软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2017,28(4):1027−1047 [doi: 10.13328/j.cnki.jos.005245]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

片上多核处理器 Cache 一致性协议优化研究综述*

胡森森, 计卫星, 王一拙, 陈 旭, 付文飞, 石 峰

(北京理工大学 计算机学院 嵌入式高性能计算实验室,北京 100081)

通讯作者: 石峰, E-mail: bitsf@bit.edu.cn

摘 要: 现代晶体管技术在单芯片上集成多个处理器已经成为现实.近年来,随着多核处理器集成核数的不断增加,高速缓存的一致性问题凸显出来,已成为多核处理器的性能瓶颈之一,亟待解决.介绍了片上多核处理器一致性问题的由来.总结了多核时代高速缓存一致性协议设计的关键问题,综述了近年来学术界对一致性的研究.从程序访存行为模式、目录组织结构、一致性粒度、一致性协议流量、目录协议的可扩展性等方面,阐述了近年来缓存一致性协议性能优化的方向.对目前片上多核处理器缓存一致性协议设计中存在的问题进行了讨论,并指出了未来进一步研究的方向.

关键词: 片上多核处理器;缓存一致性协议;性能优化

中图法分类号: TP316

中文引用格式: 胡森森,计卫星,王一拙,陈旭,付文飞,石峰.片上多核处理器 Cache 一致性协议优化研究综述.软件学报,2017,28(4):1027−1047. http://www.jos.org.cn/1000-9825/5245.htm

英文引用格式: Hu SS, Ji WX, Wang YZ, Chen X, Fu WF, Shi F. Survey on cache coherence protocol and performance optimization for chip multi-processor. Ruan Jian Xue Bao/Journal of Software, 2017,28(4):1027−1047 (in Chinese). http://www.jos.org.cn/1000-9825/5245.htm

Survey on Cache Coherence Protocol and Performance Optimization for Chip Multi-Processor

HU Sen-Sen, JI Wei-Xing, WANG Yi-Zhuo, CHEN Xu, FU Wen-Fei, SHI Feng

(High Performance Embedded Computation Laboatory, School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China)

Abstract: Modern-Day transistor technique enables the industry to integrate many cores on a single chip. As an increasing number of cores being integrated on a single chip, cache coherence has become an intractable issue as well as a bottleneck of performance. In this paper, the origin of cache coherence is carefully described. Further, the paper summarizes the key issue of cache coherence and reviews the study in this field a decade after entering the mulit-core era. From aspects of memory access, directory organization, coherence granularity, coherence traffic and scalability, the work on optimization of cache coherence in recent researches is also presented. Finally, the potential challenges in current coherence protocol and direction of future research are discussed.

Key words: chip multi-processor; cache coherence protocol; performance optimization

科学家预计:在未来的十几年中,摩尔定律仍然将统治着半导体技术的发展.然而在最近 10 年的发展中,依赖提高处理器频率的传统方法不再有效,单核处理器的性能提升基本走到了尽头[1].2001 年,IBM Power4[2]的推

* 基金项目: 国家自然科学基金(61300010, 61300011); 中国科学院计算技术研究所计算机体系结构国家重点实验室开放课题(CARCH201404)

Foundation item: National Natural Science Foundation of China (61300010, 61300011); State Key Laboratory of Computer Architecture, Institute of Computing Technology, the Chinese Academy of Sciences (CARCH201404)

收稿时间: 2016-05-23; 修改时间: 2016-08-18; 采用时间: 2016-11-27; jos 在线出版时间: 2017-01-20

CNKI 网络优先出版: 2017-01-20 16:06:36, http://www.cnki.net/kcms/detail/11.2560.TP.20170120.1606.011.html

出,标志着世界上第一个商用多核处理器(multi-core processor)正式诞生.此后,2004 年,Intel 推出了 x86 系列的多款多核处理器.Sun/Oracle 公司也先后推出了 UltraSPARC T1[3]和 T2[4]处理器.2007 年,麻省理工学院和 Tilera 公司合作研制了一款基于瓦片(tile)体系结构的片上多核处理器 Tile64[5].Tile64 片内集成 64 个核心,通过二维网格(2DMesh),将 64 个独立的核连接起来.Tile64 的成功,标志着多核处理器由多片时代进入到片上多处理器(chip multi-processor,简称 CMP)时代.Berkeley2006 年预言的“不久的将来,一个芯片内部就能够集成上百个处理器[6]”,目前已经成为现实.

然而,处理器和存储器之间的速度差异导致“存储墙”问题的出现.在单核处理器中,Cache 存储系统的引入,缓解了处理器和存储器之间的速度不匹配问题.在多核时代,随着并行执行的线程数量的增加,多核处理器的内存访问请求数量激增,重新加剧了“存储墙”问题.总体来说,多核架构的兴起,给存储系统的设计带来了新的挑战,高速缓存一致性问题已成为制约多核处理器性能提升的瓶颈之一,亟待解决.本文针对片上多核处理器高速缓存一致性相关研究进行了综述,总结了近 10 年来计算机体系结构领域三大顶级会议(Micro,ISCA,HPCA**)以及其他重要国际会议和期刊相关问题的最新研究成果.详细论述了片上多核处理器缓存一致性协议设计所面临的关键问题,着重探讨了目前和将来多核一致性的优化策略.最后,对多核架构下高速缓存一致性的研究趋势和前景进行了展望.

1 Cache 一致性问题的起源与发展

1.1 Cache一致性问题的由来

在多核处理器中,处理器核心对本地节点高速缓存的访问时延最短,对非本地节点高速缓存的访问时延则会受到片上互连结构和相对位置的影响.随着片上节点数量的增大,通信延时的增长,本地和非本地高速缓存访问延时的差异愈加明显.为了平衡这种时延,在多核处理器设计中引入了软\硬件共享存储模型,即,每个核都拥有私有一级缓存和共享的末级缓存(last level cache,简称 LLC).为了缩短数据访问片上网络(network on chip,简称 NoC)的延迟,处理器核心获取所需数据之后在本地创建数据副本,而不论其他节点的高速缓存是否存在相同的数据.私有缓存机制保证了本地节点内的处理器核心独占本地节点的高速缓存资源.由此产生了缓存一致性(cache coherence)[7]问题,其根本原因是:在一个多处理器系统中,不同的缓存中可能存在同一个数据的多个副本.这些副本的存在,严重影响了程序执行的正确性.这样就需要缓存一致性协议(cache coherence protocol)来管理共享数据的多个副本.

缓存一致性问题有多种定义方式,Gharachorloo 等学者给出的定义[8]为:

(1) 每次写操作对所有的核可见;

(2) 写操作是顺序化的,即,所有的内核观察到相同访存序列.因此,缓存一致性要求写操作必须最终广播到参与的全部处理器中.同时,要求参与的内核所观察到的对同一个地址的写操作,必须按照相同顺序进行.

Hennessy 和 Patterson 认为,缓存一致性确定了读取操作可能返回什么值,给出的定义如下所示[9].

缓存一致性由 3 个不变式组成.

(1) 处理器 P 读取位置 X,在此之前是 P 对 X 进行写入,在 P 执行的这一写入与读取操作之间,没有其他处理器对位置 X 执行写入操作,此读取操作总是返回 P 写入的值;

(2) 一个处理器向位置 X 执行写入操作之后,另一个处理器读取该位置,如果读写操作的间隔时间足够长,而且在两次访问之间没有其他处理器向 X 写入,则该读取操作将返回写入值;

(3) 对同一个位置执行的写入操作被串行化.

一致性协议可以使用软件或者硬件方式来实现.在片上多核处理器中,缓存一致性协议的实现与片上网络

** ISCA: Int’l Symp. on Computer Architecture,计算机体系结构国际研讨会;MICRO: Int’l Symp. on Microarchitecture,微体系结构国际研讨会;HPCA: Int’l Symp. on High Performance Computer Architecture,高性能计算机体系结构国际研讨会.

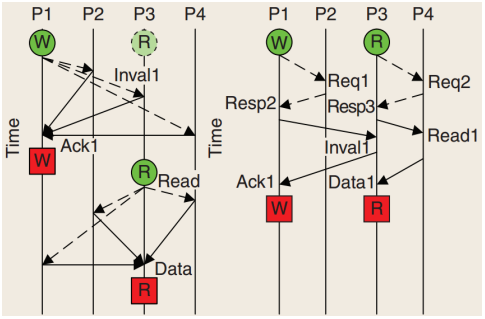
密切相关.根据维护缓存块状态方法的不同,处理器可以使用侦听(snooping)和目录(directory)两大类机制.

1.2 侦听一致性协议

侦听一致性协议是利用总线广播(broadcast)机制来实现的,是 Cache 一致性协议最早的实现方式.系统中的所有高速缓存控制器都需要侦听系统中的一致性消息,以此来确定是否有一致性请求.最为经典的总线监听协议是 MESI[10],由 James Goodman 提出,在 x86,ARM 和 Power 处理器得以具体实现.

图 1(a)是广播一致性协议的简单示意图.在广播协议中,当处理器 *P*1 发生写(W)操作时,一个无效请求(虚线)被发送到所有其他的处理器,以获得适当的权限来执行写操作.接着写操作被执行.因为广播协议在执行时,占据了整个总线,*P*3 的读操作必须推迟到写操作完成才能进行.当 *P*3 发出读请求时,所有其他处理器必须侦听,直到数据返回给 *P*3 读操作完成.总的来说,对于规模较小的多核处理器基于广播的方法是适用的.

在侦听协议中,总线是所有一致性消息的定序点.所有连接到总线上的节点都以相同的顺序观察到总线上的一致性消息.但是同一时刻只能有一个消息在总线上传输,每个缓存控制器分别管理自身数据块的状态,并通过总线进行不同缓存间的状态同步.然而,随着多核规模的不断扩大,多个通信组件必将产生对总线的争用.加之进行作废\更新(invalidate\update)操作时会进行全系统的消息广播,因此,一致性产生的片上网络负荷会不断加重,同时伴随的功耗问题也不容忽视.因此,基于侦听的一致性协议可扩展性较差.



(a) 广播 (b) 目录

Fig.1 Illustration of broadcast and directory coherence[11]

图 1 广播一致性协议和目录一致性协议示意图[11]

1.3 目录一致性协议

目录一致性协议使用目录来全局记录缓存状态,包括数据的共享者列表、一致性状态等.在目录一致性协议中,目录充当了定序点的角色.消息的传输是以点对点的方式进行的,即,所有一致性消息通过一个目录结构来转发处理,因此,它可以有效地降低一致性消息对网络带宽的需求.图 1(b)是目录一致性协议的简单示意图.当 *P*1 发生写操作时,首先查询 Home 节点(*P*2)来确定当前所有者/共享者.Home 节点响应请求,*P*1 的无效请求接着被发送给当前所有者/共享者.每个相关节点都要给予回复确认.一旦 *P*1 已经收到了所有的应答,然后执行写操作.类似的过程,*P*3 完成读操作.在这种情况下,因为网络不是完全被占用,读和写都可以并行执行.目录一致性适用于弱一致性模型和大型系统.目前,商用的有 Intel 的 Core i7 系列处理器,它采用了 QPI(quick path interconnect)技术[12,13],支持 MESIF[14]一致性协议.

目录协议避免了广播消息,减少了完成一致性请求所需的通信量,同时避免了对顺序化互连结构的依赖,具有较好的可扩展性.目前,绝大多数片上多核处理器都采用基于目录的一致性协议.目录协议最基本的硬件实现方式是为存储器的每个数据块分配一个目录条目(entry),记录下该数据在哪些缓存有副本.在一个典型的目录条目中,包含了缓存块的状态、共享者列表、拥有者标识等信息.其中,共享者列表常采用位图的形式来实现,而拥有者标识则表示相应的处理器核标识或者节点标识号.如图 2 所示:目录条目{1000}*M* 显示,数据块 *A* 位于

*Core*0 的私有缓存中,状态为 Modified;目录条目{0110}*S* 显示,数据块 *B* 在 *Core*1 和 *Core*2 的私有缓存中都有副本,状态为 Shared.

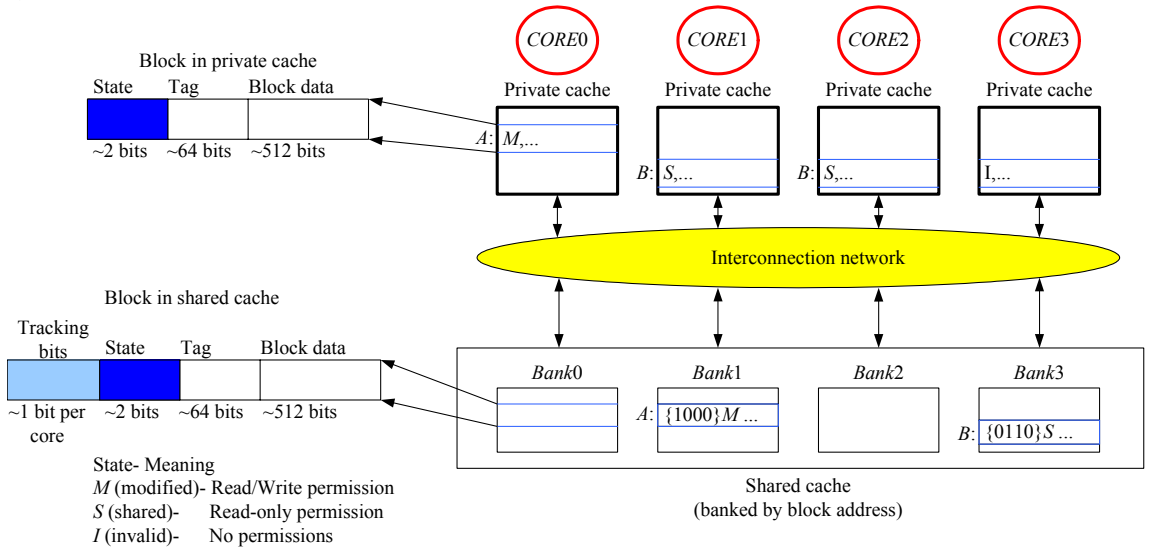


Fig.2 Directory structure of the cache coherence protocol based on directory[15]

图 2 基于目录一致性协议的目录结构[15]

1.4 经典的一致性协议

MESI 协议也称为 Illinois MESI 协议,是一种非常典型的缓存一致性协议[10].协议以缓存块的 4 种状态命名.

- M 代表修改状态,表示已修改过,当前缓存行中包含的数据与存储器中的数据不一致,只有该缓存中有最新的拷贝,其他缓存中无该块的有效拷贝;
- E 表示独占状态,说明只有一个处理器核有该内存块的缓存,并且该块在缓存中未被修改过,主存中是最新的;
- S 代表共享状态,表明该块在缓存中未被修改过,并且其他核存在该块的拷贝;
- I 代表无效状态,说明该块在缓存中是无效的,或者该块还没有进入缓存.

随着片上多处理器的出现,一些适合于新的计算机体系结构的缓存一致性协议相继提出,这些协议通常需要考虑存储系统的互联网络的特点.MESI 协议出现的变形,如 MOESI 协议和 MESIF 协议.AMD 公司的 MOESI 协议用于保证其超传输(HyperTransport)互联系统的缓存一致性[16,17].MOESI 协议中,M,E,I 这 3 种状态与 MESI 协议相同,重新定义了 S 状态,引入了一个新状态 O.状态 O 表示拥有者,在当前缓存行中包含的数据是最新的,而且在其他 CPU 中一定具有该 Cache 行的拷贝,其他拥有副本的缓存状态为 S.有且仅有一个 Cache 行状态为 O.在 MOESI 协议中,状态为 S 的缓存行中的数据与存储器中的数据不一定一致.如果在其他缓存中不存在状态为 O 的拷贝,则该缓存行中的数据与存储器一致;如果在其他缓存中存在状态为 O 的拷贝,则缓存行中的数据与存储器不一致.

Intel 提出了 MESI 协议的变种,即 MESIF 协议.该协议与 MOESI 协议类似,都是对 MESI 协议的扩充,因此比 MESI 协议更复杂.MESIF 协议解决的主要问题是 ccNUMA 架构中 SMP 子系统与 SMP 子系统之间缓存一致性问题[18].MESIF 协议引入了一个 F(forward)状态.在采用 ccNUMA 的处理器系统中,多个处理器的缓存中可能存在相同数据的拷贝,在这些拷贝中,只有一个缓存行的状态为 F,其他缓存行的状态都为 S.当缓存行的状态位为 F 时,缓存中的数据与存储器一致.当一个数据请求读取这个数据副本时,只有状态为 F 的缓存行可以将数据副本转发给数据请求者,而状态位为 S 的缓存不能转发数据副本.MESIF 协议有效地解决了 ccNUMA 处理器结构中,所有状态位为 S 的缓存同时转发数据副本给数据请求者而造成的网络拥塞问题.

2 国内外研究现状

2.1 国内外研究概况

Cache 一致性研究一直是 Cache 优化的热点问题,国内外许多著名大学和研究机构都加入到这项研究之中,以 Intel,AMD,NVIDIA 等为代表的产业界也广泛吸收先进成果,极大地促进了该领域的发展.图 3 统计了近年来三大会议中有关 Cache 主题的论文数量,浅色的数据段是其中有关 Cache 一致性的论文数量.由此可见:在每年发表的与 Cache 有关的论文中,Cache 一致性问题接近三成.

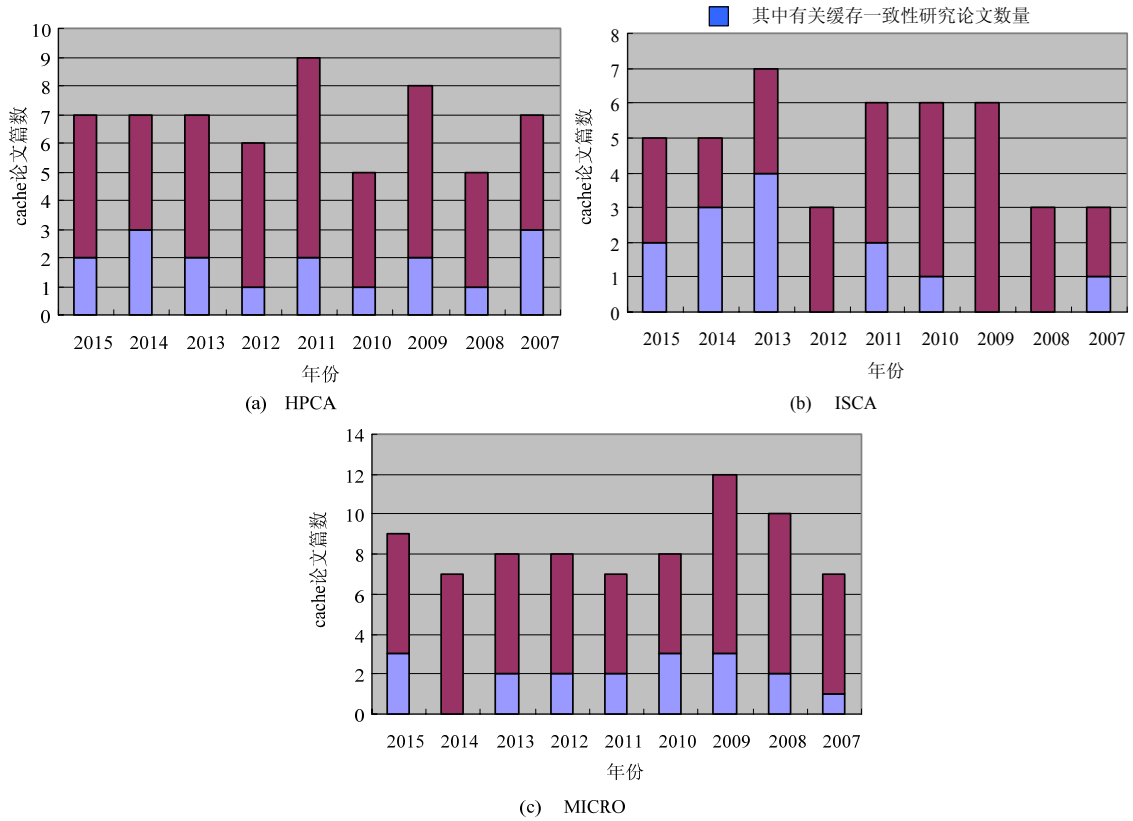


Fig.3 Publication statistics on cache coherence

图 3 三大会议有关 cache 一致性论文统计

国际上对于 Cache 一致性的研究起步在 20 世纪 80 年代,早期的研究关注于协议在多处理机上如何正确运行、高效运行.随着多核规模的扩大,程序并行度的提高,目前的研究呈现出新的特点.比较有代表性的研究有:美国犹他大学的 Cheng[19]和罗切斯特大学的 Hossain[20]倡导通过发掘程序的共享行为来优化 Cache 一致性协议;西班牙学者 Cuesta[21,22]和 Valls[23,24]等人利用操作系统在程序运行时识别出共享和私有数据,优化目录条目的数量,从而达到缩减一致性存储开销和降低功耗的目的;瓦伦西亚理工大学的 Ros 优化了传统一致性操作(transaction)的步骤,提出了直接一致性协议(DiCo-CMP)[25],使一致性协议产生的片上网络流量得以减少;针对事务内存(transactional memory,简称 TM),加州大学伯克利分校的 Qian[26]、佐治亚理工大学的 Park[27]、路易斯安那州立大学的 Chen[28]分别提出了改进的缓存一致性协议.众核(many core)的研究方兴未艾,Cache 一致性问题更多地考虑了协议的可扩展性以及层次化[29].例如伊利诺伊大学厄巴纳-香槟分校的 Kelm[30]提出的 WayPoint、斯坦福大学 Sanchez 和 Kozyrakis 提出的 SCD[31],使一致性协议可以扩展到千核规模的处理器上.最近几年,GPU 的 Cache 一致性研究也逐渐成为热点[32−35].

国内在该领域的研究也取得了一系列成果.中国科学院计算技术研究所的范东睿等学者针对 Godson-T[36]众核体系结构的特点,研究了与缓存一致性密切相关的存储一致性(consistency)问题,设计了同步机制,并对传统的 Broadcast 算法进行了优化[37,38].黄河等学者提出了一种硬件结构支持的基于同步的高速缓存一致性协议,在降低设计和验证时复杂度的同时,可以获得与基于目录的协议相当的性能[39].中国科学技术大学的安红、李功明等人[40]从功能扩展和性能优化两方面对一致性模型展开研究,提出了一个基于本地目录机制的高速缓存一致性模型 Loc-Dir,并且引入预测机制来降低目录一致性模型中的间接数据传输延迟.西北工业大学的张骏[41]提出了基于节点预测的直接 Cache 一致性协议,该优化方案使一致性交互延迟和目录存储开销不随内核数量的增加而快速增长.清华大学的郭松柳、汪东升等人[42,43]在层次化的一致性协议研究中做了许多有益的工作.

2.2 仿真平台

仿真在缓存一致性的研究中至关重要,在模拟器的环境中可以进行快速的设计空间探索,缩短研发周期.现在已有 M5[44],Simics[45],GEMS[46],GEM5[47]等多核模拟器.GEMS 由威斯康星大学的 Hill 和 Wood 等人研发,是一款高度可配置、集成多种商用 ISA 和多种处理器模型的开源模拟器,支持包括 x86,ARM[48],ALPHA,MIPS,Power,SPARC 等处理器.它可与 Simics 一起工作,负责模拟芯片内的存储互联子系统,构成 Simics+GEMS 模拟平台.目前,国内外在缓存一致性的研究中,大部分理论验证和实验仿真基本都在 GEMS 平台上进行.

GEM5 的存储模型整合了 M5 的 Classic 模型和 GEMS 的 Ruby 模型.Classic 模型是最简单的模型,它提供了简洁、快速的可配置性;Ruby 模型注重于精确度,设计者开发了专门的脚本语言 SLICC(specification language for implementing cache coherence)对 Cache 一致性协议简约地进行描述,并且支持所有传统的 Cache 一致性协议,包括 MI_example,MOESI_hammer,MOESI_CMP_token,MOESI_CMP_directory 和 MESI_CMP_directory[49].GEM5 模拟器的出现极大地推动了多核体系架构下 Cache 的研究.

3 研究的关键问题

多核 Cache 一致性研究涉及许多方面,其中要解决的关键问题可归纳如下:程序访存行为分析、目录组织结构、一致性协议流量(traffic)、一致性粒度(granularity)、目录协议的可扩展性(scalability)、低功耗以及具体软/硬件实现等.

3.1 程序访存行为分析

应用程序的访存行为是 Cache 一致性协议优化的根本依据.在单核时代,程序访存局域性理论对 Cache 的优化做出了重大的支撑.在多核时代,随着多线程程序的并行编程模型的出现,程序的并行度不断提高,这使得程序的局域性特征呈现出新的特点.比如:瑞士洛桑联邦理工学院的学者 Alisafaee 提出了时空一致性跟踪(spatiotemporal coherence tracking,简称 SCT)[50]方法.深入分析程序的访存行为,还可以发掘多种数据的共享模式(sharing pattern).利用软/硬件手段预测和识别这些共享模式,能够对一致性策略做出自适应的调整[51].目前,虽然对数据共享模式的研究在编译优化层面已经非常深入,但是该研究被应用到 Cache 一致性协议的优化中才刚刚开始.无论是在预测的精度还是为此付出的代价上都还有待进一步的优化,这也是近期研究的热点.

3.2 目录结构组织

目录能够精确记录跟踪共享数据信息,是当前多核处理器的标准解决方案,目录的组织结构与一致性协议密切相关.目录结构上的一些细微改动,就能够很大程度上降低目录的存储空间.每种目录组织结构都对应着在其上构建的一致性操作算法.目录组织结构设计的难点在于:用尽可能少的存储开销精确记录数据的共享信息,而不衰减系统性能.

3.3 一致性粒度

传统目录缓存的每个条目管理一个 Cache 行,在目前的 Cache 结构中,一个 Cache 行一般 64 个字节.而调查发现:在程序中绝大多数的数据是私有的,共享数据只占很少比例.对某些程序而言,多个并发线程可能会访问

由多个 Cache 行组成的一个连续的“区域”.考虑到这个区域中的缓存块可能有相同的特性,可以将这个区域识别为一致性区域,为这个区域设置一个目录条目.一旦区域中发生私有/共享的变化,就会导致整个区域的瓦解,而不得不更新为 Cache 行的粒度.通过操作系统在运行时发掘共享数据信息,最小的粒度为一个页面,一般为4KB~8KB,这是目前一致性研究中最大的一致性粒度.罗切斯特大学的 Zhao 等人使用缓存缺失率、作废率、数据的使用率这 3 个指标,分析了应用程序的行为特征.应用程序空间局域性和存储粒度之间的错误匹配,导致了相当大的未使用数据的移动;而应用程序共享粒度和一致性粒度之间的错误匹配,导致了伪共享(false sharing),从而带来性能和带宽的惩罚[52,53].

3.4 一致性协议流量

在片上多核处理器中,一致性消息通过片上网络进行传输.因此,一致性协议不仅决定了共享存储系统内部如何通信,还影响到存储系统与内核、片上网络之间进行数据的传输[54].比如 Ros 提出的直接一致性协议,以一致性协议的动作本身为优化对象,减少了一致性消息操作的跳数[27].然而,在当前的片上网络研究中,往往以提高片上网络的吞吐率为目的.在仿真中,并没有真正考虑一致性消息传输的问题[55].因此,结合片上网络来进行的一致性模型优化,是提高一致性协议及片上网络性能的关键方法.

3.5 可扩展性

影响一致性协议可扩展性的因素主要有目录的存储开销、一致性交互延迟以及一致性操作动作等,不仅涉及到协议本身,而且与片上网络的性能也紧密相关.随着多核规模的扩大,层次化的多核层次化分组结构是必由之路.在分组内采用广播方式,在组间采用目录方式,这种混合的一致性协议已经开始在逐步探索之中.

3.6 低功耗

在目前的多核研究中,一方面降低功耗,一方面使程序的并行性得到发挥,这样仍能提高系统的整体性能.Cache 的功耗已成为当前多核体系结构设计面临的最大问题之一.Cache 的功耗分为静态功耗和动态功耗.

- 静态功耗与其容量大小正相关,即:Cache 容量越大,功耗越大.而且随着 Cache 容量的增加,访问延时也迅速增加;
- 动态功耗与程序的访存行为有密切关系,减少不必要的一致性访问的次数和降低每次读写访问的功耗,都能降低 Cache 的功耗.

Cache 功耗的高低,已经成为评价 Cache 一致性协议优化的重要指标之一.

4 主要研究进展分析

目前,绝大多数片上多核处理器都采用基于目录的一致性协议.目录协议发展到多核时代,也暴露出潜在的问题和需要改进的地方.下面本文将分别从程序访存行为、目录组织结构、一致性粒度、一致性协议流量、目录协议的可扩展性这 5 个方面来阐述国内外最新的科研成果.

4.1 面向程序访存行为的优化

非一致 Cache 体系结构(non-uniform cache architecture,简称 NUCA)[56]的概念,在 CMP 环境中的应用研究已逐步得以推广.私有缓存可以提供快速的本地访问,但存储容量较小;共享缓存能够提供较大的缓存空间,但命中访问延迟会增加.因此,缓存一致性问题与缓存的组织结构和使用方式有关.多核处理器出现片上存储器数据不一致的根源是多个内核可能拥有同一个数据的副本,而写操作导致多个内核的副本失效.随着多核处理器应用范围的扩大,深入理解片上多核处理器上的工作负载的访存行为对于优化缓存一致性设计至关重要.目前的优化一般针对迁移(migratory)、生产者-消费者(producer-consumer)、多读者(multi-reader)、多读者-写者(multi-reader writer)这 4 种访存行为模式.

- 迁移模式.数据共享的特点是:数据被读取,然后被修改.这样的读取修改操作反复进行.这意味着数据访问需要启动两次单独的请求(导致访问远程节点,读写修改两次),以首先获得读权限,然后获得写权

限;

- 生产者-消费者模式.数据共享的特点是:生产者线程把数据放入缓冲区,而消费者线程从缓冲区读取数据.生产者发生写请求时,需要通知消费者作废原有数据,重新获取;
- 多读者模式.由于没有写者,多个处理器核以只读方式访问数据;
- 多读者-写者模式.数据共享的特点是:一个写线程和多个读线程,当写操作发生时,必须通知所有的读者.由于程序访存的局域性,还有可能导致伪共享.

Stenstrom[57]研究了通过连接依赖和从运行时系统到缓存一致性的映射信息,来引导缓存一致性优化.通过获取这些运行时缓存一致性信息,Stenstrom 优化了 downgrading 和 self-invalidation 两种一致性动作,减少了生产者-消费者共享模式和迁移共享模式的开销.

面向共享数据模式的优化的关键在于预测机制和识别机制[58].Cheng 研究了生产者-消费者共享模式,提出了针对该模式的自适应的缓存一致性协议[21].Cheng 在文献[21]中首次使用硬件机制实现了预测器,用来识别生产者-消费者数据共享模式.为了实现预测器,Cheng 在目录条目中增加了 3 个域,分别是:

① 最后写者.跟踪最后写者的节点;

② 读者计数器.这是一个 2 位的饱和计数器,记录自上一次写操作以来读请求的次数;

③ 重复写计数器.这是一个 2 位的饱和计数器,当来自相同节点的连续两次写操作(中间至少一次读),计数器加 1.当重复写计数器达到最大值时,预测器便认为发生了生产者-消费者共享模式.一旦预测器识别出生产者-消费者模式,目录的管理将委托给生产者节点进行管理.

Cheng 还提出,消费者节点通过直接向生产者节点发送数据请求,实现将 3-hop 一致性操作简化为 2-hop.在硬件结构上,Cheng 采用了远程缓存(remote access cache,简称 RAC)作为牺牲缓存(victim cache),设计了代理缓存(delegate cache)用来存储生产者和消费者信息,并且扩展了目录控制器的条目.

Hossain 分析了多线程程序的访存行为,提出了 ARMCO(adaptive replication migratory producer-consumer optimization)[22].该研究针对上述几种共享模式设计了预测器,对于迁移模式,ARMCO 避免了频繁的两次访问;对于生产者-消费者模式,ARMCO 使用直接访问生产者节点获取数据;对于多读者模式,ARMCO 从最近的拥有副本的 $L1$ 获得数据,而不是从数据拥有者获取数据.

小结.面向程序访存行为模式的一致性协议优化依赖于数据访问的历史记录以及合理的预测逻辑,涉及预测算法、识别手段、预测表数据结构以及预测机制更新等.预测精确性是性能能否提升的关键,如果预测成功,可以加速一致性事务的执行,并且简化一致性操作的步骤,从而降低了维护数据一致性的片上网络流量.

4.2 面向目录组织结构优化

在目前的商用多核处理器中,为避免目录访问的延迟,系统架构工程师一般将目录设计成目录缓存(directory cache)[59]位于片上缓存中.因此,目录的组织结构设计不仅关系到片上存储开销,也影响到一致性协议操作算法.在本节,我们将讨论针对目录缓存存储结构的优化设计.

4.2.1 传统的目录存储空间组织

最早的研究是 DUP(duplicate tag)[60],其思想是:复制每个私有缓存的 Tag 阵列,集中存储在一个结构中.Piranha[61]和 Nigara2[4]处理器使用的就是这种目录.但是,随着核数的增长,DUP 需要高度相联的 Lookup,并且产生大量的能耗,同时伴随着延迟,难以扩展,难以在多核上实现.

片上目录要准确记录下每个缓存块在当前时刻有哪些共享者.全映射向量目录(full-map directory)[15]使用全局的位图结构,要求每个缓存块都对应一个目录条目.全映射向量目录一般采用组相联的目录结构,每个目录条目包含了一个 Tag、一个 n 位向量(n 为处理器内核的数目).n 位向量用来存储共享节点信息,每一位表示相应的内核是否拥有当前缓存的副本.可以看出:全映射向量目录的存储开销,将随着核数的增长呈现出平方级 $O(n^2)$增长.

在实际中,共享数据的比例影响着全映射向量目录的使用率.对 NAS[62],PARSEC[61],SPLASH2[63]等并行程序测试集的调查结果显示,平均 77%的数据都是私有的[64].这表明,全映射位图是名副其实的稀疏矩阵.全映射

向量目录牺牲了片上面积,当多核的规模扩大时,目录存储开销是不可承受的,因此可扩展性较差.研究者开始着手缩小目录结构的存储,目录存储结构的优化呈现出两个特征:一是压缩单条目录条目的大小;另一个是采用可变的目录粒度,减少整体目录的大小(第 4.3 节给出讨论).

4.2.2 压缩单个目录条目的方法

该方法基于一个事实,即:共享的数据在程序中只占较小的比例,且数据同时被多个核共享的比例也较小.该方法一般都采用压缩格式的稀疏目录(sparse directory)结构[65].Hill 等人证明,稀疏目录能以较小的面积和更大的灵活性来获得目录的可扩展性[16].粗向量(coarse vector)[66]目录结构,如 $\text{Sparse}_{\text{COMP-}n/4}$、$\text{Sparse}_{\text{COMP-log}(n)+1}$[67]、有限指针(limited pointer)[67]、Stash 目录[68]等存储结构,都是压缩每条目录条目的方法.

$\text{Sparse}_{\text{COMP-}n/4}$ 每 4 个核用 1 比特位记录共享信息,不能准确跟踪共享信息,导致必须进行进一步的嗅探,降低了性能.$\text{Sparse}_{\text{COMP-log}(n)+1}$ 采用 log(n)位编码,用链表记录每一个共享者信息,导致了性能和带宽的开销.图 4 所示为存储配置为 64KB 私有 $L1$、256KB 私有 $L2$、1MB 共享 $L3$ 的系统中各种目录结构所占片上存储空间的比例(SPACE 和 SCD 目录在后面介绍).

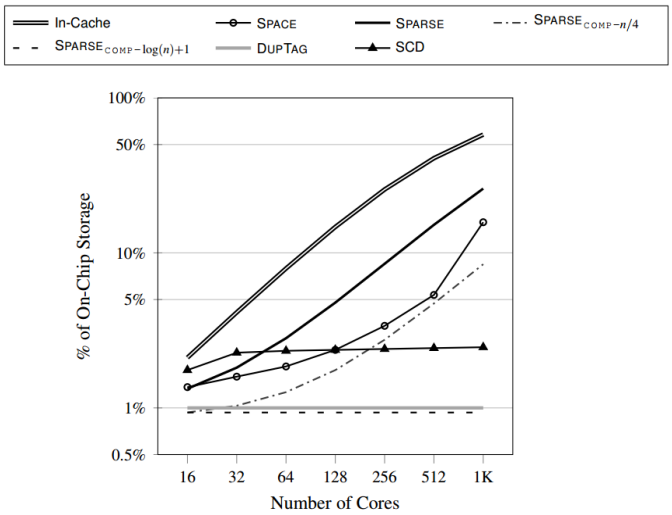


Fig.4 Fraction of on-chip storage occupied by coherence directory

图 4 各种目录所占片上存储空间

小结.采用压缩稀疏目录条目的方法,由于采用了粗粒度的跟踪方法,并不能精确地记录共享信息,还需要采用额外的手段进行共享信息的跟踪.另外,有的压缩方法设定了目录可以跟踪共享节点的上限,在共享节点数量不超过上限时能够进行准确的跟踪,而当共享节点数目超出上限时,则会发生目录溢出.因此,还需要引入冲突解决机制实现目录置换.有限指针技术采用动态指针分配的链表机制来存储共享者信息,避免了目录溢出情况的发生.在有限指针技术中,头指针占用较大的存储空间,空间利用率不高.另外,共享节点的信息是以链式来组织的,每次访问目录都要顺序遍历链表.尽管有的设计采用了双向链表来实现,但遍历速度的提升仍然十分有限.

4.3 面向一致性粒度的优化方法

一致性粒度从最小的 Cache 行粒度到区域粒度,再到最大的操作系统页面的粒度,目前都有研究成果发表.共享数据的粒度是多样的[69],但是固定粒度的一致性协议导致了不必要的整个数据块的作废,这样带来的访存带宽开销和一致性流量开销是巨大的,也影响了协议的可扩展性能.本节将着重从粒度的识别来进行论述.

4.3.1 从识别私有和共享数据着手

由于私有数据在其他缓存没有副本存在,一致性维护时可以不予考虑,这样就避免了目录随着核数的增长

而增长.目前,有多种识别私有和共享数据的方法[67].对程序的 Trace 分析发现,对同一内存地址仅仅由单个处理器完成的访存占绝大多数,因此也不需要为所有的缓存提供一致性支持.这种免除一致性机制的关键是划分私有数据和共享数据,并阻止目录缓存跟踪私有数据.Cuesta 首次从操作系统的层次来解决一致性问题,提出对私有数据免除一致性(deactivate coherence)[23,24].Cuesta 对私有数据的划分是基于 Hardavellas 的 R-NUCA 结构[70].为了能够识别出私有数据,文献[70]在页表和页表缓冲(translation lookaside buffer,简称 TLB)中增加标志位,并且借助操作系统提供支持.每个新页面装入时,默认都是私有的.当操作系统发现有其他节点访问此私有页面时,触发一致性恢复机制.一致性机制启动后,对该页面中所有的数据进行跟踪,维护一致性.

当前,多线程工作负载和多程序工作负载数据访问各有特点:尽管多线程工作负载中共享数据所占比例较高,但是绝大多数的数据仍然仅仅被一个线程所访问;而多程序工作负载中都是私有数据.既然私有数据仅仅被一个线程访问,就可以把它委托给正在访问的这个节点.这样就避免了一致性消息在片上网络的传输,减少了网络流量.从数据是否共享的角度,提出了 POPS(coherence protocol optimization for both private and shared data)[71]一致性协议.由于迁移共享模式,具有和私有数据相似的特征,所以可以与私有数据执行相同的优化策略.POPS 实现了自适应的机制,根据观察到的数据共享模式变化一致性协议动作,也加速了一致性协议的执行.

Valls 提出了 PS-Dir(private shared directory)[25,26].PS-Dir 由两个独立的缓存组成.在运行时,根据每个缓存块的类型,两个目录可以互相转化.由于程序中共享数据的比例较小,PS-Dir 的共享缓存的相联度和目录条目设计得比较小,从而节省了功耗和查找的延迟.而私有缓存使用大的相联度,不存储共享节点信息.在发生共享的情况下,目录条目能够从私有缓存移动到共享缓存,实现跟踪共享节点.在这种机制下,Valls 实现了私有共享两级目录.由于私有缓存没有存储共享节点信息,PS-Dir 的目录面积减少了 26%.

4.3.2 从程序局域性行为着手

程序访存的空间局域性行为表明,程序在一段时间内可能仅仅局限访问某个连续的地址空间,因此考虑只为局部区域进行一致性维护.比如,Alisafaee 观察到,共享数据并不是时刻处于活动的状态,即,多个内核同时访问同一个数据的概率较小[52].加拿大多伦多大学的 Zebchuk 和 IBM 的 Qureshi 研究了数据访问的热点区域,发现在某些时刻,大的连续缓存块经常仅仅只被某个特定的内核访问[72].

Alisafaee 提出了双粒度跟踪和粒度转变的方法,所谓双粒度是指数据块粒度(block level)和区域粒度(region level)[52].图 5 显示了 SCT 的目录条目结构.

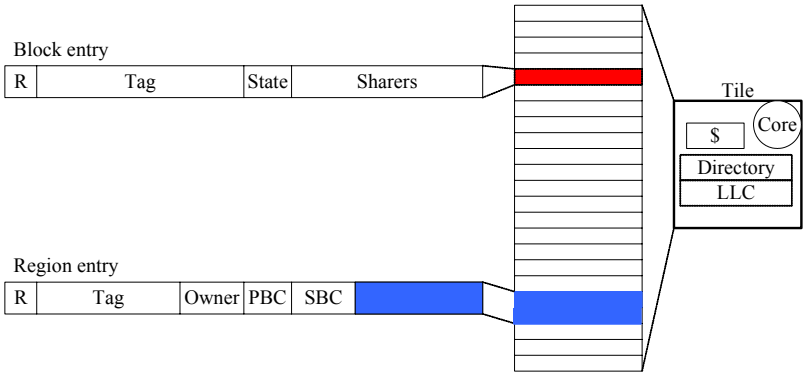


Fig.5 Structure of SCT directory[52]

图 5 SCT 的目录结构[52]

SCT 将缓存空间划分成固定大小的空间区域,每个空间区域包含了几个连续的缓存块;把在一个时间段内仅被一个内核所访问的空间区域,定义为时域私有区域(temporarily-private region).每个时域私有空间只设定一个目录条目标识该区域.当某个空间区域第 1 次被访问时,SCT 标记它为时域私有区域.如果随后有其他内核申请访问这个区域,那么这个区域就转换成时域共享区域(temporarily-shared region),SCT 将为这个区域内的每个

缓存块分配块粒度的目录条目.当时域共享区域中的所有块被置换出来时,这个区域会再次变成时域私有区域.实验结果表明:相比全映射向量目录,SCT 方法能够降低至少 75%的目录大小.研究结果表明:多粒度目录能够针对多种负载,不仅能够减少一致性流量开销,还能有效减少整体目录条目的数量.

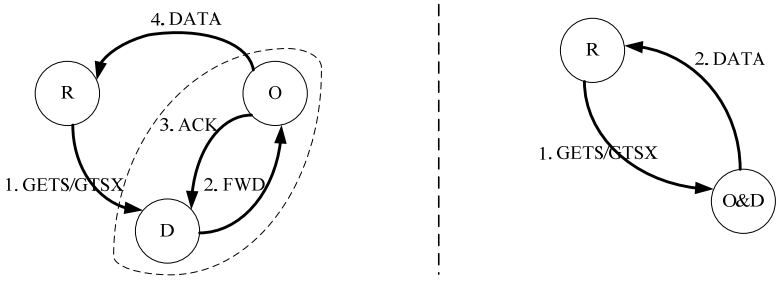
罗切斯特大学的 Zhao 等人提出了 SPACE[73,74]目录协议,该研究比较新颖.Zhao 观察到一种特殊的程序局域性,即:在全映射目录中,绝大多数的缓存块被相同组合的处理器内核共享.如果用全向量目录结构,以 16 核为例(每核用 1 位比特位来记录是否拥有数据副本,那么将共可以记录 2^{16}=65536 种内核组合).Zhao 等人对 SPLASH2, Graphmine[75]以及 Apache,SPECjbb2005[76]等测试集模拟后发现:共享数据所对应的处理器内核组合最大不超过 1 800 种,并且 80%的共享发生在 200 个左右处理器内核组合中.Zhao 等人得出一个结论:可以仅仅使用 200 个左右的目录条目,不必要用全映射向量目录精确保持共享者信息.SPACE 从每个 Cache 块中解耦了共享模式.在每个缓存的 Tag 部分增加一个 SPACE 指针,指向目录条目(即共享的处理器组合向量).Zhao 的这种目录设计也是基于感知和共享模式的.

小结.面向一致性粒度的优化方法,其核心仍然是识别共享数据,这些技术的共同点是都需要额外的系统支持来实现数据访问模式的跟踪.由于 Cache 的透明性,采用 Cache 行和区域粒度识别共享数据,精度都不高.通过操作系统识别,虽然比较精确,但是粒度较大.然而无论采用何种粒度,都会面临伪共享的问题.由于采用了较大的粒度作为跟踪单位,降低了硬件成本和操作复杂度.对于需要频繁变化一致性粒度的程序负载,得到的收益可能并不大.

4.4 面向协议流量的优化

随着片上集成内核规模的扩大以及程序并行度的增加,必然导致核间通信量的增加;考虑到核间距离,访问延迟更加明显.目录在维护数据的一致性中扮演了管理者的角色,数据请求都要通过目录节点中转,形成了间接访问效应.如图 6(a)所示:在传统的目录一致性协议中,当发生数据缺失时,由请求节点向目录节点发送请求,目录节点将请求转发给数据的拥有者节点;数据的拥有者收到请求后,给目录节点应答,目录节点完成目录的更新;最后,数据的拥有者将数据发送给请求节点.

Ros 提出直接一致性协议(DiCo-CMP)[27],该机制通过扩展缓存的 Tag 部分,在 *L*1 和 *L*2 中分别增加新的域,用于记录数据的拥有者.在 DiCo-CMP 协议中,请求节点直接向数据的拥有者发送请求,如图 6(b)所示,而不是先发送到目录缓存,再由目录控制器转发到数据拥有者.相对于传统的 4-hop 的目录协议,DiCo-CMP 协议减少了一致性操作的数量,能够使访存延迟平均减少 6%左右.



(a) 传统的 4-hop 的目录协议的数据请求过程 (b) 2-hop 的 DiCo-CMP 目录协议数据请求过程[27]

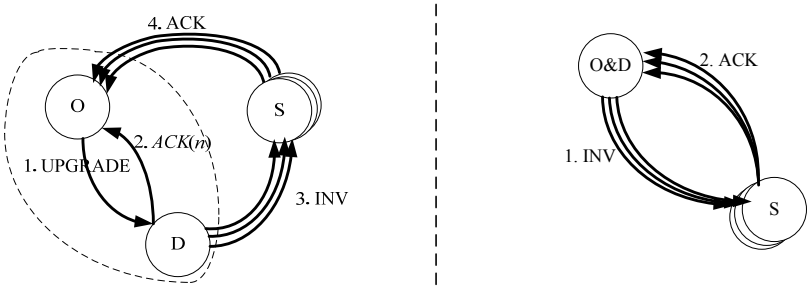
Fig.6 The data request transaction

图 6 数据请求过程

图 7 显示了传统的目录协议的数据更新过程和 DiCo-CMP 目录协议的数据更新过程.与数据请求过程类似,DiCo-CMP 目录协议数据更新过程也简化为 2-hop.

与 DiCo-CMP 协议类似,SWEL[69]一致性协议也致力于减少一致性操作步骤来优化片上网络的通信流量.SWEL 代表了 Shared,Written,Exclusivity 这 3 种状态.SWEL 协议基于一个理念:如果某个数据块仅被拥有者访

问或者是只读的,那么该数据块直接存储在私有缓存中,不需要一致性维护;如果某个块是共享的并被进行了写操作,那么将这个块存储在下层共享缓存中,以便所有内核都可以平等地访问它,这也不需要维护数据的一致性.这样,$L1$ 只存储私有和只读的数据块,而共享和可写块存放在共享 $L2$ 中.然而,对于共享数据较多的应用程序,这种设计必然导致经常访问下层共享缓存,降低了程序的执行效率.美国犹他大学的 Pugsley 提出了一种改进办法 RSWEL(reconstituted SWEL)[69],即,给予共享的数据块在适当时候能够重新被缓存到 $L1$ 中的机制.实验结果表明:相比 MESI 协议,RSWEL 可提高 15%的性能.SWEL 协议可以避免目录结构的使用,从而降低了实现目录一致性协议的开销,消除了硬件资源对系统扩展性的限制.但是由于共享的数据被存储于共享缓存中,降低了数据访问的速度.当执行访存密集型多线程程序时,对共享存储的访问必将成为系统的瓶颈.因此,SWEL 适用于规模比较小的片上多核处理器.



(a) 传统的 4-hop 的目录协议的数据更新过程 (b) 2-hop 的 DiCo-CMP 目录协议数据更新过程[27]

Fig.7 The data upgrade transaction

图 7 数据更新过程

小结.本节综述了从优化一致性操作步骤来减少一致性协议的流量的方法.利用缓存到缓存的直接传输,简化核间通信以及减少对目录缓存的间接访问.

4.5 面向协议可扩展性的优化模型

可扩展性一直是一致性协议研究的重点.随着多核逐渐向众核发展,可扩展性成为评价一致性协议的重要指标.面向可扩展的一致性协议,需要综合考虑功耗、目录存储、性能和协议通信流量等因素.

4.5.1 混合目录和广播方式

基于目录的协议可扩展性好,但是存储开销大;而采用广播方式的侦听协议存储开销小,但是当片上集成的内核规模扩大到一定程度时,总线成为系统的瓶颈.

Menezo 提出了一种混合目录和侦听机制的一致性协议 Flask[77].Menezo 采用了 3 种措施来保证系统的正确性和可扩展性,即非包含(non-inclusive)的稀疏目录最小化了功耗、仅仅对最活跃的共享数据维护一致性、使用令牌计数减少通信流量.这些措施综合考虑了性能、功耗和通信量因素,Flask 具有低存储开销、能够抑制大多数广播流量、低功耗的特点.相比相同配置的基于令牌一致性协议(token coherence)[16],Flask 使性能提升了 10%,而功耗降低了 20%.

Kelm 使用面向吞吐量并行工作负载(throughput-oriented parallel workload)评估了 1024 核的 CMP 的一致性结构[32].Kelm 发现:基于广播的探测滤波器(probe filter)[78]最多只能为高达 128 个内核提供良好的性能;而基于无效(invalidate)的稀疏目录虽然能够很好地支持面向吞吐量的工作负载,但是支持千核的目录存储开销是无法承受的.为了达到支持千核规模的目标,Kelm 提出了 WayPoint[32]目录结构.该结构使用小的、低相联度的目录缓存跟踪数据的一致性状态,当发生目录溢出时,置换不触发作废操作的目录条目.另外,WayPoint 目录结构实现了对目录关联度和存储容量的动态增加.WayPoint 以只相当于 4%全映射目录的存储开销,实现了相同的性能,而只占用不到 3%的 Die 面积.

4.5.2 以哈希表实现一致性协议

Zebchuk 和卡耐基梅隆大学的 Ferdman 将哈希表(Hash table)引入到目录缓存的存储中,做了很多有益的尝试.Zebchuk 提出了使用布鲁姆波器(Bloom filter)[79]构造 TL(tagless)目录,消除目录结构对 Tag 依赖的方法[80].TL 只需要为部分共享数据保存目录信息,因此 TL 结构降低了目录实现的开销.对于一个 16 核、1MB 私有 *L*2 缓存的片上多核处理器,相比传统的目录结构,TL 可以降低 48%的面积需求,并减少了 57%的漏电流功耗和 44%的动态功耗.但是,布鲁姆滤波器固有的伪共享缺陷会影响 TL 的性能.

Ferdman 认为:稀疏目录组织形式可能造成部分目录资源成为数据热点区域.另外,由于稀疏目录采用压缩格式,有可能出现目录溢出,导致不能跟踪数据块的所有共享节点的情况的发生,从而出现目录缺失率过高的情况. Ferdman 提出了 Cuckoo 目录结构[81],利用 Cuckoo 哈希函数[82]没有冲突传递性的特点,用 Cuckoo 哈希表来存储目录.Cuckoo 目录结构提高了片上目录资源的利用率.分析结果表明:随着核数的增长,Cuckoo 目录结构中每核功耗和每核片上面积几乎接近一个常量.Ferdman 将其设计成分布式的目录结构,有着良好的可扩展性.

4.5.3 层次化一致性协议

随着片上处理器集成数量的增加,维持一个全局的缓存一致性,代价将相当巨大.目前的解决方法有两种.一种是进行子网划分,另一种是层次化管理.在子网划分的环境中,对片上区域进行了隔离,各个子网维持自己的缓存一致性.一致性通信可以被限定在一个局部范围内,从而总的通信代价将大为降低.Jerger 论述了在分组结构的片上多核处理器中,如果一致性协议不相应地使用层次化结构来实现,这样的架构设计是有缺陷的[83,84].片上系统开始将内核划分成不同的分组(group),例如 Sun/Oracle 的 UltraSPARC T2[4]处理器、AMD 的 Bulldozer[85,86]、NVIDIA 的 Fermi GPU[87,88].在这种背景下,出现了层次化目录(hierarchical directory)[62]协议,与之对应的是平坦目录结构(flat directory)[67]协议.

普林斯顿大学的 Fu 提出了一种新的缓存一致性框架 CDR[89],该框架采用共享内存,可以创建规模从几千到几百万核心的系统,同时保持低存储和能量开销.在这种情况下,Fu 等人研究了两种约束类型:一种是限制参与一致性共享者的数量,一种是分担一致性的 home 节点位置.每个独立的一致性域仅仅跟踪在其域内节点的一致性,而不是整个系统.对共享者数量的约束,使系统存储开销保持了常量不随核数增长.

西班牙学者 Acacio 提出了一种两级目录方案:第 1 级目录采用全映射目录方式存储,第 2 级目录采用压缩方式来减少目录的存储开销[90].清华大学的汪东升、郭松柳等人提出了面向 CMP 结构的层次结构 Cache 目录协议[44,45].图 8 显示了郭松柳提出的 64 核 Tile 结构的逻辑分组方案.

郭松柳通过对内核进行逻辑上的区域划分,构建了相互间有包含关系的多层目录结构,允许从底层目录到高层目录逐步扩大一致性交互的延伸范围,尽量使一致性交互局部化,从而达到降低延迟的目的.

斯坦福大学的 Sanchez 和 Kozyrakis 提出了 SCD 目录[33].SCD 是一种混合使用有限指针和层次化位向量的组织结构,依靠高度相联的 ZCache[91]实现,可以扩展到 1024 核,如图 9 所示.SCD 使用弹性的目录条目记录数据块的共享信息:当数据块只有一个或者少数共享者时,目录条目使用有限指针单级目录;一旦数据共享程度变大,超过了有限指针的容量,SCD 使用根位向量(root bit-vector)和叶位向量(leaf bit-vector)的层次化目录来维护一致性.SCD 目录条目的弹性设计,保证了系统的可扩展性.

张骏[43]在 DiCo-CMP 一致性协议的基础上提出了使用两级目录结构——全局目录和节点目录:全局目录以粗向量方式跟踪数据所在节点,全局目录只负责跟踪其对应 *L*2 中的数据副本;节点目录采用全映射向量目录,负责跟踪节点内数据副本所在的缓存位置.该协议基于共享数据的历史,提供数据的预测机制.利用缓存的共享历史信息来保证只对曾经缓存过数据副本的节点进行更新,从而避免了网络流量开销和不完整更新.节点预测机制保存了最近节点的指针,指向距离请求节点最近的有效数据副本节点.

Ros 针对内核分组的片上结构和群集层次缓存(clustered cache hierarchy)提出了一种简化层次目录协议的优化方案[92].Ros 基于一个重要的观察,面向层次化目录结构,重新定义了私有/共享数据划分方法[93,94].即:数据在一个群集缓存(cluster cache)内是完全共享的,但从群集缓存外观察时是私有的.在一个多级的层次结构中,这个概念递归的应用开始于最靠内核的层次.由于数据在群集内是共享的,在作废数据的时候,采用写穿(write-

through)的写策略,直到群集的最低一级缓存;群集外是私有的,各个层次都只有一个数据副本,因此采用写回(write-back)的方式.这样,一致性通信只在群集缓存内发生,而不会波及到群集外.Ros 以页面为粒度来划分私有/共享数据.实验结果表明,这种划分方法减少了 12%的片上通信量.

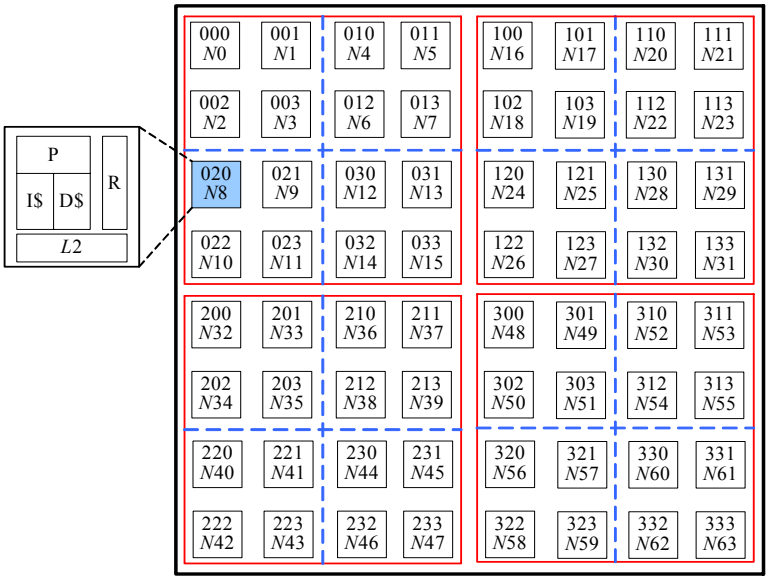


Fig.8 Partitioning of 64-core tiled CMP[44]

图 8 64 核 Tile 结构的逻辑分组[44]

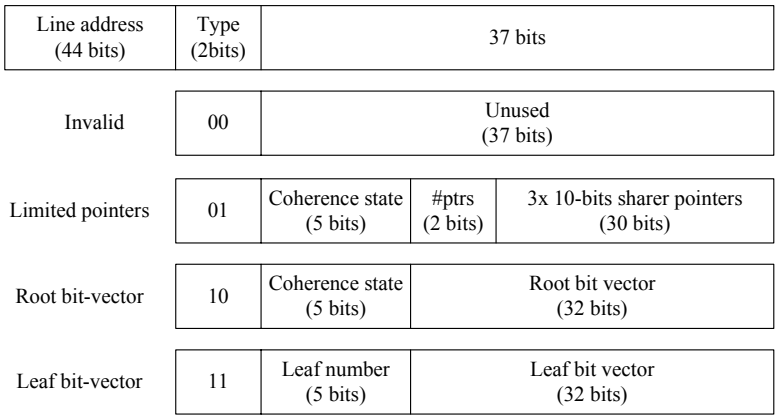


Fig.9 SCD line structure combined pointer and hierarchical bit vector[33]

图 9 SCD 的混合指针和层次化位向量的组织结构[33]

小结.层次目录无论是无效(invalidation)还是间接/降级(indirection/downgrade)功能,都必须逐层进行.这意味着:中间节点的缓存必须能够同时既作为根目录缓存(root directory cache),处理如发送无效、收集确认、间接请求等一致性事务,又作为叶缓存(leaf cache),处理如应答失效等事务.再则,私有/共享数据的划分弱仍然沿用平坦目录协议中的划分方法,维护数据一致性将产生相当大的协议的流量.层次目录结构发掘了分组的局域性.相比平坦的目录结构,层次目录结构降低了全局存储的开销以及片上网络的协议流量,具有较好的可扩展性.

5 存在的问题和进一步研究的方向

步入多核时代后,缓存一致性问题一直是困扰计算机体系结构的难题之一.在多核时代,缓存一致性问题还存在如下一些有待解决的问题.

(1) 目录节点的竞争.由于片上多核处理器每个内核的数据请求都必须汇聚到目录节点,形成了内核对目录节点的竞争.对于大规模的 CMP 来说,目录已经成为系统性能的瓶颈.另外,目录缓存的片上分布方式也是设计者需要认真考虑的问题;

(2) 解决数据冲突.在 2016 年 HPCA 大会上,这个方向的研究是一个热点议题.在硬件事务存储系统及软/硬件结合的事务存储系统中,冲突检测的处理大多依赖于高速缓存一致性模型.因此,扩展高速缓存一致性模型以支持硬件事务存储机制及软/硬件结合的事务存储机制,成为学术界研究事务存储技术必须解决的议题;

(3) 一致性流量问题.由于片上多核处理器复杂的存储体系结构,给多个处理器核间数据通信带来了严重挑战.2-hop 的一致性协议采用 Cache 到 Cache 的直接传输,仅仅在数据缺失的情况下减少了可观的一致性操作数量,然而,当发生写更新操作时,仍然要逐个将新值传输到相应的内核;

(4) 发掘共享数据.从目前的研究成果来看,共享数据的发掘主要通过历史信息预测来实现.预测机制做的好,决策机制就能轻松地得到一个最佳的选择.然而研究表明,找到一个通用的预测逻辑依然是非常困难的.另外,预测逻辑也需要额外的存储空间或者硬件开销,同时增加了实现的软/硬件代价;

(5) 一致性用软件还是硬件实现问题.对于这个问题的争论,Hill 等学者认为,软件管理的缓存一致性将硬件设计的复杂性转移到了软件设计上.硬件管理的缓存一致性以 Tile64 处理器为代表,软件管理的缓存一致性以 Intel 的 SCC[95]处理器为代表.DeNovo[96]是一个软件驱动的硬件缓存一致性协议.DeNovo 在同步点利用软件插入 self-invalidation,这样消除了跟踪共享者的目录开销[94].另外,当多程序负载在多核处理器上运行时,本身就不需要一致性的支持.当多线程负载运行时,由硬件管理的片上多核静态的分区的方法缺乏灵活性.未来的众核处理器设计中,动态可配置一致性 Cache 协议的研究将是一个重要的方向.以 Intel 的 SCC 处理器为代表的软件管理一致性方法更具灵活性[97].总体来说,软件实现和硬件实现各有优劣,硬件方法不够灵活,而软件实现则要考虑效率问题[98,99];

(6) 最近,新型非易失存储器得到了学术界和工业界的广泛关注,为下一代存储技术提供了新的解决方案.新型非易失存储器的出现,必然导致电路组成结构和工作原理的巨大变革,因此,缓存一致性操作将根据不同器件的特点进行设计.例如,文献[100]提出了一种基于缓存一致性协议的自适应刷新策略来减少 STT-RAM 刷新操作的次数.又如,忆阻器(memristor)[101]具有存储密度大、非易失性的特点,还可进行逻辑运算,这意味着可将数据处理和存储电路两者合一.文献[102]利用多级忆阻器构建了片上目录,该文从多级忆阻器的读写出发,结合片上缓存一致性的特点,设计的 SpongeDirectory 目录平衡了存储和时延的代价;

(7) 一致性协议的验证.缓存一致性协议异常复杂,协议的状态空间呈指数级增长,甚至出现爆炸现象[103].如何保证缓存一致性协议的正确性,一直被工业界和学术界所关注.学术上研究所涉及缓存一致性协议在一般结构级上进行建模,仍然属于行为和功能级的验证[104−107];

(8) 仿真平台问题.目前,虽然有多种 Cache 模拟器,但只有 GEM5(GEMS)一种模拟器能够进行 Cache 协议的仿真.GEM5 支持的核数只能达到 64 核,模拟器必然要进行“扩容”,以满足众核时代仿真的需求.未来通过改进实验仿真平台,增加对分组共享层次化 Cache 一致性协议的仿真支持,是一个值得尝试的研究方向.另外,GEM5 的稳定版目前已经能够支持异构多核,然而 CPU-GPU,AMD 的 big.Little[108]平台的一致性协议还不完善.仿真平台的研发也是一个值得探索的领域[109,110].

Hill 等人指出:在众核时代,基于目录的一致性协议还将继续存在[16].随着片上多核处理器规模越来越大,内核分组是必然趋势,相应的缓存也向群集层次方向发展.层次化目录将是一条解决众核一致性问题的有效思路.层次化目录使一致性交互局部化.通过构建多层目录、内核分区域管理和目录压缩等策略来缩短一致性消息延伸的距离,使一致性交互尽可能在小范围内完成.而且在不同的层次,可以灵活选用或者目录或者广播来维护数据状态.未来片上多核处理器的缓存一致性设计,必须综合考虑协议的流量、目录缓存大小、功耗以及性能.层

次化的目录结构正好平衡了这些因素,将是片上多核处理器缓存一致性协议研究的新方向.

6 总 结

片上多核处理器已经成为通用处理器市场的主流.本文总结了近 10 年来计算机体系结构领域三大顶级会议以及其他重要国际会议期刊有关缓存一致性问题的研究成果,从中我们可以观察到:特别是近 5 年来,三大顶级会议对缓存一致性问题的研究已经从多核聚焦到千核规模的片上多核处理器上.因此,本文主要针对基于目录的缓存一致性片上多核处理器,从不同的优化角度进行了分类、归纳和总结,并且指出了进一步的研究方向,以期望国内更多从事体系结构研究的学者加入到这个课题的研究中来.

References:

[1] Zhang YH. The primer of multi-core era. Communication of CCF, 2011,7(4):59−63 (in Chinese with English abstract).

[2] Tendler JM, Dodson JS, Fields JS, Le H, Sinharoy B. POWER4 system microarchitecture. IBM Journal of Research & Development, 2002,46(1):5−25. [doi: 10.1147/rd.461.0005]

[3] Leon AS, Langley B, Shin JL. The UltraSPARC T1 processor: CMT reliability. In: Proc. of the IEEE Custom Integrated Circuits Conf. 2006. San Jose: IEEE, 2006. 555−562. [doi: 10.1109/CICC.2006.320989]

[4] Shah M, Barren J, Brooks J, Golla R. UltraSPARC T2: A highly-treaded, power-efficient, SPARC SOC. In: Proc. of the 2007 IEEE Asian Solid-State Circuits Conf. Jeju, 2007. 22−25. [doi: 10.1109/ASSCC.2007.4425786]

[5] Bell S, Edwards B, Amann J, Conlin R, Joyce K, Leung V, MacKay J, Reif M, Bao LW, Brown J, Mattina M, Miao CC, Ramey C, Wentzlaff D, Anderson W, Berger E, Fairbanks N, Khan D, Montenegro F, Stickney J, Zook J. TILE64—Processor: A 64-Core SoC with mesh interconnect. In: Proc. of the 2008 IEEE Int'l Solid-State Circuits Conf.—Digest of Technical Papers. San Francisco, 2008. 88−598. [doi: 10.1109/ISSCC.2008.4523070]

[6] Asanovic K, Bodik R, Catanzaro BC, Gebis JJ, Husbands P, Keutzer K, Patterson DA, Plishker WL, Shalf J, Williams SW, Yelick KA. The landscape of parallel computing research a view from Berkeley. Technical Report, EECS-2006-183, UC Berkeley, 2006.

[7] Lenoski D, Laudon J, Gharachorloo K, Gupta A, Hennessy J. The directory-based cache coherence protocol for the DASH multiprocessor. In: Proc. of the 17th Annual Int'l Symp. on Computer Architecture. Seattle, 1990. 148-159. [doi: 10.1109/ISCA.1990.134520]

[8] Gharachorloo K, Lenoski D, Laudon J, Gibbons P, Gupta A, Hennessy J. Memory consistency and event ordering in scalable shared-memory multiprocessors. In: Proc. of the 17th Annual Int'l Symp. on Computer Architecture. Seattle, 1990. 15−26. [doi: 10.1109/ISCA.1990.134503]

[9] Hennessy JL, David A. Computer Architecture: A Quantitative Approach. 5th ed., San Francisco: Morgan Kaufmann Publishers Inc., 2011.

[10] Papamarcos MS, Patel JH. A low-overhead coherence solution for multiprocessors with private cache memories. SIGARCH Computer Architecture News, 1984,12(3):348−354. [doi: 10.1145/773453.808204]

[11] Blake G, Dreslinski RG, Mudge T. A survey of multicore processors. IEEE Signal Processing Magazine, 2009,26(6):26−37. [doi: 10.1109/MSP.2009.934110]

[12] Intel Corporation. An introduction to the Intel QuickPath interconnect. Document Number 320412-001US. 2009.

[13] Maddox RA, Singh G, Safranek RJ. Weaving High Performance Multiprocessor Fabric: Architecture Insights into the Intel QuickPath Interconnect. Intel Press, 2009.

[14] Hum HHJ, Goodman JR. Forward State for Use in Cache Coherency in a Multiprocessor System. United States Patent, 6922756, 2005.

[15] Martin MMK, Hill MD, Sorin DJ. Why on-chip cache coherence is here to stay. Communications of the ACM, 2012,55(7):78−89. [doi: 10.1145/2209249.2209269]

[16] Suh T, Blough DM, Lee HHS. Supporting cache coherence in heterogeneous multiprocessor systems. In: Proc. of the Conf. on Design, Automation and Test in Europe (DATE 2004), Vol. 2. Washington: IEEE Computer Society, 2004. 1150−1155. [doi: 10.1109/DATE.2004.1269047]

[17] Ros A, Cuesta B, Fernandez-Pascual R, Gomez ME, Acacio ME, Robles A, Garcia J, Duato J. Extending magny-cours cache coherence. IEEE Trans. on Computers, 2012,61(5):593−606. [doi: 10.1109/TC.2011.65]

[18] Hackenberg D, Molka D, Nagel WE. Comparing cache architectures and coherency protocols on x86-64 multicore SMP systems. In: Proc. of the 42nd Annual IEEE/ACM Int'l Symp. on Microarchitecture (MICRO). New York, 2009. 413−422. [doi: 10.1145/1669112.1669165]

[19] Cheng L, Carter JB, Dai D. An adaptive cache coherence protocol optimized for producer-consumer sharing. In: Proc. of the IEEE 13th Int'l Symp. on High Performance Computer Architecture (HPCA). IEEE, 2007. 328−339. [doi: 10.1109/HPCA.2007.346210]

[20] Hossain H, Dwarkadas S, Huang MC. Improving support for locality and fine-grain sharing in chip multiprocessors. In: Proc. of the 17th Int'l Conf. on Parallel Architectures and Compilation Techniques (PACT 2008). ACM Press, 2008. 155−165. [doi: 10.1145/1454115.1454138]

[21] Cuesta BA, Ros A, Gómez ME, Robles A, Duato JF. Increasing the effectiveness of directory caches by deactivating coherence for private memory blocks. In: Proc. of the 38th Annual Int'l Symp. on Computer Architecture (ISCA). New York: ACM Press, 2011. 93−104. [doi: 10.1145/2000064.2000076]

[22] Cuesta B, Ros A, GöMez ME, Robles A, Duato J. Increasing the effectiveness of directory caches by avoiding the tracking of noncoherent memory blocks. IEEE Trans. on Computers, 2013,62(3):482−495. [doi: 10.1109/TC.2011.241]

[23] Valls JJ, Ros A, Sahuquillo J, Gómez ME, Duato J. PS-Dir: A scalable two-level directory cache. In: Proc. of the 21st Int'l Conf. on Parallel Architectures and Compilation Techniques (PACT). New York: ACM Press, 2012. 451−452. [doi: 10.1145/2370816.2370891]

[24] Valls JJ, Ros A, Sahuquillo J, Gómez ME. PS directory: A scalable multilevel directory cache for CMPs. Journal of Supercomputing, 2015,71(8):1−30. [doi: 10.1007/s11227-014-1332-5]

[25] Ros A, Acacio ME, Garcia JMA. Direct coherence protocol for many-core chip multiprocessors. IEEE Trans. on Parallel & Distributed Systems, 2010,21(12):1779−1792. [doi: 10.1109/TPDS.2010.43]

[26] Qian X, Sahelices B, Torrellas J. OmniOrder: Directory-Based conflict serialization of transactions. In: Proc. of the ACM/IEEE 41st Int'l Symp. on Computer Architecture (ISCA). Minneapolis, 2014. 421−432. [doi: 10.1109/ISCA.2014.6853223]

[27] Park S, Prvulovic M, Hughes CJ. PleaseTM: Enabling transaction conflict management in requester-wins hardware transactional memory. In: Proc. of the 2016 IEEE Int'l Symp. on High Performance Computer Architecture (HPCA). Barcelona, 2016. 285−296. [doi: 10.1109/HPCA.2016.7446072]

[28] Chen S, Peng L. Efficient GPU hardware transactional memory through early conflict resolution. In: Proc. of the 2016 IEEE Int'l Symp. on High Performance Computer Architecture (HPCA). Barcelona, 2016. 274−284. [doi: 10.1109/HPCA.2016.7446071]

[29] Kumar S, Shriraman A, Vedula N. Fusion: Design tradeoffs in coherent cache hierarchies for accelerators. In: Proc. of the 42nd Annual Int'l Symp. on Computer Architecture (ISCA). New York: ACM Press, 2015. 733−745. [doi: 10.1145/2749469.2750421]

[30] Kelm JH, Johnson MR, Lumettta SS, Patel SJ. WAYPOINT: Scaling coherence to thousand-core architectures. In: Proc. of the 19th Int'l Conf. on Parallel Architectures and Compilation Techniques (PACT). ACM Press, 2010. 99−110. [doi: 10.1145/1854273.1854291]

[31] Sanchez D, Kozyrakis C. SCD: A scalable coherence directory with flexible sharer set encoding. In: Proc. of the 2012 IEEE 18th Int'l Symp. on High Performance Computer Architecture (HPCA). IEEE, 2012. 1−12. [doi: 10.1109/HPCA.2012.6168950]

[32] Singh I, Shriraman A, Fung WWL, O'Connor M, Aamodt TM. Cache coherence for GPU architectures. In: Proc. of the 2013 IEEE 19th Int'l Symp. on High Performance Computer Architecture (HPCA). Shenzhen, 2013. 578−590. [doi: 10.1109/HPCA.2013.6522351]

[33] Agarwal N, Nellans D, Ebrahimi E, Wenisch TF, Danskin J, Keckle SW. Selective GPU caches to eliminate CPU-GPU HW cache coherence. In: Proc. of the 2016 IEEE Int'l Symp. on High Performance Computer Architecture (HPCA). Barcelona, 2016. 494−506. [doi: 10.1109/HPCA.2016.7446089]

[34] Power J, Basu A, Gu JL, Puthoor S, Beckmann BM, Hill MD, Reinhardt SK, Wood DA. Heterogeneous system coherence for integrated CPU-GPU systems. In: Proc. of the 46th Annual IEEE/ACM Int'l Symp. on Microarchitecture (MICRO). New York: ACM Press, 2013. 457−467. [doi: 10.1145/2540708.2540747]

[35] Komuravelli R, Sinclair MD, Alsop J, Huzaifa M, Kotsifakou M, Srivastava P, Adve SV, Adve VS. Stash: Have your scratchpad and cache it too. In: Proc. of the 2015 ACM/IEEE 42nd Annual Int'l Symp. on Computer Architecture (ISCA). Portland, 2015. 707−719. [doi: 10.1145/2749469.2750374]

[36] Fan DR, Yuan N, Zhang JC, Zhou YB, Lin W, Song FL, Ye XC, Huang H, Yu L, Long GP, Zhang H, Liu L. Godson-T: An efficient many-core architecture for parallel program executions. Journal of Computer Science and Technology, 2009,24(6):1061−1073. [doi: 10.1007/s11390-009-9295-3]

[37] Xu WZ, Song FL, Liu ZY, Fan DR, Yu L, Zhang S. On synchronization and evaluation method of chipped many-core processor. Chinese Journal of Computers, 2010,33(10):1777−1787 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2010.01777]

[38] Bao E, Li W, Fan D, Yang Y, Ma X. An optimization of broadcast on Godson-T many-core system architecture. Journal of Computer Research and Development, 2010,47(3):524−531 (in Chinese with English abstract).

[39] Huang H, Liu L, Song FL, Ma XY. Architecture supported synchronization-based cache coherence protocol for many-core processors. Chinese Journal of Computer, 2009,32(8):1618−1630 (in Chinese with English abstract).

[40] Li GM. Cache coherence techniques for chip multiprocessor architecture [Ph.D. Thesis]. Hefei: University of Science and Technology of China, 2013 (in Chinese with English abstract).

[41] Zhang J, Tian Z, Mei KZ, Zhao JZ. Node predicting based direct cache coherence protocol for chip multi-processor. Chinese Journal of Computers, 2014,37(03):700−720 (in Chinese with English abstract).

[42] Guo SL, Wang HX, Xue YB, Li CM, Wang DS. Hierarchical cache directory for CMP. Journal of Computer Science & Technology, 2010,25(2):246−256. [doi: 10.1007/s11390-010-9321-5]

[43] Li C, Wang H, Xue Y, Zhang X, Wang D. Fast hierarchical cache directory: A scalable cache organization for large-scale CMP. In: Proc. of the 2013 IEEE 8th Int'l Conf. on Networking, Architecture and Storage. IEEE, 2010. 367−376. [doi: 10.1109/NAS.2010.25]

[44] Binkert NL, Dreslinski RG, Hsu LR, Lim KT, Saidi AG, Reinhardt SK. The M5 simulator: Modeling networked systems. IEEE Micro, 2006,26(4):52−60. [doi: 10.1109/MM.2006.82]

[45] Magnusson PS, Christensson M, Eskilson J, Forsgren D, Hållberg G, Högberg J, Larsson F, Moestedt A, Werner B. Simics: A full system simulation platform. Computer, 2002,35(2):50−58. [doi: 10.1109/2.982916]

[46] Martin MMK, Sorin DJ, Beckmann BM, Marty MR, Xu M, Alameldeen AR, Moore KE, Hill MD, Wood DA. Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset. SIGARCH Computer Architecture News, 2005,33,4:92−99. [doi: 10.1145/1105734.1105747]

[47] Binkert N, Beckmann B, Black G, Reinhardt SK, Saidi A, Basu A, Hestness J, Hower DR, Krishna T, Sardashti S, Sen R, Sewell K, Shoaib M, Vaish N, Hill MD, Wood DA. The GEM5 simulator. SIGARCH Computer Architecture News, 2011,39(2):1−7. [doi: 10.1145/2024716.2024718]

[48] Endo FA, Couroussé D, Charles HP. Micro-Architectural simulation of in-order and out-of-order ARM microprocessors with GEM5. In: Proc. of the 2014 Int'l Conf. on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIV). Agios Konstantinos, 2014. 266−273. [doi: 10.1109/SAMOS.2014.6893220]

[49] General memory system 2015. 2015. http://www.gem5.org/General_Memory_System

[50] Alisafaee M. Spatiotemporal coherence tracking. In: Proc. of the 45th Annual IEEE/ACM Int'l Symp. on Microarchitecture (MICRO-45). Washington: IEEE Computer Society, 2012. 341−350. [doi: 10.1109/MICRO.2012.39]

[51] Kurian G, Khan O, Devadas S. The locality-aware adaptive cache coherence protocol. In: Proc. of the 40th Annual Int'l Symp. on Computer Architecture (ISCA). New York: ACM Press, 2013. 523−534. [doi: 10.1145/2485922.2485967]

[52] Zhao HZ, Shriraman A, Kumar S, Dwarkadas S. Protozoa: Adaptive granularity cache coherence. In: Proc. of the 40th Annual Int'l Symp. on Computer Architecture (ISCA). New York: ACM Press, 2013. 547−558. [doi: 10.1145/2485922.2485969]

[53] Luo L, Sriraman A, Fugate B, Hu S. LASER: Light, accurate sharing detection and repair. In: Proc. of the 2016 IEEE Int'l Symp. on High Performance Computer Architecture (HPCA). Barcelona, 2016. 261−273. [doi: 10.1109/HPCA.2016.7446070]

[54] Zhang GW, Horn W, Sanchez D. Exploiting commutativity to reduce the cost of updates to shared data in cache-coherent systems. In: Proc. of the 48th Int'l Symp. on Microarchitecture (MICRO-48). New York: ACM Press, 2015. 13−25. [doi: 10.1145/2830772.2830774]

[55] Badr M, Jerger NE. SynFull: Synthetic traffic models capturing cache coherent behaviour. In: Proc. of the 2014 ACM/IEEE 41st Int'l Symp. on Computer Architecture (ISCA). Minneapolis, 2014. 109−120. [doi: 10.1109/ISCA.2014.6853236]

[56] Kim C, Burger D, Keckler SW. An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches. ACM SIGPLAN Notices, 2002,37(10):211−222. [doi: 10.1145/605397.605420]

[57] Manivannan M, Stenstrom P. Runtime-Guided cache coherence optimizations in multi-core architectures. In: Proc. of the 2014 IEEE 28th Int'l Parallel and Distributed Processing Symp. Phoenix, 2014. 625−636. [doi: 10.1109/IPDPS.2014.71]

[58] Demetriades S, Cho S. Predicting coherence communication by tracking synchronization points at run time. In: Proc. of the 2012 45th Annual IEEE/ACM Int'l Symp. on Microarchitecture (MICRO-45). Washington: IEEE Computer Society, 2012. 351−362. [doi: 10.1109/MICRO.2012.40]

[59] Marty MR, Hill MD. Virtual hierarchies to support server consolidation. In: Proc. of the 34th Annual Int'l Symp. on Computer Architecture (ISCA 2007). New York: ACM Press, 2007. 46−56. [doi: 10.1145/1250662.1250670]

[60] Barroso LA, Gharachorloo K, Mcnamara R, Nowatzyk A, Qadeer S, Sano B, Smith S, Stets R, Verghese B. Piranha: A scalable architecture based on single-chip multiprocessing. ACM SIGARCH Computer Architecture News, 2000,28(2):282–293. [doi: 10.1145/342001.339696]

[61] Bienia C, Kumar S, Singh JP, Li K. The PARSEC benchmark suite: Characterization and architectural implications. In: Proc. of the 17th Int'l Conf. on Parallel Architectures and Compilation Techniques (PACT 2008). New York: ACM Press, 2008. 72–81. [doi: 10.1145/1454115.1454128]

[62] Bailey D, Barszcz E, Barton J, Browning D, Carter R, Dagum L, Fatoohi R, Fineberg S, Frederickson P, Lasinski T, Schreiber R, Simon H, Venkatakrishnan V, Weeratunga S. The NAS parallel benchmarks. Int'l Journal of Supercomputer Applications, 1991, 5(3):63–73. [doi: 10.1177/109434209100500306]

[63] Singh JP, Gupta A, Ohara M, Torrie E, Woo SC. The SPLASH-2 programs: Characterization and methodical considerations. In: Proc. of the 22nd Annual Int'l Symp. on Computer Architecture (ISCA'95). New York: ACM Press, 1995. 24–36. [doi: 10.1145/225830.223990]

[64] Pugsley SH, Spjut JB, Nellans DW, Balasubramonian R. SWEL: Hardware cache coherence protocols to map shared data onto shared caches. In: Proc. of the Int'l Conf. on Parallel Architectures and Compilation Techniques (PACT). 2010. 465–476. [doi: 10.1145/1854273.1854331]

[65] Sorin DJ, Hill MD, Wood DA. A primer on memory consistency and cache coherence. In: Hill MD, ed. Proc. of the Synthesis Lectures on Computer Architecture. Morgan & Claypool Publishers, 2011.

[66] Gupta A, Weber WD, Mowry T. Reducing memory and traffic requirements for scalable directory-based cache coherence schemes. In: Proc. of the Scalable Shared Memory Multiprocessors. Springer US, 1992. 167–192. [doi: 10.1007/978-1-4615-3604-8_9]

[67] Ros A, Cuesta B, Gomez ME, Robles A, Duato J. Temporal-Aware mechanism to detect private data in chip multiprocessors. In: Proc. of the 42nd Int'l Conf. on Parallel Processing. IEEE Computer Society, 2013. 562–571. [doi: 10.1109/ICPP.2013.70]

[68] Demetriades S, Cho S. Stash directory: A scalable directory for many-core coherence. In: Proc. of the 2014 IEEE 20th Int'l Symp. on High Performance Computer Architecture (HPCA). Orlando, 2014. 177–188. [doi: 10.1109/HPCA.2014.6835928]

[69] Reinhardt SK, Hill M, Beckmann BM, Basu A. CMP directory coherence: One granularity does not fit all. Technical Report #CS-TR-2013-1798, University of Wisconsin Computer Sciences, 2013.

[70] Hardavellas N, Ferdman M, Falsafi B, Ailamaki A. Reactive NUCA: Near-Optimal block placement and replication in distributed caches. In: Proc. of the 36th Annual Int'l Symp. on Computer Architecture (ISCA 2009). 2009. 184–195. [doi: 10.1145/1555815.1555779]

[71] Hossain H, Dwarkadas S, Huang MC. POPS: Coherence protocol optimization for both private and shared data. In: Proc. of the Int'l Conf. on Parallel Architectures & Compilation Techniques (PACT). 2011. 45–55. [doi: 10.1109/PACT.2011.11]

[72] Zebchuk J, Falsafi B, Moshovos A. Multi-Grain coherence directories. In: Proc. of the 46th Annual IEEE/ACM Int'l Symp. on Microarchitecture (MICRO-46). New York: ACM Press, 2013. 359–370. [doi: 10.1145/2540708.2540739]

[73] Zhao H, Shriraman A, Dwarkadas S. Sharing Pattern-Based Directory Coherence for Multicore Scalability ("SPACE"). US Patent, 20140032848, 2014.

[74] Zhao HZ, Shriraman A, Dwarkadas S. SPACE: Sharing pattern-based directory coherence for multicore scalability. In: Proc. of the Int'l Conf. on Parallel Architectures and Compilation Techniques. New York: ACM, 2010. 135–146. [doi: 10.1145/1854273.1854294]

[75] Buehrer G, Parthasarathy S, Chen Y. Adaptive parallel graph mining for CMP architectures. In: Proc. of the 6th Int'l Conf. on Data Mining. 2006. 97–106. [doi: 10.1109/ICDM.2006.15]

[76] Alameldeen AR, Martin MMK, Mauer CJ, Moore KE, Xu M, Hill MD, Wood DA, Sorin DJ. Simulating a $2M commercial server on a $2K PC. Computer, 2003,36(2):50–57. [doi: 10.1109/MC.2003.1178046]

[77] Menezo LG, Puente V, Gregorio JA. Flask coherence: A morphable hybrid coherence protocol to balance energy, performance and scalability. In: Proc. of the IEEE Int'l Symp. on High Performance Computer Architecture (HPCA). IEEE, 2015. 198–209. [doi: 10.1109/HPCA.2015.7056033]

[78] AMD. Software Optimization Guide for AMD Family 10h Processors. 3.11 ed. 2009.

[79] Zilles C, Rajwar R. Transactional memory and the birthday paradox. In: Proc. of the 19th Annual ACM Symp. on Parallel Algorithms and Architectures (SPAA 2007). New York: ACM Press, 2007. 303–304. [doi: 10.1145/1248377.1248428]

[80] Zebchuk J, Srinivasan V, Qureshi MK, Moshovos A. A tagless coherence directory. In: Proc. of the 42nd Annual IEEE/ACM Int'l Symp. on Microarchitecture. ACM Press, 2009. 423–434. [doi: 10.1145/1669112.1669166]

[81] Ferdman M, Lotfi-Kamran P, Balet K, Falsafi B. Cuckoo directory: A scalable directory for many-core systems. In: Proc. of the 2013 IEEE 19th Int'l Symp. on High Performance Computer Architecture (HPCA). IEEE, 2011. 169−180. [doi: 10.1109/HPCA.2011.5749726]

[82] Pagh R, Rodler FF. Cuckoo hashing. Journal of Algorithms, 2004,51(2):122−144. [doi: 10.1016/j.jalgor.2003.12.002]

[83] Jerger NDE, Peh LS, Lipasti MH. Virtual tree coherence: Leveraging regions and in-network multicast trees for scalable cache coherence. In: Proc. of the 45th Annual IEEE/ACM Int'l Symp. on Microarchitecture. IEEE, 2008. 35−46. [doi: 10.1109/MICRO.2008.4771777]

[84] Jerger NE, Peh LS, Lipasti M. Virtual circuit tree multicasting: A case for on-chip hardware multicast support. In: Proc. of the Int'l Symp. on Computer Architecture (ISCA-35), Vol. 36. 2008. 229−240. [doi: 10.1145/1394608.1382141]

[85] Butler M, Barnes L, Sarma DD, Gelinas B. Bulldozer: An approach to multithreaded compute performance. IEEE Micro, 2011, 31(2):6−15. [doi: 10.1109/MM.2011.23]

[86] Jotwani R, Sundaram S, Kosonocky S, Schaefer A. An x86-64 core implemented in 32nm SOI CMOS. In: Proc. of the 2010 IEEE Int'l Solid-State Circuits Conf. on Digest of Technical Papers (ISSCC). IEEE, 2010. 106−107. [doi: 10.1109/ISSCC.2010.5434076]

[87] Patterson D. The top 10 innovations in the new NVIDIA Fermi architecture, and the top 3 next challenges. Nvidia Whitepaper, 2009.

[88] Nickolls J, Dally WJ. The GPU computing era. IEEE Micro, 2010,30(2):56−69. [doi: 10.1109/MM.2010.41]

[89] Fu YS, Nguyen TM, Wentzlaff D. Coherence domain restriction on large scale systems. In: Proc. of the 48th Int'l Symp. on Microarchitecture (MICRO-48). New York: ACM Press, 2015. 686−698. [doi: 10.1145/2830772.2830832]

[90] Acacio ME, Gonzalez J, Garcia JM, Duato J. A two-level directory architecture for highly scalable cc-NUMA multiprocessors. IEEE Trans. on Parallel & Distributed Systems, 2005,16(1):67−79. [doi: 10.1109/TPDS.2005.4]

[91] Sanchez D, Kozyrakis C. The ZCache: Decoupling ways and associativity. In: Proc. of the 43rd Annual IEEE/ACM Int'l Symp. on Microarchitecture. IEEE Computer Society, 2010. 187−198. [doi: 10.1109/MICRO.2010.20]

[92] Ros A, Davari M, Kaxiras S. Hierarchical private/shared classification: The key to simple and efficient coherence for clustered cache hierarchies. In: Proc. of the 2015 IEEE 21st Int'l Symp. on High Performance Computer Architecture (HPCA). IEEE Computer Society, 2015. 186−197. [doi: 10.1109/HPCA.2015.7056032]

[93] Cuesta B, Ros A, Gómez ME, Robles A, Duato J. Increasing the effectiveness of directory caches by avoiding the tracking of noncoherent memory blocks. IEEE Trans. on Computers, 2013,62(3):482−495. [doi: 10.1109/TC.2011.241]

[94] Sung H, Komuravelli R, Adve SV. DeNovoND: Efficient hardware for disciplined nondeterminism. IEEE Micro, 2014,34(3): 138−148. [doi: 10.1109/MM.2014.5]

[95] Baron BM. The single-chip cloud computer. Microprocessor Report, 2010.

[96] Choi B, Komuravelli R, Sung H, Smolinski R, Honarmand N, Adve SV, Adve VS, Carter NP, Chou C-T . DeNovo: Rethinking the memory hierarchy for disciplined parallelism. In: Proc. of the 2011 Int'l Conf. on Parallel Architectures and Compilation Techniques (PACT). Galveston, 2011. 155−166. [doi: 10.1109/PACT.2011.21]

[97] Gries M, Hoffmann U, Konow M, Riepen M. SCC: A flexible architecture for many-core platform research. Computing in Science & Engineering, 2011,13(6):79−83. [doi: 10.1109/MCSE.2011.109]

[98] Ashby TJ, Diaz P, Cintra M. Software-Based cache coherence with hardware-assisted selective self-invalidations using bloom filters. IEEE Trans. on Computers, 2011,60(4):472−483. [doi: 10.1109/TC.2010.155]

[99] Alvarez L, Vilanova L, Gonzàlez M, Martorell X, Navarro N, Ayguadé E. Hardware-Software coherence protocol for the coexistence of caches and local memories. IEEE Trans. on Computers, 2015,64(1):152−165. [doi: 10.1109/TC.2013.194]

[100] Li JH, Shi L, Li Q, Xue CJ, Chen YR, Xu YL, Wang W. Low-Energy volatile STT-RAM cache design using cache-coherence-enabled adaptive refresh. ACM Trans. on Design Automation of Electronic Systems, 2013,19(1):1−23. [doi: 10.1145/2534393]

[101] Strukov DB, Snider GS, Stewart DR, Williams RS. The missing memristor found. Nature, 2008,453(5):80−83. [doi: 10.1038/nature06932]

[102] Zhang LK, Strukov D, Saadeldeen H, Fan DR, Zhang MZ, Franklin D. SpongeDirectory: Flexible sparse directories utilizing multi-level memristors. In: Proc. of the 23rd Int'l Conf. on Parallel architectures and compilation (PACT 2014). New York: ACM Press, 2014. 61−74. [doi: 10.1145/2628071.2628081]

[103] Manerkar YA, Lustig D, Pellauer M, Martonosi M. CCICheck: Using μHB graphs to verify the coherence-consistency interface. In: Proc. of the 48th Int'l Symp. on Microarchitecture (MICRO-48). New York: ACM Press, 2015. 26−37. [doi: 10.1145/2830772.2830782]

[104] Voskuilen G, Vijaykumar TN. Fractal++: Closing the performance gap between fractal and conventional coherence. In: Proc. of the 2014 ACM/IEEE 41st Int'l Symp. on Computer Architecture (ISCA). Minneapolis, 2014. 409–420. [doi: 10.1109/ISCA.2014.6853211]

[105] Voskuilen G, Vijaykumar T. High-Performance fractal coherence. In: Proc. of the 19th Int'l Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2014). New York: ACM Press, 2014. 701–714. [doi: 10.1145/2541940.2541982]

[106] Zhang M, Lebeck AR, Sorin DJ. Fractal coherence: Scalably verifiable cache coherence. In: Proc. of the 43rd Annual IEEE/ACM Int'l Symp. on Microarchitecture (MICRO). 2010. 471–482. [doi: 10.1109/MICRO.2010.11]

[107] Beu JG, Poovey JA, Hein ER, Conte TM. High-Speed formal verification of heterogeneous coherence hierarchies. In: Proc. of the 2013 IEEE 19th Int'l Symp. on High Performance Computer Architecture (HPCA 2013). Shenzhen, 2013. 566–577. [doi: 10.1109/HPCA.2013.6522350]

[108] Padoin EL, Pilla LL, Castro M, Boito FZ. Performance/Energy trade-off in scientific computing: The case of ARM big.LITTLE and Intel Sandy Bridge. IET Computers & Digital Techniques, 2015,9(1):27–35. [doi: 10.1049/iet-cdt.2014.0074]

[109] Endo FA, Couroussé D, Charles HP. Micro-Architectural simulation of embedded core heterogeneity with GEM5 and McPAT. In: Proc. of the 2015 Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools (RAPIDO 2015). New York: ACM Press, 2015. 1–6. [doi: 10.1145/2693433.2693440]

[110] Butko A, Gamatié A, Sassatelli G, Torres L, Robert M. Design exploration for next generation high-performance manycore on-chip systems: Application to big.LITTLE architectures. In: Proc. of the 2015 IEEE Computer Society Annual Symp. on VLSI. Montpellier, 2015. 551–556. [doi: 10.1109/ISVLSI.2015.28]

附中文参考文献:

[1] 张悠慧.多核时代处理器设计初探.中国计算机学会通讯,2011,7(4):59–63.

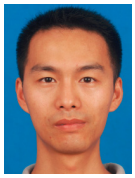
[37] 徐卫志,宋风龙,刘志勇,范东睿,余磊,张帅.众核处理器片上同步机制和评估方法研究.计算机学报,2010,33(10):1777–1787. [doi: 10.3724/SP.J.1016.2010.01777]

[38] 包尔固德,李伟生,范东睿,杨扬,马啸宇.Godson-T 众核体系结构上的 Broadcast 性能优化.计算机研究与发展,2010,47(3):524–531.

[39] 黄河,刘磊,宋风龙,马啸宇.硬件结构支持的基于同步的高速缓存一致性协议.计算机学报,2009,32(8):1618–1630.

[40] 李功明.片上多处理器体系结构中 Cache 一致性模型研究[博士学位论文].合肥:中国科学技术大学,2013.

[41] 张骏,田泽,梅魁志,赵季中.基于节点预测的直接一致性协议.计算机学报,2014,37(3):700–720.



胡森森(1979－),男,湖北荆州人,博士生,主要研究领域为计算机体系结构,高性能嵌入式系统,并行计算.

陈旭(1983－),男,博士生,主要研究领域为计算机体系结构,高性能嵌入式系统,并行计算.

计卫星(1980－),男,博士,副教授,CCF 专业会员,主要研究领域为计算机体系结构,并行程序设计,嵌入式计算.

付文飞(1991－),女,硕士生,主要研究领域为计算机体系结构,高性能嵌入式系统,并行计算.

王一拙(1979－),男,博士,讲师,CCF 专业会员,主要研究领域为计算机体系结构,并行程序设计.

石峰(1961－),男,博士,教授,博士生导师,主要研究领域为多核处理器体系结构,并行与分布式计算.