第 33 卷 第 2 期 计 算 机 工 程 2007 年 1 月
Vol.33 No.2 Computer Engineering January 2007
·软件技术与数据库· 文章编号：1000—3428(2007)02—0077—02 文献标识码：A 中图分类号：TP311.12

高速缓存参数无关漏斗排序算法及其实现

吴英杰，王晓东，王一蕾
(福州大学数学与计算机科学学院，福州 350002)

摘 要：分析了 RAM 模型与现实计算机之间存在的差异，介绍了一个新的存储模型——理想高速缓存模型。描述了漏斗结构，设计并实现了高速缓存参数无关漏斗排序算法，通过仿真试验验证了该排序算法的有效性。
关键词：理想高速缓存模型；参数无关；漏斗排序；算法

Cache Oblivious Funnelsort Algorithm and Its Implementation

WU Yingjie, WANG Xiaodong, WANG Yilei
(College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350002)

【Abstract】 The difference between RAM model and computer is analyzed, and a new memory model—ideal-cache model is introduced, then the funnel structure is analyzed detailedly, and cache-oblivious funnelsort algorithm is designed and implemented. Lastly, the availability is validated through simulation experiment.
【Key words】 Ideal-cache model; Cache oblivious; Funnelsort; Algorithm

现代计算机的存储系统变得越来越复杂，它由多级高速缓存、内存和磁盘组成一个层次结构，指令也不再是以常数时间顺序执行。对该层次存储体系中较低且速度较快的层次中数据的访问可立即得到响应，而对较高层次的访问可能导致数百万的处理器周期延迟。因此，近年来算法设计策略发展的重点在于，寻找对存储系统较高层次访问次数最小化的最优算法。目前许多优秀算法的设计与实现都是基于一个特定的层次，跨平台性差。为了克服这一点，1999 年Prokop等人[1,2]提出了理想高速缓存模型和高速缓存参数无关算法的概念。高速缓存参数无关算法的目标是，在不了解任何有关各层次参数的情况下，最优地使用存储系统。并自动使算法对多级存储体系的每一级都是有效的。然而，目前对这些类型算法在实验方面所做的工作还相当少。本文主要通过性能测试，把高速缓存参数无关算法与传统的基于RAM模型的算法做了比较，在实验方面验证高速缓存参数无关算法的有效性。

1 理想高速缓存模型

理想高速缓存模型，又称高速缓存参数无关模型。它的理论基础是，算法可以不依赖于存储系统的硬件参数(M 和 B)，并以最优方式使用每一个高速缓存，见图 1。

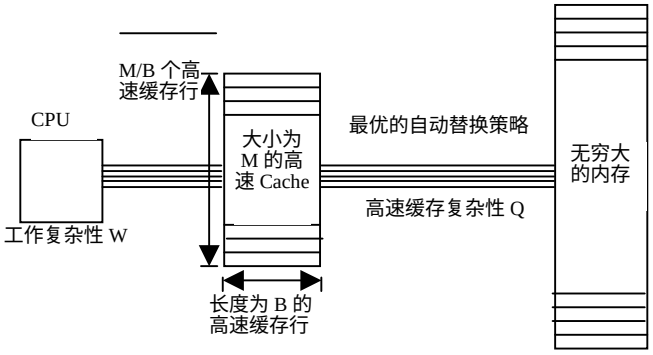

图 1 理想高速缓存模型

在理想高速缓存模型中分析一个输入规模为 N 的算法时，用两个量来衡量其计算复杂性：(1)工作复杂性，即熟知的 RAM 模型中的指令数；(2)高速缓存复杂性，即高速缓存缺失数。

2 高速缓存参数无关排序算法

2.1 合并排序

合并排序是将两个或两个以上的有序子表合并成一个新的有序子表。对于两个有序子表合并成一个有序表的两路合并排序来说，初始时，把含 n 个结点的待排序序列看作有 n 个长度都为 1 的有序子表所组成，将它们依次两两合并得到长度为 2 的若干有序子表，再对它们作两两合并，直至得到长度为 n 的有序表，排序即告完成。

显然，2－路合并排序算法是高速缓存参数无关算法。但其总的主存传输次数 T(N)的递归式为

$$T(N) = 2T(N/2) + \Theta(N/B)$$

求得：

$$T(N) = \Theta\left(\frac{N}{B}\left\lceil \log_2^{N/B} \right\rceil\right)$$

我们的目标是，在不知道 M 或 B 的情况下，把上式中对数的底数从 2 提高到 M/B。

2.2 高速缓存参数无关漏斗排序

在多路合并排序中，可用二项堆或等价的优先队列简单地实现合并。但是不知道 M 或 B 的大小，不知道它们是否能装入高速缓存中。例如，若二项堆不能装入高速缓存中，则其 I/O 性能将严重下降。因此，需要寻找一种新的合并结构，不管 M 和 B 的大小，都能进行有效的合并。

基金项目：福建省自然科学基金资助项目(A0510008)；福州大学科技发展基金资助项目(2004-XY-13)
作者简介：吴英杰(1979－)，男，助教，主研方向：数据结构，算法设计与分析；王晓东，教授；王一蕾，助教
收稿日期：2006-02-03 **E-mail：**yjwu@fzu.edu.cn

2.2.1 漏斗合并结构

漏斗是一种高速缓存参数无关的多路合并结构。k路漏斗是一种树状结构，能合并k个已排好序的输入序列，每次合并k^d (d>1)个元素。且具有 2 个显著特点：(1)它是按Van Emde Boas布局[4]递归存储，上下子树本身可看成是一个子漏斗；(2)每一个树边都放置一个缓冲区。

以下以 16－路漏斗为例加以说明，如图 2 所示。它由 5 个 4－路漏斗 L_0, L_1, L_2, L_3, R 和 4 个缓冲区 B_0, B_1, B_2, B_3 组成。漏斗是按 Van Emde Boas 布局存储在主存的连续段中：首先，顶层子树 R 的递归排列，紧接其后的是 B_0，然后是 L_0 的递归排列，接着是 B_1，依次往下。

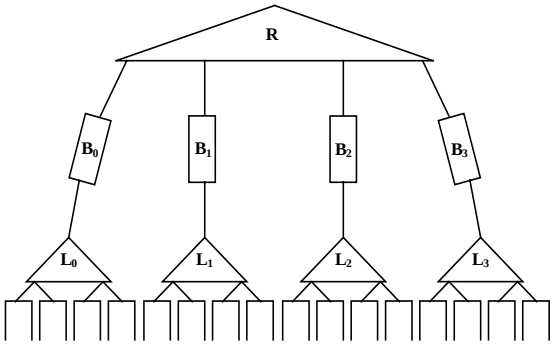

图 2 16－路漏斗结构

设取d=3，则对一个漏斗的调用，是对它顶层的子漏斗进行$k^{3/2}$次的调用，每次输出$k^{3/2}$个元素到缓冲区，共输出k^3个元素。每次调用之前，都要先检查顶层子漏斗的输入缓冲区是否有足够的元素。对每个元素个数少于$k^{3/2}$的缓冲区，必须调用以该缓冲区作为输出的底层子漏斗。基于这一点，缓冲区采用先进先出的队列，大小至少为 $2k^{3/2}$。在图 2 中，4 个缓冲区都能存放 128 个元素。这将确保当缓冲区最多只有$k^{3/2}$个元素时，还有足够的空间存放额外的$k^{3/2}$个元素，即调用下层子漏斗所产生的输出；同时也保证，即使在最极端的情况下，漏斗所有的输出都来自同一个输入流时，输入缓冲区里也有足够的元素。

2.2.2 漏斗排序算法

漏斗排序类似于合并排序。为了对数组中连续的 n 个元素进行排序，漏斗排序算法的基本思想如下：

(1)将输入分割成 $n^{1/3}$ 个大小为 $n^{2/3}$ 的连续子数组，递归地对这些子数组进行排序；

(2)使用一个 $n^{1/3}$ 路合并器对 $n^{1/3}$ 个已排好序的序列进行合并。

漏斗排序的算法描述如下：

funnelsort(Array A)

{ if $|A| \le \alpha z^3$ sort A “manually”

else

{ split A into roughly equal-sized array S_i, $0 \le i < (|A|)^{1/3}$

for $0 \le i < (|A|)^{1/3}$ funnelsort(S_i)

construct a $(|A|)^{1/3}$-funnel F

attach each S_i to F

warmup(F.root)

fill(F.root) }

}

经过分析可以得到如下结论：

用高速缓存参数无关的漏斗排序算法对n个元素进行排序时，其工作复杂度为 $O(n \lg n)$，同时发生 $\Theta(1+(n/B)(1+\log_M n))$ 次高速缓存缺失[1~4]。这在渐进意义上是最优的。

2.2.3 实验结果

测试环境如表 1。

表 1 测试环境

CPU (MHz)	L1 Cache		L2 Cache		RAM (MB)	OS	Compile
	M(KB)	B(bytes)	M(KB)	B(bytes)			
Celeron 4 1.7GHz	8	32	128	32	128	Linux	GNU3.2

基于以上测试环境，我们把二路合并排序算法与高速缓存参数无关漏斗排序算法的实验性能做了比较，结果如图 3 所示。

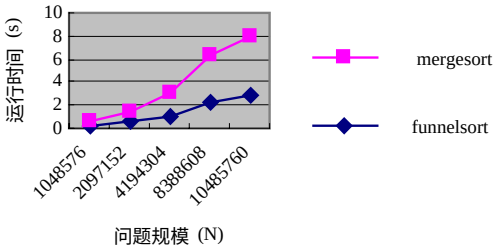

图 3 排序算法性能比较

从图 3 可以看到，随着问题的规模逐渐增大，高速缓存参数无关漏斗排序算法的实验性能明显优于二路排序算法的实验性能，这与理论分析的结果基本相符。

3 结语

本文详细分析了漏斗结构，设计并实现了高速缓存参数无关漏斗排序算法，接着通过仿真试验，把高速缓存参数无关算法与传统的基于 RAM 模型的算法做了比较。本文的主要结论是，当处理大数据量时，高速缓存参数无关算法显著优于传统的基于 RAM 模型的算法。

参考文献

1 Prokop H. Cache Oblivious Algorithms[D]. Massachusetts: Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1999.

2 Frigo M, Leiserson C E, Prokop H, et al. Cache Oblivious Algorithms (Extended Abstract)[C]. Proceedings of the 40th Annual Symposium on Foundations of Computer Science. IEEE Computer Society Press, 1999: 285-297.

3 Olsen J H. Cache-oblivious Algorithms in Practice[D]. Copenhagen: Department of Computer Science, University of Copenhagen, 2002-12.

4 Frederik. Cache-oblivious Searching and Sorting[D]. Copenhagen: Department of Computer Science, University of Copenhagen, 2003-07.