MQ对比结果

RocketMQ vs Kafka

Features	Kafka	RocketMQ
实现语言	Scala	Java
客户端SDK	Java，Scala，etc	Java，C++, go
协议与规范	Pull，支持TCP	Pull，支持TCP、JMS、OpenMessaging
消息顺序	单个Paritition中能够保证	单个queue可以保证严格消息顺序
定时消息	不支持	支持
批量消息处理	支持异步刷盘	支持同步、异步刷盘，同步可保证不丢消息
广播消息	不支持	支持
消息过滤	可以使用Kafka Stream实现	支持
服务端触发消息重试	不支持（只能通过添加retry_topic等自己模拟实现）	支持
消息存储	高性能文件存储	高性能文件存储
消息回溯	通过offset	通过offset和timestamp两种
消息重复消费	修改offeset来实现	支持按照时间重新消费
消息优先级	不支持	不支持
高可用	需要zookeeper	Master-slave，不需要其他工具
管理工具	yahoo kafka management/命令行	丰富的web管理工具和命令行
分布式事务	不支持	最新版本支持
消息查询	不支持	支持
消费并行度	依赖Topic的分区数	顺序消费与Kafka一致，乱序取决于Consumer线程数
集群选主	从ISR中自动选举leader	不支持自动选主，需要指定
主从切换	自动切换，自动选leader	不支持自动切换，master失效后，consumer感知到后从slave消费，异步复制会出现信息丢失
消息写入性能	百万条/s (每条10个字节)	单机单broker约7w/s，单机3个broker约12w/s
队列增加时性能稳定性	不稳定且明显下降	稳定
单机支持队列数	超过64个队列/分区，load会飙高	单机最多支持5万个队列，Load不会发生明显变化

Features	Kafka	RocketMQ
堆积能力	高，每个分区由一个或多个segment log文件	高，所有消息存储在同一个commit log中
消息投递实时性	具体有Consumer轮询间隔决定	支持pull，push两种模式，延时通常毫秒级
批量发送	支持，默认缓存压缩后批量发送	不支持
消息清理	指定文件保存时间，过期删除	指定文件保存时间，过期删除
系统维护	Scala语言开发，成本高	Java语言开发，成本低
部署依赖	zookeeper	nameserver

RocketMQ 生态

项目	详情
RocketMQ-Console	使用spring-boot实现的RocketMQ控制台
RocketMQ-JMS	RocketMQ JMS实现
RocketMQ-Flume	Flume RocketMQ source and sink implementation
RocketMQ-Flink	Integration of Apache Flink and Apache RocketMQ.
RocketMQ-Spark	Spark和RocketMQ集成
RocketMQ-Docker	Dockerfile以及bash脚本
RocketMQ-MySQL	Data replicator between MySQL and other systems
RocketMQ-CPP	CPP client
RocketMQ-Druid	Rocket与Druid的集成
RocketMQ-Ignite	RocketMQ与Ignite集成
RocketMQ-Storm	Storm/Trident integration for RocketMQ

Kafka生态

项目	详情
Kafka Connect	连接kafka与外部系统的媒介
Distributions & Packaging	包括 Confluent Platform ， Cloudera Kafka source， Hortonworks Kafka， Stratio Kafka ， IBM Message Hub 等
Stream Processing	原生的Kafka Streams；与Storm，Samza，SparkStreaming，Flink，IBM Streams, Spring Cloud Stream， Apache Apex等也有良好的集成工具
Hadoop Integration	包括Congfluent HDFS connector，Camus，Kafka Hadoop Loader，Flume，KaBoom等
Database Integration	包括Confluent JDBC Connector, Oracle Golden Gate Connector
Search and Query	与ElasticSearch，Presto，Hive等有良好的集成
Management Consoles	管理工具KafkaManager，命令行管理工具Kafkat，Web工具Kafka Web console等
Logging	支持丰富的日志工具
Metrics	丰富的指标监控工具

双机房容灾

双机房容灾问题，是指在两个机房中分别部署两套MQ服务之后，在其中一个机房出现故障后MQ消费切换的问题。事实上，目前我们使用的双机房是在一个局域网下，机房之间的延迟很低，所以可以认为机房间的网络是完全互通，在正常情况下，MQ的消费也可以跨机房进行，因此，不存在机房故障导致生产环境崩溃或消息丢失的问题。

总结

> Kafka在高吞吐、低延迟、高可用、集群热扩展等具有很好的表现；且producer端提供缓存、压缩能力，可以提升效率，生态较为完善，在大数据处理方面有大量的配套设施。

> RocketMQ在高吞吐、低延迟、高可用上都有非常好的表现；API，系统设计都更加适合业务处理场景，如支持多种消费方式，支持消息过滤，支持事务，支持消息的查询，定时消费等。能很好的满足未来的业务场景。

对比来看，虽然Kafka对上述业务场景特性支持的不够好，但是有可行的补充方案。RokectMQ基础性能不如Kafka高，但是目前来看应该足以满足我们的需求。

RocketMQ存在的不足是其目前最成熟的Client是Java实现的，有阿里提供的C++版本。缺乏其他语言的支持，这是后续使用的一个主要的可能出现的阻碍。

最终结论

使用RocketMQ