汉字二级存储系统及其LFU 算法分析

吴克西

(厦门大学计算机与系统科学系，361005)

TWO–LEVEL CHINESE CHARACTER STORE SYSTEM AND LFU POLICY

Wu Kexi

(Department of Computer and System Science, Xiamen University, 361005)

ABSTRACT

The chinese character store system is described in this paper. Compressed chinese characters are stored in two-level memory and managed by C–tree^{TM} keyed file access routines. LFU policy is used when character-fault. An experiment, in which chinese character access sequence is accordant with normal distribution is assumed, is made. The preliminary experimental results are presented and discussed.

摘　要

本文给出一个采用LFU 策略的汉字二级存储系统，该系统采用汉字点阵压缩存储、C-tree^{TM} 管理汉字库，大大地加快了汉字的访问速度．本文还使用该系统进行汉字访问序列符合正态随机分布的实验，给出了初步的实验结果并对其进行了讨论．

§1. 引　　言

在中文电子排版系统中，汉字存取是影响系统响应速度的主要障碍之一．以往的经验表明汉字存取时间中百分之七十五花费在磁盘查找上，因此减少磁盘访问次数可以大大提高汉字访问速度．用内存存放汉字点阵是最直接简单的方法，但由于汉字字集非常庞大，系统所需的字形笔体繁多，点阵密度各异，内存容量不可能存放全体汉字点阵库．遇到这种情况一般的处理方法是将常用汉字存放在内存，较少使用的汉字放在磁盘上，以求平均访问时间最短．不幸的是常用汉字本身是动态变化的，它与所涉及的领域有很大关系．解决这一问题的途径就是利用存储体系的思想，将内存和磁盘外存组织成二级

1989 年10 月5 日收到．1990 年7 月12 日定稿．

存储结构，用动态置换代替静态常用汉字集的内存驻留，从而达到节省空间又不致造成很高缺字率的目的. 实现二级存储层次必须解决如下两个问题:

* 最多允许多少汉字存放在内存才能使系统有效;

* 发生缺字时采用什么策略进行置换.

本文根据汉字访问特性和中文电子排版系统的要求提出了采用LFU (Least frequently used) 置换策略的汉字二级存储系统，给出了汉字访问序列符合正态随机分布时内存汉字个数与缺字率的关系曲线.

§2. 汉字二级存储系统

2.1 汉字压缩存储

中文电子排版需要使用各种不同字体的汉字，为了保留笔体，必须采用高密度点阵，对于港台等仍使用繁体字形的地区，这种要求尤为重要. 但是高密度点阵占用空间较大，一个192 × 192 的点阵占用4816 字节，汉字压缩存储是解决空间不足的唯一方法，即只有到真正显示汉字字形时再将其展开还原成原来的点阵字形. 目前有许多可以利用的压缩算法，一般的汉字点阵采用一维压缩算法，对超高密度的汉字采用二维压缩算法，它可以将1380 × 1380 的繁体字压缩到1 ~ 4K 左右，并且具有很高的还原速度.

2.2 C-tree^{TM} 管理汉字磁盘存取

汉字采用压缩存储后，磁盘就不能按照文件形式顺序存储汉字点阵信息，而必须为每个汉字点阵建立地址索引. 由于所需的字集(包括各种笔体和不同大小) 非常庞大，给索引的组织和查找带来较大困难，不合理的索引组织会造成磁盘访问次数增多，从而影响汉字的外存查找速度. Faircom 公司开发的C-tree^{TM}[2] 在提高索引查找速度上考虑了索引文件大小和磁盘物理特性，因此很适合管理大量可变长记录. C-tree^{TM}4.1 版本提供了很好的C 语言接口，为此专门设计了一个C-tree^{TM} 接口程序，用以管理维护存在磁盘上的汉字库.

2.3 LFU 置换算法

虽然系统要求支持各种笔体的汉字，但并不要求所有笔体同时处于工作状态，因此可以对各种笔体分别处理，其内存空间采用动态分配的办法. 只对每一种笔体的内存汉字个数作限定，具体可用多少内存空间要视系统当时的工作环境而定. 较常用的置换策略有FIFO、LRU 和LFU. FIFO 置换策略没有考虑汉字进入内存后的使用频度[3]，置换效率较差. 而LRU 策略需要的汉字使用信息较多，考虑到算法的效率和实现开销，系统选择了LFU 置换策略，当内存汉字达到最高限额又发生缺字时，就将内存汉字中自进入内存后使用次数最少的汉字所占空间归还给系统，以便让新的汉字进入内存. 为了记录内存汉字的使用频度，加快统计速度，设计了以下表格:

* 笔体目录表(Font Pattern Catalog Table)

Font_id: 笔体内部标识

parameter: 有关该笔体的尺寸等参数

quota: 内存汉字最大限额

Font_FIT: 内存汉字索引表指针

FIT_cnt: 内存汉字个数

Font_FFT: 内存汉字频度表指针

Hit_count: 命中次数

Mis_count: 不命中次数

* 内存汉字索引表(Font Index Table)

Ideo_id: 汉字内码

Font_ptr: 汉字点阵信息指针

FFT_ent: 该汉字在频度表中的项号

* 内存汉字频度表(Font Frequency Table)

FIT_ent: 该汉字在索引表中的项号

Frequency: 该汉字的使用次数

汉字访问过程如图1 所示. 由图中可知当系统空间不够时，为了调入一个新汉字可能逐出多个较少使用的内存汉字，这是因为同种笔体汉字所占用的空间不一致而引起.

§3. LFU 算法实验分析

3.1 实验环境

汉字二级存储系统在WANG–286微型机上实现，主频选10MH$_z$, 内存640K, 磁盘30M. 使用DOS3.2, C-tree^{TM}软件包和C5.0 及其开发工具. 本实验旨在对系统及其LFU 置换算法作初步评价，并试图找到内存汉字个数与缺字率的关系.

3.2 汉字访问序列的模拟

现有汉字使用频度的统计资料不能反映特定领域的特殊情况，例如医学文章和政论文章的常用汉字有很大差异，所以不能根据这些统计资料模拟汉字访问序列. 不失一般性，我们假设汉字的使用概率密度符合正态分布，即:

$$p(x)=\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x-x_0)^2}{2\sigma^2}} \qquad (1)$$

其中x 为汉字的顺序编码，σ 为特征常数，$p(x)$ 正比于x 出现的概率密度.

为了模拟符合上式的汉字访问序列，将其转换成汉字使用次数，即:

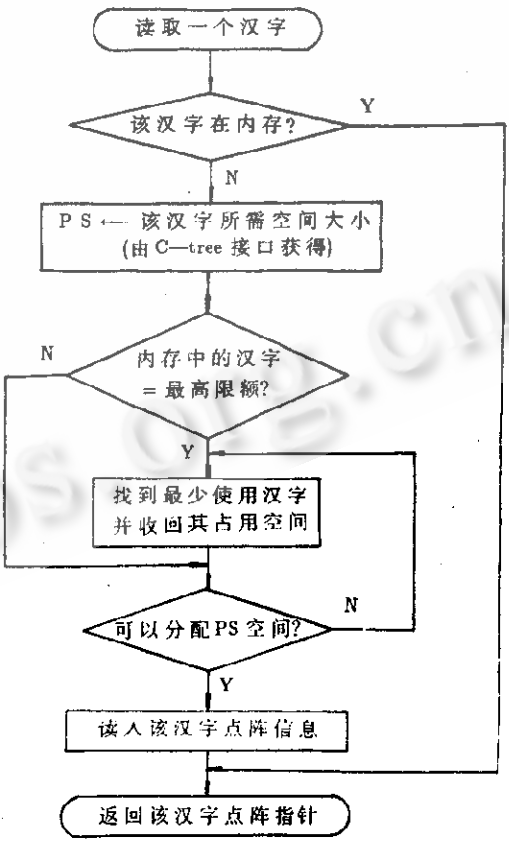

图1 汉字访问过程

$$f(x) = Ae^{-\frac{(M(x)-x_0)^2}{\lambda^2}} \tag{2}$$

其中$f(x)$为汉字访问次数，$M(x)$是一对一的随机映射，这样可以消除字库特殊存放对实验结果的影响．取汉字总个数为999个，$x_0 = 500$, 根据式(2)取不同的λ值，就反映了不同的分布特征．A取为$e^{(\frac{499}{\lambda})^2}$这样可以保证频次最少的汉字至少出现一次，$A$实际上是最常用与最少用汉字间的频次比．

确定A、λ值后就可以计算各个汉字应出现的次数，再根据次数将该汉字编码重复写入访问序列文件(因有些访问序列占用几百K空间，所以不能将访问序列放在内存)．最后用洗牌程序将放在一起的汉字分离，使访问序列具有随机性．

3.3 实验结果

我们对不同的内存汉字限额和访问序列进行了实验，表1给出了内存汉字限额quota值与不命中次数的关系．由表1计算出的缺字率(不命中次数除以总访问次数)与quota的关系如表2所示．

表1

$A(\lambda)$	总访问次数	内存限额	100	200	300	400	500	600	700
15 (300)	7795		6884	5653	4607	3632	2768	2097	1496
39 (260)	17721		14826	11734	8669	6231	4516	2852	1701
75 (240)	31485		25062	18986	14227	10106	6293	3551	2268
283 (210)	104849		80211	57352	39214	24430	13664	7443	5662
989 (190)	332771		239928	167383	96503	49915	30243	20066	12645

表2

A	内存限额	100	200	300	400	500	600	700
15		0.88	0.73	0.59	0.47	0.36	0.27	0.19
39		0.84	0.66	0.49	0.35	0.25	0.16	0.10
75		0.80	0.60	0.45	0.32	0.20	0.11	0.07
283		0.77	0.55	0.37	0.23	0.13	0.07	0.05
989		0.72	0.50	0.29	0.15	0.09	0.06	0.04

§4. 结语

1. 由表2可以得到内存汉字限额quota与缺字率f的关系曲线如图2所示．我们认为A值取40～1000较符合大多数汉字访问序列，由此可以得出：

* 当允许一半汉字存在主存时，命中率可以高达80%, 对于专业性较强的文章甚至可以达到90%.

* 为了保证缺字率低于50%, 应该允许约三分之一以上的汉字库存在内存.

我们用一些实际汉字访问序列进行实验，结果与上述结论基本符合．

2. 根据对磁盘汉字访问速度的测算证明用C-tree管理压缩存储的汉字库是有效的．本文所述的汉字二级存储系统在中文电子排版系统和其它要求多笔体高密度汉字点阵的系统中具有很高的实际应用价值．

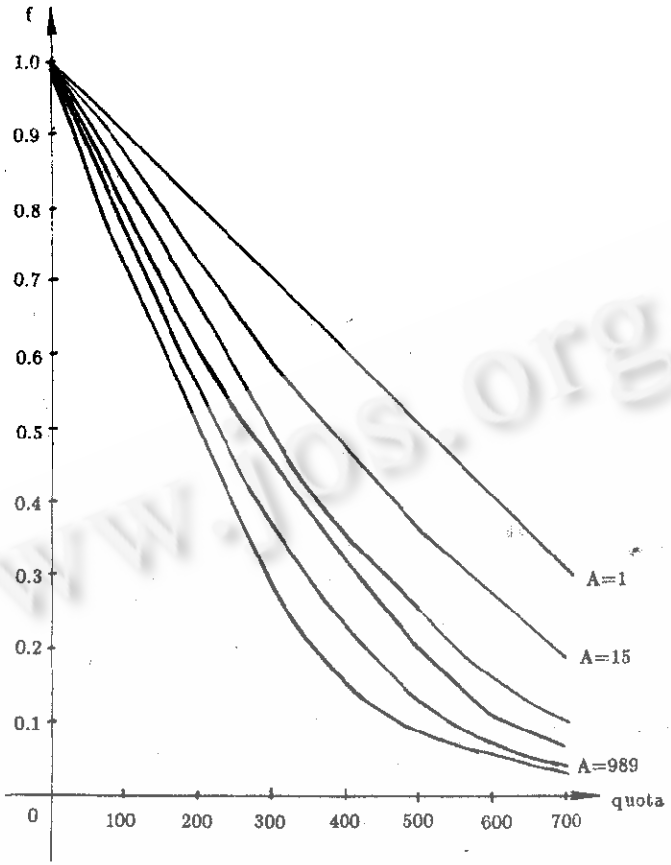

图2　缺字率与quota的关系

3. LFU 算法的一个缺陷是对先进入内存的汉字过于偏爱，新进入内存的汉字可能总成为置换对象，这对于常用汉字大批变更(例如开始编辑一篇新的专业文章)是极为不利的，我们曾试图附加“近期不使用惩罚”的工作，即当某个内存汉字长期不使用时就将其统计数字减值，但是没有找到满意的惩罚条件.

4. 为了使算法的效率得到充分发挥，我们在外存也设置了访问统计表，并根据统计表将汉字分为三类：最常使用汉字、次常使用汉字和极少使用汉字. 一种笔体一旦进入工作状态，最常使用汉字就自动装入内存并不再参予置换. 极少使用汉字不驻留内存，只有次常使用汉字使用LFU 算法进行动态置换. 这种分类处理方法一定程度上克服了LFU的上述缺陷，收到了较好的效果.

致谢：本课题的研究得到王安电脑有限公司香港R&D 的Eddie Fung 先生和厦门大学洪岷生副教授的帮助和指导，特致谢意.

参考文献

[1] 洪岷生、吴克西，“C 语言及其开发工具— 调试器及运行库”，厦门大学出版社，1988. 3.

[2] FAIRCOM, “C-tree Programmer's Guide” 4th Edition. 1987.

[3] 孙国圣，“动态字库的原理与实现”，微计算机应用，第4 期，1989, 第21-26 页.