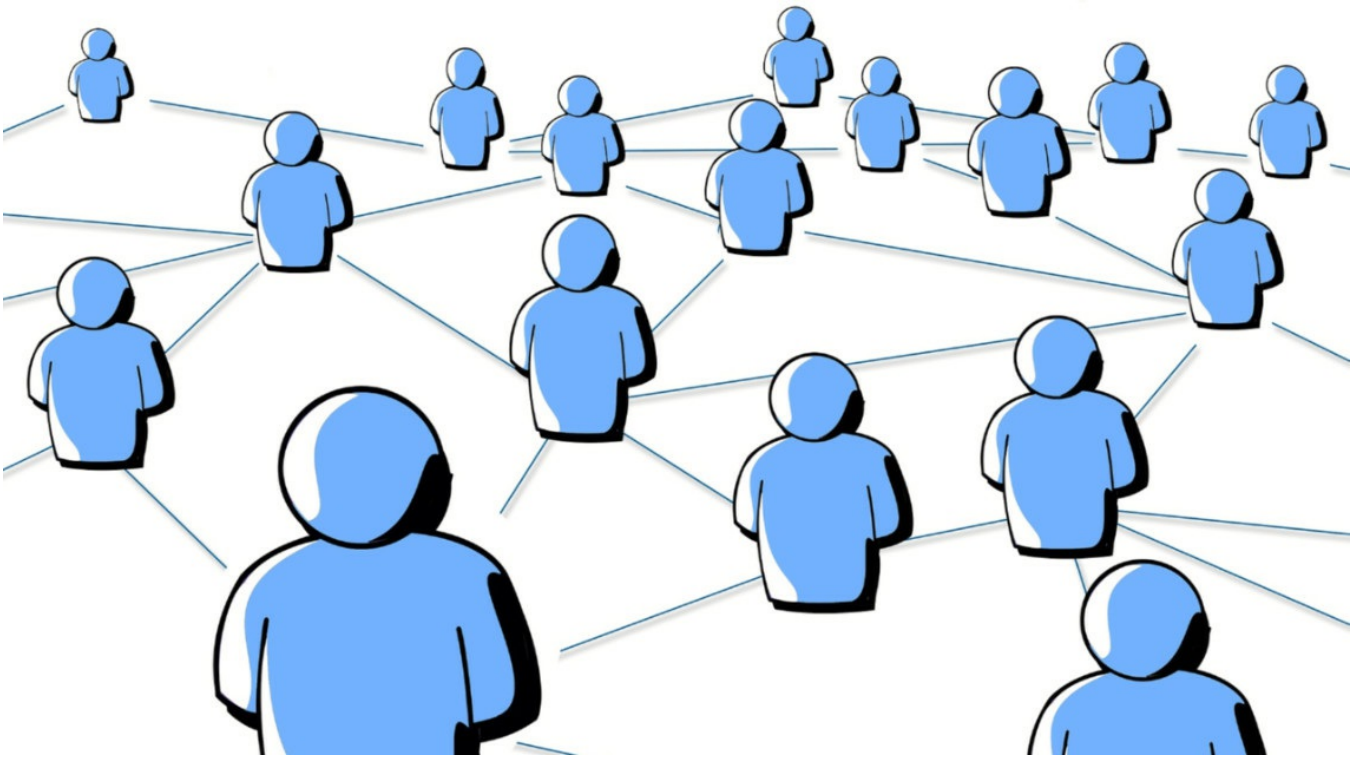
09 | 生产者消息分区机制原理剖析

2019-06-22 胡夕

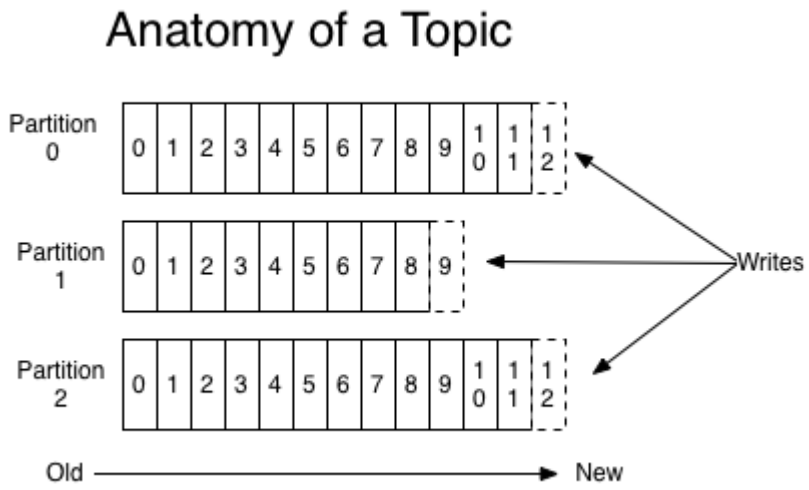


我们在使用Apache Kafka生产和消费消息的时候，肯定是希望能够将数据均匀地分配到所有服务器上。比如很多公司使用Kafka收集应用服务器的日志数据，这种数据都是很多的，特别是对于那种大批量机器组成的集群环境，每分钟产生的日志量都能以GB数，因此如何将这么大的数据量均匀地分配到Kafka的各个Broker上，就成为一个非常重要的问题。

今天我就来和你说说Kafka生产者如何实现这个需求，我会以Java API为例进行分析，但实际上其他语言的实现逻辑也是类似的。

为什么分区？

如果你对Kafka分区（Partition）的概念还不熟悉，可以先返回专栏第2期回顾一下。专栏前面我说过Kafka有主题（Topic）的概念，它是承载真实数据的逻辑容器，而在主题之下还分为若干个分区，也就是说Kafka的消息组织方式实际上是三级结构：主题-分区-消息。主题下的每条消息只会保存在某一个分区中，而不会在多个分区中被保存多份。官网上的这张图非常清晰地展示了Kafka的三级结构，如下所示：



现在我抛出一个问题你可以先思考一下：你觉得为什么Kafka要做这样的设计？为什么使用分区的概念而不是直接使用多个主题呢？

其实分区的作用就是提供负载均衡的能力，或者说对数据进行分区的主要原因，就是为了实现系统的高伸缩性（Scalability）。不同的分区能够被放置到不同节点的机器上，而数据的读写操作也都是针对分区这个粒度而进行的，这样每个节点的机器都能独立地执行各自分区的读写请求处理。并且，我们还可以通过添加新的节点机器来增加整体系统的吞吐量。

实际上分区的概念以及分区数据库早在1980年就已经有大牛们在做了，比如那时候有个叫Teradata的数据库就引入了分区的概念。

值得注意的是，不同的分布式系统对分区的叫法也不尽相同。比如在Kafka中叫分区，在MongoDB和Elasticsearch中就叫分片Shard，而在HBase中则叫Region，在Cassandra中又被称作vnode。从表面看起来它们实现原理可能不尽相同，但对底层分区（Partitioning）的整体思想却从未改变。

除了提供负载均衡这种最核心的功能之外，利用分区也可以实现其他一些业务级别的需求，比如实现业务级别的消息顺序的问题，这一点我今天也会分享一个具体的案例来说明。

都有哪些分区策略？

下面我们说说Kafka生产者的分区策略。**所谓分区策略是决定生产者将消息发送到哪个分区的算法。**Kafka为我们提供了默认的分区策略，同时它也支持你自定义分区策略。

如果要自定义分区策略，你需要显式地配置生产者端的参数partitioner.class。这个参数该怎么设定呢？方法很简单，在编写生产者程序时，你可以编写一个具体的类实现org.apache.kafka.clients.producer.Partitioner接口。这个接口也很简单，只定义了两个方法：partition()和close()，通常你只需要实现最重要的partition方法。我们来看看这个方法的方法签名：

```
int partition(String topic, Object key, byte[] keyBytes, Object value, byte[] valueBytes, Cluster cluster);
```

这里的topic、key、keyBytes、value和valueBytes都属于消息数据，cluster则是集群信息（比如当前Kafka集群共有多少主题、多少Broker等）。Kafka给你这么多信息，就是希望让你能够充分地利用这些信息对消息进行分区，计算出它要被发送到哪个分区中。只要你自己的实现类定义好了partition方法，同时设置partitioner.class参数为你自己实现类的Full Qualified Name，那么生产者程序就会按照你的代码逻辑对消息进行分区。虽说可以有无数种分区的可能，但比较常见的分区策略也就那么几种，下面我来详细介绍一下。

轮询策略

也称Round-robin策略，即顺序分配。比如一个主题下有3个分区，那么第一条消息被发送到分区0，第二条被发送到分区1，第三条被发送到分区2，以此类推。当生产第4条消息时又会重新开始，即将其分配到分区0，就像下面这张图展示的那样。

这就是所谓的轮询策略。轮询策略是Kafka Java生产者API默认提供的分区策略。如果你未指定partitioner.class参数，那么你的生产者程序会按照轮询的方式在主题的所有分区间均匀地“码放”消息。

轮询策略有非常优秀的负载均衡表现，它总是能保证消息最大限度地被平均分配到所有分区上，故默认情况下它是最合理的分区策略，也是我们最常用的分区策略之一。

随机策略

也称Randomness策略。所谓随机就是我们随意地将消息放置到任意一个分区上，如下面这张图所示。

如果要实现随机策略版的partition方法，很简单，只需要两行代码即可：

```
List<PartitionInfo> partitions = cluster.partitionsForTopic(topic);
return ThreadLocalRandom.current().nextInt(partitions.size());
```

先计算出该主题总的分区数，然后随机地返回一个小于它的正整数。

本质上看随机策略也是力求将数据均匀地打散到各个分区，但从实际表现来看，它要逊于轮询策略，所以**如果追求数据的均匀分布，还是使用轮询策略比较好**。事实上，随机策略是老版本生产者使用的分区策略，在新版本中已经改为轮询了。

按消息键保序策略

也称Key-ordering策略。有点尴尬的是，这个名词是我自己编的，Kafka官网上并无这样的提法。

Kafka允许为每条消息定义消息键，简称为Key。这个Key的作用非常大，它可以是一个有着明确业务含义的字符串，比如客户代码、部门编号或是业务ID等；也可以用来表征消息元数据。特别是在Kafka不支持时间戳的年代，在一些场景中，工程师们都是直接将消息创建时间封装进Key里面的。一旦消息被定义了Key，那么你就可以保证同一个Key的所有消息都进入到相同的分区里面，由于每个分区下的消息处理都是有顺序的，故这个策略被称为按消息键保序策略，如下图所示。

实现这个策略的partition方法同样简单，只需要下面两行代码即可：

```
List<PartitionInfo> partitions = cluster.partitionsForTopic(topic);
return Math.abs(key.hashCode()) % partitions.size();
```

前面提到的Kafka默认分区策略实际上同时实现了两种策略：如果指定了Key，那么默认实现按消息键保序策略；如果没有指定Key，则使用轮询策略。

在你了解了Kafka默认的分区策略之后，我来给你讲一个真实的案例，希望能加强你对分区策略重要性的理解。

我曾经给一个国企进行过Kafka培训，当时碰到的一个问题就是如何实现消息的顺序问题。这家企业发送的Kafka的消息是有因果关系的，故处理因果关系也必须要保证有序性，否则先处理了“果”后处理“因”必然造成业务上的混乱。

当时那家企业的做法是给Kafka主题设置单分区，也就是1个分区。这样所有的消息都只在这一个分区内读写，因此保证了全局的顺序性。这样做虽然实现了因果关系的顺序性，但也丧失了Kafka多分区带来的高吞吐量和负载均衡的优势。

后来经过了解和调研，我发现这种具有因果关系的消息都有一定的特点，比如在消息体中都封装了固定的标志位，后来我就建议他们对此标志位设定专门的分区策略，保证同一标志位的所有消息都发送到同一分区，这样既可以保证分区内的消息顺序，也可以享受到多分区带来的性能红利。

这种基于个别字段的分区策略本质上就是按消息键保序的思想，其实更加合适的做法是把标志位数据提取出来统一放到Key中，这样更加符合Kafka的设计思想。经过改造之后，这个企业的消息处理吞吐量一下提升了40多倍，从这个案例你也可以看到自定制分区策略的效果可见一斑。

其他分区策略

上面这几种分区策略都是比较基础的策略，除此之外你还能想到哪些有实际用途的分区策略？其实还有一种比较常见的，即所谓的基于地理位置的分区策略。当然这种策略一般只针对那些大规模的Kafka集群，特别是跨城市、跨国家甚至是跨大洲的集群。

我就拿“极客时间”举个例子吧，假设极客时间的所有服务都部署在北京的一个机房（这里我假设它是自建机房，不考虑公有云方案。其实即使是公有云，实现逻辑也差不多），现在极客时间考虑在南方找个城市（比如广州）再创建一个机房；另外从两个机房中选取一部分机器共同组成一个大的Kafka集群。显然，这个集群中必然有一部分机器在北京，另外一部分机器在广州。

假设极客时间计划为每个新注册用户提供一份注册礼品，比如南方的用户注册极客时间可以免费得到一碗“甜豆腐脑”，而北方的新注册用户可以得到一碗“咸豆腐脑”。如果用Kafka来实现则很简单，只需要创建一个双分区的主题，然后再创建两个消费者程序分别处理南北方注册用户逻辑即可。

但问题是你需要把南北方注册用户的注册消息正确地发送到位于南北方的不同机房中，因为处理这些消息的消费者程序只可能在某一个机房中启动着。换句话说，送甜豆腐脑的消费者程序只在广州机房启动着，而送咸豆腐脑的程序只在北京的机房中，如果你向广州机房中的Broker发送北方注册用户的消息，那么这个用户将无法得到礼品！

此时我们就可以根据Broker所在的IP地址实现定制化的分区策略。比如下面这段代码：

```
List<PartitionInfo> partitions = cluster.partitionsForTopic(topic);
return partitions.stream().filter(p -> isSouth(p.leader().host())).map(PartitionInfo::partition).findAny().get();
```

我们可以从所有分区中找出那些Leader副本在南方的所有分区，然后随机挑选一个进行消息发送。

小结

今天我们讨论了Kafka生产者消息分区的机制以及常见的几种分区策略。切记分区是实现负载均衡以及高吞吐量的关键，故在生产者这一端就要仔细盘算合适的分区策略，避免造成消息数据的“倾斜”，使得某些分区成为性能瓶颈，这样极易引发下游数据消费的性能下降。

开放讨论

在你的生产环境中使用最多的是哪种消息分区策略？实际在使用过程中遇到过哪些“坑”？

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。

精选留言

kevin 40

之前做车辆实时定位(汽车每10s上传一次报文)显示的时候，发现地图显示车辆会突然退回去，开始排查怀疑是后端处理的逻辑问题导致的，但是后台保证了一台车只被一个线程处理，理论上不会出现这种情况；于是猜测是不是程序接收到消息的时候时间序就已经乱了，查阅了kafka相关资料，发现kafka同一个topic是无法保证数据的顺序性的，但是同一个partition中的数据是有顺序的；根据这个查看了接入端的代码(也就是kafka的生产者)，发现是按照kafka的默认分区策略(topic有10个分区，3个副本)发送的；于是将此处发送策略改为按照key(车辆VIN码)进行分区，后面车辆的定位显示就正常了。

2019-06-22

邋遢的流浪剑客 6

之前学习Kafka的时候确实有点忽略了生产者分区策略这一块内容，感谢老师的分享，特意去看了一下源码，Java客户端默认的生产者分区策略的实现类为org.apache.kafka.clients.producer.internals.DefaultPartitioner。默认策略为：如果指定了partition就直接发送到该分区；如果没有指定分区但是指定了key，就按照key的hash值选择分区；如果partition和key都没有指定就使用轮询策略。而且如果key不为null，那么计算得到的分区号会是所有分区中的任意一个；如果key为null并且有可用分区时，那么计算得到的分区号仅为可用分区中的任意一个

2019-06-22

mickle 5

分区应该是实现 org.apache.kafka.clients.producer.Partitioner 接口吧，老师写的Producer

2019-06-22

作者回复

嗯嗯，确实是写错了，抱歉~~

2019-06-24

QQ怪 3

我们公司一直使用单个分区保持消息顺序性，看了老师分享的东西收益很多啊，准备回去好好分析改造下

2019-06-22

作者回复

我们公司之前也有一个业务是单分区，要保证全局顺序。后来发现其实使用key+多分区也可以实现。反正保证同一批因果依赖的消息分到一个分区就可以

2019-06-24

Adol 2

老师好，在消息重试的时候，分区策略会重新再计算一次吗？比如一开始选择到5号分区，但是5号分区有问题导致重试，重试的时候可以重试发送到别的分区上吗？

2019-06-23

作者回复

不会的。消息重试只是简单地将消息重新发送到之前的分区

2019-06-24

leaning_人生 1

kafka的主题只有一级、像mq可以进行主题分层：一级主题、二级主题。kafka为何这样设计？

2019-06-26

作者回复

嗯嗯，这个不太清楚，不敢妄言。。。

2019-06-26

蒋良权 1

老师，我有个疑问，一个topic三个分区与三个单分区的topic在吞吐量以及负载均衡上有什么区别吗？

2019-06-25

作者回复

感觉没什么区别，只是缓存中的微弱区别罢了。

2019-06-26

EricJones 1

分区实现负载均衡提高吞吐量，一台机器多个分区也会有负载均衡效果？也会提高吞吐量？如果会那我一台机器一个kafka 分多少分区合适？我看有人一台机器一个kafka也分了五六个分区。这样做有什么好处？

2019-06-24

作者回复

通常1台broker上有多个分区依然能提升TPS，毕竟单个分区消耗不掉大部分的系统资源。当然一切以实际测试结果为准。

2019-06-25

October 1

老师，在看kafka-client生产者默认分区源码时，看到了cluster.partitionsForTopic和cluster.availablePartitionsForTopic，请问什么时候分区是available，什么时候是不unavailable的？

2019-06-23

作者回复

分区没有leader的时候就是unavailable了。某些操作会导致瞬间没有leader，比如分区reassign、换leader等

2019-06-24

Geek_75b4cd 1

广州机房怎么消费广州partition的数据，consumer如何指定消费的partition。这个能讲下吗

2019-06-22

作者回复

使用这个方法：consumer.assign()直接消息指定分区

2019-06-24

mickle 1

老师的示例代码用的是kafka-clients的什么版本，Producer接口包路径不一样，而且接口类定义了很多方法

2019-06-22

作者回复

嗯嗯，写错了，应该是Partitioner接口

2019-06-24

Alpha 1

自定义分区策略应该是实现Partitioner接口，不是Producer接口

2019-06-22

作者回复

嗯嗯，确实写错了，感谢您的反馈：）

2019-06-24

WL 1

老师能不能有空能不能讲讲kafka和rocketMQ的对比, 我用下来感觉整体挺像的但是具体使用场景和性能优劣方面还是有点不知道该使用选择, 谢谢.

2019-06-22

作者回复

之前也曾经回答过，不一定客观，姑且听之。在我看来RocketMQ与Kafka的主要区别 ： 1. Kafka吞吐量大，多是面向大数据场景。RocketMQ吞吐量也很强， 不过它号称是金融业务级的消息中间件，也就是说可以用于实际的业务系统；2. RocketMQ毕竟是阿里出品，在国内技术支

持力度要比Kafka强；3. Kafka现在主要发力Streaming，RocketMQ在流处理这块表现如何我不太清楚，至少streaming不是它现阶段的主要卖点。

其他方面这两者确实都差不多~~

2019-06-24

玉剑冰锋 1

老师您好，请教个问题:我们生产是三台kafka，Filebeat采集端（目前有几百台）采集日志发送至kafka，由于当时对kafka分区规划不好，Topic的分区都是16，副本是1，这样造成的结果就是数据倾斜，时不时需要迁移Partition，线上迁移很容易造成I/O的问题，导致kafka异常。想问问老师分区数量设置有什么依据吗？另外针对上述情况后续有没有其他好的办法？

2019-06-22

作者回复

我有点没太明白，为什么会造成数据倾斜，你用了什么分区策略了吗？

2019-06-24

Geek_817ea4 0

老师，怎么理解topic消息无序性，分区内消息有序性。

2019-07-01

thicktao 0

老师，我想问一下，消费者要获取分区号是不是只能从zookeeper中获取呢，还有没有其他方式？

2019-06-30

作者回复

可以用过API的方式从Broker端获取，比如KafkaConsumer.partitionsFor方法

2019-07-01

Kun3375 0

有几个疑惑...

1.请问对于事务消息的幂等，broker 端的实现和也和普通消息一样吗？

2.如果这样的话是如何做到多分区多会话幂等的呢？

3.对于实践来说，事务的幂等是否还需要手动开启 enable.idempotence？

2019-06-28

作者回复

1. 引入了新类型的消息来支持事务，如transaction marker消息

2. 主要依赖transaction coordinator组件使用两阶段提交来保证多分区的原子性写入

3. 需要开启

2019-07-01

lyyjason 0

用key来进行分区，保证消息的顺序消费，这招太实用了，受教了！

2019-06-28

没啥好看的

0

如果我用key保序这种方式，key有2w个，搞2w个分区，这么做合适吗

2019-06-27

作者回复

不太合适。超多分区导致磁盘性能也会下降。key有2w个，为什么分区也要有2w个呢？

2019-06-28

开水

0

老师，没太想明白。如果多线程producer发送消息，虽然是按照一key一partition策略，但是不同线程发送的时间可能就是乱序的，那怎么能保证消息全局有序呢？

2019-06-27

作者回复

如果是多线程发送，本身这些线程发送的消息也是并发的，不可能有先后顺序了啊

2019-06-28