海量数据处理

原创 小神仙 在下小神仙 1周前

hello~ 大家好，我是 小神仙 ~

相信大家校招面试的时候都会遇到海量数据处理的面试题吧~

当年我作为一个非科班的学生，第一次遇到的时候直接给我整蒙了。

当前的自己确实是不会，但是小神仙之前也和大家说过，闻道有先后。

现在不会不代表自己以后不会。

现在小神仙把这个整理出来~ 也分享给大家。

如何从海量的 **URL** 中找出相同的 **URL**?

问题描述

给定 a、b 两个文件，各存放 50 亿个 URL，每个 URL 各占 64B，内存限制是 4G。

请找出 a、b 两个文件共同的 URL。

解决思路

每个 URL 占 64B，那么 50 亿个 URL 占用的空间大小约为 320GB。（5,000,000,000 * 64 B ≈ 5 GB * 64 = 320 GB

由于内存大小只有 4G，因此，我们不可能一次性把所有 **URL** 加载到内存中处理。

对于这种类型的题目，一般采用分治策略。

什么是分治策略呢？

就是把大的分成小的，就像切西瓜一样，一次没办法吃完一个西瓜，所以我们只能切成一块一快的慢慢消化。

对比到这道题，我们就需要把一个文件中的 URL 按照某个特征划分为多个小文件，使得每个小文件大小不超过 4G，这样就可以把这个小文件读到内存中进行处理了呀。

思路如下：

1 首先遍历文件 a，对遍历到的 URL 求 `hash(URL) % 1000`，根据计算结果把遍历到的 URL 存储到 a0, a1, a2, ..., a999，这样每个大小约为 300MB。

2 使用同样的方法遍历文件 b，把文件 b 中的 URL 分别存储到文件 b0, b1, b2, ..., b999 中。

3 这样处理过后，所有可能相同的 URL 就会落在对应的小文件中啦。因为处理的方式是一样的，所以 **a0** 对应 **b0, ..., a999** 对应 **b999**，不对应的小文件不可能有相同的 **URL**。

那么接下来，我们只需要求出这 1000 对小文件中相同的 URL 就好了。（这样的话我们就把一个处理海量数据的问题转化成了一个小量级数据的问题。

接着遍历 ai(`i∈[0,999]`)，把 URL 存储到一个 HashSet 集合中。然后遍历 bi 中每个 URL，看在 HashSet 集合中是否存在，若存在，说明这就是共同的 URL，可以把这个 URL 保存到一个单独的文件中。

如何从海量数据中找出高频词？

问题描述

有一个 1GB 大小的文件，文件里每一行是一个词，每个词的大小不超过 16B，内存大小限制是 1MB，要求返回频数最高的 100 个词(Top 100)。

解决思路

这道题和上一道题目的解法是一样的。同样也是内存不够的问题，小伙伴们下次看到内存不够的问题，直接想到要分治的办法哈~

思路如下：

首先遍历大文件，对遍历到的每个词 x，执行 `hash(x) % 5000`，将结果为 i 的词存放到文件 Ai 中。

遍历结束后，我们可以得到 5000 个小文件。

每个小文件的大小为 200KB 左右。

如果有的小文件大小仍然超过 1MB，则采用同样的方式继续进行分解。

接着统计每个小文件中出现频数最高的 100 个词。

最简单的方式是使用 `HashMap` 来实现。其中 key 为词，value 为该词出现的频率。具体方法是：对于遍历到的词 x，如果在 map 中不存在，则执行 `map.put(x, 1)` 若存在，则执行 `map.put(x, map.get(x)+1)`，将该词频数加 1。

上面我们统计了每个小文件单词出现的频数。

接下来，我们可以通过维护一个小顶堆来找出所有词中出现频数最高的 100 个。

具体方法是：依次遍历每个小文件，构建一个小顶堆，堆大小为 100。如果遍历到的词的出现次数大于堆顶词的出现次数，则用新词替换堆顶的词，然后重新调整为小顶堆，遍历结束后，小顶堆上的词就是出现频数最高的 100 个词。

如何找出某一天访问百度网站最多的 IP？

问题描述

现有海量日志数据保存在一个超大文件中，该文件无法直接读入内存，要求从中提取某天访问百度次数最多的那个 IP。

解决思路

这道题只关心某一天访问百度最多的 IP，因此，可以首先对文件进行一次遍历，把这一天访问百度 IP 的相关信息记录到一个单独的大文件中。

接下来采用的方法与上一题一样，大致就是先对 IP 进行哈希映射，接着使用 HashMap 统计重复 IP 的次数，最后计算出重复次数最多的 IP。

如何在大量的数据中判断一个数是否存在？

题目描述

给定 40 亿个不重复的没排过序的 unsigned int 型整数，然后再给定一个数，如何快速判断这个数是否在这 40 亿个整数当中？

解答思路

40 亿个不重复整数，我们用 40 亿个 bit 来表示，初始位均为 0，那么总共需要内存：4,000,000,000b≈512M。

我们读取这 40 亿个整数，将对应的 bit 设置为 1。接着读取要查询的数，查看相应位是否为 1，如果为 1 表示存在，如果为 0 表示不存在。

这种方法我们一般也叫 位图法。这种方法在数据处理的场景下也很常见。

如何统计不同电话号码的个数？

题目描述

已知某个文件内包含一些电话号码，每个号码为 8 位数字，统计不同号码的个数。

解答思路

这道题本质还是求解数据重复的问题，对于这类问题，一般首先考虑位图法。

对于本题，8 位电话号码可以表示的号码个数为 10^8 个，即 1 亿个。

我们每个号码用一个 bit 来表示，则总共需要 1 亿个 bit，内存占用约 100M。

思路如下：

申请一个位图数组，长度为 1 亿，初始化为 0。然后遍历所有电话号码，把号码对应的位图中的位置置为 1。

遍历完成后，如果 bit 为 1，则表示这个电话号码在文件中存在，否则不存在。

bit 值为 **1** 的数量即为 不同电话号码的个数。

方法总结

求解数据重复问题，记得考虑位图法。

如何按照 **query** 的频度排序？

题目描述

有 10 个文件，每个文件大小为 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。要求按照 query 的频度排序。

解答思路

如果 query 的重复度比较大，可以考虑一次性把所有 query 读入内存中处理；如果 query 的重复率不高，那么可用内存不足以容纳所有的 query，这时候就需要采用分治法或其他的方法来解决。

方法一：**HashMap** 法

如果 query 重复率高，说明不同 query 总数比较小，可以考虑把所有的 query 都加载到内存中的 HashMap 中。接着就可以按照 query 出现的次数进行排序。

方法二：分治法

分治法需要根据数据量大小以及可用内存的大小来确定问题划分的规模。对于这道题，可以顺序遍历 10 个文件中的 query，通过 Hash 函数 `hash(query) % 10` 把这些 query 划分到 10 个小文件中。之后对每个小文件使用 HashMap 统计 query 出现次数，根据次数排序并写入到零外一个单独文件中。

接着对所有文件按照 query 的次数进行排序，这里可以使用归并排序（由于无法把所有 query 都读入内存，因此需要使用外排序）

好了，海量数据处理基本上是这些面试题了。

一般的解题思路也是三种。位图法 + 分治法 + 堆。

大家一定要记住喽 ~

周末愉快，小神仙好久没买新衣服了，准备去商场，溜了。。。。