一种基于数据访问特征的层次化缓存优化设计

李崇民 王海霞 张 熙 汪东升

(清华大学计算机科学与技术系 北京 100084)

(清华信息科学技术实验室 北京 100084)

摘 要 随着片上可集成的处理器核数增加,多核处理器的片上通信延迟不断增大,目录存储开销也随之线性增长.层次化缓存结构将片上缓存递归划分为多级区域,并将数据复制到各级区域内以减小片上通信延迟,同时通过多级目录结构降低了目录存储开销.文中通过对数据访问特征进行分析,提出一种新型改进层次化缓存结构(EHCD),将从片外读入的数据直接放置在请求者所属的底层区域内,在降低延迟的同时,保证私有数据在片上最后一级缓存中只有一份副本,提高片上存储的空间利用率,具有良好的可扩展性.对 16 核处理器的实验结果表明,EHCD 设计比传统共享缓存结构执行时间平均减少 24%,比原有层次化缓存设计执行时间平均减少 15%,具有很好的优化效果.

关键词 片上多处理器;层次化缓存;多级目录;数据访问特征

中图法分类号 TP302 **DOI 号**: 10.3724/SP.J.1016.2011.02064

An Optimized Hierarchical Cache Design Based on Data Access Features

LI Chong-Min WANG Hai-Xia ZHANG Xi WANG Dong-Sheng

(*Department of Computer Science and Technology, Tsinghua University, Beijing* 100084)

(*Tsinghua Laboratory for Information Science and Technology, Beijing* 100084)

Abstract As more cores are integrated into one die, chip multiprocessors suffers higher on-chip communication latency, and linearly increased directory overheard. Hierarchical cache architecture partitions on-chip caches into multilevel regions recursively, reducing the communication latency by replicating the data blocks to multiple regions that contains the requestor and alleviating the storage overhead of directory by using multilevel directory. According to the data distribution in the last-level cache, we improve the data placement policy and propose an enhanced hierarchical cache directory (EHCD). EHCD directly puts an incoming off-chip data block into the lowest region that contains the requestor to reduce access latency, which guarantees only one data replica is kept in the last-level cache for private data. EHCD improves the capacity utilization of the last-level cache as well as good scalability. Simulation results on a 16-core CMP show that compared with shared organization, EHCD gets 24% execution time reduction, and 15% reduction over original hierarchical cache design.

Keywords chip multiprocessors; hierarchical cache; multilevel directory; data access feature

---

收稿日期:2011-08-29;最终修改稿收到日期:2011-09-16.本课题得到国家自然科学基金(60773146,60833004,60970002)、国家"八六三"高技术研究发展计划项目基金(2008AA01A201)资助.**李崇民**,男,1973 年生,博士研究生,中国计算机学会(CCF)学生会员,主要研究方向为多核处理器体系结构.E-mail: lcm03@mails.tsinghua.edu.cn. **王海霞**,女,1977 年生,博士,副研究员,主要研究方向为多核处理器体系结构、形式化验证等. **张 熙**,男,1983 年生,博士研究生,主要研究方向为多核处理器体系结构. **汪东升**(通信作者),男,1966 年生,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、高性能计算、网络存储与安全等.E-mail: wds@tsinghua.edu.cn.

1 引　言

随着半导体工艺的进步，单个芯片上能够集成越来越多的处理器核，片上多处理器(Chip Multi-Processor，CMP)已经成为计算机体系结构研究的主要方向之一.片上存储系统的性能是影响片上多处理器系统性能的一个主要因素.片上多处理器的片上存储主要采用非均一缓存结构(Non-Uniform Cache Architecture，NUCA)结构[1].在 NUCA 结构中，最后一级缓存被分成多个 bank 放置在片上的不同位置.一个处理器核到某个 bank 的访问延迟由两部分组成：缓存 bank 访问延迟和片上通信延迟.如果不考虑片上网络拥塞，片上通信延迟与处理器核到其要访问的缓存之间的距离成正比.处理器核访问邻近的缓存所需的延迟较低，而访问距离远的缓存就需要较高的延迟.如果处理器核所需要的大部分数据要从距离较远的缓存处获得，片上通信延迟会比较长，从而降低系统的性能.数据的复制和迁移技术通过将数据复制或迁移到使用数据的处理器核邻近的缓存中，减少片上的通信延迟，提高最后一级缓存系统的性能.

层次化缓存[2]是一种静态的数据复制技术，它针对片上多处理器中存储层次与片上网络的特点，将片上资源递归地划分成多级区域.当一个处理器核需要访问某个数据时，在包含发出请求的处理器核的每一级区域内都保存一份该数据的副本.层次化缓存保证处理器的大多数据请求能够在距离该处理器较近的底层区域内部得到满足，是降低片上通信延迟的一种有效设计.

多级区域的副本位置需要通过目录进行维护，但是全目录结构的存储开销较大.早期的层次化目录[3]采用树型结构来记录数据的共享情况，有效减少了目录的开销，但由于在树根附近的流量过于集中，容易成为系统的瓶颈，影响了层次化目录的广泛应用.PHD[4]通过将树的根节点按照地址分布到整个系统中，解决了上述瓶颈问题.不过 PHD 是应用在超立方体网络中的，其各级目录在物理上是分开的.在访问各级目录时需要较大的延迟.层次化缓存[2]借鉴 PHD 的思想，利用存储开销较小的多级目录来维护片上数据的共享信息，同时解决了数据就近访问和目录存储开销问题.

通过对数据访问特征进行分析，可以发现在最后一级缓存的容量绝大部分被私有数据占据，并且大多数的私有数据只被访问一次.对于一个私有数据，如果在各级区域内都为其保存一个数据副本，会降低最后一级缓存的空间利用率.本文对层次化缓存结构进行优化，对片外读入数据的放置策略进行了改进，将从片外读入的数据直接放置在请求者所属的底层区域内，保证私有数据在片上最后一级缓存中只有一份副本，以提高片上最后一级存储的空间利用率.模拟实验结果表明，改进的层次化缓存(Enhanced Hierarchical Cache Directory，EHCD)设计比传统共享缓存结构执行时间平均减少 24%，比原有层次化缓存设计执行时间平均减少 15%，具有很好的优化效果.

2 相关工作

目前对片上存储进行优化的研究工作主要集中在数据的复制与迁移上，主要思想是把将数据放置在与请求该数据的处理器核邻近的缓存 bank 中，这样对该数据的后续访问的延迟就会比较小，从而提高整个系统的性能.CC[5]将从本地私有缓存中替换出来的数据放到邻近的有空闲空间的 L2 缓存中，为了获得各 L2 缓存的空闲情况，CC 将各 L2 缓存的 Tag 集中在一起进行管理.DCC[6]将 CC 中集中式的 Tag 分成多个分布式的 Tag 组，每组负责对一部分 L2 缓存进行管理.D-NUCA[1]通过一个命中计数来决定是否将一个数据迁移到更快的缓存 bank 中.NuRAPID[7]将不经常使用的数据迁移到具有空闲缓存行的邻近缓存中.

VR 和 VM[8-9]将 L1 缓存中替换出来的数据保存在本地的 L2 缓存中，当再次访问该数据时，就可以从本地的 L2 缓存中直接得到该数据.ASR[10]在 VR 的基础上把数据细分为只读数据和读写数据，在数据从 L1 缓存替换时，以一定的概率将只读数据保存在本地的 L2 缓存中.ASR 将复制分为多个级别，每个级别对应不同的将只读数据复制到本地 L2 缓存的概率，ASR 在 L2 缓存中增加额外的硬件来实时选择对系统最有利的复制级别.R-NUCA[11]与操作系统相结合，将数据按页粒度分为私有数据、指令和共享数据三种.将私有数据直接复制在本地，共享数据则只在根节点处保存，指令数据按照轮转地址映射方案在片上维护多个副本.

为减少维护数据共享信息的目录开销，SCI[12]中利用链表结构来维护数据的一致性，具有较好的可扩展性，不过性能受到一定影响.Wilson 最早提

出了基于总线的层次化多处理器[13]. PHD[4]和层次化目录[3]将层次化的概念应用到应用目录协议的多处理器中. 层次化目录中的根节点可能会成为影响系统性能的瓶颈,PHD 在超立方体结构中提出了将根节点分布到各个节点中的思想,不过每一级的目录还是分开存放的. 层次化缓存[2]借鉴了 PHD 的思想,将其应用到二维的网络拓扑上,并应用多级目录来减少目录的存储开销,在提高系统性能的同时降低了存储开销问题. Huh 等人[14]提出了一种可灵活确认一系列数据共享度的 NUCA 结构,并研究了不同共享度与性能之间的关系.

3 层次化缓存的基本结构

本节以一个 16 核 Tile 结构的片上多处理器为例,对层次化缓存设计中的关键问题进行说明. Tile 结构设计简单且易于扩展,在多核处理器设计与研究中被广泛采用. 图 1 给出了一个 Tile 结构 16 核处理器,每个 Tile 结构包含有一个处理器核、分立的指令和数据 L1 缓存、一个 L2 缓存和 1 个将每个节点连接到片上网络的路由器,各个 Tile 间通过片上网络连接. 层次化缓存设计的关键在于层次区域划分方法和多级目录的结构及组织方式. 为了简化描述,本文只研究 $4^n(n>1)$ 个 Tile 的多核处理器,所有 Tile 节点组成一个 $2^n\times 2^n$ 的阵列.

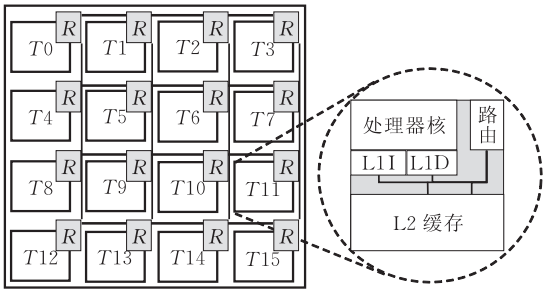


图 1 基准系统结构示意图

3.1 多级区域划分

在层次化缓存中,每 4 个邻近的 Tile 节点组成一个 1 级区域,4 个相邻的 1 级区域组成一个 2 级区域,此过程会重复下去,直到系统中所有节点都包含在一个区域中. 在每一个区域内,层次化缓存为每个缓存块指定一个节点作为其在该区域内的根节点,负责该数据块在区域内的一致性. 从处理器核发出的请求会依次访问其所属的从小到大的各级区域的根节点,直到到达一个能满足该请求的根节点. 图 1 所示的系统中,每 4 个邻近节点(如 $T0$、$T1$、$T4$、$T5$)组成一个 1 级区域,4 个 1 级区域组成一个 2 级区域,这个 2 级区域包含了系统中的全部片上节点.

层次化缓存根据每个节点在区域中的位置对这些节点重新编号. 每个节点的编号为一个 4 进制数 $a_n\cdots a_{n+1}a_i\cdots a_1$,其中 $a_i\in\{0,1,2,3\}(1\leq i\leq n)$ 是包含该节点的 $i-1$ 区域在第 i 级区域内的位置,数值 0 对应于左上位置,1、2、3 分别表示右上、左下和右下位置. 图 2 给出了基准系统中各节点的新编号,这种编号方式便于确定一个区域内任一地址对应的根节点的位置. 如果一个编号为 $a_n\cdots a_{n+1}a_i\cdots a_1$ 的节点对一个最低 n 位是 $b_n\cdots b_{n+1}b_i\cdots b_1$(除去行内偏移)的地址发出请求,那么该地址在一个 i 级区域内对应的根节点编号为 $a_n\cdots a_{i+1}b_i\cdots b_1$. 可以看出,区域根节点的编号会随着请求节点和请求地址的改变而改变. 如果请求的地址分布是均匀的,那么区域根节点的分布也是均匀的. 对于包含 4^n 个节点的片上多处理器,一个数据在一个 i 级区域的根节点同时也是该数据在一个 1 级、2 级…($i-1$)级区域的根节点. 对于每个地址,一共有 4^{n-i} 个 i 级根节点,同时所有区域根节点的数目为 4^{n-1} 个.

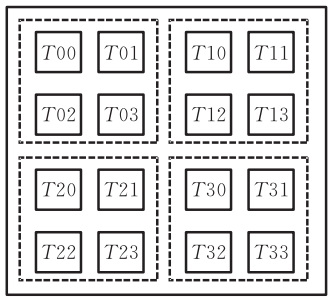


图 2 区域划分与节点编号

3.2 多级目录

层次化缓存使用多级目录来记录一个数据在区域或节点间的共享情况. 一个数据的第 i 级区域根节点维护一个 4 位 $i-1$ 级目录,目录中的每一位对应构成该区域的 4 个 $i-1$ 级区域内该数据的共享情况. 由于第 i 级节点同时也是某个 1 级、2 级直到 $i-1$ 级区域的根节点,因此该节点需要维护一个 0 级目录、1 级目录直到 $i-1$ 级目录,如图 3(a)所示,其中 0 级目录中维护的是数据在区域内 L1 缓存中的共享情况. 如果一个多级目录中的第 k 级目录中的第 m 位被置位,则表示在第 m 个 k 级子区域内存在有该数据的有效副本,没有置位则表明第 m 个 k 级子区域内没有该数据. 图 3(b)给出了一个 16 核处理器目录组织方式的实例.

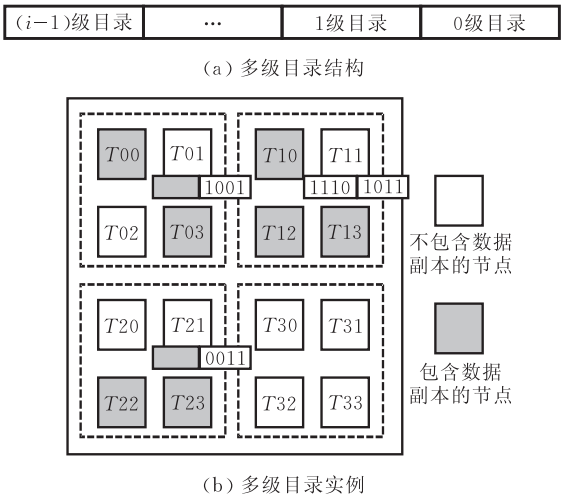


图 3　层次化缓存的多级目录(全局根节点为 $T11$,在 3 个 1 级区域内共有 7 个 L1 缓存共享该数据)

要维护层次化缓存系统中数据的一致性,需要对原有的目录协议进行修改.在层次化缓存中,一个 L2 缓存可能与其它的 L2 缓存进行交互.当一个处理器核需要数据时,首先将请求发送到其所在的 1 级区域的根节点中的 L2 缓存,如果请求可以在 1 级区域内得到满足,就可以快速地给处理器返回响应消息;否则就要把请求消息向上一级区域的根节点,直到到达一个能处理该请求的根节点为止.

根据程序在执行时的局部性原理,从处理器核发出的多数请求都可以在层次化化缓存的 1 级区域内得到满足,降低了每个消息的平均传输距离,从而降低请求的延迟,同时也会降低片上网络的流量.由于采用了多级目录来保存数据块的共享信息,层次化缓存有效地降低了目录的存储空间.

4　改进的层次化缓存

4.1　最后一级缓存中的数据分布

最后一级缓存中的数据可以分为两种,私有数据和共享数据.如果一个缓存行中的数据只被一个处理器访问,就是私有数据;如果被多个处理器访问,就是共享数据.不同的缓存组织方案适用于不同的数据分布,假如最后一级缓存中都是私有数据,则私有缓存组织方式是最佳选择.通过了解缓存一级缓存中的数据分布特性,可以为设计缓存组织方案提供参考.

通过对 SPLASH-2[15] 和 PARSEC[16] 测试程序集中的应用进行分析[17],可以发现私有数据占据了 L2 缓存的大部分空间.图 4 给出了私有数据与共享数据在 L2 缓存中所占空间的比例,可以看出私有数据(共享数目=1)平均占据了超过 92%的空间.图 5 给出了不同访问次数的数据在 L2 缓存中所占的空间,平均有 70%的数据只被访问了一次,这部分数据当然是私有数据.从图 4 和图 5 可以看出:L2 缓存空间中多数(约 70%)的数据是私有的并且只被访问了一次.基本层次化缓存既在 1 级区域的根节点处保存这些数据,同时也在全局根节点处保

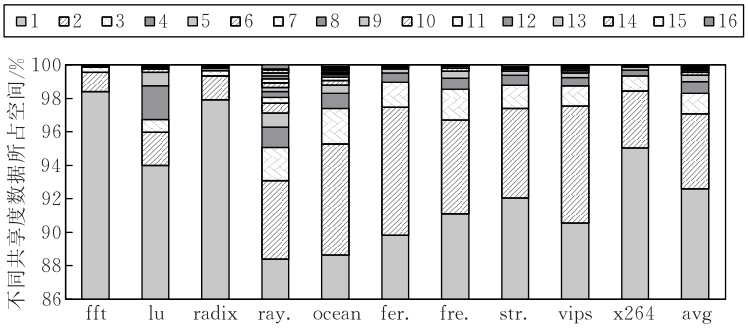


图 4　不同共享度的数据在 L2 缓存中所占空间

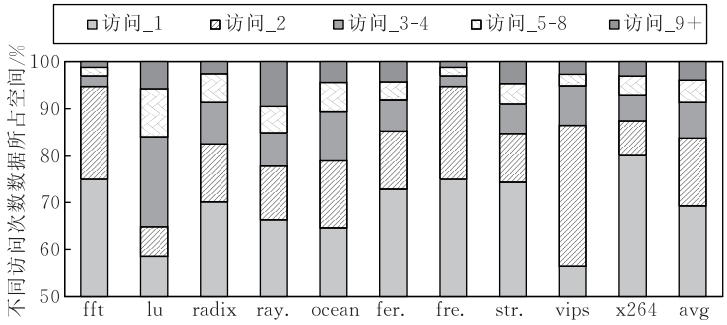
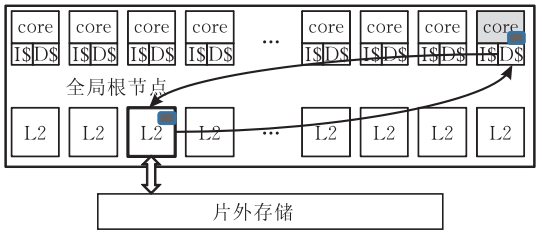


图 5　不同被访问次数的数据在 L2 缓存中所占空间

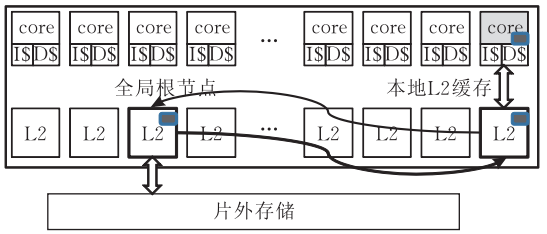
存这些数据，这对于私有数据并没有用处. 更好的方法是直接将这些数据放置在请求节点所属的 1 级区域中，这样既可以保证私有数据具有较低的访问延迟. 也能够提高 L2 缓存的空间利用率.

4.2 改进的数据放置策略

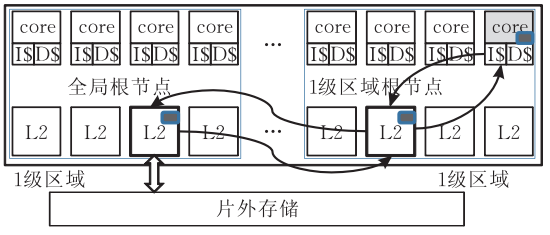
通过对最后一级缓存中的数据分布进行分析，本文提出改进的数据放置策略，比几种常用的缓存组织方式更加适用于 SPLASH-2[15] 和 PARSEC[16] 测试程序集. 图 6 给出了不同缓存组织方式中的私有数据放置策略. 在前 3 种组织方式中，根节点中总是会保存一份由片外访问来的数据. 在图 6(a)中的共享组织方式中，所有数据只在根节点保存一份，发生替换时直接同片外主存交互，具有最好的空间利用率. 而在图 6(b)和图 6(c)的方案中，一个数据片上最后一级缓存中可能有多个副本，其存储空间的利用率比共享方式要低. 当非全局根节点发生替换时，需要同全局根节点交互，替换出去的缓存行被保存在全局根节点中，直到根节点中发生替换时该数据才会被替换到片外. 根据图 4 和图 5 给出的数据分布特性，可以发现在根节点中保存私有数据的副本会降低根节点的空间利用率. 本文提出如图 6(d)的改进层次化缓存，其放置策略如下：当数据被初次访问时，只在包含请求节点的 1 级区域内保存一份该数据的副本，同时改变替换操作的处理过程，在非根节点发生替换时，不再将被替换的数据保存在全局根节点中，将提高二级缓存的空间利用率. 对于一个数据被多个 1 级区域共享的情况，可以通过片上转发来减少开销较大的片外访问.



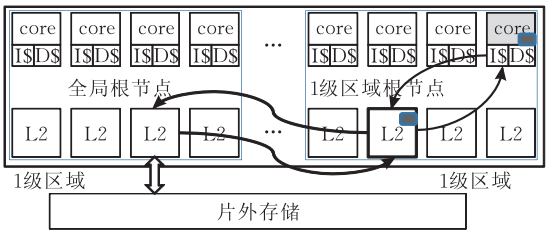
(a) 共享方式，只在全局根节点保存一份副本



(b) VR/VM/ASR，副本保存在根节点和本地L2中



(c) HCD，副本存在根节点及底层区域的根节点中



(d) EHCD，只在底层区域的根节点中保存一份副本

图 6 不同缓存组织方式下的私有数据存放方式

4.3 一致性协议的实现

改进的层次缓存实现了图 6(d)所示的数据放置策略，为此需要在每个 Tile 的二级缓存中增加一个新的目录缓存来记录数据在 1 级区域内的共享情况，同时可以简化每一个缓存行中的目录项，只记录该数据在本区域内的 L1 缓存中的共享情况. 对于图 1 的基准系统，一个目录缓存中的目录项需要 4 位记录数据在 4 个 1 级区域中的共享情况，L2 数据缓存中的一个目录项也只需要 4 位来记录数据在 L1 缓存中的共享情况. 此结构也适用于包含更多区域层次的系统，此时目录缓存中的目录项记录的是在下一级存储区域中的数据共享情况.

在修改缓存结构的同时，还需要对一致性协议进行修改，包括 L1 缓存发出的一致性请求的处理流程：即从片外读入一个数据时，直接将数据放置在请求者所在的 1 级区域，全局根节点只在新增加的目录缓存中记录该 1 级区域的共享信息. 为了说明一致性协议的修改，我们将分别描述由 L1 缓存发出的各种请求的处理过程.

首先考虑 L1 缓存发出的读请求. 在 1 级区域根节点收到由 L1 缓存发出的读请求后，将在数据缓存中查找该数据. 如果保存有该数据，只要将该数据返回发出请求的 L1 缓存并更新该缓存行的目录项即可；如果此数据被本区域内的其它一级缓存修改，具体的处理流程与基本的层次缓存相同；如果在 1 级区域根节点发生缺失，则需要向上一级区域根节点(全局根节点)发出读请求.

当全局根节点收到由 1 级区域根节点发来的读请求后，将在目录缓存中查找对应的目录项. 如果该数据被其它 1 级区域独占或被多个 1 级区域共享，读请求会被转发到距离请求节点最近的一个 1 级区域根节点，同时在目录缓存的目录项中将发来请求

的 1 级区域加入. 否则说明片上没有该数据，全局根节点会向片外发送读请求，并将由片外得到的数据送到发来请求的 1 级区域根节点，同时在目录缓存中记录数据的共享信息（被发来请求的 1 级区域所独占）.

如果一个 1 级区域根节点收到由全局根节点转发来的读请求，它负责提供该数据的最新版本. 如果数据被某个一级缓存所修改，则要向该一级缓存发一个写回请求，并在得到写回的数据后将最新的数据直接发送到最初发出读请求的 1 级区域的根节点，同时更新自己的缓存行，并将缓存行的状态更新为共享（S）.

下面考虑 L1 缓存发生写缺失的情况. 当 1 级区域根节点收到由 L1 缓存发来的写请求后，如果在 1 级区域根节点内的状态为独占或修改，那么处理过程与原始的层次缓存目录一样. 如果缓存行的状态为不存在（NP）或共享，则需要向全局根节点发送请求以取得写权限，并在收集到所有的响应消息之后将响应发送到最初的 L1 缓存.

如果全局根节点收到由 1 级区域根节点发来的写请求，将在新增加的目录缓存中查找该目录项. 如果数据正在被另外一个 1 级区域所修改，那么写请求会被转发给唯一的独占者. 如果该数据被多个 1 区域共享，则所有在其它 1 级区域中的数据副本都要被无效，同时向距离请求节点最近的一个 1 级区域发送转发请求. 如果全局根节点中并没有请求数据的目录项，它将会发起一次片外访问，处理过程与读请求类似.

如果一个 1 级区域根节点收到一个无效请求（Inv），它会将区域内所有数据副本无效，之后返回确认（Ack）消息. 如果收到的是一个转发过来的写请求，响应消息将会包含有最新的数据. 最后将一致性状态设为无效.

如果一级缓存发生替换，其处理过程与层次缓存目录相同. 如果 1 级区域根节点发生替换，如果要被替换的数据是干净的（clean），就发送一个 PUTS 请求到全局根节点的目录缓存，否则将发送一个包含最新数据的 PUTX 请求.

如果全局根节点收到一个 PUTS 请求，会在目录缓存中检查请求者是否是片上的最后一个共享者，如果片上还有其它共享者，就将发出请求的节点从目录中清除；否则将这条目录项从目录缓存中移除. 如果收到的是 PUTX 请求且为片上唯一的共享者，全局根节点将会把这个最新的数据写回到片外的主存中，同时从目录缓存中移除相应的目录项.

由于大部分一级缓存发出的请求都能在 1 级区域内得到满足，读写请求都能够很快得到响应，以较小的延迟完成操作，从而提高整个系统的性能.

5　系统评测

5.1　实验设置

为验证层次化缓存及 EHCD 的效果，我们利用 Simics 和 GEMS[18] 建立实验环境. Simics 是一个事件驱动的全系统模拟平台. 威斯康星大学的 GEMS 模拟器在 Simics 的基础之上实现了完整的片上存储层次，为片上存储层次研究提供了平台.

表 1 给出了片上多处理器系统的配置，二级缓存与一级缓存的数据是全包含的关系，片上网络中传输的是所有维护一致性所需的消息.

表 1　系统配置参数

部件名称	配置参数
CMP 规模	16 核
缓存行大小	64 字节
L1 指令缓存大小/相连度	32KB/2 路组相连
L1 数据缓存大小/相连度	32KB/2 路组相连
L1 访问延迟	2 周期
L1 替换策略	伪 LRU
片上目录缓存大小	16k-entry，6-way/Tile
片上目录延迟	15 周期
L2 缓存大小/相连度	1MB/16 路/Tile
L2 访问延迟	15 周期
L1 替换策略	伪 LRU
网络拓扑	4×4 Mesh
每跳延迟	3 周期
片外存储延迟	300 周期

5.2　实验结果

实验选择的基准测试程序来自 SPLASH-2[15] 和 PARSEC[16] 测试程序集，这些程序覆盖科学计算及通用数据处理. 表 2 给出了 10 个测试程序及其问题规模. 实验采用 L1 缺失延迟、程序执行时间和片上网络流量作为评价的标准.

表 2　测试程序描述

负载名称	问题规模
fft	256K points
lu	1024×1024 matrix，16×16 blocks
radix	1048576 numbers，1024 radix
raytrace（ray.）	Teapot. env
ocean	258×258 ocean
ferret（fer）	Simsmall
freqmine（fre.）	Simsmall
streamcluster（str.）	Simsmall
vips	Simsmall
x264	Simsmall

本文对 4 种方案进行评测，基准方案为共享组织方式(Shared)，对比系统包括 VM、层次化缓存(HCD)和改进的层次化缓存(EHCD). 实验模拟一个由 4×4 mesh 连接的 16 核的片上多处理器. 图 7 给出了平均 L1 缺失延迟的实验结果. 由于 VM 将从 L1 缓存替换出来的缓存行保存到本地的 L2 缓存中，因此它具有最小的 L1 缺失延迟，VM 的平均 L1 缺失延迟为共享组织方式的 74%. EHCD 的 L1 缺失延迟比共享组织方式平均减少 16%，比 HCD 平均减少 7%.

由于大多数 L1 请求可以在本地 1 级区域内命中，读写请求的延迟都有所降低. L1 缓存发出的请求多数不必被发送到根节点，从而减少了片上网络的流量. 如图 8 所示，3 种对比方案都比共享组织方式的片上网络流量要小. HCD 和 EHCD 方案的平均片上流量分别是共享方式的 68% 和 72%，因为 EHCD 不在根节点保存数据，引发了更多的片上转发，因此比 HCD 增加了 4% 的网络流量. 在 VM 方案中，L1 缓存可以从本地 L2 缓存中取得大部分的数据，因此具有最小的平均网络流量，是共享组织方式的 46%，不过它同时也有着最多的片外缺失.

图 9 给出了测试程序的执行时间. 从图中可以看出，EHCD 的执行时间比共享组织方式平均降低了 24%，比 VM 平均降低了 10%，比 HCD 平均提升 15%.

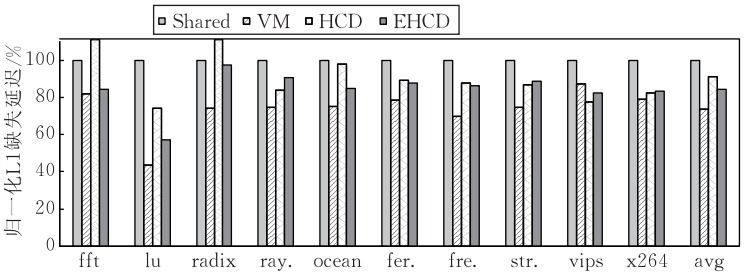


图 7 L1 缺失延迟

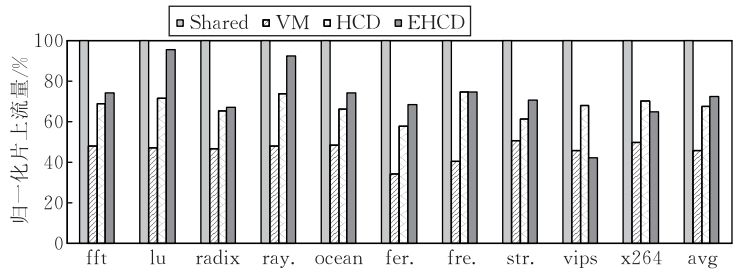


图 8 片上网络流量

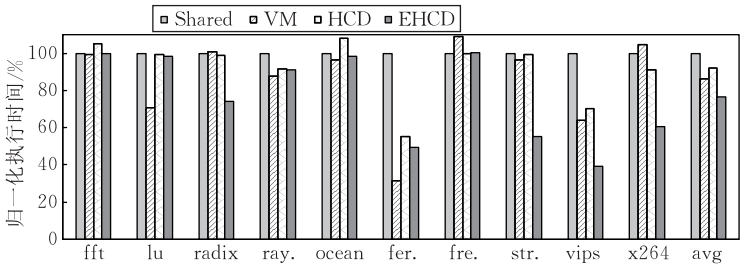


图 9 系统性能

5.3 与 VM 方案的对比

VM 方案将私有和共享方式结合起来管理片上的 L2 缓存资源，VM 基于共享组织方式，通过动态地将从 L1 缓存中替换出来的缓存行保存在本地的 L2 缓存中来减少线延迟. 为了提高 L2 缓存的空间利用率，在需要将数据替换出根节点时，对于还有共享者的缓存行，VM 将其保存在一个 Tag 缓存中. EHCD 中，对于私有数据，只在本地的 1 级区域中保存有一个副本. 对于共享数据，则在访问过该数据的 1 级区域内保存有数据副本. 下面详细比较一下两种方案.

从存储开销上来看，由于采用了多级目录，EHCD 的目录存储开销比 VM 小. 例如一个 16 核的片上多处理器中，VM 中的每个缓存行需要 16 位来记录

数据的共享信息，而在 EHCD 中，每一个缓存行和目录缓存中的目录项都只需要 4 位来记录数据在 1 级区域内或 1 级区域间的共享信息. 与此同时，如果假定物理地址长度为 40 位且状态需要 2 位来表示，每个 Tile 节点的 1MB 的 L2 缓存在共享组织方式下需要 1.26MB 的存储开销，EHCD 的开销为 1.17MB，开销比 VM 降低了 7%左右.

从片上副本数目上来看：VM 方案多个 L2 缓存中保存有数据的副本，最多可以达到 16 个，而在 EHCD 中最多只会有 4 个副本，因此 EHCD 的空间利用率比 VM 高.

从具体性能上来看：从实验结果来看，与共享组织方式相比，VM 方案的执行时间降低为 14%，EHCD 方案的执行时间降低为 24%，EHCD 比 VM 方案执行时间降低了 10%，具有较好的性能.

从可扩展性上来看：VM 方案的目录开销与片上处理器核的数据成正比，当系统规模扩大时，其目录开销会很快超过缓存容量所需存储. 而 EHCD 方案的目录存储开销与处理器核的数据成对数关系增长，远远小于 VM 方案的目录存储开销，具有良好的可扩展性.

6　结　论

随着片上可集成处理器核数目的增加，基于共享缓存组织方式的多核处理器片上通信延迟不断增大，保存数据共享信息的目录存储开销也线性增加. 本文在层次化缓存结构上，对最后一级缓存中的数据放置进行优化，将从片外读入的数据直接放置在请求者所属的底层区域内，保证私有数据在片上最后一级缓存中只有一份副本. 该技术提高了片上最后一级存储的空间利用率，提供良好的可扩展性. 对 16 核处理器的实验结果表明，EHCD 设计比传统共享缓存结构执行时间平均减少 24%，比原有层次化缓存设计执行时间平均减少 15%，具有很好的优化效果.

参　考　文　献

[1] Kim C, Burger D, Keckler S W. An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches. ACM SIGPLAN Notices, 2002, 37(10): 211-222

[2] Guo Song-Liu, Wang Hai-Xia, Xue Yi-Bo, Li Chong-Min, Wang Dong-Sheng. Hierarchical cache directory for CMP. Journal of Computer Science and Technology, 2010, 25(2): 246-256

[3] Maa Y C, Pradhan D K, Thiébaut D. A hierarchical directory scheme for large-scale cache-coherent multiprocessors//Proceedings of the 6th International Parallel Processing Symposium. Washington, USA, 1992: 43-46

[4] Wallach D A. PHD: A hierarchical cache coherent protocol [M. S. dissertation]. MIT, 1992

[5] Chang J, Sohi G S. Cooperative caching for chip multiprocessors//Proceedings of the 33rd Annual International Symposium on Computer Architecture. Boston, USA, 2006: 264-276

[6] Herrero E, Gonzalez J, Canal R. Distributed cooperative caching//Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques. Toronto, Canada, 2008: 419-428

[7] Chishti Z, Powell M D, Vijaykumar T N. Optimizing replication, communication, and capacity allocation in CMPs//Proceedings of the 32nd Annual International Symposium on Computer Architecture. Wisconsin, USA, 2005: 357-368

[8] Zhang M, Asanovic K. Victim replication: Maximizing capacity while hiding wire delay in tiled chip multiprocessors//Proceedings of the 32nd Annual International Symposium on Computer Architecture. Wisconsin, USA, 2005: 336-345

[9] Zhang M, Asanovic K. Victim migration: Dynamically adapting between private and shared CMP caches. MIT CSAIL, TR 2005-064, 2005

[10] Beckmann B, Marty M, Wood D. ASR: Adaptive selective replication for CMP caches//Proceedings of the 39th Annual International Symposium on Microarchitecture. Florida, USA, 2006: 443-454

[11] Hardavellas N, Ferdman M, Falsafi B, Ailamaki A. R-NUCA: Data placement in distributed shared caches//Proceedings of the 36th Annual International Symposium on Computer Architecture. Austin, USA, 2009: 184-195

[12] Gustavson D. The scalable coherent interface and related standards projects. IEEE Micro, 1992, 12(1): 10-22

[13] Wilson A W. Hierarchical Cache/bus architecture for shared memory multiprocessors//Proceedings of the 14th Annual International Symposium on Computer Architecture. Pittsburgh, USA, 1987: 244-252

[14] Huh J, Kim C, Shafi H, Zhang L X, Burger D, Keckler S W. A NUCA substrate for flexible CMP cache sharing//Proceedings of the 19th Annual International Conference on Supercomputing. Massachusetts, USA, 2005: 31-40

[15] Woo S C, Ohara M, Torrie E, Singh J P, Gupta A. The SPLASH-2 programs: Characterization and methodological considerations//Proceedings of the 22nd Annual International Symposium on Computer Architecture. Santa Margherita Ligure, Italy, 1995: 24-37

[16] Bienia C, Kumar S, Singh J P, Li K. The parsec benchmark suite: Characterization and architectural implication//Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques. Toronto, Canada, 2008: 72-81

[17] Wang Jing-Lei. Research on the key techniques of memory hierarchy of multi-core processors based on network on chip [Ph. D. dissertation]. Tsinghua University, Beijing, 2009 (in Chinese)
(王惊雷. 基于片上网络的多核处理器存储系统关键技术研究[博士学位论文]. 清华大学, 北京, 2009)

[18] Martin M M K, Sorin D J, Beckmann B M, Marty M R, Xu M, Alameldeen A R, Moore K E, Hill M D, Wood D A. Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset. Computer Architecture News (CAN), 2005, 33(4): 92-99

LI Chong-Min, born in 1973, Ph.D. candidates. His research interest is multiprocessor architecture.

WANG Hai-Xia, born in 1977, Ph. D., associate professor. Her research interests include high performance computing, multiprocessor architecture and formal check.

ZHANG Xi, born in 1983, Ph. D. candidates. His research interest is multiprocessor architecture.

WANG Dong-Sheng, born in 1966, Ph. D., professor, Ph. D. supervisor. His research interests include computer architecture, high performance computing, storage and file systems, and network security.

Background

Advancements in semiconductor technology enable more cores to be integrated onto one single chip. Chip Multiprocessors (CMPs) have become the de facto design for high performance processors. To achieve high performance and throughput, CMPs heavily rely on the organization of on-chip memory hierarchies, especially for the last-level cache (LLC).

Directory coherence protocols are widely used for large scale CMPs. In Directory protocols, the L2 caches can be either private or shared. A private L2 cache organization can provide fast on-chip access latency and low design complexity, whereas an organization with shared L2 caches prefers to maximize the space utilization of L2 and to minimize the off-chip accesses.

The storage overhead of directory grows linearity with the number of processing cores increases. As the system scales up, it needs more space to store directory, which limits the scalability of system. HCD (Hierarchical Cache Directory) eliminate this bottleneck by distributing root nodes according to data address. HCD divides CMP tiles into multiple regions hierarchically, and combines it with data replication. Multi-level directory is used to record the share information within a region and assist the regional home node to complete operation efficiently. The cache blocks in LLC can be divided into private and shared. While occupying the majority of LLC capacity, a considerable part of private data is accessed only once and thus has more possibility of being evicted from LLC. Keeping multiple replicas of such data in the LLC is useless.

In this paper we propose enhanced hierarchical cache directory (EHCD) that organize LLC as different shared regions. Each region provides data for the processing cores within the region, and the address mapping is similar to HCD. Each L1 cache can get data with low wire latency due to proximity. In this scheme, we replace the concept of global home node with cluster home node. For private data, there is only one copy exists in the LLC. An off-chip data block is placed in the local cluster home node directly, and an evicted cache block from the cluster home is sent back directly to the memory. For a data block shared by cores belong to different regions, data forwarding is used to send the data block from the first accessed region instead of getting data off-chip.

This paper makes the following contributions: We demonstrate that a good cache organization should keep only one copy in LLC for private data. We introduce EHCD which use the cluster home node to replace the global home node. EHCD guarantees single cache block existed in LLC for private data. The simulation results for a 16-core CMP show that EHCD can get 24% performance improvement against the shared organization, which is 15% over original hierarchical cache design.

This work is supported by the National Natural Science Foundation of China under Grant Nos. 60833004, 60970002; the National High Technology Research and Development Program (863 Program) of China under Grant No. 2008AA01A201.