

Apache Kafka在京东的演进和实践

乔超 · 京东 / 实时数据总线负责人

Apache Kafka x Apache Flink·北京— 2019年05月12日

CONTENT

目录 >>

01 / 京东数据总线介绍

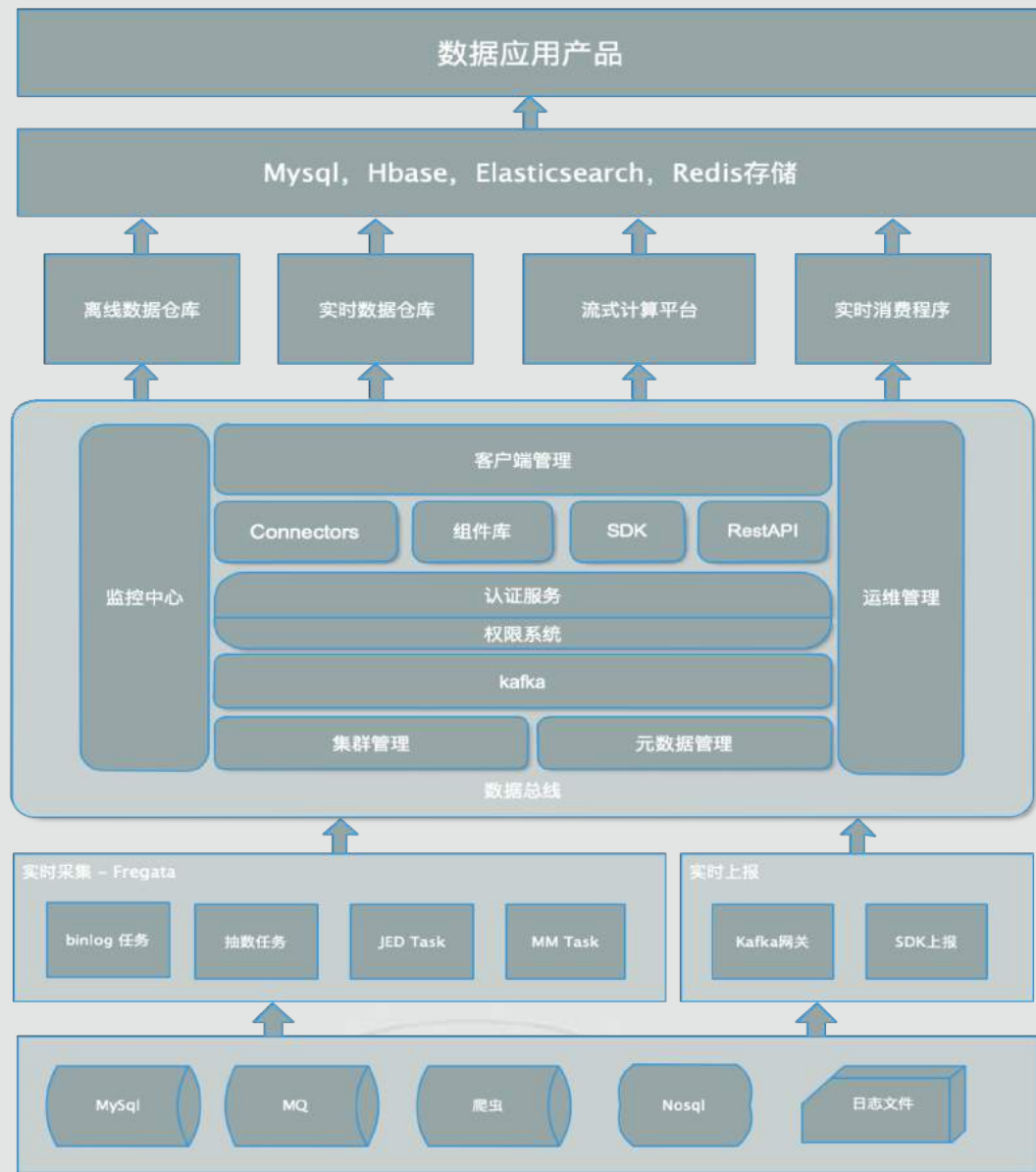
02 / Kafka在京东演进历程

03 / kafka实践与探索

01

数据总线介绍

京东kafka



京东数据总线

京东实时数据平台的中重要基础组件数据总线，基于kafka构建，提供实时数据存储服务(数据库日志，流量日志等)，大数据量系统数据缓冲以及实时数据分析数据源,支持对接公司离线和实时数据仓库，分析系统。



全流程管控产品化

使用流程，客户端管理，运行监控，位点管理，安全认证权限管理



高性能高可用

读写分离，跨机房复制，解决跨机房容灾问题



安全认证细粒度权限

细化权限粒度，针对不同的资源进行权限控制
进行服务端和客户端kerberos认证，更加安全稳定

数据总线作用

解耦&&异步通信

介于不同系统做消息传输，只需要支持处理消息来进行交互，更便于独立扩展

顺序保证

数据顺序性会适用一定场景，保证一个分区内的消息的有序性



流式数据源

充当最齐全的流式数据的数据源，可以找到京东所有的流式数据包括binlog，日志，监控等



峰值处理

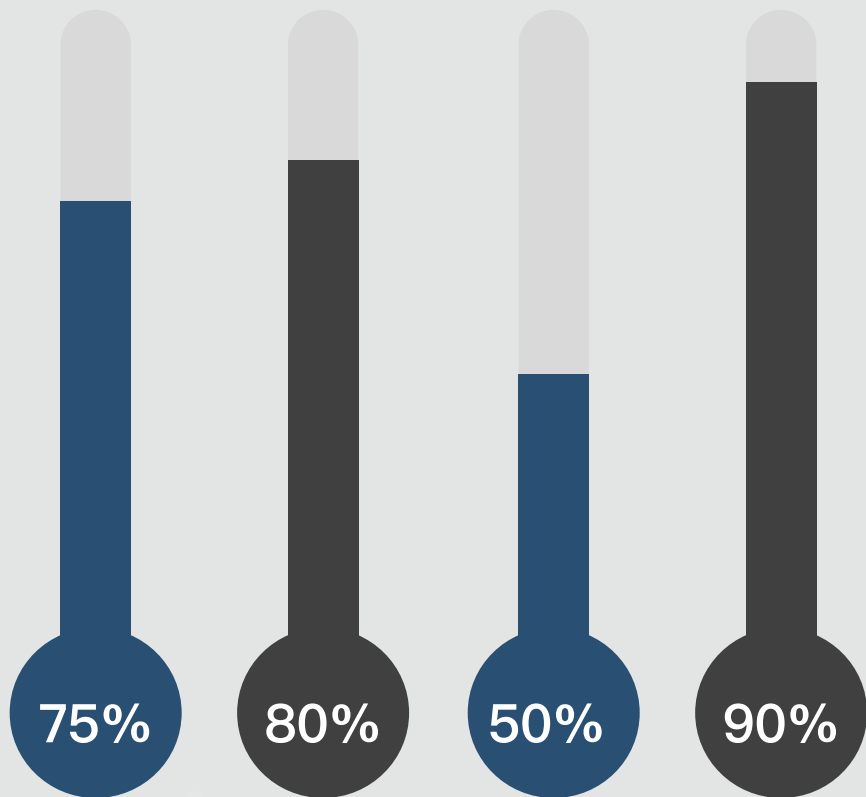
随机出现流量洪峰，总线可提供缓冲作用，进行削峰



Kafka In JD时间轴简介



业务规模



kafka状况

线上56个集群, broker1530
Topic15699, 分区460301



用户状况

覆盖公司各级部门, 申请用户数量3W+
日常请求峰值: 生产1.42亿/秒, 消费1.69亿/秒



运行状况

消息生产行数1.7万亿/天
流入量: 775TB/天, 流出量: 2868TB/天



2018年双11运行状况

消息生产行数2.78万亿/天
流入量: 1.11PB/天, 流出量: 4.23PB/天

02

Kafka在京东演进历程

稳定完善过程

早期Kafka-0.8.X版本

集群稳定运行

千兆环境流量瓶颈凸显，无限流措施
无法进行数据灾备
消费和生产相互影响
消费者不断增多，解决消费者横向扩展问题

01



周边产品完善

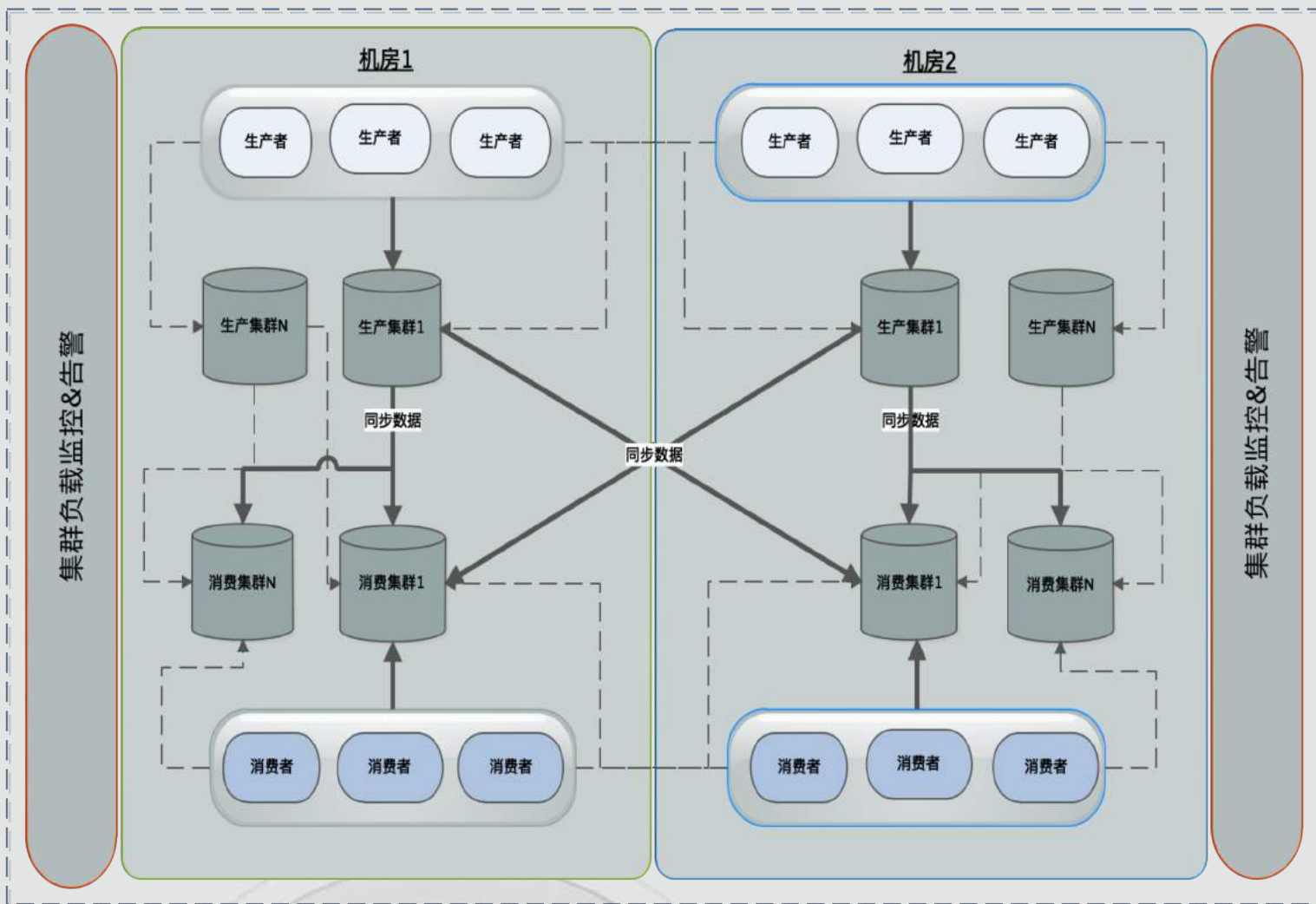
监控完善(服务端&客户端)
周边工具(位点，admin，负载)
集群客户端产品管理

02

03

权限问题

指定用户使用固定的topic
屏蔽集群信息改为鉴权信息



跨机房灾备

减少跨机房的网络传输，尽量选择本机房进行生产消费
跨机房的数据灾备
分散数据写入和消费的压力

读写分离

保证数据写入数据的稳定性

生产消费负载扩展

解决单集群消费扩展问题
生产和消费的灾备切换
数据优先级处理
生产消费自动化切换方案

版本升级 — kafka0.9+

集群内部的安全

不明身份的broker对集群的访问
不明身份的broker不应该成为集群一员
防止集群在ZK上的metadata被篡改
数据存储的安全(删除数据问题)

01

升级
kafka0.9+

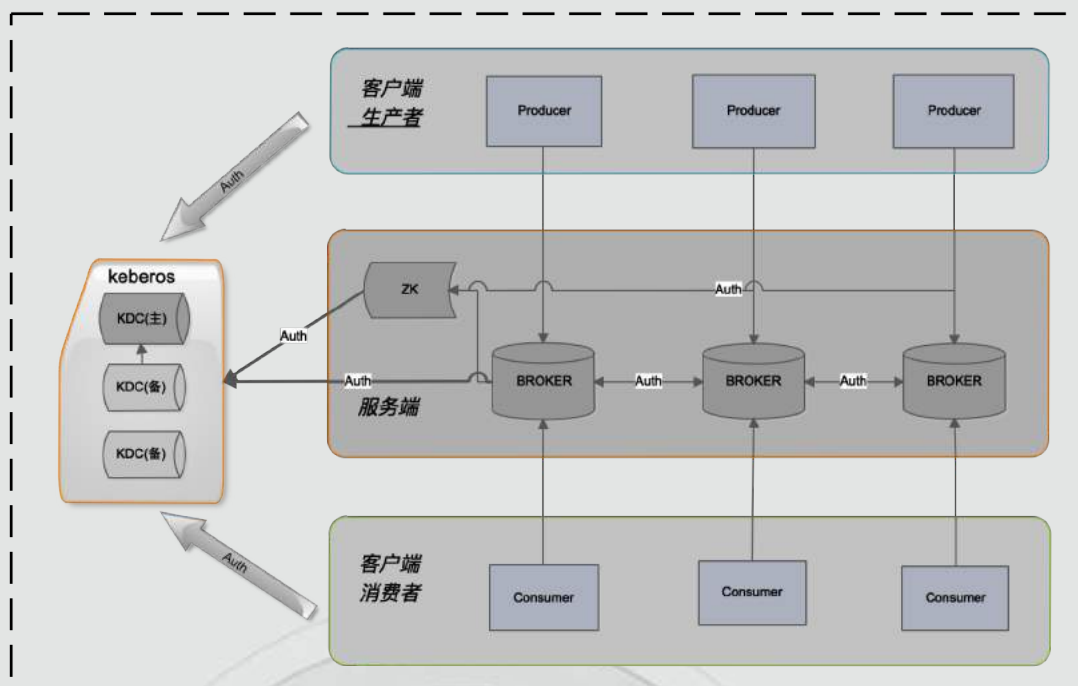
集群与外部客户端安全通讯

执行客户端的用户身份认证
C-S传递的消息不能被窃听、篡改
client只能访问被授权访问的资源

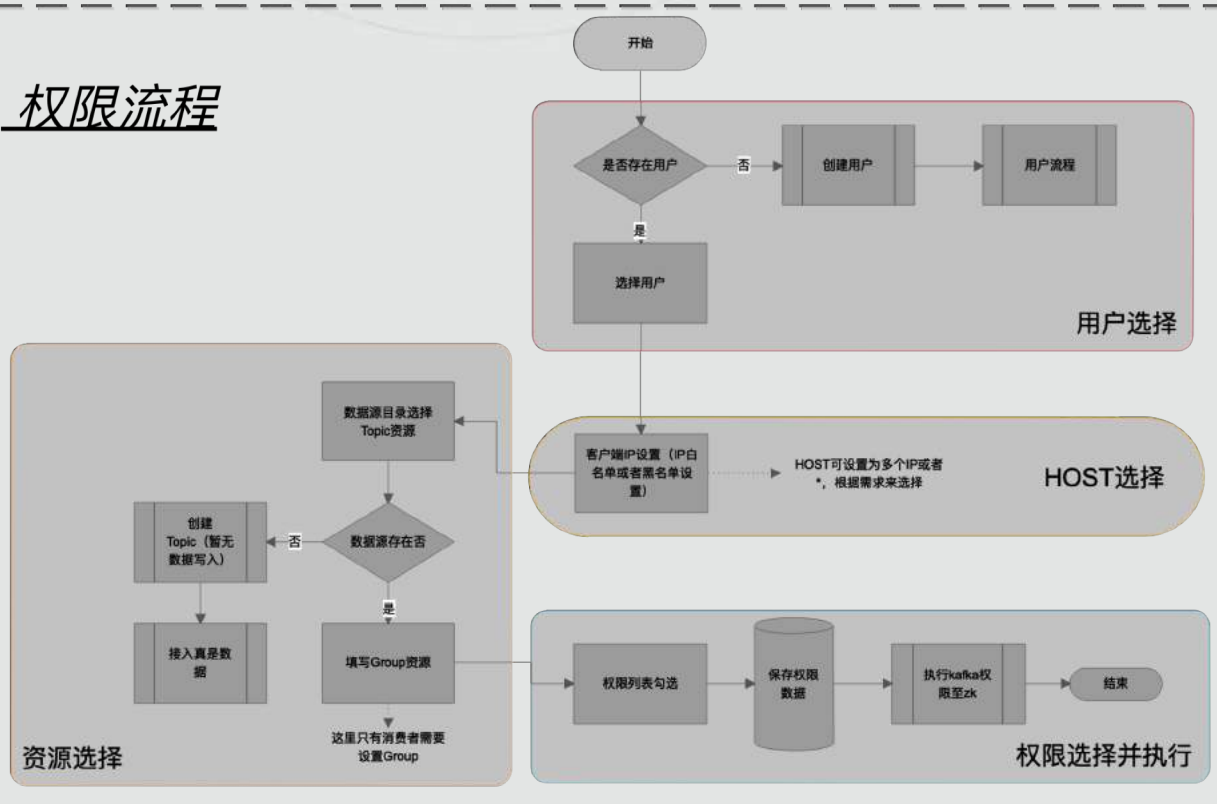
02

安全认证改进

各个环节加入认证；
方便用户使用去keytab
去jaas配置文件
Kerberos集群主备
增加服务端域名服务认证



权限流程



权限控制

资源分类：topic，group，集群等
机器黑白名单
流程绑定Kerberos用户
对接产品权限流程实现远程

周边组件完善



服务端

- 集群管理
- 运维管理
- 权限管理
- 用户&认证管理



客户端

- 客户端管理
- 元数据管理
- 位点管理
- 运行信息
- 限速管理



监控报警

- 服务端监控
- 端对端监控
- 运行指标监控
- 挤压监控
- 实时大屏



辅助工具

- JDQ-SDK
- AdminClient服务
- Util工具包
- 消息查询工具
- 样本提取过滤工具
- JA运营报表

03

Kafka实践与探索

结合实际场景改进

性能测试环境:

机器：32c+128GB内存 + SSD(Raid10)

万兆环境

测试消息：1kb + lz4压缩

Kafka版本	Kafka协议	线上版本	优化版本
0.10.1.X	V1	700w/s	1400w/s

版本：对应版本0.11之前版本包含0.8

环境：CPU 78%，带宽200MB/s

问题：存在Crc32热点问题，CPU80%消耗

在这里

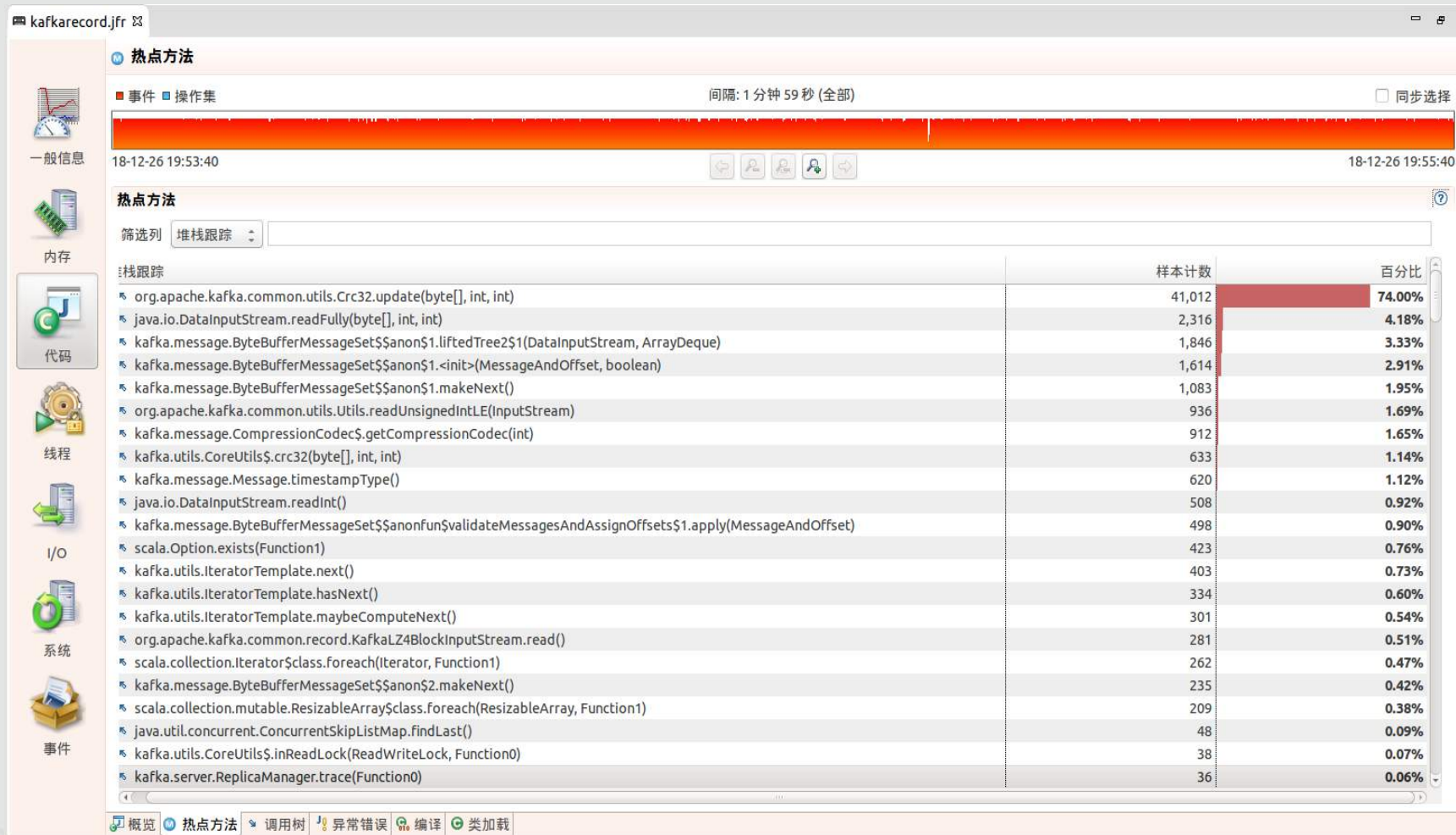
消除热点将性能翻倍：

对比Crc32，对kafka本身的Crc32，以及

hadoop实现的Crc32c，jdk8的Crc32，以及

jdk9以上的Crc32c进行了JMH性能测试，在

1kb左右的消息里jdk8的crc32性能最优





Kafka性能 - V2协议



Apache Flink

性能测试环境:

机器：32c+128GB内存 + SSD(Raid10)

万兆环境

测试消息：1kb + lz4压缩

Kafka版本	Kafka协议	线上版本	优化版本
2.1.X	V2	2400w/s	3500w/s

版本：对应版本0.11之后版本

环境：CPU 96%，带宽600MB/s

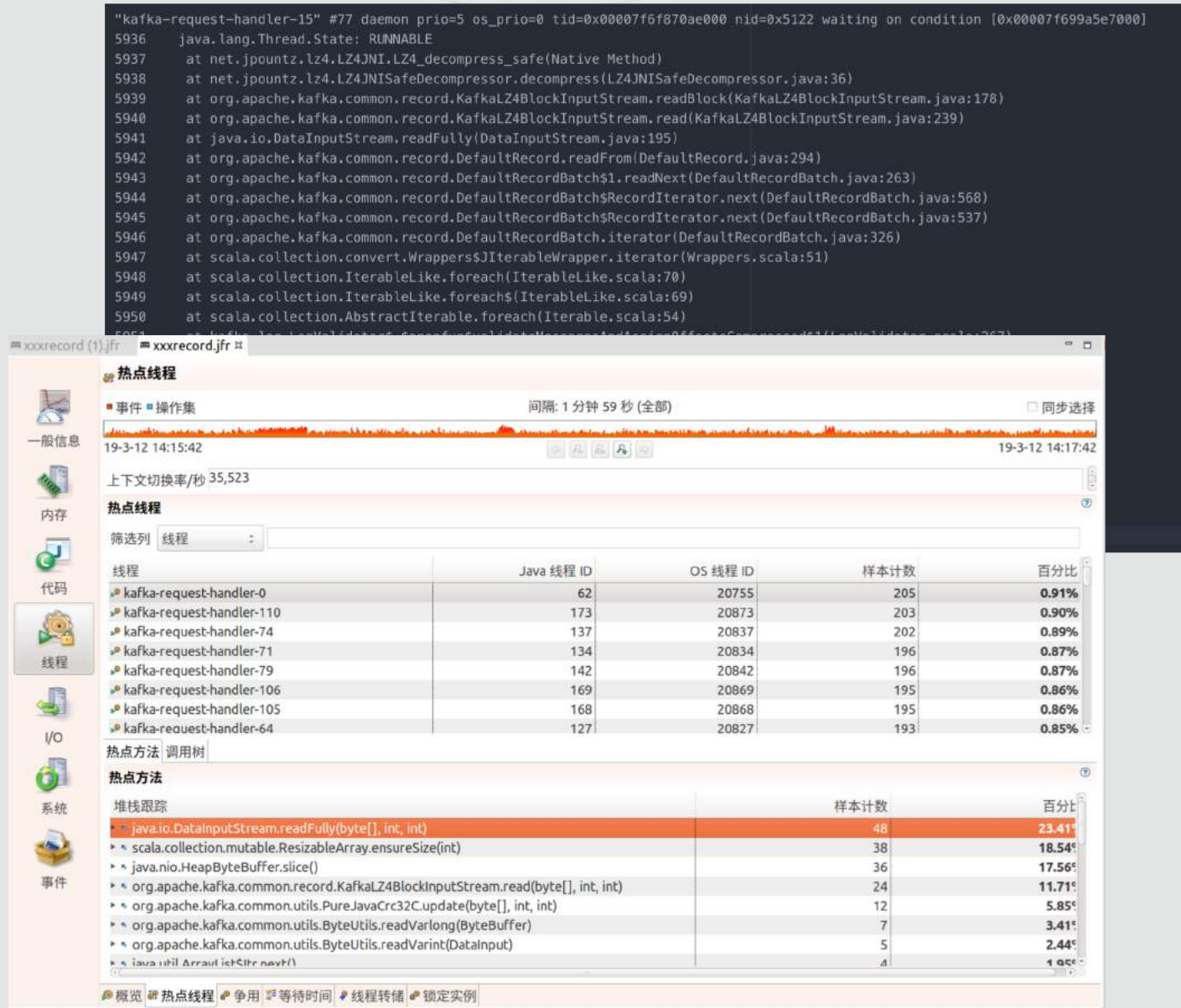
问题：CPU瓶颈明显

分析CPU消耗点进行针对性优化：

性能在V2的批代替单条的协议改造之后大幅提升，同时没有热点瓶颈（推荐大家在选择kafka版本的时候一定要选V2协议的版本）

验证CPU的消耗点

商讨改进方案



服务端解压缩数据原因：伴随着Record遍历（解压缩也会在这里进行）

拿到具体的数据做了一些逻辑处理：

1. 验证key 和 compact 配置
2. 验证消息的ts
3. 是否存在inPlaceAssignment

去掉遍历做了一轮测试：

性能达到3500w/s，CPU维持在15%上下，带宽1100MB/s
硬件瓶颈

CPU的大部分的消耗还是在遍历里面

JD优化方案：

解压缩处理暂时不能避免，但是可以在遍历的逻辑里面做一些细节优化，处理阶段每次在数据流之外有新申请了一块byteBuffer的内存，由于每秒要处理将近2kw，也就会申请2kw次，对GC的压力很大，但是数据流里面会有解压缩之后的数据，而且我们在这段逻辑里也没有对value和key的内容做处理，所以我们采用数据流里拿出key和value之外的数据，减少多余的内存申请，达到一层优化

优化效果：

性能达到3500w/s，CPU维持在58%，带宽1100MB/s硬件瓶颈
GC层面上缓解明显



全链路域名化改造



对接Kubernetes

混合部署方案

与流式计算机器共同使用物理机
CPU，内存和磁盘分摊给流式计算和kafka
完善周边运维工具：监控报警，驱逐脚本
保证kafka服务的稳定（已上线）

性能对比：按照资源分配性能减少

优势：独立用户不受k8影响

劣势：运维成本增加

&&

Kafka on K8S方案

Kafka connect on K8S稳定运行
Zookeeper&Kafka Operator(未上线)
LocalPV && configMap

性能对比：与物理机相差不大


优势：简化部署成本

劣势：K8S的不稳定影响kafka服务

THANKS

Flink China社区大群



 扫一扫群二维码，立刻加入该群。