

14 | 幂等生产者和事务生产者是一回事吗？

2019-07-04 胡夕

Kafka核心技术与实战

[进入课程 >](#)



讲述：胡夕

时长 10:12 大小 8.18M



你好，我是胡夕。今天我要和你分享的主题是：Kafka 消息交付可靠性保障以及精确处理一次语义的实现。

所谓的消息交付可靠性保障，是指 Kafka 对 Producer 和 Consumer 要处理的消息提供什么样的承诺。常见的承诺有

以下三种：

最多一次 (at most once)：消息可能会丢失，但绝不会被重复发送。

至少一次 (at least once)：消息不会丢失，但有可能被重复发送。

精确一次 (exactly once)：消息不会丢失，也不会被重复发送。

目前，Kafka 默认提供的交付可靠性保障是第二种，即至少一次。在专栏[第 11 期](#)中，我们说过消息“已提交”的含义，即只有 Broker 成功“提交”消息且 Producer 接到 Broker 的应答才会认为该消息成功发送。不过倘若消息成功“提交”，但 Broker 的应答没有成功发送回 Producer 端（比如网络出现瞬时抖动），那么 Producer 就无法确定消息是否真的提交成功了。因此，它只能选择重试，也就是再次发送相同的消息。这就是 Kafka 默认提供至少一次可靠性保障的原因，不过这会导致消息重复发送。

Kafka 也可以提供最多一次交付保障，只需要让 Producer 禁止重试即可。这样一来，消息要么写入成功，要么写入失败，但绝不会重复发送。我们通常不会希望出现消息丢失的情况，但一些场景里偶发的消息丢失其实是被允许的，相

反，消息重复是绝对要避免的。此时，使用最多一次交付保障就是最恰当的。

无论是至少一次还是最多一次，都不如精确一次来得有吸引力。大部分用户还是希望消息只会被交付一次，这样的话，消息既不会丢失，也不会被重复处理。或者说，即使 Producer 端重复发送了相同的消息，Broker 端也能做到自动去重。在下游 Consumer 看来，消息依然只有一条。

那么问题来了，Kafka 是怎么做到精确一次的呢？简单来说，这是通过两种机制：幂等性 (Idempotence) 和事务 (Transaction)。它们分别是什么机制？两者是一回事吗？要回答这些问题，我们首先来说说什么是幂等性。

什么是幂等性 (Idempotence) ?

“幂等”这个词原是数学领域中的概念，指的是某些操作或函数能够被执行多次，但每次得到的结果都是不变的。我来举几个简单的例子说明一下。比如在乘法运算中，让数字乘以 1 就是一个幂等操作，因为不管你执行多少次这样的运算，结果都是相同的。再比如，取整函数 (floor 和 ceiling) 是幂等函数，那么运行 1 次 floor(3.4) 和 100 次 floor(3.4)，结果是一样的，都是 3。相反地，让一个数加 1 这个操作就不是幂等的，因为执行一次和执行多次的结果必然不同。

在计算机领域中，幂等性的含义稍微有一些不同：

在命令式编程语言（比如 C）中，若一个子程序是幂等的，那它必然不能修改系统状态。这样不管运行这个子程序多少次，与该子程序关联的那部分系统状态保持不变。

在函数式编程语言（比如 Scala 或 Haskell）中，很多纯函数（pure function）天然就是幂等的，它们不执行任何的 side effect。

幂等性有很多好处，**其最大的优势在于我们可以安全地重试任何幂等性操作，反正它们也不会破坏我们的系统状态**。如果是非幂等性操作，我们还需要担心某些操作执行多次对状态的影响，但对于幂等性操作而言，我们根本无需担心此事。

幂等性 Producer

在 Kafka 中，Producer 默认不是幂等性的，但我们可以创建幂等性 Producer。它其实是 0.11.0.0 版本引入的新功能。在此之前，Kafka 向分区发送数据时，可能会出现同一条消息被发送了多次，导致消息重复的情况。在 0.11 之后，指定 Producer 幂等性的方法很简单，仅需要设置一个参数即可，即 `props.put("enable.idempotence" ,
ture)`，或

```
props.put(ProducerConfig.ENABLE_IDEMPOTENCE_CONFIG, true);
```

enable.idempotence 被设置成 true 后，Producer 自动升级成幂等性 Producer，其他所有的代码逻辑都不需要改变。Kafka 自动帮你做消息的重复去重。底层具体的原理很简单，就是经典的用空间去换时间的优化思路，即在 Broker 端多保存一些字段。当 Producer 发送了具有相同字段值的消息后，Broker 能够自动知晓这些消息已经重复了，于是可以在后台默默地把它们“丢弃”掉。当然，实际的实现原理并没有这么简单，但你大致可以这么理解。

看上去，幂等性 Producer 的功能很酷，使用起来也很简单，仅仅设置一个参数就能保证消息不重复了，但实际上，我们必须要了解幂等性 Producer 的作用范围。

首先，它只能保证单分区上的幂等性，即一个幂等性 Producer 能够保证某个主题的一个分区上不出现重复消息，它无法实现多个分区的幂等性。其次，它只能实现单会话上的幂等性，不能实现跨会话的幂等性。这里的会话，你可以理解为 Producer 进程的一次运行。当你重启了 Producer 进程之后，这种幂等性保证就丧失了。

那么你可能会问，如果我想实现多分区以及多会话上的消息无重复，应该怎么做呢？答案就是事务（transaction）或者依赖事务型 Producer。这也是幂等性 Producer 和事务型 Producer 的最大区别！

事务

Kafka 的事务概念类似于我们熟知的数据库提供的事务。在数据库领域，事务提供的安全性保障是经典的 ACID，即原子性（Atomicity）、一致性（Consistency）、隔离性（Isolation）和持久性（Durability）。

当然，在实际场景中各家数据库对 ACID 的实现各不相同。特别是 ACID 本身就是一个有歧义的概念，比如对隔离性的理解。大体来看，隔离性非常自然和必要，但是具体到实现细节就显得不那么精确了。通常来说，**隔离性表明并发执行的事务彼此相互隔离，互不影响**。经典的数据库教科书把隔离性称为可串行化（serializability），即每个事务都假装它是整个数据库中唯一的事务。

提到隔离级别，这种歧义或混乱就更加明显了。很多数据库厂商对于隔离级别的实现都有自己不同的理解，比如有的数据库提供 Snapshot 隔离级别，而在另外一些数据库中，它们被称为可重复读（repeatable read）。好在对于已提交读（read committed）隔离级别的提法，各大主流数据库

厂商都比较统一。所谓的 read committed，指的是当读取数据库时，你只能看到已提交的数据，即无脏读。同时，当写入数据库时，你也只能覆盖掉已提交的数据，即无脏写。

Kafka 自 0.11 版本开始也提供了对事务的支持，目前主要是在 read committed 隔离级别上做事情。它能保证多条消息原子性地写入到目标分区，同时也能保证 Consumer 只能看到事务成功提交的消息。下面我们就来看看 Kafka 中的事务型 Producer。

事务型 Producer


事务型 Producer 能够保证将消息原子性地写入到多个分区中。这批消息要么全部写入成功，要么全部失败。另外，事务型 Producer 也不惧进程的重启。Producer 重启回来后，Kafka 依然保证它们发送消息的精确一次处理。

设置事务型 Producer 的方法也很简单，满足两个要求即可：

和幂等性 Producer 一样，开启 `enable.idempotence = true`。

设置 Producer 端参数 `transactional.id`。最好为其设置一个有意义的名字。

此外，你还需要在 Producer 代码中做一些调整，如这段代码所示：

 复制代码

```
1 producer.initTransactions();
2 try {
3     producer.beginTransaction();
4     producer.send(record1);
5     producer.send(record2);
6     producer.commitTransaction();
7 } catch (KafkaException e) {
8     producer.abortTransaction();
9 }
```

和普通 Producer 代码相比，事务型 Producer 的显著特点是调用了一些事务 API，如 `initTransaction`、`beginTransaction`、`commitTransaction` 和 `abortTransaction`，它们分别对应事务的初始化、事务开始、事务提交以及事务终止。

这段代码能够保证 `Record1` 和 `Record2` 被当作一个事务统一提交到 Kafka，要么它们全部提交成功，要么全部写入失败。实际上即使写入失败，Kafka 也会把它们写入到底层的日志中，也就是说 Consumer 还是会看到这些消息。因此在 Consumer 端，读取事务型 Producer 发送的消息也是

需要一些变更的。修改起来也很简单，设置 `isolation.level` 参数的值即可。当前这个参数有两个取值：

1. `read_uncommitted`：这是默认值，表明 Consumer 能够读取到 Kafka 写入的任何消息，不论事务型 Producer 提交事务还是终止事务，其写入的消息都可以读取。很显然，如果你用了事务型 Producer，那么对应的 Consumer 就不要使用这个值。
2. `read_committed`：表明 Consumer 只会读取事务型 Producer 成功提交事务写入的消息。当然了，它也能看到非事务型 Producer 写入的所有消息。

小结

简单来说，幂等性 Producer 和事务型 Producer 都是 Kafka 社区力图为 Kafka 实现精确一次处理语义所提供的工具，只是它们的作用范围是不同的。幂等性 Producer 只能保证单分区、单会话上的消息幂等性；而事务能够保证跨分区、跨会话间的幂等性。从交付语义上来看，自然是事务型 Producer 能做的更多。

不过，切记天下没有免费的午餐。比起幂等性 Producer，事务型 Producer 的性能要更差，在实际使用过程中，我们需要仔细评估引入事务的开销，切不可无脑地启用事务。

开放讨论

你理解的事务是什么呢？通过今天的分享，你能列举出未来可能应用于你们公司实际业务中的事务型 Producer 使用场景吗？

欢迎写下你的思考和答案，我们一起讨论。如果你觉得有所收获，也欢迎把文章分享给你的朋友。

 极客时间

Kafka 核心技术与实战

全面提升你的 Kafka 实战能力

胡夕
人人贷计算平台部总监
Apache Kafka Contributor



新版升级：点击「 请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 13 | Java生产者是如何管理TCP连接的？

精选留言 (27)

写留言

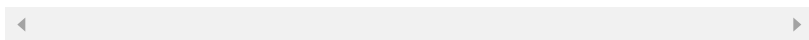


风中花

2019-07-04

我一直认为事务，不到必须时是不用得东西，那么我想知道，胡老师实际中，你们有用到过吗，在一些什么场景下使用？老师可以简单说下吗，谢谢

作者回复: 我们没有使用。事务更多用在Kafka Streams中。如果来实现流处理中的精确一次语义，事务是不可少的。



2

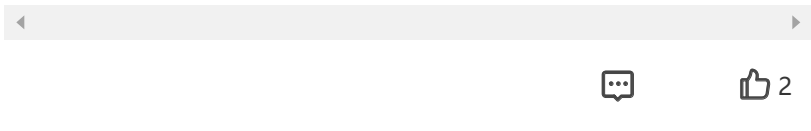


dream

2019-07-04

老师，请问一下，事务型 Producer 可以实现一组消息要么全部写入成功，要么全部失败，但是事务型 Producer 是具体怎么实现多分区以及多会话上的消息无重复的呢？

作者回复: 主要的机制是两阶段提交（2PC）。引入了事务协调器的组件帮助完成分布式事务



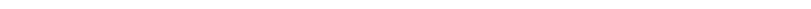
nightmare

2019-07-04

实现上可以用kafka的幂等性来保证单分区单会话的精准一次语义，如果是一批消息，可以路由到同一个分区



👍 2

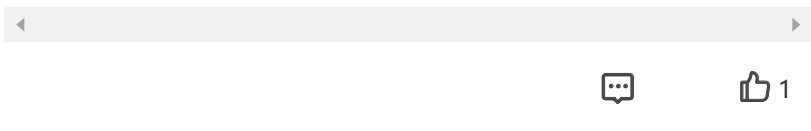


October

2019-07-04

我所理解的kafka事务是这样的：生产者的事务能够保证一条消息仅仅会保存在kafka的某一个分区上，不会出现在多个分区上，另外，能够保证多条消息原子性的发送到多个分区。也就是说它只保证了从producer端到broker端消息不丢失不重复。但对于consumer端，由于偏移...
展开 ▼

作者回复: 嗯嗯，我觉得很有道理：)

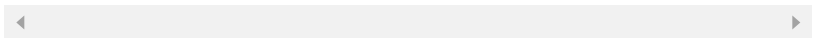


October

2019-07-04

一个幂等性 Producer 能够保证某个主题的一个分区上不出现重复消息，也就是说同一条消息可能出现在不同的分区上，可是producer端没有收到broker的ack，就会重试，重试应该能保证同一条消息分区是不会改变的，为什么这条消息会出现在其他分区呢。

作者回复：“也就是说同一条消息可能出现在不同的分区上” --- 不可能。。。。。



永恒记忆

2019-07-08

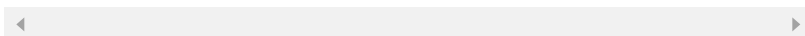
老师好，“当 Producer 发送了具有相同字段值的消息后，Broker 能够自动知晓这些消息已经重复了”，想问下幂等性producer中是按照什么字段做幂等的呢？这个字段可以设置还是一个通用字段？另外为啥事务可以保证只消费一次的语义呢，如果两个不同的事务同时发送相...

展开 ∨

作者回复: 幂等producer是Kafka内部的设计机制，用户无法干预。它保证的是同一条消息只会被保存在Broker上一次。

“两个不同的事务同时发送相同消息” 不确定这是什么意思。如果两个事务是在两个producer上创建的，那么它们

不可能发送相同的消息，至少在kafka看来它们就是不同的消息。

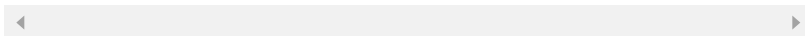


电光火石

2019-07-08

老师有相关的benchmark吗？在开启幂等和事务的情况下，会比不开启的情况分别慢多少，谢谢了！

作者回复: 没有。。。即使有也只是我自己的场景。最好还是结合你实际的情况做一下测试。使用kafka-producer-perf-test.sh就能测试



Xiao

2019-07-07

Kafka的幂等性解决的是producer向broker发送消息的过程。

展开 ✓

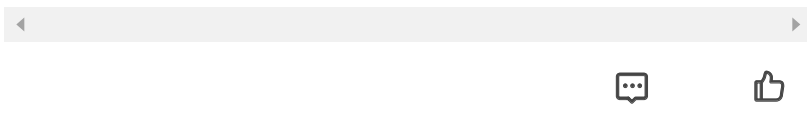




Xiao
2019-07-07

老师，在这种上亿级别消息体量的场景中，精确一次的性能和至少一次的性能差距大么！

作者回复: 不好评估。以实际测试结果为准，只能说肯定是有影响



无菇朋友
2019-07-06

而其他broker没有同步这些字段？

展开 ▾

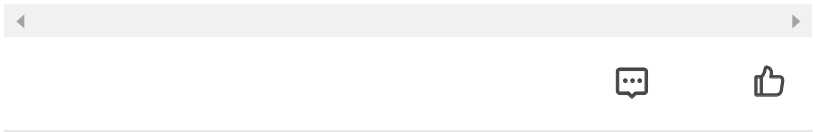


无菇朋友
2019-07-06

胡老师，为啥幂等producer只能保证单个分区没有重复消息，是因为单个分区对应的broker保存了一些必要字段做判断是么？

作者回复: 多个分区的原子性写操作单靠broker或分布式系统节点进程自己参与是无法达成的。想想2PC中的Coordinator角色。必须引入类似于这种第三方的Coordinator居中协调才能完成这种分布式事务——当然我

说的是现有的研究成果，不排除未来有天才实现了这种机制。

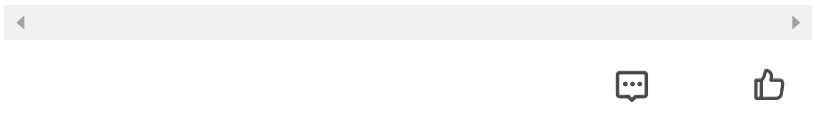


新@青春

2019-07-05

老师，同一个生产者发送同一个消息两次，主题只有一个分区，会产生两消息。我本来的理解是一条？

作者回复: 有可能重复发送造成多条消息被生产出来，但其实它们都是相同的消息（至少内容都是一样的）



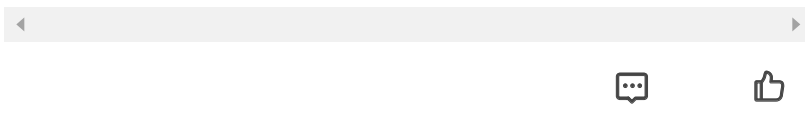
丘壑

2019-07-05

本人测试中发现2个很奇怪的现象，kafka版本0.11.0，window，

- 1、同一个transactional.id"，只能测试一次，第二次再启动程序一直卡在 producer.initTransactions();
- 2、消费端设置：isolation.level=read_committed将...
展开 ∨

作者回复: 是否能够使用更新一点的版本。毕竟0.11才引入, 有一些bug可能也是正常:)



光辉

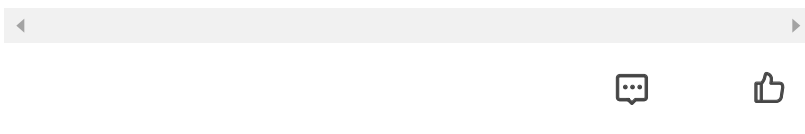
2019-07-05

老师, 你好

幂等性为什么只保证单分区有效? 是因为下一次消息重试指定发送到哪个分区么。如果这样的话是不是可以采用按消息键保序的方式? 这样重试消息还发送到同一个分区。

作者回复: 重启之后标识producer的PID就变化了, broker就不认识了。要想认识就要让broker和producer做更多的事, 也就是事务机制做的那些事。

重试还是发送到同一个分区



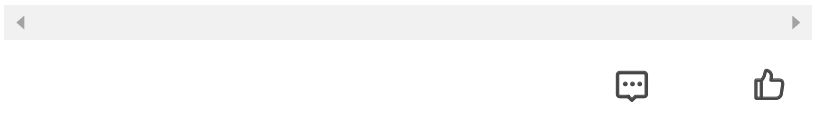
明翼

2019-07-05

老师请教个问题啊, 我们在一个生产环境是日志入

kafka，然后读取kafka的数据入到es里面，由于数据比较多，所以入到kafka的数据可能要过半天到一天才可以处理完，结果发现一个很奇怪的现象就是kafka的入的数据越快，那么入es的速度也越快，本来怀疑是kafka数据...
展开 ✓

作者回复: 如果consumer能够读取page cache中的数据而不是要去磁盘执行物理读，那么可以用上zero copy，速度应该是很快的。你可以看下你的consumer消费时broker的物理读IO，如果很低，大概率走的是page cache。另外如果读kafka很快，es忽高忽低，那是不是应该查一下ES的写入？



JasonZ

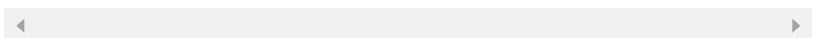
2019-07-05

最多一次 (at most once)：消息可能会丢失，但绝不会被重复发送。

至少一次 (at least once)：消息不会丢失，但有可能被重复发送。

精确一次 (exactly once)：消息不会丢失，也不会被重复发送。这个跟acks是不是有相对应的关系？

作者回复: 关系不大，acks控制的是消息的持久化程度。





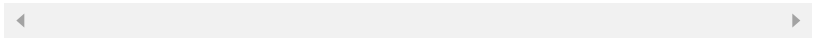
曾軾麟

2019-07-04

```
KafkaProducer<Integer, String> producer = new  
KafkaProducer<>(this.properties());  
    producer.initTransactions();  
    producer.beginTransaction();  
    for (int index = 0; index < 5; index++) {...
```

展开 ✓

作者回复: 请问你用的是哪个版本呢?



1 评论



13761642169

2019-07-04

在金融业务中，会使用事务消息

展开 ✓



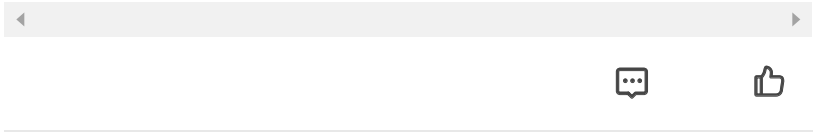
Geek_75b4cd

2019-07-04

事务型消息能保证唯一的效果就是能够保证在broker往

producer发确认消息这一环节做了优化，相对于至少一次来说，老师我理解的有问题吗

作者回复: 事务型消息是什么意思？幂等producer的机制还是在broker端做了一些元数据校验的工作



Bing

2019-07-04

老师，按key做partition，同一key消息路由同一分区，这样是不是也就可以类似全局幂等？

作者回复: 幂等和这个还是有些不同。Kafka幂等主要处理的问题是如何应对消息被发送多次的情况。

