

图文 49 精益求精：深入研究一下生产者到底如何发送消息的？

945 人次阅读 2019-12-09 07:00:00

[详情](#) [评论](#)

精益求精：深入研究一下生产者到底如何发送消息的？

石杉老哥重磅力作：《互联网Java工程师面试突击》（第3季）【强烈推荐】：



全程真题驱动，精研Java面试中**6大专题**的高频考点，从面试官的角度剖析面试

(点击下方蓝字试听)

[《互联网Java工程师面试突击》（第3季）](#)

1、为了随时准备应对线上系统的问题，要深入研究MQ

最近一段时间加班加点的工作，小猛和整个订单技术团队终于搞定了公司的RocketMQ集群，而且还对订单系统架构做了初步的改造，解决了系统面临的很多技术问题

小猛看着自己手上的一个清单：

-(1) 下单核心流程环节太多，性能较差

- (2) 订单退款的流程可能面临退款失败的风险
- (3) 关闭过期订单的时候，存在扫描大量订单数据的问题
- ~~(4) 跟第三方系统耦合在一起，性能存在抖动的风险~~
- ~~(5) 大数据团队要获取订单数据，存在不规范直接查询订单数据库的问题~~
- ~~(6) 做秒杀活动时订单数据库压力过大~~

现在这个清单里剩下的就是订单退款偶尔会失败，以及关闭过期订单时扫描大量数据的问题了

接下来应该解决哪个问题呢？

小猛正在思考的时候，明哥过来告诉了他一个消息，现在公司的核心系统都已经接入了RocketMQ，而且核心链路的运转整个就是基于RocketMQ的，一旦在RocketMQ的使用过程中出了一点问题，那么公司的核心业务就会出问题。

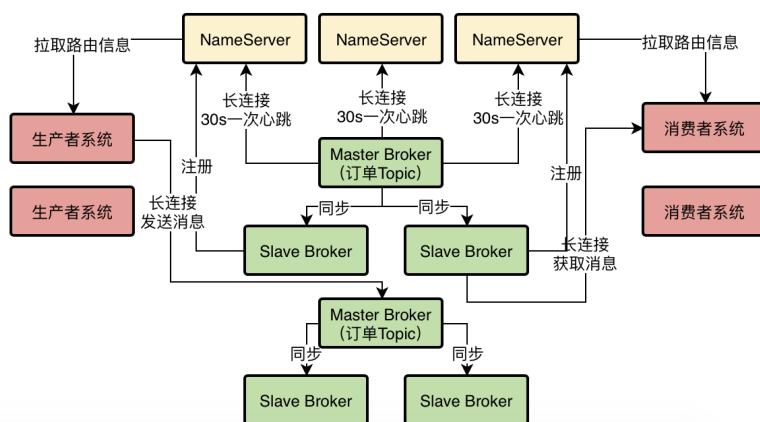
所以技术高层的意思是让我们必须要对RocketMQ进行一定深度的技术研究，保证万一线上系统出现一些问题，可以有足够的底层技术积累去分析和解决线上问题，万万不能对RocketMQ仅仅停留在简单使用的层次上。

因此明哥交给了小猛一个任务，趁着现在线上系统压力不大，出问题概率不高，而且系统刚刚接入RocketMQ不久，抓住这个时间窗口，赶紧去研究一下RocketMQ的底层运行原理，同时对团队输出技术分享，提升整个团队对MQ技术的掌控能力！

2、研究RocketMQ底层原理的顺序和思路

接着小猛就开始思考，到底应该采用一个什么样的顺序和思路去研究RocketMQ的底层原理呢？

小猛盯着自己手头的线上生产环境部署架构图。



目前公司生产环境的情况，就是部署了一个小规模的RocketMQ生产集群，基本都是在稳定运行中，可以支撑公司的核心链路以及秒杀业务，然后有订单系统、大数据系统、库存系统、积分系统等各种公司核心系统都接入了RocketMQ的生产和消费。

所以公司技术高层可能担心的是如果在RocketMQ生产消息或者消费消息的过程中出现了什么问题，就会直接导致核心链路出问题。

因此对照上面的部署架构图，小猛决定按照如下的思路来研究RocketMQ：

对生产者往Broker集群发送消息的底层原理做一个研究

看看Broker对于接收到的消息，到底是如何存储到磁盘上去的？

基于DLedger技术部署的Broker高可用集群，到底如何进行数据同步的？

消费者到底是基于什么策略选择Master或Slave拉取数据的？

消费者是如何从Broker拉取消息回来，进行处理以及ACK的？如果消费者故障了会如何处理？

按照这个思路就涵盖了RocketMQ的整个使用流程，如果把这些问题都研究一下，那么对RocketMQ的运行流程就比较了解了。

小猛接着就按照上面的思路开始一步一步的进行研究，在花了几段时间研究完RocketMQ的生产者的工作原理之后，他就对团队内部进行了一次技术分享。

下面就是小猛分析RocketMQ生产者工作原理的过程。

3、创建Topic的时候为什么要指定MessageQueue数量？

首先如果要搞明白生产者的工作原理，那么就必须先明白一个概念：MessageQueue是什么？

而要明白MessageQueue是什么，就必须把他跟Topic以及Broker综合起来看，才能搞明白。

如果我们要使用RocketMQ，你先部署出来一套RocketMQ集群这个肯定是必须的，在有了集群之后，就必须根据你的业务需要去创建一些Topic。

比如之前我们看到，我们需要一个“TopicOrderPaySuccess”的Topic去存放订单支付成功的消息。

像这些Topic就可以在之前我们讲过的RocketMQ可视化工作台里去创建，在里面就可以创建一个Topic出来，在创建Topic的时候需要指定一个很关键的参数，就是MessageQueue。

简单来说，就是你要指定你的这个Topic对应了多少个队列，也就是多少个MessageQueue。

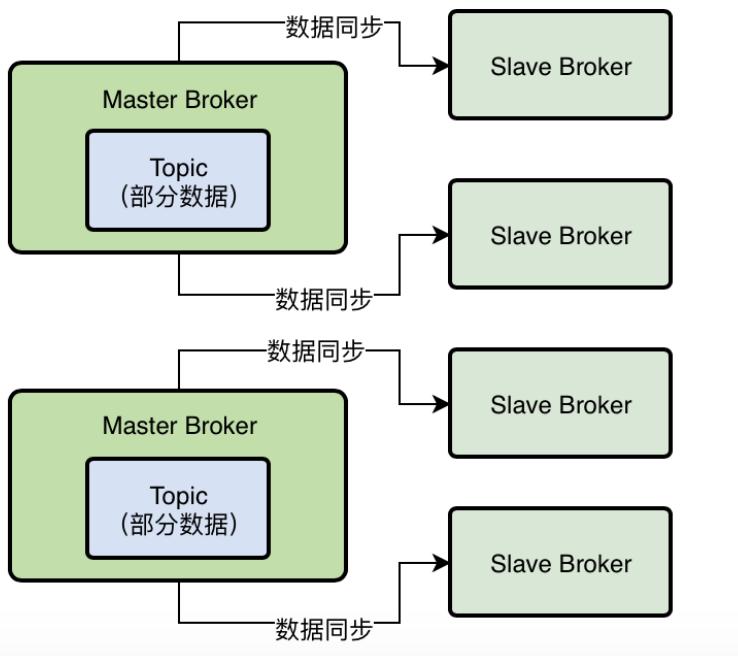
那么这个MessageQueue是用来干嘛的？大家是不是觉得很奇怪？因为此时看不出来他是干嘛用的！

4. Topic、MessageQueue以及Broker之间到底是什么关系？

其实Topic、MessageQueue以及Broker之间是有关系的，咱们来举一个例子

比如你现在有一个Topic，我们为他指定了4个MessageQueue，那么我们接着来思考一下，这个Topic的数据在Broker集群中是如何分布的？

之前最早我们给大家讲过，每个Topic的数据都是分布式存储在多个Broker中的，比如下面的图里我们会看到这个示意。



但是我们如何决定这个Topic的哪些数据放这个Broker上，哪些数据放那个Broker上？这是一个问题

所以在这里RocketMQ引入了MessageQueue的概念，本质上就是一个数据分片的机制。

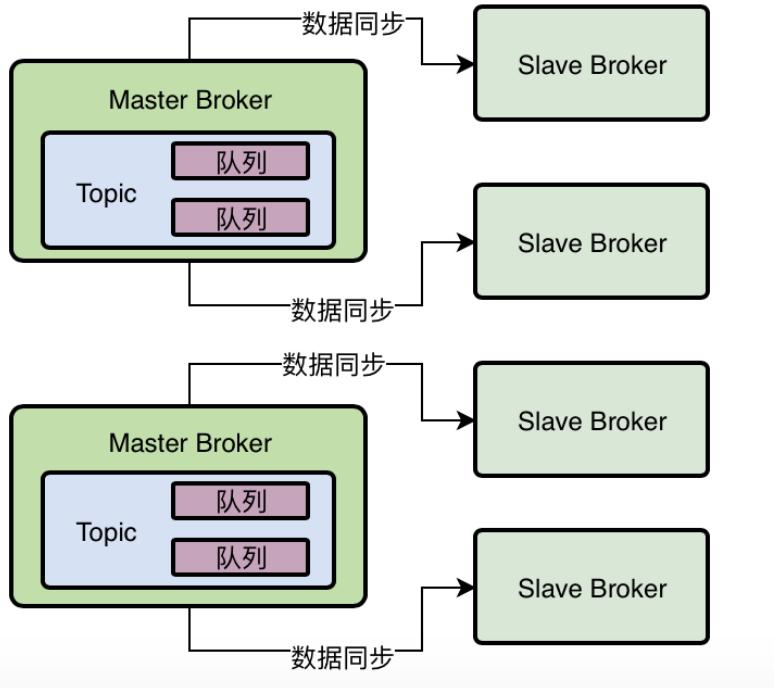
在这个机制中，假设你的Topic有1万条数据，然后你的Topic有4个MessageQueue，那么大致可以认为会在每个MessageQueue中放入2500条数据

当然，这个不是绝对的，有可能有的MessageQueue的数据多，有的数据少，这个要根据你的消息写入MessageQueue的策略来定。

但是我们这里先假定在每个MessageQueue中会平均分配Topic的数据吧，那么下一个问题来了，我们有4个MessageQueue平均分配了Topic的数据，这些MessageQueue放在哪里？

当然是放在Broker上了！

也就是说，很有可能就是在2个Broker上，每个Broker放两个MessageQueue，我们看下面的图就是这个示意。



所以其实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个数据分片，然后在每个Broker机器上都存储一些MessageQueue。

通过这个方法，就可以实现Topic数据的分布式存储！

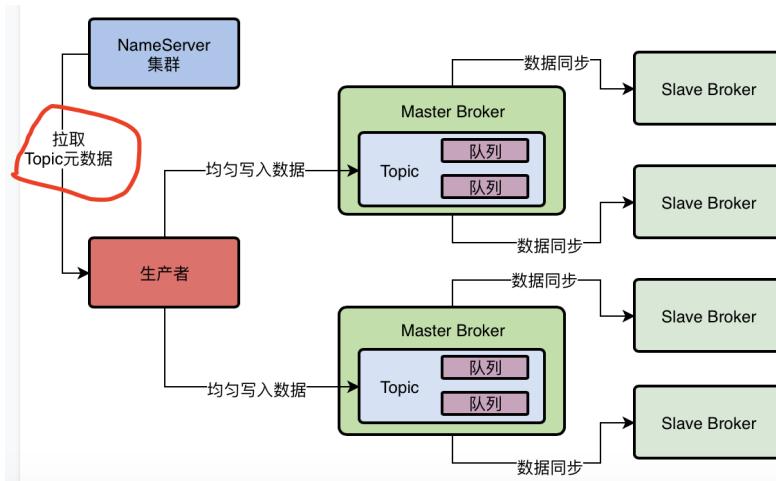
5、生产者发送消息的时候写入哪个MessageQueue？

接着我们要考虑一个问题，生产者在发送消息的时候，会写入到哪个MessageQueue中？

要解决这个问题，大家首先就要记得之前我们讲解过的一个重要的点，生产者会跟NameServer进行通信获取Topic的路由数据。

所以生产者从NameServer中就会知道，一个Topic有几个MessageQueue，哪些MessageQueue在哪台Broker机器上，哪些MessageQueue在另外一台Broker机器上，这些都会知道

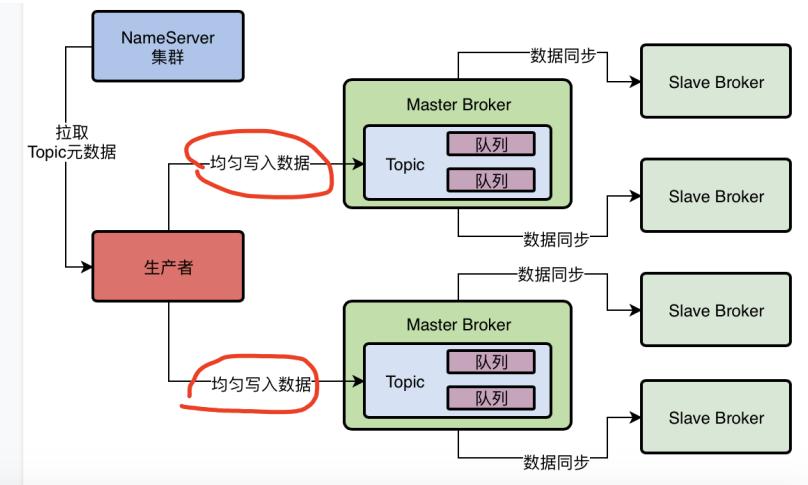
我们看下面的图。



然后呢，现在我们暂时先认为生产者会均匀的把消息写入各个MessageQueue，就是比如这个生产者发送出去了20条数据，那么4个MessageQueue就是每个都会写入5条数据。

至于其他的写入MessageQueue的策略，我们后续会结合其他的高阶功能和业务场景来讲解，现在大家先不要去纠结这个问题。

所以我们看一下下面的图，在图里就有生产者把数据均匀写入MessageQueue的示意。



通过这个方法，是不是就可以让生产者把写入请求分散给多个Broker？是不是也可以让每个Broker都均匀分摊到一定的写入请求压力？

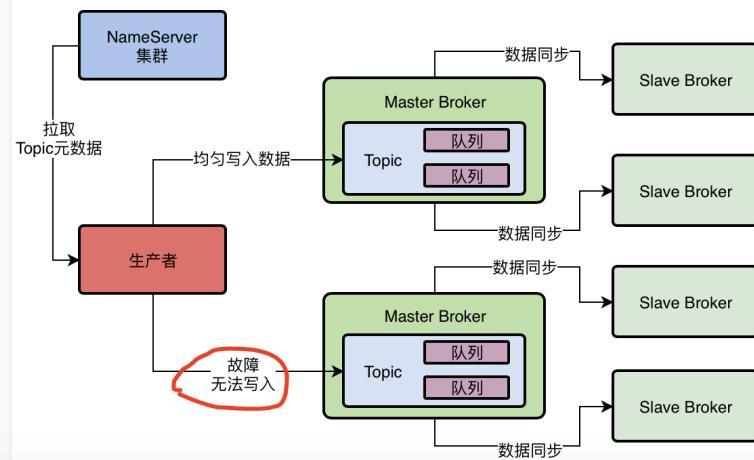
这样假设单个Broker可以抗每秒7万并发，那么两个Broker就可以抗每秒14万并发！这样就可以实现RocketMQ集群抗下每秒10万+超高并发的场景了！

另外通过这个方法，是不是就可以让一个Topic中的数据分散在多个MessageQueue中，进而分散在多个Broker机器上？这样就可以实现RocketMQ集群分布式存储海量的消息数据了！

6、如果某个Broker出现故障该怎么办？

接下来我们分析一下，如果某个Broker临时出现故障了，比如Master Broker挂了，此时正在等待的其他Slave Broker自动热切换为Master Broker，那么这个时候对这一组Broker就没有Master Broker可以写入了

大家看下面的图。



如果你还是按照之前的策略来均匀把数据写入各个Broker上的MessageQueue，那么会导致你在一段时间内，每次访问到这个挂掉的Master Broker都会访问失败，这个似乎不是我们想要的样子。

对于这个问题，通常来说建议大家在Producer中开启一个开关，就是`sendLatencyFaultEnable`

一旦打开了这个开关，那么他会有一个自动容错机制，比如如果某次访问一个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Broker了。

这样的话，就可以避免一个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。

那么这样过一段时间之后，可能这个Master Broker就已经恢复好了，比如他的Slave Broker切换为了Master可以让别人访问了。

7、对今天的内容做一点小小的总结

最后，我们来对今天的文章做一点小小的总结，今天我们讲了以下几个内容：

为了解决线上系统使用RocketMQ过程中可能遇到的问题，我们需要对RocketMQ底层运行原理做一定深入的研究

对RocketMQ底层原理的研究顺序

创建Topic的时候需要指定关键的MessageQueue

Topic、MessageQueue和Broker之间的关系是什么？

生产者是如何将消息写入MessageQueue的？

如果Broker故障的时候，生产者如何让他自动启动容错处理？

End

专栏版权归公众号**狸猫技术窝**所有

未经许可不得传播，如有侵权将追究法律责任

狸猫技术窝其他精品专栏推荐：

[《从零开始带你成为JVM实战高手》](#)

[《21天Java面试突击训练营》（分布式篇）](#) (现更名为：[互联网Java工程师面试突击第2季](#))

[互联网Java工程师面试突击（第1季）](#)

重要说明：

如何提问：每篇文章都有评论区，大家可以尽情在评论区留言提问，我会逐一答疑

如何加群：购买了狸猫技术窝专栏的小伙伴都可以加入**狸猫技术交流群**

具体加群方式，请参见[目录菜单](#)下的文档：《付费用户如何加群？》[\(购买后可见\)](#)

Copyright © 2015-2019 深圳小鹅网络技术有限公司 All Rights Reserved. 粤ICP备15020529号

 小鹅通提供技术支持