介绍

Kafka 是一个分布式的流媒体平台。主要有三大作用：

1、 发布和订阅流记录数据，类似于消息队列或企业消息系统；

2、 以容错方式存储流记录数据；

3、 当流数据出现时进行处理。

Kafka 通常用于两大类应用程序：

1、 构建实时流数据管道，在系统或应用程序之间可靠地获取数据；

2、 构建实时流应用程序，转换或响应数据流。

几个简单重要的概念

1、 Kafka 作为集群运行在一个或多个服务器上，这些服务器可以跨多个数据中心。

2、 Kafka 集群将记录流存储在称为主题的类别中。

3、 每条记录由一个键、一个值和一个时间戳组成。

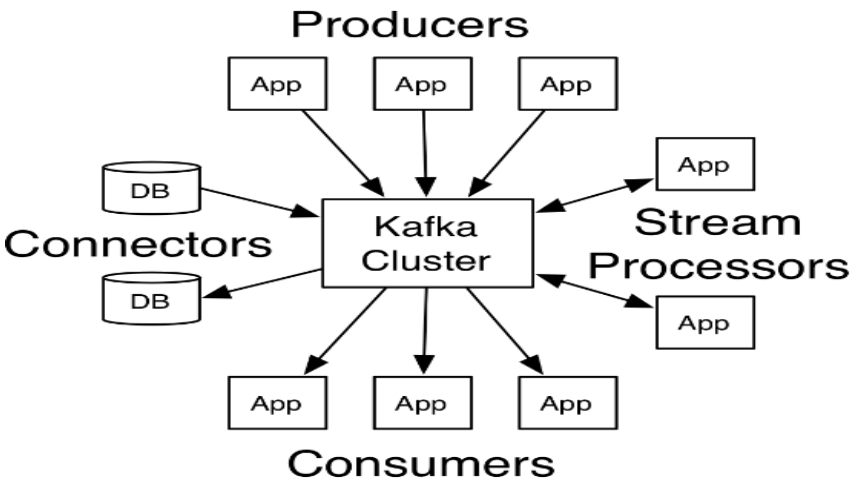
Kafka 用于四种核心的 API

1、 Producer API ：允许一个应用程序发布流记录数据到一个或者多个 kafka 主题中；

2、 Consumer API ：允许应用程序订阅一个或者多个 kafka 主题和处理给它们生成的记录流；

3、 Streams API：允许一个应用程序作为一个流处理器，消费一个或者多个主题输入流和生产一个输出流给一个或多个输出主题；

4、 Connector API：允许构建和运行可重用的生产者或消费者，这些生产者或消费者将 Kafka 主题连接到现有的应用程序或数据系统。例如，到关系数据库的连接器可能会捕获对表的每次更改。



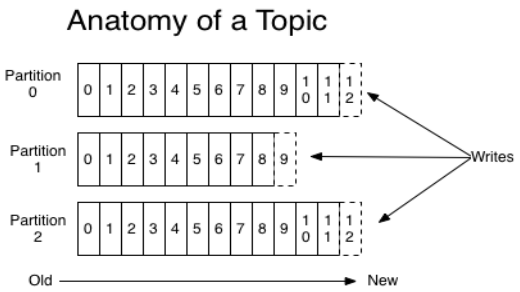
在 Kafka 中，客户端和服务器之间的通信使用简单、高性能、语言无关的 TCP 协议完成。该协议经过版本控制，并与旧版本保持向后兼容性。我们为 Kafka 提供了一个 Java 客户端，但是客户端可以使用多种语言。

主题和日志

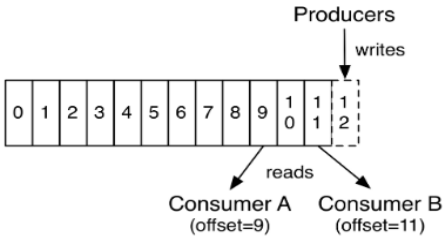
让我们首先深入了解 Kafka 为记录流(主题)提供的核心抽象-主题（topic）。

主题是发布记录的类别或提要名称。kafka 中的主题始终是多订阅者的;也就是说，一个主题可以有零个、一个或多个订阅其所写数据的消费者。

对于每个主题，Kafka 集群维护一个分区日志，如下所示:



每个分区都是一个有序的、不可变的记录序列，这些记录(数据)连续地附加到一个结构化的提交日志中。分区中的每条记录都被分配了一个名为偏移量的连续 id 号，偏移量惟一地标识分区中的每条记录。Kafka 集群使用一个可配置的保留期持久地保存所有已发布的记录(无论它们是否已被消费)。例如，如果保留策略被设置为两天，那么在记录发布后的两天内，它是可用的，在此之后，它将被丢弃以释放空间。Kafka 的性能在数据大小方面是稳定的，所以长时间存储数据不是问题。



实际上，在每个消费者的基础上保留的惟一元数据是该消费者在日志中的偏移量或位置。这个偏移量由消费者控制:通常（默认情况下），消费者在读取记录时将线性地推进偏移量，但是，实际上，由于位置由消费者控制，所以它可以按照自己喜欢的任何顺序使用记录。例如，消费者可以重置为较早的偏移量，以便重新处理之前的数据，或者跳过到最近的记录，从“现在”开始消费。

这些特性的组合意味着 Kafka 消费者非常廉价——他们可以来来去去，而不会对集群或其他消费者造成太大影响。例如，您可以使用我们的命令行工具“跟踪”任何主题的内容，而不需要更改任何现有消费者所使用的内容。

日志中的分区有几个用途。首先，它们允许日志扩展到超出单个服务器所能容纳的大小。每个单独的分区必须适合承载它的服务器，但是一个主题可能有多个分区，因此它可以处理任意数量的数据。其次，它们作为平行度的单位——稍后详细说明。

分布

日志的分区分布在 Kafka 集群中的服务器上，每个服务器处理数据并请求共享分区。为了容错，每个分区被复制到多个可配置的服务器上。

每个分区都有一个充当“leader”的服务器和零个或多个充当“followers”的服务器。leader 处理分区的所有读和写请求，而 follower 被动地复制 leader。如果 leader 失败，其中一个 follower 将自动成为新的领导者。每个服务器充当它的一些分区的 leader 和其他分

区的 follower，因此集群内的负载非常平衡。

异地备份

镜像同步为集群提供异地备份支持。使用镜像同步，消息可以跨多个数据中心或云区域复制。您可以在用于备份和恢复的主动/被动场景中使用它;或在主动/被动场景中，将数据放置到离用户更近的位置，或支持数据本地化需求。

生产者（Producers）

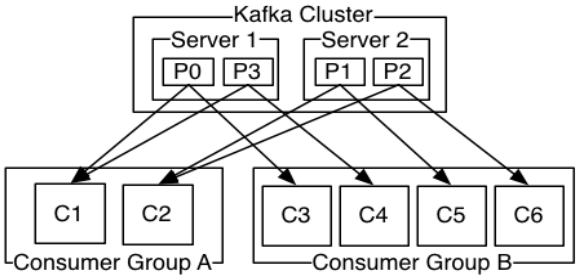
生产者将数据发布到他们选择的主题。生产者负责选择要分配给主题中的哪个分区的记录。这可以通过循环方式来完成，只是为了平衡负载，也可以根据语义分区函数(比如基于记录中的某个键)来完成。

消费者（Consumers）

消费者用消费者组名称来标记自己，每个发布到主题的记录都被交付到每个订阅用户组中的一个消费者实例。消费者实例可以位于单独的进程中，也可以位于单独的机器上。

如果所有消费者实例都具有相同的消费者组，那么记录将有效地在消费者实例上进行负载平衡。

如果所有消费者实例具有不同的消费者组，则每个记录将广播到所有消费者进程。



一个包含四个分区(P0-P3)的两台服务器 Kafka 集群，包含两个消费者组。消费者组 A 有两个消费者实例，而 B 组有四个。

然而，更常见的情况是，我们发现主题有少量的消费者组，每个“逻辑订阅者”对应一个消费组。每个组由许多消费者实例组成，用于可伸缩性和容错。这只不过是发布-订阅语义，其中订阅者是一组消费者，而不是单个流程。

Kafka 中实现消费的方法是在消费者实例上划分日志中的分区，这样每个实例在任何时候都是分区“公平共享”的独占消费者。这个维护组成员关系的过程由 Kafka 协议动态处理。如果新实例加入组，它们将从组的其他成员那里接管一些分区;如果一个实例死亡，它的分区将分配给其余的实例。

Kafka 只提供分区内记录的总顺序，而不是主题中不同分区之间的顺序。对大多数应用程序来说，按分区排序和按键分区数据的能力已经足够了。然而，如果您需要记录的总顺序，则可以使用只有一个分区的主题来实现这一点，尽管这意味着每个使用者组只有一个使用者进程。

多租户（Multi-tenancy）

您可以将 Kafka 部署为一个多租户解决方案。通过配置哪些主题可以生成或使用数据，可以启用多租户。这里还有对配额的操作支持。管理员可以对请求定义和强制配额，以控制客户端使用的代理资源。

保证（Guarantees）

Kafka 给出了下面高水平的保证：

1、 生产者发送到特定主题分区的消息将按发送顺序追加。也就是说，如果记录 M1 是由与记录 M2 相同的生产者发送的，并且 M1 是先发送的，那么 M1 的偏移量将比 M2 低，并且出现在日志的前面。

2、 消费者实例按记录存储在日志中的顺序查看记录。

3、 对于具有复制因子 N 的主题，kafka 将容忍最多 N-1 个服务器故障，而不会丢失提交到日志的任何记录。

Kafka 作为一个消息系统

Kafka 的流概念与传统的企业消息系统相比如何?

消息传递传统上有两种模型:排队和发布-订阅。在队列中，一个消费者池可以从一个服务器读取数据，并且每个记录将被发送到其中一个;在发布-订阅中，记录被广播给所有消费者。这两种模式各有优缺点。排队的优势在于，它允许您在多个使用者实例上划分数据处理，从而允许您扩展处理。不幸的是，一旦一个进程读取了数据，数据就没有了。发布-订阅允许您将数据广播到多个进程，但是由于每个消息都发送到每个订阅者，因此无法扩展处理。

Kafka 中的消费者组概念概括了这两个概念。与队列一样，消费者组允许您将处理划分到一组进程(消费者组的成员)上。与发布-订阅一样，Kafka 允许您向多个消费者组广播消息。

Kafka 模型的优势在于，每个主题都具有这两种属性——可以扩展处理，而且是多订阅者的——不需要选择其中之一。

Kafka 也比传统的消息传递系统有更强的顺序保证。

传统的队列在服务器中也保持记录的顺序性，并且如果有多个消费者消费这个队列的话，服务器会回根据记录存储的顺序分发给它们。然而，尽管服务器顺序分发，记录被异步分发给消费者。但是它们可能不是顺序到达消费者，这实际上意味着在并行消费时记录的顺序性将丢失。因此消息系统经常以“单一消费者”的方式消费，意思就是说只允许单一进程消费。但是这样意味着消费不存在并行性。

而 kafka 做的更好，通过在主题中有个并行的概念-分区。Kafka 能够在消费者进程池中提供顺序保证和负载均衡。它的实现是通过将 topic 中的 partition 分配给消费者组的消费者进程的方式以至于每个分区都能确切的被消费者组的消费者消费。通过这样的方式，消费者能够顺序的消费那个分区的数据。由于有很多分区，这仍然可以在许多消费者实例上负载均衡。但是需要注意的是，在一个消费者组中的消费者数量不能大于分区数。

Kafka 作为一个存储系统

任何允许发布与使用解耦的消息的消息队列都可以有效地充当正在运行的消息的存储系统，Kafka 的不同之处在于它是一个非常好的存储系统。写入 kafka 的数据被写入磁盘并复制，以获得容错的能力。Kafka 允许生产者等待确认，这样直到完全复制并确保持久化即使写入的服务器失败，写入也不会被认为是完整的。

Kafka 使用的磁盘结构伸缩性很好，无论服务器上有 50 KB 还是 50 TB 的持久数据，Kafka 都会执行相同的操作。

由于严格对待存储并允许客户控制其读取位置，你可以认为 kafka 是一种特殊用途的分布式文件系统，专门用于高性能，低延迟的日志提交存储。

用于流处理的 kafka

仅仅用于读、写和存储流数据是不够的，它的目的是能够处理实时流数据。

在 Kafka 中，流处理器是指从输入主题获取连续的数据流，对这个输入执行一些处理，并产生连续的数据流到输出主题。

例如，一个零售应用程序可能接受销售和发货的输入流，并根据这些数据计算出重新订购和价格调整的输出流。

使用生产者和消费者 API 能做一些简单的处理，但是对于更复杂的复杂的转换 kafka 提供了一套完整的流处理 API（Streams API）。这允许构建进行重大处理的应用程序，这些处理可以计算流之外的聚合或将流连接在一起。

此功能有助于解决这类应用程序面临的难题：处理无序数据，在代码更改时重新处理输入，执行有状态计算，等等

streams API 构建在 Kafka 提供的核心原语之上:它使用生产者和消费者 API 进行输入，使用 Kafka 进行有状态存储，并使用相同的组机制在流处理器实例之间进行容错。

拼接碎片

这种消息传递、存储和流处理的组合可能看起来不寻常，但对于 Kafka 作为流平台的角色来说，这是必不可少的。

像 HDFS 这样的分布式文件系统允许存储静态文件进行批处理。实际上，这样的系统允许存储和处理来自过去的历史数据。

传统的企业消息传递系统允许处理订阅后到达的未来消息。以这种方式构建的应用程序在未来数据到达时处理它。

Kafka 结合了这两种功能，这种结合对于 Kafka 作为流应用程序的平台以及流数据管道来说都是至关重要的。

通过结合存储和低延迟订阅，流应用程序可以以相同的方式处理过去和未来的数据。也就是说，一个应用程序可以处理历史上存储的数据，但是当它到达最后一条记录时，它可以在未来的数据到达时继续处理，而不是结束。这是流处理的一个广义概念，包括批处理和消息驱动应用程序。

同样，对于流数据管道，订阅实时事件的组合使使用 Kafka 处理非常低延迟的管道成为可能；但是，可靠地存储数据的能力使其能够用于必须保证数据交付的关键数据，或者用于与离线系统集成，离线系统只定期加载数据，或者可能在很长一段时间内停机以进行维护。流处理设施使得在数据到达时转换数据成为可能。