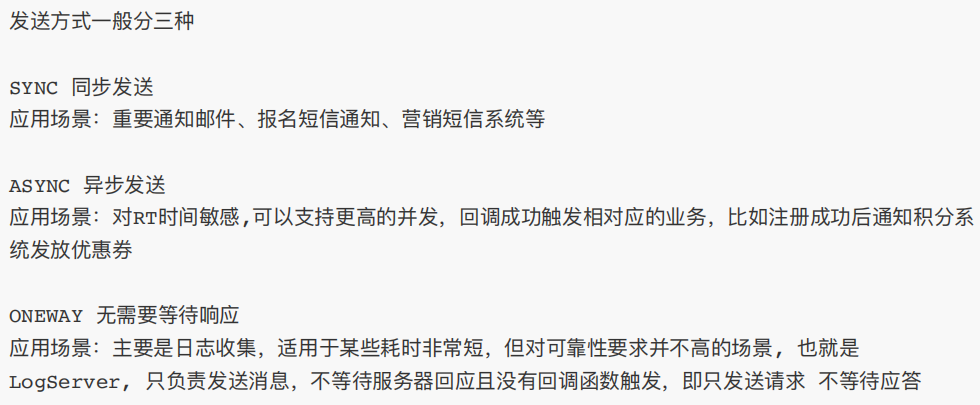
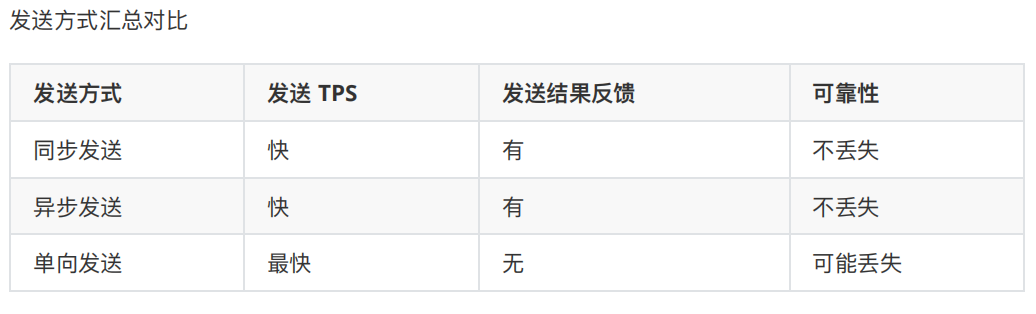


<https://blog.csdn.net/weixin_45393094/article/details/106160234>



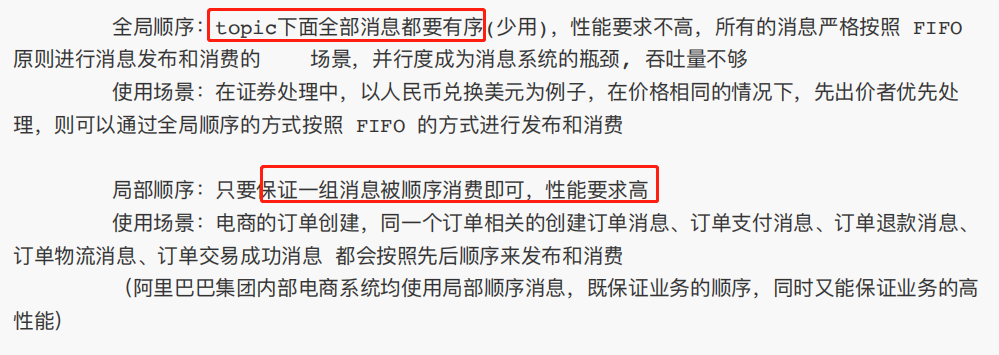
同步异步有重试机制，不会因为网络抖动而数据丢失

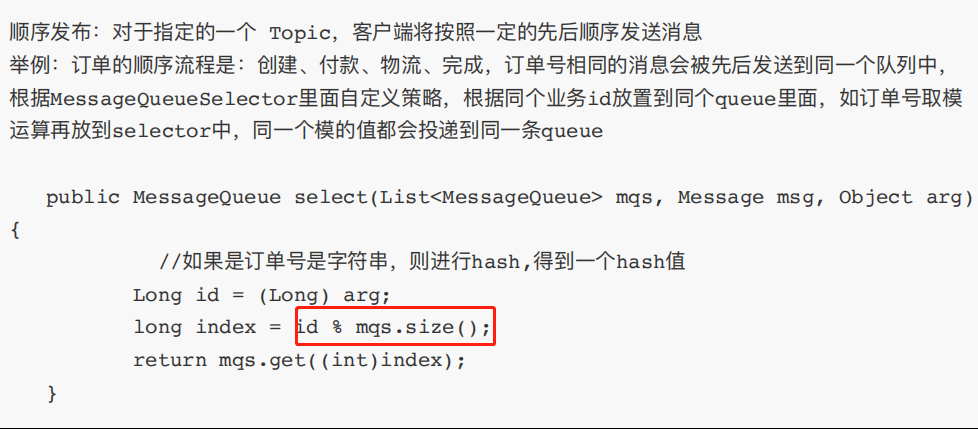


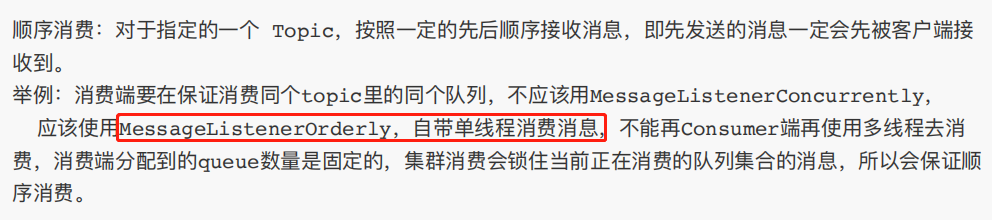


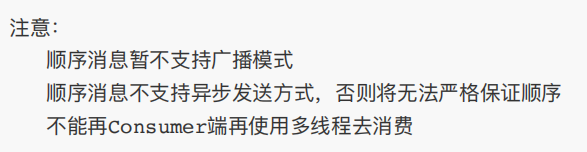












一个queue只能被一个消费者消费



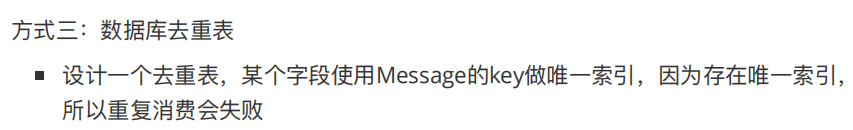
网络原因没有收到消费者回应，其实是到达消费者了的，那可能触发重试机制，导致重复消费

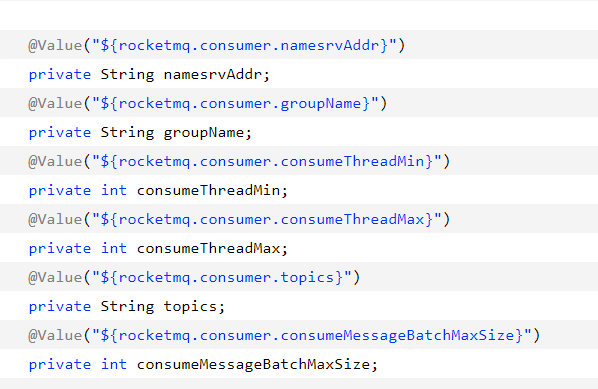






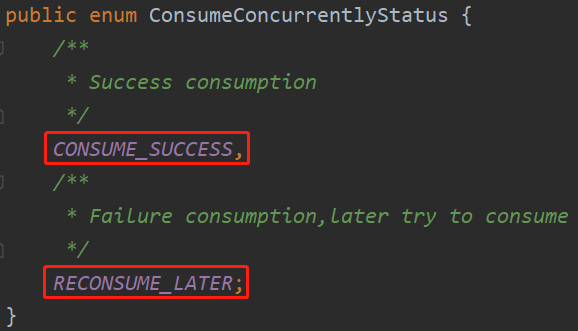






配置连接，yml配置后@value拿取

<https://www.cnblogs.com/bao-bei/p/13162975.html>

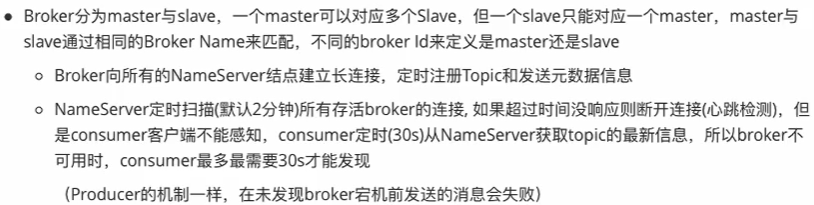


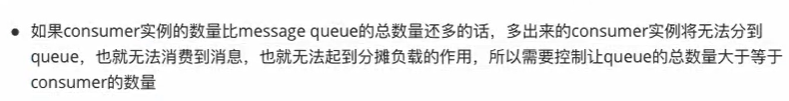
消费者返回consume\_success时，队列才会认为消费成功



消息队列提供ack机制，消费者放回成功后才是真正的成功，返回重试则不断重试，重试到一定次数得人工介入处理。

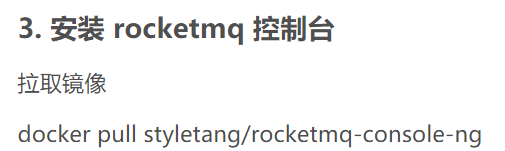
消费者需要做好幂等性处理和日志处理



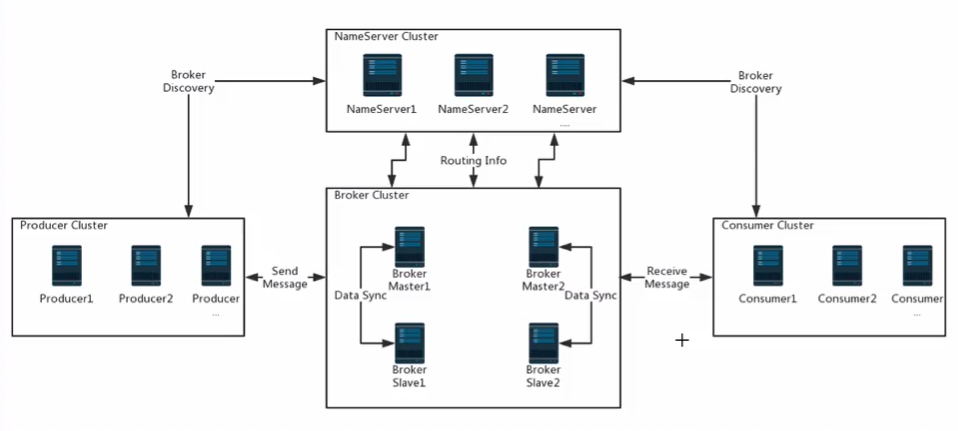








<https://blog.csdn.net/W_317/article/details/104556265>



## 10, Broker Server

消息中转角色，负责存储消息、转发消息。代理服务器在RocketMQ系统中负责接收从生产者发送来的消息并存储、同时为消费者的拉取请求作准备。代理服务器也存储消息相关的元数据，包括消费者组、消费进度偏移和主题和队列消息等。

## 11名字服务（Name Server）

名称服务充当路由消息的提供者。生产者或消费者能够通过名字服务查找各主题相应的Broker IP列表。多个Namesrv实例组成集群，但相互独立，没有信息交换。

相当于注册中心，Broker相当于数据库

电商应用转变成基于消息队列的方式后，系统可用性就高多了，

比如物流系统因为发生故障，需要几分钟的时间来修 ，在这几分钟的时间里，

物流系统要处理的内容被 存在消息队列里，用户的下单操作可以正常完成

使用消息队列进行流 消峰，很多时候不是因为能力不够，而是出于经济

性的考量 比如有的业务系统，流量最高峰也不会超过一万 PS ，而平时只有

一千左右的 QPS 这种情况下我们就可以用个普通性能的服务器（只支持一千

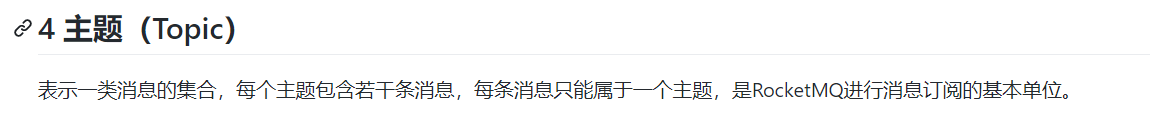
左右的 QPS 就可以），然后加个消息队列作为高峰期的缓冲，无须花大笔资金

部署能处理上万 QPS 的服务器

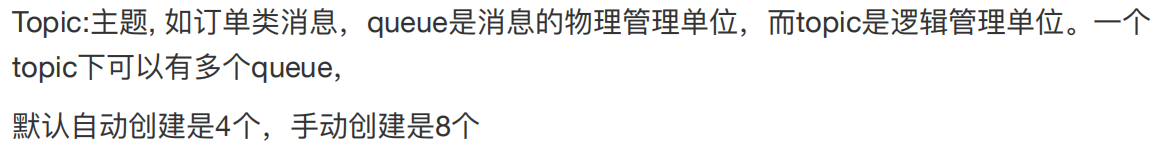
如果订单系统每秒最多能处理 万次下单，这个处理能力应对正常时段的下单是绰绰有余的，正常时段我们下单后一秒内就能返回结果双十一零点的时候，如果没有消息队列这种缓冲机制，为了保证系统稳定，只能在订单超过一万次后就不允许用户下单了；如果有消息队列做缓冲，我们可以取消这个限制，把一秒内下的订单分散成一段时间来处理，这时有些用户可

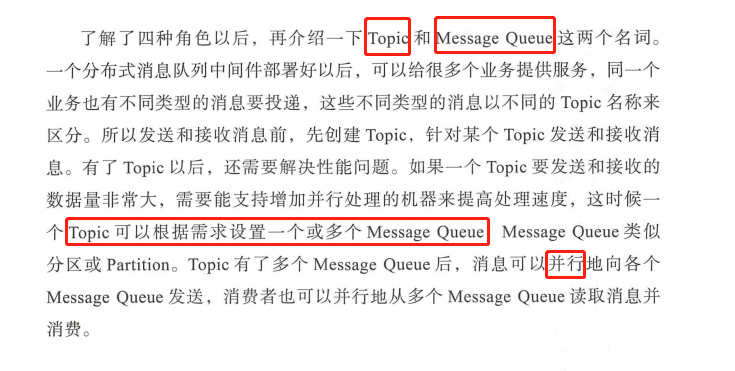
能在下单后十几秒才能收到下单成功的状态，但是也比不能下单的体验要好

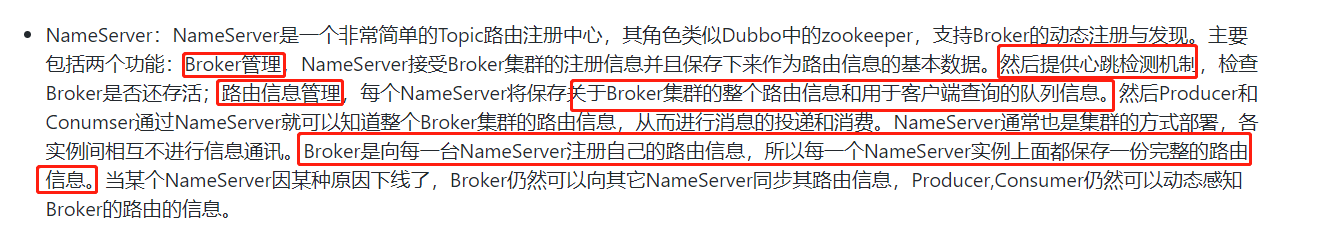
数据的产生方只需要把各自的数据写入一个消息队列即可数据使用方根据各自需求订阅感兴趣的数据，不同数据团队所订阅的数据可以重复也可以不重复，互不干扰也不必和数据产生方关联

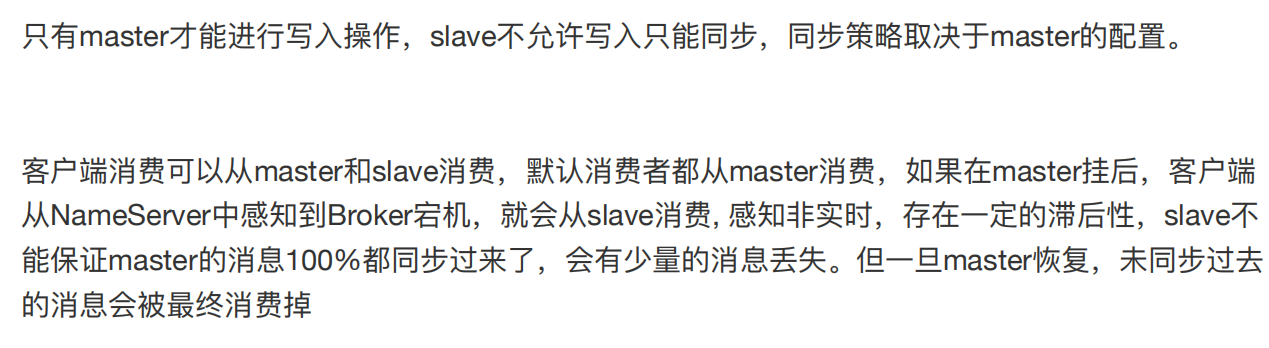












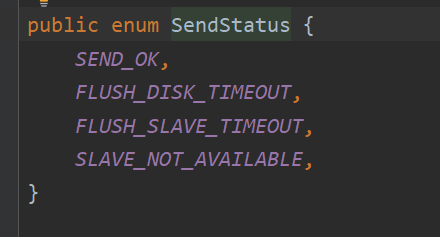
message queue数量要大于consumer

消费者在向Master拉取消息时，Master服务器会根据拉取偏移量与最大偏移量的距离（判断是否读老消息，产生读I/O），以及从服务器是否可读等因素建议下一次是从Master还是Slave拉取。

* 启动NameServer，NameServer起来后监听端口，等待Broker、Producer、Consumer连上来，相当于一个路由控制中心。
* Broker启动，跟所有的NameServer保持长连接，定时发送心跳包。心跳包中包含当前Broker信息(IP+端口等)以及存储所有Topic信息。注册成功后，NameServer集群中就有Topic跟Broker的映射关系。
* 收发消息前，先创建Topic，创建Topic时需要指定该Topic要存储在哪些Broker上，也可以在发送消息时自动创建Topic。
* Producer发送消息，启动时先跟NameServer集群中的其中一台建立长连接，并从NameServer中获取当前发送的Topic存在哪些Broker上，轮询从队列列表中选择一个队列，然后与队列所在的Broker建立长连接从而向Broker发消息。
* Consumer跟Producer类似，跟其中一台NameServer建立长连接，获取当前订阅Topic存在哪些Broker上，然后直接跟Broker建立连接通道，开始消费消息。

## 订阅与发布

消息的发布是指某个生产者向某个topic发送消息；消息的订阅是指某个消费者关注了某个topic中带有某些tag的消息，进而从该topic消费数据。



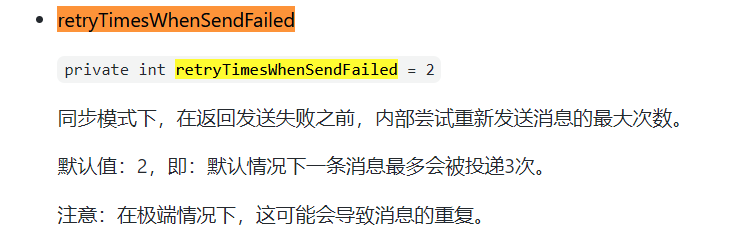


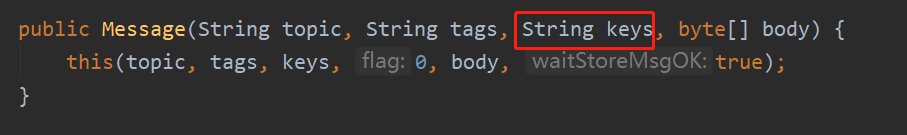
主从同步刷盘超时

主从同步双写超时

主从同步双写找不到从节点

成功发送

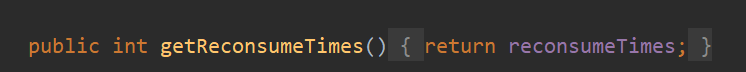


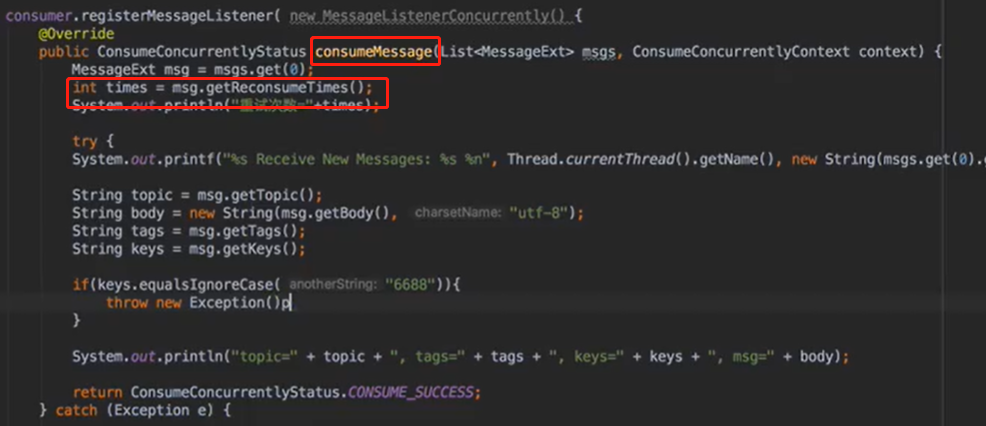


同一Message 的key是不变的（一般设置为组件主键）

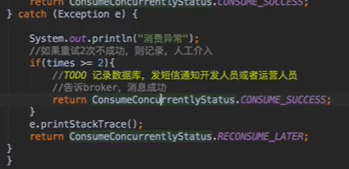


所以一般对key值做幂等性处理





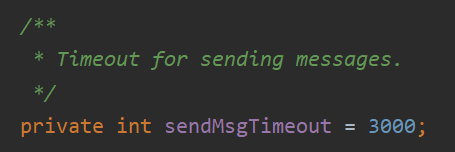
消费者获取消费次数



在消费者有个getReconsumeTimes的方法获取重试次数，但重试次数大于3的时候，发送邮件人工介入

return ConsumeConcurrentlyStatus.*CONSUME\_SUCCESS*;

发送成功



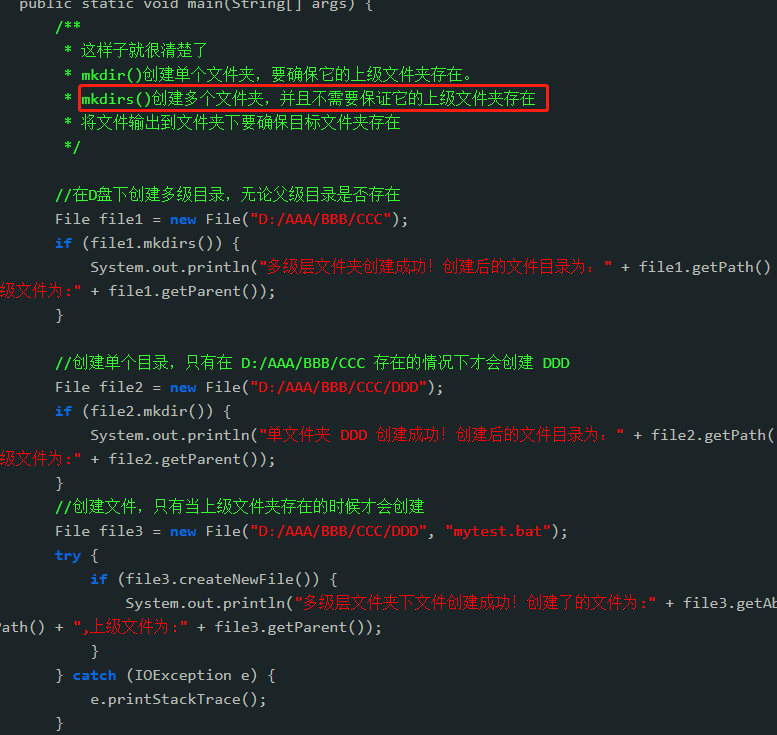
默认3S重试

生产者默认重试3次，消费者重试没默认，需要人工介入

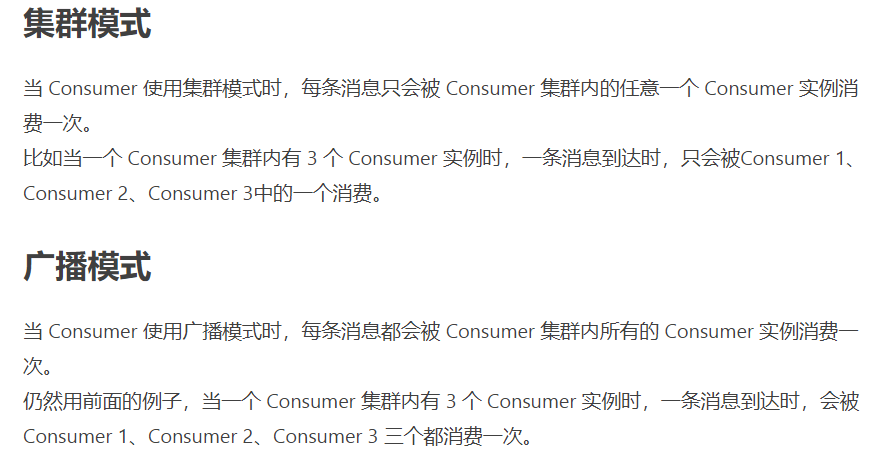


一般消费者判断并写入日志或发短信通知

可以查看发过来的数据是否有什么问题，或者不符合规范的消息



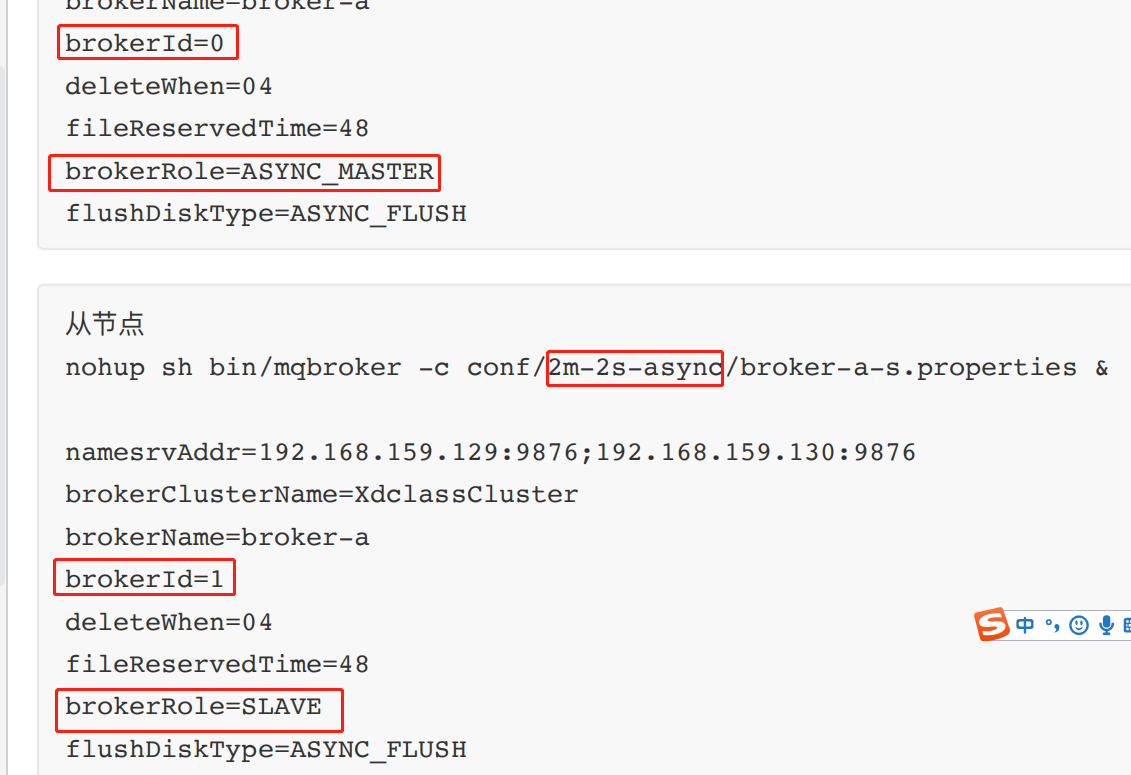




主从节点间

内存到磁盘刷盘，一般异步。

内存到内存双写，一般同步。

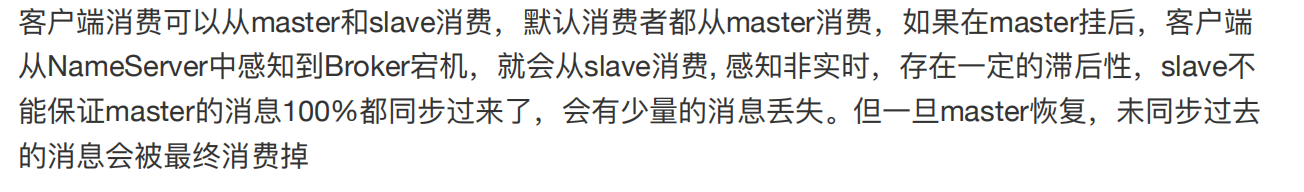


刷盘flushDiskType=ASYNC\_FLUSH异步刷盘

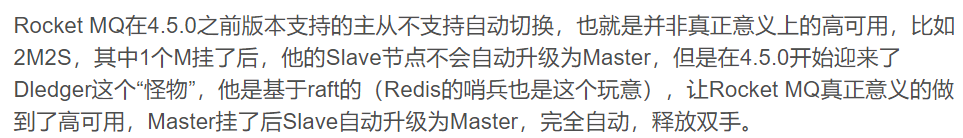
brokerID为0则为主节点

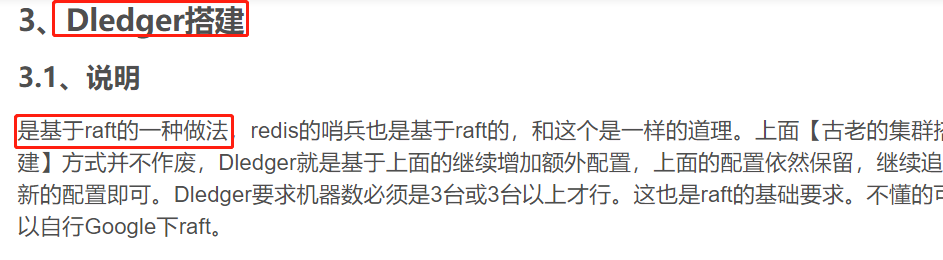
brokerRole角色ASYNC\_MASTER











<https://blog.csdn.net/ctwctw/article/details/107463704>

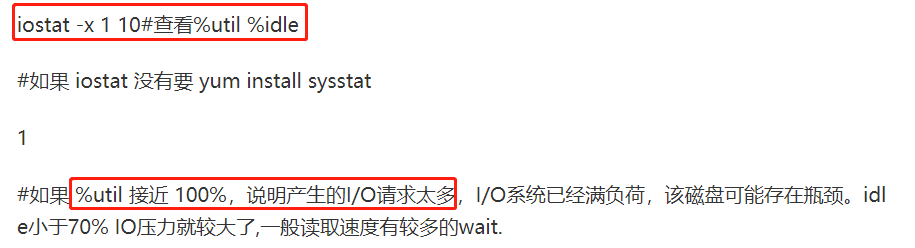


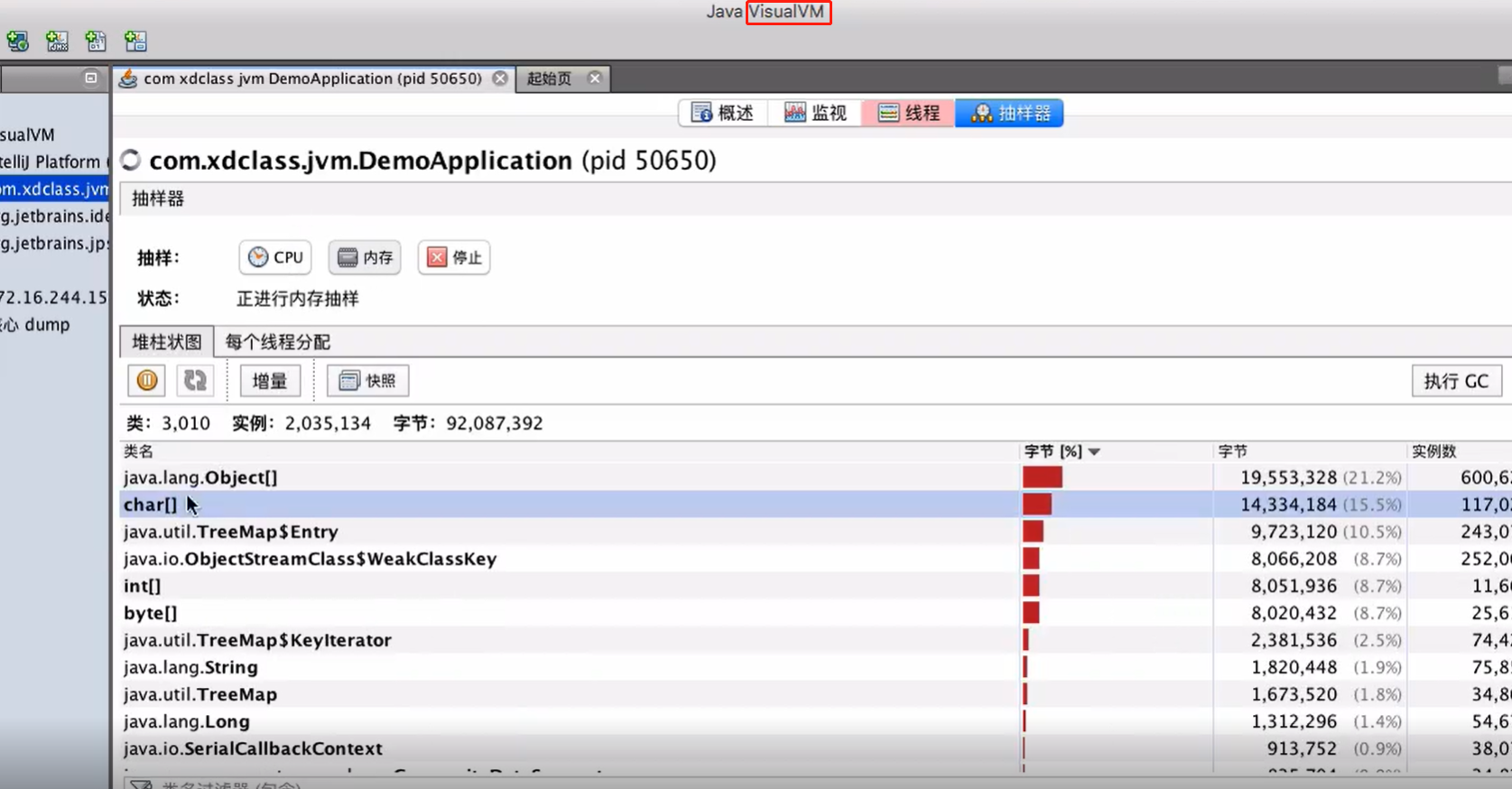
网卡查看带宽是否被占满了

<https://www.cnblogs.com/yfacesclub/p/12015466.html>



磁盘状态如何



dump堆内存快照分析，查看线程状态