ROCKETMQ 事务消息原理

基于 MQ 的分布式事务方案，本质上是对本地消息表的一个封装，整体流程与本地消息表一致，唯一不同的就是将本地消息表存在了 MQ 内部，而不是业务数据库，事务消息解决的是生产端的消息发送与本地事务执行的原子性问题，这里的界限一定要清楚，是确保 MQ 生产端正确无误地将消息发送出来，没有多发，也不会漏发，至于发送后消费端有没有正常的消费消息，这种异常场景将由 MQ 消息消费失败重试机制来保证。

RocketMQ 设计中的 broker 与 producer 端的双向通信能力，使得 broker 天生可以作为一个事务协调者；而 RocketMQ 本身提供的存储机制则为事务消息提供了持久化能力；RocketMQ 的高可用机制以及可靠消息设计则为事务消息在系统发生异常时依然能够保证达成事务的最终一致性。

RocketMQ 实现事务一致性的原理

（1）正常情况：在事务主动方服务正常，没有发生故障的情况下，发消息流程如下：

1、MQ 发送方向 MQ Server 发送 half 消息，MQ Server 标记消息状态为 prepared，此时该消息 MQ 订阅方是无法消费到的

2、MQ Server 将消息持久化成功之后，向发送方 ACK 确认消息已经成功接收

3、发送方开始执行本地事务逻辑

4、发送方根据本地事务执行结果向 MQ Server 提交二次确认，commit 或 rollback

最终步骤：MQ Server 如果收到的是 commit 操作，则将半消息标记为可投递，MQ 订阅方最终将收到该消息；若收到的是 rollback 操作则删除 half 半消息，订阅方将不会接受该消息；如果本地事务执行结果没响应或者超时，则 MQ Server 回查事务状态，具体见步骤 2 的异常情况说明。

（2）异常情况：在断网或者应用重启等异常情况下，图中的 4 提交的二次确认超时未到达 MQ Server，此时的处理逻辑如下：

5、MQ Server 对该消息进行消息回查

6、发送方收到消息回查后，检查该消息的本地事务执行结果

7、发送方根据检查得到的本地事务的最终状态再次提交二次确认

实现流程

事务消息有专门的一个队列 RMQ_SYS_TRANS_HALF_TOPIC，所有的 prepare 消息都先往这里放，当消息收到 Commit 请求后，就将消息转移到真实的 Topic 队列里，供 Consumer 消费，同时向 RMQ_SYS_TRANS_OP_HALF_TOPIC 塞一条消息。