图文 68 发送消息零丢失方案：RocketMQ事务消息的实现流程分析

详情 评论

发送消息零丢失方案：RocketMQ事务消息的实现流程分析

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）

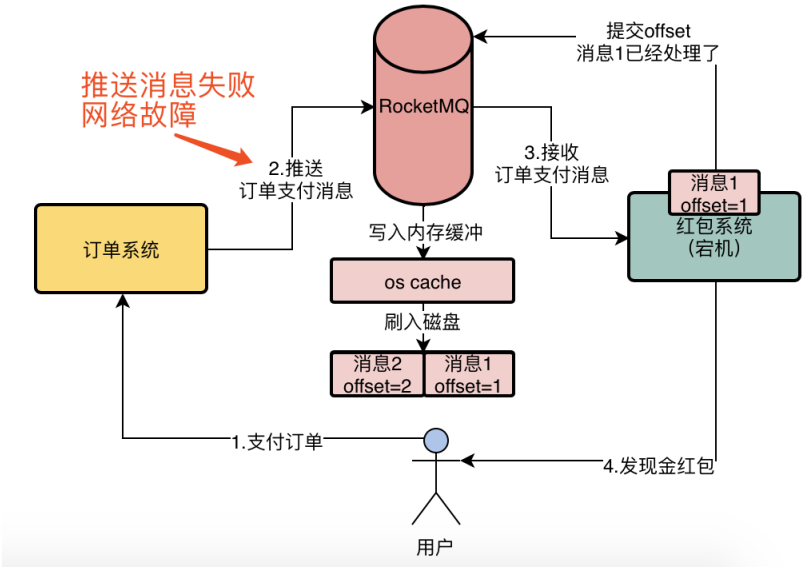
1、解决消息丢失的第一个问题：订单系统推送消息丢失

既然我们已经明确了消息在基于MQ传输的过程中可能丢失的几个地方，那么我们接着就得一步一步考虑如何去解决各个环节丢失消息的问题

首先要解决的第一个问题，就是订单系统推送消息到MQ的过程中，可能消息就丢失了。

之前我们也说过了，可能在订单系统推送消息到MQ的过程中，就因为常见的网络故障之类的问题，导致消息就丢失了

这里我们可以看一下下图中的示意。



在RocketMQ中，有一个非常强悍有力的功能，就是事务消息的功能，凭借这个事务级的消息机制，就可以让我们确保订单系统推送给出去的消息一定会成功写入MQ里，绝对不会半路就搞丢了。

今天我们就来系统的分析一下RocketMQ的事务消息机制的原理。

2、发送half消息到MQ去，试探一下MQ是否正常

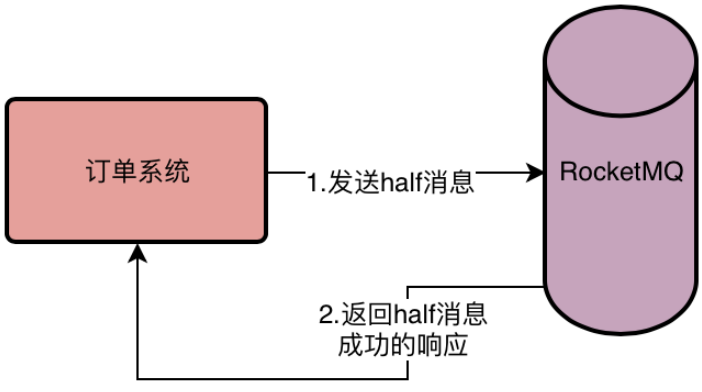
首先作为我们的订单系统而言，假设他收到了一个订单支付成功的通知之后，他必然是需要在自己的订单数据库里做一些增删改操作的，比如更新订单状态之类的。

可能有的朋友会觉得，订单系统不就是先在自己数据库里做一些增删改操作，然后就直接发个消息到MQ去，让其他关注这个订单支付成功消息的系统去从MQ获取消息做对应的处理就可以了么？

事实上还真不是这么简单。

在基于RocketMQ的事务消息机制中，我们首先要让订单系统去发送一条half消息到MQ去，这个half消息本质就是一个订单支付成功的消息，只不过你可以理解为他这个消息的状态是half状态，这个时候红包系统是看不见这个half消息的

然后我们去等待接收这个half消息写入成功的响应通知，我们看下面的图



看到这儿可能有的朋友就开始有点郁闷了，可能会觉得你没事儿先发个half消息给MQ干什么？

大家先别着急，你可以想一下，假设你二话不说让订单系统直接做了本地的数据库操作，比如订单状态都更新为了已完成，然后你再发送消息给MQ，结果报出一堆异常，发现MQ挂了。

这个时候，必然导致你没法通过消息通知到红包系统去派发红包，那用户一定会发现自己订单支付了，结果红包没收到。

所以，在这里我们首先第一件事，不是先让订单系统做一些增删改操作，而是先发一个half消息给MQ以及收到他的成功的响应，初步先跟MQ做个联系和沟通

大概这个意思就是说，确认一下MQ还活着，MQ也知道你后续可能想发送一条很关键的不希望丢失的消息给他了！

3、万一要是half消息写入失败了呢？

这里我们先来分析第一种情况，万一要是你订单系统写half消息给MQ就失败了呢？

可能你发现报错了，可能MQ就挂了，或者这个时候网络就是故障了，所以导致你的half消息都没发送成功，总之你现在肯定没法跟MQ通信了。

这个时候你的订单系统就应该执行一系列的回滚操作，比如对订单状态做一个更新，让状态变成“关闭交易”，同时通知支付系统自动进行退款，这才是正确的做法。

因为你订单虽然支付了，但是包括派发红包、发送优惠券之类的后续操作是无法执行的，所以此时必然应该把钱款退还给用户，说交易失败了。

这里给大家插播一个我曾经亲身经历过的事情，曾经有一次在一家便利店购物的时候，我这里都已经显示扫码支付成功了，但是店员那边说在等待他们系统确认，结果等了一会儿，系统显示后台系统有异常，交易失败了，然后过了一会儿就让支付宝自动退款给我了。

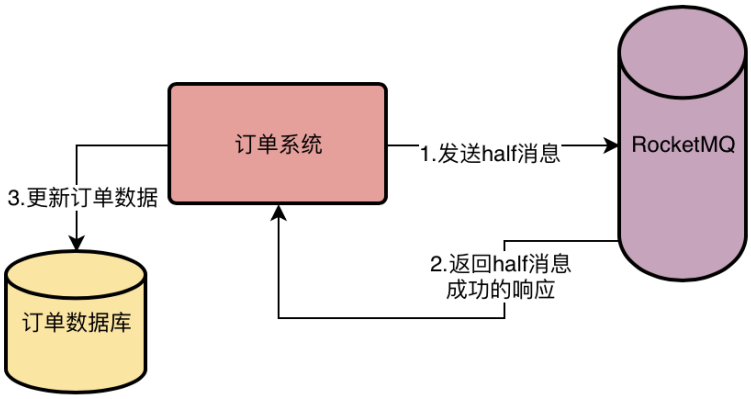
其实这就是类似的例子。

4、half消息成功之后，订单系统完成自己的任务

接着我们来考虑第二种情况，你的half消息写成功了，这个时候你应该干什么呢？

这个时候你的订单系统就应该在自己本地的数据库里执行一些增删改操作了，因为一旦half消息写成功了，就说明MQ肯定已经收到这条消息了，MQ还活着，而且目前你是可以跟MQ正常沟通的。

我们看下面的图，示意了下一步是订单系统执行自己的增删改操作。



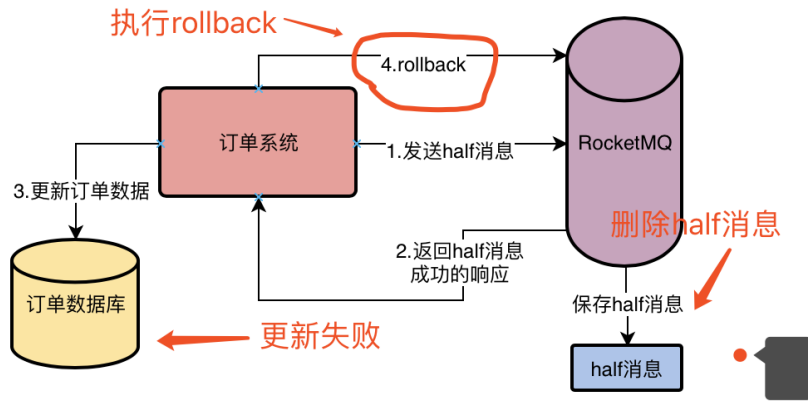
5、如果订单系统的本地事务执行失败了怎么办？

接着我们继续看下一种情况，万一要是订单系统更新自己的数据库失败了怎么办？

比如订单系统的数据库当时也有网络异常，或者数据库挂了，总而言之，就是你想把订单更新为“已完成”这个状态，是干不成了。

这个时候其实也很简单，直接就是让订单系统发送一个rollback请求给MQ就可以了。这个意思就是说，你可以把之前我发给你的half消息给删除掉了，因为我自己这里都出问题了，已经无力跟你继续后续的流程了。

我们看下面的图，我给出了这个示意。



当然你发送rollback请求给MQ删除那个half消息之后，你的订单系统就必须走后续的回退流程了，就是通知支付系统退款。

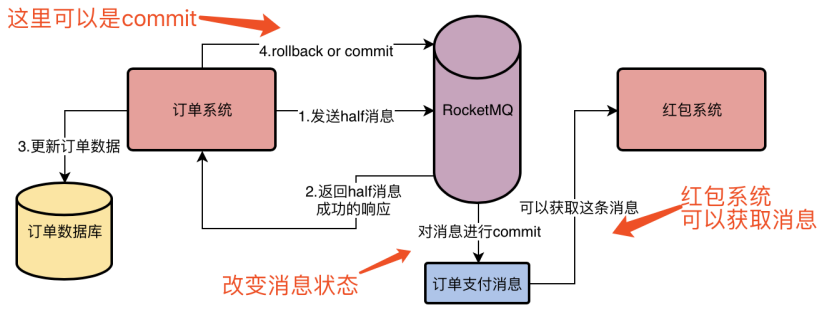
当然这里可能还有一些订单系统自己的高可用降级的机制需要考虑，比如数据库无法更新了，此时你可能需要在机器本地磁盘文件里写入订单支付失败的记录。

然后你可以开一个后台线程在MySQL数据库恢复之后，再把订单状态更新为“已关闭”。不过这个不在我们讨专栏的范围之内。

6、如果订单系统完成了本地事务之后，接着干什么？

如果订单系统成功完成了本地的事务操作，比如把订单状态都更新为“已完成”了，此时你就可以发送一个commit请求给MQ，要求让MQ对之前的half消息进行commit操作，让红包系统可以看见这个订单支付成功消息

我们看下面的图。



之前我们也提到过了，所谓的half消息实际就是订单支付成功的消息，只不过他的状态是half

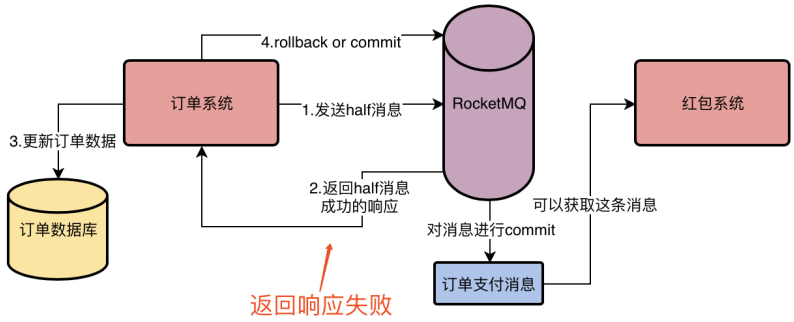
也就是他是half状态的时候，红包系统是看不见他的，没法获取到这条消息，必须等到订单系统执行commit请求，消息被commit之后，红包系统才可以看到和获取这条消息进行后续处理。

7、让流程严谨一些：如果发送half消息成功了，但是没收到响应呢？

大致的事务消息的流程是讲完了，但是接着让我们来进行比较严谨的分析，如果我们把half消息发送给MQ了，MQ给保存下来了，但是MQ返回给我们的响应我们没收到呢？此时会发生什么事情？

这个时候我们没收到响应，可能就会网络超时报错，也可能直接有其他的异常错误，这个时候订单系统会误以为是发送half消息到MQ失败了，订单系统就直接会执行退款流程了，订单状态也会标记为“已关闭”。

我们看下面的图的示意。

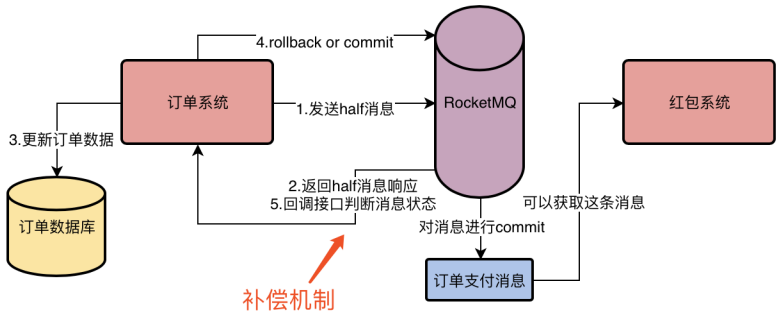


但这个时候MQ已经存储下来一条half消息了，那对这个消息怎么处理?

其实RocketMQ这里有一个补偿流程，他会去扫描自己处于half状态的消息，如果我们一直没有对这个消息执行commit/rollback操作，超过了一定的时间，他就会回调你的订单系统的一个接口

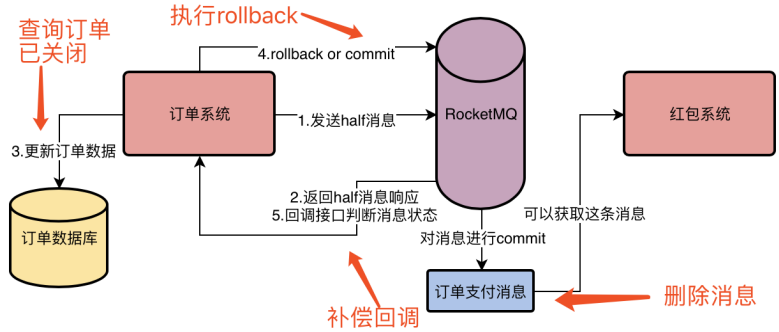
他会问问你：这个消息到底怎么回事？你到底是打算commit这个消息还是要rollback这个消息?

我们看下图示意



这个时候我们的订单系统就得去查一下数据库，看看这个订单当前的状态，一下发现订单状态是“已关闭”，此时就知道，你必然得发送rollback请求给MQ去删除之前那个half消息了!

我们看下图。



8、如果rollback或者commit发送失败了呢?

我们再假设一种场景，如果订单系统是收到了half消息写入成功的响应了，同时尝试对自己的数据库更新了，然后根据失败或者成功去执行了rollback或者commit请求，发送给MQ了，结果因为网络故障，导致rollback或者commit请求发送失败了呢?

这个时候其实也很简单，因为MQ里的消息一直是half状态，所以说他过了一定的超时时间会发现这个half消息有问题，他会回调你的订单系统的接口

你此时要判断一下，这个订单的状态如果更新为了“已完成”，那你就得再次执行commit请求，反之则再次执行rollback请求。

本质这个MQ的回调就是一个补偿机制，如果你的half消息响应没收到，或者rollback、commit请求没发送成功，他都会来找你问问对half消息后续如何处理。

再假设一种场景，如果订单系统收到了half消息写入成功的响应了，同时尝试对自己的数据库更新了，然后根据失败或者成功去执行了rollback或者commit请求，发送给MQ了。很不巧，mq在这个时候挂掉了，导致rollback或者commit请求发送失败，怎么办？

如果是这种情况的话，那就等mq自己重启了，重启之后他会扫描half消息，然后还是通过上面说到的补偿机制，去回调你的接口

9、停一下脚步，想想上面这个流程的意义在哪里？

看到这里我们来停下脚步想想，上面这个流程的意义在哪里呢？

其实很简单，如果你的MQ有问题或者网络有问题，half消息根本都发不出去，此时half消息肯定是失败的，那么订单系统就不会执行后续流程了！

如果要是half消息发送出去了，但是half消息的响应都没收到，然后执行了退款流程，那MQ会有补偿机制来回调找你询问要commit还是rollback，此时你选择rollback删除消息就可以了，不会执行后续流程！

如果要是订单系统收到half消息了，结果订单系统自己更新数据库失败了，那么他也会进行回滚，不会执行后续流程了！

如果要是订单系统收到half消息了，然后还更新自己数据库成功了，订单状态是“已完成”了，此时就必然会发送commit请求给MQ，一旦消息commit了，那么必然保证红包系统可以收到这个消息！

而且即使你commit请求发送失败了，MQ也会有补偿机制，回调你接口让你判断是否重新发送commit请求

总之，就是你的订单系统只要成功了，那么必然要保证MQ里的消息是commit了可以让红包系统看到他！

所以大家可以结合我们的图思考一下上述流程，通过这套事务消息的机制，是不是就可以保证我们的订单系统一旦成功执行了数据库操作，就一定会通知到红包系统去派发红包？至少订单系统到MQ之间的消息传输是不会有丢失的问题了！

End
