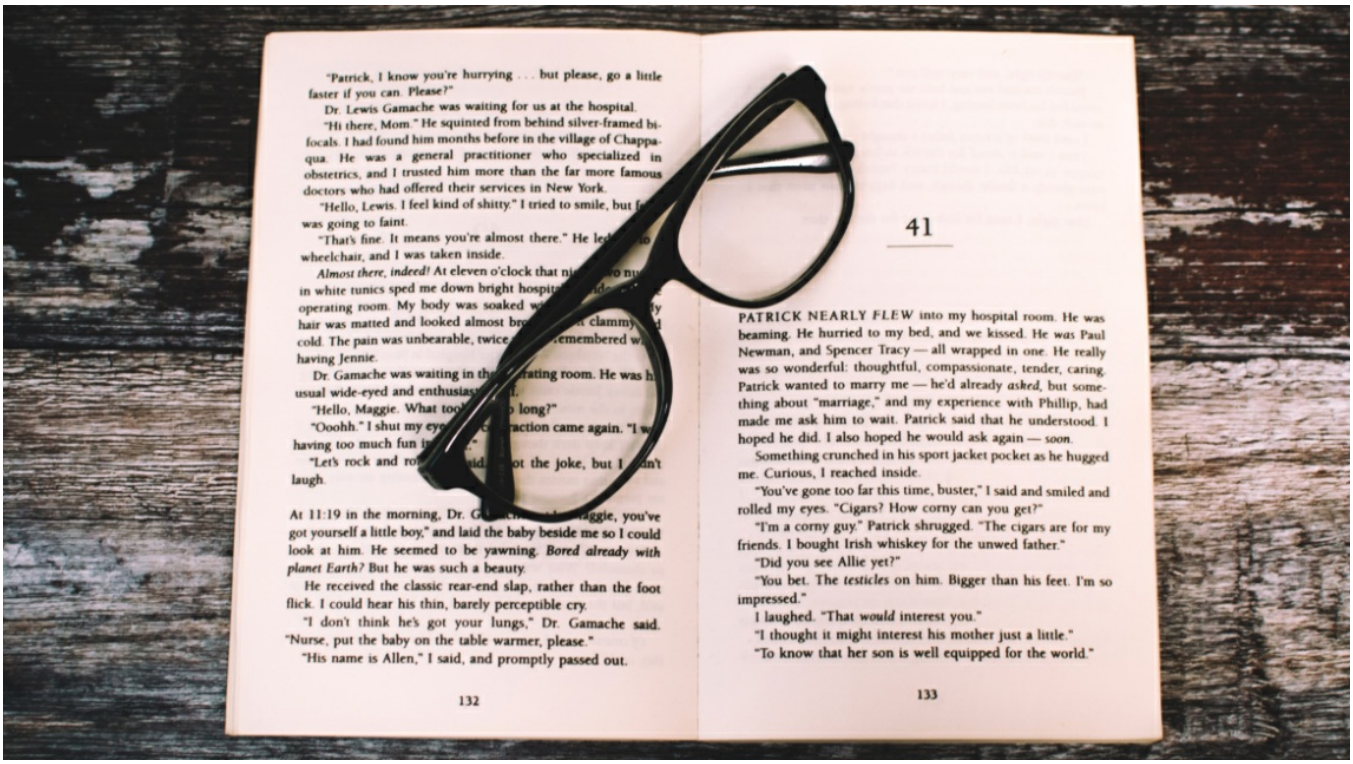
02 | 一篇文章带你快速搞定Kafka术语

2019-06-06 胡夕



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。

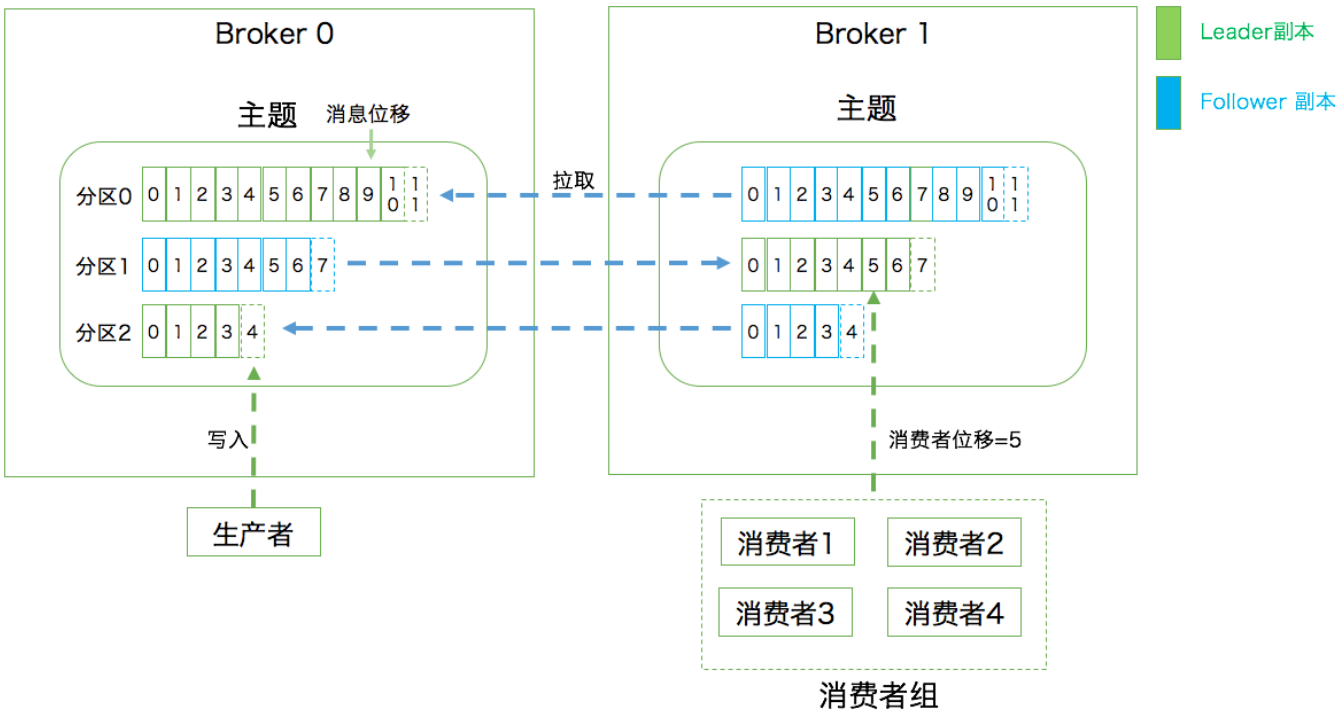
每个消费者在消费消息的过程中必然需要有个字段记录它当前消费到了分区的哪个位置上，这个字段就是消费者位移（Consumer Offset）。注意，这和上面所说的位移完全不是一个概念。上面的“位移”表征的是分区内的消息位置，它是不变的，即一旦消息被成功写入到一个分区上，它的位移值就是固定的了。而消费者位移则不同，它可能是随时变化的，毕竟它是消费者消费进度的指示器嘛。另外每个消费者有着自己的消费者位移，因此一定要区分这两类位移的区别。我个人把消息在分区中的位移称为分区位移，而把消费者端的位移称为消费者位移。

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。

最后我用一张图来展示上面提到的这些概念，希望这张图能够帮助你形象化地理解所有这些概念：



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

huxi_2b 4

结尾处增加了一张图，提炼了02中讲到的Kafka概念和术语，希望能够帮助到你：）

2019-06-11

we 56

老师 这个结构，为什么不用图表示。

2019-06-06

明翼 18

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

2019-06-06

骨汤鸡蛋面 11

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

2019-06-06

作者回复

嗯嗯，好的：）

2019-06-06

邋遢的流浪剑客 7

如果允许follower副本对外提供读服务（主写从读），首先会存在数据一致性的问题，消息从主节点同步到从节点需要时间，可能造成主从节点的数据不一致。主写从读无非就是为了减轻leader节点的压力，将读请求的负载均衡到follower节点，如果Kafka的分区相对均匀地分散到各个broker上，同样可以达到负载均衡的效果，没必要刻意实现主写从读增加代码实现的复杂程度

2019-06-06

作者回复

是的。前些天在知乎上就这个问题也解答了一下，有兴趣可以看看：https://www.zhihu.com/question/327925275/answer/705690755

2019-06-06

dbo 4

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。

2019-06-06

作者回复

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

2019-06-06

永光 3

为什么 Kafka 不像 MySQL 那样允许追随者副本对外提供读服务？

答：因为mysql一般部署在不同的机器上一台机器读写会遇到瓶颈，Kafka中的领导者副本一般均匀分布在不同的broker中，已经起到了负载的作用。即：同一个topic的已经通过分区的形式负载到不同的broker上了，读写的时候针对的领导者副本，但是量相比mysql一个还实例少太多，个人觉得没有必要在提供度读服务了。（如果量大还可以使用更多的副本，让每一个副本本身都不太大）不知道这样理解对不对?

2019-06-10

作者回复

我个人觉得是很不错的答案，自己也学到了一些：）

2019-06-10

ban 3

老师 这个结构，为什么不用图表示。

2019-06-08

莫道不销魂 2

老师，我想问下

1、 kafka是按照什么规则将消息划分到各个分区的？

2、既然同一个topic下的消息分布在不同的分区，那是什么机制将topic、partition、record关联或者说管理起来的？

2019-06-11

作者回复

1. 如果producer指定了要发送的目标分区，消息自然是去到那个分区；否则就按照producer端参数partitioner.class指定的分区策略来定；如果你没有指定过partitioner.class，那么默认的规则是：看消息是否有key，如果有则计算key的murmur2哈希值%topic分区数；如果没有key，按照轮询的方式确定分区。

2. 这个层级其实是逻辑概念。在物理上还是以日志段（log segment）文件的方式保存，日志段文件在内存中有对应的Java对象，里面关联了你说的这些。

2019-06-12

趙衍 2

我之前在学习Kafka的时候也有过这个问题，为什么Kafka不支持读写分离，让从节点对外提供读服务？

其实读写分离的本质是为了对读请求进行负载均衡，但是在Kafka中，一个topic的多个Prtition天然就被分散到了不同的broker服务器上，这种架构本身就解决了负载均衡地问题。也就是说，Kafka的设计从一刻开始就考虑到了分布式的问题，我觉得这是LinkedIn开发团队了不起的地方。

尽管如此，我觉得还有一个问题我没有想明白，如果Producer就是对某些broker中的leader副本进行大量的写入，或者Consumer就是对某些broker中的leader副本进行大量的拉取操作，那单台broker服务器的性能不还是成为了整个集群的瓶颈？请问老师，这种情况Kafka是怎么解决的呢？

2019-06-07

作者回复

只能是分散负载了，多做一些分区。

2019-06-10

QQ怪 2

kafka能否做到多个消费者消费一个生产者生产的数据，并能保证每个消费者消费的消息不会重复，做到并行消费?

2019-06-06

作者回复

Kafka提供了消费者组实现你说的这个需求~~

2019-06-06

(´田ω田`) 2

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？

2019-06-06

作者回复

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。

2019-06-06

永光 1

老师你好，每个消费者有着自己的消费者位移。“重平衡”时Kafka怎么知道已挂的消费者消费到哪里了？谢谢

2019-06-10

作者回复

重平衡时每个消费者都会尽力去做一次位移提交（如果它会提交位移的话），这样当rebalance完成后Kafka会告诉它们订阅分区当前消费到了那里。

2019-06-10

永光 1

老师你好，
有两个疑问：
1、文中说的Leader-Follower 指的是副本之间的关系，broker之间是对等关系，对吧？如果是对等关系，生产和消费时我怎么找到领导者副本所在的broker呢？
2、希望老师大致描述一下生产和消费时Kafka内部的处理流程？（如果能画个图,那将感激不尽）
谢谢老师。

2019-06-10

作者回复

1. 客户端会首先请求topic分区的leader副本在哪个broker上，内部自动执行的，你无需操心；
2. 记下了你的建议，后面我会注意的。另外专栏后面关于客户端运行机制的介绍：）

2019-06-10

永光 1

老师你好，

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。

“副本是在分区层级下的”感觉这么说不太准确，副本应该是分区的拷贝，和分区应该属于同一层级。

2019-06-10

作者回复

我个人觉得只要有助于正确理解Kafka，具体的说法并不是那么重要。"副本是在分区层级下的"的依据是在Kafka的源码Partition.scala中，每个分区对象都保存了一个hashmap，map中的对象就是Replica副本对象。

2019-06-10

双叶

1

Follower 不提供服务我认为是必要性不大吧，性能已经满足要求的情况下，follower 提供服务只会提高复杂度。为什么 kafka 性能满足高而 mysql 不行，除了做了分区以外，还有 kafka 读写都基本上是顺序的，I/O 压力小，计算相比数据库感觉也不复杂，所以就性能高呗。

2019-06-06

大番茄

1

首次接触kafka，有个理解疑问麻烦解惑：
在：刚才提到的副本如何与这里的分区联系在 ，生产者向分区写入消息， 这个消息是和副本之间是种什么关系的啊？

另外，这个关系老师可以出张图表示吗，感谢！

2019-06-06

作者回复

就像专栏里面说的，副本是在分区层级下定义的，即一个分区能够配置多个副本。生产者向分区写消息时其实是向leader副本写消息。其他follower副本主动去leader副本拉取这条消息实现同步。不知道这么解释清楚点了吗

2019-06-06

石妖

1

课后问题，我觉得是以下两个原因：
1.数据库中数据的读取只是数据的展示，而kafka中消息的读取意味着消费，consumer offset就会随之增加，kafka中的读取“相当于”MySQL中的更新，这两者的逻辑是不同的；
2.kafka集群可以通过增加partition及broker的方式实现负载均衡，并不需要从follower replica读取（消费）消息，那样会增加维护consumer offset的难度；而由于第一点的差异，使得MySQL可以通过将follower数据对外提供服务的方式来减轻leader节点的压力，实现读写分离，从而达到负载均衡。

2019-06-06

Francis

1

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

2019-06-06

哥本

0

老师讲解的很精彩！但是图太少，对于初学者不易理解，希望老师能多加些图 谢谢

2019-06-15

作者回复

好好，感谢您的建议。后面我会增加更多的图例：）

2019-06-15