

# Kafka的消息可靠性（防止消息丢失）

原创

刘Java

于 2022-03-24 10:48:54 发布

2736

收藏

12

版权

分类专栏：


Kafka

 文章标签：

kafka

消息队列

消息丢失

 Kafka 专栏收录该内容

0 订阅 5 篇文章 

订阅专栏

当思考消息队列的消息丢失的时候，通常可以从三方面来思考：

- 1. Producer生产者端丢失。
- 2. Broker服务器端丢失。
- 3. Consumer消费者端丢失。

以上三点思考方式是通用的，例如RocketMQ的消息可靠性（防止消息丢失）。

## 1 Producer端

Producer调用send方法发送消息之后会直接返回，消息可能因为网络问题并没有发送过去。

因此我们可以调用get方法，该方法会在Broker返回结果之前一致阻塞，或者注册一个回调函数，这样能够避免同步阻塞，而是等到响应返回的时候自动触发回调函数，该函数中可以检测是否发送到Broker中，如果没有发送成功，则重复发送即可。

另外，为了得到更好的性能，Kafka 支持在生产者客户端使用一个本地buffer累积到一定数量的消息才统一发送，参数是producer.type，默认是sync，改为async即异步发送。如果想要保证数据安全，那么必须使用默认的sync。

另外，还可以设置Producer的acks参数，该参数表示在认为请求完成之前，生产者要求领导者收到的确认数，设置acks=all，表示要求ISR列表里跟Leader保持同步的那些Follower都要把消息同步过去，才能认为这条消息是写入成功了。

这个acks参数用于防止replica机制异步复制还没有成功时Leader选举造成的数据丢失。这个值默认是1，即Leader写入成功即算作成功。

## 2 Broker端

即使消息到达了Broker，要想消息不丢失，必须要保证消息能够持久化到磁盘中才行。

默认情况下，Kafa为了优化数据写入数据，采用了PageCache磁盘缓存技术，默认情况下，数据写入磁盘缓存中，PageCache中的数据会随着内核中 flusher 线程的调度以及对 sync()/fsync() 的调用写回到磁盘，就算进程崩溃，也不用担心数据丢失。

但是如果是系统崩溃、系统掉电等重大事故，那么PageCache中的数据也会丢失。

默认情况下，Broker将消息写入PageCache终之后就会向Producer响应成功写入的消息，为了防止系统崩溃、系统掉电等重大事故时PageCache中的数据丢失，那么将Broker的刷盘策略改为同步刷盘即可，目前Kafka的Broker没有直接提供同步选项，但是有两个broker参数可以选择：

- 1. flush.messages：默认空，表示写入多少条消息之后进行一次fsync刷盘。
- 2. flush.ms：默认空，表示进过多少毫秒之后进行一次fsync刷盘。

## 3 Consumer端

kafka基于offset机制，保证每条消息至少被消费者消费一次，但是不管具体的业务是否成功，只要被消费者拉取到了，就算作消费了一次。另外，还有一个关于enable.auto.commit，即自动提交的参数的常见误解。

enable.auto.commit 为 true，表示设置自动提交。Kafka 会保证在开始调用 poll 方法时，检查是否满足auto.commit.interval.ms条件，如果满足（默认5s），则会首先提交此前所有poll返回的最大offset。注意，并不是到了一定时间就会提交，而是在到了一定时间之后，在下次poll的时候先提交再poll。

从顺序上来说，poll 方法的逻辑是先提交上一批消息的位移，再处理下一批消息，而不是一批数据到了就马上自动提交了，因此它能保证不出现消费丢失的情况。

但自动提交位移的一个问题在于，它可能会出现重复消费。假设拉去了一批消息，还没消费完毕的时候，消费者宕机了，此时这一批的消息并不会被提交，当消费者再次启动的时候，这一批消息会被重新拉取消费，导致重复消费。

将enable.auto.commit设置为false表示使用手动提交，这样的好处就是自己控制是否提交offset，但是这同样无法避免重复消费。比如拉取的消息，消费完毕，在准备提交的时候，消费者宕机了，那么重启时将会再次消费该消息。

所以说，无论是自动提交还是手动提交，kafka都会保证消息至少到达消费者一次，从这方面来说，只会可能导致重复消费，是不存在消费者消息丢失的，除非编写错误的消费代码。

#### 注意：

对于使用spring-kafka依赖的项目并且是使用@KafkaListener机制来消费消息的项目。如果仅仅把enable.auto.commit设置为false，还需要设置AckMode属性，即设置手动提交的模式，一般设置为MANUAL，也就是在代码中手动提交，如果不设置AckMode，那么默认为AckMode为BATCH，也就是poll的一批数据处理完之后由Spring的代码来给你提交。

所以说很多人使用@KafkaListener机制来消费消息，并且仅仅设置了enable.auto.commit为false，然后测试消费的时候，发现虽然在消费者代码中没有手动提交，但是消息还是被提交了，实际上就是因为默认的手动提交模式为BATCH——一次poll数据消费完之后Spring帮助我们提交了，这是一种简化人工提交的方式。