海量数据处理

---

1. 分而治之/hash映射+hash统计+堆/快速/归并排序
2. 双层桶划分
3. Bloom filter/Bitmap;
4. Trie树/数据库/倒排索引
5. 外排序
6. 分布式处理之Hadoop/Mapreduce

第一部分：关于TOP K算法详解

搜索引擎会通过日志文件把用户每次检索使用的所有检索串都记录下来，每个查询串的长度为1-255字节。 假设目前有一千万个记录（这些查询串的重复度比较高，虽然总数是1千万，但如果除去重复后，不超过3百万个。一个查询串的重复度越高，说明查询它的用户越多，也就是越热门。），请你统计最热门的10个查询串，要求使用的内存不能超过1G。

首先使用HASH表统计每个Query出现的次数，O(N)；然后第二步、采用堆的护具结构找出Top10，N*O（logK）。所以我们最终的时间复杂度是:O(N)+N*O(logk).

第二部分处理海量数据问题之六把密匙

密匙一，分而治之/Hash映射+Hash_map统计+堆/快速/归并排序

1. **海量日志数据，提取出某日访问百度次数最多的那个ip。**

   首先是这一天，并且是访问百度的日志中的IP提取出来。注意到IP是32位的，最多有2^32个IP。同样可以采用映射的方法，比如%1000，把整个大文件映射肠胃1000个小文件，再找出每个小文件中出现频率最大的IP（可以采用hash_map对那1000个文件中的所有IP进行频率统计，然后依次找出各个文件中频率最大的那个IP）及相应频率。然后再在这1000个最大的IP中，找出最大的那个频率最大的IP，即为所求。

2. **寻找热门查询，300万个查询字符串中统计最热门的10个查**询

   原题：搜索引擎会通过日志文件把用户每次检索使用的所有检索串都记录下来，每个查询串的长度为1-255字节。假设目前有一千万个记录（这些查询串的重复度比较高，虽然总数是1千万，但如果除去重复后，不超过3百万个。一个查询串的重复度越高，说明查询它的用户越多，也就是越热门），请你统计最热门的10个查询串，要求使用的内存不能超过1G。

   1. hadhMap统计：先对这批海量数据预处理。具体方法是：维护一个Key为Query的字符串，Value为该Query出现的次数的table，即hash_map（Query，Value），每次读取一个Query，如果该字符串不再Table中，那么假如该字符串，并将Value值置为1；如果该字符串在Table

中，那么将该字符串的技术加一即可。最终我么你在O(N)的时间复杂度完成了对Hash的统计。

2. 堆排序：第二步，借助堆这个数据结构，找出TopK，时间复杂度为N*LogK.即借助堆结构，我们可以在log量级的时间内查找和调整。因此维护一个K（该题目中为10）大小的小根堆，然后遍历300万Query，分别和根元素进行对比。所以最终的时间复杂度是：O（N） + N' * O（logK），（N为1000万，N’为300万）。

3.有一个1G大小的一个文件，里面每一行是一个词，词的大小不超过 16字节，内存大小的限制是1M。返回频率最高的那100个词。

1. 分而治之：
2. hashmap统计加堆排序

4.海量数据分布在100台电脑上，想个办法高效统计出这批数据的Top10

分为两种情况：

一：如果每个相同的数据元素只出现在同一台电脑上

1.堆排序：首先利用堆排序求出每个电脑上的Top10

2.求出每台电脑上的Top10后，可以把这台电脑上的Top10组合起来，共1000个数据，再利用上面类似的方法求出top10

二：相同数据元素可能出现在不同的机器上

第一种解决方案：首先对所有数据进行一次hash，将相同数据hash到同一台机器上去。

第二种暴力求解：直接统计每台电脑上各个元素出现的次数，然后把同一个元素在不同的机器中出现次数相加，最终从所有数据中得出TOP10

5.有10个文件，每个文件1G，每个文件的每一行存放的都是用户的query,每个文件的query都可能重复。要求你按照Query的频度排序

方案一：

1. 首先对十个文件重新hash，将相同的queryhash到相同的文件上。
2. 对每个文件内的query进行hash频率统计，并排序。
3. 然后对十个文件进行归并排序。

方案二：

1. 一般query的总量总是有限的，只是重复的次数比较多而已，可能对于所有的query，一次性就可以加入内存。这样我们就可以采用trie树/hash_map等直接来统计每个query出现的次数，然后按每次出现的次数做快速/堆/归并排序就可以了。

方案三：与方案1类似，但在做完hash，分成多个文件后，可以交给多个文件来处理，采用分布式的架构来处理，最后再进行合并。

6.给定a,b两个文件，各存放50亿个url，每个url各占64字节，内存限制是4G，然你找出a,b文件共同的url

1. 首先采用分而治之的方法，将url分别hash到1000个小文件中，这样每个小文件大约300M。遍历文件B，采取和a相同的方式进行hash，这样只有对应的文件中才有相同的url。
2. hash_set统计：求每个小文件中相同的url时，可以把其中一个小文件的url存储到hashset中。然后遍历另一个小文件的每个url，看其是否在刚才构建的hash_set中，如果是，那么就是共同的url，存到文件中就可以了。

7.怎么在海量数据中找到重复次数最多的一个

方案：先做Hash，然后求摸映射小文件，求出每个小文件中重复次数最多的一个，并记录重复次数。然后找出上一步求出数据中重复次数最多的一个就是所求。

8.100w个数中找出最大的100个数

方案一：采用局部淘汰发。选取前100个元素，并排序，记为序列L。然后依次扫描剩余元素Z，与排好序的100个元素中的最小元素比，如果比这个最小的大，那么把最小的元素删除，并利用插入排序，把x插入到序列L中。

方案二：采用快拍思想每次分割之后，每次分割只考虑比轴大的那一部分。知道比轴大的一部分在100多的时候，采用传统排序算法排序，取前100个。复杂度为O(100W*100)

方案三：用一个含100个元素的最小堆完成。

密匙二、多层划分

多层划分-----本质上还是分而治之的思想，重在“分的技巧上”！

使用范围：第K大，中位数，不重复或重复数字

基本原理及要点：因为元素的范围很大，不能用直接寻址表，所以通过多次划分，逐步确定范围最后在一个可以结束的范围内进行。

2.5亿个整数中找出不重复的整数的个数，内存空间不足以容纳这2.5亿个整数

5亿个int中找出他们的中位数

思路一：这个例子比上面那个更明显。首先我们将int划分为2^16个区域，然后读取数据统计落到各个区域里的数的个数，之后我们根据统计结果就可以判断中位数落到那个区域了，同时知道这个区域中的第几大数刚好是中位数。然后第二次扫描我们只统计落在这个区域的那些数就可以了。

思路二：同样需要做两遍统计，如果数据存在硬盘上，就需要读取两次。

方法同基数排序有些像，开一个大小为65536的Int数组，第一遍读取，统计Int32的高16位的情况，也就是0-65535，都算作0,65536

131071都算作1。就相当于用该数除以65536。Int32 除以 65536的结果不会超过65536种情况，因此开一个长度为65536的数组计数就可以。每读取一个数，数组中对应的计数+1，考虑有负数的情况，需要将结果加32768后，记录在相应的数组内。第一遍统计之后，遍历数组，逐个累加统计，看中位数处于哪个区间，比如处于区间k，那么0- k-1的区间里数字的数量sum应该<n/2（2.5亿）。而k+1 - 65535的计数和也<n/2，第二遍统计同上面的方法类似，但这次只统计处于区间k的情况，也就是说(x / 65536) + 32768 =k。统计只统计低16位的情况。并且利用刚才统计的sum，比如sum = 2.49亿，那么现在就是要在低16位里面找100万个数(2.5亿-2.49亿)。这次计数之后，再统计一下，看中位数所处的区间，最后将高位和低位组合一下就是结果了。

密匙三：Bloom filter/Bitmap

基本原理及要点：　　对于原理来说很简单，位数组+k个独立hash函数。将hash函数对应的值的位数组置1，查找时如果发现所有hash函数对应位都是1说明存在，很明显这个过程并不保证查找的结果是100%正确的。同时也不支持删除一个已经插入的关键字，因为该关键字对应的位会牵动到其他的关键字。所以一个简单的改进就是 counting Bloom filter，用一个counter数组代替位数组，就可以支持删除了。

给你A，B两个文件，个存放50亿条URL，每条URL占用64字节，内存限制是4G，让你找出A，B文件的共同URL。如果是三个乃至n个文件呢

根据这个问题我们来计算下内存的占用，4G=2^32大概是40亿*8大概是340亿，n=50亿，如果按出错率0.01算需要的大概是650亿个bit。现在可用的是340亿，相差并不多，这样可能会使出错率上升些。另外如果这些urlip是一一对应的，就可以转换成ip，则大大简单了。

同时，上文的第5题：给定a、b两个文件，各存放50亿个url，每个url各占64字节，内存限制是4G，让你找出a、b文件共同的url？如果允许有一定的错误率，可以使用Bloom filter，4G内存大概可以表示340亿bit。将其中一个文件中的url使用Bloom filter映射为这340亿bit，然后挨个读取另外一个文件的url，检查是否与Bloom filter，如果是，那么该url应该是共同的url（注意会有一定的错误率）。”

使用bitmap的方法解决问题

在2.5亿个整数中找出不重复的整数，注，内存不足以容纳这2.5亿个整数

方案1：采用2-Bitmap（每个数分配2bit，00表示不存在，01表示出现一次，10表示多次，11无意义）进行，共需内存2^32 * 2 bit=1 GB内存，还可以接受。然后扫描这2.5亿个整数，查看Bitmap中相对应位，如果是00变01，01变10，10保持不变。所描完事后，查看bitmap，把对应位是01的整数输出即可。 方案2：也可采用与第1题类似的方法，进行划分小文件的方法。然后在小文件中找出不重复的整数，并排序。然后再进行归并，注意去除重复的元素。”

给40亿个不重复的unsigned int的整数，没排过序u，然后再给一个数，如何快速判断这个数是否在哪40亿个数中?

方案一：用位图/Bitmap的方法，申请512M的内存，一个bit为代表一个unsigned int值。读入40亿个整数，设置相应的bit为，读入要查询的数据，查看相应的bit为是否为1，为1表示存在，为0表示不存在。

密匙4 Tire树/数据库/倒排索引

密匙5： 外排序

密匙6：分布式处理值MapReduce

Redis将整个key的数值域分为4096个hash槽，每个节点上可以存储一个或多个hash槽，也就是说当前Redis Cluster支持的最大节点数是4096.Redis Cluster使用的分布式算法也很简单：crc 16（key）%HASH_SLOTS_NUMBER。整体的设计可总结为：

- 数据hash分布在不同的Redis节点实例上；
- M/S的切换采用Sentinel；
- 写：只会写master Instance，从sentinel获取当前的master Instance；
- 读：从Redis Node中基于权重选取一个Redis Instance读取，失败/超时则轮询其他Instance；Redis本身就很好的支持读写分离，在单进程的I/O场景下，可以有效的避免主库的阻塞风险；
- 通过RPC服务访问，RPC server端封装了Redis客户端，客户端基于Jedis开发。