不同的机器上，具体多少台机器由面试官规定或者由更多的限制来决定。对每一台机器来说，如果分到的数据量依然很大，比如，内存不够或其他问题，可以再用哈希函数把每台机器的分流文件拆成更小的文件处理。处理每一个小文件的时候，哈希表统计每种词及其词频，哈希表记录建立完成后，再遍历哈希表，遍历哈希表的过程中使用大小为 100 的小根堆来选出每一个小文件的 top 100（整体未排序的 top 100）。每一个小文件都有自己词频的小根堆（整体未排序的 top 100），将小根堆里的词按照词频排序，就得到了每个小文件的排序后 top 100。然后把各个小文件排序后的 top 100 进行外排序或者继续利用小根堆，就可以选出每台机器上的 top 100。不同机器之间的 top100 再进行外排序或者继续利用小根堆，最终求出整个百亿数据量中的 top 100。对于 top *K* 的问题，除哈希函数分流和用哈希表做词频统计之外，还经常用堆结构和外排序的手段进行处理。

40 亿个非负整数中找到出现两次的数和所有数的中位数

【题目】

32 位无符号整数的范围是 0～4294967295，现在有 40 亿个无符号整数，可以使用最多 1GB 的内存，找出所有出现了两次的数。

【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？

【难度】

尉 ★★☆☆

【解答】

对于原问题，可以用 bit map 的方式来表示数出现的情况。具体地说，是申请一个长度为 4294967295×2 的 bit 类型的数组 bitArr，用 2 个位置表示一个数出现的词频，1B 占用 8 个 bit，所以长度为 4294967295×2 的 bit 类型的数组占用 1GB 空间。怎么使用这个 bitArr 数组呢？遍历这 40 亿个无符号数，如果初次遇到 num，就把 bitArr[num*2 + 1]和 bitArr[num*2]设置为 01，如果第二次遇到 num，就把 bitArr[num*2+1]和 bitArr[num*2]设置为 10，如果第三次遇到 num，就把 bitArr[num*2+1]和 bitArr[num*2]设置为 11。以后再遇

到 num，发现此时 bitArr[num*2+1]和 bitArr[num*2]已经被设置为 11，就不再做任何设置。遍历完成后，再依次遍历 bitArr，如果发现 bitArr[i*2+1]和 bitArr[i*2]设置为 10，那么 *i* 就是出现了两次的数。

对于补充问题，用分区间的方式处理，长度为 2MB 的无符号整型数组占用的空间为 8MB，所以将区间的数量定为 4294967295/2M，向上取整为 2148 个区间。第 0 区间为 0~2*M*-1，第 1 区间为 2*M*~4*M*-1，第 *i* 区间为 2*M*×*i*~2*M*×(*i*+1)-1……

申请一个长度为 2148 的无符号整型数组 arr[0..2147]，arr[i]表示第 *i* 区间有多少个数。arr 必然小于 10MB。然后遍历 40 亿个数，如果遍历到当前数为 num，先看 num 落在哪个区间上（num/2M），然后将对应的进行 arr[num/2M]++操作。这样遍历下来，就得到了每一个区间的数的出现状况，通过累加每个区间的出现次数，就可以找到 40 亿个数的中位数（也就是第 20 亿个数）到底落在哪个区间上。比如，0~*K*-1 区间上数的个数为 19.998 亿，但是发现当加上第 *K* 个区间上数的个数之后就超过了 20 亿，那么可以知道第 20 亿个数是第 *K* 区间上的数，并且可以知道第 20 亿个数是第 *K* 区间上的第 0.002 亿个数。

接下来申请一个长度为 2MB 的无符号整型数组 countArr[0..2M-1]，占用空间 8MB。然后再遍历 40 亿个数，此时只关心处在第 *K* 区间的数记为 numi，其他的数省略，然后将 countArr[numi-K*2M]++，也就是只对第 *K* 区间的数做频率统计。这次遍历完 40 亿个数之后，就得到了第 *K* 区间的词频统计结果 countArr，最后只在第 *K* 区间上找到第 0.002 亿个数即可。

一致性哈希算法的基本原理

【题目】

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 通过哈希函数转换成一个哈希值，记为 key。

2．如果目前机器有 *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带来的问题，并提出改进的方案。