教你如何迅速秒杀99%的海量数据处理面试题

前言

一般而言，标题有“秒杀”，“史上最全/最强”等词汇的往往都脱不了哗众取宠之嫌，但进一步来讲，如果读者读罢此文，却无任何收获，那么，我也甘愿背负这样的罪名，:-)，同时，此文可以看做是对这篇文章：十道海量数据处理面试题与十个方法大总结的一般抽象性总结。

毕竟受文章和理论之限，本文摒弃绝大部分的细节，只谈方法/模式论，且注重用最通俗最直白的语言阐述相关问题。最后，有一点必须强调的是，全文行文是基于面试题的分析基础之上的，具体实践过程中，还是得具体情况具体分析，且场景也远比本文所述的任何一种场景复杂得多。若有任何问题，欢迎随时不吝赐教。谢谢。

何谓海量数据处理？

所谓海量数据处理，其实很简单，海量，海量，何谓海量，就是数据量太大，所以导致要么是无法在较短时间内迅速解决，要么是数据太大，导致无法一次性装入内存。

那解决办法呢?针对时间，我们可以采用巧妙的算法搭配合适的数据结构，如 Bloom filter/Hash/bit-map/堆/数据库或倒排索引/tire/，针对空间，无非就一个办法：大而化小：分而治之/hash 映射，你不是说规模太大嘛，那简单啊，就把规模大化为规模小的，各个击破不就完了嘛。

至于所谓的单机及集群问题，通俗点来讲，单机就是处理装载数据的机器有限(只要考虑 cpu，内存，硬盘的数据交互)，而集群，机器有多辆，适合分布式处理，并行计算(更多考虑节点和节点间的数据交互)。

再者，通过本 blog 内的有关海量数据处理的两篇文章，我们已经大致知道，处理海量数据问题，无非就是：

1 分而治之/hash 映射 + hash 统计 + 堆/快速/归并排序；
2 Bloom filter/Bitmap；
3 Trie 树/数据库/倒排索引；
4 外排序；
5 分布式处理之 hadoop/mapreduce。

本文接下来的部分，便针对这5种方法模式结合对应的海量数据处理面试题分别具体阐述。

密匙一、分而治之/hash 映射 + hash 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**。既然是海量数据处理，那么可想而知，给我们的数据那就一定是海量的。针对这个数据的海量，我们如何着手呢?对的，无非就是分而治之/hash 映射 + hash 统计 + 堆/快速/归并排序，说白了，就是先映射，而后统计，最后排序：

6 分而治之/hash 映射：针对数据太大，内存受限，智能是：把大文件化成(取模映射)小文件，即16字方针：大而化小，各个击破，缩小规模，逐个解决
7 hash 统计：当大文件转化了小文件，那么我们便可以采用常规的 hashmap(ip，value)来进行频率统计。

8 堆/快速排序：统计完了之后，便进行排序(可采取堆排序)，得到次数最多的 IP。

具体而论，则是：“首先是这一天，并且是访问百度的日志中的 IP 取出来，逐个写入到一个大文件中。注意到 IP 是32位的，最多有个2^32 方法，比如模1000，把整个大文件映射为1000个小文件，再找出每个小文中出 hash_map 进行频率统计，然后再找出频率最大的几个）及相应的频率。然后 找出那个频率最大的 IP，即为所求。”--十道海量数据处理面试题与十个方法大总结。

找到第 k 个
找到第一个

但如果数据规模比较小，能一次性装入内存呢?比如下面的这道题，题目中说，虽然有一千万个 Query，但是由于重复度比较高，因此事实上只有300万的 Query，每个 Query255Byte，因此我们可以考虑把他们都放进内存中去，而现在只是需要一个合适的数据结构，在这里，Hash Table 绝对是我们优先的选择。OK，请看第2题：

2、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。

前面说了，数据比较小，摒弃分而治之/hash 映射的方法，直接上 hash 统计，然后排序。So，

9 hash 统计先对这批海量数据预处理(维护一个 Key 为 Query 字串，Value 为该 Query 出现次数的 HashTable，即 hashmap(Query，Value)，每次读取一个 Query，如果该字串不在 Table 中，那么加入该字串，并且将 Value 值设为1；如果该字串在 Table 中，那么将该字串的计数加一即可。最终我们在 O(N)的时间复杂度内用 Hash 表完成了统计；

10 堆排序：第二步、借助堆这个数据结构，找出 Top K，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/移动。因此，维护一个 K(该题目中是10)大小的小根堆，然后遍历300万的 Query，分别和根元素进行对比所以，我们最终的时间复杂度是：O（N） + N'*O（logK），（N 为1000万，N’为300万）。

别忘了这篇文章中所述的堆排序思路：“维护 k 个元素的最小堆，即用容量为 k 的最小堆存储最先遍历到的 k 个数，并假设它们即是最大的 k 个数，建堆费时 O(k)，并调整堆(费时 O(logk))后，有 k1>k2>...kmin（kmin 设为小顶堆中最小元素）。继续遍历数列，每次遍历一个元素 x，与堆顶元素比较，若 x>kmin，则更新堆（用时 logk），否则不更新堆。这样下来，总费时 O（k*logk+（n-k）*logk）=O（n*logk）。此方法得益于在堆中，查找等各项操作时间复杂度均为 logk。”--第三章、寻找最小的 k 个数。

当然，你也可以采用 trie 树，关键字域存该查询串出现的次数，没有出现为0。最后用10个元素的最小推来对出现频率进行排序。

由上面这两个例题，分而治之 + hash 统计 + 堆/快速排序这个套路，我们已经开始有了屡试不爽的感觉。下面，再拿几道再多多验证下。请看第3题：

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，内存限制大小是**1M**。返回频数最高的**100**个词。

又是文件很大，又是内存受限，咋办?还能怎么办呢?无非还是

11 分而治之/hash 映射：顺序读文件中，对于每个词 x，取 hash(x)%5000，然后按照该值存到5000个小文件（记为 x0,x1,...x4999）中。这样每个文件大概是200k 左右。如果其中的有的文件超过了1M 大小，还可以按照类似的方法继续往下分，直到分解得到的小文件的大小都不超过1M。

12 hash 统计：对每个小文件，采用 trie 树/hash_map 等统计每个文件中出现的词以及相应的频率。

13 堆/归并排序：取出出现频率最大的100个词（可以用含100个结点的最小堆），并把100个词及相应的频率存入文件，这样又得到了5000个文件。最后就是把这5000个文件进行归并（类似与归并排序）的过程了。

4、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。

直接上：

14 hash 映射：顺序读取10个文件，按照 hash(query)%10的结果将 query 写入到另外10个文件（记为）中。这样新生成的文件每个的大小大约也1G（假设 hash 函数是随机的）。

15 hash 统计：找一台内存在2G 左右的机器，依次对用 hash_map(query, query_count)来统计每个 query 出现的次数。注：hash_map(query,query_count)是用来统计每个 query 的出现次数，不是存储他们的值，出现一次，则 count+1。

16 堆/快速/归并排序：利用快速/堆/归并排序按照出现次数进行排序。将排序好的 query 和对应的 query_cout 输出到文件中。这样得到了10个排好序的文件（记为）。对这10个文件进行归并排序（内排序与外排序相结合）。

除此之外，此题还有以下两个方法：

方案 2：一般 query 的总量是有限的，只是重复的次数比较多而已，可能对于所有的 query，一次性就可以加入到内存了。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，然后按出现次数做快速/堆/归并排序就可以了。

方案 3：

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（比如 MapReduce），最后再进行合并。

5、 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**？

可以估计每个文件安的大小为5G×64=320G，远远大于内存限制的4G。所以不可能将其完全加载到内存中处理。考虑采取分而治之的方法。

17 分而治之/hash 映射：遍历文件 a，对每个 url 求取 ，然后根据所取得的值将 url 分别存储到1000个小文件（记为 ）中。这样每个小文件的大约为300M。遍历文件 b，采取和 a 相同的方式将 url 分别存储到1000小文件中（记为 ）。这样处理后，所有可能相同的 url 都在对应的小文件（ ）中，不对应的小文件不可能有相同的 url。然后我们只要求出1000对小文件中相同的 url 即可。

18 hash 统计：求每对小文件中相同的 url 时，可以把其中一个小文件的 url 存储到 hash_set 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，存到文件里面就可以了。

OK，此第一种方法：分而治之/hash 映射 + hash 统计 + 堆/快速/归并排序，再看最后三

道题，如下：

8、怎么在海量数据中找出重复次数最多的一个？

方案1：先做 hash，然后求模映射为小文件，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的数据中重复次数最多的一个就是所求（具体参考前面的题）。

9、上千万或上亿数据（有重复），统计其中出现次数最多的钱 **N** 个数据。

方案1：上千万或上亿的数据，现在的机器的内存应该能存下。所以考虑采用 hash_map/搜索二叉树/红黑树等来进行统计次数。然后就是取出前 N 个出现次数最多的数据了，可以用第2题提到的堆机制完成。

10、一个文本文件，大约有一万行，每行一个词，要求统计出其中最频繁出现的前**10**个词，请给出思想，给出时间复杂度分析。

方案1：这题是考虑时间效率。用 trie 树统计每个词出现的次数，时间复杂度是 O(n*le)（le 表示单词的平准长度）。然后是找出出现最频繁的前10 个词，可以用堆来实现，前面的题中已经讲到了，时间复杂度是 O(n*lg10)。所以总的时间复杂度，是 O(n*le)与 O(n*lg10)中较大的哪一个。

接下来，咱们来看第二种方法，Bitmap。

密匙二：Bloom filter/Bitmap

关于什么是 Bloom filter，请参看此文：海量数据处理之 Bloom Filter 详解。

适用范围：可以用来实现数据字典，进行数据的判重，或者集合求交集

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 co
unter 数组代替位数组，就可以支持删除了。

还有一个比较重要的问题，如何根据输入元素个数 n，确定位数组 m 的大小及 hash 函数个数。当 hash 函数个数 k=(ln2)*(m/n)时错误率最小。在错误率不大于 E 的情况下，m 至少要等于 n*lg(1/E)才能表示任意 n 个元素的集合。但 m 还应该更大些，因为还要保证 bit 数组里至少一半为0，则 m 应该>=nlg(1/E)*lge 大概就是 nlg(1/E)1.44倍(lg 表示以2为底的对数)。

举个例子我们假设错误率为0.01，则此时 m 应大概是 n 的13倍。这样 k 大概是8个。

注意这里 m 与 n 的单位不同，m 是 bit 为单位，而 n 则是以元素个数为单位(准确的说是不同元素的个数)。通常单个元素的长度都是有很多 bit 的。所以使用 bloom filter 内存上通常都是节省的。

扩展：

Bloom filter 将集合中的元素映射到位数组中，用 k（k 为哈希函数个数）个映射位是否

全1表示元素在不在这个集合中。Counting bloom filter（CBF）将位数组中的每一位扩展为一个 counter，从而支持了元素的删除操作。Spectral Bloom Filter（SBF）将其与集合元素的出现次数关联。SBF 采用 counter 中的最小值来近似表示元素的出现频率。

问题实例：给你 A,B 两个文件，各存放50亿条 URL，每条 URL 占用64字节，内存限制是4G，让你找出 A,B 文件共同的 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）。

至于什么是Bitmap，请看此文：http://blog.csdn.net/v_july_v/article/details/6685962。下面关于 Bitmap 的应用，直接上题，如下第6、7道：

6、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。

7、腾讯面试题：给**40**亿个不重复的 **unsigned int** 的整数，没排过序的，然后再给一个数，如何快速判断这个数是否在那**40**亿个数当中？

方案1：oo，申请512M 的内存，一个 bit 位代表一个 unsigned int 值。读入40亿个数，设置相应的 bit 位，读入要查询的数，查看相应 bit 位是否为1，为1表示存在，为0表示不存在。

密匙三、Trie 树/数据库/倒排索引

Tire 树

适用范围：数据量大，重复多，但是数据种类小可以放入内存

基本原理及要点：实现方式，节点孩子的表示方式

扩展：压缩实现。

问题实例：

1).有10个文件，每个文件1G，每个文件的每一行都存放的是用户的 query，每个文件的 query 都可能重复。要你按照 query 的频度排序。

2).1000万字符串，其中有些是相同的(重复),需要把重复的全部去掉，保留没有重复的字

符串。请问怎么设计和实现？

3).寻找热门查询：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，每个不超过255字节。

更多有关 Trie 树的介绍，请参见此文：从 Trie 树（字典树）谈到后缀树。

数据库索引

适用范围：大数据量的增删改查

基本原理及要点：利用数据的设计实现方法，对海量数据的增删改查进行处理。

倒排索引(Inverted index)

适用范围：搜索引擎，关键字查询

基本原理及要点：为何叫倒排索引？一种索引方法，被用来存储在全文搜索下某个单词在一个文档或者一组文档中的存储位置的映射。

以英文为例，下面是要被索引的文本：

T0 = "it is what it is"

T1 = "what is it"

T2 = "it is a banana"

我们就能得到下面的反向文件索引：

"a": {2}

"banana": {2}

"is": {0, 1, 2}

"it": {0, 1, 2}

"what": {0, 1}

检索的条件"what","is"和"it"将对应集合的交集。

正向索引开发出来用来存储每个文档的单词的列表。正向索引的查询往往满足每个文档有序频繁的全文查询和每个单词在校验文档中的验证这样的查询。在正向索引中，文档占据了中心的位置，每个文档指向了一个它所包含的索引项的序列。也就是说文档指向了它包含的那些单词，而反向索引则是单词指向了包含它的文档，很容易看到这个反向的关系。

扩展：

问题实例：文档检索系统，查询那些文件包含了某单词，比如常见的学术论文的关键字搜索。

关于倒排索引的应用，更多请参见：第二十三、四章：杨氏矩阵查找，倒排索引关键词 Hash 不重复编码实践，及第二十六章：基于给定的文档生成倒排索引的编码与实践。

密匙四、外排序

适用范围：大数据的排序，去重

基本原理及要点：外排序的归并方法，置换选择败者树原理，最优归并树

扩展：

问题实例：

1).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个字节，内存限制大小是1M。返回频数最高的100个词。

这个数据具有很明显的特点，词的大小为16个字节，但是内存只有1m 做 hash 有些不

够，所以可以用外排序来排序。内存可以当输入缓冲区使用。
关于多路归并算法及外排序的具体应用场景，请参见此文：第十章、如何给10^7个数据量的磁盘文件排序。

密匙五、分布式处理 Mapreduce

适用范围：数据量大，但是数据种类小可以放入内存
基本原理及要点：将数据交给不同的机器去处理，数据划分，结果归约。
扩展：
问题实例：
1).The canonical example application of MapReduce is a process to count the appearances of each different word in a set of documents:
2).海量数据分布在100台电脑中，想个办法高效统计出这批数据的 TOP10。
3).一共有 N 个机器，每个机器上有 N 个数。每个机器最多存 O(N)个数并对它们操作。如何找到 N^2个数的中数(median)？

更多具体阐述请参见：从 Hadhoop 框架与 MapReduce 模式中谈海量数据处理，及 MapReduce 技术的初步了解与学习。

参考文献

1 十道海量数据处理面试题与十个方法大总结；
2 海量数据处理面试题集锦与 Bit-map 详解；
3 十一、从头到尾彻底解析 Hash 表算法；
4 海量数据处理之 Bloom Filter 详解；
5 从 Trie 树（字典树）谈到后缀树；
6 第三章、寻找最小的 k 个数；
7 第十章、如何给10^7个数据量的磁盘文件排序；
8 第二十三、四章：杨氏矩阵查找，倒排索引关键词 Hash 不重复编码实践；
9 第二十六章：基于给定的文档生成倒排索引的编码与实践；
10 从 Hadhoop 框架与 MapReduce 模式中谈海量数据处理。