图文 80 基于订单数据库同步场景，来分析RocketMQ的顺序消息机制的代码实现

详情 评论

基于订单数据库同步场景，来分析RocketMQ的顺序消息机制的代码实现

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、如何让一个订单的binlog进入一个MessageQueue?

我们先来看第一个代码落地的分析，首先要实现消息顺序，必须让一个订单的binlog都进入一个MessageQueue中，此时我们可以写如下的代码：

```
SendResult sendResult = producer.send(
        message,
        new MessageQueueSelector() {
            @Override
            public MessageQueue select(
                    List<MessageQueue> mqs,
                    Message msg, Object arg) {
                Long orderId = (Long) arg;  //根据订单id选择发送queue
                long index = id % mqs.size(); // 用订单id对MessageQueue数量取模
                return mqs.get((int) index); // 返回一个MessageQueue
            }
        },
        orderId // 这里传入订单id
);
```

在上面的代码片段中，我们可以看到，关键因素就是两个，一个是发送消息的时候传入一个MessageQueueSelector，在里面你要根据订单id和MessageQueue数量去选择这个订单id的数据进入哪个MessageQueue。

同时在发送消息的时候除了带上消息自己以外，还要带上订单id，然后MessageQueueSelector就会根据订单id去选择一个MessageQueue发送过去，这样的话，就可以保证一个订单的多个binlog都会进入一个MessageQueue中去。

2、消费者如何保证按照顺序来获取一个MessageQueue中的消息?

接着我们来看第二块，就是消费者如何按照顺序，来获取一个MessageQueue中的消息?

我们看下面的代码:

```
consumer.registerMessageListener(
        new MessageListenerOrderly() {
            @Override
            public ConsumeOrderlyStatus consumeMessage(
                    List<MessageExt> msgs,
                    ConsumeOrderlyContext context) {
                context.setAutoCommit(true);
                try {
                    for (MessageExt msg : msgs) {
                        // 对有序的消息进行处理
                    }
        return ConsumeOrderlyStatus.SUCCESS;
                } catch(Exception e) {
                    // 如果消息处理有问题
                    // 返回一个状态，让他暂停一会儿再继续处理这批消息
                    return SUSPEND_CURRENT_QUEUE_A_MOMENT;
                }
            }
        }
);
```

在上面的代码中，大家可以注意一下，我们使用的是MessageListenerOrderly这个东西，他里面有Orderly这个名称

也就是说，Consumer会对每一个ConsumeQueue，都仅仅用一个线程来处理其中的消息。

比如对ConsumeQueue01中的订单id=1100的多个binlog，会交给一个线程来按照binlog顺序来依次处理。否则如果ConsumeQueue01中的订单id=1100的多个binlog交给Consumer中的多个线程来处理的话，那还是会有消息乱序的问题。

3、作业：大家自己去实验一下消息的顺序性

今天给大家留的一个作业就是，大家自己部署一个MQ，然后自己构造不同订单id下的有序的binlog数据，然后用上述的方法把他们发送到一个MessageQueue里去，然后在Consumer端观察一下。

是不是你可以拿到每个订单id下的有序的binlog，可以完全按照顺序拉处理?

如果处理失败了，是不是可以返回特殊状态，暂停一会儿再继续处理这批binlog，而不会跳过他们去处理下一批binlog?

希望大家都去测试一下，然后把自己的实验结果分享到评论区里去。

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**）