计算机研究与发展
Journal of Computer Research and Development
ISSN 1000-1239/CN 11-1777/TP
48(Suppl.): 322-327, 2011

龙芯处理器上的 TLB 性能优化技术

张晓辉[1,2] 程归鹏[1] 从 明[1]

[1](中国科学院计算技术研究所 北京 100190)

[2](中国科学院研究生院 北京 100049)

(zhangxiaohui@ict.ac.cn)

Performance Optimization Technology for TLB on Godson Processors

Zhang Xiaohui[1,2], Cheng Guipeng[1], and Cong Ming[1]

[1](*Institute of Computing Technology, Chinese Academy of Sciences, Beijing* 100190)

[2](*Graduate University of Chinese Academy of Sciences, Beijing* 100049)

Abstract TLB (translation lookaside buffer) is a cache of recent address translations to speed up memory access, which is indispensable in computer systems that use page based virtual memory. TLB is in the critical path of memory accessing, and affects the performance of the system significantly. Meanwhile, the cost of TLB miss is huge, which is the performance bottleneck of Godson processors. So the optimization of TLB system performance is important for the performance increase of Godson processors. In this paper, the superpages technology and software caching for TLB (STLB) technology are described, which optimizes the performance of system's TLB through decreasing TLB misses and reducing the overhead of TLB miss respectively. The optimized STLB technology -STLB combines with the support of locking level-two cache provided by Godson-3 processor is also introduced.

Key words Godson processors; TLB; superpages; STLB; locking level-two cache; Godson-3 processor

摘 要 TLB(translation look-aside buffer)是分页式虚拟存储系统用于加速虚实地址转换的必不可少的性能优化部件. TLB 处于访存的关键路径上,对系统性能有着至关重要的影响. 同时 TLB 失效开销大,是龙芯处理器的系统性能瓶颈. 因此,优化系统 TLB 的性能对于龙芯处理器系统性能的提升意义重大. 实现了龙芯处理器上通过减少 TLB 失效次数以及降低 TLB 失效开销的 TLB 性能优化方法而分别采用的超页技术和软 TLB 技术,以及结合龙芯 3 号处理器新增的锁 L2cache 功能,进一步优化了的软 TLB 技术.

关键词 龙芯处理器; TLB; 超页技术; 软 TLB; 锁 L2cache; 龙芯 3 号处理器

中图法分类号 TP303

现代处理器系统为了在较小的物理内存下运行规模相对庞大的应用程序,大都采用了分页式虚拟存储系统. 虚拟存储系统中虚实地址的映射关系是程序运行时动态建立的,存放于系统页表中. 为加快虚实地址的转译,绝大多数处理器都采用 TLB 缓存最近访问过的地址映射关系. 地址转译存在于每条

收稿日期:2010-12-10

基金项目:国家自然科学基金项目(60736012,60921002);国家"九七三"重点基础研究发展计划基金项目(2005CB321600);国家"八六三"高技术研究发展计划基金项目(2008AA110901)

指令的读取和数据读写上,因此 TLB 处于处理器访存的关键路径上.

随着处理器结构的不断改进,存储系统的快速发展以及应用程序规模的不断变大,TLB 失效例外处理占系统 CPI(cycles per instruction)的比重越来越大,TLB 的失效次数也越来越多.龙芯处理器采用类 MIPS 的 RISC 体系结构,其 TLB 例外由软件管理.MIPS 结构采用精确例外,例外发生时需刷新流水线而导致存在于 reorder buffer 中的大部分指令无效.软件管理 TLB 失效给予操作系统更大的灵活性,却增大了 TLB 的失效开销.通常用于 TLB 失效例外处理的指令多达几十条,这些指令本身也可能引发指令 cache 失效.为满足应用程序规模的不断增大的需求,页表的组织结构需要多级化(目前龙芯处理器的 64 位 Linux 系统采用 3 级页表结构).页表的级数越多,TLB 失效例外开销就越大.此外,龙芯处理器的 L1cache 采用虚地址索引物理地址标志,即每次从 L1cache 中读取指令或数据都要先经过 TLB 完成虚实地址转译.即使数据在 L1cache 中命中,而相应的 TLB 失效时,数据访问延迟依然很大.因此,TLB 是目前龙芯处理器系统性能的瓶颈因素.

1 相关工作

1.1 减少 TLB 失效次数的方法

减少 TLB 失效次数最为有效的方法就是超页技术,目前大多数处理器结构对超页技术都有基本的硬件支持.通常,硬件对超页的支持是有限制的,即组成超页的基本页必须连续且对齐到超页边界.目前操作系统采用请求调页,在页面请求时一页页地任意分配.这种页面分配策略无法满足超页的分配需求.操作系统需要修订页面分配策略以满足超页的分配需求,并对多个相关部分进行修改以完成对超页的透明支持,或者添加系统调用或库函数为应用程序映射超页提供适当的接口.支持超页的操作系统的关键实现即超页分配策略.主流的超页分配策略有两种:基于页面迁移的和基于保留的超页分配策略.另外一种比上述两种更激进的方法是修改硬件对超页的支持,突破超页固有的硬件限制.

基于页面迁移的超页分配策略按照请求调页机制分配,在适当的时刻通过将已分配的页面拷贝到符合条件的连续区域来创建大页.该策略可以灵活地在任意时刻创建任意大小的超页.创建大页的开销主要是页面拷贝的开销.所以,需要权衡超页创建的开销及超页对性能的提升效果.Romer 等人提出了类似竞争算法的 APPROX-ONLINE 算法[1],该算法在 TLB 失效例外处理中动态统计 TLB 失效次数,并根据程序局部性原理来确定是否生成及何时生成超页.该策略对应用程序使用超页的分析有很好的帮助,但在实际应用中,由于附加了较多额外开销使得系统臃肿而不能得到较好的效果.

基于保留的超页分配策略在超页的某一基本页分配时,预留属于该超页的其他基本页,当超页内的所有基本页都被访问过之后将它们合并为超页.该策略通过维护一个 reserved 链表管理那些为超页保留但未分配的基本页.保留基本页使得内存碎片增多,而造成超页请求得不到满足.此时可抢占保留页,抢占的后果是被抢占了的超页不可能再成为超页.因此,如果被抢占的超页恰巧是一个热点超页,则对系统性能影响较大.该策略由 Talluri 和 Hill 提出[2].Navarro 等人[3]通过在系统中增加碎片处理机制使得发生抢占的几率降低,大大弥补了该策略的不足.而且在内存容量越来越充足的今天,更倾向于基于保留的超页分配策略.

硬件支持的超页分配策略与上面的两种策略有着本质的不同,它们通过添加硬件的支持突破超页固有的硬件限制,即组成超页的基本页不必是连续对齐的.Talluri 和 Hill 参考子块 cache 的结构设计提出子块 TLB 结构[2].该策略仅需要对操作系统做微小的改动,但由于需要修改 TLB 自身的结构而没有得到广泛的应用.Swanson 等人提出的基于影子内存的超页 TLB 技术[4].该技术虽然不需要改变处理器结构,总线结构及 cache 等,却依然需要修改内存控制器(MMC).该方法仍然需要操作系统提供重映射的系统调用,并且映射失效的检测被提迟到 MTLB 层,增大了 TLB 失效例外开销.

1.2 降低 TLB 失效开销的方法

TLB 失效开销主要源于 TLB 失效例外处理时要查询页表,降低 TLB 失效开销的主要方法就是要尽量减少该失效处理过程中的访存次数以及访存时引发的 cache 失效次数.

FAST-TLB-REFILL 方法[5]通过缓存最后一级页表的指针使得 TLB 失效例外处理跳过对前两级页表的访问减少了访存次数,从而减小 TLB 失效开销.但该方法需要在进程切换时把新进程的第三级页表的指针拷贝到内核启动时初始化好的缓存中,增大了进程切换的开销.

Bala 等人[6]针对 MACH 3.0 操作系统提出了软 TLB 技术.该技术通过在内核中建立一个全局的软 TLB 表缓存更多最近发生的虚实地址映射.在 TLB 失效发生时,首先查询软 TLB 表而不用去查找多级页表,以减少 TLB 失效例外处理的访存次数和 cache 失效次数.MACH 3.0 内核是一种微内核结构,与 linux 内核有着较大的不同.微内核结构的客户进程、内核和服务进程间通信频繁,产生的 TLB 失效次数较多,性能提高幅度比较大但软 TLB 本身需要较大的内存.

2 龙芯处理器的相关特性

2.1 对超页技术的支持

提供超页机制以协助控制 TLB 映射空间的大小.通过协处理器 0(CP0)寄存器 PageMask 记录映射的页大小,同时在载入 TLB 中时将该记录写入 TLB 中相应的域.龙芯处理器可支持的页大小可以是 4 KB 到 16 MB,但必须是按 4 倍递增.龙芯 GS464 处理器核在将来可以在同一运行时刻支持不同大小的页,允许操作系统产生特定目的的映射.这种映射可以是对特定结构,如内核代码和数据,帧缓冲和数据库应用等;不同大小的页面也可以透明的应用于通用应用程序.

2.2 锁 L2cache 功能

龙芯系列处理器使用了 3 种独立 cache[4]:一级指令 cache 和一级数据 cache 容量均为 64 KB,cache 行大小为 32 B,采用四路组相联结构.一级 cache 采用虚地址索引物理地址标志的模式.二级 cache 为片上混合 cache.龙芯 3 号的片上二级 cache 通过 128 位 AXI 总线和处理器核通信,4 个二级 cache 模块实行全局编址,cache 行大小 32 B,1 MB 容量,采用四路组相联的结构和写回策略.龙芯 3 号的二级 cache 提供了锁机制.通过配置锁窗口寄存器,可以确保最多 4 个被锁住的区域不被替换出二级 cache(除非四路二级 cache 中都是被锁住的块).通过 confbus 可以对二级 cache 模块内部的四组锁窗口寄存器进行动态配置.每组窗口的大小可根据 mask 进行调整,但不能超过整个二级 cache 大小的 3/4.此外当二级 cache 收到 DMA 写请求时,若被写区域在二级 cache 中命中且被锁住,DMA 将直接写入到二级 cache 而不是内存.这种特性可用于优化系统系能及某些特定计算任务的性能.

3 龙芯处理器上的超页技术

超页由一组虚实地址连续且属性一致的基本页组成,并且超页的大小是基本页的 2 的幂次方倍.最初超页的提出主要是为了满足数据库应用,大型矩阵计算等程序的需求,这类程序在运行时需要较大的线性空间,核心循环中有较大的运算集合或者经常需要跳转[7].而对于像 web 需要访问较多小型文件的应用等就不会从超页中获利.相反,还可能因为超页的分配需要额外的系统开销而导致性能下降.因此,何时及如何分配超页是超页技术需要解决的问题.

3.1 超页标志的选取

内核中标识页的数据结构为 struct page,但由于页是内存管理系统的基本单位,涉及很多内核核心算法及虚拟存储管理子系统,通过修改 page 结构来标识不同大小的超页的方法是不可行的.而页表项与基本页是一一对应的,我们可以选择页表项的最高位作为超页大小的标志位.为提高系统性能和内存利用率,几乎所有的超页系统都需要有自动降解超页的能力.当超页中的某一基本页的访问位等保护权限被改变时也需要降解超页.我们采用的是一种渐进式的降解方式,该方式尽可能保留较大的页面.为了简化页面降解的难度,组成超页的基本页仍然占有单独的页表项.这样不但不需要修改页表查找方式,同时简化了超页降解和提升的过程(只需要修改页表项的页大小标志和对应的 TLB 表项即可).

由 2.1 节可知,龙芯处理器上可支持的页大小为 4 KB,16 KB,64 KB,256 KB,1 MB,4 MB 和 16 MB.而由于要解决虚地址索引的 cache 别名的问题,在采用 4 KB 页大小时需要操作系统支持页着色算法或者动态检测 cache 别名出现的情况,并在出现别名问题时通过内核虚地址将 cache 中的内容写回并置无效[8].这两种解决 cache 别名的算法都会降低系统的性能.第 3 种方法是通过增大页大小到 16 KB.这种方法简洁且对于多数应用程序而言,其 TLB 命中率高于页大小为 4 KB 的系统.所以系统一般采用大小为 16 KB 的内存页.为简化超页支持后 TLB 失效例外处理过程,系统目前仅支持 3 种页大小,大小分别为 16 KB,64 KB 和 256 KB.用页表项的高位作为页大小的标志位,可以从有符号数的符号位快捷地判断该页面是否为超页[7].标志位 10 和 11 则分别

代表页大小为 64 KB 和 256 KB 的超页. 图 1 为超页系统的页表项结构:

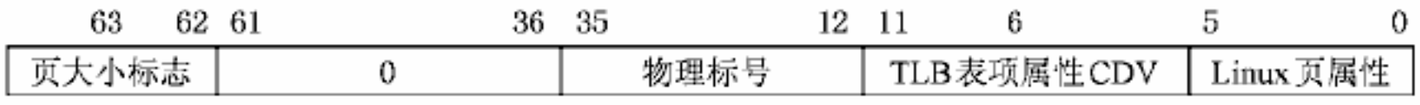

图 1 超页系统的页表项结构

3.2 龙芯处理器上的超页分配策略

龙芯处理器采用的是一种类基于保留的超页分配策略,程序在建立线性空间时根据请求的内存对象的大小尽可能分配大页(大页的大小为不超过内存对象大小的页的最大值)以达到按需分配的目的. 超页中某一基本页缺页失效时,根据其所属内存对象的大小保留超页区域,并将超页的页大小标志填到对应的页表项中. 当页面分配时,尽可能分配所需超页的大小,如果不成功则通过逐步降解的方式直到页面分配成功. 这种超页分配策略并不像真正的基于保留的超页分配策略一样维护一个 reserved 链表,而是在线性空间的建立时预定超页的大小;在超页分配时,尽量分配其对应页表项中所标志的超页. 分配成功后通过修改相应的页表项和 TLB 表项完成超页的直接提升. 虽然这种方式可能会增加系统 IO 的负担,但由于在真实的系统中我们采取的很多简化方法,并且省去了维护额外的 reserved 链表的开销,系统的实际性能还是有较大的提升的.

4 龙芯处理器上的软 TLB 技术

4.1 软 TLB 技术的原理

软 TLB 技术主要通过减少 TLB 失效例外处理时查询页表造成的访存开销来优化 TLB 失效例外处理性能. 以此为目的可以从两个不同的角度来定义软 TLB,这其实也是两种不同的优化思路.

借鉴 Bala 等人提出的软 TLB 技术,通过在系统空间中建立全局的软 TLB 表来缓存页表中更多的表项. 由于是软件结构不受硬件资源及芯片面积和时钟频率的限制,该软 TLB 表可以远大于硬件 TLB,从而容纳比硬件 TLB 多得多的表项. 这样软 TLB 表可以看做软件实现的二级 TLB 表.

进程切换频繁时,为减少进程间软 TLB 冲突,可以把软 TLB 做成一级页表的形式. 软 TLB 的命中率跟硬件 TLB 一样主要取决于软 TLB 表的表项数及其相联度. 用户程序通常最大覆盖 2 GB 的地址空间,软 TLB 表大小为 64 KB 项(系统页大小为 16 KB,软 TLB 表项和硬件 TLB 相同以便于软硬件 TLB 的一致性维护,软 TLB 表项为 16 B),1 MB 大小. 系统初始化时,建立 256 个 1 MB 大小的软 TLB 表. 这样就避免可同时共享硬件 TLB 的 256 个进程对软 TLB 表的竞争. 但该方法一次性静态分配的内存空间太大,造成内存不足而严重影响性能. 可以选择在进程切换时,动态为每个进程分配一个 1 MB 的软 TLB 表. 该方法的优点是减少了进程间对软 TLB 的竞争,并且可以通过结合超页技术加快软 TLB 表的分配和查询. 并可以将映射软 TLB 表本身的超页 TLB 表项通过 wired 寄存器锁定在 TLB 中,以防止查询软 TLB 表引发的级联 TLB 失效,以此来加快对软 TLB 表的访问.

目前龙芯处理器的系统中采用的是二级 TLB 形式的软 TLB 结构. 龙芯处理器的 64 位 Linux 操作系统采用三级页表,且其硬件 TLB 按奇偶表项成对组织,每个 TLB 表项中可以缓存两个连续的页表项,则每次 TLB 失效例外处理需要 5 次访存. 而这 5 次访存中很可能产生 2 到 4 次 L1cache 失效. 这些访存指令造成了 TLB 失效例外处理的主要开销. 软 TLB 作为硬件 TLB 的缓存,其表项构成和硬件 TLB 相同,以便于软硬件 TLB 的一致性维护及简化 TLB 失效例外处理. TLB 失效时首先查找软 TLB,若命中则取出软 TLB 的表项并将其回填到硬件 TLB 中. 该过程仅需要 3 次访存,且该访存访问的是一个软 TLB 表项,访存地址连续;软 TLB 表项大小为 16 B,一个 cache 行的大小为 32 B,因此这 3 次方存最多可引发一次 cache 失效. 由此可见,若软 TLB 命中将会大大降低 TLB 失效例外的开销. 但软 TLB 不命中时,仍然需要查询三级页表并更新软 TLB 表. 反而增大了 TLB 失效例外的开销. 故软 TLB 对性能的优化力度取决于软 TLB 的命中率. 在统计 TLB 失效例外开销时可只考虑访存指令和 cache 失效产生的开销,经计算软 TLB 的命中率必须要达到 50%以上才能得到好处[7].

软 TLB 的命中率跟硬件 TLB 一样,主要取决于软 TLB 的表项数及其相联度. 如图 2 所示,软 TLB 表项数相同,相联度与软 TLB 的命中率成正比. 但相联度越高,查询软 TLB 表的过程越复杂,TLB 失效开销就会变大. 因此,在存储资源越来越充足的现代处理器中,可以选择增大软 TLB 表项数

来增大软 TLB 的命中率.实验表明,软 TLB 表项达到 4K(K=1024)项时,不管采用何种组织结构对多数应用程序,软 TLB 的命中率都已经达到 90%以上,实验结果见图 2,图 3.因此,在龙芯处理器系统中,采用 4K 项,直接映射的软 TLB 结构.

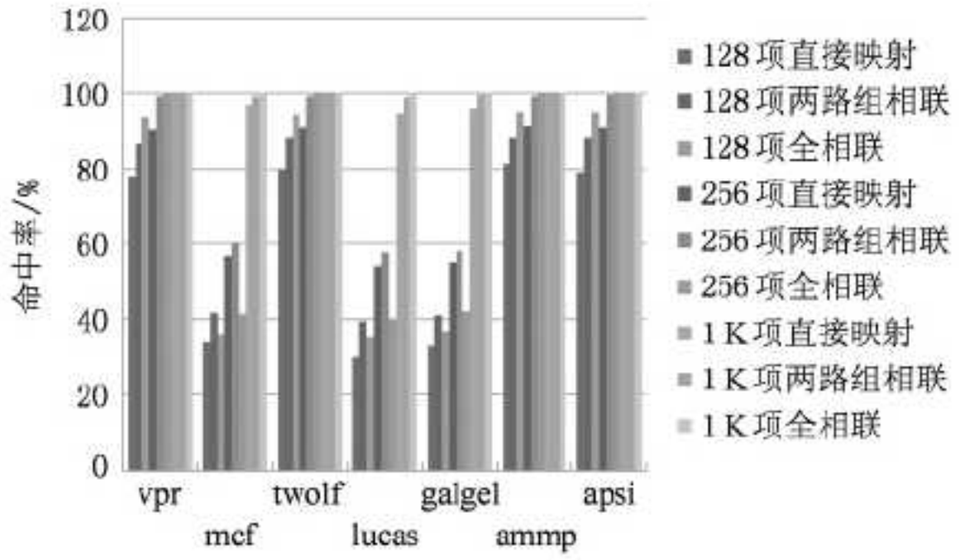

图 2 软 TLB 相联度及大小与其命中率的关系

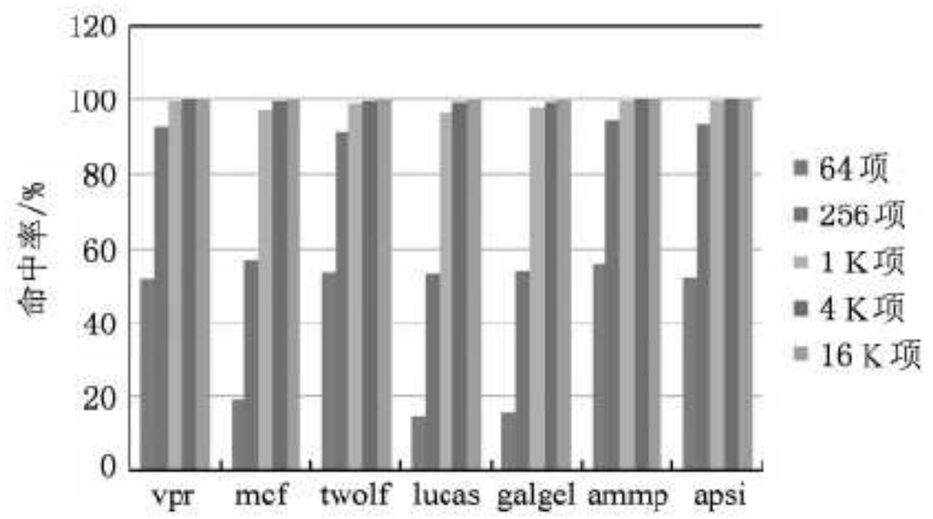

图 3 软 TLB 大小与其命中率的关系

4.2 锁 L2cache 对软 TLB 的优化

龙芯处理器采用二级 TLB 模式的软 TLB 结构,该方法简单易实现.且龙芯 3 号处理器增加了锁 L2cache 的功能,我们利用该功能进一步优化了软 TLB 的性能.龙芯 3 号处理器的 L1cache 和 L2cache 均采用随机替换算法.但 L2cache 提供了锁机制,该机制是通过物理地址及偏移量将要锁住的区域锁进 L2cache,被锁住的内容将不会被替换出去.锁 L2cache 机制需要知道要锁区域的物理地址,这对于应用程序而言是困难的.应用程序映射的物理页面是操作系统动态任意分配的,所以该机制应用于应用程序较难实现而应用于内核的数据结构却非常简单.软 TLB 表是一个全局缓冲区,我们把它放在了 unmapped cached 的 kseg0 区域,避免 TLB 失效例外时访问软 TLB 表可能会引发的级联 TLB 失效.同时,可以直接得到其映射的物理地址.软 TLB 表的大小只有 64 KB,符合锁 cache 对于所锁大小的要求且占整个二级 cache 的比例较小,从而不会对其他程序性能产生大的影响.在得知软 TLB 表的起始物理地址及其大小后,就可以很容易地将软 TLB 表锁进 L2cache.被锁住的软 TLB 表不会被替换出二级 cache,在 TLB 失效例外过程中就不会因为软 TLB 自身产生访存失效,降低了 TLB 失效开销.

5 性能测试结果

性能测试选取的测试程序集是 SPECCPU2000,该测试程序集包括测试系统定点和浮点性能的两套测试程序集 CIN2000 和 CFP2000.它们主要针对处理器运算能力和内存系统性能的评测,是评价处理器综合性能的标准测试程序集.超页技术和软 TLB 技术都是通过优化 TLB 性能来优化系统性能的,因此我们只选择了 SPECCPU2000 中 TLB 失效频繁的一组测试程序.这些程序占了 SPECCPU2000 测试集的 1/4,它们分别是 ammp,apsi,galgel,lucas,mcf,twolf 和 vpr[9].

为测试超页技术对系统性能的影响,我们同时测试了页大小分别为 16 KB 和 64 KB 系统(因为 4 KB 系统还需要解决 cache 别名的问题而需要额外的操作系统的支持,性能较差.因此,系统一般不采用 4 KB 的页大小,我们也不对其做比较)对以上 SPEC2000 测试程序的性能,并以 16 KB 系统为基准对它们进行了比较.性能测试结果如图 4 所示,页大小由 16KB 增大的 64 KB 及超页系统性能都有所提高,超页系统中 vpr,mcf 和 apsi 等 TLB 失效严重的程序性能提升达到 20%左右,mcf 的性能提升达到 29%.

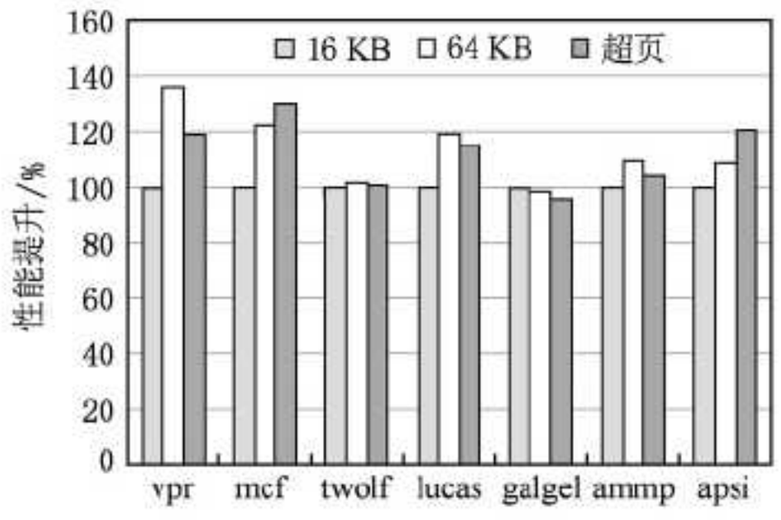

图 4 不同页大小系统对 SPEC2000 测试性能比较

软 TLB 及经锁 L2cache 优化了的软 TLB 的性能测试结果如图 5 所示,对 SPEC2000 中 TLB 失效较为严重的测试程序,如 vortex,vpr 等,软 TLB 性能提升分别达到了 6.24%和 5.99%;软 TLB& 锁 L2cache 的性能提升分别达到了 6.99%和 6.27%.

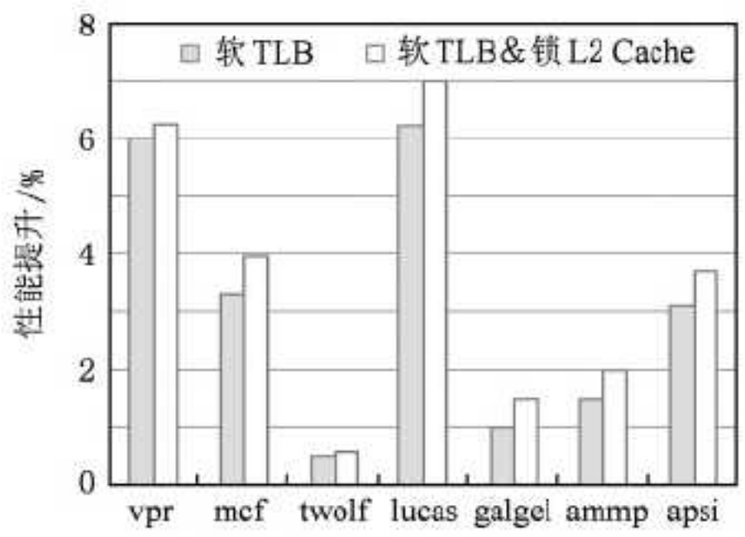

图 5 软 TLB 对 SPEC2000 测试程序的性能

6 总 结

本文总结了龙芯处理器上实现的两种 TLB 系统性能优化技术:超页技术和软 TLB 技术,这两种技术从不同的角度充分利用了龙芯处理器提供的硬件支持,如超页支持和锁二级 cache 支持,挖掘了 TLB 性能优化空间.其中超页技术以不增大 TLB 为前提提高了 TLB 的覆盖率,并根据程序的需求动态分配大页,避免了系统使用大页带来的 IO 系统及文件系统的性能损失.超页技术是 TLB 性能优化的重要方法之一,该方法涉及到操作系统内存管理的物理页面分配算法、页面替换算法、换入换出子系统、文件子系统及 IO 子系统.该方法虽然复杂但却提供了更多优化空间.采用二级 TLB 模式的软 TLB 技术简单易实现,存储资源充足的情况下,可以通过增大其大小来提高其命中率.同时结合龙芯 3 号处理器的锁二级 cache 功能,该方法得到了进一步的优化,带来了更大的性能提升.并且伴随着多核时代的到来,TLB 行为新特性随之而来[10].我们可以利用软 TLB 技术提供的全局缓存表来减少核间冗余的 TLB 失效,从而进一步优化多核处理器的系统性能.

参 考 文 献

[1] Romer T H, Ohlrich W H, Karlin A R, et al. Reducing TLB and memory overhead using online superpage promotion //Proc of the 22nd Annual Int Symp on Computer Architecture. New York: ACM, 1995: 176-187

[2] Talluri M, Hill M D. Surpassing the TLB performance of superpages with less operating system support //Proc of the 6th Int Conf on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 1994: 171-182

[3] Navarro J, Iyer S, Druschel P, et al. Practical, transparent operating system support for superpages //Proc of the 5th Symp on Operating Systems Design and Implementation. New York: ACM, 2002: 89-104

[4] Swanson Mark R, Stoller Leigh, Carter John B. Increasing tlb reach using superpages backed by shadow memory //Proc of the 25th Annual Int Symp on Computer Architecture. New York: ACM, 1998: 204-213

[5] 许先超. 减少 TLB 失效开销提高 64 位 Linux 系统性能的方法. 计算机工程, 2006, 32(2): 70-72

[6] Bala K, Kaashoek M F, Weihl W E. Software prefetching and caching for translation lookaside buffers //Proc of the Usenix Symp on Operating Systems Design and Implementation. Berkeley, CA: USENIX, 1994: 243-253

[7] 林伟. Linux 内存管理子系统在龙芯 2 上的优化. 北京: 中国科学院计算技术研究所, 2005

[8] 伍鸣, 张福, 林伟, 等. 龙芯 2 号处理器系统优化关键技术. 计算机研究与发展, 2006, 43(6): 980-986

[9] Kandiraju G B, Sivasubramaniam A. Characterizing the d-TLB behavior of SPEC CPU2000 benchmarks //Proc of the ACM SIGMETRICS Conf on Measurement and Modeling of Computer Systems. New York: ACM, 2002: 129-139

[10] Bhattacharjee A, Martonosi M. Inter-core cooperative tlb for chip multiprocessors //Proc of the 15th Edition of ASPLOS on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2010: 359-370

张晓辉 女,1984 年生,博士研究生,主要研究方向为计算机体系结构、性能评估和操作系统.

程归鹏 男,1984 年生,硕士,主要研究方向为性能评估和操作系统.

从 明 女,1983 年生,博士,主要研究方向为计算机体系结构和系统性能评估.