http://blog.csdn.net/v\_JULY\_v/article/details/6256463

**Hash方法**

**假设数组长度为16**

1) index = value % 16

2) index = (value \* value) >> 28 (int 32 bit minus 28 = 4. 2^4 = 16)

3) fibonacci hashing

对于32位整数而言, index = (value \* 2654435769) >> 28

散列结果普遍比较好.这与fibonacci数列和黄金分割有关

typedef unsigned short int HashIndexType;

HashIndexType hash(int key) {

//16bit:40503, 32:2654435769, 64:11400714819323198485

static const HashIndexType M = 40503;

//s = w - n; n : table size, w : input bit size

//if array size is 16, w = n, because 16 = 2^4,

//if bit is 16, table size is 2^10, then, S = 16 - 10 = 6

static const int S = 6;

return (HashIndexType)(M \* key) >> S;

}

//for normal int, size is 32, then, M = 2654435769, S = 32 - 4 = 28

//index = (value \* M) >> S

Hash冲突解决办法

1每一个slot挂链表，并保存key值 也可用多个hash值 一个用来寻址 另外几个用来校验

2 顺序往下移一个位置, 如果到表尾则移到表头

Hash表size通常取质数

如果发现不够用了 一般double size,并对已有的数据重新hash

字符串Hash

32bit整数无法对应所有字符串,但是仍然可以设计一些算法,使得input的string的hash值尽可能不相同

Multi hash 验算?

六把秘钥

秘钥一 分为治之/hash映射 + hash统计 + 堆/快速/归并排序

1. 分而治之/hash映射：针对数据太大，内存受限，只能是：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
2. hash\_map统计：当大文件转化了小文件，那么我们便可以采用常规的hash\_map(ip，value)来进行频率统计。
3. 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，再找出每个小文中出现频率最大的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）及相应的频率。然后再在这1000个最大的IP中，找出那个频率最大的IP，即为所求。”

(同一IP取模后肯定会存在同一个小文件里)

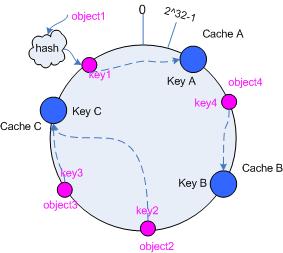
**一致性hash**

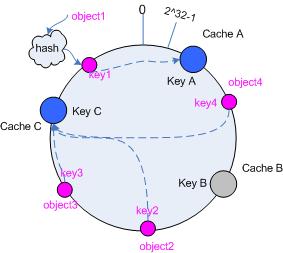
为了方便缓存数据. QQ有N个服务器,原来是ServerIPIndex[QQNum % n]. 现在有一台服务器死掉了

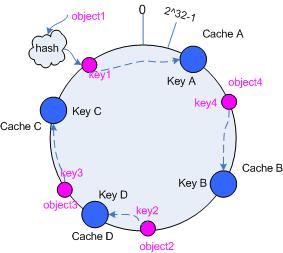
解决方案一 先%N 如果服务器是坏的 则尝试%(N-1)…

解决方案二 一致性Hash (Consistent Hashing)

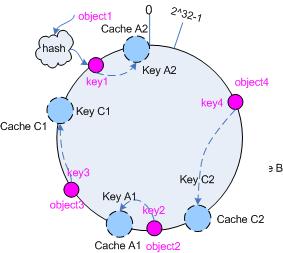
<http://blog.csdn.net/v_july_v/article/details/6879101>

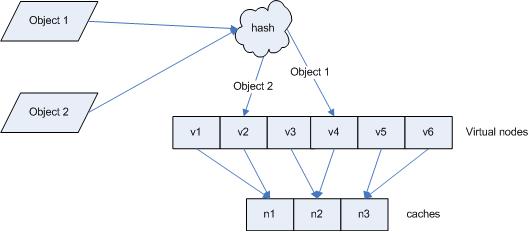






虚拟节点 (平衡负载)





虚拟节点的计算可以采用IP加后缀的方式 比如 Cache A(Server A) 地址为 202.168.14.241

之前hash值为 Hash(“202.168.14.241”) 32 bit

虚拟节点值可以为

Hash(“202.168.14.241#1”)

Hash(“202.168.14.241#2”)

秘钥二 分层次

适用范围: 第K大, 中位数,不重复 重复数字

5亿整数找不重复的个数

划分2^8个区域 每个区域用bitmap解决

找中位数

划分2^16个区域, 扫描一遍统计各区域个数,可以得到中位数落在哪个区

第二次扫表落在这个区域的数 确定中位数

**秘钥三: Bloom filter/Bitmap**

Bloom Filter

<http://blog.csdn.net/v_july_v/article/details/6685894>

位数组+k个独立hash函数, 查找value时, k个hash分别查找,看hashk[value] 是不是为1, 若为1则value在set里

3、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。**”**

秘钥四: Trie树/数据库/倒排索引

秘钥五: 外排序

秘钥六: 分布式处理(Map reduce)

Hadoop = HDFS + HBase + MapReduce