两种对 URL 的散列效果很好的函数*

李晓明+, 凤旺森

(北京大学 计算机科学技术系,北京 100871)

Two Effective Functions on Hashing URL

LI Xiao-Ming+, FENG Wang-Sen

(Department of Computer Science and Technology, Peking University, Beijing 100871, China)

+ Corresponding author: Phn: +86-10-62756589, Fax: +86-10-62765813, E-mail: lxm@pku.edu.cn, http://netpku.edu.cn

Received 2003-03-05; Accepted 2003-06-18

Li XM, **Feng WS**. **Two effective functions on hashing URL**. ***Journal of Software***, **2004**,**15(2):179~184**.
http://www.jos.org.cn/1000-9825/15/179.htm

Abstract: Hashing large collection of URLs is an inevitable problem in many Web research activities. Through a large scale experiment, three hash functions are compared in this paper. Two metrics were developed for the comparison, which are related to web structure analysis and Web crawling, respectively. The finding is that the well-known function for hashing sequence of symbols, ELFhash, is not very good in this regard, and the other two functions are better and thus recommended.

Key words: hashing; ELFhash; URL; even distribution; Web mining; load balance

摘 要: 在Web信息处理的研究中,不少情况下需要对很大的URL序列进行散列操作.针对两种典型的应用场合,即Web结构分析中的信息查询和并行搜索引擎中的负载平衡,基于一个含有2 000多万个URL的序列,进行了大规模的实验评测.说明在许多文献中推荐的对字符串散列效果很好的ELFhash函数对URL的散列效果并不好,同时推荐了两种对URL散列效果很好的函数.

关键词: 散列;ELFhash;URL;均匀分布;Web挖掘;负载平衡

中图法分类号: TP314 **文献标识码**: A

散列(hashing)是信息存储和查询所用的一项基本技术,许多教科书中都有介绍[1].这项技术虽然发明于50年前,其间也有过不少非常系统、完整的研究工作[2],但直到近几年仍然有新的工作进展[3~5].散列技术的基础性及其在许多领域的可应用性是其不断被人们研究的源泉,而对它进行的研究能不断有新意的基本原因在于,其优化性能在很大程度上取决于输入键的结构.本文研究散列的键是用于访问网页的url,即形如“http://net.cs.pku.edu.cn/…”之类的字符串[6].

考虑如下两种应用情形:

* Supported by the National Grand Fundamental Research 973 Program of China under Grant No.G1999032706 (国家重点基础研究发展规划(973))

作者简介: **李晓明**(1957－),男,湖北荆州人,教授,博士生导师,主要研究领域为计算机系统,网络计算;**凤旺森**(1980－),男,硕士生,主要研究领域为算法设计,复杂性理论.

(1) 假设我们已经有了全国网页的集合,记作 $P=\{p_1,p_2,p_3,...\}$.对应地,所有 URL 的集合,记作 $URL=\{url_1,url_2,url_3,...\}$.作为研究 Web 结构的一个方面,我们关心的是,给定两个网页 p_i 和 p_j,从 p_i 到 p_j 的最少链接数是多少.为此,我们可以首先从网页中抽取出基本结构信息,将每一个网页表达为

$$p_i=\{url_i:url_{i,1},url_{i,2},...,url_{i,j},...\},$$

即我们将每篇网页看成是一个有向图的节点,这样表达的 p_i 就给出了节点的标识和它的"外向"相邻关系.注意,并不是所有的 $url_{i,j}$ 都属于上述 URL 集合(可能有死链,还可能有我们在此不关心的链,例如指向国外的).此时,得到从 p_i 到 p_j 的最少链接数的方法之一就是从 p_i 开始做先宽搜索.在搜索过程中,为了不重复劳动,要记住已经查过的节点(记作集合 visited-nodes);凡是遇到一个节点,就要用它和 visited-nodes 中的元素对比,若在其中,则放弃,否则,将它放入 visited-nodes,继续沿其含有的 url 向下搜索.显然,将 visited-nodes 组织成一个散列表是最自然的.散列的对象(键)是 url.

(2) 在并行搜索引擎的工作过程中,若干台计算机并行地从 Web 上抓取网页[7].并行系统维护一个全局的 visited-pages,每台计算机维护一个局部的 unvisited-pages.每当从一个刚抓来的网页中提取出一个 url 时,首先和 visited-pages 的元素对比(这里可能也用散列,与前述情况类似,在此不赘述),看是否存在;如果不存在,则将它散列到某台计算机上,让它来负责该 url 对应网页的抓取.

我们注意到,这两种应用的要求是有区别的.前者追求的是查询效率,后者追求的是负载平衡;前者的散列槽数可能很大(对于千万量级的节点数,槽数可以大到百万量级),后者的散列槽数相比要小得多(并行系统中计算机台数).同时,前者追求的是某种"平均"效果,后者要考虑"最大值"的情况.这样的认识是后面进行实验设计和分析的基础.

首先,我们针对上述两种应用需求给出相应的散列函数质量的评估测度.随后,描述参与实验的数据源和几个散列函数;特别地,其中包括在许多文献中推荐的对字符串散列效果很好的 ELFhash 函数(一种散列 URL 的自然选择)和阎宏飞、谢正茂设计的用于我们的天网搜索引擎的散列函数,以及我们在"Web 信息博物馆"中使用的另一种函数.在介绍了实验设计的细节之后,我们给出执行的结果.最后是对结果的几点分析和讨论.

1 实验的设计和执行

1.1 评估测度

为了比较不同的散列函数,需要有与应用背景相关的评估测度.对于上述第 1 种应用,我们关心的是对散列后 url 的平均查询时间.对于第 2 种应用,我们关心最大的槽中元素的个数.为此,有下面两种对应的测度定义.

信息查询性能测度.设有 N 个 url 字符串待处理,将这些字符串用散列函数 F 散列到 M 个槽中,对 $i(1\le i\le M)$ 号槽,散列到其中的 url 个数为 N_i.我们假设查询每个 url 的概率是相等的,则对 N 个 url 各作一次查询的时间算术平均值,便可当作 F 的查询性能测度.按照通常散列表项的链表结构,对 i 号槽中第 k 个元素作查询,需要访问存储 k 次,所以,对散列在 i 号槽中的所有字符串都作一次查询,则需要访问存储 $\frac{N_i(N_i+1)}{2}$ 次.考虑所有的槽,我们可以用下面的平均存储访问次数来表示散列函数 F 的查询性能测度,越小越好.

$$A=\frac{1}{N}\sum_{i=1}^{M}\frac{N_i(N_i+1)}{2}=\frac{1}{2}+\frac{1}{2}\sum_{i=1}^{M}\frac{N_i^2}{N},\ \text{其中}\ \sum_{i=1}^{M}N_i=N.$$

根据上述条件和凹函数(x^2)的基本性质易知,当所有 $N_i=\frac{N}{M}$ 时 A 取最小值,当只有 1 个 $N_i=N$,其他 N_i 都为 0 时取最大值.两个极值分别为 $\frac{N/M+1}{2}$ 和 $\frac{N+1}{2}$.

负载平衡性能测度.设有 N 个 url 所指网页待抓取,将这些 url 字符串用散列函数 F 散列到 M 台机器中,对 $i(1\le i\le M)$号机器,散列到其中的 url 字符串个数为 N_i,则 M 台机器并行完成 N 个网页抓取的时间取决于任务最多

的机器的完成时间,即与 $\max(N_i)$成比例.做一定的规格化,我们用 $B=\frac{M}{N}\max(N_i)$ 来作为 F 的负载平衡性能测度,这个量也可以解释成抓取一个网页所要消耗的平均机器时间,越小越好.同样,我们不难看出它的极值分别为 1 和 M,发生在 $\max(N_i)=\frac{N}{M}$ 和 $\max(N_i)=N$.我们注意到,B 值的物理意义是散列结果和最佳期望值的倍数差距,即 $\max(N_i)=B\cdot\frac{N}{M}$.

1.2 待评估的散列函数

我们考虑 3 个函数,第 1 个取自文献[8],源于 UNIX System V,号称“实际生活散列函数中的典型魔术”,也是我们在信息查询研究工作中的首选,但发现不理想,从而导致了本文的研究工作.

```
int ELFhash(char* url, int size)
{
    unsigned int h=0;
    while (*url)
    {
        h=(h<<4) + *url++;
        unsigned int g= h & 0xF0000000;
        if (g)
            h^=g>>24;
        h&=~g;
    }
    return h%size;
}
```

第 2 个是由阎宏飞和谢正茂采用折叠方法设计的,一直用在天网搜索引擎中,感觉负载平衡效果比较好,但没有系统评估过.

```
unsigned int HfIp(char* url, int size)
{
    unsigned int n=0;
    char* b=(char*)&n;
    for (int i=0; i<strlen(url); i++)
        b[i%4]^=url[i];
    return n%size;
}
```

第 3 个是谢正茂为了在多个数据库中分布网页设计的,也是出于负载平衡的目的,主要的一个考虑因素是实现简单.

```
int _hf(char* url, int size)
{
    int result=0;
    char* ptr=url;
    int c;
    for (int i=1; c=*ptr++; i++)
        result += c*3*i;
    if (result<0)
        result = –result;
```

```
    return result%size;
}
```

1.3 实验方案和数据结果

我们以天网搜索引擎搜集的 2 000 万个 url 序列为源数据*,用上述 3 种散列函数分别针对两种应用的需求实验,考察在不同的 N 和 M 情况下测度 A 和 B 的情况.

对信息查询应用,我们分别取槽数 M=20,80,100,200(单位:万),以 50 万为间隔,以 N=50 万,100 万,…,2000 万个 url 为输入执行散列函数,得到测度 A.共执行 4 次,每次执行得到 3×40 个数据.

对负载平衡应用,我们分别取槽数 M=8,10,12,40,同样也是以 50 万为间隔,以 N=50 万,100 万,…,2000 万个 url 为输入执行散列函数,得到测度 B.也是共执行 4 次,每次执行得到 3×40 个数据.

图 1 和图 2 是这些数据的图示.如图 1 所示为信息查询的性能,如图 2 所示为负载平衡的性能,其中标有 A 的图与测度 A 相关,标有 B 的图与测度 B 相关.

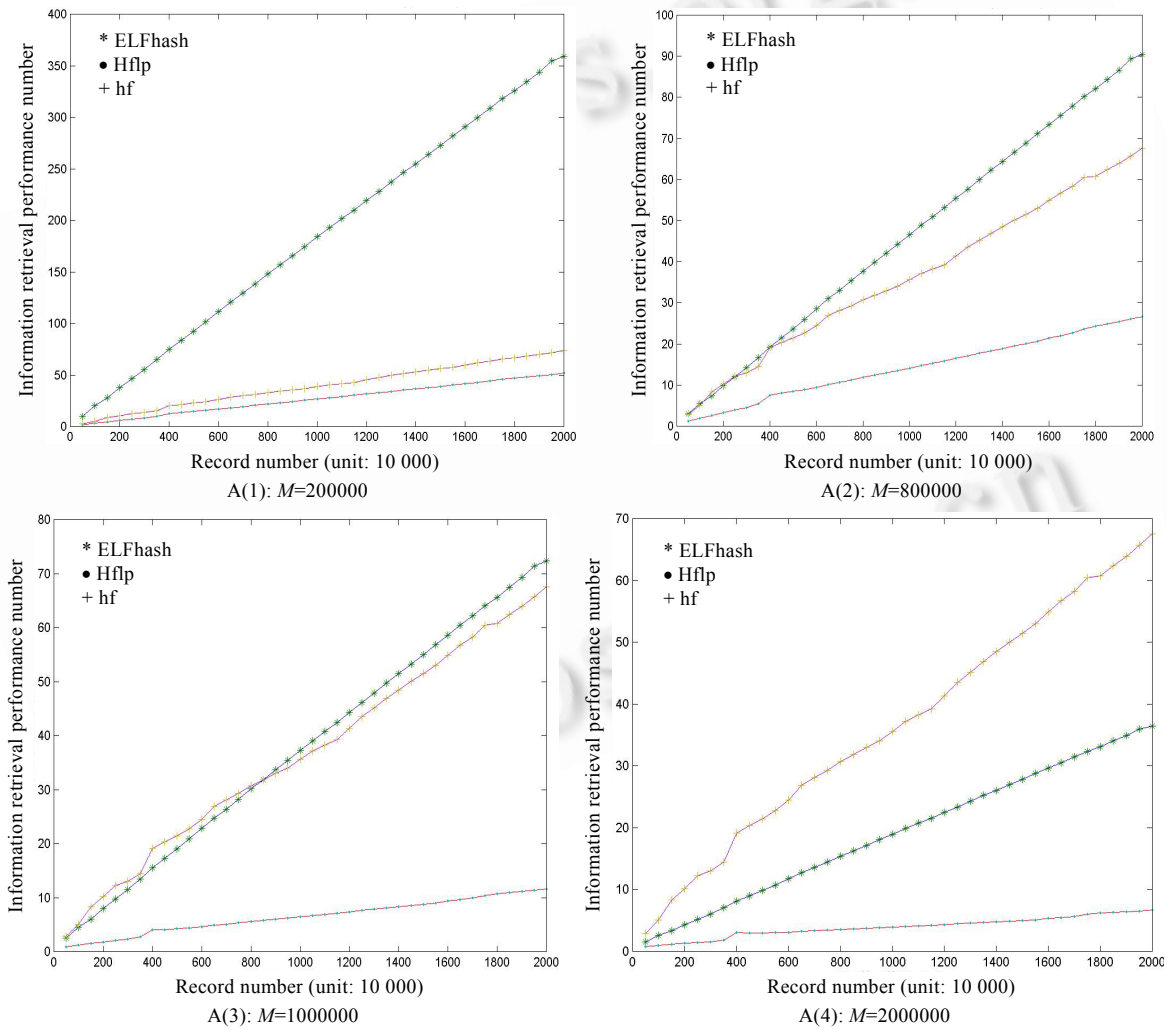

Fig.1 Performance comparison for information retrieval
图 1 检索性能比较

* 可以免费提供给有兴趣做类似研究工作的同行,燕穹产品号:YQ-URLLIST-2002-03.

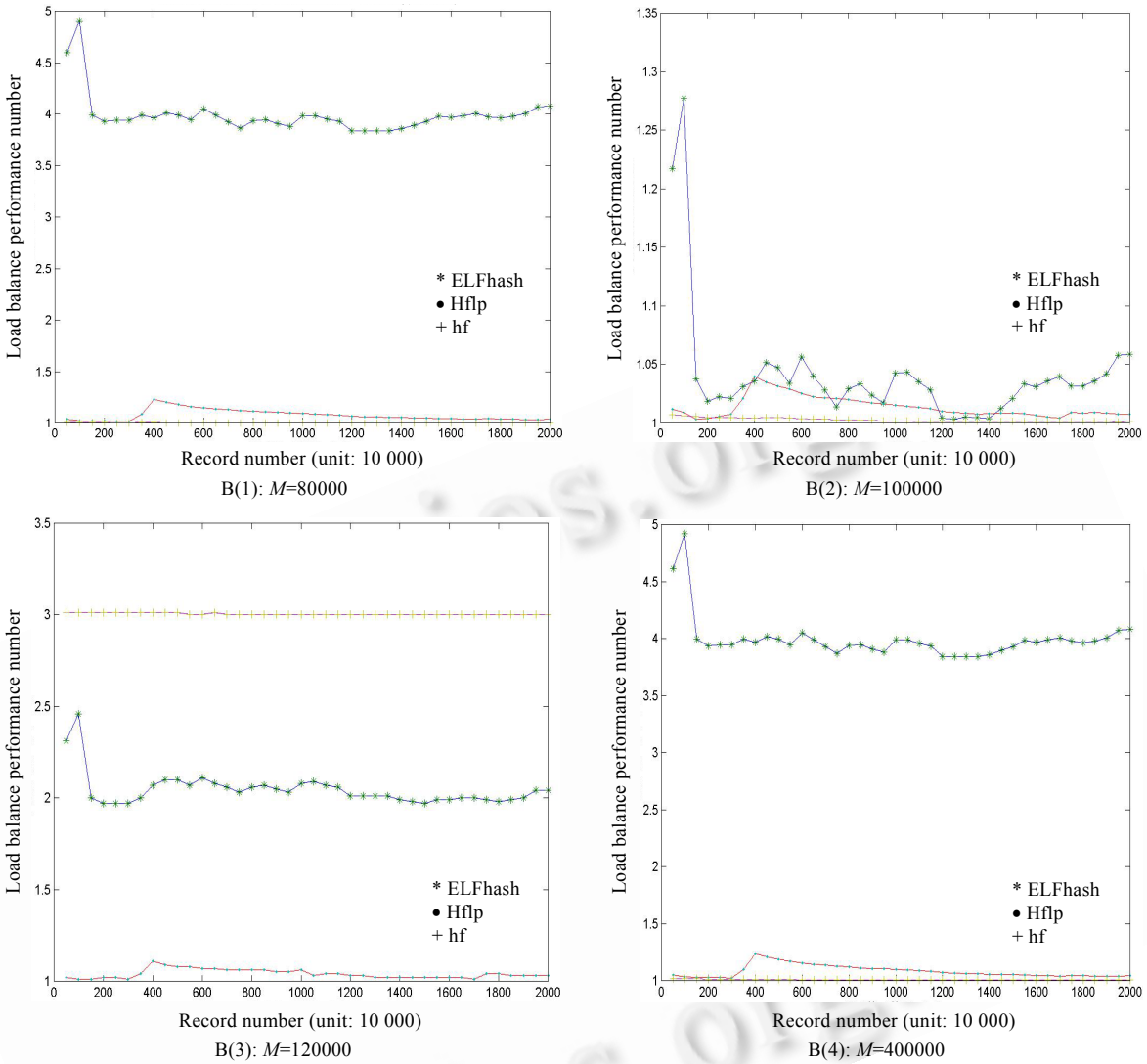

Fig.2 Performance comparison for load balance

图 2 负载平衡性能比较

2 实验的结论与分析

我们分信息查询(对应于 A)和负载平衡(对应于 B)两种情况加以讨论.

观察图 1 中 A(1),A(2),…,A(4)的形态,我们有:

- HfIp 散列函数具有绝对的优势,在所有情况下都是最好的.
- hf 的性能在 M 较小的情况下与 HfIp 相当,比 ELFhash 要好;但随着 M 的增大,hf 的相对性能逐步下降,最后比 ELFhash 要差.
- 特别地,我们注意到 HfIp 和最优值相当接近.例如,在 M=20 万的情形下,我们有下面的对比(其中最优 A 值可由前面的测度表达式定义计算出):

N	500 000	1 000 000	1 500 000	2 000 000	2 500 000	3 000 000	…	20 000 000
A(opt)	1.75	2.75	4.25	5.5	6.75	8.00	…	50.50
Hflp	1.89	3.23	4.54	5.85	7.01	8.31	…	51.75

这些数据显示 HfIp 不仅很接近最优,而且相当稳定.

观察图 2 中 B(1),B(2),…,B(4)的形态,我们有:

- HfIp 散列函数在负载平衡方面表现也不错.它的 B 测度基本上都在 1.1 以下,接近 B 的最优值(即 1.0,与图的横轴重合).它最大的优点是稳定,对不同的 M 表现很一致.
- hf 在 M=8,10,40 时表现最好(比 HfIp 要好),但在 M=12 时最差(比 ELFhash 要差).这种不稳定性的原因不清楚.
- ELFhash 的表现较差,B 值在 1~5 之间大范围波动.

由于 hf 在负载平衡实验中的异常表现,我们在 2 000 万个 url 序列中随机选取 40 万个 url 作为样本,单独作一个比较实验,证明 hf 的不稳定性.对负载平衡应用,我们分别取槽数 M=11,12,14,16,18,以 1 万为间隔,以 N=1 万,2 万,…,40 万个 url 为输入执行散列函数 hf,得到测度 B.共执行 5 次,每次执行得到 40 个数据.图 3 是这些数据的图示,可以很清楚地看到 hf 性能的波动.

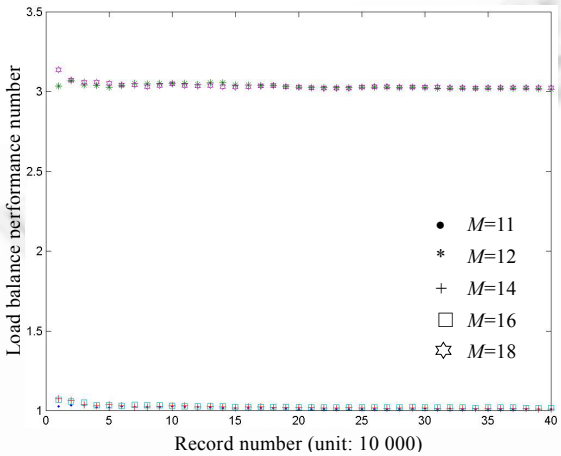

Fig.3 Instability of algorithm hf in load balance application

图 3 hf 算法在负载平衡应用中的不稳定性

作为本项研究的结论,我们有:对于 URL 的散列,(1) 无论出于信息查询还是负载平衡的需求,HfIp 都是很可靠的,可以首选使用;(2) Hf 在有些情况下对负载平衡很好,但它对槽数的敏感性是一个值得注意的缺陷,有时可能效果很不好,需要小心;(3) 不推荐使用 ELFhash.

致谢 许多人在这项工作中给予了帮助:阎宏飞和谢正茂提供了他们的散列函数,孙磊收集并整理了用于实验的数据,肖明忠对论文的初稿提出了一些有益的建议,张立昂老师一直关心这项工作的进行.在此,我们对他们表示衷心的感谢.

References:

[1] Cormen TH, Leiserson CE. Introduction to Algorithms. 2nd ed., Cambridge: MIT Press, 2001. 221~252.
[2] Knuth DE. Sorting and Searching, Volume 3 of the Art of Computer Programming. New York: Addison-Wesley, 1973. 506~549.
[3] McKenzie BJ, Harries R, Bell T. Selecting a hashing algorithm. Software Practice and Experience, 1990,20(2):208~210.
[4] Tong MCF. General hashing [Ph.D. Thesis]. Computer Science Department, University of Auckland, 1996.
[5] Peter K. Pearson, fast hashing of variable length text strings. Communications of the ACM, 1990,33(6):676~678.
[6] Berners-Lee T. Universal resource locator. 2003. http://www.w3.org/Addressing/URL/Overview.html
[7] Yan HF, Wang JY, Li XM, Guo L. Architectural design and evaluation of an efficient Web-crawling system. Journal of System and Software, 2002,60(3):185~193.
[8] Shaffer CA. Zhang M, Liu XD, Trans. Data Structure and Algorithm Analysis. Beijing: Publishing House of Electronics Industry, 1998. 211~213 (in Chinese).

附中文参考文献:

[8] Shaffer CA,著.张铭,刘晓丹,译.数据结构与算法分析.北京:电子工业出版社,1998.211~213.