# 1.RocketMQ整体架构

首先，RocketMQ整体架构中包含了四个角色：NameServer Broker Poducer Consumer。其中NameServer相当于集群管理服务，主要用于管理整个集群的元数据以及对集群进行管理的。

Broker相当于单个节点负责数据的读写，而Producer就是负责生产消息后写入Broker的，Consumer则是从Broker来消费消息。

而且Broker在部署的时候，一般是分为主备角色的。也就是说，每个Broker都部署了一个Master角色在一台机器上，然后另外会部署一个Slave角色在另外一台机器上。

这样如果某个Broker Master挂了，他的Broker Slave就可以继续接替工作，保证了高可用。

# 2.RocketMQ工作流程

首先需要启动NameServer，接着启动Broker。

Broker启动之后就会跟NameServer建立长连接，每隔一段时间发送心跳包过去。

心跳包里需要包含自己当前存储的数据信息，让NameServer感知到各个Broker的情况。

然后就可以创建topic，创建topic的时候就会决定这个topic的数据会放在哪些Broker上。

每个Broker在进行心跳的时候，就可以上报自己当前存储的topic相关的数据信息给NameServer知道了。

而NameServer就保存了整个集群的元数据，知道了每个topic当前数据保存在哪些Broker上。

Producer在生产消息到topic的时候，先得找NameServer问一下topic存放在哪些Broker上，然后就可以跟那些Broker建立连接发送消息过去了。

Consumer要从Topic消费信息，也会找NameServer问一下可以从哪些Broker上消费消息，接着会跟Broker建立连接并且开始消费消息。

# 3.NameServer原理分析

NameServer一般是集群化部署，各个机器上部署的NameServer会同步互相收到的元数据，保证各个NameServer上包含的集群的元数据是相同的。这样的架构可以保证即使任意一个NameServer宕机了也不要紧。

NameServer主要的责任就是管理Broker的心跳，如果有人一段时间没心跳了，那么就认为那个Broker已经宕机了，此时要触发对应的Slave切换为Master保证高可用。

此外，NameServer还要负责管理和维护集群元数据，比如有哪些Topic，每个Topic在哪些Broker上。

NameServer承受的压力是很小的，因为心跳拉取元数据，其实是很低频的行为，所以一般部署两三台机器就足够了。

但是需要注意的是，如果集群里维护了几十万个，甚至几百万个Topic，会导致Broker每次心跳上报Topic数据时，带上几十万个Topic的数据信息，可能多达即使MB，那么会导致心跳时网络负载压力过重。

# 4.分布式架构分析

再说说Topic，每个Topic都由很多队列组成，一个队列代表这个Topic的部分数据，然后这个Topic的多个队列会均匀分散在多个Broker上。

比如说Topic有6个队列，对应有三台Broker，那么每个Broker上就可以放2个队列的数据。

这样就可以做到每个Topic的数据都可以分布式存储在集群中的多台Broker，形成了分布式的存储架构，这样就可以利用多台机器的资源来存储数据。

同时，可以利用多台机器的资源来应对高并发的请求，因此对一个Topic的读写请求，就会均匀的落到多台Broker机器上去了。

# 5.Broker原理分析

Broker主要就是两点：负责接收Producer写入的消息，同时提供消息给Consumer来读取。

写入消息时，Broker可以实现高并发高性能的写入。

这个主要采用的是对磁盘文件的顺序写的方式，磁盘顺序写的性能是很高的，几乎跟内存随机读写的性能是差不多的。

另外，就是优先把数据写入os page cache，也就是是操作系统管理的缓存中。

写操作系统的内存缓存，那性能就更高了，不需要每次都写入物理磁盘中。

读取消息时，Broker也是主要基于os page cache读取消息，也就是优先从操作系统管理的内存缓存中读取数据，提供给Consumer消费。因此是优先读内存，所有同样性能也是很高的。

# 6.Consumer原理分析

如果我们部署了Consumer在多台机器上，默认也会把Topic下的多个队列的数据均匀的分配给各个Consumer实例进行消费。

比如Topic下一共6个队列，假设部署了3个Consumer实例，此时每个Consumer实例就会消费2个队列的数据。

# 7.总结

对于任何一个消费系统，对他的原理进行研究的时候，都应该关注几点：

1. 集群架构(元数据如何管理、集群如何管理)。
2. 数据模型(Topic、队列的概念)。
3. 分布式架构(消息如何分布式存储)。