图文 74 基于 RocketMQ 设计的全链路消息零丢失方案总结

505 人次阅读 2020-01-08 07:00:00

详情 评论

基于 RocketMQ 设计的全链路消息零丢失方案总结

继《从零开始带你成为JVM实战高手》后，救火队长携新作再度出山，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》

1、对全链路消息零丢失方案进行总结

基于我们之前讲解的内容，我们现在可以对全链路的消息零丢失方案进行一个总结了：

发送消息到MQ的零丢失：

方案一（同步发送消息 + 反复多次重试）

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

MQ收到消息之后的零丢失：开启同步刷盘策略 + 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MQ自己不会弄丢消息

狸猫技术
进店逛
相关频道
从 0 开始
间件实战
已更新9

消费消息的零丢失：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程异步处理消息的方式即可

如果大家想要保证在一个消息基于MQ流转的时候绝对不会无缘无故的丢失，那么可以采取上述一整套的方案，包括落地的代码演示都已经给大家看到了。

但是今天我们除了总结这个方案之外，我们还需要对消息零丢失方案进行一些优劣分析。

2、消息零丢失方案的优势与劣势

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。

但是他的劣势在哪里呢？

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链路的性能大幅度下降，让你的MQ的吞吐量大幅度的下降

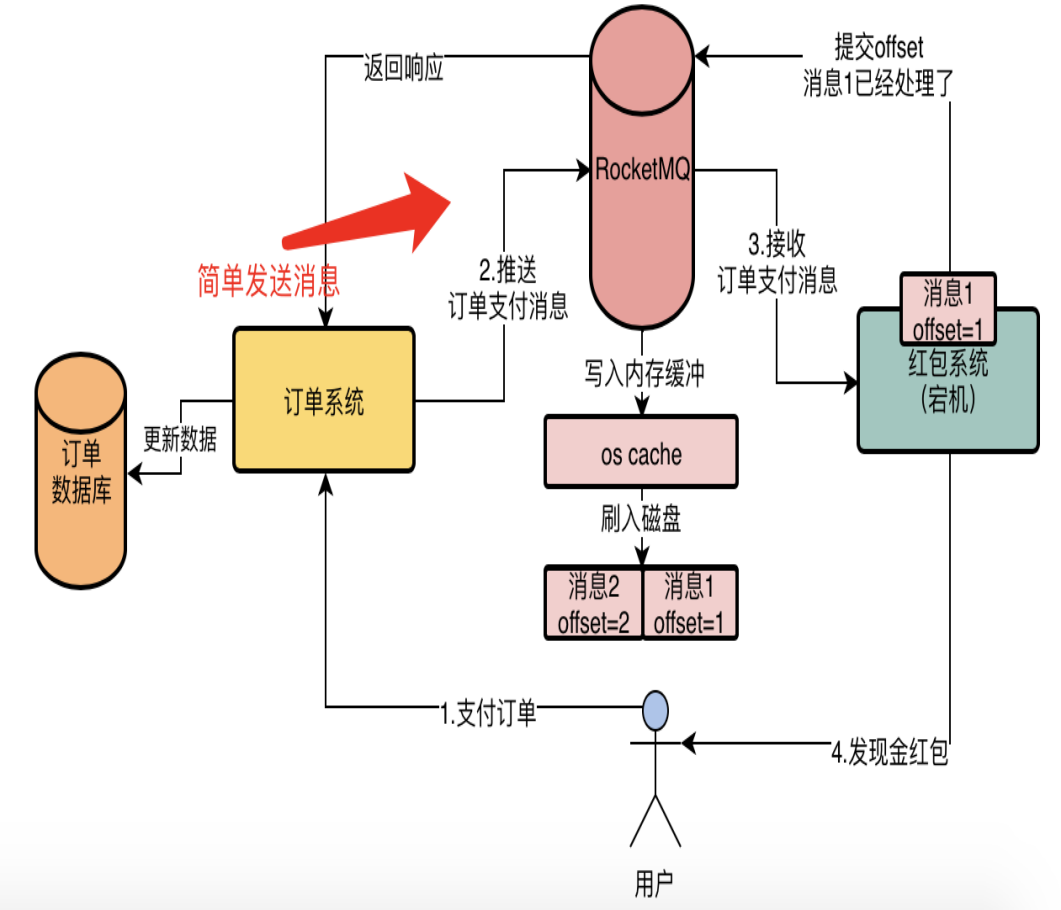
比如本身你的系统和MQ配合起来，每秒可以处理几万条消息的，结果当你落地消息零丢失方案之后，可能每秒只能处理几千条消息了。

为什么会这样呢？我们接下来一步一步分析一下。

3、为什么消息零丢失方案会导致吞吐量大幅度下降？

我们先来看这个发送消息到MQ的环节，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，吞吐量也是很高的

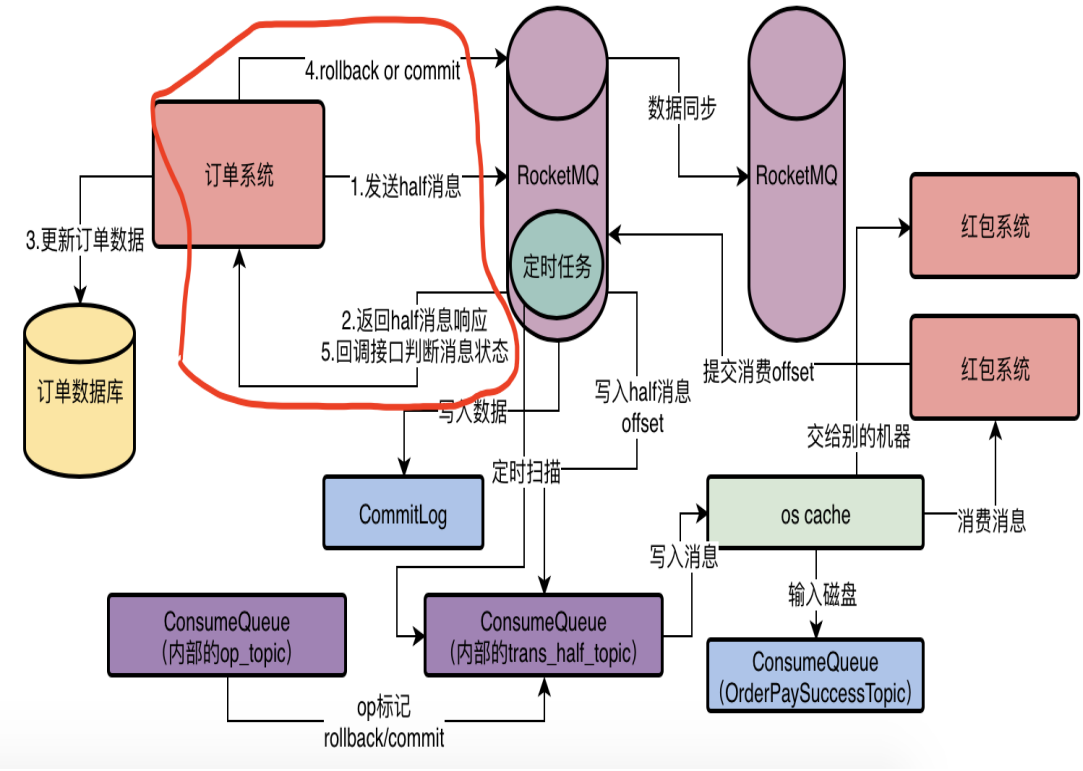
我们看下面的图示



但是如果你改成了基于事务消息的机制之后呢？

那么此时这里的实现原理图如下所示，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

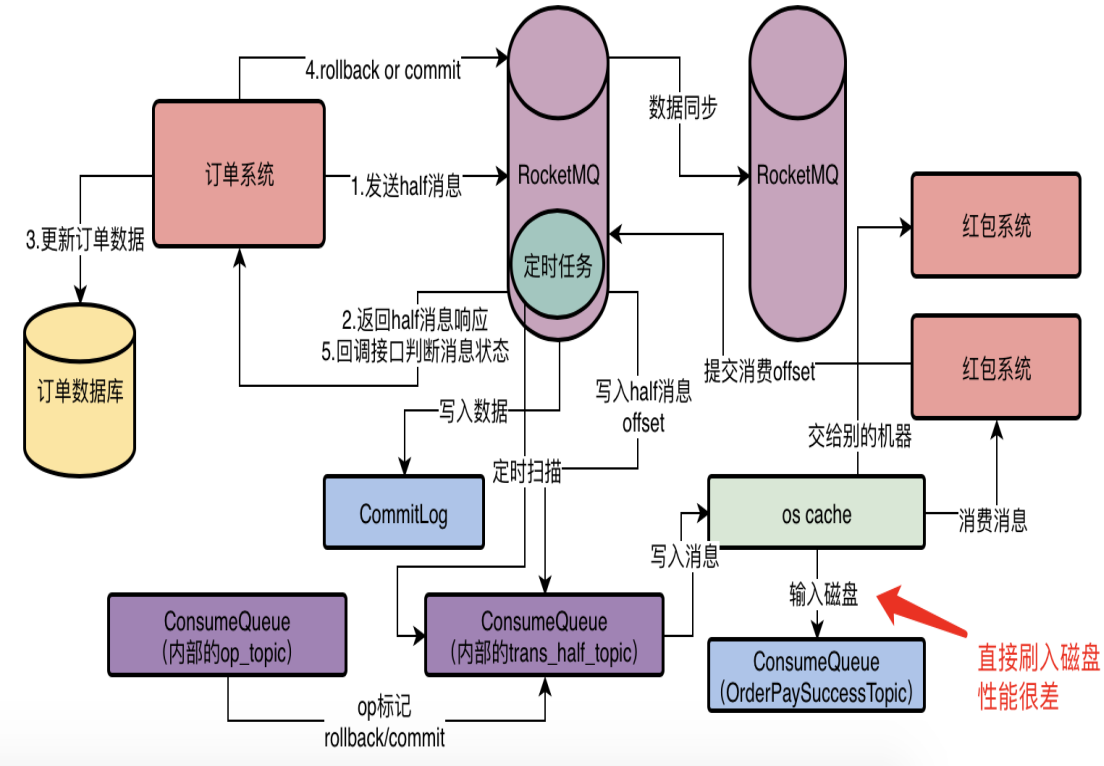
不说别的，光是你成功发送一条消息，都至少要half + commit两次请求。



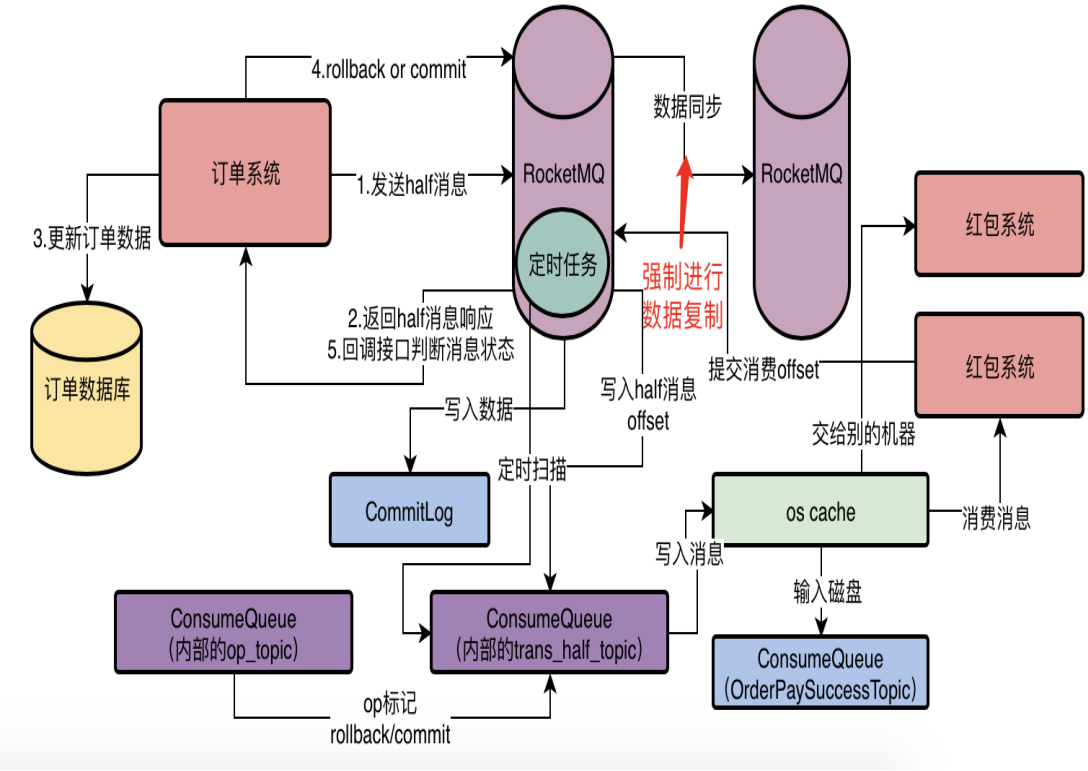
所以当你一旦上了如此复杂的方案之后，势必会导致你的发送消息的性能大幅度下降，同时发送消息到MQ的吞吐量大幅度下降。

接着我们再看MQ收到消息之后的行为，在MQ收到消息之后，一样会让性能大幅度下降。

首先MQ的一台broker机器收到了消息之后，必然直接把消息刷入磁盘里，这个性能就远远低于你写入os cache了，完全不是一个数量级的，比如你写入os cache相当于是内存，可能仅仅需要0.1ms，但是你写入磁盘文件可能就需要10ms！如下图。



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机器写数据到自己本地磁盘去，同样会比较慢，如下图。



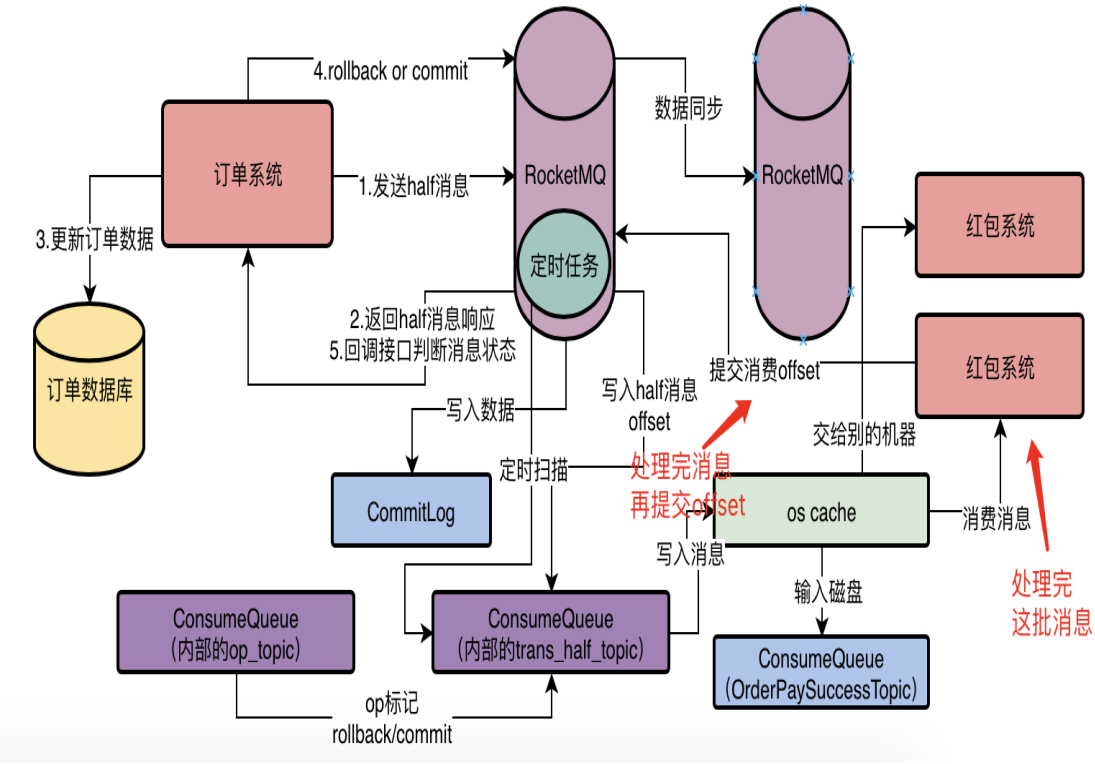
在broker完成了上述两个步骤之后，接着才能返回响应告诉你说这次消息写入已经成功了，大家试想一下，写入一条消息需要强制同步刷磁盘，而且还需要同步复制消息给其他的broker机器

这两个步骤一加入，可能原本10ms的事儿就会变成100ms了！所以这里也势必会导致性能大幅度下降，MQ的broker的吞吐量会大幅度下降。

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低，吞吐量 自然也会下降了！

我们看下图的示意



4、消息零丢失方案到底适合什么场景?

所以简单一句话，如果你一定要上消息零丢失方案，那么必然导致从头到尾的性能下降以及MQ的吞吐量下降。

所以一般大家不要轻易在随便一个业务里就上如此重的一套方案，要明白这背后的成本!

那么消息零丢失方案到底适用于什么场景呢?

一般我们建议，对于跟金钱、交易以及核心数据相关的系统和核心链路，可以上这套消息零丢失方案。

比如支付系统，他是绝对不能丢失任何一条消息的，你的性能可以低一些，但是不能有任何一笔支付记录丢失。

比如订单系统，公司一般是不能轻易丢失一个订单的，毕竟一个订单就对应一笔交易，如果订单丢失，用户还支付成功了，你轻则要给用户赔付损失，重则弄不好要经受官司，特别是一些B2B领域的电商，一笔线上交易可能多大几万几十万。

所以对这种非常非常核心的场景和少数几条核心链路，才会建议大家上这套复杂的消息0丢失方案。

而对于其他大部分没那么核心的场景和系统，其实即使丢失一些数据，也不会导致太大的问题，此时可以不采取这些方案，或者说你可以在其他的场景里做一些简化。

比如你可以把事务消息方案退化成“**同步发送消息 + 反复重试几次**”的方案，如果发送消息失败，就重试几次，但是大部分时候可能不需要重试，那么也不会轻易的丢失消息的!最多在这个方案里，可能会出现一些数据不一致的问题。

或者你把broker的刷盘策略改为异步刷盘，但是上一套主从架构，即使一台机器挂了，os cache里的数据丢失了，但是其他机器上还有数据。但是大部分时候broker不会随便宕机，那么异步刷盘策略下性能还是很高的。

所以说，对于非核心的链路，非金钱交易的链路，大家可以适当简化这套方案，用一些方法避免数据轻易丢失，但是同时性能整体很高，即使有极个别的数据丢失，对非核心的场景，也不会有太大的影响。

5、小作业：给你自己的项目设计一套消息0丢失方案

今天给大家留一个小作业，是跟每个人自己的系统相关的

大家思考一下自己负责的项目，你是否需要上消息零丢失方案?如果不需要上完整的复杂方案的话，是否可以上一些简化的方案尽量避免数据丢失?

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**）

Copyright © 2015-2020 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