内置Scheduler

测试依赖时间，导致测试速度慢，测试不稳定

解决方案：

1.合理预估代码执行时间

2.不依赖内置的TimerGraphStage，自定义调度接口，暴露出供测试

避免GraphStage之间状态共享

类似

```
val storage = …


source
.via(UpStreamGraphStage(storage))
.via(DownStreamGraphStage(storage))
.toMat(CustomSink)
(Keep.right).run()
```

容易引起多线程问题

小结

优先选择自带算子，然后选择采用自定义GraphStage

结束!

问题?