哈希算法在分布式系统中的应用

1.负载均衡

1.1.需求

如何实现一个会话粘滞（session sticky）的负载均衡算法？也就是说，在一次会话中的所有请求都路由到同一个服务器上。

1.2.解决方案

通过哈希算法对客户端IP或会话ID计算哈希值，将取得的哈希值与服务器列表的大小进行取模运算，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过来的请求都路由到同一个后端服务器上。

2.数据分片

2.1.如何统计“搜索关键词”出现的次数?

①需求描述

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，该怎么做呢？

②问题分析

这个问题有两个难点，第一个是搜索的日子很大，没办法放到一台机器的内存中。第二个是只用一台机器来处理这么巨大的数据，处理时间会很长。

③解决方案

先对数据进行分片，然后采用多台（比如n台）机器进行处理。具体做法：从搜索记录的日志文件中依次读取每个关键词，并通过哈希函数计算该关键词的哈希值，然后跟机器的台数n取模，最终得到值就是该关键词应该被分到的机器编号，这样相同的关键词一定会被分配到同一台机器上，数据分配完成后，由多台机器并行进行统计，最后合并起来就是最终结果。

实际上，这里的处理过程也是 MapReduce 的基本设计思想。

2.2.如何快速判断图片是否存在图库中?

①需求描述

假设现在我们的图库中有1亿张图片，如何快速判断图片是否在图库中？基本方式是给每个图片去唯一表示（或者信息摘要），然后构建散列表。

②问题分析

很显然，在单台机器上构建散列表示行不通的，因为单台机器的内存有限，而1亿张图片构建散列表远远超过了单台机器的内存上限。

②解决方案

准备n台机器，让每台机器只维护一部分图片对应的散列表。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，得到的值就对应要分配的机器编号，然后

将这个图片的唯一表示和图片路径发往对应的机器构建散列表。

当我们要判断一个图片是否在图库中时，我们通过同样的哈希算法，计算这个图片的唯一表示，然后与机器个数n求余取模。假设得到的值是k，那就去编号为k的机器构建的散列表中查找。

如何估算给1亿张图片构建散列表大约需要多少台机器？

散列表中每个数据单元包含两个信息，哈希值和图片文件的路径。假设我们通过 MD5 来计算哈希值，那长度就是 128 比特，也就是 16 字节。文件路径长度的上限是 256 字节，我们可以假设平均长度是 128 字节。如果我们用链表法来解决冲突，那还需要存储指针，指针只占用 8 字节。所以，散列表中每个数据单元就占用 152 字节（这里只是估算，并不准确）。

假设一台机器的内存大小为 2GB，散列表的装载因子为 0.75，那一台机器可以给大约 1000 万（2GB*0.75/152）张图片构建散列表。所以，如果要对 1 亿张图片构建索引，需要大约十几台机器。在工程中，这种估算还是很重要的，能让我们事先对需要投入的资源、资金有个大概的了解，能更好地评估解决方案的可行性。

实际上，针对这种海量数据的处理问题，我们都可以采用多机分布式处理。借助这种分片的思路，可以突破单机内存、CPU 等资源的限制。

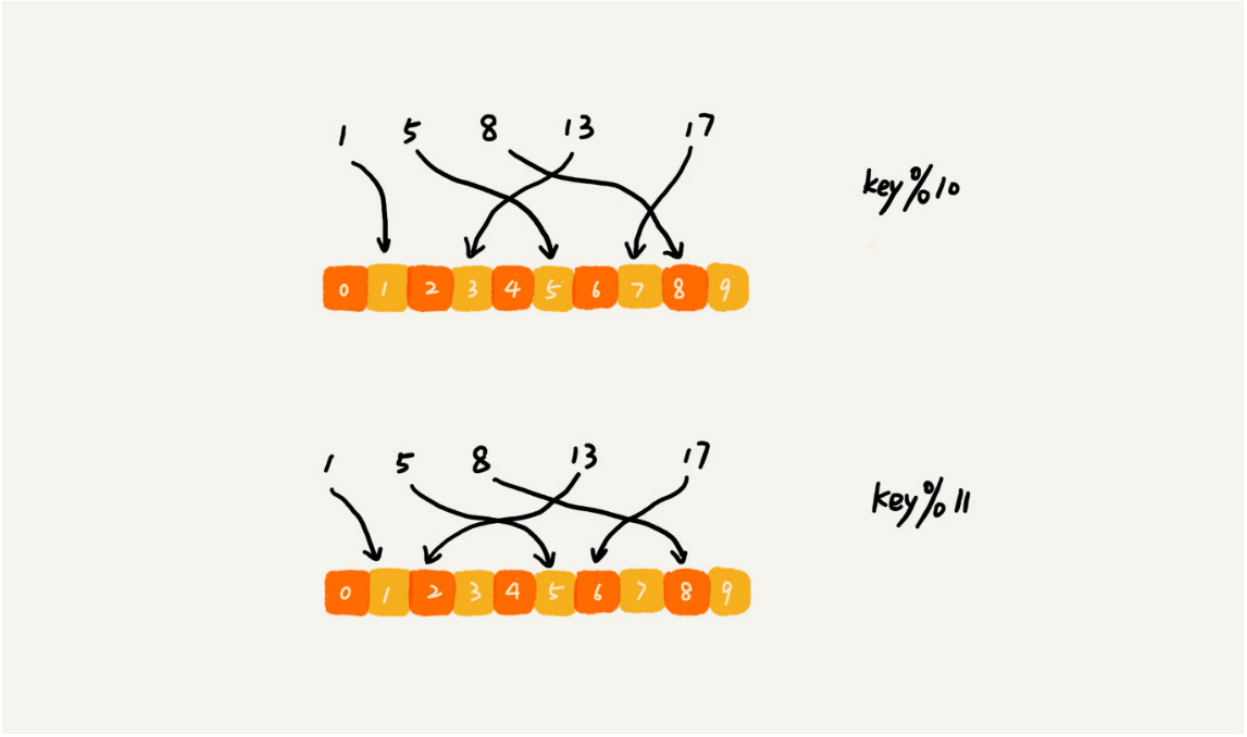
3.分布式存储

3.1.什么是分布式存储?

分布式存储就是将数据存储在多台机器上并提供高效的读取、写入支持。那如何决定将哪个数据放到哪个机器上呢？可以利用数据分片的思想，即通过哈希算法对数据取哈希值，然后对机器个数取模，这个最终值就是应该存储的缓存机器编号。

3.2.遇到的问题是什么?

如果数据持续增多，原来的机器数量已经不能满足需求，就需要增加机器，这时就麻烦了，因为所有的数据都需要重新哈希值进行再次分配。这就相当于，缓存中的数据一下子都失效了，所有的数据请求都会穿透缓存，直接去请求数据库。这样就可能发生雪崩效应，压垮数据库。



3.3.解决方案是什么?

①这时，需要一种方法，使得新加入一个机器后，并不需要做大量的数据搬移。那就是在分布式系统中应用非常广泛的一致性哈希算法。

②一致性哈希算法的基本思想是什么呢？为了说清楚这个问题，我们假设有k个机器，数据的哈希值范围是[0-MAX]，我们将整个范围划分成m个小区间（m远大于k），每个机器复杂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量的均衡。

极客时间文档：https://time.geekbang.org/column/article/67388