# 1网站应用题

神书解决一切（大型网站）

包含网站，手机，各种需要有人参与，后台控制的场景。

* 分层（应用层--服务层--数据层）
* 主要手段：
* 缓存：程序与缓存部署在同一台服务器上；分布式缓存（集中管理）
* 集群
* 异步：消息队列（生产者消费者模式）+ 异步推拉

微博：用户发表微博后系统将微博写入消息队列后立即返回，用户响应迅速，消息队列消费者任务将微博推送给所有当前在线粉丝的订阅列表中，非在线用户登录后再根据关注列表拉取微博订阅列表

“刷”微博-->大部分是缓存访问操作

* 分布式服务部署：

1. 在集群中的多台服务器上，每台都部署多个服务，每个服务使用不同的端口对外提供服务；
2. 将物理机虚拟化成多个虚拟机后，在虚拟机上部署应用1的方法。

* 高可用（失效转移）

失效确认（心跳检测）、访问转移（重新路由）、数据恢复（保证副本数）

* 伸缩性（集群添加新主机） 需要负载均衡-->一致性hash算法
* 扩展性（系统增加新功能）

模块化，低耦合（使用分布式消息队列实现低耦合）

# 2 场景应用题

消费者，算法场景（如酒店匹配案例），等等。

平时多看多积累，尤其注意以下几点：

（1）线程并发

1. 大数据，集群，负载
2. 消息推送，发布-订阅模式

# 3海量数据处理

常见的方法有Hash法，位图法，Bloom-filter法、数据库优化法、倒排索引法、外排序法、Trie树、堆、双层桶法以及MapReduce法。

分而治之/hash映射+hash统计+堆/快速/归并排序（先映射，然后统计，最后排序）

双层桶排序（求第K大，中位数，不重复或重复的数字）：通过多次划分，逐步确定范围，最后在一个可以接受的范围内进行

Bloom filter（集合求交集、数据判重）/BitMap

Trie树/数据库/倒排索引

外排序

分布式处理之Hadoop/MapReduce

## TopK问题（先映射，然后统计，最后排序）

* 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。

3百万\*255B=0.75G <1G 可以将所有串放在内存中进行

* Hashmap统计：先对数据预处理，维护一个key为query串，value为该query串出现次数的hashmap，若该query串在map中，那么将该query串的计数加一即可，若该query串不在map中，那么加入该query串，并将value设为1
* 堆排序：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。
* 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。

(1G/16B):(1M/16B)=2000 可取5000

* 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。
* Hashmap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。
* 堆排序/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）后，再把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（类似于归并排序）的过程了。
* 提取某日访问网站次数最多的那个IP
* 分而治之/hash映射：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个IP。同样可以采用映射的方法，比如%1000，把整个大文件映射为1000个小文件
* Hashmap统计：采用hash\_map对那1000个文件中的所有IP进行频率统计
* 堆/快速排序：进行排序(可采取堆排序)，得到次数最多的IP。
* 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。
* 如果每个数据元素只出现一次，而且只出现在某一台机器中，那么可以采取以下步骤统计出现次数TOP10的数据元素：
* 堆排序：：在每台电脑上求出TOP10，可以采用包含10个元素的堆完成，求出每台电脑上的TOP10后，然后把这100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就可以了。
* 如果同一个元素重复出现在不同的电脑中：遍历一遍所有数据，重新hash取摸，如此使得同一个元素只出现在单独的一台电脑中，然后采用上面所说的方法，统计每台电脑中各个元素的出现次数找出TOP10，继而组合100台电脑上的TOP10，找出最终的TOP10

## 重复问题

### 内存够--使用位图法

* 10亿个正整数，只有一个数重复出现过，要求在O(n)时间里找出这个数

每个数对应1bit 10亿/1MB=120 所以使用120MB内存即可

* 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

最多有99M个数字 99999999

一个数对应1bit 99M/8B = 12.375 所以使用12.375MB内存即可

* 给40亿个不重复额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？

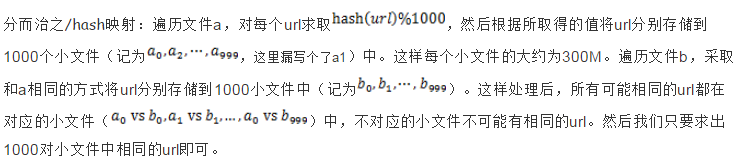
一个数对应1bit 40亿/1MB = 512 所以使用512MB内存即可

* 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

整数个数为2^32,也就是，我们可以将这2^32个数，划分为2^8个区域(比如用单个文件代表一个区域)，然后将数据分离到不同的区域，然后不同的区域在利用bitmap就可以直接解决了。也就是说只要有足够的磁盘空间，就可以很方便的解决。

### 内存不够--分而治之/hash映射之后再处理

* 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？
* 分而治之/hash映射：



* hash\_set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。

## 排序问题

### 映射之后排序

* 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。
* 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a0,a1,..a9）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）。
* hash\_map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。
* 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b0,b1,...b10）。最后，对这10个文件进行归并排序（内排序与外排序相结合）。

### 位图法

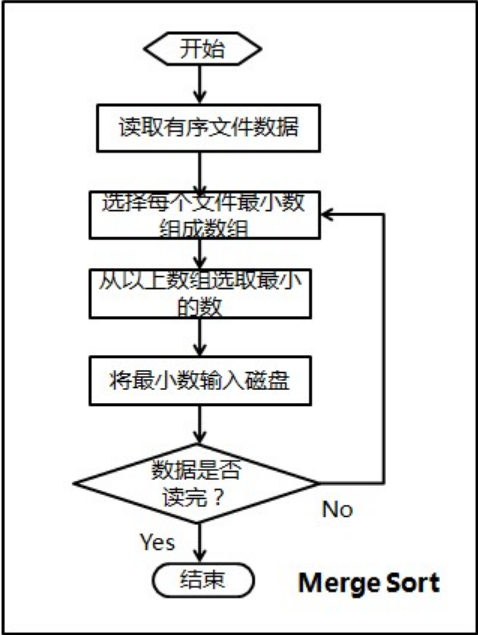
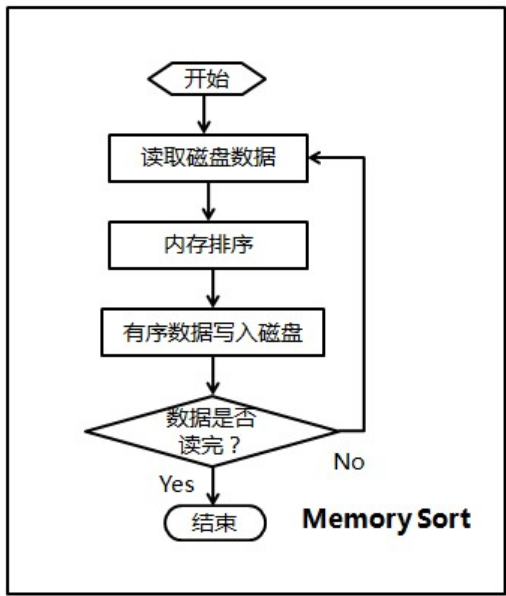
* 一个文件中有9亿条不重复的9位整数，对这个文件中的数字进行排序

声明一个可以包含9位整数的bit数组，10亿/1MB = 120 一共需要120MB内存

### 外排序

一般使用归并排序方式实现外排序

假设文件中整数个数为N(N是亿级的)，整数之间用空格分开。首先分多次从该文件中读取M（十万级）个整数，每次将M个整数在内存中使用快速排序之后存入临时文件，然后使用多路归并将各个临时文件中的数据再次整体排好序后存入输出文件。显然，该排序算法需要对每个整数做2次磁盘读和2次磁盘写。以下是本程序的流程图：



* 如何给10^7个数据量的磁盘文件排序

1、内存排序

由于要求的可用内存为1MB，那么每次可以在内存中对250K的数据进行排序，然后将有序的数写入硬盘。

那么10M的数据需要循环40次，最终产生40个有序的文件。

2、归并排序

将每个文件最开始的数读入(由于有序，所以为该文件最小数)，存放在一个大小为40的first\_data数组中；

选择first\_data数组中最小的数min\_data，及其对应的文件索引index；

将first\_data数组中最小的数写入文件result，然后更新数组first\_data(根据index读取该文件下一个数代替min\_data)；

判断是否所有数据都读取完毕，否则返回2。

## 中位数问题

一般使用双层桶（桶排序）法

* 5亿个int找它们的中位数

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，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中的那些数就可以了。

# 4一致性hash算法

