rocketmq 基本概念

消息

消息即数据，对于 rocketmq 来说，数据只要是 **byte**数组即可，能给数据来点标示更好。
至于数据是什么格式，中间件并不关心。

数据结构

有数据结构，数据才能关联在一起，有关联才能操作。比如 redis，就是 键值数据库。使用hash作为数据结构。
那么消息中间件的数据结构是 链表。queue只有两个操作添加与获得。添加只能添加在 链表的末端。
获得是通过下标获得，注意获得并不会删除数据。删除操作由中间件按照一定策略批量删除。
注意没有修改操作

消息中间件

实现链表结构的软件。在MySQL的引擎中，有一个只能添加与查询的引擎。

broker

消息中间件实例。这个实例里面只有一个库

nameServer

broker 与 **client** 都需要链接 nameServer才能工作。
nameServer负责管理在本实力注册的 broker节点，以及broker的ha，主动识别broker是否存活
nameServer负责 topic与 **group**的路由。**client**直接链接 nameServer。从nameServer得到topic对应的broker

topic(主题)

一类数据的保存，与读取都是从 topic进行操作的，注意topic是逻辑概念。可以认为 topic就是一张表。
同一个topic可以在多个 borker上出现。borker可以存在多个topic。

Message Queue(消息队列)

消息队列的作用是：通过消息队列可以快速定位消息的具体消息。从而非常快的获得数据。
可以把消息队列理解为，数据库里面的索引。只是数据库索引的数据结构是 B+ Tree，而**Message** Queue是顺序索引而已。
每个borker上的topic都有 **Message** Queue。
Message Queue 分读队列与写队列，读写队列可以有多个，不一定相等。写队列是逻辑操作。读队列是物理操作。
可以理解为：borker 里面的 topic分表了。每个子表就是一个读 **Message** Queue。

Producer

负责产生数据。就是往topic里面写数据的实例

Consumer（消费者）

负责消费数据，就是做查询操作的实例

group

共同行为的实例。所以每个实例必须要有**group**

Producer Group

生产者组，没有任何实际意义。只是一种识别而已。
尤其是在查看路由或者其他信息的时候非常有用。
可以设定一组生产者的添加行为细节

Consumer Group

消费者组，这个不仅仅只是表示
尤其是在查看路由或者其他信息的时候非常有用。
除上面之外，其他地方也是是很有用的。
比如 offset的在broker保存，tags维护。等等。

broker Group

组里面所有 **broker**都有一样的行为，一样的**topic**，一样的配置。
主要实现 **broker**的集群，主从功能。
比如，集群里面一个**borker**宕机，那么其他的**borker**可以提供同样的服务。
注意 **borker** 主主不会同步数据。**borker**的主是无状态的。

一些细节要点

消息id与消息key

消息key是 发送消息的时候加入的。目的是用来做幕等操作，可以日志跟踪。
建议key值是唯一，那么无法做幕等操作。
也可以用来做日志跟踪

消息id是 数据文件的时候，broker生成的。

由broker ip+端口+数据物理offset。可以通过消息id找到这条消息。

tags（标签）

给消息添加类型。
Consumer的时候可以只pull一个或多个类型的消息。这是一个很强大的功能，建议大家灵活使用。

Consumer模式

pushComsumer模式

push是推送消息。由某个机制主动把消息推送给消费者使用。

pullComsumer模式

pull是拉取消息。消息的获得细节，规则等由消费者决定。
编写难度大，难控制。性能好，容易优化。

总结

push模式在源码中的实现就是pull模式，**push**的所有操作是由client实现的。

集群消费

消息不需要重复消费（重复拉取）。

比如有三个Comsumer，有30条数据。A消费1-5条，b消费6-15条，c消费16-30条。一条数据只会被一个comsumer消费。

比如有三个Comsumer，有30条数据。每个消费10条。这是均衡消费。这个也不现实。请以后不要说集群消费，就是均衡消费。

广播消费

消息被消费组成员都消费。（消息被多个comsumer消费）
比如有三个Comsumer，有30条数据。A消费30条，b消费30条，c消费30条。每条数据被多个comsumer消费。

总结

Comsumer group里面基本都是使用 集群消费

不同 Comsumer **group** 对一个**topic**消费。那么是广播消费。

分布式情况下，集群消费比 广播消费更难实现。如果用幕等加广播模式实现，意义不大。幕等操作十分消费性能

，增加延迟。

消息事务

目前rocketmq不支持。在大多数情况下，可以曲线完成。
列如：把多条数据合成一条。
不建议过于的追求消息事务。

循序消息，普通顺序消息与严格循序消费

offset

本地保存offset
broker保存offset