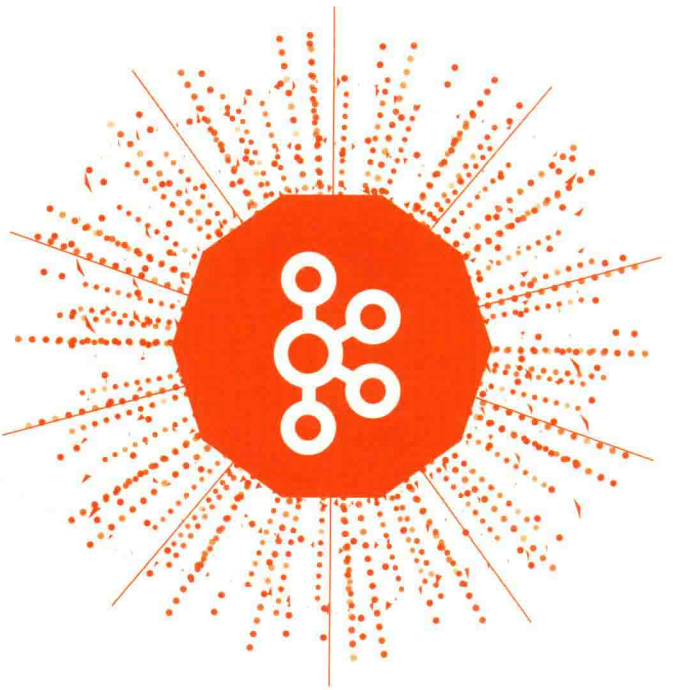
Broadview
www.broadview.com.cn



深入理解Kafka
核心设计与实践原理

朱忠华 | 著

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

深入理解Kafka
核心设计与实践原理

朱忠华 | 著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

本书从 Kafka 的基础概念切入，循序渐进地转入对其内部原理的剖析。本书主要阐述了 Kafka 中生产者客户端、消费者客户端、主题与分区、日志存储、原理解析、监控管理、应用扩展及流式计算等内容。虽然 Kafka 的内核使用 Scala 语言编写，但本书基本以 Java 语言作为主要的示例语言，方便大多数读者的理解。虽然本书没有明确的界定，但总体上可以划分为三个部分：基础篇、原理篇和扩展篇，前 4 章为基础篇，包括基础概念、生产者、消费者，以及主题与分区，学习完这 4 章的内容完全可以应对绝大多数的开发场景。第 5 章至第 8 章为原理篇，包括对日志存储、协议设计、控制器、组协调器、事务、一致性、可靠性等内容的探究，学习完这 4 章的内容可以让读者对 Kafka 有一个深刻的认知。最后 4 章从应用扩展层面来做讲解，可以归类为扩展篇，主要内容包括监控、应用工具、应用扩展（延时队列、重试队列、死信队列、消息轨迹等）、与 Spark 的集成等，让读者可以对 Kafka 的生态有一个更加全面的认知。

本书定位为一本实战与原理相结合的书，既适合 Kafka 的初学者，也适合于对 Kafka 有一定深度认知的老手。

图书在版编目（CIP）数据

深入理解 Kafka：核心设计与实践原理 / 朱忠华著. —北京：电子工业出版社，2019.1
ISBN 978-7-121-35902-6

Ⅰ. ①深… Ⅱ. ①朱… Ⅲ. ①分布式操作系统 Ⅳ. ①TP316.4

中国版本图书馆 CIP 数据核字（2019）第 007397 号

责任编辑：陈晓猛
印　　刷：天津嘉恒印务有限公司
装　　订：天津嘉恒印务有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　　邮编：100036
开　　本：787×980　1/16　　印张：28.5　　字数：547.2 千字
版　　次：2019 年 1 月第 1 版
印　　次：2019 年 3 月第 2 次印刷
定　　价：99.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：010-51260888-819，faq@phei.com.cn。

前言

初识 Kafka 时，笔者接触的还是 0.8.1 版本，Kafka 发展到目前的 2.0.0 版本，笔者也见证了 Kafka 的蜕变，比如旧版客户端的淘汰、新版客户端的设计、Kafka 控制器的迭代优化、私有协议的变更、事务功能的引入等。Kafka 从昔日的新星逐渐走向成熟，再到今日的王者地位不可撼动，这期间有太多的故事可讲。

刚接触 Kafka 时，市面上很少有关于 Kafka 的书籍。在学习 Kafka 的过程中也经历过很多挫败，比如 Scala 这门编程语言就让笔者在 Kafka 的源码大门外却步良久。那时候就在想，如果有一本书能够全方位地解析 Kafka 该有多好啊。

随着对 Kafka 的逐步了解，也渐渐地萌生了自己写一本关于 Kafka 的书的想法，产生这一想法至今已超过两年。在这期间，笔者阴差阳错地先写了一本关于 RabbitMQ 的书，也就是《RabbitMQ 实战指南》，此时已是 2017 年年末，市面上已经陆续出现了好几本有关 Kafka 的书，而且此时 Kafka 的版本也已经升级到 1.0.0。

笔者认真看过几乎所有现存的 Kafka 的书籍，回想这一路学习和使用 Kafka 的经历，深感这些都不是自己理想中的书籍，那么不如自己再“操刀”写一本。本书秉承能用文字表述的就不贴源码、能用图形辅助的就不乏味陈述；既要让新手能够快速入门，也要让老手有所收获，从基础概念入手，再到原理深入，让读者能够由浅入深地理解 Kafka。

本书依据 Kafka 2.0.0 版本编写，书中所有内容都具备理论基础并全部实践过，书中的内容也是笔者在工作中的认知积累，希望本书能够让读者有所收获。

内容大纲

本书共 12 章，前后章节都有相应的联系，基本上按照由浅入深、由表及里的层次逐层进行讲解，如果读者对其中的某些内容已经掌握，可以选择跳过而翻阅后面的内容，不过还是建议读者按照先后顺序进行阅读。

第 1 章对 Kafka 的基础概念进行笼统的介绍，之后讲解如何安装与配置 Kafka，以及通过简单的生产消费消息的示例让读者能够快速地入门。

第 2 章主要是针对生产者客户端的讲解，包括生产者客户端参数、消息的发送、序列化、分区器、拦截器、原理解析等内容。

第 3 章主要是针对消费者客户端的讲解，包括消费者客户端参数、主题与分区的订阅、反序列化、消息的消费、位移提交、再均衡、拦截器、多线程实现等内容。

第 4 章主要介绍主题与分区的管理，包括创建主题、修改主题、删除主题、主题端参数配置、优先副本、分区重分配、复制限流，以及对分区数抉择的探讨等内容。

第 5 章主要讲解日志存储相关的内容，包括文件目录的布局、日志格式的演变、日志清理的细节、底层存储的原理等内容。

第 6 章主要对 Kafka 服务端的一些内部核心内容进行详细的阐述，包括协议设计、延时操作、控制器、leader 的选举等内容。

第 7 章主要是对 Kafka 客户端相关的原理剖析，当然其中也需要牵涉服务端的内容。这一章包括消费端分区分配策略、消费者协调器和组协调器、__consumer_offsets 的剖析、事务的介绍等内容。

第 8 章主要对可靠性、一致性等核心原理进行陈述，本章内容最为抽象，主要包括失效副本、ISR 伸缩、LEO 与 HW、Leader Epoch 的介入、日志同步机制、可靠性分析等内容。

第 9 章主要是对 Kafka 相关应用的一些补充，包括一些重要的管理工具，还有 Kafka Connect、Kafka Mirror Maker 和 Kafka Streams 等内容。

第 10 章是与 Kafka 监控相关的内容，监控作为 Kafka 生态中的一个必备内容，有着相当重要的地位，通过学习本章的内容可以让读者对整个监控的脉络设计和底层实现有清晰的认知。

第 11 章是对 Kafka 做一些功能性的扩展，包括过期时间、延时队列、死信队列、重试队列、消息路由、消息轨迹、消息审计、消息代理等内容，最后还通过对消息中间件选型的阐述以期让读者对整个消息中间件领域有发散性的思考。

第 12 章主要讲述的是 Kafka 与 Spark 集成的一些内容，包括 Spark 基本概念、Spark Streaming、Structured Streaming，以及它们与 Kafka 集成的细节等内容。

读者讨论

由于笔者水平有限，书中难免有错误之处。在本书出版后的任何时间，若您对本书有任何疑问都可以通过 zhuzhonghua.ideal@qq.com 发送邮件给笔者，也可以到笔者的个人博客 http://blog.csdn.net/u013256816 中留言，向笔者阐述您的建议和想法。书中的源码会在本书发行之后进行整理，最后会公布在笔者的个人微信公众号（朱小厮的博客，二维码在封面上）中。

致谢

首先要感谢我身处的平台，让我有机会深入地接触 Kafka。同时要感谢我身边的同事，正因为有了你们的鼓励和帮助，才让我能够迅速地成长，本书的问世，离不开与你们在工作中一起积累的点点滴滴。

感谢蒋晓峰同学不辞辛苦地为本书校稿，有了你的帮助才会让本书更加完善。

感谢裘晟、顾忠国、刘建刚、芋道源码、益达-兰小伟、程超-小程故事多、HBase 技术社区、Kirito 的技术分享、徐志芳、肖宇、匠心零度、闪电侠的博客、方欧巴、zhisheng、Java 技术驿站、亚普的技术轮子、阿飞的博客、黄淋、Java 进阶架构师、占小狼的博客、黄晓峰、IT 牧场、服务端思维、涤生的博客等朋友的鼎力支持。

感谢天蚕变、Aruen、wangfeiyang12345、不写程序只算命、gyzhs20 等对本书勘误修正提供了帮助。

感谢在我博客、微信公众号中提问留言的朋友，感谢消息生态圈的朋友，有了你们的意见和建议才能让本书更加完善。

感谢博文视点的编辑们，本书能够顺利、迅速地出版，多亏了你们的敬业精神和一丝不苟的工作态度。

最后还要感谢我的家人，在我占用绝大部分的业余时间进行写作的时候，能够给予我极大的宽容、理解和支持，让我能够全身心地投入写作之中。

朱忠华

读者服务

轻松注册成为博文视点社区用户（www.broadview.com.cn），扫码直达本书页面。

- **下载资源**：本书如提供示例代码及资源文件，均可在 下载资源 处下载。
- **提交勘误**：您对书中内容的修改意见可在 提交勘误 处提交，若被采纳，将获赠博文视点社区积分（在您购买电子书时，积分可用来抵扣相应金额）。
- **交流互动**：在页面下方 读者评论 处留下您的疑问或观点，与我们和其他读者一同学习交流。

页面入口：http://www.broadview.com.cn/35902

目　录

第 1 章 初识 Kafka

Kafka 起初是由 LinkedIn 公司采用 Scala 语言开发的一个多分区、多副本且基于 ZooKeeper 协调的分布式消息系统，现已被捐献给 Apache 基金会。目前 Kafka 已经定位为一个分布式流式处理平台，它以高吞吐、可持久化、可水平扩展、支持流数据处理等多种特性而被广泛使用。目前越来越多的开源分布式处理系统如 Cloudera、Storm、Spark、Flink 等都支持与 Kafka 集成。

Kafka 之所以受到越来越多的青睐，与它所“扮演”的三大角色是分不开的：

- **消息系统**：Kafka 和传统的消息系统（也称作消息中间件）都具备系统解耦、冗余存储、流量削峰、缓冲、异步通信、扩展性、可恢复性等功能。与此同时，Kafka 还提供了大多数消息系统难以实现的消息顺序性保障及回溯消费的功能。
- **存储系统**：Kafka 把消息持久化到磁盘，相比于其他基于内存存储的系统而言，有效地降低了数据丢失的风险。也正是得益于 Kafka 的消息持久化功能和多副本机制，我们可以把 Kafka 作为长期的数据存储系统来使用，只需要把对应的数据保留策略设置为“永久”或启用主题的日志压缩功能即可。
- **流式处理平台**：Kafka 不仅为每个流行的流式处理框架提供了可靠的数据来源，还提供了一个完整的流式处理类库，比如窗口、连接、变换和聚合等各类操作。

1.1 基本概念

一个典型的 Kafka 体系架构包括若干 Producer、若干 Broker、若干 Consumer，以及一个 ZooKeeper 集群，如图 1-1 所示。其中 ZooKeeper 是 Kafka 用来负责集群元数据的管理、控制器的选举等操作的。Producer 将消息发送到 Broker，Broker 负责将收到的消息存储到磁盘中，而

Consumer 负责从 Broker 订阅并消费消息。

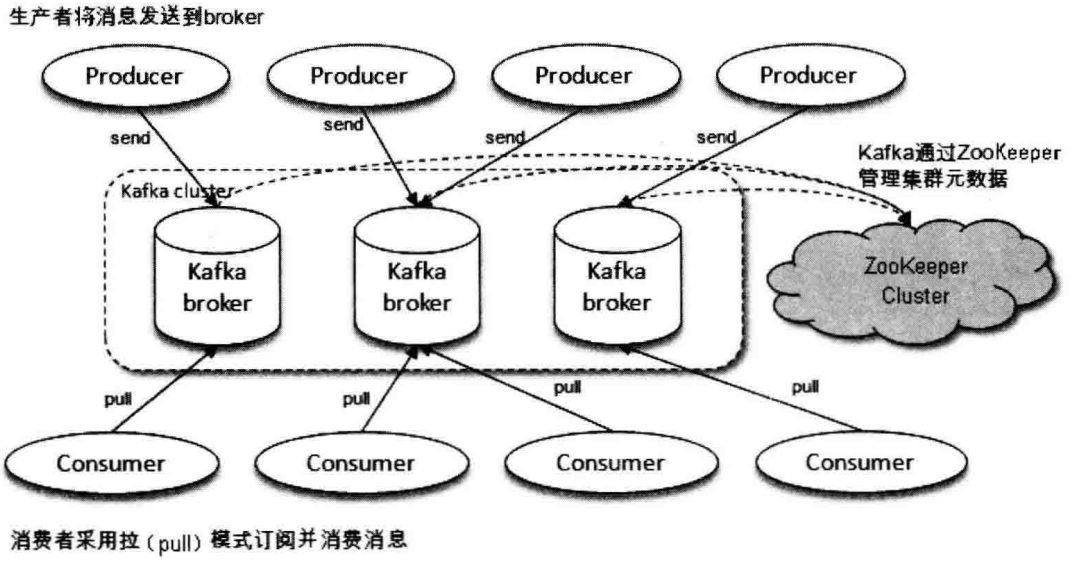


图 1-1 Kafka 体系结构

整个 Kafka 体系结构中引入了以下 3 个术语。

（1）Producer：生产者，也就是发送消息的一方。生产者负责创建消息，然后将其投递到 Kafka 中。

（2）Consumer：消费者，也就是接收消息的一方。消费者连接到 Kafka 上并接收消息，进而进行相应的业务逻辑处理。

（3）Broker：服务代理节点。对于 Kafka 而言，Broker 可以简单地看作一个独立的 Kafka 服务节点或 Kafka 服务实例。大多数情况下也可以将 Broker 看作一台 Kafka 服务器，前提是这台服务器上只部署了一个 Kafka 实例。一个或多个 Broker 组成了一个 Kafka 集群。一般而言，我们更习惯使用首字母小写的 broker 来表示服务代理节点。

在 Kafka 中还有两个特别重要的概念——主题（Topic）与分区（Partition）。Kafka 中的消息以主题为单位进行归类，生产者负责将消息发送到特定的主题（发送到 Kafka 集群中的每一条消息都要指定一个主题），而消费者负责订阅主题并进行消费。

主题是一个逻辑上的概念，它还可以细分为多个分区，一个分区只属于单个主题，很多时候也会把分区称为主题分区（Topic-Partition）。同一主题下的不同分区包含的消息是不同的，分区在存储层面可以看作一个可追加的日志（Log）文件，消息在被追加到分区日志文件的时候都会分配一个特定的偏移量（offset）。offset 是消息在分区中的唯一标识，Kafka 通过它来保证消息在分区内的顺序性，不过 offset 并不跨越分区，也就是说，Kafka 保证的是分区有序而不是主题有序。

如图 1-2 所示，主题中有 4 个分区，消息被顺序追加到每个分区日志文件的尾部。Kafka 中的分区可以分布在不同的服务器（broker）上，也就是说，一个主题可以横跨多个 broker，以此来提供比单个 broker 更强大的性能。

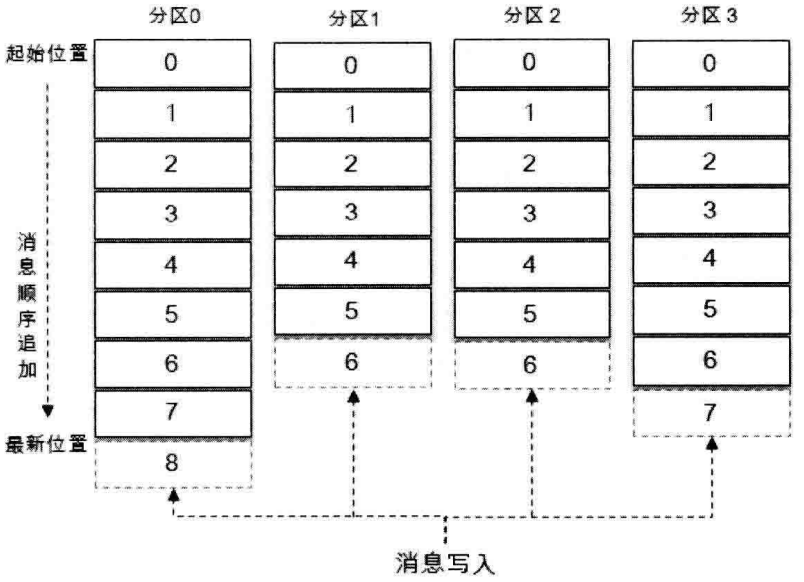


图 1-2 消息追加写入

每一条消息被发送到 broker 之前，会根据分区规则选择存储到哪个具体的分区。如果分区规则设定得合理，所有的消息都可以均匀地分配到不同的分区中。如果一个主题只对应一个文件，那么这个文件所在的机器 I/O 将会成为这个主题的性能瓶颈，而分区解决了这个问题。在创建主题的时候可以通过指定的参数来设置分区的个数，当然也可以在主题创建完成之后去修改分区的数量，通过增加分区的数量可以实现水平扩展。

Kafka 为分区引入了多副本（Replica）机制，通过增加副本数量可以提升容灾能力。同一分区的不同副本中保存的是相同的消息（在同一时刻，副本之间并非完全一样），副本之间是“一主多从”的关系，其中 leader 副本负责处理读写请求，follower 副本只负责与 leader 副本的消息同步。副本处于不同的 broker 中，当 leader 副本出现故障时，从 follower 副本中重新选举新的 leader 副本对外提供服务。Kafka 通过多副本机制实现了故障的自动转移，当 Kafka 集群中某个 broker 失效时仍然能保证服务可用。

如图 1-3 所示，Kafka 集群中有 4 个 broker，某个主题中有 3 个分区，且副本因子（即副本个数）也为 3，如此每个分区便有 1 个 leader 副本和 2 个 follower 副本。生产者和消费者只与 leader 副本进行交互，而 follower 副本只负责消息的同步，很多时候 follower 副本中的消息相对 leader 副本而言会有一定的滞后。

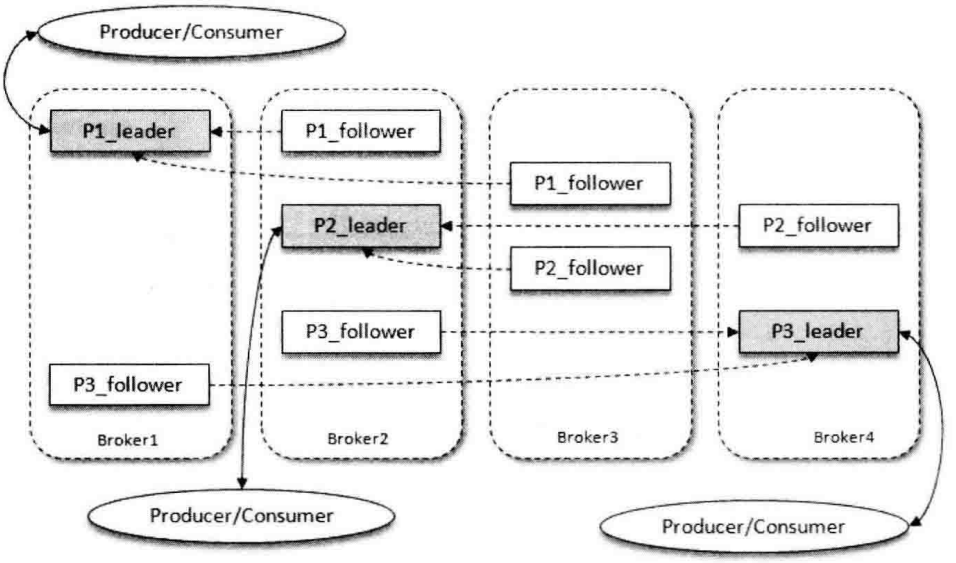


图 1-3 多副本架构

Kafka 消费端也具备一定的容灾能力。Consumer 使用拉（Pull）模式从服务端拉取消息，并且保存消费的具体位置，当消费者宕机后恢复上线时可以根据之前保存的消费位置重新拉取需要的消息进行消费，这样就不会造成消息丢失。

分区中的所有副本统称为 AR（Assigned Replicas）。所有与 leader 副本保持一定程度同步的副本（包括 leader 副本在内）组成 ISR（In-Sync Replicas），ISR 集合是 AR 集合中的一个子集。消息会先发送到 leader 副本，然后 follower 副本才能从 leader 副本中拉取消息进行同步，同步期间内 follower 副本相对于 leader 副本而言会有一定程度的滞后。前面所说的“一定程度的同步”是指可忍受的滞后范围，这个范围可以通过参数进行配置。与 leader 副本同步滞后过多的副本（不包括 leader 副本）组成 OSR（Out-of-Sync Replicas），由此可见，AR=ISR+OSR。在正常情况下，所有的 follower 副本都应该与 leader 副本保持一定程度的同步，即 AR=ISR，OSR 集合为空。

leader 副本负责维护和跟踪 ISR 集合中所有 follower 副本的滞后状态，当 follower 副本落后太多或失效时，leader 副本会把它从 ISR 集合中剔除。如果 OSR 集合中有 follower 副本“追上”了 leader 副本，那么 leader 副本会把它从 OSR 集合转移至 ISR 集合。默认情况下，当 leader 副本发生故障时，只有在 ISR 集合中的副本才有资格被选举为新的 leader，而在 OSR 集合中的副本则没有任何机会（不过这个原则也可以通过修改相应的参数配置来改变）。

ISR 与 HW 和 LEO 也有紧密的关系。HW 是 High Watermark 的缩写，俗称高水位，它标识了一个特定的消息偏移量（offset），消费者只能拉取到这个 offset 之前的消息。

如图 1-4 所示，它代表一个日志文件，这个日志文件中有 9 条消息，第一条消息的 offset

（LogStartOffset）为0，最后一条消息的offset为8，offset为9的消息用虚线框表示，代表下一条待写入的消息。日志文件的HW为6，表示消费者只能拉取到offset在0至5之间的消息，而offset为6的消息对消费者而言是不可见的。

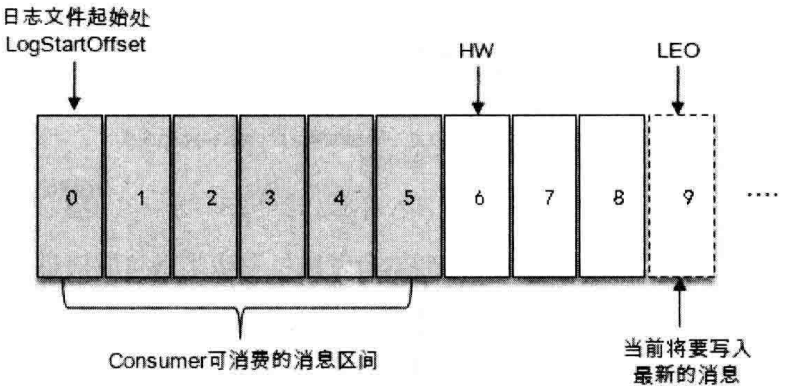


图1-4 分区中各种偏移量的说明

LEO是Log End Offset的缩写，它标识当前日志文件中下一条待写入消息的offset，图1-4中offset为9的位置即为当前日志文件的LEO，LEO的大小相当于当前日志分区中最后一条消息的offset值加1。分区ISR集合中的每个副本都会维护自身的LEO，而ISR集合中最小的LEO即为分区的HW，对消费者而言只能消费HW之前的消息。

注意要点：很多资料中误将图1-4中的offset为5的位置看作HW，而把offset为8的位置看作LEO，这显然是不对的。

为了让读者更好地理解ISR集合，以及HW和LEO之间的关系，下面通过一个简单的示例来进行相关的说明。如图1-5所示，假设某个分区的ISR集合中有3个副本，即一个leader副本和2个follower副本，此时分区的LEO和HW都为3。消息3和消息4从生产者发出之后会被先存入leader副本，如图1-6所示。

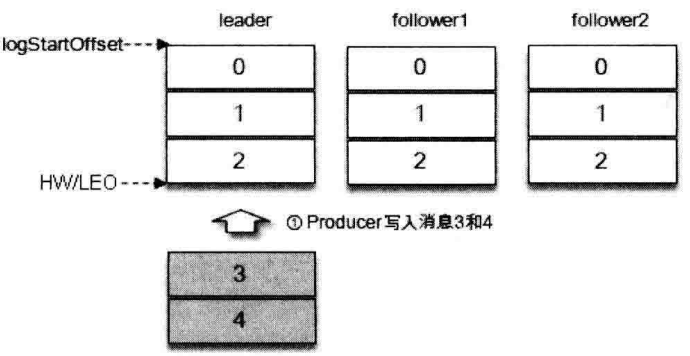


图1-5 写入消息（情形1）