

RocketMQ实战（四）



张丰哲 (/u/cb569cce501b) 已关注

2017.04.30 15:26 字数 1858 阅读 2012 评论 19 喜欢 30

(/u/cb569cce501b)

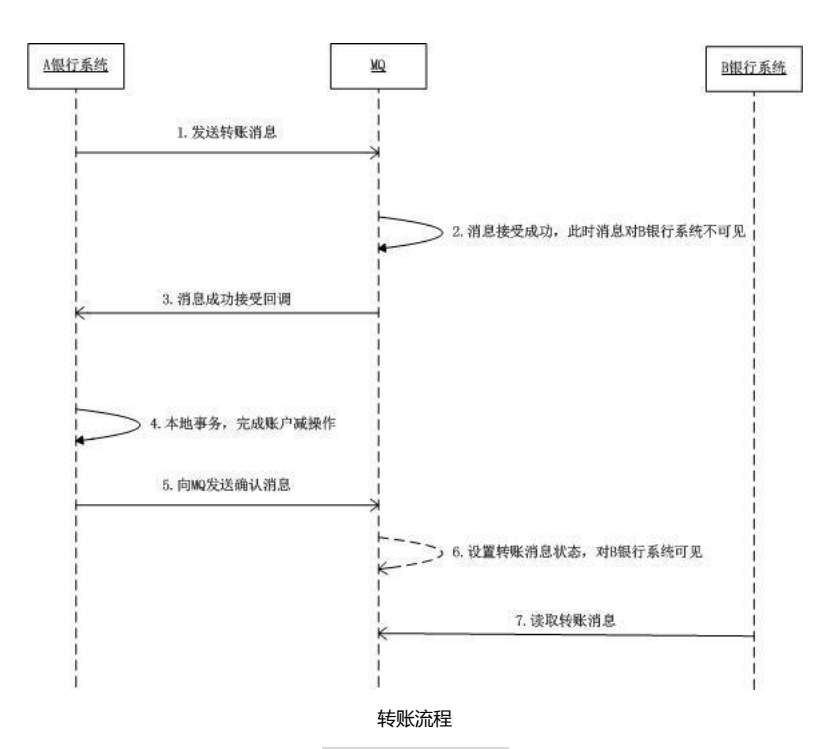
前言

这将是RocketMQ实战系列的最后一篇文章，该系列的文章列表如下：

- 《RocketMQ实战（一）》(<http://www.jianshu.com/p/3afd610a8f7d>)
- 《RocketMQ实战（二）》(<http://www.jianshu.com/p/790d6bc4a1c1>)
- 《RocketMQ实战（三）：分布式事务》(<http://www.jianshu.com/p/53324ea2df92>)

RocketMQ 3.2.6的事务机制

在上一篇博客中，已经知道RocketMQ 3.0.8是支持事务回查机制，但是在RocketMQ 3.2.6中取消了这个功能，下面我们继续以转账功能分析我们自己如何解决这个问题。



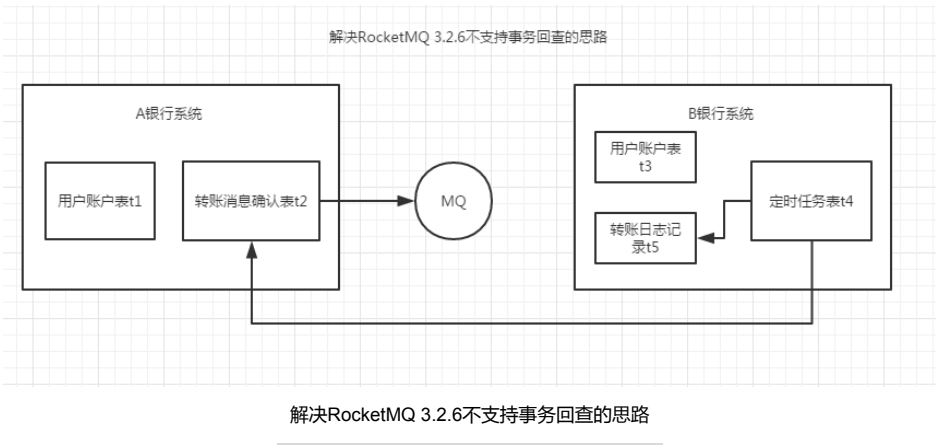
在正常情况下，当然没有问题，如果第五步（向MQ发送确认消息）出现失败，加上RocketMQ 3.2.6版本没有事务回查机制，就会导致这条转账消息，在A银行完成了操作，但是迟迟对B银行系统不可见！

^

+

🔖

🔗



用户U1从A银行系统转账给B银行系统的用户U2的处理过程如下：

第一步：A银行系统生成一条转账消息，以事务消息的方式写入RocketMQ，此时B银行系统不可见这条消息

第二步：写入MQ成功后，回调A银行系统，对T1，T2表进行操作（很显然需要是一个事务）

我们重点关注下T2表，这个表是用来干嘛的呢？每条转账消息都会在T2表中，该表有2个特殊的字段：status，updatetime。（用途会在后文详述）

第三步：完成第二步，接下来发送确认消息给MQ，如果这个确认消息发送成功，那么这条转账消息，将对B银行系统可见。然后B银行系统，会在一个事务中完成对t3，t5的操作。

如果发送确认消息给MQ失败的处理思路：

首先，B银行系统，有一个定时任务（比如说每隔1MIN执行一次），扫描表t5，取得一段时间内的数据，发送给A银行系统。要知道t5中的数据，必然是A银行系统成功处理并发送确认消息成功的转账数据。为什么要发送给A银行系统呢，其实就是为了找到那些发送确认消息失败的转账数据。那么怎么发给A银行系统呢，这个方式比较多，可以考虑在来一个Topic，也可以考虑Netty等。发送给A银行系统，其实就是为了更新t2表的status，updatetime。

这里有一个关键，如何“扫描表t5，取得一段时间内的数据”？这就是t4的作用，在t4中记录一个time字段，每次定时任务启动，先更新时间（比如设定为当前系统时间，设置前的的时间为old），然后扫描出t5中大于这个old时间的转账数据，如此循环往复。

其次，A银行系统，也有一个定时任务（可以根据业务消费能力定，可以大一些），扫描t2表（指定status及updatetime条件），将那些确认消息发送失败的转账消息找出来，更新updatetime并发送给MQ。

这样，我们并没有改动RocketMQ 3.2.6的源码，而是在外围解决了事务回查！

其实到这里，你可以发现RocketMQ的一个特点，就是将生产者和MQ绑定，而不需要特别处理消费者，这是为什么呢？因为消息只要发往RocketMQ成功，那么就意味着成功，为什么这么说？



前面，我们说过，消费者端消费消息只会产生2种错误，第一：timeout，第二：exception。要知道RocketMQ对于超时，会不断重试；对于消费异常，会根据消费端的返回码，会有重试机制保证。也就是，RocketMQ一定会让消息得到消费，如果消费有问题，只能是消费者的问题，而不会是RocketMQ的问题！

Pull Or Push

在前面的博客已经提到，在RocketMQ中Consumer分为2类：Push Consumer、Pull Consumer。以前的例子都是Push Consumer，接下来，为大家介绍下Pull Consumer。

```
MQPullConsumerScheduleService mqPullConsumerScheduleService = new MQPullConsumerScheduleService("PullConsumer");
mqPullConsumerScheduleService.getDefaultMQPullConsumer().setNamesrvAddr("192.168.99.121:9876");
mqPullConsumerScheduleService.setMessageModel(MessageModel.CLUSTERING);
```

通过MQPullConsumerScheduleService进行操作

```
mqPullConsumerScheduleService.registerPullTaskCallback("PullTopic", new PullTaskCallback() {
    @Override
    public void doPullTask(MessageQueue messageQueue, PullTaskContext pullTaskContext) {
        System.out.println(new Date() + "=====拉取数据开始");
        MQPullConsumer mqPullConsumer = pullTaskContext.getPullConsumer();
        try {
            long offset = mqPullConsumer.fetchConsumeOffset(messageQueue, false);
            if (offset < 0) {
                offset = 0;
            }
            PullResult pullResult = mqPullConsumer.pull(messageQueue, "*", offset, 32);
            switch (pullResult.getPullStatus()) {
                case FOUND:
                    List<MessageExt> msgFoundList = pullResult.getMsgFoundList();
                    for (MessageExt messageExt : msgFoundList) {
                        //业务开始消费数据
                        try {
                            System.out.println(new String(messageExt.getBody()));
                        } catch (Exception e) {
                            //TODO:异常处理
                        }
                    }
                    break;
                case NO_MATCHED_MSG:
                case NO_NEW_MSG:
                case OFFSET_ILLEGAL:
                default:
                    break;
            }
            //更新offset 5s 以及 设置下次拉取时间
            mqPullConsumer.updateConsumeOffset(messageQueue, pullResult.getNextBeginOffset());
            pullTaskContext.setPullNextDelayTimeMillis(10 * 1000);
        } catch (Exception e) {
            e.printStackTrace();
        }
        System.out.println(new Date() + "=====拉取数据结束");
    }
});
mqPullConsumerScheduleService.start();
```

注册回调并启动



从表面意思上来看，好像Push是MQ推送给消费者，而Pull是消费者从MQ中拉取；其实本质上都是拉取模式PULL，即消费者从MQ中轮询取得消息。

在Push模式下，Consumer把轮询过程封装了，并注册了MessageListener监听器，取到消息后，唤醒MessageListener监听器中的consumeMessage()进行消费，所以给我们造成了感觉上好像是“推消息”。

在Pull模式下，需要特别注意的是，本质上是从一个Topic下的所有Queue进行拉取，而且每个Queue都必须记录拉取位置，否则会导致重复消费。还有拉取的时间间隔，拉取的大小等等。不过所有的这一切，MQPullConsumerScheduleService都替我们考虑清楚了，提供updateConsumeOffset去更新消费的队列的位置（默认5S同步一次），提供setPullNextDelayTimeMillis设置下次拉取的时间间隔（应该设置的大一些，至少大于5S）。

仔细回想下，对于Push方式的回调 和 Pull方式的回调，还有什么关键区别么？

对于Push而言，不论是基于MessageListenerConcurrently的，还是基于MessageListenerOrderly的，都有返回值的；而Pull的doPullTask的返回值却是void？

这意味，我们需要在pull方式中，注意自己处理每条消息消费的异常情况！

```
Sun Apr 30 07:34:44 CST 2017=====拉取数据开始
Sun Apr 30 07:34:44 CST 2017=====拉取数据开始
Sun Apr 30 07:34:44 CST 2017=====拉取数据开始
Sun Apr 30 07:34:44 CST 2017=====拉取数据开始
Hello RocketMQ 2
Hello RocketMQ 6
Sun Apr 30 07:34:44 CST 2017=====拉取数据结束
Hello RocketMQ 1
Hello RocketMQ 5
Sun Apr 30 07:34:44 CST 2017=====拉取数据结束
Hello RocketMQ 3
Hello RocketMQ 7
Sun Apr 30 07:34:44 CST 2017=====拉取数据结束
Hello RocketMQ 0
Hello RocketMQ 4
Hello RocketMQ 8
Sun Apr 30 07:34:44 CST 2017=====拉取数据结束
Sun Apr 30 07:34:54 CST 2017=====拉取数据开始
Sun Apr 30 07:34:54 CST 2017=====拉取数据开始
Sun Apr 30 07:34:54 CST 2017=====拉取数据开始
Sun Apr 30 07:34:54 CST 2017=====拉取数据开始
Sun Apr 30 07:34:54 CST 2017=====拉取数据结束
Sun Apr 30 07:34:54 CST 2017=====拉取数据结束
Sun Apr 30 07:34:54 CST 2017=====拉取数据结束
Sun Apr 30 07:34:54 CST 2017=====拉取数据结束
```

运行结果

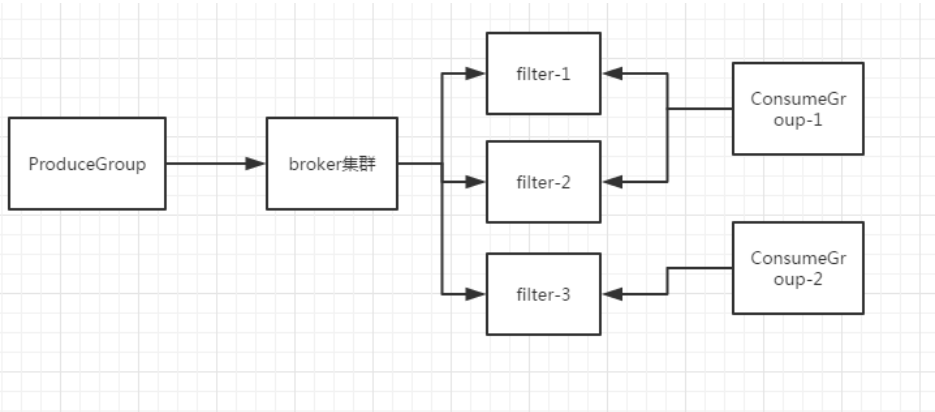
通过运行结果，可以印证上面的观点：为什么每次消费都是4条开始，4条结束呢？因为一个Topic下有4个Queue，而且上面的代码实际上会针对每个Queue开启一个线程去消费！

RocketMQ Filter组件介绍



对于ActiveMQ而言，我们可以通过JMS Selectors机制（就是类似于SQL的语法）来实现过滤，很easy。那么和RocketMQ Filter组件有什么区别呢？

虽然，2者都能实现过滤，但是RocketMQ Filter的性能要更高效些，因为RocketMQ是在broker上将过滤后的数据发往filter，然后消费者直接从filter上取得数据；而ActiveMQ是消费者直接在broker上进行过滤消费！（当然，对于RocketMQ而言，Tag机制已经足够应付日常绝大数的过滤功能，除非你的业务对性能有特别高的要求）



RocketMQ Filter机制

具体怎么做呢？这里我就不演示了，网上有很多例子，这里只说下大致的过程：

第一：broker-xxx.properties中指定filter个数

第二：上传一段JAVA代码，其实就是一个类

到这里，整个RocketMQ实战系列就结束呢，你学到了么，体会到RocketMQ的强大了么？

See u next blog!

📖 日记本 (/nb/10261827) 举报文章 © 著作权归作者所有



张丰哲 (/u/cb569cce501b) ♂

写了 63893 字，被 2144 人关注，获得了 1674 个喜欢

(/u/cb569cce501b)




✓ 已关注

资深Java工程师 51CTO博客【2014-2016】：<http://zhangfengzhe.blog.51cto.com/>


好好学习，天天赞赏~

赞赏支持

👍 喜欢 | 30



更多分享



写下你的评论...

19条评论

只看作者


按喜欢排序 按时间正序 按时间倒序

 春风十里撸代码 (/u/86c917423624)
8楼 · 2017.09.10 12:08
(/u/86c917423624)
不错不错，收藏了。

推荐下，RocketMQ 源码解析 14 篇：<http://www.yunai.me/categories/RocketMQ/?jianshu&401> (<http://www.yunai.me/categories/RocketMQ/?jianshu&401>)

👍 5人赞

💬 回复

 阿布554_ (/u/755483edf2a9)
2楼 · 2017.05.02 22:32
(/u/755483edf2a9)
之前有遇到一个问题，consumerGroupName不同但是订阅的Topic是相同的。。。采用集群消费模式，两个consumer都会对Topic进行消费。。。。这个怎么搞


👍 赞

💬 回复

张丰哲 (/u/cb569cce501b)：之前博客中提及过，不论是对于producer，还是consumer，它们的groupname都是全局唯一的，特别是consumer的groupname将是集群消费模式下消息进行负载均衡的关键。你上面的情况，就是因为2个消费组都订阅了啊。

2017.05.03 15:26 💬 回复

✎ 添加新评论

 风车车 (/u/d53b94c2c297)
3楼 · 2017.05.28 21:43
(/u/d53b94c2c297)
目前可用于生产的版本是哪个版本呢

👍 赞

💬 回复

张丰哲 (/u/cb569cce501b)：我司使用的是3.2.6

2017.05.30 09:21 💬 回复

✎ 添加新评论

 李柱鹏 (/u/80e1ac301699)
4楼 · 2017.06.10 09:58
(/u/80e1ac301699)
博主现在任职于什么公司

👍 赞

💬 回复

张丰哲 (/u/cb569cce501b)：一家互联网公司，~😊

2017.06.11 15:18 💬 回复

✎ 添加新评论

 李柱鹏 (/u/80e1ac301699)
5楼 · 2017.06.10 09:59
(/u/80e1ac301699)
看了你写的文章,写得都十分不错

⬆

+


🔖

🔗

👍 赞 💬 回复

张丰哲 (/u/cb569cce501b)：谢谢啦~😊
2017.06.11 15:18 💬 回复


✍️ 添加新评论

 I_小乌鸦 (/u/1b41feaa555c)
6楼 · 2017.07.24 15:58
(/u/1b41feaa555c)
楼主可以转载吗？

👍 赞 💬 回复

张丰哲 (/u/cb569cce501b)：可以啊，😊
2017.07.25 16:19 💬 回复

✍️ 添加新评论


 ArvinLI (/u/5acdc5f64fda)
7楼 · 2017.07.25 11:50
(/u/5acdc5f64fda)
有个问题，通过定时任务再次发送给mq，是发送什么类型的消息？另外，原来已经在mq那个prepare状态的消息怎么办？

👍 赞 💬 回复

ArvinLI (/u/5acdc5f64fda)：比如已经确认某一条确认消息没有发送成功，是重新把业务数据重新发送一次到mq？还是重新发送确认消息？
2017.07.25 11:52 💬 回复

张丰哲 (/u/cb569cce501b)：@ArvinLI (/u/5acdc5f64fda) 如果确认消息没有发送成功，那么实际上就需要我们自己做对比处理了，在本文是有这方面的叙述的。（如果在发一次业务消息的话，需要注意幂等性，另外如果再次出现确认消息没有发送呢？）😊
2017.07.25 16:23 💬 回复

✍️ 添加新评论

 hahaee (/u/902a81947898)
9楼 · 2017.10.11 11:27
(/u/902a81947898)
这耦合度是怎么考虑的？


👍 赞 💬 回复

张丰哲 (/u/cb569cce501b)：用MQ是可以降低2个系统的耦合的。
2017.10.12 11:24 💬 回复

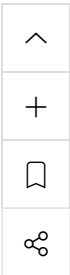
dbwu (/u/fd5635c3c201)：咨询下大佬，出于什么原因，在3.2.6版本中取消了事务回查机制？
2017.10.24 11:46 💬 回复

张丰哲 (/u/cb569cce501b)：@dbwu (/u/fd5635c3c201) 这个我就只能猜测了。（也许是基于商业化的考虑.....）😊
2017.10.28 20:36 💬 回复

✍️ 添加新评论

 IDST (/u/14a7bc26f707)
10楼 · 2017.10.30 17:59
(/u/14a7bc26f707)
大佬，那个autoCreateSubscriptionGroup这个参数怎么手动创建啊


👍 赞 💬 回复





被以下专题收入，发现更多相似内容


+ 收入我的专题


 程序员 (/c/NEt52a?utm_source=desktop&utm_medium=notes-included-collection)


 java进阶干货 (/c/addfce4ca518?utm_source=desktop&utm_medium=notes-included-collection)

 首页投稿 (/c/bDHhpK?utm_source=desktop&utm_medium=notes-included-collection)

 Java学习笔记 (/c/04cb7410c597?utm_source=desktop&utm_medium=notes-included-collection)

 Java 杂谈 (/c/0b39448c4e08?utm_source=desktop&utm_medium=notes-included-collection)

 RocketMQ (/c/613c1ca3873c?utm_source=desktop&utm_medium=notes-included-collection)

 MQ (/c/0c6b140b87f8?utm_source=desktop&utm_medium=notes-included-collection)

展开更多

推荐阅读

更多精彩内容 > (/)

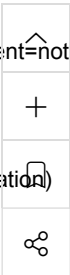
玩转Redis集群(上) (/p/dbc62ed27f03?utm_campaign=maleskine&utm_content=note&utm_source=recommendation) (/p/dbc62ed27f03?utm_campaign=maleskine&utm_content=note&utm_source=recommendation)
这是redis集群介绍的上篇，主要是关于Redis集群的搭建。后续将为大家介绍Redis集群的常用命令、Java操作Redis集群、以及与Spring/Spring MVC的整合...
张丰哲 (/u/cb569cce501b?utm_campaign=maleskine&utm_content=user&utm_medium=pc_all_hots&utm_source=recommendation)

写出我的第一个框架：迷你版Spring MVC (/p/f454662f497e?utm_campaign=maleskine&utm_content=note&utm_source=recommendation) (/p/f454662f497e?utm_campaign=maleskine&utm_content=note&utm_source=recommendation)
你没有看错标题，今天，我将实现我人生中第一个框架，^_^前期准备 我这里要写的是一个迷你版的Spring MVC，我将在一个干净的web工程开始开发，...
张丰哲 (/u/cb569cce501b?utm_campaign=maleskine&utm_content=user&utm_medium=pc_all_hots&utm_source=recommendation)

裸贷、传销、酒精中毒、不合群跳楼：四个大学生的死因 (/p/66a49208a4d7?utm_campaign=maleskine&utm_content=note&utm_source=recommendation) (/p/66a49208a4d7?utm_campaign=maleskine&utm_content=note&utm_source=recommendation)
我迷茫迷茫，游荡游荡。灵魂在这人世间跌跌撞撞，一身是伤。世间的悲剧大都和金钱有关。——三毛 接到朋友催债电话的时候，我还在Dior的官网上刷...
上景Leona (/u/e92d4eef6adb?utm_campaign=maleskine&utm_content=user&utm_medium=pc_all_hots&utm_source=recommendation)

我们家不需要性教育 (/p/4379a577fb26?utm_campaign=maleskine&utm_content=note&utm_source=recommendation) (/p/4379a577fb26?utm_campaign=maleskine&utm_content=note&utm_source=recommendation)
“如果将来哪天我谈了男朋友，我希望我可以把我们两个人之间的故事毫无保留地告诉我妈妈。如果我们发生了矛盾，妈妈可以帮我出谋划策；如果我们做了...
大发的号 (/u/21dec36d8c44?utm_campaign=maleskine&utm_content=user&utm_medium=pc_all_hots&utm_source=recommendation)

平价百搭！如何在淘宝上买到既好看又便宜的包包？ (/p/5bc8f8b5d45d?utm_campaign=maleskine&utm_content=note&utm_source=recommendation) (/p/5bc8f8b5d45d?utm_campaign=maleskine&utm_content=note&utm_source=recommendation)
大家好，我是小丸子~ 最近一个月，有好多同学给我私信，说让我推荐一些好看



的包包，所以呀，这次又花了很长时间给大家整理了一波淘宝上比较知名，有...

小丸子的杂物集 (/u/24bca2bb387d?
utm_campaign=maleskine&utm_content=user&utm_medium=pc_all_hots&utm_source=recommendation)

