TURING 图灵原创

- 图文详解Kafka的内部原理、设计与实现
- 全面分析以Kafka为中心的分布式流平台
- Kafka新特性详解，包括连接器和流处理

# Kafka技术内幕

## 图文详解Kafka源码设计与实现

郑奇煌◎著

中国工信出版集团

人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵原创



# Kafka技术内幕

## 图文详解Kafka源码设计与实现

郑奇煌◎著

人民邮电出版社
北京

图书在版编目（CIP）数据

Kafka技术内幕 : 图文详解Kafka源码设计与实现 / 郑奇煌著. -- 北京 : 人民邮电出版社, 2017.11
（图灵原创）
ISBN 978-7-115-46938-0

Ⅰ. ①K… Ⅱ. ①郑… Ⅲ. ①分布式操作系统－研究
Ⅳ. ①TP316.4

中国版本图书馆CIP数据核字(2017)第235945号

## 内 容 提 要

Kafka 自 LinkedIn 开源以来就以高性能、高吞吐量、分布式的特性著称。本书以 0.10 版本的源码为基础，深入分析了 Kafka 的设计与实现，包括生产者和消费者的消息处理流程，新旧消费者不同的设计方式，存储层的实现，协调者和控制器如何确保 Kafka 集群的分布式和容错特性，两种同步集群工具 MirrorMaker 和 uReplicator，流处理的两种 API 以及 Kafka 的一些高级特性等。

本书适合 Kafka 开发人员阅读。

◆ 著　　郑奇煌
责任编辑　王军花
责任印制　彭志环
◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路11号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
大厂聚鑫印刷有限责任公司印刷
◆ 开本：800×1000　1/16
印张：44.5
字数：1191千字　　2017年11月第1版
印数：1－4 000册　　2017年11月河北第1次印刷

定价：119.00元

读者服务热线：(010)51095186转600　印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京东工商广登字 20170147 号

站在巨人的肩上

Standing on Shoulders of Giants


iTuring.cn

站在巨人的肩上

Standing on Shoulders of Giants


iTuring.cn

# 前　言

Apache Kafka（简称Kafka）最早是由LinkedIn开源出来的分布式消息系统，现在是Apache旗下的一个子项目，并且已经成为开源领域应用最广泛的消息系统之一。Kafka社区也非常活跃，从0.9版本开始，Kafka的标语已经从“一个高吞吐量、分布式的消息系统”改为“一个分布式的流平台”。

## 如何阅读本书

本书主要以0.10版本的Kafka源码为基础，并通过图文详解的方式分析Kafka内部组件的实现细节。对于Kafka流处理的一些新特性，本书也会分析0.11版本的相关源码。本书各章的主要内容如下。

- 第1章首先介绍了Kafka作为流式数据平台的3个组成，包括消息系统、存储系统和流处理系统，接着从分区模型、消费模型和分布式模型这三个模型介绍了Kafka的几个基本概念，然后介绍了Kafka几个比较重要的设计思路，最后讨论了如何在一台机器上模拟单机模式与分布式模式，以及如何搭建开发环境。
- 第2章从一个生产者的示例开始，引出了新版本生产者的两种消息发送方式。生产者客户端通过记录收集器和发送线程，对消息集进行分组和缓存，并为目标节点创建生产请求，发送到不同的代理节点。接着介绍了与网络相关的Kafka通道、选择器、轮询等NIO操作。另外，还介绍了Scala版本的旧生产者，它使用阻塞通道的方式发送请求。最后，介绍了服务端采用Reactor模式处理客户端的请求。
- 第3章首先介绍了消费者相关的基础概念，然后从一个消费者的示例开始，引出了基于ZooKeeper（后面简称ZK）的高级消费者API。要理解高级API，主要是要理解消费线程的模型以及变量的传递方式。接着介绍了消费者提交分区偏移量的两种方式。最后，我们举了一个低级API的示例。开发者需要自己实现一些比较复杂的逻辑处理，才能保证消费程序的健壮性和稳定性。
- 第4章介绍了新版本的消费者。不同于旧版本的消费者，新版本去除了ZK的依赖，统一了旧版本的高级API和低级API，并提供了两种消费方式：订阅和分配。新版本引入订阅状态来管理消费者的订阅信息，并使用拉取器拉取消息。新版本的消费者没有使用拉取线程，而是采用轮询的方式拉取消息，它的性能比旧版本的消费者更好。另外，还介绍了消费者采用回调器、处理器、监听器、适配器、组合模式和链式调用等实现不同类型的异步请求。最后，我们介绍了新消费者的心跳任务、消费者提交偏移量以及3种消息处理语义的使用方式。

- 第5章介绍了新版本消费者相关的协调者实现，主要包括“加入组”与“同步组”。每个消费者都有一个客户端的协调者，服务端也有一个消费组级别的协调者负责处理所有消费者客户端的请求。当消费组触发再平衡操作时，服务端的协调者会记录消费组元数据的变化，并通过状态机保证消费组状态的正常转换。本章会通过很多不同的示例场景来帮助读者理解消费组相关的实现。
- 第6章介绍了Kafka的存储层实现，包括读写、管理、压缩等一些常用的日志操作。服务端通过副本管理器处理客户端的生产请求和拉取请求。接着介绍了副本机制相关的分区、副本、最高水位、复制点等一些概念。最后，介绍了延迟操作接口与延迟缓存。服务端如果不能立即返回响应结果给客户端，会先将延迟操作缓存起来，直到请求处理完成或超时。
- 第7章介绍了作为服务端核心的Kafka控制器，它主要负责管理分区状态机和副本状态机，以及多种类型的监听器，比如代理节点上线和下线、删除主题、重新分配分区等。控制器的一个重要职责是选举分区的主副本。不同代理节点根据控制器下发的请求，决定成为分区的主副本还是备份副本。另外，我们还分析了本地副本与远程副本的区别，以及元数据缓存的作用。
- 第8章首先介绍了两种集群的同步工具：Kafka内置的MirrorMaker和Uber开源的uReplicator。接着，介绍了新版本Kafka提供的连接器框架，以及如何开发一个自定义的连接器。最后，介绍了连接器的架构模型的具体实现，主要包括数据模型、Connector模型和Worker模型。
- 第9章介绍了Kafka流处理的两种API：低级Processor API和高级DSL。这一章重点介绍了流处理的线程模型，主要包括流实例、流线程和流任务。我们还介绍了流处理的本地状态存储，它主要用来作为备份任务的数据恢复。高级DSL包括两个组件——`KStream`与`KTable`，它们都定义了一些常用的流处理算子操作，比如无状态的操作（过滤和映射等）、有状态的操作（连接和窗口等）。
- 第10章介绍了Kafka的一些高级特性，比如客户端的配额、新的消息格式和事务特性。

本书相关的示例代码在笔者的GitHub主页https://github.com/zqhxuyuan/kafka-book上。另外，限于篇幅，本书的附录部分会放在个人博客上。此外，本书的源代码和附录部分也可以去图灵社区本书主页（http://www.ituring.com.cn/book/1927）免费下载。

由于个人能力有限，文中的错误在所难免，如果读者在阅读的过程中，发现不妥之处，可以私信我的微博：http://weibo.com/xuyuantree，我会定期将勘误更新到个人博客上。

## 致谢

感谢图灵的编辑王军花老师，是您的辛勤工作让本书的出版成为可能。同时还要感谢许多我不知道名字的幕后工作人员为本书付出的努力。

感谢冯嘉、时金魁、吴阳平在百忙之中抽出时间给本书写推荐。

# 目　录

# 第1章 Kafka入门

在0.10版本之前，Kafka仅仅作为一个消息系统，主要用来解决应用解耦、异步消息、流量削峰等问题。不过在0.10版本之后，Kafka提供了连接器与流处理的能力，它也从分布式的消息系统逐渐成为一个流式的数据平台。本章首先介绍Kafka流式数据平台的基本组成，然后分析它的一些架构设计和基本概念，最后通过几个示例快速理解它的一些重要特性。

## 1.1 Kafka 流式数据平台

作为一个流式数据平台，最重要的是要具备下面3个特点。

- 类似消息系统，提供事件流的发布和订阅，即具备数据注入功能。
- 存储事件流数据的节点具有故障容错的特点，即具备数据存储功能。
- 能够对实时的事件流进行流式地处理和分析，即具备流处理功能。

下面我们分析作为一个流式数据平台，Kafka是如何实现并组合上面的3个功能特点的。

- **消息系统**：如图1-1所示，消息系统（也叫作消息队列）主要有两种消息模型：队列和发布订阅。Kafka使用消费组（consumer group）统一了上面两种消息模型。Kafka使用队列模型时，它可以将处理工作平均分配给消费组中的消费者成员；使用发布订阅模式时，它可以将消息广播给多个消费组。采用多个消费组结合多个消费者，既可以线性扩展消息的处理能力，也允许消息被多个消费组订阅。
- **队列模式（也叫作点对点模式）**。多个消费者读取消息队列，每条消息只发送给一个消费者。
- **发布–订阅模式（pub/sub）**。多个消费者订阅主题，主题的每条记录会发布给所有的消费者。

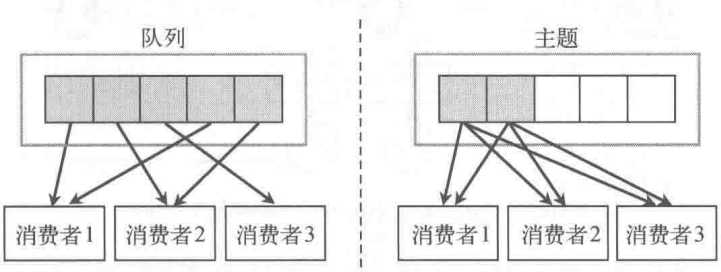


图1-1 消息系统的两种消息模型：队列模型与发布–订阅模型

- **存储系统**：任何消息队列要做到“发布消息”和“消费消息”的解耦合，实际上都要扮演一个存储系统的角色，负责保存还没有被消费的消息。否则，如果消息只是在内存中，一旦机器宕机或进程重启，内存中的消息就会全部丢失。Kafka也不例外，数据写入到Kafka集群的服务器节点时，还会复制多份来保证出现故障时仍能可用。为了保证消息的可靠存储，Kafka还允许生产者的生产请求在收到应答结果之前，阻塞式地等待一条消息，直到它完全地复制到多个节点上，才认为这条消息写入成功。
- **流处理系统**：流式数据平台仅仅有消息的读取和写入、存储消息流是不够的，还需要有实时的流式数据处理能力。对于简单的处理，可以直接使用Kafka的生产者和消费者API来完成；但对于复杂的业务逻辑处理，直接操作原始的API需要做的工作非常多。Kafka流处理（Kafka Streams）为开发者提供了完整的流处理API，比如流的聚合、连接、各种转换操作。同时，Kafka流处理框架内部解决很多流处理应用程序都会面临的问题：处理乱序或迟来的数据、重新处理输入数据、窗口和状态操作等。
- **将消息系统、存储存储、流处理系统组合在一起**：传统消息系统的流处理通常只会处理订阅动作发生之后才到达的新消息，无法处理订阅之前的历史数据。分布式文件存储系统一般存储静态的历史数据，对历史数据的处理一般采用批处理的方式。现有的开源系统很难将这些系统无缝地整合起来，Kafka则将消息系统、存储系统、流处理系统都组合在一起，构成了以Kafka为中心的流式数据处理平台。它既能处理最新的实时数据，也能处理过去的历史数据。Kafka作为流式数据平台的核心组件，主要包括下面4种核心的API，如图1-2所示。
  - **生产者**（producer）应用程序发布事件流到Kafka的一个或多个主题。
  - **消费者**（consumer）应用程序订阅Kafka的一个或多个主题，并处理事件流。
  - **连接器**（connector）将Kafka主题和已有数据源进行连接，数据可以互相导入和导出。
  - **流处理**（processor）从Kafka主题消费输入流，经过处理后，产生输出流到输出主题。

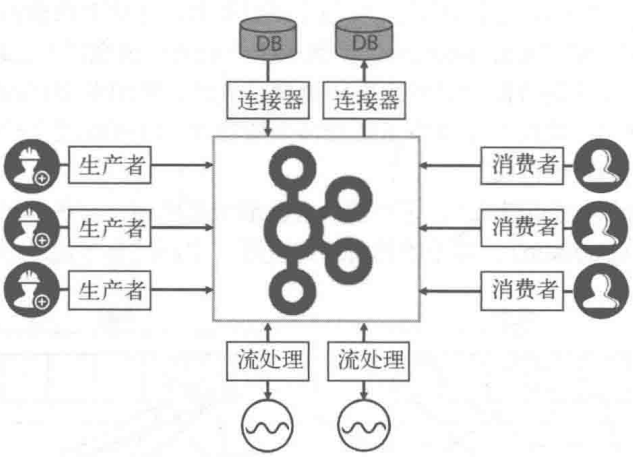


图1-2　Kafka流式数据平台的4种核心API

建立以Kafka为核心的流式数据管道，不仅要保证低延迟的消息处理，还需要保证数据存储的可靠性。另外，在和离线系统集成时，将Kafka的数据加载到批处理系统时，要保证数据不遗漏；Kafka

集群的某些节点在停机维护时，要保证集群可用。上面从整体上分析了Kafka如何作为一个流式的数据处理平台，下面开始分析Kafka的架构实现，这里先从基本概念说起，然后分析它的一些重要实现细节。

## 1.2 Kafka的基本概念

下面我们会从3个角度分析Kafka的几个基本概念，并尝试解决下面3个问题。

- Kafka的主题与分区内部是如何存储的，它有什么特点？
- 与传统的消息系统相比，Kafka的消费模型有什么优点？
- Kafka如何实现分布式的数据存储与数据读取？

### 1.2.1 分区模型

Kafka集群由多个消息代理服务器（broker server）组成，发布到Kafka集群的每条消息都有一个类别，用主题（topic）来表示。通常，不同应用产生不同类型的数据，可以设置不同的主题。一个主题一般会有多个消息的订阅者，当生产者发布消息到某个主题时，订阅了这个主题的消费者都可以接收到生产者写入的新消息。

Kafka集群为每个主题维护了分布式的分区（partition）日志文件，物理意义上可以把主题看作分区的日志文件（partitioned log）。每个分区都是一个有序的、不可变的记录序列，新的消息会不断追加到提交日志（commit log）。分区中的每条消息都会按照时间顺序分配到一个单调递增的顺序编号，叫作偏移量（offset），这个偏移量能够唯一地定位当前分区中的每一条消息。

如图1-3（左）所示，主题有3个分区，每个分区的偏移量都从0开始，不同分区之间的偏移量都是独立的，不会互相影响。右图中，发布到Kafka主题的每条消息包括键值和时间戳。消息到达服务端的指定分区后，都会分配到一个自增的偏移量。原始的消息内容和分配的偏移量以及其他一些元数据信息最后都会存储到分区日志文件中。消息的键也可以不用设置，这种情况下消息会均衡地分布到不同的分区。

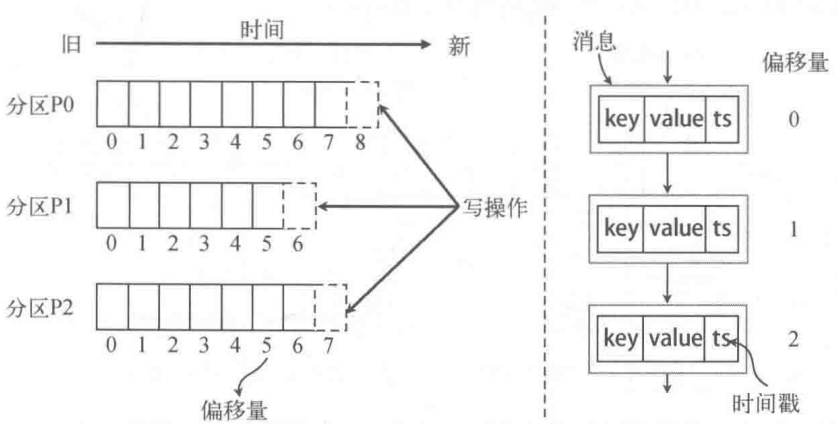


图1-3 事件流构成的主题，物理上由多个分区日志组成

传统消息系统在服务端保持消息的顺序，如果有多个消费者消费同一个消息队列，服务端会以消息存储的顺序依次发送给消费者。但由于消息是异步发送给消费者的，消息到达消费者的顺序可能是无序的，这就意味着在并行消费时，传统消息系统无法很好地保证消息被顺序处理。虽然我们可以设置一个专用的消费者只消费一个队列，以此来解决消息顺序的问题，但是这就使得消费处理无法真正执行。

Kafka比传统消息系统有更强的顺序性保证，它使用主题的分区作为消息处理的并行单元。Kafka以分区作为最小的粒度，将每个分区分配给消费组中不同的而且是唯一的消费者，并确保一个分区只属于一个消费者，即这个消费者就是这个分区的唯一读取线程。那么，只要分区的消息是有序的，消费者处理的消息顺序就有保证。每个主题有多个分区，不同的消费者处理不同的分区，所以Kafka不仅保证了消息的有序性，也做到了消费者的负载均衡。

### 1.2.2　消费模型

消息由生产者发布到Kafka集群后，会被消费者消费。消息的消费模型有两种：推送模型（push）和拉取模型（pull）。基于推送模型的消息系统，由消息代理记录消费者的消费状态。消息代理在将消息推送到消费者后，标记这条消息为已消费，但这种方式无法很好地保证消息的处理语义。比如，消息代理把消息发送出去后，当消费进程挂掉或者由于网络原因没有收到这条消息时，就有可能造成消息丢失（因为消息代理已经把这条消息标记为已消费了，但实际上这条消息并没有被实际处理）。如果要保证消息的处理语义，消息代理发送完消息后，要设置状态为“已发送”，只有收到消费者的确认请求后才更新为“已消费”，这就需要在消息代理中记录所有消息的消费状态，这种做法也是不可取的。

Kafka采用拉取模型，由消费者自己记录消费状态，每个消费者互相独立地顺序读取每个分区的消息。如图1-4所示，有两个消费者（不同消费组）拉取同一个主题的消息，消费者A的消费进度是3，消费者B的消费者进度是6。消费者拉取的最大上限通过最高水位（watermark）控制，生产者最新写入的消息如果还没有达到备份数量，对消费者是不可见的。这种由消费者控制偏移量的优点是：消费者可以按照任意的顺序消费消息。比如，消费者可以重置到旧的偏移量，重新处理之前已经消费过的消息；或者直接跳到最近的位置，从当前时刻开始消费。

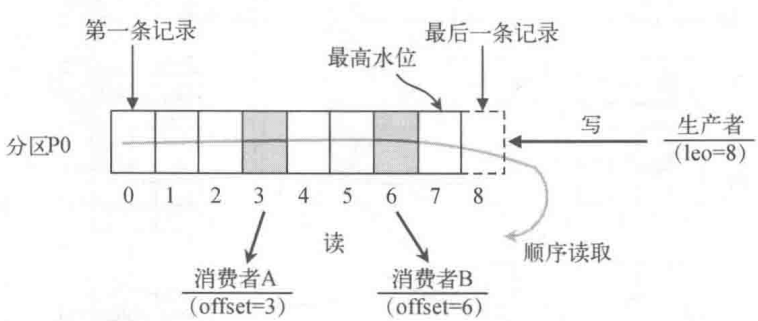


图1-4　采用拉取模型的消费者自己记录消费状态

在一些消息系统中，消息代理会在消息被消费之后立即删除消息。如果有不同类型的消费者订阅同一个主题，消息代理可能需要冗余地存储同一条消息；或者等所有消费者都消费完才删除，这就需

要消息代理跟踪每个消费者的消费状态，这种设计很大程度上限制了消息系统的整体吞吐量和处理延迟。Kafka的做法是生产者发布的所有消息会一直保存在Kafka集群中，不管消息有没有被消费。用户可以通过设置保留时间来清理过期的数据，比如，设置保留策略为两天。那么，在消息发布之后，它可以被不同的消费者消费，在两天之后，过期的消息就会自动清理掉。

### 1.2.3 分布式模型

Kafka每个主题的多个分区日志分布式地存储在Kafka集群上，同时为了故障容错，每个分区都会以副本的方式复制到多个消息代理节点上。其中一个节点会作为主副本（Leader），其他节点作为备份副本（Follower，也叫作从副本）。主副本会负责所有的客户端读写操作，备份副本仅仅从主副本同步数据。当主副本出现故障时，备份副本中的一个副本会被选择为新的主副本。因为每个分区的副本中只有主副本接受读写，所以每个服务端都会作为某些分区的主副本，以及另外一些分区的备份副本，这样Kafka集群的所有服务端整体上对客户端是负载均衡的。

Kafka的生产者和消费者相对于服务端而言都是客户端，生产者客户端发布消息到服务端的指定主题，会指定消息所属的分区。生产者发布消息时根据消息是否有键，采用不同的分区策略。消息没有键时，通过轮询方式进行客户端负载均衡；消息有键时，根据分区语义确保相同键的消息总是发送到同一个分区。

Kafka的消费者通过订阅主题来消费消息，并且每个消费者都会设置一个消费组名称。因为生产者发布到主题的每一条消息都只会发送给消费组的一个消费者。所以，如果要实现传统消息系统的“队列”模型，可以让每个消费者都拥有相同的消费组名称，这样消息就会负载均衡到所有的消费者；如果要实现“发布-订阅”模型，则每个消费者的消费组名称都不相同，这样每条消息就会广播给所有的消费者。

分区是消费者线程模型的最小并行单位。如图1-5（左）所示，生产者发布消息到一台服务器的3个分区时，只有一个消费者消费所有的3个分区。在图1-5（右）中，3个分区分布在3台服务器上，同时有3个消费者分别消费不同的分区。假设每个服务器的吞吐量是300 MB，在图1-5（左）中分摊到每个分区只有100 MB，而在图1-5（右）中集群整体的吞吐量有900 MB。可以看到，增加服务器节点会提升集群的性能，增加消费者数量会提升处理性能。

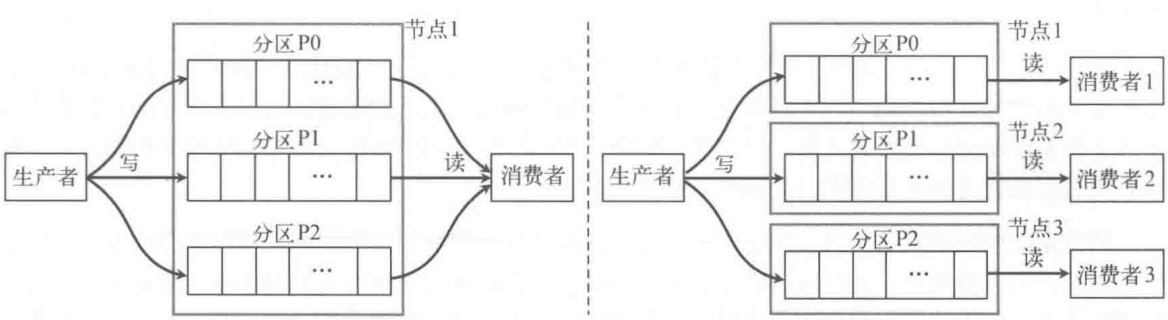


图1-5 分区作为消费者线程模型的最小并行单位

同一个消费组下多个消费者互相协调消费工作，Kafka会将所有的分区平均地分配给所有的消费者实例，这样每个消费者都可以分配到数量均等的分区。Kafka的消费组管理协议会动态地维护消费组的成员列表，当一个新消费者加入消费组，或者有消费者离开消费组，都会触发再平衡操作。

Kafka的消费者消费消息时，只保证在一个分区内消息的完全有序性，并不保证同一个主题中多个分区的消息顺序。而且，消费者读取一个分区消息的顺序和生产者写入到这个分区的顺序是一致的。比如，生产者写入“hello”和“kafka”两条消息到分区P1，则消费者读取到的顺序也一定是“hello”和“kafka”。如果业务上需要保证所有消息完全一致，只能通过设置一个分区完成，但这种做法的缺点是最多只能有一个消费者进行消费。一般来说，只需要保证每个分区的有序性，再对消息加上键来保证相同键的所有消息落入同一个分区，就可以满足绝大多数的应用。

上面从宏观角度分析了Kafka的3种基本模型，下面分析Kafka在底层实现上的一些设计细节与考虑。

## 1.3　Kafka的设计与实现

下面我们会从3个角度分析Kafka的一些设计思路，并尝试回答下面3个问题。

- 如何利用操作系统的优化技术来高效地持久化日志文件和加快数据传输效率？
- Kafka的生产者如何批量地发送消息，消费者采用拉取模型带来的优点都有哪些？
- Kafka的副本机制如何工作，当故障发生时，怎么确保数据不会丢失？

### 1.3.1　文件系统的持久化与数据传输效率

人们普遍认为一旦涉及磁盘的访问，读写的性能就严重下降。实际上，现代的操作系统针对磁盘的读写已经做了一些优化方案来加快磁盘的访问速度。比如，**预读**（read-ahead）会提前将一个比较大的磁盘块读入内存。**后写**（write-behind）会将很多小的逻辑写操作合并起来组合成一个大的物理写操作。并且，操作系统还会将主内存剩余的所有空闲内存空间都用作**磁盘缓存**（disk cache/page cache），所有的磁盘读写操作都会经过统一的磁盘缓存（除了直接I/O会绕过磁盘缓存）。综合这几点优化特点，如果是针对磁盘的顺序访问，某些情况下它可能比随机的内存访问都要快，甚至可以和网络的速度相差无几。

如图1-6（左）所示，应用程序写入数据到文件系统的一般做法是：在内存中保存尽可能多的数据，并在需要时将这些数据刷新到文件系统。但这里我们要做完全相反的事情，右图中所有的数据都立即写入文件系统的持久化日志文件，但不进行刷新数据的任何调用。数据会首先被传输到磁盘缓存，操作系统随后会将这些数据定期自动刷新到物理磁盘。

消息系统内的消息从生产者保存到服务端，消费者再从服务端读取出来，数据的传输效率决定了生产者和消费者的性能。生产者如果每发送一条消息都直接通过网络发送到服务端，势必会造成过多的网络请求。如果我们能够将多条消息按照分区进行分组，并采用批量的方式一次发送一个消息集，并且对消息集进行压缩，就可以减少网络传输的带宽，进一步提高数据的传输效率。

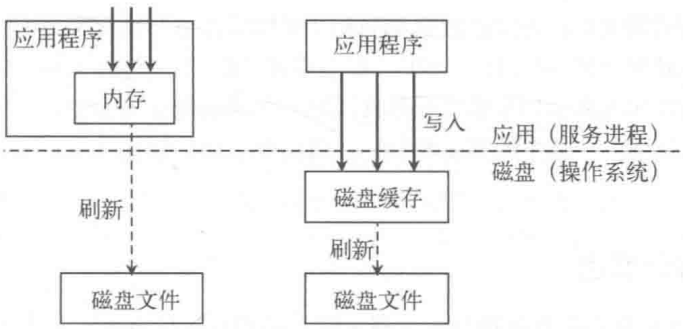


图1-6 应用程序写入数据到文件系统的两种做法

消费者要读取服务端的数据，需要将服务端的磁盘文件通过网络发送到消费者进程，而网络发送通常涉及不同的网络节点。如图1-7（左）所示，传统读取磁盘文件的数据在每次发送到网络时，都需要将页面缓存先保存到用户缓存，然后在读取消息时再将其复制到内核空间，具体步骤如下。

(1) 操作系统将数据从磁盘中读取文件到内核空间里的页面缓存。
(2) 应用程序将数据从内核空间读入用户空间的缓冲区。
(3) 应用程序将读到的数据写回内核空间并放入socket缓冲区。
(4) 操作系统将数据从socket缓冲区复制到网卡接口，此时数据才能通过网络发送出去。

结合Kafka的消息有多个订阅者的使用场景，生产者发布的消息一般会被不同的消费者消费多次。如图1-7（右）所示，使用“零拷贝技术”（zero-copy）只需将磁盘文件的数据复制到页面缓存中一次，然后将数据从页面缓存直接发送到网络中（发送给不同的使用者时，都可以重复使用同一个页面缓存），避免了重复的复制操作。这样，消息使用的速度基本上等同于网络连接的速度了。

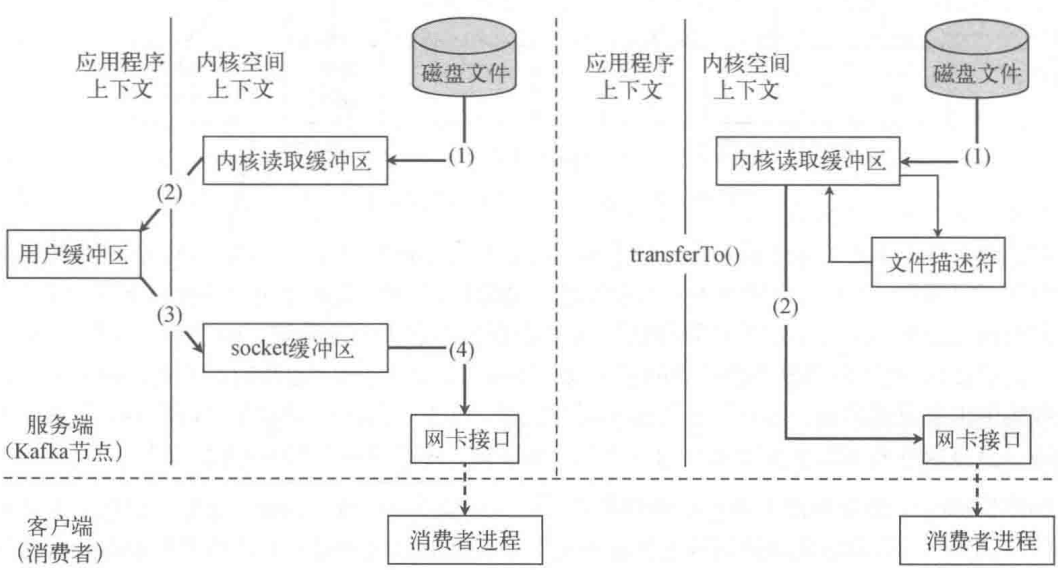


图1-7 传统的数据复制方法和优化的零拷贝