02-一篇文章带你快速搞定Kafka术语

你好，我是胡夕。今天我们正式开启Apache Kafka学习之旅。

在Kafka的世界中有很多概念和术语是需要你提前理解并熟练掌握的，这对于后面你深入学习Kafka各种功能和特性将大有裨益。下面我来盘点一下Kafka的各种术语。

在专栏的第一期我说过Kafka属于分布式的消息引擎系统，它的主要功能是提供一套完备的消息发布与订阅解决方案。在Kafka中，发布订阅的对象是主题（Topic），你可以为每个业务、每个应用甚至是每类数据都创建专属的主题。

向主题发布消息的客户端应用程序称为生产者（Producer），生产者程序通常持续不断地向一个或多个主题发送消息，而订阅这些主题消息的客户端应用程序就被称为消费者（Consumer）。和生产者类似，消费者也能够同时订阅多个主题的消息。我们把生产者和消费者统称为客户端（Clients）。你可以同时运行多个生产者和消费者实例，这些实例会不断地向Kafka集群中的多个主题生产和消费消息。

有客户端自然也就有服务器端。Kafka的服务器端由被称为Broker的服务进程构成，即一个Kafka集群由多个Broker组成，Broker负责接收和处理客户端发送过来的请求，以及对消息进行持久化。虽然多个Broker进程能够运行在同一台机器上，但更常见的做法是将不同的Broker分散运行在不同的机器上，这样如果集群中某一台机器宕机，即使在它上面运行的所有Broker进程都挂掉了，其他机器上的Broker也依然能够对外提供服务。这其实就是Kafka提供高可用的手段之一。

实现高可用的另一个手段就是备份机制（Replication）。备份的思想很简单，就是把相同的数据拷贝到多台机器上，而这些相同的数据拷贝在Kafka中被称为副本（Replica）。好吧，其实在整个分布式系统里好像都叫这个名字。副本的数量是可以配置的，这些副本保存着相同的数据，但却有不同的角色和作用。Kafka定义了两类副本：领导者副本（Leader Replica）和追随者副本（Follower Replica）。前者对外提供服务，这里的对外指的是与客户端程序进行交互；而后者只是被动地追随领导者副本而已，不能与外界进行交互。当然了，你可能知道在很多其他系统中追随者副本是可以对外提供服务的，比如MySQL的从库是可以处理读操作的，但是在Kafka中追随者副本不会对外提供服务。对了，一个有意思的事情是现在已经不提倡使用Master-Slave来指代这种主从关系了，毕竟Slave有奴隶的意思，在美国这种严禁种族歧视的国度，这种表述有点政治不正确了，所以目前大部分的系统都改成Leader-Follower了。

副本的工作机制也很简单：生产者总是向领导者副本写消息；而消费者总是从领导者副本读消息。至于追随者副本，它只做一件事：向领导者副本发送请求，请求领导者把最新生产的消息发给它，这样它能保持与领导者的同步。

虽然有了副本机制可以保证数据的持久化或消息不丢失，但没有解决伸缩性的问题。伸缩性即所谓的Scalability，是分布式系统中非常重要且必须要谨慎对待的问题。什么是伸缩性呢？我们拿副本来说，虽然现在有了领导者副本和追随者副本，但倘若领导者副本积累了太多的数据以至于单台Broker机器都无法容纳了，此时应该怎么办呢？一个很自然的想法就是，能否把数据分割成多份保存在不同的Broker上？如果你就是这么想的，那么恭喜你，Kafka就是这么设计的。

这种机制就是所谓的分区（Partitioning）。如果你了解其他分布式系统，你可能听说过分片、分区域等提法，比如MongoDB和Elasticsearch中的Sharding、HBase中的Region，其实它们都是相同的原理，只是Partitioning是最标准的名称。

Kafka中的分区机制指的是将每个主题划分成多个分区（Partition），每个分区是一组有序的消息日志。生

产者生产的每条消息只会被发送到一个分区中，也就是说如果向一个双分区的主题发送一条消息，这条消息要么在分区0中，要么在分区1中。如你所见，Kafka的分区编号是从0开始的，如果Topic有100个分区，那么它们的分区号就是从0到99。

讲到这里，你可能有这样的疑问：刚才提到的副本如何与这里的分区联系在一起呢？实际上，副本是在分区这个层级定义的。每个分区下可以配置若干个副本，其中只能有1个领导者副本和N-1个追随者副本。生产者向分区写入消息，每条消息在分区中的位置信息由一个叫位移（Offset）的数据来表征。分区位移总是从0开始，假设一个生产者向一个空分区写入了10条消息，那么这10条消息的位移依次是0、1、2、…、9。

至此我们能够完整地串联起Kafka的三层消息架构：

- 第一层是主题层，每个主题可以配置M个分区，而每个分区又可以配置N个副本。
- 第二层是分区层，每个分区的N个副本中只能有一个充当领导者角色，对外提供服务；其他N-1个副本是追随者副本，只是提供数据冗余之用。
- 第三层是消息层，分区中包含若干条消息，每条消息的位移从0开始，依次递增。
- 最后，客户端程序只能与分区的领导者副本进行交互。

讲完了消息层次，我们来说说Kafka Broker是如何持久化数据的。总的来说，Kafka使用消息日志（Log）来保存数据，一个日志就是磁盘上一个只能追加写（Append-only）消息的物理文件。因为只能追加写入，故避免了缓慢的随机I/O操作，改为性能较好的顺序I/O写操作，这也是实现Kafka高吞吐量特性的一个重要手段。不过如果你不停地向一个日志写入消息，最终也会耗尽所有的磁盘空间，因此Kafka必然要定期地删除消息以回收磁盘。怎么删除呢？简单来说就是通过日志段（Log Segment）机制。在Kafka底层，一个日志又近一步细分成多个日志段，消息被追加写到当前最新的日志段中，当写满了一个日志段后，Kafka会自动切分出一个新的日志段，并将老的日志段封存起来。Kafka在后台还有定时任务会定期地检查老的日志段是否能够被删除，从而实现回收磁盘空间的目的。

这里再重点说说消费者。在专栏的第一期中我提到过两种消息模型，即点对点模型（Peer to Peer，P2P）和发布订阅模型。这里面的点对点指的是同一条消息只能被下游的一个消费者消费，其他消费者则不能染指。在Kafka中实现这种P2P模型的方法就是引入了消费者组（Consumer Group）。所谓的消费者组，指的是多个消费者实例共同组成一个组来消费一组主题。这组主题中的每个分区都只会被组内的一个消费者实例消费，其他消费者实例不能消费它。为什么要引入消费者组呢？主要是为了提升消费者端的吞吐量。多个消费者实例同时消费，加速整个消费端的吞吐量（TPS）。我会在专栏的后面详细介绍消费者组机制，所以现在你只需要了解消费者组是做什么的即可。另外这里的消费者实例可以是运行消费者应用的进程，也可以是一个线程，它们都称为一个消费者实例（Consumer Instance）。

消费者组里面的所有消费者实例不仅“瓜分”订阅主题的数据，而且更酷的是它们还能彼此协助。假设组内某个实例挂掉了，Kafka能够自动检测到，然后把这个Failed实例之前负责的分区转移给其他活着的消费者。这个过程就是Kafka中大名鼎鼎的“重平衡”（Rebalance）。嗯，其实既是大名鼎鼎，也是臭名昭著，因为由重平衡引发的消费者问题比比皆是。事实上，目前很多重平衡的Bug社区都无力解决。

每个消费者在消费消息的过程中必然需要有个字段记录它当前消费到了分区的哪个位置上，这个字段就是消费者位移（Consumer Offset）。注意，这和上面所说的位移完全不是一个概念。上面的“位移”表征的是分区内的消息位置，它是不变的，即一旦消息被成功写入到一个分区上，它的位移值就是固定的了。而消费者位移则不同，它可能是随时变化的，毕竟它是消费者消费进度的指示器嘛。另外每个消费者有着自己的消费者位移，因此一定要区分这两类位移的区别。我个人把消息在分区中的位移称为分区位移，而把消费者端

的位移称为消费者位移。

小结

我来总结一下今天提到的所有名词术语：

- 消息：Record。Kafka是消息引擎嘛，这里的消息就是指Kafka处理的主要对象。
- 主题：Topic。主题是承载消息的逻辑容器，在实际使用中多用来区分具体的业务。
- 分区：Partition。一个有序不变的消息序列。每个主题下可以有多个分区。
- 消息位移：Offset。表示分区中每条消息的位置信息，是一个单调递增且不变的值。
- 副本：Replica。Kafka中同一条消息能够被拷贝到多个地方以提供数据冗余，这些地方就是所谓的副本。副本还分为领导者副本和追随者副本，各自有不同的角色划分。副本是在分区层级下的，即每个分区可配置多个副本实现高可用。
- 生产者：Producer。向主题发布新消息的应用程序。
- 消费者：Consumer。从主题订阅新消息的应用程序。
- 消费者位移：Consumer Offset。表征消费者消费进度，每个消费者都有自己的消费者位移。
- 消费者组：Consumer Group。多个消费者实例共同组成的一个组，同时消费多个分区以实现高吞吐。
- 重平衡：Rebalance。消费者组内某个消费者实例挂掉后，其他消费者实例自动重新分配订阅主题分区的过程。Rebalance是Kafka消费者端实现高可用的重要手段。

开放讨论

请思考一下为什么Kafka不像MySQL那样允许追随者副本对外提供读服务？

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：

- we 2019-06-06 08:34:19
  老师 这个结构，为什么不用图表示。 [10赞]

- 明翼 2019-06-06 08:38:25
  看了不少留言，大有裨益，算是总结。不从follower读几个原因：1，kafka的分区已经让读是从多个broker读从而负载均衡，不是MySQL的主从，压力都在主上；2，kafka保存的数据和数据库的性质有实质的区别就是数据具有消费的概念，是流数据，kafka是消息队列，所以消费需要位移，而数据库是实体数据不存在这个概念，如果从kafka的follower读，消费端offset控制更复杂；3，生产者来说，kafka可以通过配置来控制是否等待follower对消息确认的，如果从上面读，也需要所有的follower都确认了才可以回复生产者，造成性能下降，如果follower出问题了也不好处理 [4赞]

- 骨汤鸡蛋面 2019-06-06 08:46:16
  建议在文章中使用topic、consumer 等代替 主题、消费者实例等表述，对了解kafka的人来说，更自然一点 [3赞]

  作者回复2019-06-06 09:12:06
  嗯嗯，好的：)

- dbo 2019-06-06 08:13:26
  Myaql中从追随者读取数据对server和client都没有影响，而Kafka中从追随者读取消息意味着消费了数据，需要标记该数据被消费了，涉及到做一些进度维护的操作，多个消费实例做这些操作复杂性比较高，如果可以从追随者读也可能会牺牲性能，这是我的理解，请老师指正。 [3赞]

  作者回复2019-06-06 08:33:16
  我个人认为维护成本不高。Kafka中消费进度由clients端来操作，即消费者来决定什么时候提交位移，而且是提交到专属的topic上，与副本本身关联不大。实际上社区最近正在讨论是否允许follower副本提供读服务。不过我同意的是，follower副本提供读服务后会推高follower所在broker的磁盘读IO

- (´田ω田`) 2019-06-06 01:52:37
  1、主题中的每个分区都只会被组内的一个消费者实例消费，其他消费者实例不能消费它。
  2、假设组内某个实例挂掉了，Kafka 能够自动检测到，然后把这个 Failed 实例之前负责的分区转移给其他活着的消费者。

  意思是1个分区只能同时被1个消费者消费，但是1个消费者能同时消费多个分区是吗？那1个消费者里面就会有多个消费者位移变量？
  如果1个主题有2个分区，消费者组有3个消费者，那至少有1个消费者闲置？ [1赞]

  作者回复2019-06-06 08:17:14
  在一个消费者组下，一个分区只能被一个消费者消费，但一个消费者可能被分配多个分区，因而在提交位移时也就能提交多个分区的位移。

  针对你说的第二种情况，答案是：是的。有一个消费者将无法分配到任何分区，处于idle状态。

- 柠檬C 2019-06-06 13:32:14
  不提供主写从读，第一是要保证一致性，逻辑就更复杂；第二是kafka相比redis还需要刷盘，流程更长；第三本身kafka通过partition已经实现了良好的负载均衡了

- 皮皮 2019-06-06 13:23:49
  前面有同学在问replica的leader和follower之间如何复制数据保证消息的持久化的问题，我了解的是有3种模式：1.生产者消息发过来以后，写leader成功后即告知生产者成功，然后异步的将消息同步给其他follower，这种方式效率最高，但可能丢数据；2.同步等待所有follower都复制成功后通知生产者消息发送成

功，这样不会丢数据，但效率不高；3.折中的办法，同步等待部分follower复制成功，如1个follower复制成功再返回，这样兼顾效率和消息的持久化。具体选择哪种要根据业务情况进行选择。说的不对的地方请老师斧正。

- 你看起来很好吃 2019-06-06 10:37:55
  胡老师您好，讲得真好，通俗易懂，感觉很值。
  不过我有一点困惑，想咨询一下您，我是一名刚从android转到后端的开发，现在工作里大部分工作都是做一些业务逻辑和使用一些常用的中间件，公司的业务也很难遇到真正的解决高并发，也很难遇到那种需要设计很好的分布式，高可用的系统。但是我想了解这些，想知道如果要做，要怎么来实现。
  然后我看这讲中有讲kafka有这种高可用，分布式的设计，所以我想我是否可以通过阅读kafka这相关的源码和设计，学习这样的系统如何设计。那如果可以的话，有没有好的相关的书可以推荐一下呢？感觉胡老师

- Francis 2019-06-06 10:27:31
  kafka客户端读操作是会移动broker中分区的offset，如果副本提供读服务，副本更变offset，再回同步领导副本，数据一致性就无法得到保障

- 口天小山己 2019-06-06 10:16:53
  follower副本向leader副本pull数据，是要等pull成功了才返回给生产者这条信息发送成功了吗，还是生产者数据发送到leader副本就算这条数据发送成功了，后面follower副本自己向生产者pull数据，保证最终一致性？如果是要等follower副本pull成功了才返回给生产者这条数据发送成功的话，这个pull不是异步的吗，是怎么通知生产者这次follower副本pull成功呢？

  作者回复2019-06-06 10:42:06
  producer是否等待follower拉取成功取决于producer端参数acks的设置。至于follower异步同步的完整机制在专栏的后面有详细的介绍~~

- 美美 2019-06-06 09:44:00
  副本提供对外读要解决的问题：
  1 可读的消息通过高水位隔离，HWM meta data需要保证一致性
  2 rebalance 和 lead replica failover需要保consumer offset一致性

- pain 2019-06-06 09:12:05
  老师，如果消费者组里面的消费者数量比分区数多，会有部分消费者闲置吗，还是多个消费者一直竞争。应该怎么选择分区与消费者的数量呢

  作者回复2019-06-06 09:21:31
  理想情况下消费者数量=消费者组订阅主题的总分区数，当然也不是绝对的。不过能肯定的是，如果消费者数>总分区数，那么就会有消费者闲置

- Summer 2019-06-06 08:37:51
  mysql读写分离是为了把大的“读”流量落到读库上，避免高并发下去锁表。

  而kafka是以 “追加”形式写，所以不存在随机io问题

- cricket1981 2019-06-06 08:33:17
  因为ISR机制，follower不一定具备leader一样全量最新消息集，为防止读不一致，所以不支持。

- alwyn 2019-06-06 08:33:12
  那就是消费者数一般小于分区数，不然会有消费者处于空闲状态

作者回复2019-06-06 08:37:24
消费者数<=分区数

- Johnson 2019-06-06 08:28:34
个人觉得主要是实现读写分离的性价比不高，技术难度太大

作者回复2019-06-06 08:38:45
社区最近正在讨论是否开放某些follower，允许其提供读服务。如果你有兴趣的话可以参与讨论：https://cwiki.apache.org/confluence/display/KAFKA/KIP-392%3A+Allow+consumers+to+fetch+from+closest+replica
就我个人而言，感觉难度也没有想象的那么大：）

- 伟子 2019-06-06 08:18:58
我记得kafka中是有保证领导者和追随者的数据一致性的机制的，即生产者要收到领导者反馈的和追随者同步消息成功的消息后才发送下一条消息。我想，这样理论上追随者是可以提供对外服务的。但是这样会导致消息发布的速度变慢，相当于是用牺牲吞吐量来换取的。而kafka是有分区机制和消费者组的机制来确保吞吐量的。所以，追随者不提供对外服务是不是为了不影响吞吐量？

作者回复2019-06-06 08:36:11
实话实说我们毕竟不是Kafka的作者，也许只能根据我们自己的理解来揣测Kafka的设计。我不能说你的这个观点是错误的，只是就我个人而言，为了避免处理一致性问题是这么设计的主要原因：）

- 杨俊 2019-06-06 08:13:13
领导者将数据同步到追随者副本既不是同步复制又不是异步复制，有一个isr列表维护，追随者副本自己去拉数据，有时候可能网络问题导致追随者副本之间存在数据不一致问题，高低水位不一样，isr列表的副本数也会不一样

作者回复2019-06-06 08:31:01
Kafka的副本拉取是完全异步的。另外实际上最新版本已经不单纯依赖高水位来判断了，而是依靠leader epoch

- mini希 2019-06-06 08:06:08
追随者副本同步数据有一定延迟吧，可能造成多个客户端读取到的数据不一致或者同一个客户端访问不同追随者副本不一致吧，

作者回复2019-06-06 08:28:51
follower不对外提供服务，所以不会出现这种情况。但可能的情形是因为客户端要等follower副本同步而出现的延时

- kxct 2019-06-06 08:01:00
副本不对外提供服务是为了强一致性吧

作者回复2019-06-06 08:28:04
不对外提供服务的确可以避免很多因异步副本拉取出现lag导致的一致性问题。