第 33 卷 第 9 期 计 算 机 工 程 2007 年 5 月
Vol.33 No.9 Computer Engineering May 2007
·开发研究与设计技术· 文章编号：1000—3428(2007)09—0272—03 文献标识码：A 中图分类号：TP333

一种基于近似 LRU 算法的高缓方案

鲍东星[1]，李晓明[2]
(1. 黑龙江大学电子工程学院，哈尔滨 150080；2. 哈尔滨工业大学微电子中心，哈尔滨 150001)

摘 要：提出了一个用于扩充高缓块管理的近似 LRU 算法。利用该算法，设计了一个可过滤 LRU 数据块的扩充高缓方案——LRU 块过滤高缓(LBF 高缓)。仿真结果显示，LBF 高缓的性能优于类似结构的扩充高缓(如牺牲高缓和辅助高缓)，与具有 2 倍容量的直接映像高缓相比性能有所提高。

关键词：高缓性能；命中率；LRU 算法

Cache Scheme Based on LRU-like Algorithm

BAO Dongxing[1], LI Xiaoming[2]
(1. Institute of Electronic Engineering, Heilongjiang University, Harbin 150080;
2. Microelectronics Center, Harbin Institute of Technology, Harbin 150001)

【Abstract】 The LRU-like algorithm is proposed for the block management of augmented cache scheme. Based on the LRU-like algorithm, a kind of augmented cache named least-recently-used blocks filter cache (LBF cache) is designed. After the simulation, the performance of LBF cache shows better than some augmented caches with similar architectures (such as victim cache and assist cache), and also better than the traditional direct-mapping cache with double size.

【Key words】 Cache performance; Hit ratio; LRU algorithm

到目前为止，大部分高缓结构优化技术均着眼于数据的重用，即把近期可重用数据尽量保存在片上高缓中，但是由于应用程序的任意性、不确定性，使得数据块重用性很难被甄别出来，因此高缓的利用率无法达到最优。

在高缓设计者的眼中，高缓访问时间和高缓的命中率是同样重要的，因此将高缓访问设计成在两条访问路径上进行，即通过让快路径满足绝大多数 CPU 的访存操作，并且与此同时通过另外的慢路径来减少访存的缺失代价。

依照这种思路提高高缓性能的方法可大略分为 4 类[1]：解耦高缓(decoupled cache)，多访问高缓(multiple-access cache)，扩充高缓(augmented cache)以及多级高缓(multilevel cache)。其中扩充高缓是在直接映像高缓(Direct-Mapped Cache，DM高缓)的基础上扩充了一个小的全相联高缓，2 个高缓可并行访问，同时由于全相联高缓的容量很小，因此总的访存时间与DM高缓相当。牺牲高缓[2]和辅助高缓[3]是最典型的扩充高缓，此外近年来还出现一些结构及策略较为复杂的STAS 高缓[4]、DASAT 高缓[5]等。

受二路组相联高缓的 LRU 替换算法的实现方法的启发，本文提出了一个用于扩充高缓结构的近似 LRU 算法。

1 近似 LRU 算法

当 CPU 对某一数据进行读写时，从程序指令上看它只是在访问一个字(半字、字节等)，但是从高缓的角度看，CPU 在访问整个数据块，因此局部性定理可以从不同角度来阐释。

从程序指令访问的角度来看，当某个字被 CPU 访问，那么其相邻存储区的字很有可能在最近被访问；并且这个字很有可能在不久被再次访问。

从存储器的角度来看，当存储器中某一块被访问，那么其相邻存储区的块很有可能在最近被访问；并且这个块很有可能不久被再次访问。

从 LRU 替换算法的原理出发，可以重新看待局部性定理。LRU 算法使用了一个局部性定理的推论：一个最近被使用的数据块很可能会被再次访问，那么最应该被替换的数据块就是最近最少使用的那一块。

设若有n个竞争高缓同一组的块B_{n-1}，B_{n-2}…B_m…B_1，B_0，在程序运行的一段时间里，依照从 0 到n-1 次序调入高缓中。程序访问不一定是只集中在新调入的块上，在更小的时间间隔内，它可能在其中任何一块上，此块附近存储区最有可能是当前工作集所在。因此将每个块做一个标志，以区分哪一个块是LRU块以进行替换，该块附近的存储区是当前程序的工作集的可能性最小。目前广泛应用的组相联高缓就是应用此原理。

在 DM 高缓结构下是否可以借用 LRU 算法呢？如果有一个缓冲器用来过滤 LRU 块，使得 DM 高缓中总保存 MRU 块，就可能提高高缓命中率。

假设B_m在DM 高缓中，而其它同组块在缓冲器中，这n块最理想的实现LRU算法的方法是每一块都附加一个计数器，但是这样一来必然大大增加硬件开销，因此考虑是否可以向 2 路组相联高缓的LRU算法实现那样只增加少量硬件，来进行近似的LRU替换。2 路组相联高缓的LRU算法实现方法就是在每一组中设立一个指示位，指明该组中 2 块之中哪一块是最近被访问，此位相当于最近最多使用(MRU)块 指针。

作者简介：鲍东星(1969－)，女，讲师，主研方向：电子电路设计，计算机体系结构技术；李晓明，博士

收稿日期：2006-05-26 **E-mail：**bdx2008@yahoo.com.cn

那么将B_m设置为一组，而缓冲器中其它同组块为另一组，分别称为B_{DM}’，B_{BUF}’。也就是说，缓冲器中包涵的所有块访问行为由B_{BUF}’负责。当DM 高缓命中时，B_{DM}’标识为MRU块；反之当缓冲器命中时，B_{DM}’ 中同组块标识为LRU块。当缺失块取回时，按照DM 高缓中同组块是否为MRU块来决定填充行为：当其为MRU块，缺失块向缓冲器填充；否则替换DM 高缓同组块，即LRU块被替换。由于DM 高缓中的LRU块B_{DM}’相对于缓冲器中其它同组块来说，不一定是真正意义上的LRU块，因此这种对DM 高缓块的替换方法称为近似LRU算法。

这里只有缓冲器中存在同组块时，才可以应用此方法，也就是说，如果缓冲器中不存在同组块，DM 高缓中的块总是 MRU 块。当缓冲器中无同组块时，缺失块应只填充缓冲器,因为当高缓中只有一个同组块时,参照 2 路组相联的 LRU 算法最好保留之。这样一来，DM 高缓中总保存最近一次的 MRU 块，来躲避被过滤的危险。

这种算法是一种近似的LRU算法，但是在多数条件下，2 个存储体之间基本上只存在 2 个同组块，即B_{BUF}’中基本上只包涵一个可能的高缓同组块。而当B_{BUF}’中包涵同组块时，过滤掉的块一定不是MRU块，因为此时MRU块在DM 高缓中或者为新填充的块，而且由于这种情况发生几率很小，因此对性能影响不会很大。

2 LRU 块过滤高缓结构

依据上节的分析,本文提出了一个有通道扩充高缓结构，称为最近最少使用块过滤高缓(Least-Recently-Used Block Filter 高缓，LBF 高缓)。如图 1 所示，该结构包括 2 个存储体：一个大的 DM 高缓，以及一个全相联组织的缓冲器，称之为 LBF 高缓，二者之间存在着一条单向数据转移通道。

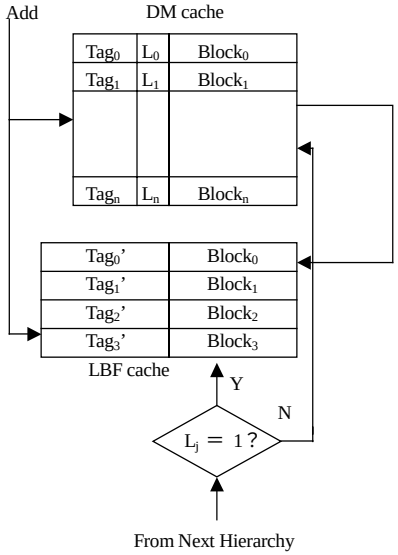

图 1 LBF 高缓示意图

在DM高缓中每一块的标签都增加了一个LRU块判断位(简称 L 位)，它用来探测该高缓块是否最近被访问。LBF 高缓的应用方法如下：

(1)高缓访问命中：当 CPU 访问高缓时，高缓控制机构同时访问 DM 高缓和 LBF 高缓。当 DM 高缓命中，那么相应块标签中的 L 位被置为 1；如果 LBF 高缓命中，则将 DM 高缓同组块标签中的 L 位清零。

(2)高缓访问缺失：如果 CPU 访问高缓缺失的话，新的高缓块将从下一级存储器中被取回。此时硬件机制检查 DM 高缓中同组的高缓块的 L 位，如果 L 位为 0，则取回块填充 DM 高缓；如果 L 位为 0，则取回块将填充 LBF 高缓。

(3)高缓块转移：在流水停顿等待的时间里，从 DM 高缓中驱逐出来的块填充 LBF 高缓。

(4)高缓块填充：当高缓块第 1 次填充到 LBF 高缓中时，DM 高缓中同组块的 L 位清零，表明为 LRU 块，因为新块已经被访问过一次；当 DM 高缓首次填充时，其 L 位置为 1。

3 实验仿真环境

本文采用了SimpleScalar工具集[6]中的sim-outorder仿真器作为仿真平台，其中用mlcache高缓仿真器[7]来取代sim-outorder中相关的数据高缓模型。本实验只检验L1 数据高缓的性能，因此指令高缓、L2 及总线假设为理想结构。仿真器和存储器系统的参数采用sim-outorder缺省设置。我们采用 SPEC95 标准测试程序集中 8 个典型的程序来进行仿真，其中除了compress程序采用训练数据集(train data set)作为仿真输入外，其它程序的输入文件均来自测试数据集(test data set)。所有的仿真程序都运行到结束。

对于给定结构的高缓，定义失效率(miss rate)为

L1 数据高缓失效数/ L1 数据高缓访问总数 (1)

总线交通量(bus traffic)，即 L1 与下一级存储器通过总线交换的数据量，是另一个重要的性能参数，对于整个处理器系统而言，总线交通量越小，系统的速度会越快。本文将由数据高缓申请的通过总线的数据总量(以百万字为单位)作为总线交通量的衡量标准。为便于比较，定义相对总线交通量为

基准结构的交通量——目标结构的交通量 (2)

我们限定所有参与比较的双路径高缓容量固定为(8+1)KB，其中主高缓为 8KB，辅高缓为 1KB，所有的全相联结构的硬件组织都应用 LRU 替换算法。

4 仿真结果

同前 2 种高缓设计方案采用的系统仿真环境一样，我们在 SimpleScalar＋mlcache 仿真器上进行了仿真。选择了 4 种高缓结构进行仿真和性能比对，即 DM 高缓、2 路组相联高缓、牺牲高缓和辅助高缓结构。所有参与比较的扩充高缓容量固定为(8+1)KB，32B 块长，其中 DM 高缓容量为 8KB，全相联高缓为 1KB。作为基准结构的 DM 高缓则分别为 8KB 和 16KB。

4.1 缺失率

表 1 给出了不同组织形式的 L1 数据高缓结构在运行 8 个 SPEC95 程序后得到的缺失率的数值结果。

表 1 8 个高缓方案的缺失率

	DM8K	DM16K	8K2W	Victim	Assist	LBF
comp.	0.0673	0.0557	0.0563	0.0553	0.0562	0.0543
gcc	0.0462	0.0288	0.0300	0.0259	0.0287	0.0265
li	0.0231	0.0178	0.0148	0.0141	0.0148	0.0137
ijpeg	0.0564	0.0221	0.0448	0.0111	0.0118	0.0104
perl	0.0597	0.0373	0.0288	0.0246	0.0281	0.0235
hydro.	0.1195	0.1063	0.1112	0.1094	0.1095	0.1078
su2cor	0.0891	0.0796	0.0768	0.0709	0.0722	0.0723
Swim	0.4068	0.1424	0.3904	0.0473	0.0472	0.0463

从结果可以看出，对于不同程序，LBF 高缓缺失率基本上均低于辅助高缓(除 su2cor)；除了在 gcc 和 su2cor 中，LBF 高缓的缺失率均低于牺牲高缓。在 3 种方案中，LBF 高缓的平均缺失率最低，为 5.34%，而牺牲高缓为 5.38%，辅助高缓为 5.48%。

与 16KB DM 高缓相比，(8＋1)KB LBF 高缓平均缺失率

减少约 26%，与 8KB 2 路组相联高缓相比减少 53%。由资料可知，(8+1)KB STAS 高缓与(8+1)KB 牺牲高缓缺失率(块长 32 字)相当，由此可推知略差于本文的 LBF 高缓。

4.2 总线交通量

将 16KB DM 高缓与下一级存储器总线上数据交换总数为基准，与 LBF 高缓、牺牲高缓、辅助高缓以及 2 路组相联高缓的相应值按式(2)计算后，便得到表 2。从表中可以看出，在总线读写次数上，LBF 高缓最少，尤其是在 swim 中少得更多。但是 LBF 高缓总线交通量除了在 ijpeg、perl 以及 swim 中少于 16KB DM 高缓外，相应指标要弱于 16KB DM 高缓。

表 2 以百万字为单位的相对总线交通量

	LBF	Victim	Assist	8K2W
comp.	-3.3	-11.1	-8.2	0.1
gcc	-6.4	-64.4	-30.1	-0.4
li	-3.0	-122.9	-57.9	16.4
ijpeg	0.5	-6.4	-0.3	-2.1
perl	2.07	-14.3	-3.4	2.4
hydro.	-129.8	-319.8	-274.9	-8.8
su2cor	-90.9	-309.1	-212.9	11.4
Swim	177.9	-449.5	124.8	-144.0

4.3 LBF 高缓硬件实现分析

与牺牲高缓和辅助高缓相比，LBF 高缓在硬件实现方面有自己的优势。首先，LBF 高缓没有牺牲高缓的双向通道，因此它的单通道设计节省了硬件设计复杂度。其次，与辅助高缓相比，它在数据转移判断上要更加合理。同 PA7200 辅助高缓相比，LBF 高缓转移判断直接通过硬件进行，而无需软件编译器的暗示支持，因此其可移植性要大大强于 PA7200 辅助高缓。

5 结束语

本文提出了一种利用近似 LRU 算法进行块过滤的办法，以将使用效率低的块驱逐出片上高缓，来提高高缓的利用率。按照此方法，设计了一种新的扩充高缓，称为 LRU 块过滤高缓(LBF 高缓)。仿真结果证明，(8＋1)KB LBF 高缓的缺失率明显低于 16KB 传统 DM 高缓和 8KB 2 路组相联高缓。与同类结构的牺牲高缓和辅助高缓相比，LBF 高缓结构平均缺失率有所降低，硬件相对容易实现。

参考文献

1 Peir J K, Hsu W W, Smith A J. Functional Implementation Techniques for CPU Cache Memories[J]. IEEE Transaction on Computers, 1999, 48(2): 100-110.

2 Jouppi N P. Improving Direct Mapping Cache Performance by the Addition of a Small Full Associative Cache and Prefetch Buffers[C]// Proceedings of the 17th International Symposium on Computer Architecture, Seattle. 1990: 364-373.

3 Kurpanek G, Chan G, Zheng K, et al. PA7200: A PA-RISC Processor with Integrated High Performance MP Bus Interface[C]//Proceedings of IEEE International Computer Conference, San Francisco. 1994: 375-382.

4 Lee Jung-Hoon, Lee Jang-Soo, Kim Shin-Dug. A Selective Temporal and Aggressive Spatial Cache System Based on Time Interval[C] //Proceedings of the IEEE International Conference on Computer Design, Austin. 2000: 287-293.

5 Lee Jung-Hoon, Kim Shin-Dug, Weems C C. Application-adaptive Intelligent Cache Memory System[J]. ACM Transactions on Embedded Computing Systems, 2002, 1(1): 56-78.

6 Burger D, Austin T M. Evaluating Future Processors: The Simple Scalar Tool Set[R]. Madison: University of Wisconsin, 1997.

7 Tam E S, Rivers J A, Tyson G S, et al. Mlcache: A Flexible Multilateral Cache Simulater[C]//Proceedings of the 6th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, Montreal. 1998: 19-26.

（上接第 247 页）

从而进行有效的数据传输。对于监控代理程序而言，主要是设备状态数据传输的设计。

(1)设备状态数据传输的设计

设备状态数据传输主要指把设备状态数据信息以何种数据形式发送给远程客户端使其能够识别不同设备工作状况。

在双容水箱系统中，采用如下格式实现系统的状态信息。

Poin (测点)	DataLength (数据长度)	Data (数据)	Time (时间)	Others (其它信息)

状态数据的传输有两种情况：一种是本地监控程序连续、高频率地扫描需要监视测点的状态，并通过远程监控代理程序发送给远方监控人员；另一种是低频率或一次性查看，远程监控人员通过控制命令请求来要求查看设备某些测点某一时刻的状态，本地监控站扫描该测点，然后把测点监控数据发送给远方监控端。前者主要是针对设备处于过渡工况下进行远程监控，后者是针对设备处于稳定工况下进行远程监控。基于 PLC 的双容水箱实验系统是让远方实验者验证控制规律对系统调节过程的影响，主要是监视系统的过渡过程，故采用第 1 种状态数据的传输方式。

(2)Socket 的服务器端实现

实现 Socket 的服务器端主要是利用 C++Bulider 提供的 TServerSocket 组件，对 TServerSocket 对象进行初始化。

在这里允许一个 Socket 服务器同时连接多个客户端，但只允许一个用户对实验台进行实验，其它用户以“只读”的形式访问已被占用的实验台，观看做实验用户的实验情况。

(3)实验数据的存储

在网络服务器中用 SQL Server 2000 定义了一个名为“实验数据”的数据库，在其中产生一个表名为“HistoryDataTable”的数据表，用于存放实验产生的历史数据。在远程监控代理程序中通过 ADO 实现实验数据以及用户的实验参数的存储。

本网络实验室的设备服务器都是处于在实验室里的局域网的内网中，为了能够让本局域网外的校园里其他计算机能够进行远程实验，我们在局域网的对外服务器上执行了一个名为 PorTunnel 的小软件，以实现端口映射的功能，通过访问服务器上 PorTunnel 设定的端口（外界当然可以直接访问服务器），达到访问局域网内运行实验设备的主机的目的。

参考文献

1 王洪猛, 谢建君. 基于 PLC 的过程控制系统的设计与实现[J]. 自动化技术与应用, 2000, 19(7).

2 谢建君, 王洪猛. 基于 LabVIEW 与 PLC 的串级控制系统设计[J]. 工业仪表与自动化装置, 2005, (2).

3 陈灿煌. C++Builder6 彻底研究[M]. 1 版. 北京: 中国铁道出版社, 2003.

4 张清林. 基于 Internet 的工业锅炉运程监控技术[J]. 自动化博览, 2004, (3): 55-56.

5 杨　静. 基于 OPC 规范和 Actirex 技术的 Web 实时监控[J]. 工业控制计算机, 2003, 16(5): 19-20.

6 李洪宝. 基于 Web 实时信息发布系统的设计与实现[J]. 计算机应用, 1999, 19(12): 55.