Bloom Filter Technical Report

NDD@newsmth.net

2008 lunar new year

Ver. 0.0.1

Introduction

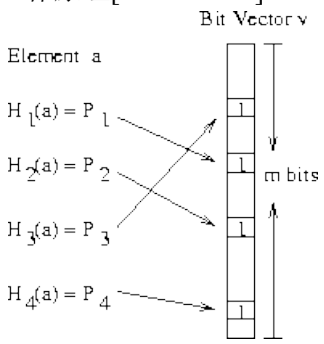
Hash Table 是以某个 Hash Function 实现 Key：Value 对查和存储的高速数据结构。不过 Hash Table 也有缺点，这就是它的内存利用效率大约只有 50%[GOOGLE BLOG]。当存储大量的拥有长 Key 值的数据集合的时候，这个问题将十分突出。

例如我们打算存储全球的垃圾邮件地址，约摸有几十亿个吧[GOOGLE BLOG]，加入按照标准 Hash Table 直接都存下来仅仅 Key 就要消耗数百个 G 的内存，这实在大的有点离谱。好在前人 Burton Bloom 先生在 1970 年[Bloom Filter Wiki]就设计出了一个大为节约内存，而且具有很高可靠性的数据结构——Bloom Filter 来对付这类问题。

Concept

Bloom Filter 是通过一组 k 个定义在 n 个输入 key 上的 Hash Function，将上述 n 个 key 映射到 m 位上的数据容器。它提供将新 key 插入的函数 VOID Insert(key)，以及查询 key 是否在集合中的函数 BOOL Query(key)。这里 BOOL Query(key)的返回值以概率 $\left(1-e^{kn/m}\right)^k$ [WISC EDU]可能犯以假当真错误，除此以外，Query 不会出以真当假的错误。

下图画出了 Bloom Filter 的工作原理[WISC EDU]：



在 Insert 时，输入 A 给定之后，k 个互相独立的 Hash Function 将它们映射到 m 位中的 k 个位上，并且将这位置 1。如此，Bloom Filter 就记住了元素 A 的“案底”，可备以后查询用。

在 Query 时，输入 A 给定之后，我们一样得到 k 个位。当它们都为 1 时，就认为元素 A 是之前 Insert 的某个成员。

当然这可能会犯以假当真的错误，其概率是 $\left(1-e^{kn/m}\right)^k$。显然我们不会犯以真当假的错误，原因大家想一想就知道了。

Theory

上面的这个“假阳性”概率是怎么计算出的呢？如下：
由于 k 个 hash function 互相独立，n 个输入互相独立，那么 m 次插入之后，m 位中有某位为 0 的概率是 $\left(1-\frac{1}{m}\right)^{kn}$。

发生“假阳性”错误时，要求 k 个 hash function 都不遇到 0，这就是概率 $\left(1-\left(1-\frac{1}{m}\right)^{kn}\right)^{k}$。

大约是 $\left(1-e^{kn/m}\right)^{k}$。显然 n 与 m 越接近，这错误率越大。反之则越小。

再对上式求偏导数，得到其最小值时有 $k=\frac{m}{n}\ln 2$。

Tuning

由于内存的大小与误识率的指标通常是确定的，我们可以根据上述公式，确定 k 的最优大小。在[Bloom Filter Calculator]这里有一个计算器可以帮助我们计算出来。例如，我们假设某人有 1 亿个邮件地址加进了黑名单，她打算出 16 亿 bit 内存来存储它们，并且要求错误率为 0.001.则我们需要 6 个不同的 Hash Function 就够了。

Hash Function

此类函数是将 Key 映射为数字的函数。对字符串来说，常见的 Hash Function 虽然也有 10 来个，但是其实只是 2 大类。分别是累加型与异或移位型。其他的都是它们的变体或者混合。累加型的基本形式为：h(n+1) = P*Key[n+1] + h(n)其中 P 为质数。例如 stlport 中的默认 string hash 就是 h += 5*key[n]。异或移位型的基本形式为：h(n+1) = (Key[n+1] <</>> P) XOR h(n); 这里 P 还是质数。VC++ 8.0 带的 STL 里面的 string hash 基本就是这类。当然，好的 Hash Function 不仅要求速度快，还要求分布均匀，所以往往经过许多实验，带有一些神秘的数字。但是——它们都是质数，神奇的种族……[General Purpose Hash Function Algorithms]

如果你找不到 k 个独立的 Hash Function 给你的 Bloom Filter 使用，你可以造假：先找一个 Hash Function 比如 h += 5*key[n]，然后在后面加几个常数得到其它的 Hash Function。不过代价就是错误率的提高:-P[WISC EDU]

Misc

Bloom Filter 是一个有趣的数据结构，我觉得其实它是一个特殊的退化 Hash Table。退化到不处理 Collision，不存储 Key 值。你看，如过 Hash Table 假如不存储 Key，其实不就是一个 k==1 的 Bloom Filter 而已嘛。

Reference

[GOOGLE BLOG]数学之美系列二十一,布隆过滤器（Bloom Filter）

http://www.googlechinablog.com/2007/07/bloom-filter.html

[Bloom Filter Wiki] Bloom Filter Wiki in Answers.com

http://www.answers.com/bloom%20filter

[WISC EDU]Bloom Filters - the math

http://pages.cs.wisc.edu/~cao/papers/summary-cache/node8.html

[Bloom Filter Calculator]

http://www.cc.gatech.edu/~manolios/bloom-filters/calculator.html

[General Purpose Hash Function Algorithms]

http://www.partow.net/programming/hashfunctions/index.html#AvailableHashFunctions