哈希算法下：哈希算法在分布式系统中有哪些应用

哈希算法的剩余三种应用：负载均衡、数据分片、分布式存储；

哈希算法是如何解决这些分布式问题的

应用五：负载均衡

负载均衡的算法很多，比如轮询、随机、加权轮询等。那如何才能实现一个会话粘滞（session sticky）的负载均衡算法呢？也就是说，同一个客户端在一次会话中的所有请求都路由到同一个服务器上。

最直接的方法是：维护一张映射关系表，这张表的内容是客户端IP地址或者会话ID与服务器编号的映射关系。客户端发出的每次请求，都要先在映射表中查找应该路由到的服务器编号，然后再请求编号对应的服务器。

1. 如果客户端很多，映射表可能会很大，比较浪费内存空间。
2. 客户端上线、下线，服务器扩容、缩容都会导致映射失效，这样维护映射表的成本就会很大；

借助哈希算法。我们可通过哈希算法，对客户端IP地址或者会话ID计算哈希值，将取到的哈希值与服务器列表大小进行取模运算，最终得到的值就是应该被路由到的服务器编号。这样就可以把同一个IP过来的所有请求，都路由到同一个后端服务器上。

应用六：数据分片

1、如何统计“搜索关键词”出现的次数？

假如我们有1T的日志文件，这里面记录了用户的搜索关键词，我们想要快速统计出每个关键词被搜索的次数，怎么办？ 问题两个难点：第一、搜索日志太大，没办法放到一台机器的内存中。第二、如果只用一台机器来处理这么巨大的数据，处理时间会很长。

针对两个难点：我们可以先对数据分片，然后采用多台机器处理的方法，来提高处理速度。

具体思路是：为了提高速度，使用n台机器并行处理。我们从搜索记录的日志文件中，一次读出每个关键词，并且通过哈希函数计算哈希值，然后再跟n取模，最终得到的值，就是应该被分配的机器编号。

这样，哈希值相同的关键词就被分配到同一个机器上。每个机器分别计算关键词出现的次数，最终合并起来就是最终的结果。这样的处理过程也是MapReduce的基本设计思想。

2、如何快速判断图片是否在图库中？

上节讲述的例子中，给每个图片取唯一标识，或构建散列表。

假设我们的图库中有1亿章图片，很显然，单台机器上构建散列表是行不通的。我们可以对数据分片，然后采用多机处理。我们准备n台机器，让每台机器只维护某一部分的图片对应的散列表。我们每次从图库中读取一个图片，计算唯一标识，然后与机器个数n求余取模，得到的值对应要分配的机器编号，然后将这个图片的唯一标识和图片路径发给对应的机器构建散列表。

当我们判断一个图片是否在图库中的时候，我们通过同样的哈希算法，计算这个图片的唯一标识，然后与机器数n求余取模。假设值为k，就去k编号的机器查找。

现在我们来估算一下散列表大约需要多少台机器？ MD5加密哈希值，128比特，16字节。文件路径上限是256字节，平均长度是128字节。假设用链表法解决冲突，存储指针8字节，所以每个数据单元占用152字节。

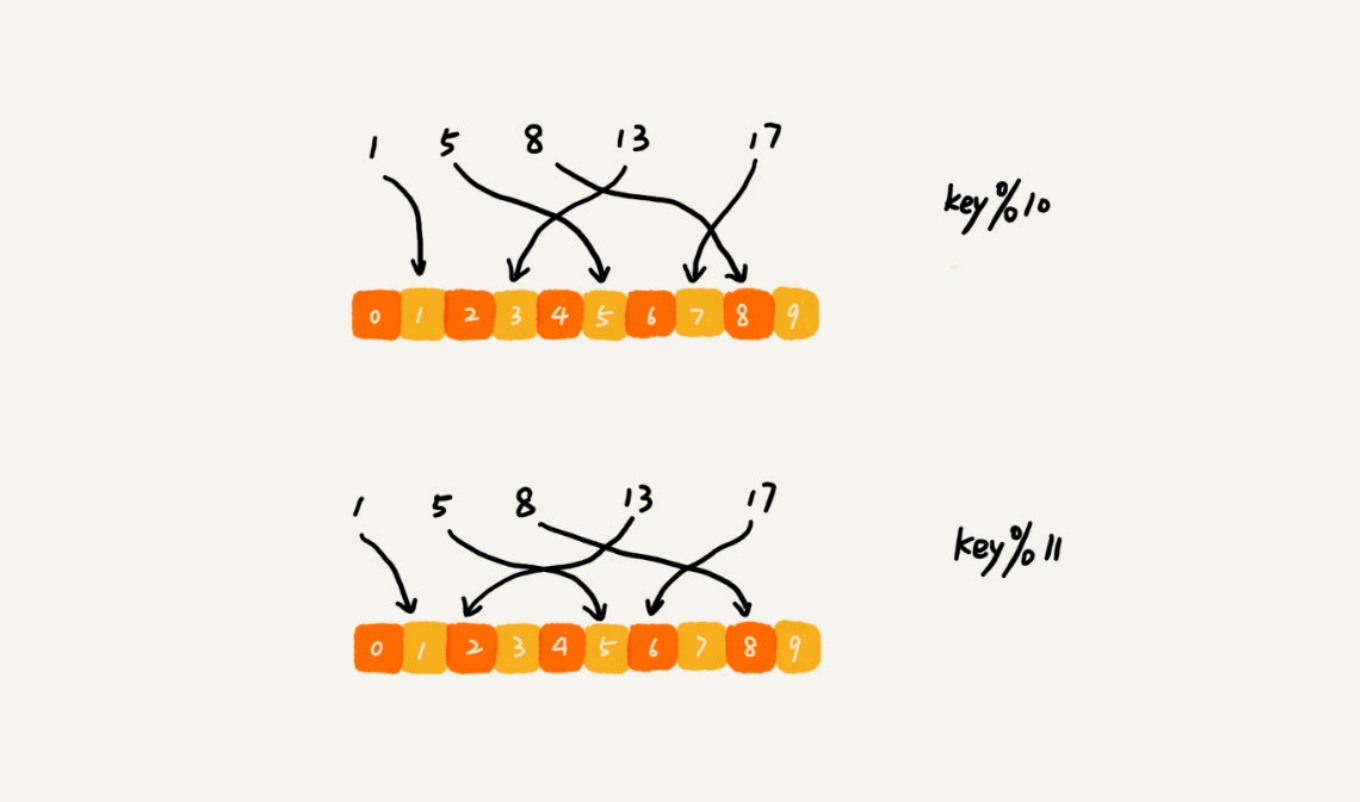
假设一台机器内存大小2GB，散列表的装载因子0.75，那么一台机器可以给大约1000万章图片构建散列表。所以对一亿张图片构建索引就需要大约十几台机器。

应用七：分布式存储

互联网面对都是海量的数据，为了提高数据的读取，写入能力，一般都采用分布式的方式来存储数据，比如分布式缓存，海量数据的缓存，需要将数据分布到多台机器上。

如何决定将那个数据放到那个机器上呢？我们通过哈希算法取哈希值，然后对机器个数取模，最终值就是应该存储的缓存机器编号。

如果数据增多，原来10个机器已经无法承受，我们需要扩容，比如扩大到11个机器，这个时候就有麻烦了。



这样的话所有的数据重新计算哈希值然后搬移到正确的机器上，相当于缓存中数据全部失效，所有数据都会直接去访问数据库，发生雪崩效应，压垮数据库。

这个时间就需要**一致性哈希算法**

假设我们有k个机器，数据的哈希值范围是[0,MAX]。我们将整个范围划分为m个小区间（m远大于k）,每个机器负责m/k个小区间。当有新机器加入的时候，我们就将某几个小区间的数据，从原来的机器中搬移到新的机器中。这样既不用全部重新哈希、搬移数据，也保持了各个机器上数据数量的均衡。

课后思考

这两节我总共讲了七个哈希算法的应用。实际上，我讲的也只是冰山一角，哈希算法还有很多其他的应用，比如网络协议中的 CRC 校验、Git commit id 等等。除了这些，你还能想到其他用到哈希算法的地方吗？