doi:10.3772/j.issn.1002-0470.2015.05.002

一种无目录的共享高速缓存一致性协议①

刘道福②*,**,*** 陈天石*,** 郭 琦*,**

(*计算机体系结构国家重点实验室 北京 100190)

(**中国科学院计算技术研究所 北京 100190)

(***中国科学院研究生院 北京 100049)

摘 要 针对使用目录记录各共享缓存块在各核心的私有备份信息的多核和众核并行系统共享高速缓存一致性协议因使用目录造成性能下降的问题进行了研究。研究发现,实际应用的多核和众核系统可以不存储共享缓存块的共享信息,因为多核和众核系统大都采用弱一致性协议,根据这种协议,某个核心的写操作不需要立即被其他核心观察到,可以延迟到下一个同步点观察到。基于这一发现,提出了一种不用记录共享信息的无目录的(DirectoryLess)共享高速缓存(Shared cache)一致性协议,简称 DLS 协议。该协议通过在同步点对不确定是否被其他核心更改的缓存块主动无效的方法,在不需要存储共享信息的目录的情况下来保证多核系统符合弱一致性。用并行程序测试集 SPLASH-2 对一个16 核处理器进行了试验,试验结果表明,相比基于目录的 MESI 协议,DLS 不仅可以完全消除目录及其电路面积,而且可平均提高 11.08% 的程序性能,减少 28.83% 的片上网络通讯,以及减少 15.65% 的功耗。而这一切,只需要改变处理器的设计,并不需要改变编程语言和编译器,因此,该协议无需更改或重新编译即可以兼容现有的代码。

关键词 存储一致性, 高速缓存一致性协议, 多核/众核系统, 弱一致性

0 引 言

随着大规模集成电路工艺的发展,片内能集成的晶体管数目越来越多,片内多核处理器(chip multi-processor,CMP)乃至众核处理器已成为主流处理器,代表产品有英特尔(Intel)公司 2012 年推出的 50 核 Phi 众核处理器[1],AMD 公司 2011 年推出的 16 核的 Opteron 6200 处理器[2],ARM 公司 2012 年推出的 64 核 Centip3De 处理器[3]。随着工艺的发展,未来的处理器甚至能在片内集成 1000 个核心。片内多核处理器(CMP)往往具有复杂的高速缓存(Cache)层次,具体来说,其通常由一个大容量的最后一级高速缓存(Last-level Cache,LLC)以及第一级高速缓存(L1 Cache,L1C)组成。LLC 一般是所有核心共享的,而 L1C 则是每个核心私有的,一般来说 L1C 上的数据是 LLC 上数据的子集的备份。各级 Cache 之间的数据一致性是通过高速缓存一致性协议来保证的。本文在研究现有高速缓存一致性协议性能的基础上,提出了一种不用记录共享信息的无目录的共享高速缓存一致性协议(coherence protocol for DirectoryLess Shared cache, DLS),试验表明,该协议的性能明显优于现有协议。

1 相关研究

高速缓存一致性协议有两类:基于广播的一致

① 国家自然科学基金(61100163, 61133004, 61222204, 61221062, 61303158, 61432016, 61472396, 61473275),863 计划(2012AA012202),中国科学院战略性先导科技专项(XDA06010403)及中国科学院国际合作(171111KYSB20130002)资助项目。

② 男,1988 年生,博士生;研究方向:计算机系统结构,异构并行计算;联系人,E-mail: liudaofu@ict.ac.cn

(收稿日期:2014-12-15)

性协议[4]和基于目录的一致性协议[5]。基于广播的一致性协议的每次独占写操作(RdEx)的请求会广播给所有处理器核心(无论其是否有该块的备份),从而通知所有核心无效其私有备份(假如有)。随着处理器核心数目的增加,广播需要的成本越来越高,因此,无论是学术界还是工业界,都倾向于使用基于目录的一致性协议[6,7]。基于目录的一致性协议的基本思想是使用一个专门的目录用来记录每个 LLC 缓存块在 L1C 的备份情况。一般来说,每个块的目录都是一个位宽等于处理器核数的位向量,当某个处理器核(L1C)有该 LLC 缓存块备份时,则对应位的位向量为 1。基于目录的一致性协议在某个核心发生独占写(RdEx)操作时,LLC 会根据目录信息,向对应目录位向量为 1 的(即含有该块备份的)核心的 L1C 发去无效的信息,对应 L1C 收到无效信息后会将其私有备份无效。通过这种方法,所有独占写操作都能马上被所有核心全局观察到,从而保证了高速缓存的一致性。

然而,基于目录的缓存一致性协议有两个缺陷。第一个缺陷是目录位向量的位宽随着核心数目的增加急剧增加,会消耗大量而宝贵的片上 RAM 面积和功耗。比如,对于一个 256 核的 CMP,假如其缓存块大小(cache block size)是 256 位,则每个缓存块的目录也是 256 位,因此,其目录就占了所有高速缓存 RAM 的一半,目录浪费了大量的 RAM 面积功耗。第二个缺陷是会导致大量的写无效消息和写回应消息,这些消息不仅会大大增加片上网络的功耗,而且可能造成片上网络拥堵,从而降低性能。

本文针对上述问题进行了研究,研究发现,在实际大量应用的多核/众核系统中,可以不存储共享缓存块的共享信息。因为现在大部分多核/众核系统采用的一致性协议都是弱一致性或者更松的一致性协议,根据弱一致性协议,某个核心的写操作允许不立即被其他核心观察到,直到同步点才被其他核心观察到。基于这个发现,我们提出了一种不用记录共享信息的无目录的一致性协议(DLS)。该协议通过在同步点对不确定是否被其他核心更改的缓存快主动无效的方法,在不需要存储共享信息的目录的情况下来保证多核系统符合弱一致性。该协议的有效性已通过试验得到了验证。

2 DLS 协议的基本思想

本文提出的一致性协议 DLS 是针对现代处理器中最广泛的弱一致性存储模型,弱一致性协议[8]由 Dubois 于 1986 年提出,其具体内容如下:

在一个多处理器系统中,当且仅当(1)访问全局的同步变量是顺序一致的;(2)在所有非同步变量写操作被全局观察到之前,不允许访问任何同步变量;(3)前一个同步写操作被全局观察到之前,不允许访问任何非同步变量,访存操作符合弱一致性。

相比于顺序一致性需要保证所有访存操作被所有处理器核心以相同的顺序观察到,弱一致性只保证同步变量以相同的顺序被观察到(即符合顺序一致性),对于非同步变量弱一致性协议允许其延时到同步点才被其他处理器核心观察到。

因此,在弱一致性存储模型中,非同步变量(普通变量)的写操作并不需要立即被全局观察到,而是可以延迟到同步点才被其他处理器观察到。从而,任何一个写操作都广播或者根据目录发送写无效消息是一个非必要的操作,写无效的操作可以延迟到同步点进行。

基于此观察,本文提出了一种新的无目录的高速缓存一致性协议 DLS,其基本思想如下:

(1)将写操作中的写无效延迟到同步点。

(2)不通过目录来无效,而是在同步点,每个处理器核心 L1C 根据每个块的状态自无效。具体来说,在同步点,对于 L1C 中的共享状态(详见第 3 节的状态介绍)的块都无效。

(3)为了避免(2)中自无效导致很多没被更新(没有其他处理器写过的)L1C 块也被无效,无效时采用猜测机制,并不直接无效,而是将其标志成可疑块(详见第 3 节的状态介绍),对可疑块的访问采用猜测执行的机制。当访问可疑块时,在进行猜测执行的同时访问 LLC,将缓存块的最新值取回来,并和猜测所用块比较,假如一致,则猜测执行正确,直接继续。否则,则猜测执行错误,中止执行并回滚用最新值重新执行。

DLS 不仅可以不通过广播的方法去除目录,从而大大减少目录的面积和功耗,而且可以将所有普通写操作中不必要的写无效消息和无效返回消息都消除掉,从而大大减少片上网络通讯,降低片上网络的功耗和拥堵,提高整个处理器的性能。

3 DLS 协议的实现

传统的共享缓存一致性协议主要有 MESI[9] 和 MSI[10],其中 MSI 是 MESI 的简化版本。MESI 协议主要包括四个状态,即修改(modified,M)、独占(exclusive,E)、共享(shared,S)和无效(invalid,I),其中 MSI 相比 MESI 少了独占状态(E)。

DLS 可以通过对传统的基于目录的 MESI 协议进行轻量级改动来实现。DLS 对 MESI 的改动有两方面。第一方面去除了 LLC 目录,改为了一个属主 ID;第二方面是为了猜测执行,增加一个可疑状态(SUS 状态)。

3.1 DLS 缓存状态

在 DLS 协议下,对于第一级高速缓存(L1C),缓存块的状态有 5 个:无效状态(invalid, INV),表示该块的数据确定不是最新的;共享状态(shard, SHD),表示该块是通过读操作从最后一级高速缓存(LLC)获取的备份,且暂时没被其他写操作无效;独占状态(exclusive,EXC),表示该块是通过写操作获取的备份;更改状态(modified,MOD),表示 L1C 通过写操作获得 EXC 块之后,继续写该块的缓存块状态;可疑状态(suspicious,SUS),表示一个共享状态的 L1C 块不确定是否被其他处理器的写操作更新过。在前面的 5 个状态中,其中前 4 个都是 MESI 缓存一致性协议中有的,且意义也相近。

3.2 DLS 网络消息

在 DLS 协议下,大部分用于保证一致性的网络消息都和与 MESI 协议的对应消息类似。比如,无论是 DLS 协议还是 MESI 协议,对于一个读操作,L1C 发生缓存缺失(cache miss)时,L1C 会发送一个读请求(Read)给 LLC;对于一个写操作,L1C 发生缓存缺失时,L1C 会发送一个读独占(RdEx)请求给 LLC;对于 Read 请求,LLC 将返回一个共享回应(RepShd)消息,并同时返回一个共享(SHD)块给 L1C;对于 RdEx 请求,LLC 将返回一个独占回应(RepExc)消息,并同时返回一个独占(EXC)块给 L1C。当 LLC 收到 L1C 的一个 Read 请求时,假如 LLC 上缓存块的状态是 MOD 状态,则 LLC 发送一个共享干预(ShdIntervention)请求给 L1C;当 LLC 收到 L1C 的一个 RdEx 请求时,假如 LLC 缓存块的状态是 EXC/MOD 状态,则 LLC 会给 L1C 发送一个独占干预(ExcIntervention)消息。

DLS 协议和 MESI 协议的最显著区别是 DLS 没有无效(Invalidation)消息以及无效应答(Ack)消息。这两个消息是某个 L1C 写操作时无效其他 L1C 上的 SHD 块的请求和应答消息。另外,在 DLS 下,ShdIntervention 消息会将 L1C 上的 MOD 块转成 EXC 状态(而在 MESI 下转成了 SHD 状态);ExcIntervention 消息将 L1C 上的 EXC/MOD 块转成了 SHD 状态(而在 MESI 下转成了 INV 状态),DLS 协议允许一个 LLC 块在有不多于一个 EXC/MOD L1C 块的同时有多个 SHD L1C 块。

3.3 DLS 状态转化

图 1 比较了在 DLS 协议和 MESI 协议下 L1C 的缓存状态转化。其中图 1(a)是 MESI 的 L1C 状态转化图,图 1(b)是 DLS 的 L1C 状态转化图。

如图 1(b)所示,当 LLC 上对应块有属主时,INV 的 L1C 块会被 Read 操作转化成 SHD 块,当 LLC 上对应块没有属主时,INV 的 L1C 块会被 Read 操作转化成 EXC 块(这是为了保证任何一个块都至少有一个属主,LLC 缓存块的属主属于最近写过该块的 L1C 或者在没有 L1C 写过该块情况下第一次读该块的 L1C)。L1C 上的 INV 块也可以直接被 RdEx 操作转成 EXC 状态。

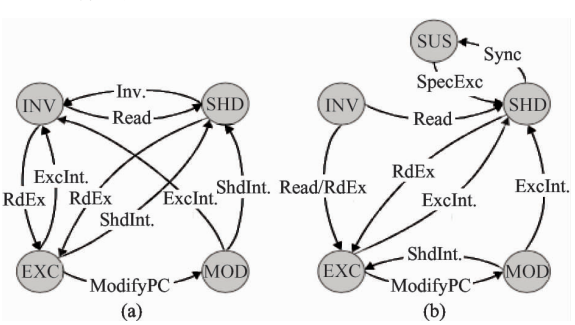

图 1 MESI 和 DLS 状态转化图比较

如果遇到 RdEx 操作,L1C 中的 SHD 块会转成 EXC 状态;如果碰到同步操作(Sync),则 SHD 块会转成 SUS 块,表示该块可能在同步点之前被其他块修改过。

如果遇到写操作,L1C 的 EXC 块会转成 MOD 状态;如果收到一个 ExcIntervention 消息,则 EXC 块会转成 SHD 状态。

如果收到 ShdIntervention 消息,L1C 的 MOD 块会转成 EXC 状态,如果收到 ExcIntervention 消息,则转成 SHD 状态。

当一个 SUS 遇到读操作时,会转化成 SHD 状态(假如 SUS 块的数据是最新的,则直接转化,假如 SUS 块的数据不是最新的,则从 LLC 取回最新块后再转成 SHD 状态)。

在 DLS 协议下,LLC 的状态转化相对简单。遇到一个 Read/RdEx 操作,INV 的 LLC 块会转化成 EXC。当某个 L1C 发生对 EXC 块的写操作时,EXC 的 LLC 块会转成 MOD 状态。当收到 Read/RdEx 请求时,MOD 的 LLC 块会转成 EXC 状态。

4　试验和结果

为了验证 DLS 协议在性能以及功耗方面的优势,我们基于模拟器搭建了一个试验平台,首先对比了 MESI 和 DLS 的性能,其次对比了 MESI 和 DLS 的 L1C 性能,片上网络性能,以及片上网络功耗。

4.1　试验方法

本试验是基于龙芯模拟器进行的。龙芯模拟器是龙芯团队为了开发验证龙芯处理器[11-13]而开发的精确到每一拍的模拟器,和实际流片的处理器校准过。该模拟器是基于 SimpleScalar 工具集[14]开发的,支持 MIPS 指令集体系结构。为了评估 DLS 的片上网络功耗,我们在该模拟器上集成了 Orion[15]片上网络功耗模拟器。Orion 使用的参数是 65nm 工艺,1.0V 的核心电压,1GHz 的主频,以及 2500μm 的核间连线长度。

我们开发了龙芯模拟器的多核版本,具体为 16 核片上多核处理器,其结构示意图见图 2。

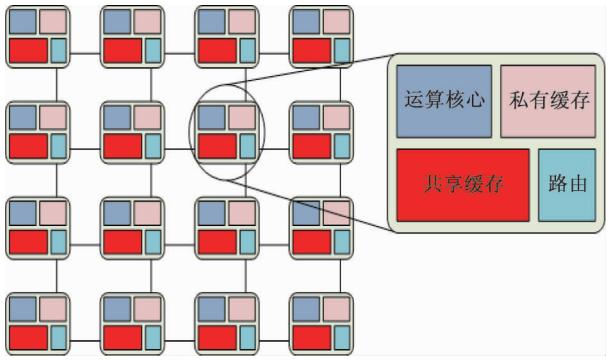

图 2　模拟的 16 核处理器结构

在该龙芯多核模拟器中,集成了 16 个龙芯 464V 核心[13],每个核心集成了一个 4 发射,9 级流水的乱序执行单元,每个核心分别有 64KB 的 L1 数据缓存和 L1 指令缓存。所有核心共享 16MB 的 L2 缓存(LLC 缓存),这 16MB 的 L2 采用 S-NUCA 的结构,分成 16 个分块,每个分块 1MB,每个核心有 1 个分块。该多核模拟器的片上网络才用了 4×4 的二维(2D)网格(Mesh)结构。多核模拟器所模拟的处理器的具体参数如表 1 所示。

表 1　处理器参数

参数	值
核心数	16
处理器核心	1G 主频,乱序 4 发射
分支预测	GShare,4096 项 PHT
ALU/FPU	2/2
ROQ	64 项
L1/指令缓存	64KB,4 路,3 拍延迟
共享 L2 缓存	16MB,4 路,10 拍延迟
缓存块大小	256 位
片上网络拓扑	4×4,2D 网格
路由延迟	2 拍
线延迟	2 拍
路由能耗	3.77e－10J
链路能耗	2.22e－10J
总线仲裁能耗	9.01e－13J

本试验采用了斯坦福大学的并行程序测试集 SPLASH-2[16]作为测试基准程序。具体每个程序使用的数据集大小详见表 2。

表 2　测试集和输入大小

测试集	规模
barnes	4096
cholesky	d750.0
fft	65536
fmm	256
lu	512×512
ocean-con	130×130
ocean-non	130×130
radix	262144
raytrace	teapot
water-nsquared	512 分子,3 步

4.2　试验结果

为了验证本文提出的 DLS 协议的性能和功耗优势,首先将 DLS 的整体性能和 MESI 进行了对比,接着分别比较了在两种协议下 L1C 的缺失率和片上网络的功耗。

4.2.1　整体性能

本文采用了总执行时间来描述整体性能,没有采用每拍执行的指令数(instruction per cycle,IPC),原因是对于 IPC,不同的线程可能差别很大。图 3 给出了每个基准测试程序在 DLS 上的执行时间,其执行时间归一化为 MESI 的执行时间。

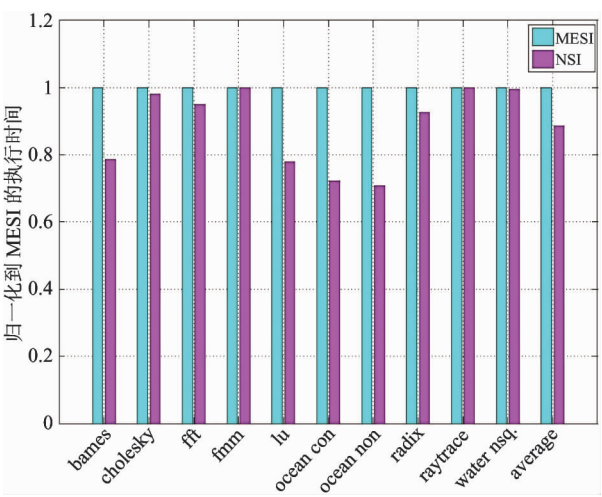

图 3　MESI 和 DLS 的性能比较

总体而言,相比于 MESI,DLS 的性能优势很明显,10 个基准测试程序有 8 个显示出 DLS 的性能比 MESI 要好,2 个(fmm 和 raytrace)显示与 MESI 性能相差不大。其中,性能提升最大的是 ocean-non,DLS 将性能提升了 28.34%。相比于 MESI,DLS 平均将性能提升了 11.08%,性能显著提升的主要原因是 DLS 协议可以消除写操作中非必要的 Invalidation 和 Ack 消息。

4.2.2　L1 缓存性能

为了进一步分析性能提升的原因,我们分析了 L1 缓存性能。具体来说,我们对比了 DLS 和 MESI 的 L1 缓存缺失。图 4 给出了 L1C 分别在 DLS 和 MESI 下的缺失率。和总体性能不同的是,在 L1 缓存缺失方面,所有程序显示,DLS 的结果都要比 MESI 要好。一个显著的例子是 lu,对于 lu,DLS 的缓存缺失率只有 MESI 的 1/3 左右(缓存缺失率从 1.85%减到了 0.61%)。一个可能的原因是对于在 DLS 的 L1C SHD 块,MESI 认为是 INV 状态的块(在 DLS 下,RdEx 操作并不会将 L1C 中的 SHD 无效成 INV 块,只有在同步点才会猜测成 SUS 块,而在 MESI 下,RdEx 会将其他 L1C 上的对应 SHD 块无效成 INV 块,导致下一次访问时出现缓存缺失)。

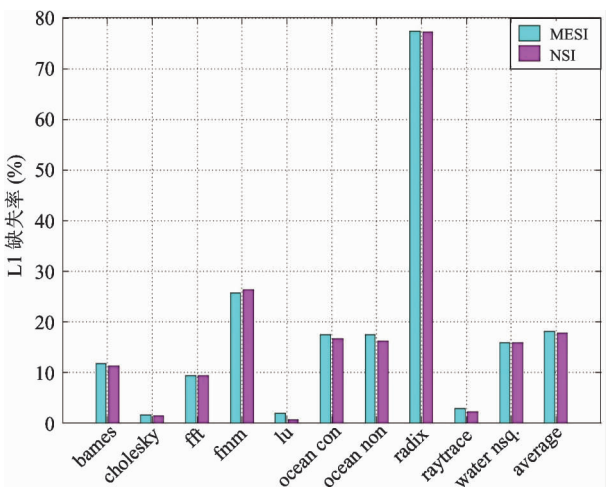

图 4　MESI 和 DLS 缓存缺失率比较

我们同时也分析了不同基准测试程序反映出的 SUS 块的被再次访问的分布。如图 5 所示,我们发现对于大部分的 SUS 块(平均 57.29%),在被替换出去之前,不会再被访问,这些 SUS 块并不会导致任何的猜测执行,出现此现象的原因是大部分程序对数据的使用都具有阶段特征,而同步点恰好是不同阶段的分割点。同时,有 38.12% 的 SUS 块会导

致猜测执行,但 SUS 块是最新的数据,猜测执行不会带来回滚,可以直接提交。只有剩下 4.59% 的 SUS 块中的数据是老的,猜测执行会导致回滚。总体而言,DLS 协议对 SUS 块的猜测执行只会导致很少一部分的无效执行。

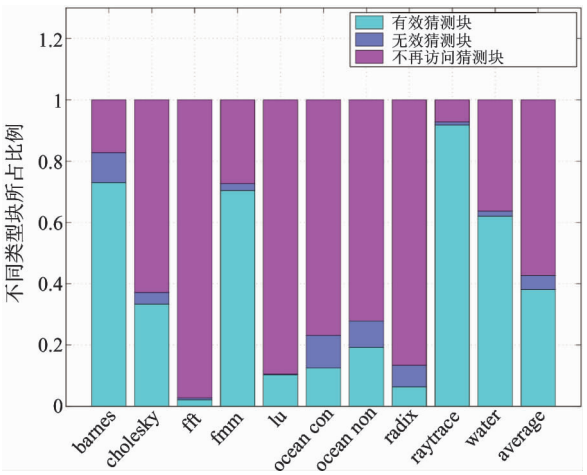

图 5　猜测块的分布

4.2.3　片上网络通讯

DLS 协议能消除所有的 Invalidation 和 Ack 消息,相比 MESI 协议,这能显著地减少片上网络通讯。图 6 比较了 DLS 和 MESI 的片上网络通讯情况。总体而言,DLS 可以大大地减少片上网络通讯(平均达到 28.83%)。具体而言,对于最好的情况(lu),DLS 可以将片上网络通讯减少 81.92%,即使对于最差的情况(fmm),DLS仍然可以减少6.23%

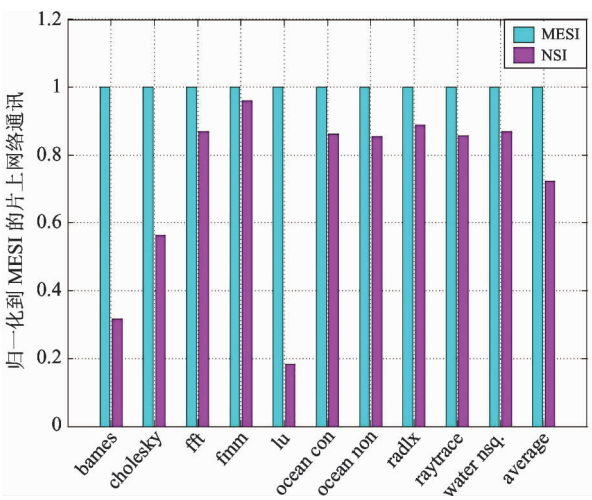

图 6　MESI 和 DLS 片上网络通讯比较

的片上网络通讯。相比于 MESI,片上网络通讯的显著减少解释了 DLS 相比于 MESI 的性能提升。

在减少片上网络通讯的同时,DLS 也能减少片上网络的功耗,从而降低整个处理器的功耗。因为 DLS 能消除 MESI 不必要的 Invalidation 和 Ack 消息,从而减少片上网络路由转发以及连线翻转的功耗。如图 7 所示,从所有 10 个基准测试程序看,相比于 MESI,DLS 平均降低了 15.65% 的功耗,尤其是对程序 barnes,DLS 甚至减少了 46.52% 的片上网络功耗。

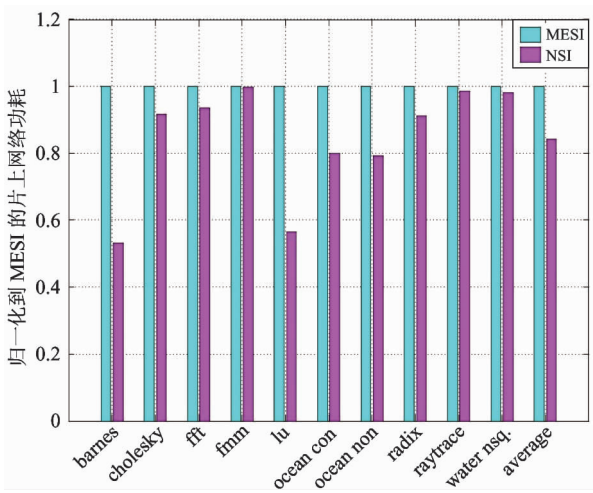

图 7　MESI 和 DLS 片上网络功耗比较

试验表明,在一个 16 核的多核处理器中,相比于传统基于目录的 MESI 协议,DLS 不仅可以完全消除用于存储备份信息的目录以及其电路面积,而且平均可以提高 11.08% 的程序性能,减少 28.83% 的片上网络通讯,以及减少 15.65% 的功耗。而这一切,只需要改变处理器的设计,并不需要改变编程语言和编译器,因此,该协议无需更改或重新编译即可以兼容现有的代码。

5　结　论

在共享存储并行系统中,存储一致性模型和高速缓存一致性协议是两个至关重要的方面。存储一致性模型定义了在共享存储系统中,何种执行行为(访存序列)是正确的,而高速缓存一致性协议则从底层定义了何种的缓存层次和行为(包括状态转

化,片上网络消息等)可以符合该一致性模型。存储一致性模型和高速缓存一致性协议是紧密相连的,因此,设计高速缓存一致性协议必须充分考虑到存储一致性模型的特性。

通过研究当今使用最广泛的弱一致性模型发现,弱一致性模型的写操作并不需要立即被其他核心(或 L1C)观察到,可以延迟到下一个同步点观察到。因此,在传统的 MESI 缓存一致性协议下,写操作发出的 Invalidation 消息和其他核心(或 L1C)的 Ack 消息是非必要的。

基于上述观察提出的缓存一致性协议 DLS,通过引入在同步点自无效的机制,保证了写操作无需发送 Invalidation 和 Ack,程序的执行仍然是符合弱一致性的。由于写操作无需往其他核心发送无效消息,因此该协议也无需 MESI 协议的目录,从而节省了目录的同步动态随机存储器(SDRAM)面积和功耗。为了避免自无效导致无效仍是有效的且会被再次访问的缓存块,可引入猜测执行机制,将自无效的块标志成可疑块,从而避免直接自无效带来的性能损耗。除了消除了目录的面积和功耗,受益于对片上网络消息的减少,相比于 MESI,DLS 可以在提高性能的同时降低片上网络的功耗。

参考文献

[1] Jeffers J. Intel ® Xeon Phi™ Coprocessor-Modern Accelerator Technologies for Geo-graphic Information Science. Springer US, 2013. 25-39

[2] AMD Opteron 6200 Processors. http://www.amd.com/us/products/server/processors/ 6000-booktitle-platform/6200/Pages/6200-booktitle-processors.aspx,2011

[3] Fick D, Dreslinski R G, Giridhar B, et al. Centip3De: A 3930DMIPS/W configurable near-threshold 3D stacked system with 64 ARM Cortex-M3 cores. In: Proceedings of the 2012 IEEE International Solid-State Circuits Conference Digest of Technical Papers, 2012. 190-192

[4] Archibald J, Baer J L. Cache coherence protocols: Evaluation using a multiprocessor simulation model. *ACM Transactions on Computer Systems*, 1986, 4(4): 273-298

[5] Agarwal A, Simoni R, Hennessy J, et al. An evaluation of directory schemes for cache coherence. *ACM SIGARCH Computer Architecture News*, 1988, 16(2): 280-298

[6] Lenoski D, Laudon J, Gharachorloo K, et al. The directory-based cache coherence protocol for the DASH multiprocessor. ACM, 1990

[7] Laudon J, Lenoski D. The SGI Origin: a ccNUMA highly scalable server. *ACM SIGARCH Computer Architecture News*, 1997, 25(2): 241-251

[8] Dubois M, Scheurich C, Briggs F. Memory access buffering in multiprocessors. *ACM SIGARCH Computer Architecture News*, 1986, 14(2): 434-442

[9] Papamarcos M S, Patel J H. A low-overhead coherence solution for multiprocessors with private cache memories. *ACM SIGARCH Computer Architecture News*, 1984, 12(3): 348-354

[10] Emerson E A, Kahlon V. Rapid parameterized model checking of snoopy cache coherence protocols. In: Tools and Algorithms for the Construction and Analysis of Systems. Springer Berlin Heidelberg, 2003. 144-159

[11] Hu W, Wang J, Gao X, et al. Godson-3: A scalable multicore RISC processor with x86 emulation. *IEEE micro*, 2009,(2): 17-29

[12] Hu W W, Chen Y J. GS464V: A high-performance low-power XPU with 512-bit vector extension. In: Proceedings of the 22nd IEEE Symposium on High Per-formance Chips. 2010. 22-24

[13] Hu W, Yang L, Fan B, et al. An 8-Core MIPS-Compatible Processor in 32/28 nm Bulk CMOS. *Solid-State Circuits, IEEE Journal of*, 2014, 49(1): 41-49

[14] Austin T, Larson E, Ernst D. SimpleScalar: An infrastructure for computer system modeling. *Computer*, 2002, 35(2): 59-67

[15] Wang H S, Zhu X, Peh L S, et al. Orion: a power-performance simulator for interconnection networks. In: Proceedings of the 35th Annual IEEE/ACM International Symposium on Microarchitecture, 2002. 294-305

[16] Woo S C, Ohara M, Torrie E, et al. The SPLASH-2 programs: Characterization and methodological considerations. *ACM SIGARCH Computer Architecture News*, 1995, 23(2): 24-36

DLS: a directoryless coherence protocol for shared cache

Liu Daofu[*,**,***], Chen Tianshi[*,**], Guo Qi[*,**]

([*] State Key Laboratory of Computer Architecture, ICT, CAS, Beijing 100190)

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

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

Abstract

The directory caused performance decline of the shared cache coherence protocol using a directory to record each shared cache block's private reserved information for multi/many core parallel systems was studied. The study discovered that multi/many core systems in practical use need not to store the shared information of shared cache blocks because the systems mostly use a weak consistency protocol. According to the protocol, a core's write-operation need not to be immediately observed by other cores until the next synchronous point comes. Based on the discovery, a directoryLess shared cache coherence protocol needing not to record shared information, called DLS, was put forward. The DLS completely removes the directory and Invalidation/Ack messages, and efficiently maintains cache coherence by using a novel self-suspicion + speculative execution mechanism. The SPLASH-2 benchmark was used to test a 16-core processor, and the testing results show that the DLS not only completely removes the chip area cost of the directory, but also improves processor performance by 11.08%, reduces the overall network traffic by 28.83%, and reduces the energy-consumption in network communication by 15.65% on average compared with the traditional MESI protocol with full directory. Moreover, the DLS does not involve any modification to programming languages and compilers, and hence is seamlessly compatible with legacy codes.

Key words: memory consistency, cache coherence protocol, multi/many core system, weak consistency