PMESI:一种优化进程私有数据访问的缓存一致性协议*

王绍刚,徐炜遐,庞征斌,吴 丹,戴 艺,陆平静
(国防科技大学 计算机学院,湖南 长沙 410073)

摘 要:并行应用程序中绝大部分的访存是对私有数据的访问,在cache一致性协议上不会产生冲突。传统一致性协议没有根据程序私有数据的访问模式进行针对性设计,存在着很大的优化空间。针对以上的问题,提出了一种支持私有状态的cache一致性协议PMESI,通过动态关闭和激活内存空间的cache一致性目录,优化私有内存空间的访问延迟和功耗。通过时钟精确模拟器的测试,PMESI协议优化了程序中54%的访存,并行程序的执行时间平均缩短了9%。

关键词:PMESI协议;私有内存空间;目录协议;操作系统
中图分类号:TP316 **文献标志码**:A **文章编号**:1001-2486(2013)01-0097-06

PMESI: optimizing memory performance by selectively deactivating cache coherence for private pages

WANG Shaogang, XU Weixia, PANG Zhengbin, WU Dan, DAI Yi, LU Pingjing
(College of Computer, National University of Defense Technology, Changsha 410073, China)

Abstract: Parallel program has significant percentage of memory requests that target only private data, which does not need to resolve cache coherence conflicts. Yet traditional coherence protocol does not distinguish between shared and private blocks, which leaves much optimization space. An optimized cache coherence protocol, called PMESI, which dynamically deactivates coherence maintenance for private memory space, was suggested. PMESI achieves two distinguishing features: the reduction of memory access latency and system power consumption. Simulation results on the cycle accurate simulator show that 54% memory references can be efficiently optimized and the program execution time is reduced 9% on average.

Key words: PMESI protocol; private memory space; directory coherence protocol; operating system

目前,高性能多路服务器大都是基于商用处理器直连技术构建的SMP系统,通过对大量应用程序的统计发现,处理器发出的访存请求中,绝大多数是对进程私有地址空间的访问。文献[2]对多个并行应用程序的访存行为进行统计,从已有的研究结果可以看出,对进程私有空间的访问是存储系统最主要的访存模式,平均比例达到了75%。

对私有内存空间进行cache一致性协议的优化可以带来很多好处:(1)由于私有空间的访存不会产生冲突,因而在协议处理上,可以旁路协议目录,降低访存延迟;(2)目录cache中可以腾出更多空间存放共享数据的目录,降低了目录cache的失效率[11-12];(3)有利于优化协议事务报文的数量、减少目录存储器访问次数、降低存储系统功耗。

1 系统架构

本文提出的方案主要面向高性能多路服务器系统,系统支持全局共享主存,通过处理器直连协议实现全系统cache一致性。一种典型的系统架构如图1所示,4个处理器之间通过两两的直连通路,形成全互联结构。处理器集成存控,可直接挂接片外DRAM存储器。

在该架构下,处理器间以及处理器核内采用基于目录的协议维护cache一致性,目录与内存数据共同保存在片外DRAM中。处理器访问主存数据时,首先需要访问片外DRAM中的目录,目录中维护了内存数据最新副本位置的信息,一致性协议通过协议事务保证返回给处理器的数据是最新的。

在传统目录协议流程中,在大多数情况下(没有cache-to-cache的转发),处理器的一次访存操作需要访问两次片外DRAM存储器,第一次访问目录以获取最新副本位置并更新目录,第二次才是获取访存数据。而对于进程私有数据,程序行为决定了不会出现访问冲突的情况,对一致性协议目录的访问显然是不必要的。

* **收稿日期**:2012-03-17
基金项目:国家自然科学基金资助项目(60803040,61003301);国家863计划项目(2012AA01A301)
作者简介:王绍刚(1979—),男,山东烟台人,助理研究员,博士,E-mail:wshaogang@nudt.edu.cn

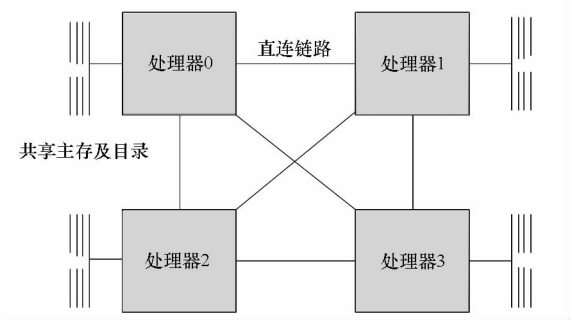


图1 4处理器直连SMP系统架构

Fig. 1 Baseline SMP architecture with 4-socket direct connection

2 支持私有状态的cache一致性协议

为了有效利用进程独占访问私有内存空间的特性,本文在传统MESI协议的基础上,增加了新的状态P,在P态下,cache块是进程私有数据,拥有者对该数据块独占访问,且不会产生访问冲突。内存初始化时,cache一致性协议将所有的内存块的初始状态置为P,也就是目录的状态初始化为P。

正常的私有数据访问时,PMESI协议并不需要访问目录。目录状态为P时,只是在内存的访问模式切换时才可能被访问。当内存块的访问模式(私有或共享)切换时,协议支持P状态与其他共享状态(MESI)平滑的转换。

2.1 访存模式检测

本文以内存页为单位检测内存的访问模式(共享或私有)。在操作系统内核中,管理内存物理页面的数据结构中包含共享计数的变量,表示当前有几个进程的虚拟地址空间映射到该物理页面,因而可以通过共享计数的值来确定页面当前的访问模式[4-5]。当共享计数为1时,表明该页面是某个进程的私有数据;共享计数大于1时,表明页面由多个进程共享。

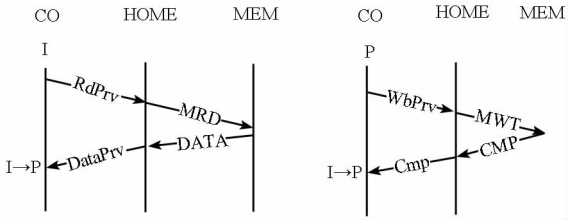
在访问内存之前,进程中使用虚拟地址需要通过操作系统变换为物理地址,地址转换是通过虚实转换页表进行的。为了加速转换,处理器的TLB(Translate Look-aside Buffer)中缓存了最近的页表项[10]。为了检测处理器的访存模式,本文的方案为每个虚实转换表项增设一个标志位P,表示对应的页面是否为私有访问模式,操作系统在物理内存页的共享计数变化时更新P标志。处理器访存请求在进行虚实地址转换的同时即可确定其模式。

为了处理访存模式切换时可能产生的冲突,页表项中增加标志位B,表示当前页面是否处于模式转换过程中。当B置1时,PMESI协议需要借助目录来处理私有的访存事务。

2.2 私有数据无冲突访问协议流程

处理器的访存请求在进行虚实地址转换时,通过TLB表项中的P标志位来确定访存模式。如果是私有数据访问模式(P=1),且页面当前不处于模式转换流程中(B=0),则处理器通过RdPrv报文命令访问存储器,如图2(a)所示。HOME(存控)接收到RdPrv命令后,不需要访问目录,而直接返回存储器中的数据。RdPrv命令的响应数据返回给cache后,将该cache块置为P状态。当前cache作为数据块的拥有者,具有独占的访问权。

当P状态的cache块需要替换时,如果未被修改,则cache可以直接丢弃;如果已经被修改,则修改后的数据需要写回存储器。为此,PMESI新增了P状态下的写回事务WbPrv。HOME对WbPrv命令的处理与RdPrv命令类似,不需要访问目录,而直接将数据写回存储器,如图2(b)所示。



(a) 私有数据读协议流程 (b) 私有数据写协议流程

图2 无冲突情况下私有数据读写协议流程

Fig. 2 Conflict free private memory RD/WT flow

2.3 内存页访问模式转换

在进程的生命周期中,物理页面的访问模式可能发生变化,因而需要将页面的cache一致性协议切换到对应的模式上。PMESI协议支持私有与共享两种模式之间的切换。在进程执行期间,申请和释放内存页时导致共享计数的变化,即访问模式的切换时,操作系统为页面的每个cache块启动相应的一致性协议事务,将该块的协议处理模式切换到对应的状态上。

2.3.1 内存页私有态向共享态的转换

假设处理器核C0上执行的进程P0正在私有访问页A,另一个处理器核C1上执行的进程P1也申请访问页面A时,P1的系统调用使C1进入操作系统内核服务程序。内核检测到页面A需要转为共享访问模式,启动如下的操作:

Step1:C1上执行的操作系统内核锁定页表数据结构,向系统广播PurgeTC报文,失效C0中TLB缓存中的页表项,并更新虚实转换页表,清除页表项中的P标志位,调用顺序如图3所示。

Step2:对页面 A 中的每个 cache 块,P1 的内核服务程序启动 InvPrv 事务,将 C0 中私有数据写回存储器,协议流程如图 4 所示。待页面 A 中所有 cache 块的 InvPrv 事务完成后,将页表解锁。

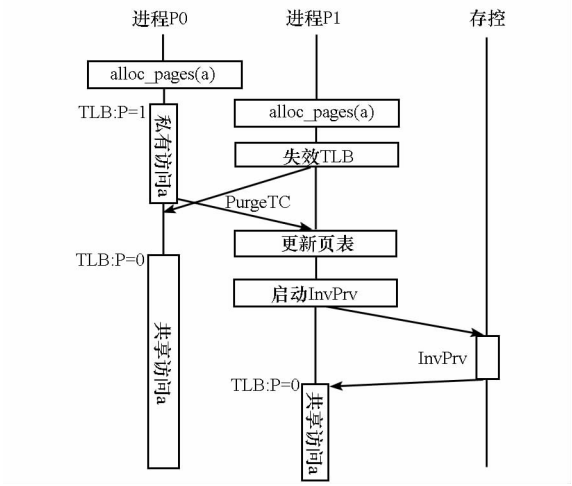


图 3　私有页转共享页流程

Fig. 3　Processing flow of private-to-shared mode transformation

InvPrv 报文由 C1 的 cache 发给 HOME,HOME 接收到 InvPrv 报文后,查询协议目录:

(1)如果目录的状态为 P,则向所有 cache 广播 IntInvPrv 报文,IntInvPrv 报文的响应报文(RspI, RspP)携带了数据块在目的 cache 中的状态,HOME 根据响应报文的信息更新目录状态和所有者指针;

(2)如果 HOME 目录的状态不为 P,则立即返回 Cmp 响应,不作处理。

InvPrv 事务的主要目的是将系统 cache 中私有数据的状态更新为对应的共享态(E、S 或 I),并更新目录使其记录数据最新副本的位置。InvPrv 事务执行完毕后,以后共享访存事务的处理便切换到传统的 MESI 协议上完成。

在页面由私有态向共享态转换的过程中,可能会出现以下几种情况:

Case1:进程 P0 的 RdShrd 事务对应的目录状态为 P,目录将 RdShrd 事务广播给所有 cache,包含有 P 态数据的 cache 将私有数据写回或作废。存控收齐了所有 cache 的响应后,返回 RdShrd 事务的响应,并将目录状态更新为共享态。

Case2:如果进程 P1 启动的 InvPrv 事务未完成,进程 P0 发出了对同一页面的访存请求(由于页表项已经被 P1 进程的系统调用修改为共享模式,TLB 将重新装载修改过的页表项,因而 P0 的请求将被检测为共享访存模式),所请求的数据可能在 C0 的 cache 中仍然以 P 状态存在,导致共享访存请求命中私有缓存数据的情况。

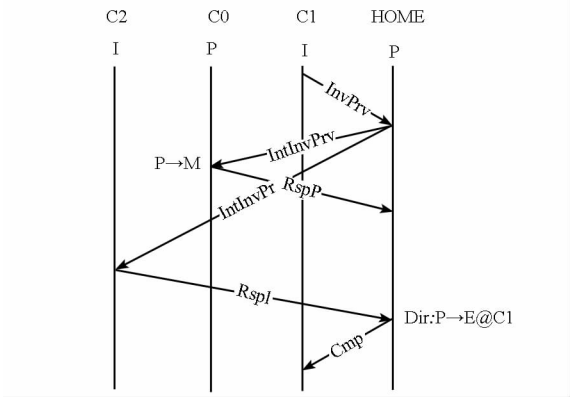
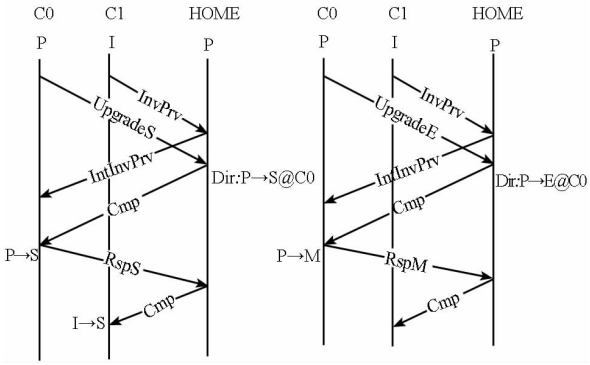


图 4　cache 一致性协议 P→E 状态转换流程

Fig. 4　Protocol flow for P→E state transformation

为了处理这种情况,PMESI 协议引入了进程 P0 主动升级私有状态的协议事务(P→S 或 P→E),协议的流程如图 5 所示。当处理器的共享访存请求命中缓存 P 态的数据时,cache 向 HOME 发出 UpdateS 或 UpdateE 报文,先通知 HOME 将目录的状态更新为 S 或 E 状态,响应返回后,再将数据块在 cache 中的状态升级为 S 或 E,并向处理器返回请求的数据。

Case3:RdShrd/UpgradeS/UpgradeE 事务与 InvPrv 事务冲突:cache 在发出 RdShrd/ UpgradeS/UpgradeE 请求但还未收到响应时,先阻塞 HOME 广播的 IntInvPrv 请求。待事务的响应返回后,根据数据块新的状态,cache 向 HOME 发出 InvInvPrv 请求的响应报文,完成 IntInvPrv 事务,协议流程如图 5 所示。



(a)　　(b)

图 5　Upgrade 事务与 InvPrv 事务冲突协议处理流程

Fig. 5　Protocol flow to handle the upgrade and invprv conflict

2.3.2　内存页共享态向私有态的转换

当物理页面的共享计数减为 1 时,操作系统可以根据页面的访问模式,将该页转入了私有访问模式,或者保留共享模式。当页面转为私有访问模式时,需要将该页面的一致性协议处理模式

重新转入私有模式,具体的流程如下:

Step1:操作系统锁定页表项数据结构,启动PurgeTC流程,失效处理器TLB缓冲区中页表项,修改进程的虚实转换页表,置页表项中P和B标志位,调用顺序如图6所示。

Step2:操作系统对该页面的每个内存块启动InvC协议事务,其作用是将该页的数据从cache中清除,并将目录重置为P状态。如果数据块在cache中为脏状态,则通过响应报文将脏数据写回存储器。待页面包含的所有InvC事务完成后,将页表解锁。

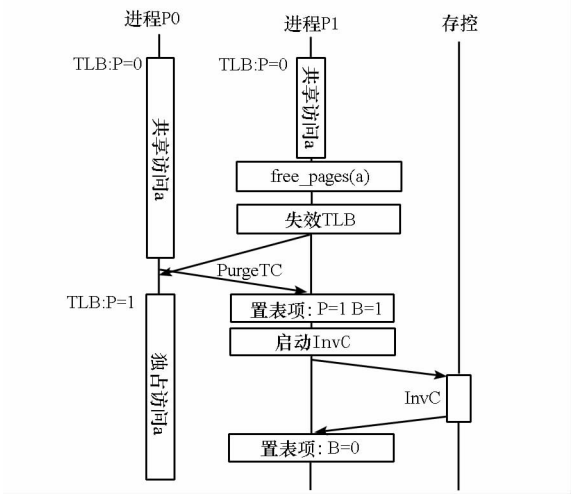


图6 共享页转私有页流程

Fig. 6 Processing flow of shared-to-private mode transformation

InvC事务的协议流程如图7所示。

InvC事务的主要目的是将系统cache中共享态(M、E或S)数据失效并写回内存,并将目录重新切换到私有状态。InvC事务执行完毕后,以后私有访存事务不需要依赖目录完成。

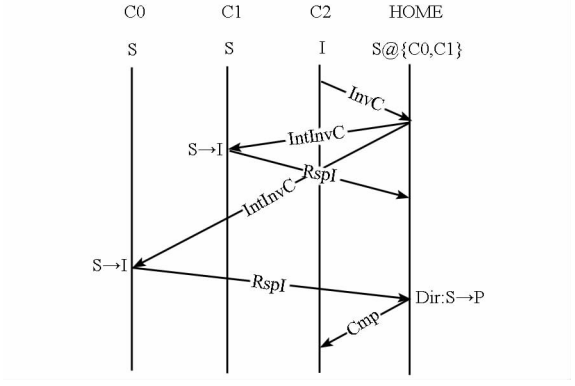
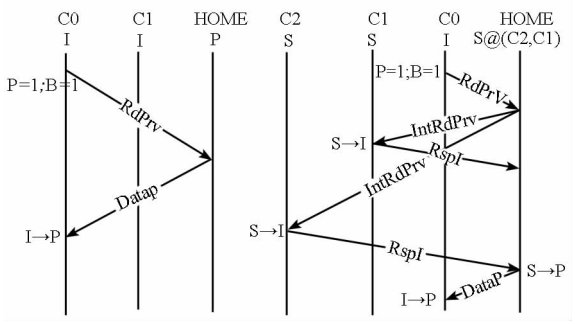


图7 cache一致性协议恢复P态流程

Fig. 7 Protocol flow to restore the p state

内存页由共享态转为私有态时,边界情况的处理方式如下:

Case1:页表项P和B标志同时有效时,私有访存事务需要借助目录完成。RdPrv事务到达存控时,目录的状态可能为私有态(P)或共享态(S,E)。当目录为P态时,存控直接返回请求数据,如图8所示;当目录为S或E态时,则存控根据目录信息,先将数据从系统cache中作废或写回,待cache无数据副本时,返回RdPrv事务的响应,并将目录更新为P态,如图8(b)所示。



(a) 目录状态为P (b)目录状态为S

图8 共享态转为私有态时,RdPrv事务处理流程

Fig. 8 Rdprv protocol flow to handle the shared-to-private transformation

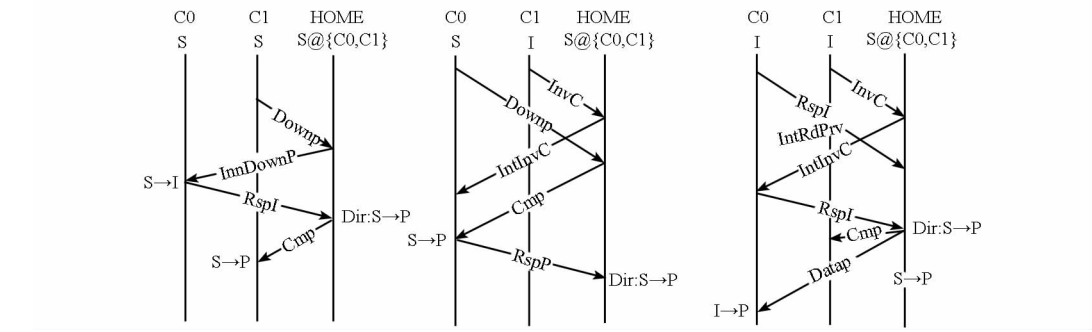
Case2:在图8的例子中,P1启动的InvC事务未完成时,有可能出现进程P0发出了私有访存请求,命中缓存S态的情况(InvC来不及失效P0缓存中的S态数据)。PMESI引入了私有数据拥有者进程主动将共享态(S)降级为P态的协议事务,如图9(a)所示,缓存C1发出DownP报文将S态的数据降级为P态,该事务将目录状态重置为P。

Case3:DownP事务有可能与InvC事务冲突,在这种情况下,发出DownP事务的cache将阻塞HOME发出的IntInvC报文,待DownP事务完成后,再失效cache中的数据,并返回响应(RspP),如图9(b)所示。

Case4:InvC事务与进程的私有访存请求RdPrv事务冲突,在这种情况下,缓存私有数据的cache无条件返回RspI响应,如图9(c)所示。

2.3.3 停顿处理策略

上述的方案在处理共享转私有或私有转共享的情景时,要解决操作系统事务与进程事务间的冲突,需要协议提供一定的支持。另一种简单的处理方式是:操作系统通过PurgeTC将系统中TLB表项作废后,先将该虚实转换的页表项置为忙状态。硬件在进行虚实转换时,如果页表项为忙状态,则停顿访存请求。当整个页面所有块的InvPrv或InvC事务完成后,再将页表项转换为正常状态。通过以上的方式,避免了InvPrv事务与InvC事务冲突的情况,有利于简化协议的实现,



(a)私有 cade 主动降级 p 状态　(b) Downp 事务与 InvC 事务冲突　(c) Rdprv 事务与 InvC 事务冲突

图 9　内存页由共享态转为私有态时冲突事务的协议处理流程

Fig. 9　Conflicting protocol flow to handle the private-to-shared mode transformation

但会由于停顿造成性能损失。

3　性能评测

本文将 PMESI 协议实现在 GEMS 模拟器[6-7]的存储子系统子中,并在模拟器中执行了 SPLASH2[9]和 PARSEC[8]测试程序集。模拟器参数如表 1 所示。

表 1　GEMS 模拟器参数

Tab. 1　GEMS simulation parameters

处理器	UltraSparc-III-plus, 单发射,顺序执行,4 处理器直连
内存	每个处理器 1GB 共享主存,本地 100 周期访存延迟,远程 200 周期访存延迟
Cache	两级 cache,1 级 cache 512KB,4 路组相连,3 周期延迟;2 级 cache 2MB,8 路组相连,10 周期延迟

3.1　Cache 数据状态统计

本文对 cache 中的 P 态块的比例做了统计,统计方法是在程序执行时间内,每隔 1000 个时钟周期统计一次 P 态 cache 块占所有 cache 块的比例,在程序执行完成后取平均值,测试结果如图 10 所示。

通过统计 P 态 cache 块的比例,大致可以了解程序中可加速私有访存占访存总数的比例。从统计结果来看,程序的访存事务中,有很大的比例是对私有数据的访问,平均比例达 54%。这项测试结果验证了本文研究的出发点,即对程序私有数据的访问存在着很大的性能优化空间,并且说明本文的策略能够针对私有数据进行有效的优化。

在本文的策略中,操作系统以页面为单位来检测访存模式,检测粒度大于 cache 块的粒度,因而存在伪共享的情况,即一个内存页中部分 cache 块是共享的,但在实现上仍然将该页以共享模式处理。根据 Cuesta 等人的统计,程序中对私有空间的访存比例达到了 75%[2]。本文的策略大致加速效果较明显,程序中 72% 的私有空间数据访问得到了优化。

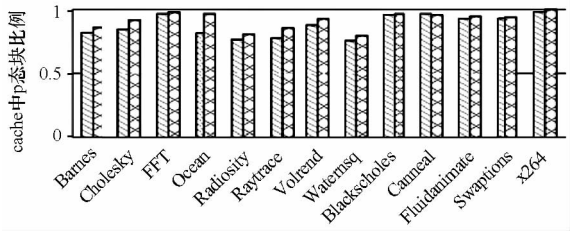


图 10　cache 中 P 态块占所有 cache 块的比例

Fig. 10　The percentage of p state blocks

3.2　程序执行时间

为了测试 PMESI 优化策略对程序性能的影响,本文对优化方案的执行时间进行了统计。优化方案分别采用停顿策略和非停顿策略来处理进程内存页的状态转换,并与程序在未优化系统上的执行时间对比,测试结果如图 11 所示。

PMESI 方案对程序执行时间的优化体现在以下几个方面:(1)由于旁路了目录,私有空间的访存延迟减少;(2)目录 cache 中不再保存私有数据的目录,因而能够腾出更多的空间保存共享数据的目录,减少了目录 cache 的失效;(3)在传统 MESI 协议下,系统中存在着很多伪共享的情况,处理伪共享带来了一定的延迟。

从测试结果来看,本文的方案具有明显的性能优势,与程序在未优化系统上的执行时间相比,在非停顿策略下,程序平均执行时间减少了 9%;而在停顿策略下,平均执行时间减少了 7.4%。此外,从停顿策略与非停顿策略的性能对比来看,二者的差距不明显。在 Ocean 测试程序上,两种策略的性能差距最大,为 4%;而在大多数测试程序上,两种策略的性能差距基本在 2% 以内。这说明采用简单的停顿策略就能获得很好的性能提升,停顿策略的实现并不需要对软硬件进行复杂的修改。

3.3　协议可扩展性测试

PMESI 协议具有很好的扩展性,当系统的

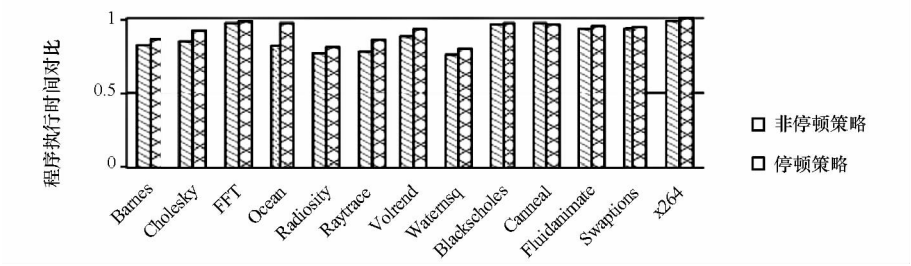


图 11 程序执行时间对比

Fig. 11 Benchmark execution time

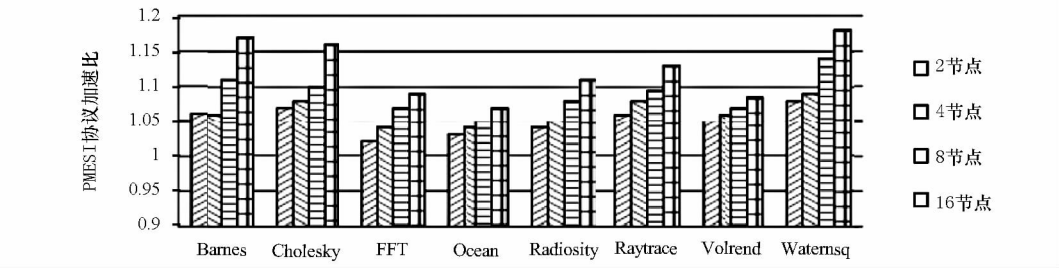


图 12 PMESI 协议可扩展性测试

Fig. 12 Scalability test result of the PMESI protocol

cache 数目增多时,私有数据的访存延迟并不受影响。在传统 MESI 协议下,伪共享的协议处理开销随着 cache 数目的增多而增大,可扩展性不好。为了测试 PMESI 协议的可扩展性,本文测试了在不同并行规模下,PMESI 协议相对于传统 MESI 协议的加速比,实现的传统 MESI 协议采用了 4 位共享向量的方式记录共享者指针,测试结果如图 12 所示。

从测试结果来看,PMESI 协议相对于传统 MESI 协议的性能加速比随着节点数目的增加而增大,在 16 个节点规模下,平均加速比为 12%,说明对私有数据的访存优化能够有效提升协议的可扩展性。

4 结束语

本文针对进程对私有数据具有独占访问的特点,在 cache 一致性协议层面进行了相关的性能优化工作。通过在 cache 一致性协议中引入私有状态 P,以及相关的协议流程,结合操作系统的简单支持,实现了进程私有数据的访存不需要经由目录的优化目标,因而有效降低了访存延迟,降低了系统功耗,协议实现简单。从测试的结果上来看,方案取得了很好的性能加速效果,说明进程私有数据的访存加速是有效的存储系统性能优化方向,对目前全局共享存储系统的设计具有很好的借鉴意义。

参考文献(References)

[1] Conway P, Kalyanasundharam N, Donley G, et al. Cache hierarchy and memory subsystem of the AMD opteron processor [J]. IEEE Micro, 2010,30(2):16 - 29.

[2] Blas C, Alberto R, María E G, et al. Increasing the effectiveness of directory caches by deactivating coherence for private memory blocks [C]//38th Int'l Symp. on Computer Architecture (ISCA),2011.

[3] Shah M, Barreh J, Brooks J, et al. UltraSPARC T2: A highly-threaded, power-efficient SPARC SoC[C]//IEEE Asian Solid-State Circuits Conference, 2007.

[4] Hardavellas N, Ferdman M, Falsafi B, et al Reactive NUCA: Near-optimal block placement and replication in distributed caches [C]//36th Int Symp. on Computer Architecture (ISCA), 2009.

[5] Kim D, Ahn J, Kim J, et al. Subspace snooping: Filtering snoops with operating system support[C]//19th Int Conference on Parallel Architectures and Compilation Techniques (PACT), 2010.

[6] Magnusson P S, Christensson M, Eskilson J, et al. Simics: A full system simulation platform[J]. IEEE Computer, 2002,35(2):50 - 58.

[7] Martin M M, Sorin D J, Beckmann B M, et al. Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset[J]. Computer Architecture News, 2005,33(4):92 - 99.

[8] Bienia C, Kumar S, Singh J P, et al. The PARSEC benchmark suite: characterization and architectural implications [C]//17th Int Conference on Parallel Architectures and Compilation Techniques (PACT), 2008.

[9] Woo S C, Ohara M, Torrie E, et al. The SPLASH-2 programs: Characterization and methodological considerations[C]//22nd Int Symp. on Computer Architecture (ISCA), 1995.

[10] Ekman M, Dahlgren F, Stenström P. TLB and snoop energy-reduction using virtual caches[C]//Int Symp. on Low Power Electronics and Design (ISLPED), 2002.

[11] Krafka B W, Newton A R. An empirical evaluation of two memory-efficient directory methods[C]//17th Int Symp. on Computer Architecture (ISCA), 1990.

[12] Gupta A, Weber W D, Mowry T C. Reducing memory traffic requirements for scalable directory-based cache coherence schemes[C]//Int Conference on Parallel Processing (ICPP), 1990.

[13] Zebchuk J, Safi E, Moshovos A. A framework for coarse-grain optimizations in the on-chip memory hierarchy. [C]//40th IEEE/ACM Int Symp. On Microarchitecture (MICRO), 2007.