基于改进LRU替换策略的共享Cache划分

王 涛，朱怡安，黄姝娟
（西北工业大学 计算机学院，陕西 西安 710072）

摘 要：本文提出了一种基于改进的LRU替换策略划分最后一级共享Cache的算法，隔离了线程间的数据冲突，实现了改进的Cache替换策略，通过划分最后一级共享Cache也减少了访存延迟，提高了系统吞吐率.

关键词：多核；共享Cache；划分；替换策略
中图分类号：TP302 **文献标识码**：A **文章编号**：1000－7180(2012)01－0080－04

Shared Cache Partition Based on Improved LRU Replacement Policy

WANG Tao，ZHU Yi-an，HUANG Shu-juan
(College of Computer，Northwestern Polytechnic University，Xi′an 710072，China)

Abstract：There is some concern about how to partition the last shared cache to decrease the bad influence of resource competition and the rate of miss. This paper propose a method of partitioning the last shared cache using the modified LRU algorithm. It can eliminate the resource competition，decrease the memory latency，improve the replacement policy and improve system throughout.

Key words：multi-core；shared cache；partition；replacement policy

1 引言

当前在多核体系结构中Cache层次数目的选择和共享方式的选择仍然是个开放性问题.目前的多核处理器一般采用私有的L1Cache，而二级或者三级Cache是共享的.图1给出了共享Cache结构的示意图，共享Cache结构虽然访问速度较低，但具有失效率低的优点.传统的Cache替换策略如LRU（最近最少使用）并不能有效解决命中率低等问题，除此之外它并不区分不同应用的访问，因此一个应用的Cache失效可能会替换另一个应用的Cache块（又称之为Cache污染）.而且随着同时可执行处理器核的增多，Cache中的数据放置冲突也变得严重，进而增加了存储系统的平均访存延迟，而影响了系统吞吐率和性能的进一步提升，本文利用改进传统的LRU替换策略对共享L2Cache进行划分，从而解决之前访存命中率不高，访存延迟高和系统吞吐率低等问题.

收稿日期：2011－05－16；修回日期：2011－06－08
基金项目：航空科学基金项目(20100753022)

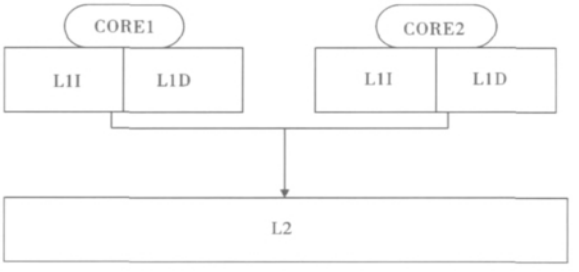

图1 共享二级Cache

2 相关研究

目前国内外已经有很多关于Cache划分机制的研究，它们主要分为静态划分和动态划分Cache容量.文献[1]引入了一种建立单独的硬件检测电路的方法，运用动态划分算法得到最优的Cache划分，IPC平均提高了11.8%.文献[2]针对CMP结构提出了基于统计信息静态地将Cache以存储体为粒度分配给不同的应用程序，其不足之处在于

需要在每次执行前对所有的应用程序都要统计执行状态. 文献[3]提出了共享 Cache 动态划分策略，该机制能够在同时执行的多个线程之间划分 Cache 容量. 该机制使用在 Cache 中命中 Cache 的缓存块的位置预测每个应用程序对 Cache 的使用性，具体操作是在线程执行过程中记录 Cache 的缺失类别，并根据操作系统的调度方式对 Cache 容量进行划分. 所光等人提出了面向多线程多道程序的加权共享 Cache 划分法，通过在处理器中加入面向应用的失效率监控器(AMRM)；在操作系统中加入共享 Cache 划分算法；并且给用户提供 Cache 划分支撑工具以便用户能够控制共享 Cache 的划分策略[4]. AMRM 以应用为单位预测该应用在不同共享 Cache 容量下的失效率. 内核态的 Cache 划分算法根据 AMRM 提供的失效率信息在并行执行的多个应用间动态调整 Cache 容量. 用户可以使用 Cache 划分支撑工具动态的配置 Cache 划分算法[5-6]. 该算法具有较高的 IPC 吞吐率、加权加速比和公平性，但依旧没有解决有较高失效率的问题. 文献[7-8]做了进一步研究.

以上研究提出的划分算法都要求对硬件的消耗，并且一部分需要操作系统的支持，从而减少了可应用性，本文提出了一种基于改进的 LRU 划分算法，实现硬件开销小，而且不需要其他额外支持，可减少缺失率，并提高系统吞吐率，该算法针对两级 Cache 进行讨论，一级 Cache 为私有，二级 Cache 为共享.

3　基于改进的 LRU 划分算法描述

Cache 替换算法有多种，例如 RAND，FIFO，LFU，LRU 等，而商用处理器更多的是采用伪 LRU(伪最近最少使用算法)进行替换. 传统的 LRU 算法是当 L2Cache 发生缺失从主存中替换缓存块时，选择最近最少使用的缓存块进行替换，在实际实现过程中一般需要利用栈结构记录每个 Cache 块的使用情况，从而实现最近最少使用算法. 这一方面需要一些硬件支持；另一方面需要记录使用情况，以便实现最近最少使用算法，这在数据量大的时候，有一定开销. 本文利用了一种改进的 LRU 替换算法，每个缓存块只需要一位的标记即可，减少了硬件的开销，当发生替换时只需要找到标记位为零且索引地址最小的数据进行替换即可，算法复杂度和实现难度都较之简单.

在介绍改进的 LRU 算法之前，需要先介绍支持该算法的硬件机制，其结构如图 2 所示，其中 TID 为线程标识号，AC 为访问标记位，Cache 数据块为存储的数据，Tag 为标识信息. 除此之外每个线程还需要有一个计数器用于记录替换自己所拥有 Cache 块的次数 Counter.

图 2　Cache 块结构

改进的 LRU 算法具体实现过程如下：其中 AC 位占用 1 比特数据，对组中某块 Cache 进行访问时就将其对应的 AC 位设置为 1. 当需要进行 Cache 替换时，选择 AC 位为零且地址最小的块. 为防止出现所有位都为 1 的情况，需要加入特别的机制，如果某次访问后发现所有的 AC 位均为 1，那么先将其他所有的 AC 位置零，再将这次访问对应的 AC 位置 1，如 4 路组相连 Cache 中，初始数据为[a，b，c，d]，AC 位为 0111. 若进行对 d 的访问，因为命中，不改变 AC 位的状态，若再对 a 进行访问，置 AC 为 1，此时发现所有 AC 位为 1，那么将对其他的 AC 位进行复位，此时 AC 状态为[1，0，0，0]，若再进行对 e 的访问，选择 AC 为零且地址最小的数据，替换 b，结果为[a，e，c，d]，其 AC 位分别为[1，1，0，0]. 具体过程如图 3 所示.

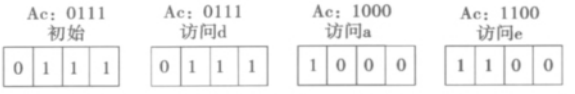

图 3　替换块过程描述

基于改进的 LRU 替换策略的划分算法主要是根据线程来进行 L2Cache 划分，这是因为多线程并行执行是多核处理器的一大特点，每个线程都在一个处理器核上执行，由于 Cache 结构中有线程标识号(TID)，因而这相当于按照线程进行了 L2Cache 划分，主要替换过程如下：线程 T1 进行数据访问，发现 L1Cache 失效. 此时需要对 L2Cache 进行访问，根据 L2Cache 是否命中进行选择，若不命中首先选择一个无效的缓存块进行替换. 若该组中没有无效的缓存块，则选择一个有无效 TID 的有效缓存块(该线程可能退出)，这表明该缓存块尽管有效，但被重新使用的可能性较低，且利用率较低，因此这个缓存块可以被其他处理器核新读入的数据块替换，如果不存在无效 TID 的有效缓存块，则取出线程 T1 的 Counter 值，若线程 T1 替换组内的次数小于

限定值，则采用改进的 LRU 算法选择一个属于同一个线程且使用频率最低的缓存块进行替换，具体步骤描述如下[8]：当没有无效的数据块，且不存在 TID 无效的数据块，那么就需要选择线程 T1 所拥有的 Cache 块进行替换，具体操作步骤根据改进的 LRU 算法进行，若线程 T1 替换组内的次数大于限定值，则从其他线程占有的 Cache 进行替换，选择为零且索引地址最小的 Cache 块进行替换，并将该替换进的 Cache 块的 TID 设置为 T1. 替换策略的整个流程如图 4 所示.

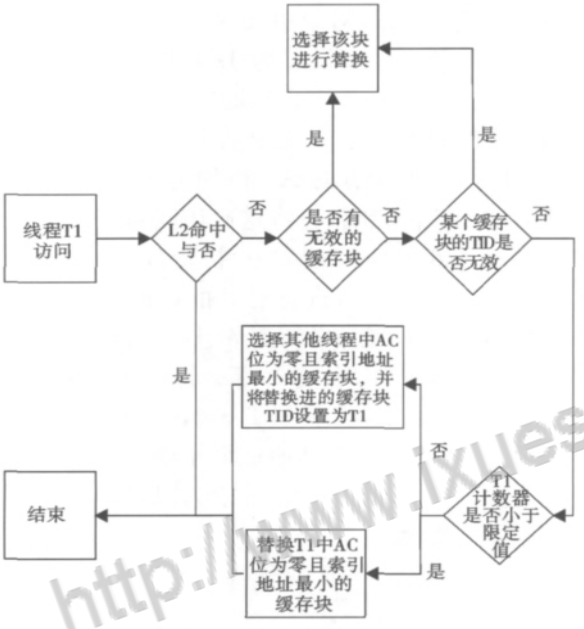

图 4 划分算法流程

根据以上改进的 LRU 算法对 L2Cache 进行了划分，划分是按照线程进行的，每个缓存块都拥有 TID 标识号，也就是实现了 L2Cache 按线程的划分，具体操作如下：首先根据目前所活跃的线程数对 L2Cache 进行平均划分，若不能划分完全则将剩余的部分作为无效的缓存块以备后面使用，然后在 Cache 的使用过程中根据以上提出改进的 LRU 算法进行动态划分 L2Cache.

4 实验结果及分析

仿真实验中，选择 M5 模拟器作为实验工具. M5 多处理器系统架构由初始化文件定义，一个初始化文件被分为几个配置片段，每个片段包含不同的参数. 简单来说，目标系统为一颗树，树中的结构都由初始化文件中的配置片段来定义. 树形结构的配置机制很好地适应了基于总线的多处理器系统. 我们的硬件模拟以及算法实现只需要修改配置文件和算法源码即可.

所选取的 Benchmark 是 SPEC CPU2006，考虑到是对 Cache 的性能进行仿真和测试，因此在其中挑选了一些具有代表性的 Benchmark. SPEC CPU2006 分别为 CINT2006（整数应用）和 CFP2006(浮点数应用)，因此在 CINT2006 中选择了 bzip2，gcc，mcf 这 3 个应用，在 CFP2006 中选取了 milc，soplex，named 这 3 个应用.

对于每个 Benchmark，分别将 L2 大小设置为 512KB，每块 Cache 大小为 32B，同时组相联度设为 8 路，分别利用改进 LRU 算法未划分最后一级 Cache 和改进的 LRU 算法划分最后一级 Cache 对命中率、IPC 进行了实验，实验结果如图 5、图 6 所示.

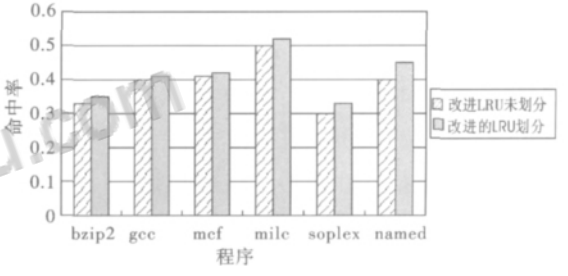

图 5 划分后命中率实验对比

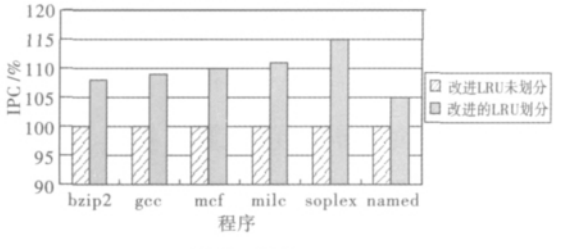

图 6 IPC

根据实验结果看出，基于改进的 LRU 算法进行的 L2Cahce 划分机制比起只简单改进 LRU 替换算法而未进行 L2Cache 划分机制有着更高的命中率，从而有着更好的性能. 另外在实际替换过程中，只需要找到访问标志位 AC 为零，且索引地址最小的 Cache 块用以替换，算法复杂度和实现难度也较 LRU 算法减小. 在开销方面，每个缓存块只需要一位标志位即可，硬件开销很小，这对于整体命中率的提高来说，几乎没有任何影响. 另外由于隔离了线程间的数据冲突，从而解决了数据冲突造成的访存延迟问题. 根据实验数据表明提高了 IPC 约 8%，进而提高了系统吞吐率.

5　结束语

对于片上多核处理器结构而言，片上最后一级 Cache 设计为所有处理器核共享是非常必要的，这样可以支持更大的 Cache 容量和减少片外缺失率. 然而由于多核多线程程序的出现，多个线程之间会对共享最后一级 Cache 进行资源竞争，从而产生冲突. 这不仅降低了 Cache 的命中率，而且影响了系统的吞吐率. 本文提出了一种改进的 LRU 算法，这种改进的 LRU 算法只需要每个 Cache 块多增加一个标记位，这种硬件开销较小，而且采用的是索引地址最小的方式进行查找，不用对 Cache 块进行统计以保证 LRU 算法的实现. 基于这种替换策略，进行了 Cache 的划分，将被替换的 Cache 从最后一级 Cache 中剔除，新进 Cache 块的 TID 设置为该线程 TID 号，从而根据不同的 TID 实际对片上最后一级 Cache 进行了划分. 通过对最后一级共享 Cache 进行划分隔离了多个线程之间的数据冲突，减少了访问延迟，提高了系统吞吐率. 实验表明，该方案是可行的，提高了命中率和吞吐率.

然而，为了减少硬件的开销使得对于 Cache 使用的统计不充分，这样造成了一定程度的 Cache 缺失，下一步工作是考虑如何实现既降低硬件开销又更好提高命中率的 Cache 划分机制.

参考文献：

[1] 杨磊，时磊，张铁军，等. 多核系统中共享 cache 的动态划分 [J]. 微电子学与计算机，2009，26(5)：56-59.

[2] Chun Liu, Anand Sivasubramaniam, Mahmut Kandemir. Organizing the last line of defense before hitting the memory wall for CMPs [C]// Proceedings of the International Symposium on High-Performance Computer Architecture (HPCA'004). Washington, DC, USA: IEEE Computer Society, 2004: 176-185.

[3] Suh G E, Rudolph L, Decadas S. Dynamic partitioning of shared cache memory [J]. Journal of Supercomputing, 2004, 28 (1): 7-26.

[4] 所光，杨学军. 面向多线程多道程序的加权共享 Cache 划分 [J]. 计算机学报，2008(11)：1938-1947.

[5] 所光，杨学军. 双核处理器性能最优的共享 Cache 划分 [J]. 微电子学与计算机，2008，25(9)：28-30.

[6] Qureshi M K, Patt Yale N. Utility-based cache partitioning: a low-overhead, high-performance, runtime mechanism to partiti-on shared caches [C]// Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture. Washington, DC, USA: IEEE Computer Society, 2006: 423-432.

[7] 宋风龙，刘志勇，范东睿，等. 一种片上众核结构共享 Cache 动态隐式隔离机制研究 [J]. 计算机学报，2009(10)：1896-1904.

[8] 李洪，毛志刚. PLRU 替换算法在嵌入式系统 cache 中的实现 [J]. 微处理机，2010(1)：16-19.

作者简介：

王　涛　男，(1986－)，硕士研究生. 研究方向为嵌入式计算及其应用.

(上接第 79 页)

[8] Adb-Elmoniem K Z, Youssef A M. Real-time speckle reduction and coherence enhancement in ultrasound imaging via nonlinear anisotropic diffusion [J]. IEEE Trans Biomed Eng., 2002, 9(49): 997-1014.

[9] Nasonov A V, Krylov A S. Fast super-resolution using weighted median filtering [C]// ICPR'2010. Istanbul, Turkey, 2010.

作者简介：

刘　刚　男，(1982－)，博士研究生. 研究方向为图像处理.