Kafka 入门

什么是 Kafka

kafka 最初是 LinkedIn 的一个内部基础设施系统。最初开发的起因是，LinkedIn 虽然有了数据库和其他系统可以用来存储数据，但是缺乏一个可以帮助处理持续数据流的组件。所以在设计理念上，开发者不想只是开发一个能够存储数据的系统，如关系数据库、Nosql 数据库、搜索引擎等等，更希望把数据看成一个持续变化和不断增长的流，并基于这样的想法构建出一个数据系统，一个数据架构。

Kafka 外在表现很像消息系统，允许发布和订阅消息流，但是它和传统的消息系统有很大的差异，

首先，Kafka 是个现代分布式系统，以集群的方式运行，可以自由伸缩。

其次，Kafka 可以按照要求存储数据，保存多久都可以，

第三，流式处理将数据处理的层次提示到了新高度，消息系统只会传递数据，Kafka 的流式处理能力可以让我们用很少的代码就能动态地处理派生流和数据集。所以 Kafka 不仅仅是个消息中间件。

Kafka 不仅仅是一个消息中间件，同时它是一个流平台，这个平台上可以发布和订阅数据流（Kafka 的流，有一个单独的包 Stream 的处理），并把他们保存起来，进行处理，这个是 Kafka 作者的设计理念。

大数据领域，Kafka 还可以看成实时版的 Hadoop，但是还是有些区别，Hadoop 可以存储和定期处理大量的数据文件，往往以 TB 计数，而 Kafka 可以存储和持续处理大型的数据流。Hadoop 主要用在数据分析上，而 Kafka 因为低延迟，更适合于核心的业务应用上。所以国内的大公司一般会结合使用，比如京东在实时数据计算架构中就使用了到了 Kafka,具体见《张开涛-海量数据下的应用系统架构实践》

常见的大数据处理框架：storm、spark、Flink、(Blink 阿里)

Kafka 名字的由来：卡夫卡与法国作家马塞尔·普鲁斯特，爱尔兰作家詹姆斯·乔伊斯并称为西方现代主义文学的先驱和大师。《变形记》是卡夫卡的短篇代表作，是卡夫卡的艺术成就中的一座高峰，被认为是 20 世纪最伟大的小说作品之一（达到管理层的高度应该多看下人文相关的书籍，增长管理知识和人格魅力）。

本次课程，将会以 kafka_2.11-2.3.0 版本为主，其余版本不予考虑，并且 Kafka 是 scala 语言写的，小众语言,没有必要研究其源码，投入和产出比低，除非你的技术级别非常高或者需要去开发单独的消息中间件。

Kafka 中的基本概念

消息和批次

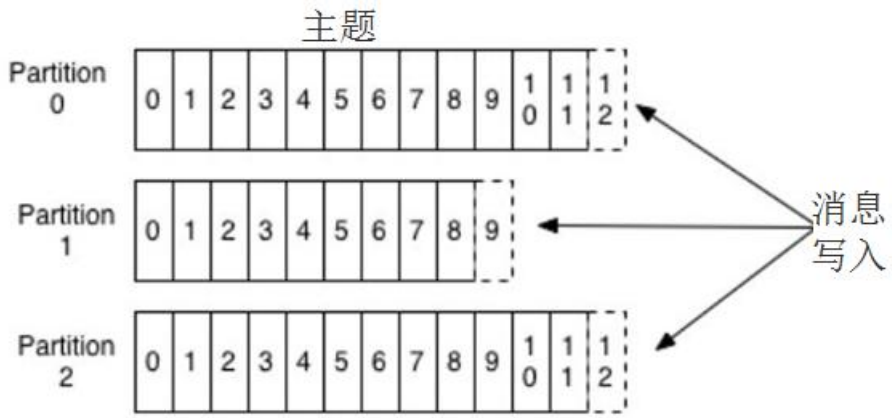
消息，Kafka 里的数据单元，也就是我们一般消息中间件里的消息的概念（可以比作数据库中一条记录）。消息由字节数组组成。消息还可以包含键（可选元数据，也是字节数组），主要用于对消息选取分区。

作为一个高效的消息系统，为了提高效率，消息可以被分批写入 Kafka。**批次**就是一组消息，这些消息属于同一个主题和分区。如果只传递单个消息，会导致大量的网络开销，把消息分成批次传输可以减少这开销。但是，这个需要权衡（时间延迟和吞吐量之间），批次里包含的消息越多，单位时间内处理的消息就越多，单个消息的传输时间就越长（吞吐量高延时也高）。如果进行压缩，可以提升数据的传输和存储能力，但需要更多的计算处理。

对于 Kafka 来说，消息是晦涩难懂的字节数组，一般我们使用序列化和反序列化技术，格式常用的有 JSON 和 XML，还有 Avro（Hadoop 开发的一款序列化框架），具体怎么使用依据自身的业务来定。

主题和分区

Kafka 里的消息用**主题**进行分类（主题好比数据库中的表），主题下有可以被分为若干个**分区（分表技术）**。分区本质上是个提交日志文件，有新消息，这个消息就会以追加的方式写入分区（写文件的形式），然后用先入先出的顺序读取。



但是因为主题会有多个分区，所以在整个主题的范围内，是无法保证消息的顺序的，单个分区则可以保证。

Kafka 通过分区来实现数据冗余和伸缩性，因为分区可以分布在不同的服务器上，那就是说一个主题可以跨越多个服务器（这是 Kafka 高性能的一个原因，多台服务器的磁盘读写性能比单台更高）。

前面我们说 Kafka 可以看成一个流平台，很多时候，我们会把一个主题的数据看成一个流，不管有多少个分区。

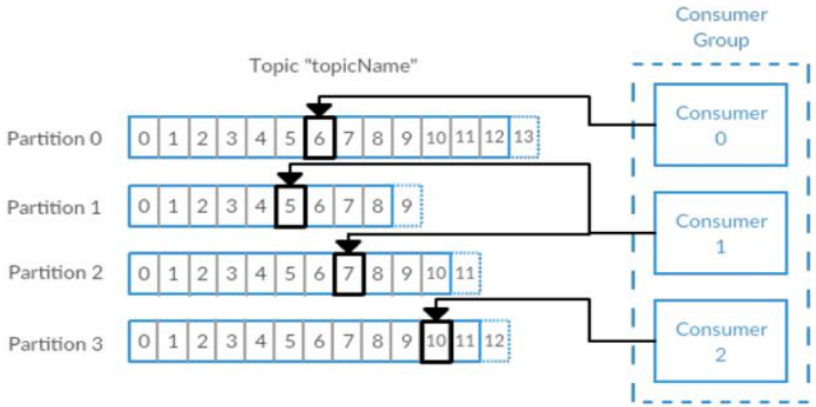
生产者和消费者、偏移量、消费者群组

就是一般消息中间件里生产者和消费者的概念。一些其他的高级客户端 API，像数据管道 API 和流式处理的 Kafka Stream，都是使用了最基本的生产者和消费者作为内部组件，然后提供了高级功能。

生产者默认情况下把消息均衡分布到主题的所有分区上，如果需要指定分区，则需要使用消息里的消息键和分区器。

消费者订阅主题，一个或者多个，并且按照消息的生成顺序读取。消费者通过检查所谓的偏移量来区分消息是否读取过。偏移量是一种元数据，一个不断递增的整数值，创建消息的时候，Kafka 会把他加入消息。在一个主题中一个分区里，每个消息的偏移量是唯一的。每个分区最后读取的消息偏移量会保存到 Zookeeper 或者 Kafka 上，这样分区的消费者关闭或者重启，读取状态都不会丢失。

多个消费者可以构成一个消费者群组。怎么构成？共同读取一个主题的消费者们，就形成了一个群组。群组可以保证每个分区只被一个消费者使用。



消费者和分区之间的这种映射关系叫做消费者对分区的所有权关系，很明显，一个分区只有一个消费者，而一个消费者可以有多个分区。

（吃饭的故事：一桌一个分区，多桌多个分区，生产者不断生产消息(消费)，消费者就是买单的人，消费者群组就是一群买单的人），一个分区只能被消费者群组中的一个消费者消费（不能重复消费），如果有一个消费者挂掉了<James 跑路了>，另外的消费者接上）

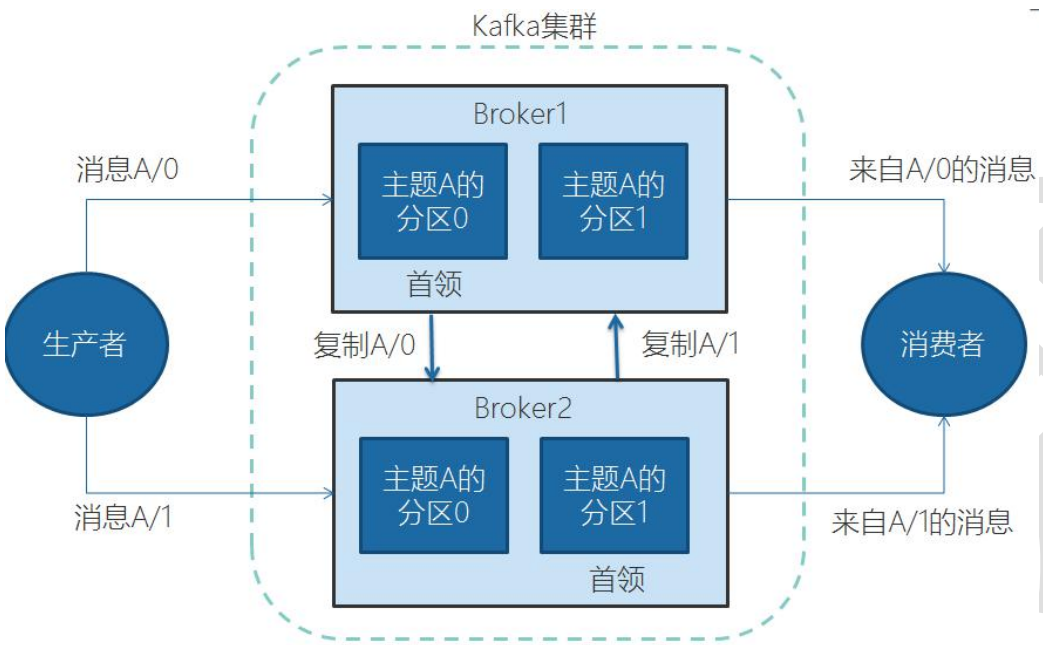
Broker 和集群

一个独立的 Kafka 服务器叫 Broker。broker 的主要工作是，接收生产者的消息，设置偏移量，提交消息到磁盘保存；为消费者提供服务，响应请求，返回消息。在合适的硬件上，单个 broker 可以处理上千个分区和每秒百万级的消息量。（要达到这个目的需要做操作系统调优和 JVM 调优）

多个 broker 可以组成一个集群。每个集群中 broker 会选举出一个集群控制器。控制器会进行管理，包括将分区分配给 broker 和监控 broker。

集群里，一个分区从属于一个 broker，这个 broker 被称为首领。但是分区可以被分配给多个 broker，这个时候会发生分区复制。

集群中 Kafka 内部一般使用管道技术进行高效的复制。



分区复制带来的好处是，提供了消息冗余。一旦首领 broker 失效，其他 broker 可以接管领导权。当然相关的消费者和生产者都要重新连接到新的首领上。

保留消息

在一定期限内保留消息是 Kafka 的一个重要特性，Kafka broker 默认的保留策略是：要么保留一段时间（7 天），要么保留一定大小（比如 1 个 G）。到了限制，旧消息过期并删除。但是每个主题可以根据业务需求配置自己的保留策略（开发时要注意，Kafka 不像 Mysql 之类的永久存储）。

为什么选择 Kafka

优点

多生产者和多消费者

基于磁盘的数据存储，换句话说，Kafka 的数据天生就是持久化的。

高伸缩性，Kafka 一开始就被设计成一个具有灵活伸缩性的系统，对在线集群的伸缩丝毫不影响整体系统的可用性。

高性能，结合横向扩展生产者、消费者和 broker，Kafka 可以轻松处理巨大的信息流（LinkedIn 公司每天处理万亿级数据），同时保证亚秒级的消息延迟。

常见场景

活动跟踪

跟踪网站用户和前端应用发生的交互，比如页面访问次数和点击，将这些信息作为消息发布到一个或者多个主题上，这样就可以根据这些数据为机器学习提供数据，更新搜素结果等等（头条、淘宝等总会推送你感兴趣的内容，其实在数据分析之前就已经做了活动跟踪）。

传递消息

标准消息中间件的功能

收集指标和日志

收集应用程序和系统的度量监控指标，或者收集应用日志信息，通过 Kafka 路由到专门的日志搜索系统，比如 ES。（国内用得较多）

提交日志

收集其他系统的变动日志，比如数据库。可以把数据库的更新发布到 Kafka 上，应用通过监控事件流来接收数据库的实时更新，或者通过事件流将数据库的更新复制到远程系统。

还可以当其他系统发生了崩溃，通过重放日志来恢复系统的状态。（异地灾备）

流处理

操作实时数据流，进行统计、转换、复杂计算等等。随着大数据技术的不断发展和成熟，无论是传统企业还是互联网公司都已经不再满足于离线批处理，实时流处理的需求和重要性日益增长 。

近年来业界一直在探索实时流计算引擎和 API，比如这几年火爆的 Spark Streaming、Kafka Streaming、Beam 和 Flink，其中阿里双 11 会场展示的实时销售金额，就用的是流计算，是基于 Flink，然后阿里在其上定制化的 Blink。

Kafka 的安装、管理和配置

安装

预备环境

Kafka 是 Java 生态圈下的一员，用 Scala 编写，运行在 Java 虚拟机上，所以安装运行和普通的 Java 程序并没有什么区别。

安装 Kafka 官方说法，Java 环境推荐 Java8。

Kafka 需要 Zookeeper 保存集群的元数据信息和消费者信息。Kafka 一般会自带 Zookeeper，但是从稳定性考虑，应该使用单独的 Zookeeper，而且构建 Zookeeper 集群。

下载和安装 Kafka

在 http://kafka.apache.org/downloads 上寻找合适的版本下载，我们这里选用的是 kafka_2.11-2.3.0，下载完成后解压到本地目录。

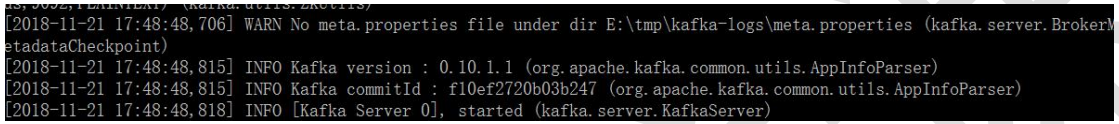
运行

启动 Zookeeper

进入 Kafka 目录下的 bin\windows

执行 kafka-server-start.bat ../../config/server.properties，出现以下画面表示成功

Linux 下与此类似，进入 bin 后，执行对应的 sh 文件即可



基本的操作和管理

##列出所有主题

kafka-topics.bat --zookeeper localhost:2181 --list

##列出所有主题的详细信息

kafka-topics.bat --zookeeper localhost:2181 --describe

##创建主题 主题名 my-topic，1 副本，8 分区

kafka-topics.bat --zookeeper localhost:2181 --create --topic my-topic --replication-factor 1 --partitions 8

##增加分区，注意：分区无法被删除

kafka-topics.bat --zookeeper localhost:2181 --alter --topic my-topic --partitions 16

##删除主题

kafka-topics.bat --zookeeper localhost:2181 --delete --topic my-topic

##创建生产者（控制台）

kafka-console-producer.bat --broker-list localhost:9092 --topic my-topic

##创建消费者（控制台）

kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic my-topic --from-beginning

##列出消费者群组（仅 Linux）

kafka-topics.sh --new-consumer --bootstrap-server localhost:9092 --list

##列出消费者群组详细信息（仅 Linux）

kafka-topics.sh --new-consumer --bootstrap-server localhost:9092 --describe --group 群组名

Broker 配置

配置文件放在 Kafka 目录下的 config 目录中，主要是 server.properties 文件

常规配置

broker.id

在单机时无需修改，但在集群下部署时往往需要修改。它是个每一个 broker 在集群中的唯一表示，要求是正数。当该服务器的 IP 地址发生改变时，broker.id 没有变化，则不会影响 consumers 的消息情况

listeners

监听列表(以逗号分隔 不同的协议(如 plaintext,trace,ssl、不同的 IP 和端口)),hostname 如果设置为 0.0.0.0 则绑定所有的网卡地址；如果 hostname 为空则绑定默认的网卡。如果
没有配置则默认为 java.net.InetAddress.getCanonicalHostName()。

如：PLAINTEXT://myhost:9092,TRACE://:9091 或 PLAINTEXT://0.0.0.0:9092,

zookeeper.connect

zookeeper 集群的地址，可以是多个，多个之间用逗号分割。（一组 hostname:port/path 列表,hostname 是 zk 的机器名或 IP、port 是 zk 的端口、/path 是可选 zk 的路径，如果不指定，默认使用根路径）

log.dirs

Kafka 把所有的消息都保存在磁盘上，存放这些数据的目录通过 log.dirs 指定。可以使用多路径，使用逗号分隔。如果是多路径，Kafka 会根据“最少使用”原则，把同一个分区的日志片段保存到同一路径下。会往拥有最少数据分区的路径新增分区。

num.recovery.threads.per.data.dir

每数据目录用于日志恢复启动和关闭时的线程数量。因为这些线程只是服务器启动（正常启动和崩溃后重启）和关闭时会用到。所以完全可以设置大量的线程来达到并行操作的目的。注意，这个参数指的是每个日志目录的线程数，比如本参数设置为 8，而 log.dirs 设置为了三个路径，则总共会启动 24 个线程。

auto.create.topics.enable

是否允许自动创建主题。如果设为 true，那么 produce（生产者往主题写消息），consume（消费者从主题读消息）或者 fetch metadata（任意客户端向主题发送元数据请求时）一个不存在的主题时，就会自动创建。缺省为 true。

主题配置

新建主题的默认参数

num.partitions

每个新建主题的分区个数（分区个数只能增加，不能减少 ）。这个参数一般要评估，比如，每秒钟要写入和读取 1000M 数据，如果现在每个消费者每秒钟可以处理 50MB 的数据，那么需要 20 个分区，这样就可以让 20 个消费者同时读取这些分区，从而达到设计目标。（一般经验，把分区大小限制在 25G 之内比较理想）

log.retention.hours

日志保存时间，默认为 7 天（168 小时）。超过这个时间会清理数据。bytes 和 minutes 无论哪个先达到都会触发。与此类似还有 log.retention.minutes 和 log.retention.ms，都设置的话，优先使用具有最小值的那个。（提示：时间保留数据是通过检查磁盘上日志片段文件的最后修改时间来实现的。也就是最后修改时间是指日志片段的关闭时间，也就是文件里最后一个消息的时间戳）

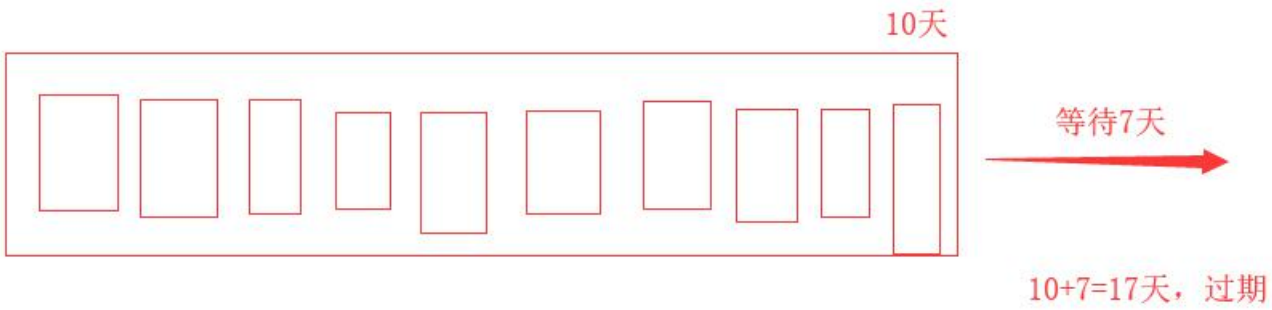
log.retention.bytes

topic 每个分区的最大文件大小，一个 topic 的大小限制 = 分区数*log.retention.bytes。-1 没有大小限制。log.retention.bytes 和 log.retention.minutes 任意一个达到要求，都会执行删除。(注意如果是 log.retention.bytes 先达到了，则是删除多出来的部分数据)，一般不推荐使用最大文件删除策略，而是推荐使用文件过期删除策略。

log.segment.bytes

分区的日志存放在某个目录下诸多文件中，这些文件将分区的日志切分成一段一段的，我们称为日志片段。这个属性就是每个文件的最大尺寸；当尺寸达到这个数值时，就会关闭当前文件，并创建新文件。被关闭的文件就开始等待过期。默认为 1G。

如果一个主题每天只接受 100MB 的消息，那么根据默认设置，需要 10 天才能填满一个文件。而且因为日志片段在关闭之前，消息是不会过期的，所以如果 log.retention.hours 保持默认值的话，那么这个日志片段需要 17 天才过期。因为关闭日志片段需要 10 天，等待过期又需要 7 天。



log.segment.ms

作用和 log.segment.bytes 类似，只不过判断依据是时间。同样的，两个参数，以先到的为准。这个参数默认是不开启的。

message.max.bytes

表示一个服务器能够接收处理的消息的最大字节数，注意这个值 producer 和 consumer 必须设置一致，且不要大于 fetch.message.max.bytes 属性的值(消费者能读取的最大消息,这个值应该大于或等于 message.max.bytes)。该值默认是 1000000 字节，大概 900KB~1MB。如果启动压缩，判断压缩后的值。这个值的大小对性能影响很大，值越大，网络和 IO 的时间越长，还会增加磁盘写入的大小。

Kafka 设计的初衷是迅速处理短小的消息，一般 10K 大小的消息吞吐性能最好（LinkedIn 的 kafka 性能测试）

硬件配置对 Kafka 性能的影响

为 Kafka 选择合适的硬件更像是一门艺术，就跟它的名字一样，我们分别从磁盘、内存、网络和 CPU 上来分析，确定了这些关注点，就可以在预算范围之内选择最优的硬件配置。

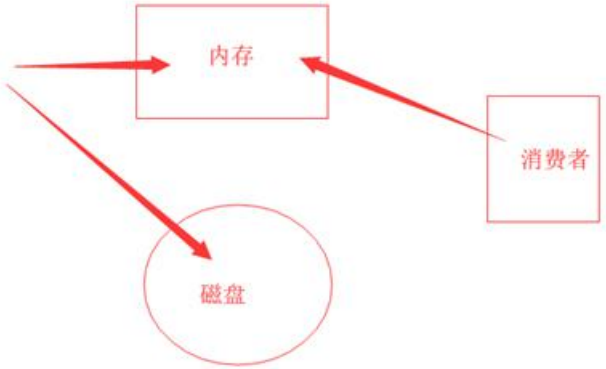
磁盘吞吐量/磁盘容量

磁盘吞吐量（IOPS 每秒的读写次数）会影响生产者的性能。因为生产者的消息必须被提交到服务器保存，大多数的客户端都会一直等待，直到至少有一个服务器确认消息已经成功提交为止。也就是说，磁盘写入速度越快，生成消息的延迟就越低。（SSD 固态贵单个速度快，HDD 机械偏移可以多买几个，设置多个目录加快速度，具体情况具体分析）

磁盘容量的大小，则主要看需要保存的消息数量。如果每天收到 1TB 的数据，并保留 7 天，那么磁盘就需要 7TB 的数据。

内存

Kafka 本身并不需要太大内存，内存则主要是影响消费者性能。在大多数业务情况下，消费者消费的数据一般会从内存（页面缓存，从系统内存中分）中获取，这比在磁盘上读取肯定要快的多。一般来说运行 Kafka 的 JVM 不需要太多的内存，剩余的系统内存可以作为页面缓存，或者用来缓存正在使用的日志片段，所以我们一般 Kafka 不会同其他的重要应用系统部署在一台服务器上，因为他们需要共享页面缓存，这个会降低 Kafka 消费者的性能。



网络

网络吞吐量决定了 Kafka 能够处理的最大数据流量。它和磁盘是制约 Kafka 拓展规模的主要因素。对于生产者、消费者写入数据和读取数据都要瓜分网络流量。同时做集群复制也非常消耗网络。

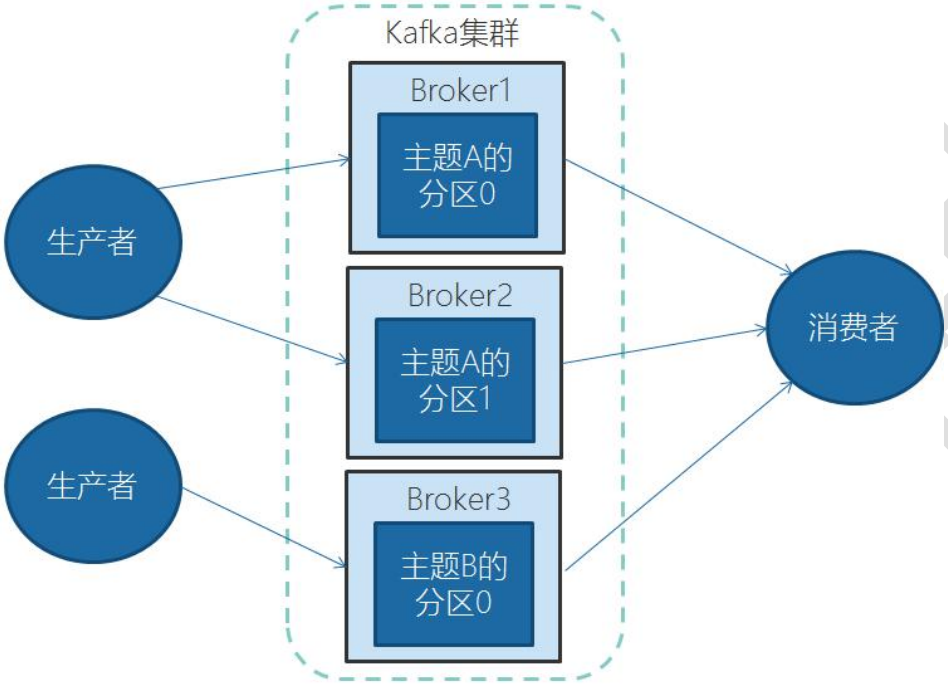
CPU

Kafka 对 cpu 的要求不高，主要是用在对消息解压和压缩上。所以 cpu 的性能不是在使用 Kafka 的首要考虑因素。

总结

我们要为 Kafka 选择合适的硬件时，优先考虑存储，包括存储的大小，然后考虑生产者的性能（也就是磁盘的吞吐量），选好存储以后，再来选择 CPU 和内存就容易得多。网络的选择要根据业务上的情况来定，也是非常重要的一环。

Kafka 的集群



为何需要 Kafka 集群

本地开发，一台 Kafka 足够使用。在实际生产中，集群可以跨服务器进行负载均衡，再则可以使用复制功能来避免单独故障造成的数据丢失。同时集群可以提供高可用性。

如何估算 Kafka 集群中 Broker 的数量

要估量以下几个因素：

需要多少磁盘空间保留数据，和每个 broker 上有多少空间可以用。比如，如果一个集群有 10TB 的数据需要保留，而每个 broker 可以存储 2TB，那么至少需要 5 个 broker。如果启用了数据复制，则还需要一倍的空间，那么这个集群需要 10 个 broker。

集群处理请求的能力。如果因为磁盘吞吐量和内存不足造成性能问题，可以通过扩展 broker 来解决。

Broker 如何加入 Kafka 集群

非常简单，只需要两个参数。第一，配置 zookeeper.connect，第二，为新增的 broker 设置一个集群内的唯一性 id。

Kafka 中的集群是可以动态扩容的。