```
心跳机制
                                                                                                                                                                                                                                                                                                                        NameServer会每隔10s运行一个任务,去检查一下各个Broker的最近一次心跳时间,如果某个Broker超过120s都没发送心跳了,
                                                                                                                                                                                                                                                                                                                        那么就认为这个Broker已经挂掉了
                                                                                                                                                                                                                                                                                                      Producer: 消息的发送者; 举例: 发信者
                                                                                                                                                                                                                                                                                                      Consumer: 消息接收者; 举例: 收信者
                                                                                                                                                                                                                                                                                                      Broker: 暂存和传输消息; 举例: 邮局
                                                                                                                                                                                                                                                                                                      NameServer: 管理Broker; 举例: 各个邮局的管理机构
                                                                                                                                                                                                                                                                                                      Topic:区分消息的种类;一个发送者可以发送消息给一个或者多个Topic;一个
                                                                                                                                                                                                                                                                                                      消息的接收者可以订阅一个或者多个Topic消息
                                                                                                                                                                                                                                                                                                      Message Queue:相当于是Topic的分区;用于并行发送和接收消息
                                                                                                                                                                                                                                                                                                                   RocketMQ主要由 Producer、Broker、Consumer 三部分组成,其中Producer
                                                                                                                                                                                                                                                                                                                   负责生产消息,Consumer 负责消费消息,Broker 负责存储消息。Broker 在实际
                                                                                                                                                                                                                                                                                                                   部署过程中对应一台服务器,每个 Broker 可以存储多个Topic的消息,每个Topic
                                                                                                                                                                                                                                                                                            消息模型 (Message Model)
                                                                                                                                                                                                                                                                                                                   的消息也可以分片存储于不同的 Broker。Message Queue 用于存储消息的物理地
                                                                                                                                                                                                                                                                                                                   址,每个Topic中的消息地址存储于多个 Message Queue 中。ConsumerGroup
                                                                                                                                                                                                                                                                                                                     由多个Consumer 实例构成。
                                                                                                                                                                                                                                                                                                                 负责生产消息,一般由业务系统负责生产消息。一个消息生产者会把业务应用系
                                                                                                                                                                                                                                                                                                                 统里产生的消息发送到broker服务器。RocketMQ提供多种发送方式,同步发送、
                                                                                                                                                                                                                                                                                            消息生产者 (Producer)
                                                                                                                                                                                                                                                                                                                 异步发送、顺序发送、单向发送。同步和异步方式均需要Broker返回确认信息,单
                                                                                                                                                                                                                                                                                                                  向发送不需要。
                                                                                                                                                                                                                                                                                                                  负责消费消息,一般是后台系统负责异步消费。一个消息消费者会从Broker服务
                                                                                                                                                                                                                                                                                                                  器拉取消息、并将其提供给应用程序。从用户应用的角度而言提供了两种消费形
                                                                                                                                                                                                                                                                                                                  式: 拉取式消费、推动式消费。
                                                                                                                                                                                                                                                                                                                                拉取式消费的应用通常主动调用Consumer的拉消息方法从Broker服务器拉消
                                                                                                                                                                                                                                                                                                                               息、主动权由应用控制。一旦获取了批量消息,应用就会启动消费过程。
                                                                                                                                                                                                                                                                                                                                推动式消费模式下Broker收到数据后会主动推送给消费端,该消费模式一般实时
                                                                                                                                                                                                                                                                                            消息消费者 (Consumer)
                                                                                                                                                                                                                                                                                                                  消费者同样会把同一类Consumer组成一个集合,叫做消费者组,这类
                                                                                                                                                                                                                                                                                                                   Consumer通常消费同一类消息且消费逻辑一致。消费者组使得在消息消费方面,
                                                                                                                                                                                                                                                                                                                  实现负载均衡和容错的目标变得非常容易。要注意的是,消费者组的消费者实例必
                                                                                                                                                                                                                                                                                                                  须订阅完全相同的Topic。RocketMQ 支持两种消息模式:集群消费
                                                                                                                                                                                                                                                                                                                   (Clustering) 和广播消费 (Broadcasting)。
                                                                                                                                                                                                                                                                                                                                 相同Consumer Group的每个Consumer实例平均分摊消息
                                                                                                                                                                                                                                                                                                                                相同Consumer Group的每个Consumer实例都接收全量的消息
                                                                                                                                                                                                                                                                                                            表示一类消息的集合,每个主题包含若干条消息,每条消息只能属于一个主题,
                                                                                                                                                                                                                                                                                                            是RocketMQ进行消息订阅的基本单位
                                                                                                                                                                                                                                                                                                                    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
                                                                                                                                                                                                                                                                                           代理服务器 (Broker Server)
                                                                                                                                                                                                                                                                                                                   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
                                                                                                                                                                                                                                                                                                                   名称服务充当路由消息的提供者。Broker Server会在启动时向所有的Name
                                                                                                                                                                                                                                                                                                                  Server注册自己的服务信息,并且后续通过心跳请求的方式保证这个服务信息的实
                                                                                                                                                                                                                                                                                            名字服务 (Name Server)
                                                                                                                                                                                                                                                                                                                  时性。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
                                                                                                                                                                                                                                                                                                                  Namesry实例组成集群,但相互独立,没有信息交换。
                                                                                                                                                                                                                                                                                                                                                         RocketMQ的Master-Slave模式采取的是Slave Broker不停的发送请求到Master Broker去拉取消息
                                                                                                                                                                                                                                                                                                                            Master Broker消息同步给Slave Broker
                                                                                                                                                                                                                                                                                                                                                         写入消息肯定选择Master Broker写入
                                                                                                                                                                                                                                                                                                                                                                            Master Broker在返回消息给消费者系统的时候,会根据当时Master Broker的负载情况和Slave Broker的同步情况,向消费者系
                                                                                                                                                                                                                                                                                                                             消费者有可能从Master Broker获取消息,也有可能从Slave Broker获取消息
                                                                                                                                                                                                                                                                                                                                                                           统建议下一次拉取消息的时候是从Master Broker拉取还是从Slave Broker拉取。
                                                                                                                                                                                                                                                                                                                                                                      都是用Slave Broker同步数据,尽量保证数据不丢失,但是一旦Master故障了,Slave是没法自动切换成Master的。这时就得手动做一些运维操作,把Slav
                                                                                                                                                                                                                                                                                                                                                                      e Broker重新修改一些配置,重启机器给调整为Master Broker,这是有点麻烦的,而且会导致中间一段时间不可用
                                                                                                                                                                                                                                                                                                                                                 RocketMQ 4.5版本之前
                                                                                                                                                                                                                                                                                                                                                                      不是彻底的高可用模式,他没法实现自动把Slave切换为Master
                                                                                                                                                                                                                                                                                                                             如果Master Broker挂掉
                                                                                                                                                                                                                                                                                                                                                                                                   Dledger是基于Raft协议实现的一个机制
                                                                                                                                                                                                                                                                                                                                                                   基于Dledger实现RocketMQ高可用自动切换
                                                                                                                                                                                                                                                                                                                                                                                                  此时一旦Master Broker宕机了,就可以在多个副本,也就是多个Slave中,通过Dledger技术和Raft协议算法进行leader选举,直接将
                                                                                                                                                                                                                                                                                                                                                                                                   一个Slave Broker选举为新的Master Broker,然后这个新的Master Broker就可以对外提供服务了,整个过程也许只要10秒或者几十秒的时间
                                                                                                                                                                                                                                                                                                                                                                                                                  Broker会每隔30秒发送心跳到所有的NameServer上去,然后每个NameServer都会每隔10s检查一次有没有哪个Broker超过120s没发送心跳的,如果有,就
                                                                                                                                                               采用类加载器把编译好的class字节码文件加载到jvm中
                                                                                                                         jvm会基于自己的字节码执行引擎,来执行加载到内存里的类
                                                                                                                                                                                                    由编译器编译成class文件
                                                                                                                                                                                                                                                                                                                                                                                                                  认为那个Broker已经宕机了,从路由信息里要摘除这个Broker。
                                                                                                                                                                                                                                                                                                                            Broker是如何跟NameServer进行通信
                                                                                                                                                                                                                                                                                                                                                       Broker会跟每个NameServer都建立一个TCP长连接,然后定时通过TCP长连接发送心跳请求过去
                                                                                                                                                                                                                                                                                                                                                                                                                  每个Broke在进行定时的心跳汇报给NameServer的时候,都会告诉NameServer自己当前的数据情况,
                                                                                                                                                                                                                                                                                                                                                                                                                  比如有哪些Topic的哪些数据在自己这里,这些信息都是属于路由信息的一部分
                                                                                                                                                                                          验证是否符合jvm规范
                                                                                                                                                                                                                                                                                                                                                        先有一个Topic,那么就可以跟NameServer建立一个TCP长连接,然后定时从他那里拉取到最新的路由信息,包括集群里有哪些Broker,集群里有哪些Topi
                                                                                                                                                                                                                                                                                                                                                        c,每个Topic都存储在哪些Broker上。然后生产者系统自然就可以通过路由信息找到自己要投递消息的Topic分布在哪几台Broker上,此时可以根据负载均算
                                                                                                                                                                                                                                                                                                                            生产者系统是如何将消息发送给Broker
                                                                                                                                                                                                                                                                                                                                                        法,从里面选择一台Broke机器出来,比如round robine轮询算法,或者是hash算法,都可以。总之,选择一台Broker之后,就可以跟那个Broker也建立一
                                                                                                                                                                                                                                                                                                                                                         个TCP长连接,然后通过长连接向Broker发送消息即可
                                                                                                                                                                                    给类变量分配空间,和默认初始值
                                                                                                                                                                                                                         加载到使用过程
                                                                                                                                                                                                                                                                                                                                                       他们也会跟NameServer建立长连接,然后拉取路由信息,接着找到自己要获取消息的Topic在哪几台Broker上,就可以跟Broker建立长连接,从里面拉取消
                                                                                                                                                                                                                                                                                                                             消费者是如何从Broker上拉取消息的
                                                                                                                                                                                        符号引用替换位直接引用
                                                                                                                                                                                                                                                                                                                                                       息了。消费者系统可能会从Master Broker拉取消息,也可能从Slave Broker拉取消息,都有可能
                                                                                                                                                                  对类变量进行赋值,如果父类还没有初始化,先对父类进行初始化
                                                                                                                                                                                                                                                                                                                                         当Broker接收到一个消息的时候,会直接写入磁盘上的一个日志文件,顺序写入叫做CommitLog,这个CommitLog是很多磁盘文件,每个文件限定最多1G
                                                                                                                                                                                                                                                                                                                                        B, Broker收到消息后追加到文件的末尾,一个CommitLog写满1G,就会创建一个新的CommitLog文件
                                                                                                                                                                                                                                                                                                                                         Broker的磁盘上,会有$HOME/store/consumequeue/{topic}/{queueld}/{fileName}一系列文件,当你的Broker收到一条消息写入了CommitLog之后,其
                                                                                                                                                                                                                                                                                                                                        实他同时会将这条消息在CommitLog中的物理位置,也就是一个文件偏移量,就是一个offset,写入到这条消息所属的MessageQueue对应的ConsumeQu
                                                                                                                                                       他主要是负责加载我们在机器上安装的Java目录下的核心类的
                                                                                                                                                                                                                                                                                                                                        在ConsumeQueue中存储的每条数据不只是消息在CommitLog中的offset偏移量,还包含了消息的长度,以及tag
                                                                                                                                                                                                                                                                                                                                        hashcode,一条数据是20个字节,每个ConsumeQueue文件保存30万条数据,大概每个文件是5.72MB
                                                                                                                                              这个类加载器其实也是类似的,就是你的Java安装目录下,有一个"lib\ext"目录
                                                                                                                                                                                               Extension ClassLoader(扩展类加载器)
                                                                                                                                                                                                                                                                                                                                         所以实际上Topic的每个MessageQueue都对应了Broker机器上的多个ConsumeQueue文件,保存了这个MessageQueue的所有消息
                                                                                                                                                                                            Application ClassLoader(应用程序类加载器)
                                                                                                                                                                                                                                                                                                                                        在CommitLog文件中的物理位置,也就是offset偏移量。
                                                                                                                                                 这类加载器就负责去加载 "ClassPath" 环境变量所指定的路径中的类
                                                                                                                                                                                                                                                                                                                                                                                    Broker是基于OS操作系统的PageCache和顺序写两个机制
                                                                                                                                                                                                           自定义类加载器
                                                                                                                                                                                                                                                                                                                                                                                    首先Broker是以顺序的方式将消息写入CommitLog磁盘文件的,也就是每次写入就是在文件末尾追加一条数据就可以了,对文件进行
                                                                                                                                                                                                      1、继承ClassLoader
                                                                                                                                                                                                                                                                                                                                                                                    顺序写的性能要比对文件随机写的性能提升很多
                                                                                                                                                                                                                         自定义类加载器
                                                                                                                                                   2、重写findClass,在findClass里获取类的字节码,并调用ClassLoader中的defineClass方法来加载类,获取class对象。
                                                                                                                                                                                                                                                                                                                                                                                    另外,数据写入CommitLog文件的时候,其实不是直接写入底层的物理磁盘文件的,而是先进入OS的PageCache内存缓存中,然后后
                                                                                                                                                   注意:如果要打破双亲委派机制,需要重写loadClass方法。
                                                                                                                                                                                                                                                                                                                                                      消息写入CommitLog文件近乎内存写性能
                                                                                                                                                                                                                                                                                                                                                                                    续由OS的后台线程选一个时间,异步化的将OS PageCache内存缓冲中的数据刷入底层的磁盘文件。
                                                                                                                                             JVM的类加载器是有亲子层级结构的,就是说启动类加载器是最上层的,扩展类加载器在第二层,第三层是应用程序类加载器,最后一
                                                                                                                                                                                                                          双亲委派机制
                                                                                                                                                                                                                                                                                                                                                                                    所以在这样的优化之下,采用磁盘文件顺序写+OS PageCache写入+OS异步刷盘的策略,基本上可以让消息写入CommitLog的性能
                                                                                                                                             层是自定义类加载器。
                                                                                                                                                                                                                                                                                                                                                                                    跟你直接写入内存里是差不多的,所以正是如此,才可以让Broker高吞吐的处理每秒大量的消息写入
                                                                                                                                                                                     JDK1.8以前代表JVM的一块区域
                                                                                                                                                                                                                                                                                                                            消息存储
                                                                                                                                                                                                                                                                                                                                                                                    问题:异步刷盘实际消息在Broker机器上的os cache中,如果宕机,消息会丢失
                                                                                                                                                                                 JDK1.8后改名为Metaspace元数据空间
                                                                                                                                                                                                                                                                                                                                                       ConsumeQueue文件也是基于os cache的,以为ConsumeQueue很小,所以在访问的时候,基本上都可以在os cache中找到
                                                                                                                                                                                  主要存放我们自己写的各种类相关信息
                                                                                                                                                                                                                                                                                                                                                      生产者发送一条消息出去,broker收到了消息,必须直接强制把这个消息刷入底层的物理磁盘文件中,然后才会返回ack给producer,此时你才知道消息写入
                                                                                                                                                         首先该类的所有实例对象都已经从java堆内存里被回收
                                                                                                                                                                                                                                                                                                                                                       导致每条消息写入性能急剧下降,导致消息写入吞吐量急剧下降,但是可以保证数据不会丢失。
                                                                                                                                                               其次加载这个类的ClassLoader已经被回收
                                                                                                                                                                                             方法区类被回收
                                                                                                                                                                                                                                                                                                                                                                         一条数据就会在三个Broker上有三份副本,此时如果Leader Broker宕机,那么就直接让其他的Follower Broker自动切换
                                                                                                                                                                 最后,这该类的Class对象没有任何引用
                                                                                                                                                                                                                                                                                                                                                                        为新的Leader Broker,继续接受客户端的数据写入就可以了
                                                                                                                                                                      来记录当前执行的字节码指令的位置的,每一个线程一个
                                                                                                                                                                                                                                                                                                                                                                                                        发起一轮一轮的投票,通过三台机器互相投票选出来一个人作为Leader,Broker01是投票给自己的,Broker02是投票给自己的,Broker03是投票给自己的,
                                                                                                                                                                                                                                                                                                                                                                                                        第一轮选举是失败,接着每个人会进入一个随机时间的休眠,比如说Broker01休眠3秒,Broker02休眠5秒,Broker03休眠4秒,此时Broker01必然是先苏醒
                                                                                                                                                                                                                                                                                                                                                                                                        过来的,他苏醒过来之后,直接会继续尝试投票给自己,并且发送自己的选票给别人,接着Broker03休眠4秒后苏醒过来,他发现Broker01已经发送来了一个
                                                                                                                                                                    每个线程都有自己的虚拟机栈,存放自己执行的方法的局部变量
                                                                                                                                                                                                                                                                                           中间件路由中心的架构
                                                                                                                                                                                                                                                                                                                                                                        DLedger是基于Raft协议选举Leader Broker
                                                                                                                                                                                                                                                                                                                                                                                                        选票是投给Broker01自己的,此时他自己因为没投票,所以会尊重别人的选择,就直接把票投给Broker01了,同时把自己的投票发送给别人,此时所有人都
                                                                                                                                                                                                                                                                                                                                                                                                        会收到三张投票,都是投给Broker01的,那么Broker01就会当选为Leader,其实只要有(3台机器 / 2) + 1个人投票给某个人,就会选举他当Leader,这个
                                                                                                                                                                            如果线程每调用一个方法就创建对应的一个栈帧
                                                                                                                                                                                                                                                                                                                                         基于DLedger技术替换Broker的CommitLog
                                                                                                                                                                                                                                                                                                                                                                                                        (机器数量 / 2) + 1就是大多数的意思
                                                                                                                                                            存对象的地址,指向堆中对象
                                                                                                                                                                                                                                                                                                                                                                                                     数据同步会分为两个阶段,一个是uncommitted阶段,一个是commited阶段
                                                                                                                                                                                                                                                                                                                                                                                                     Leader Broker上的DLedger收到一条数据之后,会标记为uncommitted状态,然后他会通过自己的DLedgerServer组件把这个uncommitted数据发送给Fol
                                                                                                                                                                                                                                                                                                                                                                         DLedger基于Raft协议进行多副本同步
                                                                                                                                                                                                                                                                                                                                                                                                     lower Broker的DLedgerServer。接着Follower Broker的DLedgerServer收到uncommitted消息之后,必须返回一个ack给Leader Broker的DLedgerServe
                                                                                                                                                                                                                                                                                                                                                                                                     r,然后如果Leader Broker收到超过半数的Follower Broker返回ack之后,就会将消息标记为committed状态,然后Leader Broker上的DLedgerServer就会
                                                                                                                                                                                                                                                                                                                                                                                                     发送commited消息给Follower Broker机器的DLedgerServer,让他们也把消息标记为comitted状态
                                                                                                                                                                                                                                                                                                                                                            本质就是根据你要消费的MessageQueue以及开始消费的位置,去找到对应的ConsumeQueue读取里面对应位置的消息在CommitLog中的物理offset偏移
                                                                                                                                                                                                                                                                                                                           Broker是如何将消息读取出来返回给消费机器
                                                                                                                                                                                                                                                                                                                                                           量,然后到CommitLog中根据offset读取消息数据,返回给消费者机器
                第一种可能,Minor GC过后,剩余的存活对象的大小,是小于Survivor区的大小的,那么此时存活对象进入Survivor
                                                                                     在执行任何一次Minor GC之前,JVM会先检查一下老年代可用的可用内存空间,是否大于新生代所有对象的总大小,所以假如Minor GC之前,发现老年代的
                第二种可能,Minor GC过后,剩余的存活对象的大小,是大于 Survivor区域的大小,但是是小于老年代可用内存大小
                                                                                     可用内存已经小于了新生代的全部对象大小了,就会看一个"-XX:-HandlePromotionFailure"的参数是否设置了,如果有这个参数,那么就会继续尝试进行
                                                                                                                                                                                                                                                                                                                                                                                     首先从磁盘上把数据读取到内核IO缓冲区里去,然后再从内核IO缓存区里读取到用户进程私有空间里去,然后我们才能拿到这个文件里的数据,然后从这里再
               的,此时就直接进入老年代即可。
                                                                                     下一步判断,下一步判断,就是看看老年代的内存大小,是否大于之前每一次Minor GC后进入老年代的对象的平均大小。如果上面那个步骤判断失败了,或
                                                                                                                                                                                                                                                                                                                                                         传统文件IO操作的多次数据拷贝问题
                                                                                                                                                                                                                                                                                                                                                                                     进入内核IO缓冲区,最后进入磁盘文件里去
                                                                                     者是"-XX:-HandlePromotionFailure"参数没设置,此时就会直接触发一次"Full GC",就是对老年代进行垃圾回收,尽量腾出来一些内存空间,然后再
                                                                                                                                                                              Minor GC(Young GC)
第三种可能,很不幸,Minor GC过后,剩余的存活对象的大小,大于了Survivor区域的大小,也大于了老年代可用内
存的大小。此时老年代都放不下这些存活对象了,就会发生"Handle Promotion Failure"的情况,这个时候就会触
发一次 "Full GC"。是Full GC过后,老年代还是没有足够的空间存放Minor GC过后的剩余存活对象,那么此时就会导致所谓的"OOM"内存溢出了
                                                                                                                                                                                                                                                                                                                                                                                               如果具体到代码层面,就是基于JDK NIO包下的MappedByteBuffer的map()函数,来先将一个磁盘文件(比如一个CommitLog文
                                                                                                                                                                                                                                                                                                                                                                                               件,或者是一个ConsumeQueue文件)映射到内存里来
                                                                                                                                                        速度较快一次可能100ms
                                                                                                                                                                                                                                                                                                                                                                                               因为刚开始你建立映射的时候,并没有任何的数据拷贝操作,其实磁盘文件还是停留在那里,只不过他把物理上的磁盘文件的一些地址
                                                                                                                                                           对象经历过15次垃圾回收后,还是没有被回收,进入老年代
                                                                                                                                                                                                                                                                                                                                                                                               和用户进程私有空间的一些虚拟内存地址进行了一个映射
                                                                                                                                                                                                                                                                                                                                                                                               这个mmap技术在进行文件映射的时候,一般有大小限制,在1.5GB~2GB之间,所以RocketMQ才让CommitLog单个文件在1GB,ConsumeQueue文件在
                                                                                                     一批对象的总大小大于这块Survior区域的内存大小的50%,那么大于等于这批年龄的对象直接进入老年代
                                                                                                                                                                                                                                                                                                                            mmap: Broker读写磁盘文件的核心技术
                                                                                                                                                                                                                                                                                                                                                                                               接下来就可以对这个已经映射到内存里的磁盘文件进行读写操作了,比如要写入消息到CommitLog文件,你先把一个CommitLog文件
                                                                                                                                        的逻辑: 年龄1+年龄2+年龄n的多个年龄对象总和超过了Survivor区
                                                                                                                                                                                                                                                                                                                                                                                               通过MappedByteBuffer的map()函数映射其地址到你的虚拟内存地址。
                                                                                                                                       域的50%,此时就会把年龄n以上的对象都放入老年代。
                                                                                                                                                                                                                                                                                                                                                          RocketMQ是如何基于mmap技术+page cache技术优化
                                                                                                                                                                                                                                                                                                                                                                                               接着就可以对这个MappedByteBuffer执行写入操作了,写入的时候他会直接进入PageCache中,然后过一段时间之后,由os的线程
                                                                         JVM参数,就是"-XX:PretenureSizeThreshold",可以把他的值设置为字节数,比如"1048576"字节,就是1MB。大于这个值对象直接进入老年代
                                                                                                                                                                                                                                                                                                                                                                                               异步刷入磁盘中,这样的话只是进行了一次数据拷贝的过程
                                                                                                                                                                                                                                                                                                                                                                                               读取数据的时候,如果数据在PageCache中可以直接读取,如果不在的话会到磁盘中获取,然后会将你加载的数据块的临近的其他数据块也一起加载到Page
                                                                                                                  这个时候就必须得把这些对象直接转移到老年代去
                                                                                                                                                   Minor GC后的对象太多无法放入Survivor区
                                                                                                                                                                                                                                                                                                                                                                                                                                          Broker会针对磁盘上的各种CommitLog、ConsumeQueue文件预先分配好MappedFile,也就是提前对一些
                                                                                                                                                                                                                                                                                                                                                                                                                                          可能接下来要读写的磁盘文件,提前使用MappedByteBuffer执行map()函数完成映射,这样后续读写文件的时候,就可以直接执行
                                                                                                                         垃圾回收算法的速度至少比新生代的垃圾回收算法的速度慢10倍
                                                                                                                                                                                                                                                                                                                                                                                               总结: 预映射机制 + 文件预热机制
                                                                                                                                             回收速度最慢,长的话可能要几秒甚至要几十秒,造成系统停顿
                                                                                                                                                                                                                                                                                                                                                                                                                                       在提前对一些文件完成映射之后,因为映射不会直接将数据加载到内存里来,那么后续在读取尤其是CommitLog、
                                                                                                                                                                                                                                                                                                                                                                                                                                       ConsumeQueue的时候,其实有可能会频繁的从磁盘里加载数据到内存中去。
                                                                                                                                                                                   存放方法区
                                                                                                                                                                                                                                                                                                                           MQ中的核心数据模型, Topic, 数据集合
                                                                                         把Eden区中的存活对象都标记出来,然后全部转移到Survivor1去,接着一次性清空掉Eden中的垃圾对象,Eden再次塞满的时候,就又要触发Minor GC
                                                                                         了,此时已然是垃圾回收线程运行垃圾回收器中的算法逻辑,也就是采用复制算法逻辑,去标记出来Eden和Survivor1中的存活对象,然后一次性把存活对象
                                                                                                                                                                                                                                                                                                                                                               分布式存储,创建Topic的时候指定让他里面的数据分散存储在多台Broker机器上,比如一个Topic里有1000万条数据,此时有2台
                                                                                                                                                                                                                                                                                                                           Topic作为一个数据集合是怎么在Broker集群里存储
                                                                                         转移到Survivor2中去,接着把Eden和Survivor1中的垃圾对象都回收掉
                                                                                                                                                                                              垃圾回收算法
                                                                                                                                                                                                                                                                                                                                                              Broker, 那么就可以让每台Broker上都放500万条数据
                                                                                                                                                               详细在CMS中查询
                                                                                                                                                                                                                                                                                                                                              MessageQueue就是RocketMQ中非常关键的一个数据分片机制,他通过MessageQueue将一个Topic的数据拆分为了很多
                                                                                                                                                                                                                                                                                                                Topic、MessageQueue以及Broker关系
                                                                                                                                                                                                                                                                                                                                              个数据分片,然后在每个Broker机器上都存储一些MessageQueue
                                                                                                                                           ParNew垃圾回收器针对新生代采用的就是复制算法来垃圾回收
                                                                                                                              他针对服务器一般都是多核CPU做了优化,他是支持多线程个垃圾回收的,可以大
                                                                                                                              幅度提升回收的性能,缩短回收的时间,会暂停工作线程,产生Stop the World 问题
                                                                                                                                                                                                                                                                                                                MessageQueue
                                                                                                                                                                                                                                                                                                                                自动容错机制
                                                                                                                                                                                                                                                                                                                                                如果某次访问一个Broker发现网络延迟有500ms,然后还无法访问,那么就会自动回避访问这个Broker一段时间,比如接下来3000ms内,就不会访问这个Br
                                                                                                                                                                                                                                                                                                                                                oker了,以避免一个Broker故障之后,短时间内生产者频繁的发送消息到这个故障的Broker上去,出现较多次数的异常。而是在一个Broker故障之后,自动
                                                                                                                                                                                                                                                                                                                                                回避一段时间不要访问这个Broker,过段时间再去访问他
                                                                                                                             是用一个线程进行垃圾回收,然后此时暂停系统工作线程,所以一般我们在服务器程序中很
                                                                                                                                                                                                                                                                                                                                记录消费者消费到哪一条消息,类似指针指向CommitLog
                                                                                               垃圾回收线程和系统工作线程尽量同时执行的模式来处理的。
                                                                                                                                                     标记清理算法:专门负责老年代的垃圾回收
                                                                                                                                                                                                                                                                                                                                               CommitLog是基于os cache+磁盘一起实现的
                                                                                                        这个阶段会让系统的工作线程全部停止,但是影响不大,因为速度很快,仅仅标记GC Roots直接引用的对象
                                                                                                                                                                                                                                                                                                                CommitLog
                                                                                                                                                                                                                                                                                                                                               会将老的消息数据存储在磁盘上,新的数据放到os cache中
                                                                                                                                                                                                                                                                                                                                                                                    问题:读取新的消息会特别快,内存级别的,但是如果消息过多,读取消息在磁盘中,读取就会变慢
                                                                               这个阶段会让系统线程可以随意创建各种新对象,继续运行,对老年代所有对象进行GC Roots追踪,其实是最耗时的,需要追踪所有对象是否从根源上被G
                                                                               C Roots引用了
                                                                                                                                                                                                                                                                                                                                                         所有的消息都会熟顺序存储在CommitLog中,MessageQueue中存储CommitLog的地址,ConsumeQueue中存储的是消费者消费消息所在MessagesQue
                                                                                                                                                                                                                                                                                                                MessageQueue、CommitLog、ConsumeQueue之间的关系
                                                                               因为第二阶段里,你一边标记存活对象和垃圾对象,一边系统在不停运行创建新对象,让老对象变成垃圾,所以第二阶段结束之后,绝对会有很多存活对象和
                                                                               垃圾对象,是之前第二阶段没标记出来的,所以此时进入第三阶段,要继续让系统程序停下来,然后重新标记下在第二阶段里新创建的一些对象,还有一些已
                                                                                                                                                                                                                                                                                                                如果消费组中出现机器宕机或者扩容加机器,会怎么处理
                                                                                                                                                                                                                                                                                                                                                     这个时候其实会进入一个rabalance的环节,也就是说重新给各个消费机器分配他们要处理的MessageQueue。
                                                                               有对象可能失去引用变成垃圾的情况,这个重新标记的阶段,是速度很快的,他其实就是对在第二阶段中被系统程序运行变动过的少数对象进行标记,所以运
                                                                                                                                                                                                                                                                                                                                                   Kafka性能也很高,基本上发送消息给Kafka都是毫秒级的性能。可用性也很高,Kafka是可以支持集群部署的,其中部分机器宕机是可
                                                                               这个阶段就是让系统程序随意运行,然后他来清理掉之前标记为垃圾的对象即可,这个阶段其实是很耗时的,因为需要进行对象的清理,但是他也是跟系统程
                                                                               序并发运行的,所以其实也不影响系统程序的执行
                                                                                                                                                                                                                                                                                                                                                  Kafka的吞吐量几乎是行业里最优秀的,在常规的机器配置下,一台机器
                                                                                                                                                                                                                                                                                                                                                   可以达到每秒十几万的QPS,相当的强悍
                                                                                 问题:在并发标记和并发清理两个最耗时的阶段,垃圾回收线程和系统工作线程同时工作,会导致有限的CPU资源被垃圾回收线程占用了一部分,CMS的垃
                                                                                 圾回收线程是比较耗费CPU资源的。CMS默认启动的垃圾回收线程的数量是(CPU核数 + 3) / 4
                                                                                                                                                                                                                                                                                                                                                  是Kafka比较为人诟病的一点,似乎是丢数据方面的问题
                                                                    在并发清理阶段,CMS只不过是回收之前标记好的垃圾对象,这个阶段系统一直在运行,可能会随着系统运行让一些对象进入老年代,同时还变成垃圾对象,
                                                                                                                                                                                                                                                                                                                                                  Kafka另外一个比较大的缺点,就是功能非常的单一,主要是支持发送消息给他,然后从里面消费消息,其他就没有什么额外的高
                                                                   这种垃圾对象是"浮动垃圾",是CMS只能回收之前标记出来的垃圾对象,不会回收他们,需要等到下一次GC的时候才会回收他们,所以为了保证在CMS垃
                                                                    圾回收期间,还有一定的内存空间让一些对象可以进入老年代,一般会预留一些空间。CMS垃圾回收的触发时机,其中有一个就是当老年代内存占用达到一定
                                                                                                                                                                                                                                                                                                                                                  级功能了。所以基于Kafka有限的功能,可能适用的场景并不是很多。
                                                                                                                                                                                                                                                                                                                                        因此综上所述,以及查阅了Kafka技术在各大公司里的使用,基本行业里的一个标准,是把Kafka用在用户行为日志的采集和传输上,比如大数据团队要收集A
                                                                                                                                                                                                                                                                                                                                       PP上用户的一些行为日志,这种日志就是用Kafka来收集和传输的。因为那种日志适当丢失数据是没有关系的,而且一般量特别大,要求吞吐量要高,一般就
                                                                                 "-XX:CMSInitiatingOccupancyFaction"参数可以用来设置老年代占用多少比例的时候触发CMS垃圾回收,DK 1.6里面默认的值是
                                                                                                                                                                                                                                                                                                                                        是收发消息,不需要太多的高级功能,所以Kafka是非常适合这种场景的。
                                                                                92%。老年代占用了92%空间了,就自动进行CMS垃圾回收,预留8%的空间给并发回收期间,系统程序把一些新对象放入老年代
                                                                                                                                                                                                                                                                                                                                                     RabbitMQ的优势在于可以保证数据不丢失,也能保证高可用性,即集群部署的时候部分机器宕机可以继续运行,然后支持部分高级功
                                                                                                                                                                                                                                                                                                                                                     能,比如说死信队列,消息重试之类的,这些是他的优点
                                  此时就会自动用 "Serial Old" 垃圾回收器替代CMS,就是直接强行把系统程序 "Stop the World",重新进行长时间的GC Roots追
                                                                                                              如果CMS垃圾回收期间,系统程序要放入老年代的对象大于了可用内存空间
                                 踪,标记出来全部垃圾对象,不允许新的对象产生
                                                                                                                                                                                                                                                                                                                                                     最为人诟病的,就是RabbitMQ的吞吐量是比较低的,一般就是每秒几万的级别,所以如果遇到特别特别高并发
                                                                                                                                                                                                                                                                                                                                                     的情况下,支撑起来是有点困难的。
                                                                                                                         太多的内存碎片实际上会导致更加频繁的Full GC
                                                                                                                                                                                                                                                                                                                                                    他进行集群扩展的时候(也就是加机器部署),还比较麻烦。
                                                                                                        CMS有一个参数是"-XX:+UseCMSCompactAtFullCollection",默认就打开了
                                                                                                                                                                                                                                                                                                                                                     另外还有一个较为致命的缺陷,就是他的开发语言是erlang,国内很少有精通erlang语言的工程师,因此也没办法去阅读他的源代码,
                                                                                                                                                           内存碎片问题
                                                                                                                                                                                                                                                                                           Kafka、RabbitMQ以及RocketMQ的调研对比
                                                                  他意思是在Full GC之后要再次进行"Stop the World",停止工作线程,然后进行碎片整理,就是把存活对象挪到一起,空出来大片连续内存空间,避免内
                                                                                                                                                                                                                                                                                                                                          所以现在行业里的一个情况是,很多BAT等一线互联网大厂都切换到使用更加优秀的RocketMQ了,但是很多中小型公司觉得
                                                                             还有一个参数是"-XX:CMSFullGCsBeforeCompaction",这个意思是执行多少次Full GC之后再执行一次内存碎片整理的工作,默认
                                                                                                                                                                                                                                                                                                                                          RabbitMQ基本可以满足自己的需求还在继续使用中,因为中小型公司并不需要特别高的吞吐量,RabbitMQ已经足以满足他们的需求
                                                                             是0,意思就是每次Full GC之后都会进行一次内存整理
                                                                                                                                                                                                                                                                                                                                           了,而且也不需要部署特别大规模的集群,也没必要去阅读和修改RabbitMQ的源码。
                                                                                 新生代执行速度其实很快,因为直接从GC Roots出发就追踪哪些对象是活的就行了,新生代存活对象是很少的,这个速度是极快的,
                                                                                                                                                                                                                                                                                                                                                     RocketMQ的吞吐量也同样很高,单机可以达到10万QPS以上,而且可以保证高可用性,性能很高,而且支持通过配置保证数据绝对不
                                                                                  不需要追踪多少对象,然后直接把存活对象放入Survivor中,就一次性直接回收Eden和之前使用的Survivor了
                                                                                                                                                                                                                                                                                                                                                     丢失,可以部署大规模的集群,还支持各种高级的功能,比如说延迟消息、事务消息、消息回溯、死信队列、消息积压,等等。
                                                                     在并发标记阶段,他需要去追踪所有存活对象,老年代存活对象很多,这个过程就会很慢,其次并发清理阶段,他不是一次性回收一大片内存,而是找到零零
                                                                                                                                                              Full GC比Minor GC慢10倍
                                                                     散散在各个地方的垃圾对象,速度也很慢,最后完事儿了,还得执行一次内存碎片整理,把大量的存活对象给挪在一起,空出来连续内存空间,这个过程还得
                                                                                                                                                                                                                                                                                                                                                     而且RocketMQ是基于Java开发的,符合国内大多数公司的技术栈,很容易就可以阅读他的源码,甚至是修改他的源码。
                                                                                                                                                                                                                          JVM内存空间
                                                                      "Stop the World",那就更慢了。万一并发清理期间,剩余内存空间不足以存放要进入老年代的对象了,引发了"Concurrent Mode Failure"问题,那更
                                                                     是麻烦,还得立马用"Serial Old"垃圾回收器,"Stop the World"之后慢慢重新来一遍回收的过程,这更是耗时了
                                                                                                                                                                                                                                                                                                                                                     高吞吐量, 大规模集群部署能力, 以及各种高阶
                                                                                                                                                                                                                                                                                                                                                     的功能去支撑自己的各种业务场景,同时还可以根据自己的需求定制修改RocketMQ的源码。
                                                                                                              G1垃圾回收器是可以同时回收新生代和老年代的对象,最大的一个特点,就是把Java堆内存拆分为多个大小相等的Region
                                                                                                                                                                                                                                                                                                                                                     RocketMQ是非常适合用在Java业务系统架构中的,因为他很高的性能表现,还有他的高阶功能的支持,可以让我们解决各种业务问
                                                                                                                                                                                               垃圾回收器
                                                        G1通过追踪发现,1个Region中的垃圾对象有10MB,回收他们需要耗费1秒钟,另外一个Region中的垃圾对象有20MB,回收他们需要耗费200毫秒,然后
                                                        在垃圾回收的时候,G1会发现在最近一个时间段内,比如1小时内,垃圾回收已经导致了几百毫秒的系统停顿了,现在又要执行一次垃圾回收,那么必须是回
                                                                                                                                                 最大的特点就是可以设置一个垃圾回收的预期停顿时间
                                                        收上图中那个只需要200ms就能回收掉20MB垃圾的Region啊!
                                                                                                                                                                                                                                                                                                                                                     RocketMQ的大优势: 可视化的管理界面
                                                                                         所以简单来说,G1可以做到让你来设定垃圾回收对系统的影响,他自己通过把内存拆分为大量小Region,以及追踪每个Region中可以
                                                                                                                                                                                                                                                                                                                                                     当然,RocketMQ也有一点美中不足的地方,就是经过我的调查发现,RocketMQ的官方文档相对简单一些,但是Kafka和RabbitMQ
                                                                                         回收的对象大小和预估时间,最后在垃圾回收的时候,尽量把垃圾回收对系统造成的影响控制在你指定的时间范围内,同时在有限的时
                                                     最少回收时间和最多回收对象的Region进行垃圾回收
                                                                                                                                                                                                                                                                                                                                                     的官方文档就非常的全面和详细,这可能是RocketMQ目前唯一的缺点。
                                                                                                                                                                      核心设计思路
                                                                                                                                                                                                                                                                                                                                          RocketMQ是阿里开源的消息中间件,久经沙场,非常的靠谱。他几乎同时解决了Kafka和RabbitMQ的缺陷。
                                                                                                            Region可能属于新生代也可能属于老年代,所以没有新生代给多少内存老年代给多少内存这一说
                                                                                                                                                                                                                                                                                                                                              "vm.overcommit_memory" 这个参数有三个值可以选择, 0、1、2。
                                                                      因为JVM最多可以有2048个Region,然后Region的大小必须是2的倍数,比如说1MB、2MB、4MB之类的,比如说堆大小是4G,那么就是4096MB,此时
                                                                                                                                                              如何设定G1对应的内存大小
                                                                     除以2048个Region,每个Region的大小就是2MB。大概就是这样子来决定Region的数量和大小的,大家一般保持默认的计算方式就可以
                                                                                                                                                                                                                                                                                                                                              如果值是0的话,在你的中间件系统申请内存的时候,os内核会检查可用内存是否足够,如果足够的话就分配内存给你,如果感觉剩余
                                                                                                                                                                                                                                                                                                                                              内存不是太够了,干脆就拒绝你的申请,导致你申请内存失败,进而导致中间件系统异常出错。
                                                                  个新生代的参数,"-XX:SurvivorRatio=8",比如新生代之前说刚开始初始的时候,有100个Region,那么可能80个Region就是Eden,两个Survivor各自
                                                                                                                                                           新生代还有Eden和Survivor的概念
                                                                                                                                                                                                                                                                                                                       vm.overcommit memory
                                                                                                                                                                                                                                                                                                                                              因此一般需要将这个参数的值调整为1,意思是把所有可用的物理内存都允许分配给你,只要有内存就给你来用,这样可以避免申请内
                                                                         随着不停的在新生代的Eden对应的Region中放对象,JVM就会不停的给新生代加入更多的Region,直到新生代占据堆大小的最大比例60%。而且Eden区还
                                                                         占满了对象,这个时候还是会触发新生代的GC,G1就会用之前说过的复制算法来进行垃圾回收,进入一个"Stop the World"状态,然后把Eden对应的Reg
                                                                                                                                                                  G1的新生代垃圾回收
                                                                                                                                                                                                                                                                                                                                              比如我们曾经线上环境部署的Redis就因为这个参数是0,导致在save数据快照到磁盘文件的时候,需要申请大内存的时候被拒绝了,进
                                                                         ion中的存活对象放入S1对应的Region中,接着回收掉Eden对应的Region中的垃圾对象,但是这个过程跟之前是有区别的,因为G1是可以设定目标GC停顿
                                                                         时间的,也就是G1执行GC的时候最多可以让系统停顿多长时间,可以通过"-XX:MaxGCPauseMills"参数来设定,默认值是200ms。
                                                                                                                                                                                                                                                                                                                                              可以用如下命令修改: echo 'vm.overcommit_memory=1' >> /etc/sysctl.conf。
                                                                                     对象在新生代躲过了很多次的垃圾回收,达到了一定的年龄了,"-XX:MaxTenuringThreshold"参数可以设置这个年龄,他就
                                                                                                                                                                                                                                                                                                                                           这个参数的值会影响中间件系统可以开启的线程的数量,同样也是非常重要的
                                                                                                                                                              对象什么时候进入老年代?
                                                                                                                                                                                                                                                                                                                                          如果这个参数过小,有的时候可能会导致有些中间件无法开启足够的线程,进而导致报错,甚至中间件系统挂掉。
                                                                                                          动态年龄判定规则,如果一旦发现某次新生代GC过后,存活对象超过了Survivor的50%
                                                                                                                                                                                                                                                                                                                                          他的默认值是65536,但是这个值有时候是不够的,比如我们大数据团队的生产环境部署的Kafka集群曾经有一次就报出过这个异常,
                                                                                                                                                                                                                                                                                                                                           说无法开启足够多的线程,直接导致Kafka宕机了。
                                                                                        G1提供了专门的Region来存放大对象,而不是让大对象进入老年代的Region中,一个大对象如果太大,可能会横跨多个Region来存放
                                                                                                                                                                                                                                                                                                                                           因此建议可以把这个参数调大10倍,比如655360这样的值,保证中间件可以开启足够多的线程。
                                                                                            新生代、老年代在回收的时候,会顺带带着大对象Region一起回收,所以这就是在G1内存模型下对大对象的分配和回收的策略
                                                                                                                                                                                                                                                                                                                                           可以用如下命令修改: echo 'vm.max_map_count=655360' >> /etc/sysctl.conf。
                                                                  G1有一个参数,是"-XX:InitiatingHeapOccupancyPercent",他的默认值是45%,果老年代占据了堆内存的45%的Region的时候,此时就会尝试触发一
                                                                                                                                                                                                                                                                                                                                        这个参数是用来控制进程的swap行为的,这个简单来说就是os会把一部分磁盘空间作为swap区域,然后如果有的进程现在可能不是太
                                                                                                                                                          触发新生代+老年代的混合垃圾回收
                                                                  个新生代+老年代一起回收的混合回收阶段
                                                                                                                                                                                                                                                                                                                                        活跃,就会被操作系统把进程调整为睡眠状态,把进程中的数据放入磁盘上的swap区域,然后让这个进程把原来占用的内存空间腾出
                                                                                                                                                                                                                                                                                                                                        来,交给其他活跃运行的进程来使用。
                                                                                                        进入 "Stop the World"的,仅仅只是标记一下GC Roots直接能引用的对象,
                                                                                                        这个过程速度是很快的
                                                                                                                                                                                                                                                                                                                                        如果这个参数的值设置为0,意思就是尽量别把任何一个进程放到磁盘swap区域去,尽量大家都用物理内存。
                                                                        这个阶段会允许系统程序的运行,同时进行GC Roots追踪,从GC Roots开始追踪所有的存活对象,耗时,但是因为是并发执行,所以问题不大
                                                                                                                                                                                                                                                                                                                                        如果这个参数的值是100,那么意思就是尽量把一些进程给放到磁盘swap区域去,内存腾出来给活跃的进程使用。
                                                                这个阶段会进入"Stop the World",系统程序是禁止运行的,但是会根据并发标记阶段记录的那些对象修改,最终标记一下有哪些存活对象,有哪些是垃
                                                                                                                                                                      垃圾回收过程
                                                                                                                                                                                                                                                                                                                                        默认这个参数的值是60,有点偏高了,可能会导致我们的中间件运行不活跃的时候被迫腾出内存空间然后放磁盘swap区域去。
                                                                                                                                                                                                                                                                                                                                        因此通常在生产环境建议把这个参数调整小一些,比如设置为10,尽量用物理内存,别放磁盘swap区域去。
                                                                                       这个阶段会计算老年代中每个Region中的存活对象数量,存活对象的占比,还有执行垃圾回收的预期性能和效率
                                                                                                                                                                                                                                                                                                                                        可以用如下命令修改: echo 'vm.swappiness=10' >> /etc/sysctl.conf。
                                                                    接着会停止系统程序,然后全力以赴尽快进行垃圾回收,此时会选择部分Region进行回收,因为必须让垃圾回收的停顿时间控制在我们指定的范围内
                                                                                                                                                                                                                                                                                                                                   这个是用来控制linux上的最大文件链接数的,默认值可能是1024,一般肯定是不够的,因为你在大量频繁的读写磁盘文件的时候,或
                                                                                                                                                                                                                                                                                                                                   者是进行网络通信的时候,都会跟这个参数有关系
                                                                                                          还有一个参数,"-XX:G1MixedGCLiveThresholdPercent",他的默认值是85%,意思就是确定要回收的Region的时候,必须是存
                                                                                                         活对象低于85%的Region才可以进行回收
                                                                                                                                                                                                                                                                                                                                   对于一个中间件系统而言肯定是不能使用默认值的,如果你采用默认值,很可能在线上会出现如下错误:error: too many open files。
                                                                                                                                                                                                                                                                                                                                   因此通常建议用如下命令修改这个值: echo 'ulimit -n 1000000' >> /etc/profile。
                                                                          如果在进行Mixed回收的时候,无论是年轻代还是老年代都基于复制算法进行回收,都要把各个Region的存活对象拷贝到别的Region里去,此时万一出现拷
                                                                          贝的过程中发现没有空闲Region可以承载自己的存活对象了,就会触发一次失败,一旦失败,立马就会切换为停止系统程序,然后采用单线程进行标记、清
                                                                                                                                                                   回收失败的Full GC
                                                                                                                                                                                                                                                                                                                         内存区域的大小分配,垃圾回收器以及对应的行为参数,GC日志存放地址,OOM自动导出内存快照的配置
                                                                          理和压缩整理,空闲出来一批Region,这个过程是极慢极慢的
                                                                                                                                                                                                                                                                                           部署注意
                                                                                                                                                                                                                                                                                                                         -server:这个参数就是说用服务器模式启动,这个没什么可说的,现在一般都是如此
                                                                                                                                  G1回收掉300个Region 600MB内存,大致需要200ms
                                                                                                                                                                                                                                                                                                                         -Xms8g -Xmx8g -Xmn4g:这个就是很关键的一块参数了,也是重点需要调整的,就是默认的堆大小是8g内存,新生代是4g内存,
                                                                             随着系统运行,每秒创建3MB的对象,大概1分钟左右就会塞满100个Region(200MB内存),此时很可能G1会觉得,要是我现在就触发一次新生代gc,那
                                                                                                                                                                                                                                                                                                                         但是我们的高配物理机是48g内存的
                                                                             么回收区区200MB只需要大概几十ms,最多就让系统停顿几十ms而已,跟我的主人设定的"-XX:MaxGCPauseMills"参数限制的200ms停顿时间相差甚
                                                                                                                                                                                                                                                                                                                        所以这里完全可以给他们翻几倍,比如给堆内存20g,其中新生代给10g,甚至可以更多一些,当然要留一些内存给操作系统来用
                                                                             远,还不如给新生代先增加一些Region,然后让系统继续运行着在新生代Region中分配对象好了,这样就不用过于频繁的触发新生代gc了,然后系统继续运
                                                                              行,一直到可能300个Region都占满了,此时通过计算发现回收这300个Region大概需要200ms,那么可能这个时候就会触发一次新生代gc了
                                                                                                                                                                                                                                                                                                                         -XX:+UseG1GC -XX:G1HeapRegionSize=16m:这几个参数也是至关重要的,这是选用了G1垃圾回收器来做分代回收,对新生代
                                                                                                                                                                                                                                                                                                                        和老年代都是用G1来回收
                                                                                                                                                                                         -Xms: java堆内存大小
                                                                                                                                                                                                                                                                                                                         这里把G1的region大小设置为了16m,这个因为机器内存比较多,所以region大小可以调大一些给到16m,不然用2m的region,会
                                                                                                                                                                                                                                                                                                                        导致region数量过多的
                                                                                                                                                                                     -Xmx: java堆内存的最大大小
                                                                                                                                                                                                                                                                                                                         -XX:G1ReservePercent=25:这个参数是说,在G1管理的老年代里预留25%的空闲内存,保证新生代对象晋升到老年代的时候有足
                                                                                                                                                                                                                                                                                                                        够空间,避免老年代内存都满了,新生代有对象要进入老年代没有充足内存了,默认值是10%,略微偏少,这里RocketMQ给调大了一些
                                                                                                                                                            -Xmn: java堆内存中的新生代大小, 扣除新生代剩下的就是老年代的内存大小
                                                                                                                                                                                                                                                                                                                         -XX:InitiatingHeapOccupancyPercent=30:这个参数是说,当堆内存的使用率达到30%之后就会自动启动G1的并发垃圾回收,开
                                                                                                                                                                              -XX:SurvivorRatio=8 Eden占新生代的8/10
                                                                                                                                                                                                                                                                                                                         始尝试回收一些垃圾对象
                                                                                                                                                                                                                                                                                                                        默认值是45%,这里调低了一些,也就是提高了GC的频率,但是避免了垃圾对象过多,一次垃圾回收耗时过长的问题
                                                                                                                                                                                      -XX:PermSize: 永久代大小
                                                                                                                                                                                                                                                                                                        JVM各种参数
                                                                                                                                                                                                                                                                                                                         -XX:SoftRefLRUPolicyMSPerMB=0:这个参数默认设置为0了,在JVM优化专栏中,救火队队长讲过这个参数引发的案例,其实建
                                                                                                                                                                                 -XX:MaxPermSize: 永久代最大大小
                                                                                                                                                                                                                                                                                                                         议这个参数不要设置为0,避免频繁回收一些软引用的Class对象,这里可以调整为比如1000
                                                                                                                                                                                     -Xss:每个线程的栈内存大小
                                                                                                                                                                                                                                                                                                                         -verbose:gc -Xloggc:/dev/shm/mq_gc_%p.log -XX:+PrintGCDetails -XX:+PrintGCDateStamps -
                                                                                                                                                                                                                                                                                                                        XX:+PrintGCApplicationStoppedTime -XX:+PrintAdaptiveSizePolicy -XX:+UseGCLogFileRotation -
                                                                                                                                                                                                                                                                                                                         XX:NumberOfGCLogFiles=5 -XX:GCLogFileSize=30m: 这一堆参数都是控制GC日志打印输出的,确定了gc日志文件的地址,要
                                                                                                                                                                      -XX:MaxTenuringThreshold 参数可以设置这个年龄,他就
                                                                                                                                                                                                                                                                                                                        打印哪些详细信息,然后控制每个gc日志文件的大小是30m,最多保留5个gc日志文件。
                                                                                                                                                                      会进入老年代
                                                                                                                                                                                                                                                                                                                         -XX:-OmitStackTraceInFastThrow:这个参数是说,有时候JVM会抛弃一些异常堆栈信息,因此这个参数设置之后,就是禁用这个
                                                                                                                                                                  -XX:PretenureSizeThreshold:对象大小大于这个值之间进入老年代
                                                                                                                                                                                                                                                                                                                        特性,要把完整的异常堆栈信息打印出来
                                                                                                                                                                        -XX:ParallelGCThreads 调节ParNew垃圾回收线程数量
                                                                                                                                                                                                                                                                                                                         -XX:+AlwaysPreTouch:这个参数的意思是我们刚开始指定JVM用多少内存,不会真正分配给他,会在实际需要使用的时候再分配给
                                                                                                                                                                                                                                                                                                                        他,所以使用这个参数之后,就是强制让JVM启动的时候直接分配我们指定的内存,不要等到使用内存的时候再分配
                                                                                                                                              "-XX:CMSInitiatingOccupancyFaction"参数可以用来设置老年代占用多少比例的时候触发CMS垃圾回收
                                                                                                                                                                                                                                                                                                                         -XX:MaxDirectMemorySize=15g: 这是说RocketMQ里大量用了NIO中的direct buffer,这里限定了direct buffer最多申请多少,
                                                                                                                                                     -XX:CMSFullGCsBeforeCompaction 执行多少次Full GC之后再执行一次内存碎片整理的工作
                                                                                                                                                                                                                                                                                                                         如果你机器内存比较大,可以适当调大这个值,如果有朋友不了解direct buffer是什么,可以自己查阅一些资料。
                                                                                                                                                                     -XX:+UseG1GC 指定使用G1垃圾回收器,此时会自动用堆大
                                                                                                                                                                                                                                                                                                                         -XX:-UseLargePages -XX:-UseBiasedLocking: 这两个参数的意思是禁用大内存页和偏向锁,这两个参数对应的概念每个要说清楚
                                                                                                                                                                                                                                                                                                                        都得一篇文章, 所以这里大家直接知道人家禁用了两个特性即可。
                                                                                                                                                                                                                                                                                                                        最后我们做一点小的总结,RocketMQ默认的JVM参数是采用了G1垃圾回收器,默认堆内存大小是8G
                                                                                                                                                                          -XX:G1HeapRegionSize: 指定G1 Region的大小
                                                                                                                                                                                                                                                                                                                                  比如你的中间件系统会开启很多线程处理请求和工作负载,然后还会进行大量的网络通信,同时会进行大量的磁盘IO类的操作。比如你的机器配置很高,是2
                                                                                                                                                                                                                                                                                                         中间件系统本身的一些核心参数
                                                                                                                                                                                                                                                                                                                                4核CPU,结果你的中间件系统默认就开启了4个工作线程去处理请求,这不是在开玩笑么!相当于24核CPU里很多都是空闲状态,是没有任何事情可以干
                                                                                                                                                                           -XX:G1NewSizePercent"来设置新生代初始占比
                                                                                                                                                                          -XX:G1MaxNewSizePercent 设置新生代最大占比
                                                                                                                                                                                                                                                                                                         往往必须要对os内核参数、jvm参数以及自身核心参数都做出相对应的合理的调整,再进行压测和上线。
                                                                                                                                                                             -XX:MaxGCPauseMills G1目标GC停顿时间的
                                                                                                                                                                                                                                                                                                                      执行完发送消息后不等待,直接执行下面的逻辑,MQ返回结果的时候,Producer会回调SendCallback里的函数,如果发送成功了就回调onSuccess函数,
                                                                                                                                                                                                                                                                                                                      如果发送失败了就回调onExceptino函数。
                                                                                                                                                          -XX:InitiatingHeapOccupancyPercent G1老年代占据多少比例内存触发混合回收
                                                                                                                                                                                                                                                                                                                     所谓同步,意思就是你通过这行代码发送消息到MQ去,SendResult sendResult = producer.send(msg),然后你会卡在这里,代码
                                                                                                                                              -XX:G1MixedGCLiveThresholdPercent 默认值是85%,意思就是确定要回收的Region的时候,必须是存
                                                                                                                                                                                                                                                                                                                      不能往下走了你要一直等待MQ返回一个结果给你,你拿到了SendResult之后,接着你的代码才会继续往下走
                                                                                                                                              活对象低于85%的Region才可以进行回收
                                                                                                                                                                                                                                                                                                                      这个sendOneway的意思,就是你发送一个消息给MQ,然后代码就往下走了,根本不会关注MQ有没有返回结果给你,你也不需要
                                                                                                                                                                                                                                                                                                                      MQ返回的结果,无论发送的消息是成功还是失败,都不关你的事
                                                                                                                                              一个对象大概几百字节,可以按照一个500字节,较大情况
                                                                                                                                                                                    Integer 4字节Long8字节
                                                                                                                                                                                                         对象大小计算
                                                                                                                                                                                                                                                                                                                         Broker会主动把消息发送给你的消费者,你的消费者是被动的接收Broker推送给过来的消息,然后进行处理
                                                                                                                              每个对象,都分析一下有谁在引用他,然后一层一层往上去判断,看是否有一个GC Roots。
                                                                                                                                                                                                      判断对象是否可回收
                                                                                                                                                                                                                                                                                                                       Broker不会主动推送消息给Consumer, 而是消费者主动发送请求到Broker去拉取消息过来
                                                                                                         回收的环节了,假设没有GC Roots引用的对象,先尝试调用一下他的finalize()方法,看是否把自己这个实例对象给了某个GC Roots变量
                                                                                                                                                                                                                                                                                                                                                没拉取的消息超过了最大能使用的内存的量,那么说明你后续会频繁从磁盘加载数据,此时就让你从slave broker去加载数据了
                                                                                                                                                                                                                                                                                                         到底什么时候Master Broker会让你从Slave Broker拉取数据
                                                                                                                                                                          虚拟机栈(栈帧中的本地变量表)中的引用的对象
                                                                                                                                                                                                                                                                                                                                                       可能因为系统处理速度慢,导致重复调用,重复推送两次消息到MQ中
                                                                                                                                                                                   方法区中类静态属性引用的对象
                                                                                                                                                                                                                                                                                                                                   重复发送, 有多条消息
                                                                                                                                                                                                                                                                                                                                                       发送了一条消息到MQ了,其实MQ是已经接收到这条消息了,结果MQ返回响应给你的时候,网络有问
                                                                                                                                                                                       方法区中常量引用的对象
                                                                                                                                                                                                                                                                                                                                                       题超时了,就是你没能及时收到MQ返回给你的响应,然后重试发送
                                                                                                                                                                       本地方法栈中JNI (一般说的Native方法) 的引用的对象
                                                                                                                                                                                                                                                                                                                                    一条消息重复消费
                                                                                                                                                                                                                                                                                                                                                     当消息消费完后,还没来得及提交消息offset到broker,系统进行了重启,然后又重新消费了一次
                                                                                                                                                                  只要是强引用的类型,那么垃圾回收的时候绝对不会去回收这个对象的。
                                                                                                                                                                                                                                                                                                                                                   会创建一个重试队列,在1s 5s 10s 30s 1m 2m 3m 4m 5m...... 会进行重试,重试16次如果还失败就会放到死信队列中
                                                                                                                                                                                                                                                                                                                     可以返回一个状态RECONSUME LATER
                                                                                                                                                                    软引用就是说有的对象可有可无,如果内存实在不够了,可以回收他。
                                                                                                                                                                                                               软引用
                                                                                                                                                                                                                                                                                                                                    每个Topic指定多个MessageQueue,然后你写入消息的时候,其实是会把消息均匀分发给
                                                                                                                                                                                                                                                                                           消费消息
                                                                                                                                                                                                                                                                                                                                                                                           让属于同一个订单根据订单号进入一个MessageQueue
                                                                                                                                                                                                                                                                                                                                    不同的MessageQueue的。每个MessageQueue消费速度不一样
                                                                                                                                                                  弱引用就跟没引用是类似的,如果发生垃圾回收,就会把这个对象回收掉
                                                                                                                                                                                                               弱引用
                                                                                                                                                                                                                                                                                                          消息乱序问题
                                                                                                                                                                                                                                                                                                                                                                                          如果消费失败的话,就返回SUSPEND_CURRENT_QUEUE_A_MOMENT状态,意思是先等一会儿,一会儿再继续处理这批消息,而不能把这批消息放入重试
                                                                                                                                                                                                                                                                                                                                    在同一个MessageQueue中,但是前面消息消费失败了,进入了重试队列,同样也会乱序
                                                                                                                                                         一个对象仅持有虚引用,那么它就和没有任何引用一样,在任何时候都可能被垃圾回收器回收
                                                                                                                                                                                                                                                                                                                                                                                          队列去, 然后直接处理下一批消息
                                                                                                                                                                  SOC: 这是From Survivor区的大小
                                                                                                                                                                                                                                                                                                                     在发送消息的时候,给消息设置tag和属性,在消费数据的时候根据tag和属性进行过滤
                                                                                                                                                                  S1C: 这是To Survivor区的大小
                                                                                                                                                                  SOU: 这是From Survivor区当前使用的内存大小
                                                                                                                                                                  S1U: 这是To Survivor区当前使用的内存大小
                                                                                                                                                                  EC: 这是Eden区的大小
                                                                                                                                                                  EU: 这是Eden区当前使用的内存大小
                                                                                                                                                                                                                                                                                                                         首先需要在broker的配置文件里开启traceTopicEnable=true这个选项,此时就会开启消息轨迹追踪的功能
                                                                                                                                                                  OC:这是老年代的大小
                                                                                                                                                                  OU: 这是老年代当前使用的内存大小
                                                                                                                                                                                                                                                                                                                         接着当我们开启了上述的选项之后,我们启动这个Broker的时候会自动创建出来一个内部的Topic,就是RMQ_SYS_TRACE_TOPIC,
                                                                                                                                                                  MC: 这是方法区 (永久代、元数据区) 的大小
                                                                                                                                                                                                                                                                                                         消息轨迹的追踪
                                                                                                                                                                                                                                                                                                                         这个Topic就是用来存储所有的消息轨迹追踪的数据的。
                                                                                                                                                                  MU:这是方法区(永久代、元数据区)的当前使用的内存大小
                                                                                                                                                                  YGC: 这是系统运行迄今为止的Young GC次数
                                                                                                                                                                                                                                                                                                                         接着做好上述这一切事情之后,我们需要在发送消息的时候开启消息轨迹,此时创建Producer的时候要用如下的方式,下面构造函数
                                                                                                                                                                  YGCT: 这是Young GC的耗时
                                                                                                                                                                                                                                                                                                                         中的第二个参数,就是enableMsgTrace参数,他设置为true,就是说可以对消息开启轨迹追踪
                                                                                                                                                                  FGC: 这是系统运行迄今为止的Full GC次数
                                                                                                                                                                  FGCT:这是Full GC的耗时
                                                                                                                                                                  GCT: 这是所有GC的总耗时
                                                                                                                                                                                                                                                                                                          不同的系统应该设置不同的消费组,如果不同的消费组订阅了同一个Topic,对Topic里的一条消息,每
                                                                                                                                                                                                                           GC日志查看
                                                                                                                                                                                                                                                                                                          个消费组都会获取到这条消息,在每个消费者组内只会消费一次
                                                                                                                                            jstat -gccapacity PID:堆内存分析
                                                                                                                                            jstat-gcnew PID: 年轻代GC分析,这里的TT和MTT可以看到对象在年轻代存活的年龄和存活的最大年龄
                                                                                                                                                                                                                                                                                                                       默认情况下我们都是集群模式,也就是说,一个消费组获取到一条消息,只会交给组内的一台机器去处理,不是每台机器都可以获取到
                                                                                                                                            jstat -gcnewcapacity PID:年轻代内存分析
                                                                                                                                           jstat -gcold PID: 老年代GC分析
                                                                                                                                            jstat -gcoldcapacity PID:老年代内存分析
                                                                                                                                                                                                                                                                                                                        广播模式,那么对于消费组获取到的一条消息,组内每台机器都可以获取到这条消息。但是相对而言广播模式其实用的很
                                                                                                                                            jstat -gcmetacapacity PID:元数据区内存分析
                                                                                                                                                                                                                                                                                                                         少, 常见基本上都是使用集群模式来进行消费的。
                                                                                                                                       会打印出来堆内存相关的一些参数设置,然后就是当前堆内存里的一些基本各个区域的情况
                                                                                                                                                                                             jmap -heap PID
                                                                                                                                                                                                                                                                                                                                   在Broker中有一个Reactor线程,专门监听端口建立连接
                                                                                                                                                                                             jmap -histo PID
                                                                                                                                                                按照各种对象占用内存空间的大小降序排列
                                                                                                                                                                                                                                                                                                          Producer和Broker建立一个长连接
                                                                                                                                                                                                                                                                                                                                   Producer中有一个SocketChannel与Broker的SocketChannel创建一个长连接
    接着你就在浏览器上访问当前这台机器的7000端口号,就可以通过图形化的方式去分析堆内存里的对象分布情况了。
                                                                         使用jhat在浏览器中分析堆转出快照 jhat dump.hprof -port 7000
                                                                                                                   用jmap命令生成一个堆内存快照放到一个文件里去,会在当前目录下生成一个dump.hrpof文件
                                                                                                                                                                           jmap -dump:live,format=b,file=dump.hprof PID
                                                                                                                                                                                                                                                                                                                                    由Reactor主线程建立连接SocketChannel然后放到Reactor线程池中,然后去监听这个SocketChannel中的请求
                                                                                                                                                                                                                                                                                           创建连接
                                                                                                                                                                                                                                                                                                                                      接着Reactor线程从SocketChannel中读取出来一个请求,这个请求在正式进行处理之前,必须就先要进行一些准备工作和预处理,比如SSL加密验证、编码
                                                                                                                                                                                                                                                                                                                                      解码、连接空闲检查、网络连接管理,诸如此类的一些事
                                                                                                                                                                                                                                                                                                         基于Worker线程池完成一系列准备工作
                                                                                                                                                                                                                                                                                                                                      这个时候需要引入一个新的概念,叫做Worker线程池,他默认有8个线程,此时Reactor线程收到的这个请求会交给Worker线程池中的一个线程进行处理,会
                                                                                                                                                                                                                                                                                                                                     完成上述一系列的准备工作
                                                                                                                                                                                                                                                                                                         基于业务线程池完成请求的处理
                                                                                                                                                                                                                                                                                                                                 对于处理发送消息请求而言,就会把请求转交给SendMessage线程池
                                                                                                                                                                                                                                                                                                                      在系统给MQ推送的时候可能遇到网络故障等原因导致他推送消息失败
                                                                                                                                                                                                                                                                                                                       使用重试机制发送消息到MQ中问题:当完成订单本地事务后,消息还没有发送到MQ中,系统突然宕机了,造成消息丢失
                                                                                                                                                                                                                                                                                                                                                                                                           原理:发送的这个half消息是写入到自己内部的"RMQ_SYS_TRANS_HALF_TOPIC"这个Topic对应的一个ConsumeQueue里去,当接受到commit后,才
                                                                                                                                                                                                                                                                                                                                                                                                           会放入到应该放入的topic中
                                                                                                                                                                                                                                                                                                                                                                                                            因为RocketMQ都是顺序把消息写入磁盘文件的,所以在这里如果你执行rollback,他的本质就是用一个OP操作来标记half消息的状态,RocketMQ内部有一
                                                                                                                                                                                                                                                                                                                            1.首先发送half消息到MQ中去,试探MQ是否正常,正常的话就会继续下一步操作,不正常就会执行回滚,消息发送成功后对消费者不可见
                                                                                                                                                                                                                                                                                                                                                                                                            个OP_TOPIC,此时可以写一条rollback OP记录到这个Topic里,标记某个half消息是rollback了
                                                                                                                                                                                                                                                                                                                                                                                                           假设你一直没有执行commit/rollback, RocketMQ会回调订单系统的接口去判断half消息的状态, 但是他最多就
                                                                                                                                                                                                                                                                                                         发送消息到MQ流程
                                                                                                                                                                                                                                                                                                                                                                                                           是回调15次,如果15次之后你都没法告知他half消息的状态,就自动把消息标记为rollback
                                                                                                                                                                                                                                                                                                                            2.MQ正常会返回half消息成功的响应,继续执行后面的逻辑
                                                                                                                                                                                                                                                                                                                            3.执行rollback or commit 操作,rollback:让MQ中的消息回滚消失,commit:提交,让消息对消费者可见
                                                                                                                                                                                                                                                                                                                            4.回调接口判断消息状态,这个会进行补偿回调,是MQ对系统进行回调,去判断是rollback还是commit
                                                                                                                                                                                                                                                                                                                                写入MQ中大概率是仅仅写入到os cache中,一旦机器宕机,内存中的数据都会丢失,哪怕在磁盘中的数据,如果磁盘坏了,消息也会丢失
                                                                                                                                                                                                                                                                                                                                              把消息直接写入磁盘文件中,这样机器宕机数据不会丢失,但是吞吐量会降低
                                                                                                                                                                                                                                                                                                                                             如果一定要确保数据零丢失的话,可以调整MQ的刷盘策略,我们需要调整broker的配置文件,将其中的flushDiskType配置设置
                                                                                                                                                                                                                                                                                                                                             为: SYNC FLUSH, 默认他的值是ASYNC FLUSH, 即默认是异步刷盘的
                                                                                                                                                                                                                                                                                                         发送消息到MQ中也可能丢失
                                                                                                                                                                                                                                                                                                                                            会将消息保存到os cache中,吞吐量增加,但是机器宕机数据会丢失
                                                                                                                                                                                                                                                                                                                                                           让一个Master Broker有一个Slave Broker去同步他的数据,而且你一条消息写入成功,必须是让Slave Broker也写入
                                                                                                                                                                                                                                                                                                                                                           成功,保证数据有多个副本的冗余。
                                                                                                                                                                                                                                                                                                                                通过主从架构磁盘故障导致数据丢失
                                                                                                                                                                                                                                                                                                                                                           基于DLedger技术和Raft协议的主从同步架构
                                                                                                                                                                                                                                                                                                                            系统已经拿到了这条消息,但是消息目前还在他的内存里,还没执行派发红包的逻辑,此时他就直接提交了这条消息的offset到broker去说自己已经处理过
                                                                                                                                                                                                                                                                                                                             了,接着红包系统在上图这个状态的时候就直接崩溃了,内存里的消息就没了,红包也没派发出去,结果Broker已经收到他提交的消息offset了,还以为他已
```

经处理完这条消息了, 等红包系统重启的时候, 就不会再次消费这条消息了。

者获取到一批消息之后,就会回调你的这个监听器函数,让你来处理这一批消息

事务消息

消费者导致消息丢失

RocketMQ的消费者中会注册一个监听器,就是上面小块代码中的MessageListenerConcurrently这个东西,当你的消费

然后当你处理完毕之后,你才会返ConsumeConcurrentlyStatus.CONSUME_SUCCESS作为消费成功的示意,告诉RocketMQ,这批

对一批消息都没提交他的offset给broker的话,broker不会认为你已经处理完了这批消息,此时你突然红包系统的一台机器宕机了,他其实会感知到你的红包 系统的一台机器作为一个Consumer挂了,接着他会把你没处理完的那批消息交给红包系统的其他机器去进行处理,所以在这种情况下,消息也绝对是不会丢

方案一 (同步发送消息 + 反复多次重试)

消息系统所传输信息的物理载体,生产和消费数据的最小单位,每条消息必须属于一个主题Topic。RocketMQ中每个消息拥有唯一的Message ID,且可以携带具

NameServer是集群里非常关键的一个角色,他要管理Broker信息,别人都要通过他才知道跟哪个Broker通信

Broker会每隔30s给所有的NameServer发送心跳,告诉每个NameServer自己目前还活着,每次NameServer收到一个Broker的心跳,就可以更新一下他的

有业务标识的Key。系统提供了通过Message ID和Key查询消息的功能。

集群化部署,高可用

每个Broker启动都得向所有的NameServer进行注册

生产者和消费者自己主动去NameServer拉取Broker信息的

最近一次心跳的时间

消息 (Message)

NameServer

发送消息到MQ的零丢失

方案二(事务消息机制),两者都有保证消息发送零丢失的效果,但是经过分析,事务消息方案整体会更好一些

对全链路消息零丢失方案进行总结

MQ收到消息之后的零丢失

开启同步剧盘策略 + 主从架构同步机制,只要让一个Broker收到消息之后同步写入磁盘,同时同步复制
给其他Broker,然后再返回响应给生产者说写入成功,此时就可以保证例自己不会弄妄消息

采用RocketMQ的消费者天然就可以保证你处理完消息之后,才会提交消息的offset到broker去,只要记住别采用
多线照升步处理消息的方式即可

如果在系统中落地一套消息零丢失方案,不管是哪个系统,不管是哪个场景,都可以确保消息流转的过程中不会丢失,看起来似乎很有
吸引力,这也是消息零丢失方案的优势所在,可以让系统的数据都是正确的,不会有丢失的。

显而易见的是,你用了这套方案之后,会让你整个从头到尾的消息流转镀路的性能大幅废下降,让你的MQ的吞吐量大幅废的下降比如本身你的系统和MQ配合起来,每秒可以处理几万条消息的,结果当你落地消息零丢失方案之后,可能每秒只能处理几千条消息了。

如果我们仅仅只是简单的把消息发送到MQ,那么不过就是一次普通的网络请求罢了,我们就
是发送请求到MQ然后接收响应回来,这个性能自然很高,吞吐氧也是很高的

但是如果你改成了甚于事务消息的机制之后,这里涉及到half消息、commit or rollback、写入内部topic、回调机制,等诸多复杂的环节

接着你的这台broker机器还必须直接把消息复制给其他的broker,完成多副本的冗余,这个过程涉及到两台broker机器之间的网络通信。另外一台broker机器还必须直接把消息复制给其他的broker,完成多副本的冗余,这个过程涉及到两台broker机器之间的网络通信。另外一台broker机器可数据到自己本地磁盘去,同样会比较慢

在broker完成了上述两个步骤之后,接着才能返回响应告诉你说这次消息写入已经成功了,大家试想一下,写入一条消息需要强制同步 刷磁盘,而且还需要同步复制消息给其他的broker机器

最后看你的消费者,当你的消费者拿到消息之后,比如他直接开启一个子线程去处理这批消息,然后他就直接返回 CONSUME_SUCCESS状态了,接着他就可以去处理下一批消息了!如果这样的话,你消费消息的速度会很快,吞吐量会很高! 但是如果为了保证数据不丢失,你必须是处理完一批消息再返回CONSUME_SUCCESS状态,那么此时你消费者处理消息的速度会降

吞吐量下降原因

低,吞吐量 自然也会下降了!