# RectetMq

rectetmq这里分为四个部分，nameserve、broker、生产者、消费者

## Nameserve

1. 接收Broker（master和slave）启动时的注册路由信息;

2. 为producer和consumer提供路由服务，即通过topic名字获取所有broker的路由信息；

3. 接收broker发送的心跳信息，如果心跳的时间戳过期NameServer关闭与broker的连接。

## broker

Broker向NameServer注册topic配置信息

{

"perm":6,

"readQueueNums":2,

"topicFilterType":"SINGLE\_TAG",

"topicName":"Topic-Lance",

"writeQueueNums":5

}

Broker的消息存储

Rocketmq的消息的存储是由consumeQueue和 commitLog 配合完成的，commitLog保存消息的物理数据，consumeQueue是消息的逻辑队列，类似于索引，存储的是指向物理存储的地址。在一个Broker上，只有一个commitLog，所有consumeQueue共享同一个commitLog。

## 生产者

1. Producer发送消息时（必须制定topic）,首先从本地的Producer集合中获取topic->broker的路由信息，如果没有，则从nameserver中获取topic->broker路由，并缓存在本地集合中；

2. 定时从nameServer获取最新的topic路由信息；

3. Producer定时将Producer的group信息发送到对应的broker上；

4. Producer发送消息到Master的broker上，通过Broker的主从复制copy到slave的broker上。

## 消费者

1. 向NameServer注册Consumer；

2. 定时从NameServer获取topic路由信息；

3. 定时清理下线的broker;

4. 向所有broker发送心跳；

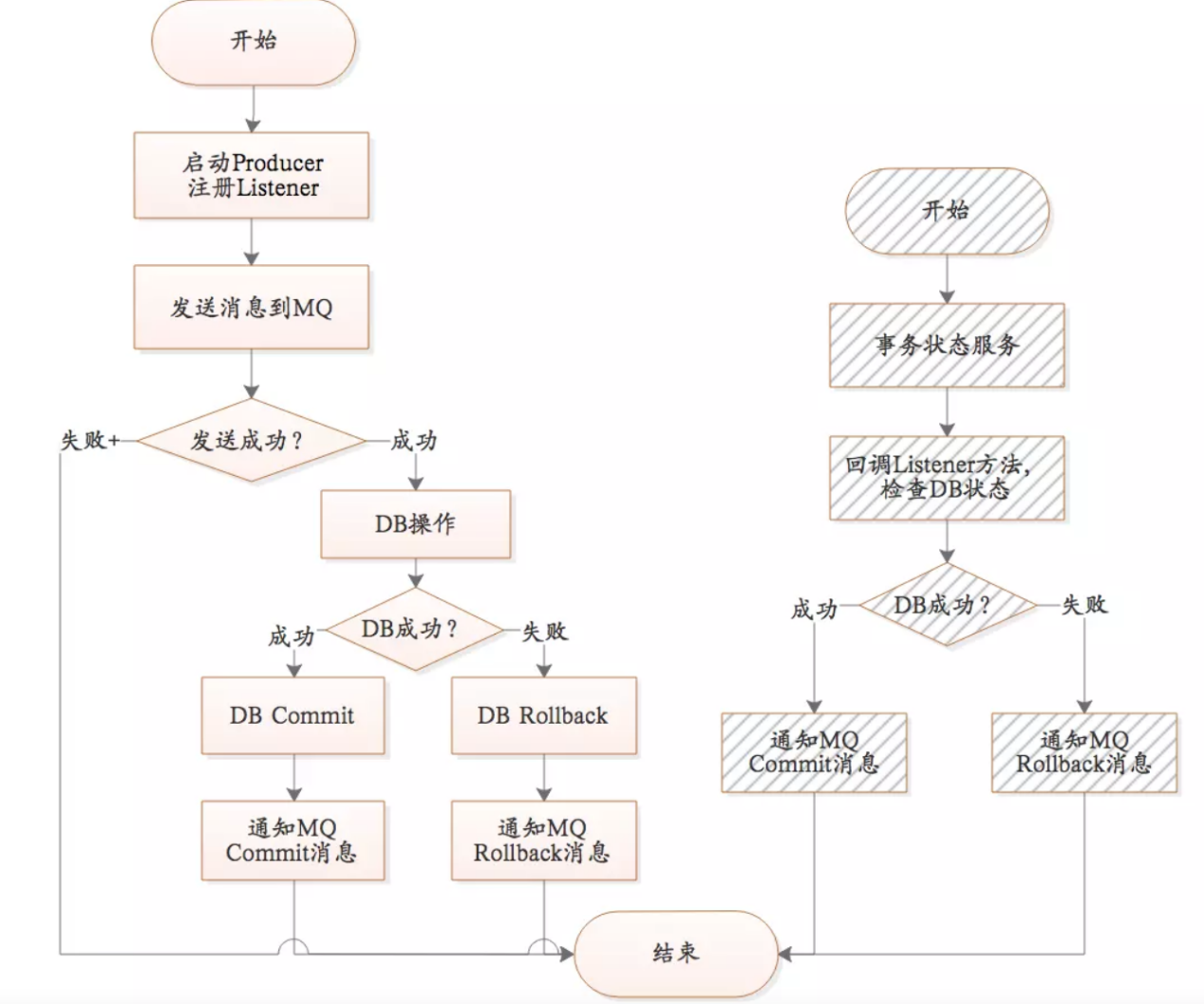
5. 动态调整消费线程池;

6. 负责负载均衡服务RebalanceService。

RocketMQ是基于pull模式拉取消息，consumer做负载均衡并通过长轮询向broker拉消息，长轮询拉取消息后回调MessageListener接口实现完成消费。

## MQ与DB一致性原理（两方事务）

原理：大事务 = 小事务 + 异步



### 为什么使用消息队列

为了解藕、异步处理、消峰处理

**解藕**：使用了消息队列，就不需要接入不同的多个应用。只需要将消息发送到消息队列，然后需要消息的应用去订阅这个消息。

**异步处理**：加快相应速度，把一些非必要的业务逻辑处理异步处理。

**消峰**：在并发量很大，请求很多的时候，减少数据库的压力。把请求放到消息队列中，然后从消息队列中一个一个的去拉取数据。

### 使用消息队列会用哪些缺点

**系统可用性降低**：毕竟是引入了一个中间件，如果中间件挂了，系统依赖中间件的处理就没了。

**系统复杂度增加**：使用了消息中间件，就要考虑数据一致性、消费的幂等性、消息的可靠传输。