第二次讨论课

1252899 阮康乐

参考的业界架构为 RocketMQ

一. RocketMQ 是一款分布式、队列模型的消息中间件，具有以下特点：

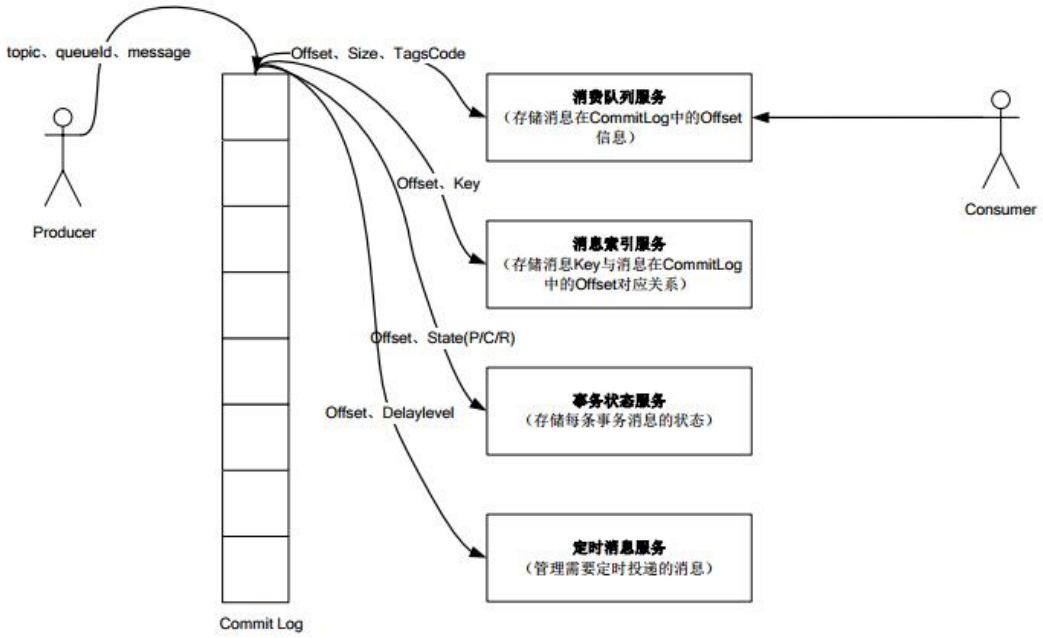
1. 有高性能、高可靠、高实时、分布式
2. Producer、Consumer、队列都可以分布式。
3. Producer 向一些队列轮流发送消息，队列集合称为 Topic，Consumer 如果做广播消费，则一个 consumer 实例消费这个 Topic 对应的所有队列，如果做集群消费，则多个 Consumer 实例平均消费这个 topic 对应的队列集合。
4. 能够保证严格的消息顺序
5. 提供丰富的消息拉取模式
6. 高效的订阅者水平扩展能力
7. 实时的消息订阅机制
8. 亿级消息堆积能力
9. 较少的依赖

二. RocketMQ 的数据存储

1. 零拷贝

   Consumer 消费消息过程，使用了零拷贝，零拷贝包含以下两种方式 1. 使用 mmap + write 方式 优点：即使频繁调用，使用小块文件传输，效率也很高 缺点:不能很好的利用 DMA 方式,会比 sendfile 多消耗 CPU,内存安全性控制复杂，需要避免 JVM Crash 问题。 2. 使用 sendfile 方式 优点:可以利用 DMA 方式,消耗 CPU 较少,大块文件传输效率高,无内存安全新问题。 缺点：小块文件效率低于 mmap 方式，只能是 BIO 方式传输,不能使用 NIO。RocketMQ 选择了第一种方式,mmap+write 方式，因为有小块数据传输的需求，效果会比 sendfile 更好。

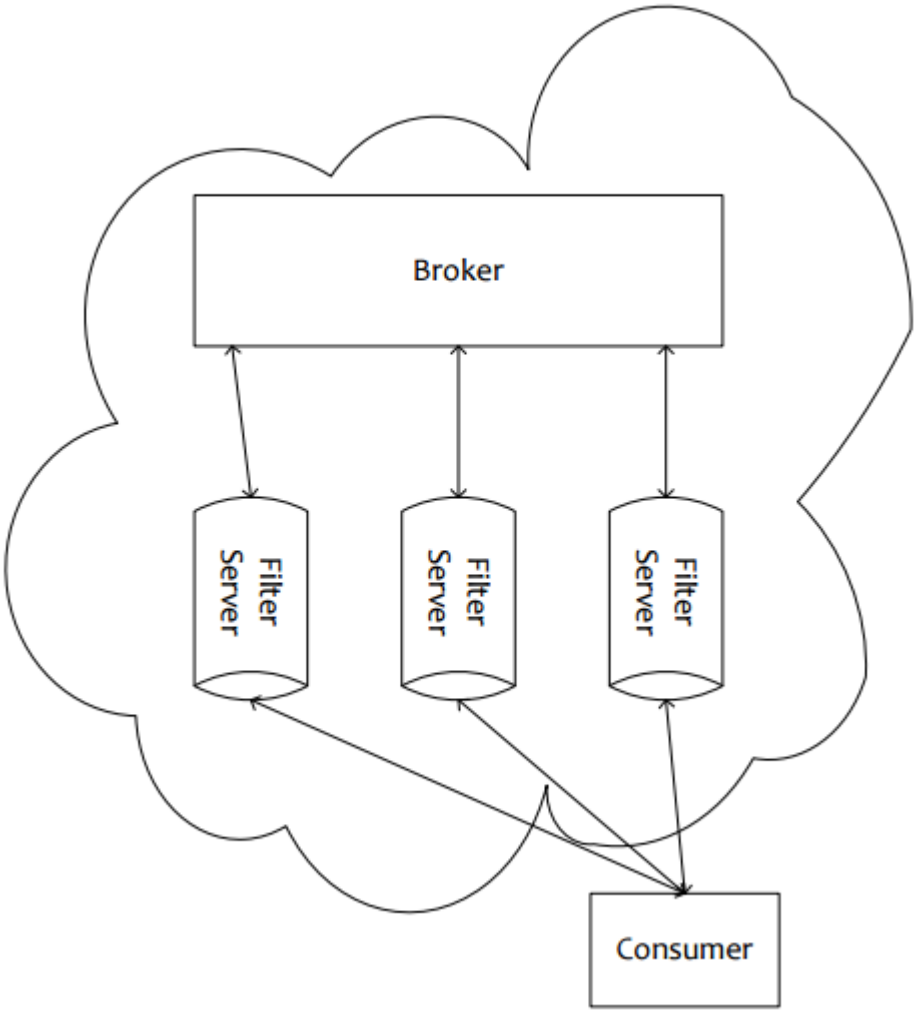
2. 数据存储结构



3. 文件系统
   RocketMQ 选择 Linux Ext4 文件系统

三. RocketMQ 消息过滤

1. 简单消息过滤通过指定多个 Tag 来过滤消息，过滤动作在服务器进行。
   RocketMQ 的消息过滤方式有别于其他消息中间件，是在订阅时，再做过滤，先来看下 Consume Queue 的存储 结构。 CommitLog Offset Size 8 Byte 4 Byte Message Tag Hashcode 8 Byte 图表 7-4Consume Queue 单个存储单元结构 (1). 在 Broker 端进行 Message Tag 比对，先遍历 Consume Queue，如果存储的 Message Tag 与订阅的 Message Tag 不符合，则跳过，继续比对下一个，符合则传输给 Consumer。注意：Message Tag 是字符串形式，Consume Queue 中存储的是其对应的 hashcode，比对时也是比对 hashcode。 (2). Consumer 收到过滤后的消息后，同样也要执行在 Broker 端的操作，但是比对的是真实的 Message Tag 字 符串，而不是 Hashcode。
2. 高级消息过滤



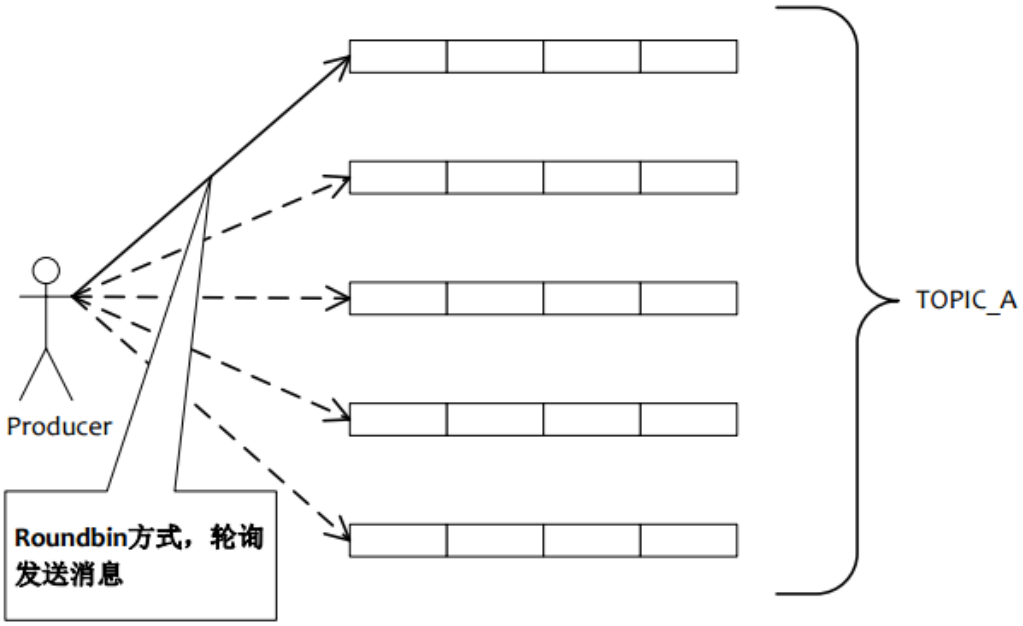
1. Broker 所在的机器会启动多个 FilterServer 过滤进程
2. Consumer 启动后，会向 FilterServer 上传一个过滤的 Java 类
3. Consumer 从 FilterServer 拉消息，FilterServer 将请求转发给 Broker，FilterServer 从 Broker 收到消息后，按照 Consumer 上传的 Java 过滤程序做过滤，过滤完成后返回给 Consumer。

四. RocketMQ 通信协议

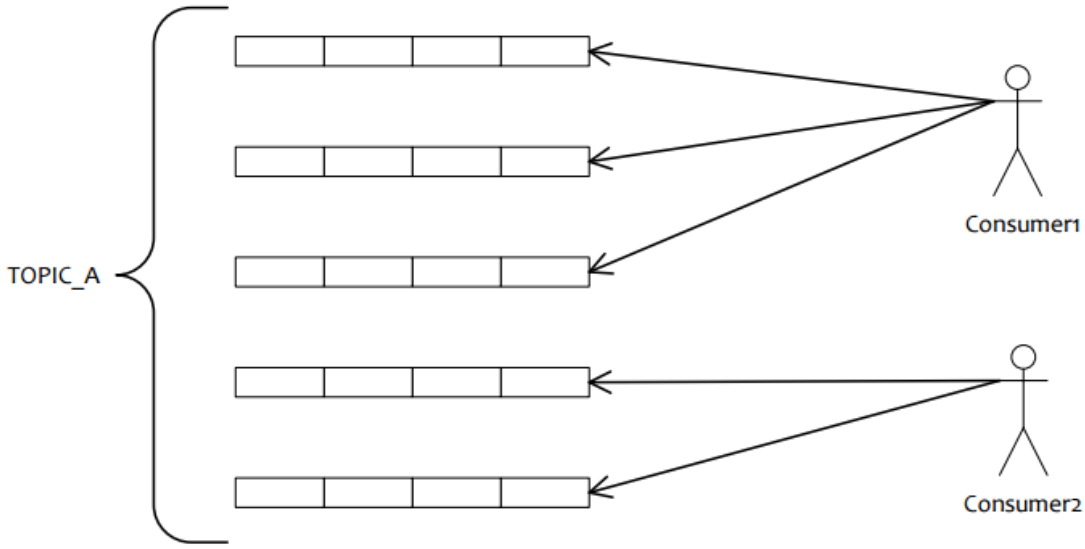
RocketMQ 通信组件使用了 Netty-4.0.9.Final，在之上做了简单的协议封装。

五. RocketMQ 负载均衡

1. 发送消息的负载均衡
   如图所示，5 个队列可以部署在一台机器上，也可以分别部署在 5 台不同的机器上，发送消息通过轮询队列的方式 发送，每个队列接收平均的消息量。通过增加机器，可以水平扩展队列容量。 另外也可以自定义方式选择发往哪个队列。



2. 订阅消息负载均衡



如图所示，如果有 5 个队列，2 个 consumer，那么第一个 Consumer 消费 3 个队列，第二 consumer 消费 2 个队列。 这样即可达到平均消费的目的，可以水平扩展 Consumer 来提高消费能力。但是 Consumer 数量要小于等于队列数 量，如果 Consumer 超过队列数量，那么多余的 Consumer 将不能消费消息。