举个例子，以前做过一个mysql binlog同步的系统，压力还是非常大的，日同步数据要达上亿，就是说数据从一个mysql库原封不动地同步到另一个mysql库里面去（mysql -> mysql）。常见的一点在于说比如大数据team，就需要同步一个mysql库过来，对公司的业务系统的数据做各种复杂的操作。

你在mysql里增删改了一条数据，对应出来增删改3条binlog日志，接着这三条binlog发送到MQ里面，再消费出来依次执行，起码得保证人家是按照顺序来的吧？不然本来是：增加、修改、删除；你愣是换了顺序给执行成删除、修改、增加，不全错了么。

本来这个数据同步过来，应该最后这个数据被删除了；结果你搞错了这个顺序，最后这个数据保留下来了，数据同步就出错了。

先看看顺序会错乱的俩场景：

#### RabbitMQ

一个queue，多个consumer。比如，生产者向RabbitMQ里发送了三条数据，顺序依次是data1/data2/data3，压入的是RabbitMQ的一个内存队列。有三个消费者分别从MQ中消费这三条数据中的一条，结果消费者2先执行完操作，把data2存入数据库，然后是data1/data3。这不明显乱了。



#### Kafka

比如说我们们建了一个topic，有三个partition。生产者在写的时候，其实可以指定一个key，比如说我们指定了某个订单id作为key，那么这个订单相关的数据，一定会被分发到同一个partition中去，而且这个partition中去，而且这个partition中的数据一定是有顺序的。

消费者从partition中取出来数据的时候，也一定是有顺序的。到这里，顺序还是ok的，没有错乱。接着，我们在消费者里可能会搞多个线程来并发处理消息。因为如果消费者是单线程消费处理，而处理比较耗时的话，比如处理一条消息耗时几十ms，那么1秒钟只能处理几十条消息，这吞吐量太低了。而多个线程并发跑的话，顺序可能就乱掉了。



#### 解决方案

RabbitMQ：

拆分多个queue，每个queue一个consumer，就是多一些queue而已，确实是麻烦点；或者就一个queue但是对应一个consumer，然后这个consumer内部用内存队列做排队，然后分发给底层不同的worker来处理。



Kafka：

* 一个topic，一个partition，一个consumer，内部单线程消费，单线程吞吐量太低，一般不会用这个。
* 写N个内存queue，具有相同key的数据都到同一个内存queue；然后对于N个线程，每个线程分别消费一个内存queue即可，这样就能保证顺序性。

