图文 110 对于一条消息，Producer是如何选择MessageQueue去发送的?

278 人次阅读 2020-03-10 07:00:00

详情 评论

对于一条消息，Producer是如何选择MessageQueue去发送的?

继《从零开始带你成为JVM实战高手》后，阿里资深技术专家携新作再度出山，重磅推荐：

（点击下方蓝字试听）

《从零开始带你成为MySQL实战优化高手》

上一次我们讲完了Producer发送消息的时候，上来不管三七二十一，其实会先检查一下要发送消息的Topic的路由数据是否在本地缓存，如果不在的话，就会通过底层的Netty网络通信模块去发送一个请求到NameServer去拉取Topic路由数据，然后缓存在Producer的本地。

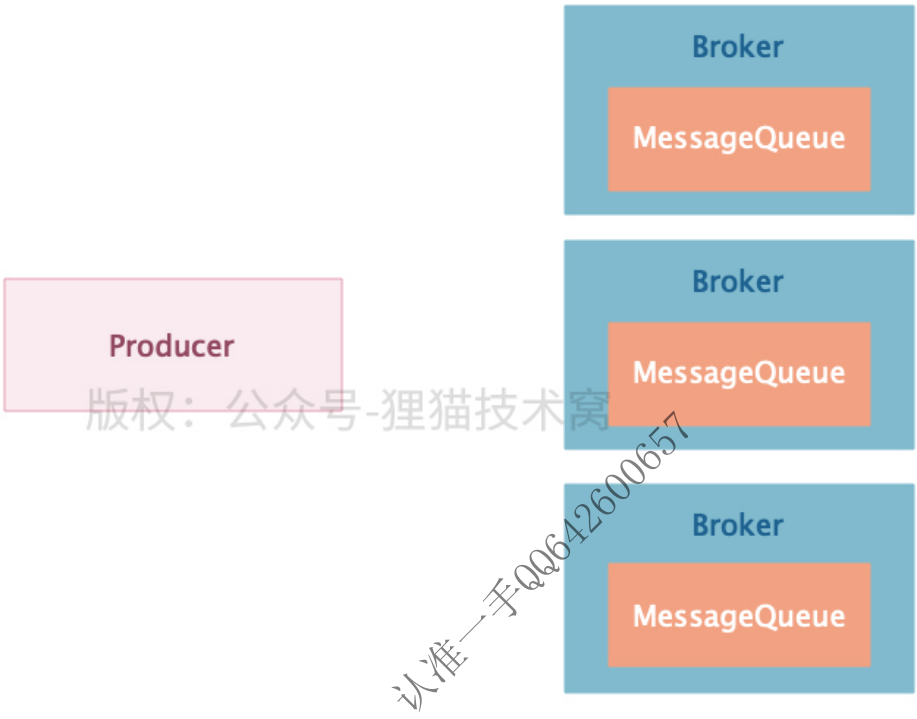
然后今天我们就来继续讲解这里的底层逻辑，之前已经跟大家说过了，我们现在已经转换了讲解的方式，之前是通过分析源码细节来讲解，重点是教会大家读源码的一些思路和方法，不要陷于细节，抓住主要流程，而且要懂得如何推测源码的意思。

但是RocketMQ的源码量实在太大了，根本不是一本书或者一个专栏能分析完的，所以昨天开始，我们已经切换了讲解方式，开始用一步一图的方式讲解RocketMQ的底层原理了，同时辅之以 一些源码的片段，让大家在理解底层原理和流程的同时，如果有兴趣，可以自己去阅读对应的源码细节。

那么今天我们应该继续讲解的是，当你拿到了一个Topic的路由数据之后，其实接下来就应该选择要发送消息到这个Topic的哪一个MessageQueue上去了！

因为大家都知道，Topic是一个逻辑上的概念，一个Topic的数据往往是分布式存储在多台Broker机器上的，因此Topic本质是由多个MessageQueue组成的。

每个MessageQueue都可以在不同的Broker机器上，当然也可能一个Topic的多个MessageQueue在一个Broker机器上，如下图所示。



所以今天我们主要就是讲解，你要发送的消息，到底应该发送到这个Topic的哪个MessageQueue上去呢？

只要你知道了要发送消息到哪个MessageQueue上去，然后就知道这个MessageQueue在哪台Broker机器上，接着就跟那台Broker机器建立连接，发送消息给他就可以了。

之前给大家说过，发送消息的核心源码是在DefaultMQProducerImpl.sendDefaultImpl()方法中的，在这个方法里，只要你获取到了Topic的路由数据，不管从本地缓存获取的，还是从NameServer拉取到的，接着就会执行下面的核心代码。

```
MessageQueue mqSelected = this.selectOneMessageQueue(topicPublishInfo, lastBrokerName);
```

这行代码其实就是在选择Topic中的一个MessageQueue，然后发送消息到这个MessageQueue去，在这行代码里面，实现了一些Broker故障自动回避机制，但是这个我们后续再讲，先看最基本的选择MessageQueue的算法

```java
int index = tpInfo.getSendWhichQueue().getAndIncrement();

for (int i = 0; i < tpInfo.getMessageQueueList().size(); i++) {
    int pos = Math.abs(index++) % tpInfo.getMessageQueueList().size();
    if (pos < 0)
        pos = 0;
    MessageQueue mq = tpInfo.getMessageQueueList().get(pos);
    if (latencyFaultTolerance.isAvailable(mq.getBrokerName())) {
        if (null == lastBrokerName || mq.getBrokerName().equals(lastBrokerName))
            return mq;
    }
}

```

上面的代码其实非常的简单，他先获取到了一个自增长的index，大家注意到没有？

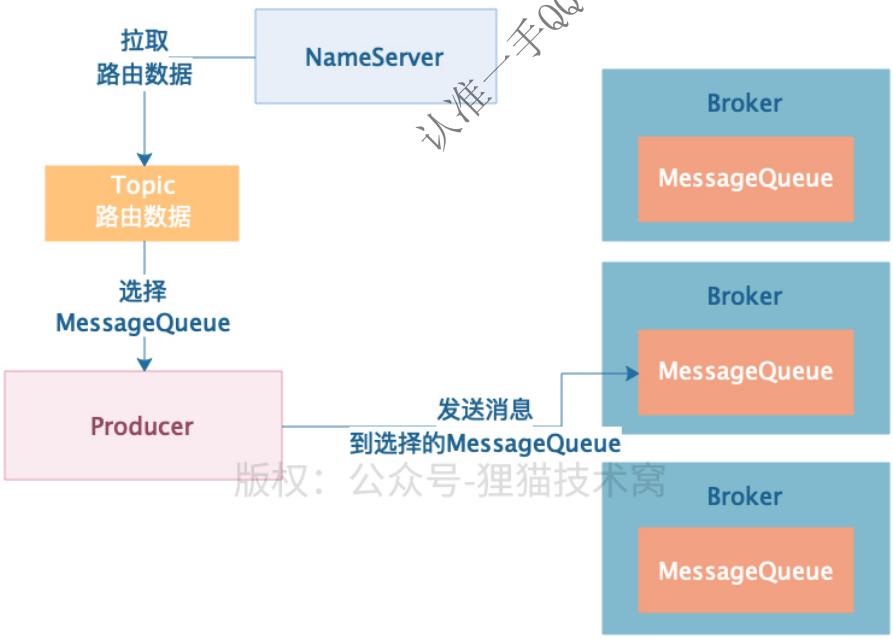
接着其实他核心的就是用这个index对Topic的MessageQueue列表进行了取模操作，获取到了一个MessageQueue列表的位置，然后返回了这个位置的MessageQueue。

说实话，你只要自己去试试就知道了，这种操作就是一种简单的负载均衡的算法，比如一个Topic有8个MessageQueue，那么可能第一次发送消息到MessageQueue01，第二次就发送消息到MessageQueue02，以此类推，就是轮询把消息发送到各个MessageQueue而已！

这就是最基本的MessageQueue选择算法，但是肯定有人会说了，那万一某个Broker故障了呢？此时发送消息到哪里去呢？

所以其实这个算法里有很多别的代码，都是实现Broker规避机制的，这个后续我们再讲。

我们先看下图，给大家展示了一个最基本的MessageQueue选择算法。



End

狸猫技术窝精品专栏及课程推荐：

- 《从零开始带你成为JVM实战高手》

- 《21天互联网Java进阶面试训练营》（分布式篇）
- 《互联网Java工程师面试突击》（第1季）
- 《互联网Java工程师面试突击》（第3季）

重要说明：

- 如何提问：每篇文章都有评论区，大家可以尽情留言提问，我会逐一答疑
- 如何加群：购买狸猫技术窝专栏的小伙伴都可以加入狸猫技术交流群，一个非常纯粹的技术交流的地方

  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**）

返回
前进
重新加载
打印