11，大数据整章注解

2018年4月7日　　19:35

由于ipv4的数量是42亿左右，因此，可以将每个ip用不重复位的32位无符号整数表示，即4个字节
那么10亿个ip，转化为10亿个整数（4字节，需要余约4G内存空间

（二）.哈希函数

1.哈希函数即散列函数

哈希函数的输入域可以是非常大的范围，但是输出域是固定范围。

2.哈希函数的性质：

a.典型的哈希函数都有无线的输入值域
b.输入值相同时，返回值相同，返回值即哈希值
c.输入值不同时，返回值可能一样，也可能不一样
d.不同输入值得到的哈希值，整体均匀的分布在输出域s上。（重要）
前三点性质是哈希函数的基础，最后一点是评价一个哈希函数优劣的关键。

（3）无论哈希函数设计有多么精细，都会产生冲突现象，也就是2个关键字处理函数的结果映射在了同一位置上，因此，有一些方法可以避免冲突。

一个优秀的哈希函数可以使不同输入值得到的哈希值均匀分布，哈希值越均匀的分布在s上，该哈希函数越优秀。这种均匀分布与输入值无关，比如” aaa1”,”aaa2”,”aaa3”，虽然相似，但一个优秀的哈希函数计算出的哈希值应该差异巨大。这样s（输出域）对%m后的结果，也会均匀的分布在0~m-1这个值域上。这一点在哈希函数的应用中是非常重要的。

4.如何设计哈希函数以及如何处理冲突，请看这篇博客：散列查找

（三）.Map-Reduce思想

把大任务分成俩个阶段：

1.Map阶段（分）

通过哈希函数把任务分成若干个子任务，哈希函数是系统自带的或是用户指定的，相同哈希值的任务会被分配到同一个节点上。在分布式系统中，一个节点可以是一个计算节点，也可以是一台计算机。

2.Reduce阶段

分开处理，并行运算，然后合并结果的阶段。

3.Map-Reduce的原理简单，但工程上的实现会遇到很多问题。

a.备份的考虑，分布式存储的设计细节，以及容灾策略
b.任务分配策略与任务进度跟踪的细节设计，以及节点状态的呈现
c.分布式系统多用户权限的控制

4.用Map-Reduce方法统计一篇文章中每个单词出现的个数。

（1）.首先把文章进行预处理，最终得到一篇单词字符串文本。
比如去掉文章中的标点符号；对连字符”-“的处理，如pencil-box，和换行处的连字符；对于缩写的处理，如I’m、don’t等等；大小写的转换等。
（2）.输入文本，生成每个单词词频为1的记录，进入map阶段
对每个单词都生成词频为1的记录。比如(dog,1)、(pig,1)等。此时一个单词可能有多个词频为1的记录，比如说可能有多个(dog,1)的记录。 此时还未合并
（3）.通过哈希函数，将单词文本分流成多个小文件。
根据哈希函数的性质我们知道，单词相同的记录会被分配到一起。
（4）.进入Reduce阶段。
在子任务中，同一种单词的词频进行合并，最后得到所有记录并统一合并。因为每个子任务都是并行处理的，所以效率会比较高。

（四）.常见海量数据处理题目的解题关键

**1. 分而治之，通过哈希函数将大文件分流到机器，或是分流成小文件，对每一个小文件进行处理，然后再把结果给合并起来
2. 常用HashMap，bitmap等数据结构。
3. 难点在于通讯，时间和空间的估算。**

（五）大数据相关题目

【题目1】请对10亿个IPV4的ip地址进行排序，每个ip只会出现一次。每个ip只会出现一次。

关键：只会出现一次，所以可以考虑使用bitmap模型。

IPv4中规定IP地址长度为32（按TCP/IP参考模型划分），即有2^32-1个地址。
IPv6协议的地址长度是128位，全部可分配地址数为2的128次方（2^128）个。

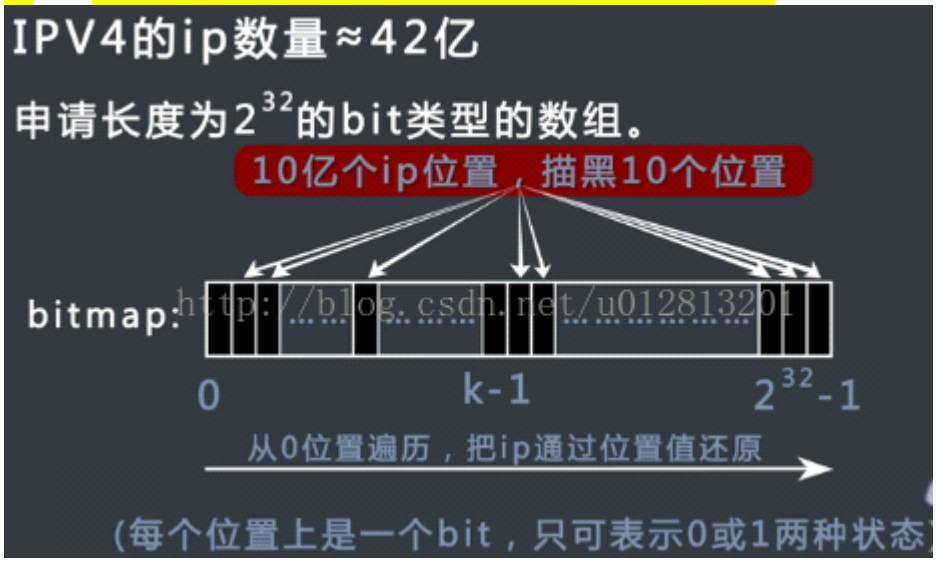
（1）普通方法
IPV4的ip数量大约为42亿，我们要记住2^32约等于42亿！每个ip需要4B来存储，10亿个ip

全部转换为无符号整数，然后使用快速排序方法，然后把10亿个排好序的数字转换为ip地址。需要4GB的内存空间！但是有更好的方法。

（2）.bitmap方法

申请一个长度为2^32的bit类型的数组，每个位置上是一个bit，只可表示0或者1两种状态，空间为512MB。

每个IP地址转化成无符号整数k（注意，k是一位下标），数组下标0~2^32-1与k对应起来。如果k==1,就把bitmap[0]=1；如果k==n,把bitmap[n-1 =1；



（图中的意思应该是，10亿个ip对应的整数k，描黑10亿个位置，这里的k可能是个很大的数，比如2^30次方，毕竟下标范围是很大的。）

如果整数1出现，就把bitmap对应的位置从0变到1，这个数组可以表示任意一个（注意是一个）32位无符号整数是否出现过，

然后把10亿个ip地址全部转换成整数，相应的在bitmap中相应的位置描黑即可，

最后只要从bitmap的零位一直遍历到最后，然后提取出对应为1的下标整数k，再转换成ip地址，就完成了从小到大的排序。空间复杂度很小，时间复杂度O（n）

【题目2】：对10亿个年龄进行排序----桶排序

建立0到200个桶表示年龄从0到200，然后将10亿数据入桶，再依次倒出

【题目3】：有一个包含20亿个全是32位整数的大文件，在其中找到出现次数最多的数。,只用2G内存

【分析】通过哈希表对20亿个整数进行词频统计。哈希表的key是32位的整数，value是出现次数

最坏打算是20亿个（4B）所以一条<k,v>记录是8B。

最多20亿条记录，需要的内存远超于2GB！一条记录需要8B存储，当哈希表的记录数为2亿个时，至少需要1.6GB的内存！

【解决方法】把包含20亿个整数的大文件通过哈希函数分流为16小文件，根据哈希函数的基本性质，相同的整数必将被分配到同一小文件中，并且只要哈希函数足够优秀，得到哈希值均匀，那么所有小文件最坏打算也不超过2亿个整数，这时候在符合内存要求的情况下通过哈希表进行词频统计，最终将每个小文件出现最多的词频进行比较，最终得出总的结论。

（通过分流解决内存限制）

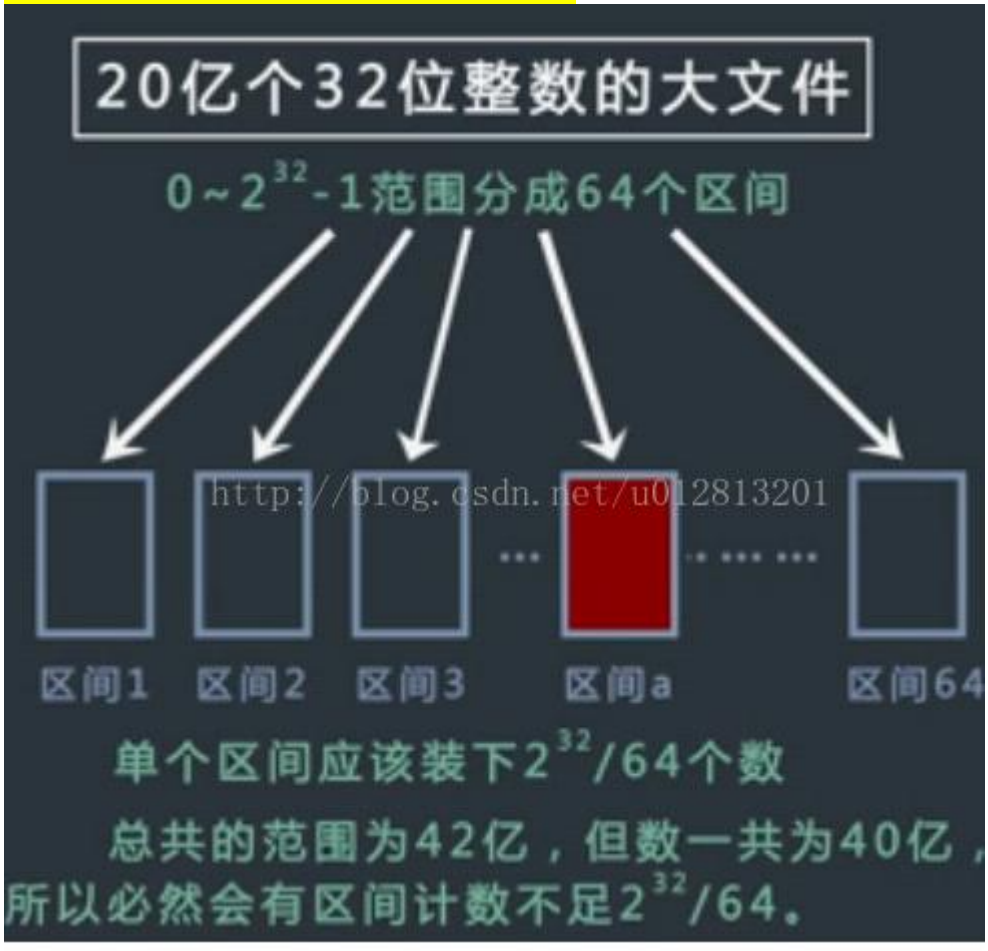
【题目4，】32位无符号整数范围0~4294967295，现在有一个正好包含40亿个整数的文件，所以整个范围中必然有没有出现的数。可以使用最多1GB的内存，怎么找到所有没出想过的数。

【分析】如果用哈希表存储，40亿条记录，40亿*4B 约等于16GB！所以考虑用bitmap模型！2^32位 =512MB，符合内存要求！当出现7000时，在arr[7000]处赋值为1，以此类推，将40亿个整数遍历处理！ 当处理完成之后，在遍历一次，arr[i]==0的下标即为没有出现过数的值！

【题目】内存限制为10MB，但是只用找到一个没有出现过的数即可。

【分析】考虑512/64 =8MB，8MB符合内存要求。所以将2^32个分为64个区间。由题目可知必然有区间计数不足2^32/64。

如何找出计数不足的区间呢？ 通过取模方法，遍历40亿个整数，将整数落在某个区间上对计数变量加一，最终遍历完后随意取出一个区间进行关注。假设文件a中的数据个数不足2^32/64个，那么这里面必然有没出现的。



再遍历40亿个数，对这40亿个数只关注落在取出区间的整数。并且用bitmap统计出区间内的数出现的情况。

总结：

（1）根据内存限制决定区间大小，32位无符号整数2^32=42亿个，哈希表存储的话2^32位=512MB。

（2）而允许区间为10MB，所以要对数据hash分块，512/64=8<10，所以要分成64块。

【题目5，】有一个包含100亿个URL的大文件，假设每个URL占用64B，请找出其中所有重复的URL。

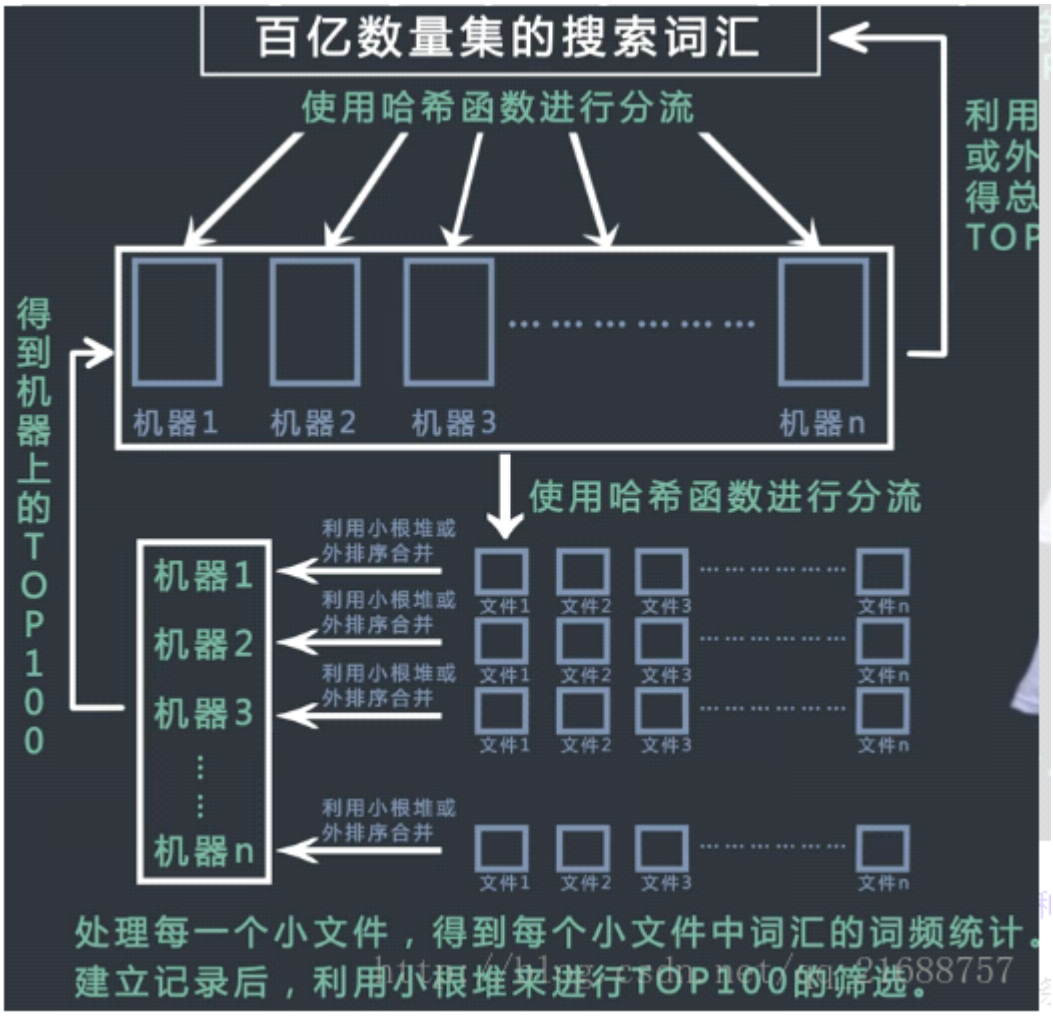
【分析】bitmap使用于给定范围，这样可以固定bit类型数组的大小！此题URL不能固定范围，所以使用哈希函数的分流功能。哈希函数有一个重要性质：同一个输入值将产生同一个哈希值，最终在哈希表中的分布地址也必然相同，利用该性质，我们常常在处理大数据问题时由于空间不够，把大文件通过哈希函数分配到机器，或者通过哈希函数把大文件划分成小文件，一直划分，直到划分的结果满足资源限制的要求。

每个url和其重复的url一定会被分流到同一个小文件里，故在每个小文件里查找重复即可

【题目6，】某搜索公司一天的搜索词汇是海量的（百亿级数据量），请设计出一种每天最热top100词汇的可行方法

【分析】先用哈希函数把百亿数据分流到不同服务器中，如果分流到的数据过大，继续把大文件通过哈希函数分流为子文件。在子文件中用哈希表统计每种词及其词频。遍历哈希表时使用大小为100的最小堆来选出每个小文件的Top100(整体未排序)，再将每个小文件词频的最小堆排序，得到每台机器的Top100。再用每台机器的Top100进行外排序，最终求得整个百亿数据量的top100。
这里写图片描述

（1，每台机器用hash函数分流，分成若干个小文件（每个词和其重复词一定出现在同一个小文件里）
2，每个小文件里用哈希表统计词频。再用最小堆，先随机初始化挑选100个词，然后将最小的调整至堆顶，依次对堆顶元素进行替换。从而得到前top100的词。
3，然后对每个小文件的最小堆排序，得到每台机器的top100，再对每台机器进行外排，从而得到海量数据的top100.



【注意】对于Top100来说，除了哈希函数和哈希表做词频统计外，还常用堆结构和外排序的手段进行处理。

用最小堆：起初前100个条数据建立最小堆，堆顶肯定是词频最低的那条，第101条记录过来时和堆顶比较，如果比堆顶小直接pass，反之替换堆顶，重新调整维持最小堆性质。也就是说堆顶为截止当前位置，词频第100位的那条记录，新记录只要和堆顶比较即可。

【题目7】.40亿个非负整数中找到出现俩次的数

【分析】bitmap模型变形，申请一个2^32 *2长度的bit数组进行操作。

【补充题目】可以使用最多10MB的内存，怎么找到这40亿个整数的中位数！

【分析】用分区间的方式进行处理，长度为2MB的无符号整数数组占用内存空间为8MB。所以将区间数量定义为2^32/2M ,向上取整为2148个区间。
遍历40亿个整数，对每个整数进行遍历，得到每个区间的整数个数（不需要进行词频统计）！假设0~k-1的区间为19.998亿，加上k区间的话肯定就超过20亿，所以中位数在k区间上！

申请2MB长度的数组（占用内存为8MB），遍历40亿整数对K区间的数进行词频统计，最后在区间k上找到第0.002亿个数即可。

【题目8,】工程师常使用服务器集群来设计和实现数据缓存，以下是常见的策略，1，无论是添加，查询还是删除数据，都先将数据的id通过哈希函数转化为一个哈希值，记为key。2，如果当前机器有N台，则计算key%N的值，这个值就是该数据所属的机器编号，无论是添加，删除还是查询操作，都只在这台机器上进行。请分析这种缓存策略可能存在的问题，并提出改进方案。

潜在问题：如果增加和删除机器时，数据迁移的代价很大。所有的数据都需要重新计算一遍hash值，对N取余。数据需重新迁移。

一致性哈希算法，是一种很好的数据缓存设计方案。

现在我们假设数据id通过哈希函数映射为的结果域为0-2^32，
那么我们将0到2^32这些数首尾相连形成一个环。一个数据id在计算出哈希值之后，可以对应到环中的一个位置上。

接下来想象这三台机器也处在这个环中，环中位置，根据机器id计算出哈希值得出。那么一个数据就归属于顺时针离他最近的机器。

这种情况下，添加机器和删除机器的代价都很小。