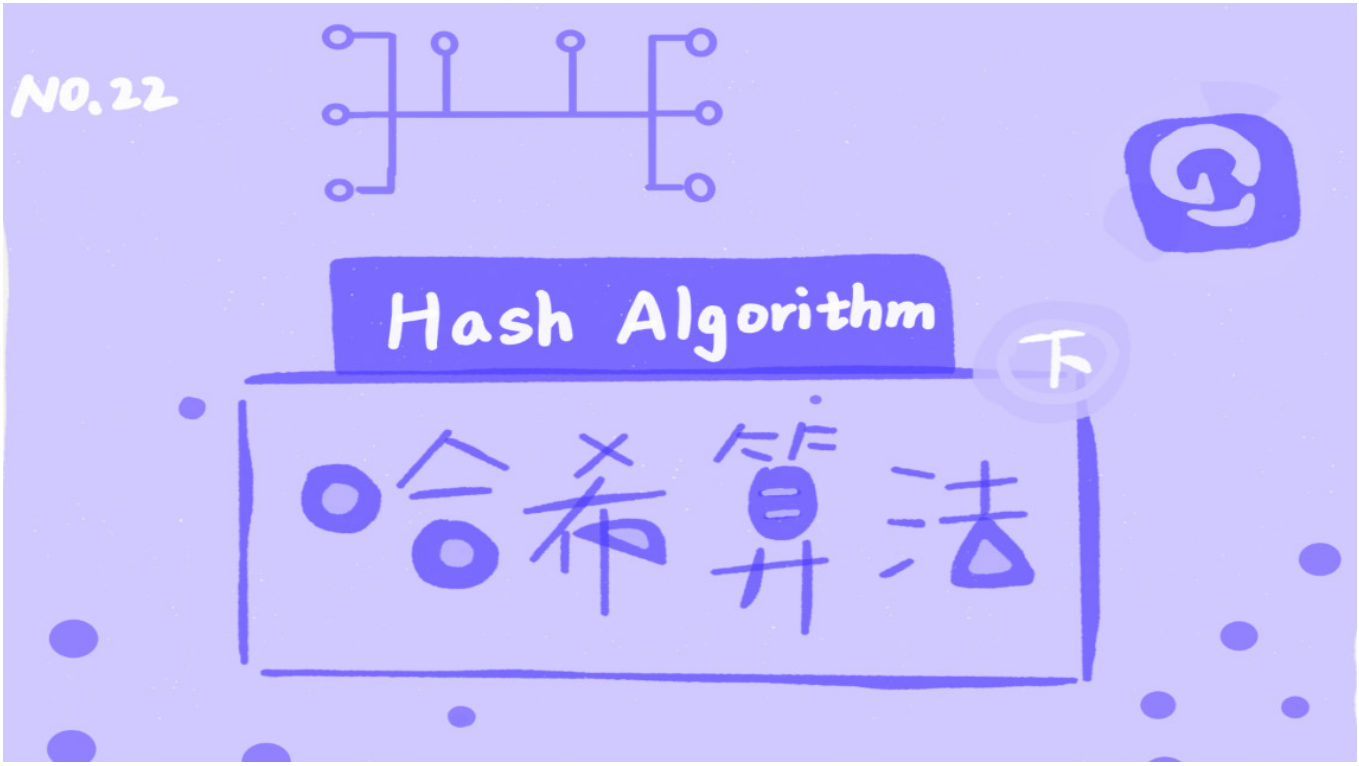
22 | 哈希算法（下）：哈希算法在分布式系统中有哪些应用?

2018-11-09 王争



22 | 哈希算法（下）：哈希算法在分布式系统中有哪些应用?
朗读人：修阳　09′20″ | 4.29M

上一节，我讲了哈希算法的四个应用，它们分别是：安全加密、数据校验、唯一标识、散列函数。今天，我们再来看剩余三种应用：**负载均衡、数据分片、分布式存储**。

你可能已经发现，这三个应用都跟分布式系统有关。没错，今天我就带你看下，**哈希算法是如何解决这些分布式问题的**。

应用五：负载均衡

我们知道，负载均衡算法有很多，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，我们需要在同一个客户端上，在一次会话中的所有请求都路由到同一个服务器上。

最直接的方法就是，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。这种方法简单直观，但也有几个弊端：

- 如果客户端很多，映射表可能会很大，比较浪费内存空间；
- 客户端下线、上线，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；

如果借助哈希算法，这些问题都可以非常完美地解决。**我们可以通过哈希算法，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，我们就可以把同一个 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

哈希算法还可以用于数据的分片。我这里有两个例子。

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，该怎么做呢?

我们来分析一下。这个问题有两个难点，第一个是搜索日志很大，没办法放到一台机器的内存中。第二个难点是，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。

针对这两个难点，**我们可以先对数据进行分片，然后采用多台机器处理的方法，来提高处理速度**。具体的思路是这样的：为了提高处理的速度，我们用 n 台机器并行处理。我们从搜索记录的日志文件中，依次读出每个搜索关键词，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，就是应该被分配到的机器编号。

这样，哈希值相同的搜索关键词就被分配到了同一个机器上。也就是说，同一个搜索关键词会被分配到同一个机器上。每个机器会分别计算关键词出现的次数，最后合并起来就是最终的结果。

实际上，这里的处理过程也是 MapReduce 的基本设计思想。

2. 如何快速判断图片是否在图库中?

如何快速判断图片是否在图库中？上一节我们讲过这个例子，不知道你还记得吗？当时我介绍了一种方法，即给每个图片取唯一标识（或者信息摘要），然后构建散列表。

假设现在我们的图库中有 1 亿张图片，很显然，在单台机器上构建散列表是行不通的。因为单台机器的内存有限，而 1 亿张图片构建散列表显然远远超过了单台机器的内存上限。

我们同样可以对数据进行分片，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标识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，然后与机器个数 n 求余取模。假设得到的值是 k，那就去编号 k 的机器构建的散列表中查找。

现在，我们来估算一下，给这 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。

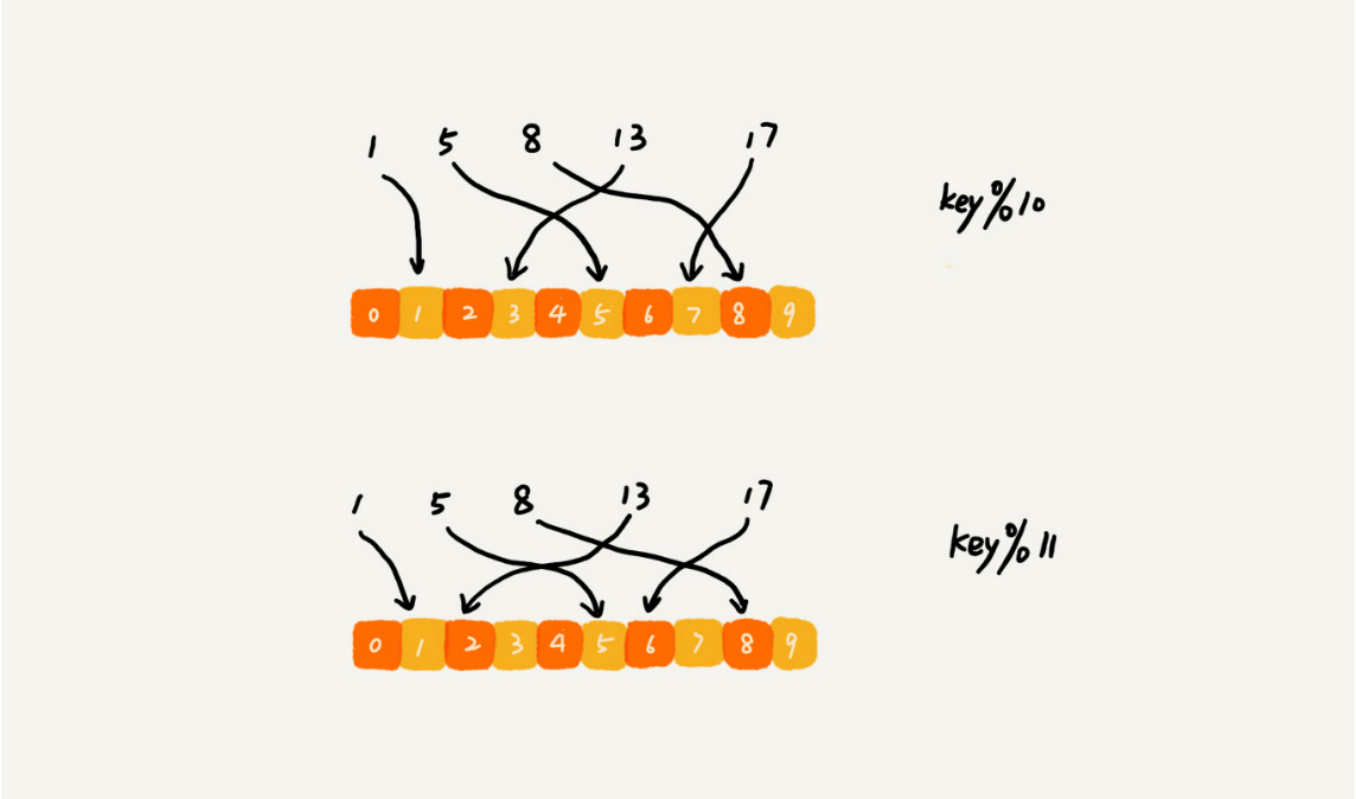
应用七：分布式存储

现在互联网面对的都是海量的数据、海量的用户。我们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。我们有海量的数据需要缓存，所以一个缓存机器肯定是不够的。于是，我们就需要将数据分布在多台机器上。

该如何决定将哪个数据放到哪个机器上呢？我们可以借用前面数据分片的思想，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。

但是，如果数据增多，原来的 10 个机器已经无法承受了，我们就需要扩容了，比如扩到 11 个机器，这时候麻烦就来了。因为，这里并不是简单地加个机器就可以了。

原来的数据是通过与 10 来取模的。比如 13 这个数据，存储在编号为 3 这台机器上。但是新加了一台机器中，我们对数据按照 11 取模，原来 13 这个数据就被分配到 2 号这台机器上了。



因此，所有的数据都要重新计算哈希值，然后重新搬移到正确的机器上。这样就相当于，缓存中的数据一下子就都失效了。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。

所以，我们需要一种方法，使得在新加入一个机器后，并不需要做大量的数据搬移。这时候，一**致性哈希算法**就要登场了。

假设我们有 k 个机器，数据的哈希值的范围是 [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。

一致性哈希算法的基本思想就是这么简单。除此之外，它还会借助一个虚拟的环和虚拟结点，更加优美地实现出来。这里我就不展开讲了，如果感兴趣，你可以看下这个介绍。

除了我们上面讲到的分布式缓存，实际上，一致性哈希算法的应用非常广泛，在很多分布式存储系统中，都可以见到一致性哈希算法的影子。

解答开篇 & 内容小结

这两节的内容理论不多，比较贴近具体的开发。今天我讲了三种哈希算法在分布式系统中的应用，它们分别是：负载均衡、数据分片、分布式存储。

在负载均衡应用中，利用哈希算法替代映射表，可以实现一个会话粘滞的负载均衡策略。在数据分片应用中，通过哈希算法对处理的海量数据进行分片，多机分布式处理，可以突破单机资源的

限制。在分布式存储应用中，利用一致性哈希算法，可以解决缓存等分布式系统的扩容、缩容导致数据大量搬移的难题。

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？

欢迎留言和我分享，我会第一时间给你反馈。

©版权归极客邦科技所有，未经许可不得转载

上一篇 21 | 哈希算法（上）：如何防止数据库中的用户信息被脱库？

写留言

精选留言

Hesher 12

一致性哈希算法没看懂，只能说看完文章知道了有这么个概念可以解决扩容rehash问题

2018-11-09

作者回复

主要是展开讲内容会很多 网上关于一致性哈希算法的文章很多 你可以看下我给的那个链接。这个算法的核心思想非常简单，网上讲的都很复杂 只是为了实现起来优美。

2018-11-09

null 7

一致性哈希算法，举个栗子：
我们钟表有 60 分钟，从 0 开始到 59，共 60 个点。
现在我们将机器往这 60 个点分配，规则如下：
hash(ip) % 60。

假设有 3 台机器 A，B 和 C，分别被分配到了 14，37 和 46 这三个点上。

图片的分配规则类似：
hash(image_id) % 60。
现有 3 张图片 x， y， z，分别被分配到 5，30，50 这三个点。

很明示，图片都没被分配到机器的节点上，怎么办呢?在钟表上顺时钟往前寻找，第一台遇到的机器，就是它的归属。

--- 我是分割线 ---

现在很不凑巧，A B C 三台机器分别分配到 5，10，15 这三个点。这样对 A 是很不公平的吖，要负责存储绝大多数的图片，那这怎么办呢?我们社会主义核心价值观基本内容：和谐、平等、公正。为建设和谐社会努力奋斗！！

为了避免不必要的争端，我们引入“虚拟节点”，每台机器都可以拔一根汗毛，变成若干台，把虚拟节点分散到 60 个点上，归属“虚拟节点”的图片，均保存到它的真身。这样就能解决分配不均匀的问题。

应用时，将 60 替换下即可，如替换为 2的 32 次方。

2018-11-09

ban 5

一致性算法讲的有的有点抽象，不够详细。我网上找到一个漫画图解，各位可以参考一下：https://www.sohu.com/a/158141377_479559

2018-11-09

Geek_fbe6fe 2

跟着作者学习整个数据结构和算法，感觉如醍醐灌顶，好像整个世界被重新打开了，最近也想学习go所以用go实现了到目前为止的所有算法和数据结构，用于自己学习和理解希望对大家有帮助
https://github.com/xiangdong1987/studyAlgorithm
对于一致性算法：我理解是先从整体上将hash 分好区间m 在通过自己维护一套在K台机器上m区间的分布来实现不需要rehash 的扩容方式

2018-11-09

会网络的老鼠 2

上几节讲过扩容冗余算法，可以避免搬移数据，如果对当前n取模未中再对扩容前的m取模，直到都未中再返回值是不是也可以?

2018-11-09

作者回复

👍 也是可以的

2018-11-09

spark 1

感觉评论里好多技术大佬，如果老师能附上一致性哈希算法代码案例就更好了

2018-11-09

cyf 1

哈希值相同的搜索关键词就被分配到了同一个机器上，这里数据是分片存储到不同的机器上的，而同一个机器只搜索固定的关键词，最后的结果会不会不完整？可能我没get到老师的点。

2018-11-09

道 1

希望对一致性哈希有深入的讲解。

2018-11-09

勤劳的小胖子-libo 0

在负载均衡那一块，客户端上线下线和服务器扩容缩容怎么影响映射表呢啊?
这部分没看明白。"如果借助哈希算法，这些问题都可以非常完美地解决。"这个方法也会对服务器列表进行取模运算，那为什么扩容，缩容没影响？难道是应用到了一致性哈希?

2018-11-11

远方夕阳 0

一致性哈希也会存在映射差异的问题， A ,C节点中插入B节点，那么A B之间原先映射到C的请求都会B，这样的情况，是要C分割一些数据给B吗

2018-11-11

jinghaitingfeng 0

这个买的值了

2018-11-11

Demter 0

在负载均衡那一块，客户端上线下线和服务器扩容缩容怎么影响映射表呢啊

2018-11-11

0

老师，MD5计算的哈希值是128位，是不是意味着，用MD5计算小于2^128个不同数据，不会出现哈希冲突?

2018-11-10

Smallfly 0

1 亿张图片在 n 台机器上，是散乱的，而且机器都被存满了，如果不提供额外的空间，能做到对数据进行分片实现高效查找么?

如果不能，额外大概需要几台机器（空间）？

2018-11-10

lovetechlovelife 0

这一节怎么看都像是在为大数据做铺垫哈哈

2018-11-09

美团点评技术团队 0

crc 认证和hash 有什么关系吗 crc 我印象中是校验码 。 方便老师解答下吗

2018-11-09

cweioo 0

分布式，理由到同一个服务器，那万一这台服务器挂了，后面怎么处理，session怎么同步?

2018-11-09

CCC 0

Redis集群就是应用的一致性哈希算法

2018-11-09

沉睡的木木夕 0

一致性哈希算法那里可以用图来进一步解释的，
memcached 就用到了 Consistent Hashing 。在这里发链接会屏蔽么?
https://kb.cnblogs.com/page/42734/ 这里也说到了哈希一致性算法
这里提个问题，应用五那个场景，如果搜索关键词比较集中怎么办？也就是说虽然有多个server，但是根据取余法，相同的关键词肯定会到相同server中，那么在相对坏的情况下（搜索词多，单重复的很多）那么这种算法不仅没有达到相应的并发计算的效果，也浪费了其他server

2018-11-09

Ryoma 0

看到有小伙伴提到将缓存机器扩容与散列表扩容比较，其实这里有一些不同点：
①：散列表扩容：即使使用动态扩容，在下一次扩容之前，数据会逐渐迁移到新的散列表
②：缓存机器扩容：机器扩容一般是人为去加机器，可能会出现连续几次加机器的操作，此时，两次查找也无法解决问题

个人觉得最大的不同是：散列表的扩容至少是double（或者保证下次扩容时，数据全部在现有的散列表中）；而缓存机器扩容一般是加少许，此时就需要一致性hash算法了

2018-11-09

凡 0

最近几节课都没什么代码！纯理论看的有些吃力！然后没只做过移动端开发，对这几个例子理解有点难度！

2018-11-09

魏全运 0

请问一致性哈希算法是先对m取模，然后再对k取模吗？因为不是直接对机器数取模，所以当扩容时不会对所有数据进行重新哈希，老师，我的理解对吗？

2018-11-09

拉欧 0

MySQL分库分表后，根据分片键查询数据

2018-11-09