# Kafka与RabbitMQ

- 讲师：pansir

# 功能维度

## 优先级队列

普通队列：先进先出

优先级队列：优先级高点消息拥有优先被消费的特权。如果消费速度大于生产速度，broker中没有消息堆积，那么消息优先级就失去意义了

## 延迟队列

消息发送后，希望等待特定的时间后，消费者才能进行消费。例如：自动取消30分钟未支付的订单

基于消息的延迟：为每条消息设置不同的延迟时间，所以消息需要进行延迟排序，从而影响性能

基于队列的延迟：所有消息延迟时间一样，通过定时扫描，投递超时消息

## 死信队列

异常场景：消息消费出现异常，没有进行ack，触发消息回滚，但回滚消息处于队列顶部，然后不断被处理和回滚，导致队列陷入死循环

死信队列：投递失败的消息，写到一个特殊的队列，避免死循环

## 重试队列

消息第一次消费失败，进入重试队列1，延迟5s；再次投递后又消费失败，则进入重试队列2，延迟10s；以此类推。重试次数超过上限，则进入死信队列。

注意区分延迟队列，都需要设定延迟时间，但：

延迟队列：延迟动作由内部触发，只作用一次

重试队列：延迟动作由消费端触发，作用范围向后传递

## 消费模式

推模式(push)：broker主动推送消息到消费端，实时性好，但需要流机制确保服务端不会压垮消费端

拉模式(pull)：消费端主动向broker拉消息(定时或定量)，实时性较差，但可以根据自身处理能力控制拉取消息量

## 广播消费

点对点(P2P)：消息只消费一次，队列删除消息，消费者不可消费已经被消费端消息。尽管可以配置多个队列和多个消费者，但一条消息只能被一个消费者消费。典型的就是RabbitMQ

发布/订阅(pub/sub)：消息发布者往内容节点(topic主题即消息传递的中介)发布消息，消息订阅者从主题中订阅消息。订阅者与发布者相互独立，不需要接触即可保证消息出传递，常用在一对多的广播场景。典型的就是Kafka

补充：RabbitMQ可以通过设置交换机类型来实现发布/订阅，kafka也能实现点对点消费(把消费者当中队列)。但因为kafka由消息回溯功能，比RabbitMQ更适合广播消费

## 消息回溯

消息会经常出现“丢失”情况，但很难追查是中间件的问题，还是消费端问题。如果支持消息回溯，则可以方便定位原因。消息回溯支持消息消费完成后，还能消费到之前被消费掉的消息。常用于：索引恢复，本地缓存重建，业务补偿方案等

## 消息堆积与持久化

通过消息堆积来达到流量削峰的目的

内存式堆积：如RabbitMQ，但某些条件触发后会有换页动作将内存的消息换页到磁盘

磁盘式堆积：如Kafka，所有消息都存储在磁盘中，磁盘容量比内存大得多，堆积能力就是整个磁盘的大小

## 消息追踪

消息的链路追踪：追踪消息从哪里来，存到哪里，发往哪里去，进行问题的快速定位与排查

## 消息过滤

中间件按照既定规则为下游服务过滤消息

topic过滤：将不同类别的消息，发到不同的topic，即可实现过滤

分区过滤：通过分区对同一个topic中的消息进行分类

filter：kafka steam的fileter功能

## 多租户

即隔离环境：RabbitMQ通过vhost实现环境隔离，避免队列/交换机命名冲突

## 多协议

消息层面协议：AMQP、MQTT、STOMP、XMPP

RabbitMQ支持AMQP和MQTT；Kafka基于本身的私有协议

## 跨语言支持

消息中间件支持的语言越多，则越流行，解耦场景则更多

## 流量控制

针对发送方和接收方速度不匹配，提供一种速度匹配服务，抑制发送速率，使接收方能够适应。流控方法：Stop-and-wait，滑动窗口，令牌桶等

## 消息顺序性

保证消息有序，这个功能有个很常见的应用场景就是CDC（Change Data Chapture），以MySQL为例，如果其传输的binlog的顺序出错，比如原本是先对一条数据加1，然后再乘以2，发送错序之后就变成了先乘以2后加1了，造成了数据不一致

## 安全机制

提供身份认证和权限控制两种安全机制

## 消息幂等性

At most once：至多一次，消息可能丢失，但却不会重复传递

At least once：至少一次，消息绝不会丢失，但可能重复

Exactly once：精确一次，肯定且仅传输一次；一般中间件很难做到这一点

## 事务性

指生产者发送消息的事务，要么成功，要么失败。但本身不提供全局分布式事务的功能

| | **Kafka** | **RabbitMQ** |
| --- | --- | --- |
| 语言 | Scala | Erlang |
| 优先级队列 | 不支持 | 支持，建议优先级大小设置0-10之间 |
| 延迟队列 | 不支持 | 支持 |
| 死信队列 | 不支持 | 支持 |
| 重试队列 | 不支持 | 不支持，用延迟队列实现一个重试队列，可进行二次封装 |
| 消费模式 | 拉模式 | 推模式/拉模式 |
| 广播消费 | 支持，更优秀 | 支持，但力度弱 |
| 消息回溯 | 支持，offset/timestamp两种维度 | 不支持 |
| 消息堆积 | 支持 | 支持，但堆积效率低 |
| 持久化 | 支持 | 支持 |
| 消息追踪 | 不支持 | 支持 |
| 消息过滤 | 客户端级别支持 | 不支持，但可二次封装 |
| 多租户 | 支持 | 支持 |
| 多协议 | 不支持 | AMQP，MQTT，STOMP |
| 跨语言 | 支持多种客户端 | 支持多种客户端 |
| 流量控制 | 支持 | 支持 |
| 消息顺序 | 支持 | 不支持 |
| 安全机制 | 支持(TLS/SSL、SASL) | 支持 |
| 幂等性 | 支持 | 不支持 |
| 事务性 | 支持 | 支持 |
| 性能 | 万级别QPS | 十万/百万级别QPS |