片上多核Cache资源管理机制研究

贾小敏 张民选 齐树波 赵天磊
(国防科技大学计算机学院 长沙 410073)

摘 要 随着片上多核成为处理器发展的主流和片上Cache资源的持续增长,Cache资源的管理已成为片上多核的关键问题。介绍了片上多核Cache资源管理的研究进展,依据研究内容将Cache资源的管理分为Cache划分和Cache共享两类。对Cache划分,探讨了其主要组成部分和一般形式,分析和比较了典型的片上多核Cache划分机制。对Cache共享,给出了其主要研究内容,并介绍和比较了几种主流的片上多核Cache共享机制。通过分析,认为软硬件协同管理的页划分应是未来片上多核Cache划分机制的研究重点;而片上多核Cache共享机制的研究则应从目标应用的Cache行为特征着手。

关键词 片上多核,Cache资源管理,Cache划分,Cache共享,非一致Cache
中图法分类号 TP368 **文献标识码** A

Cache Resource Management Mechanisms of Chip Multiprocessors

JIA Xiao-min ZHANG Min-xuan QI Shu-bo ZHAO Tian-lei
(School of Computer Science, National University of Defense Technology, Changsha 410073, China)

Abstract With the trend towards Chip Multi-processors(CMPs) for the next leap in computing performance and the increasing of on-chip cache capacity, many architectures have explored cache resource management for better use and allocation of cache resource to accommodate various optimizing goals. Representive cache resource management mechanisms were evaluated and divided into two main catogaries, i. e. Cache Partitioning and Cache Sharing, based on their research focus. For cache partitioning, its main components and common form were described. For cache sharing, the key research directions were presented. Several mainstream cache partitioning mechanisms and cache sharing mechanisms were then investigated and compared respectively. The conclusion is that hardware and software co-designed page management is a promising field for future cache partitioning research, and also cache sharing mechanisms should be based on the thorough analysis of application background and the corresponding cache access behaviors.

Keywords Chip multi-processors, Cache resource management, Cache partitioning, Cache sharing, Non-uniform cache architecture

片上多核(Chip Multi-Processors,CMP)已成为微处理器的主流和未来发展方向。在此背景下,片上L2及更高层次Cache的管理这一CMP的关键问题也变得日益重要。

CMP的多个核可同时执行多个不同的进程或线程。为充分利用资源,CMP的Cache系统需为多个并行执行在不同核上的进程或线程提供快速的数据访问。因此,CMP下Cache系统面临着新的挑战[1,2]:

(1)应用访存行为呈现出多样性:CMP可支持单线程应用、多线程应用及多个应用的组合等多种负载。不同类型的应用程序,其Cache行为相异,空间局部性、时间局部性、工作数据集大小和共享度等特性差别显著。为动态适应不同应用多样的Cache行为,需要灵活的Cache组织和管理策略。

(2)CMP中并发线程之间相互干扰。为提高空间利用率,CMP的L2及更高层次Cache一般会全部或部分共享;此外CMP上同时执行的多个进程或线程间可能需要通信、竞争使用共享资源。由于应用访存行为呈现出多样性,通信和竞争可能导致抖动、总体性能降低、性能不可预测、性能不可控和实时任务服务质量无法保证等问题。

传统Cache通过替换策略[3](LRU等)间接影响Cache资源的分配,实际上是一种自由的按请求分配的方法。CMP环境下,多个线程、应用竞争Cache资源,这种自由分配方式已不适用,需加以限制和规范,以使多线程、多应用对Cache资源的利用在满足一定约束的条件下能够尽可能地优化。因此,CMP上Cache资源的管理机制已经成为一个关键问题。

1 片上多核Cache组织方式

CMP系统中,L1 Cache重在高速,组织方式比较单一,多为处理器核私有,与处理器核紧密耦合;而L2及更高层次

到稿日期:2010-03-09 返修日期:2010-05-14 本文受国家863高技术研究发展计划(No. 2009AA01Z124),国家自然科学基金(No. 60970036)和国家自然科学基金(No. 60873016)资助。

贾小敏(1982-),女,博士生,主要研究方向为多核微处理器片上存储层次,E-mail:xmjia@nudt.edu.cn;**张民选**(1954-),男,教授,博士生导师,主要研究方向为计算机体系结构与实现技术、新型微处理器体系结构等;**齐树波**(1982-),男,博士生,主要研究方向为片上互连网络;**赵天磊**(1982-),男,博士生,主要研究方向为微处理器体系结构和编译技术。

Cache 则多为大容量 Cache[4]，组织方式比较多样[5-8]，因此 Cache 资源管理的研究多集中在 L2 及更高层次的 Cache 上。大容量 Cache 有私有和共享两种基本组织方式[5]。研究表明[4]，完全共享的大容量 Cache 查找空间大，易受不断增长的线延迟的影响，导致访问速度变慢，同时充分的共享也会引起线程间负面的相互干扰。私有组织方式访问速度快，但缺乏灵活性，当负载不平衡时，会出现工作集较大的核的 Cache 发生抖动，而同时工作集较小的核的 Cache 却空闲的情况[5]，不能有效利用宝贵的片上 Cache 资源，从而限制了整体吞吐率[2]。

Cache 资源管理机制试图在私有和共享这两种基本方式之间寻找某个平衡点，采用二者混合的组织方式，在结合二者优点的同时规避各自的缺点[5]，使得既能减少延迟及相互干扰，又能充分利用 Cache 空间。这 3 种组织方式的对比如图 1 所示。

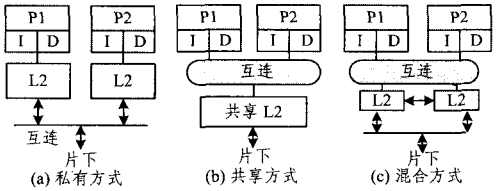

图 1　CMP Cache 的不同组织方式[4,9]

Cache 资源管理的研究方向可分为两类：一类以核为基础，研究如何在不同核（线程、进程）间分配 Cache 空间，本文称之为 Cache 划分（Cache Partioning）机制[5,9-16]；一类则以数据分类（共享或私有）为基础，研究不同类型数据如何使用 Cache 空间，包括相应的组织方式及放置、查找、替换等策略，本文称之为 Cache 共享（Cache Sharing）机制[17-22]。

CMP 中 Cache 资源管理有两种研究方式：一种立足于私有组织方式[19-22]，在此基础上研究如何实现空间共享；一种立足于共享组织方式[5,10-12]，在此基础上添加限制策略，以减少延迟，缩小相互干扰。

2　片上多核 Cache 划分机制

本节讨论片上多核 Cache 划分机制的一般形式，比较几种主流的划分机制的特点，并分析得出 Cache 划分机制未来的研究方向。

2.1　Cache 划分机制概述

片上多核 Cache 划分机制研究如何将 Cache 资源分配给同时执行的多个线程/应用，以达到特定的优化目标。组成部分有优化目标[15]（Optimization Target）、实现方式[12]（Enforcement Scheme）、划分策略[15]（Partitioning Policy）、策略指标[15]（Policy Metrics）、监测系统[12]（Monitor）和搜索算法[12]（Search Algorithm）。

（1）优化目标：片上多核 Cache 划分机制的优化目标一般是吞吐率[12]、公平性[23]、QoS[24]或以上三者中某种满足一定约束条件下优化的另一目标[11]。实现时优化目标需具体化，如优化吞吐率可由优化总 IPC 或 Weighted IPC 来表述。划分机制对优化目标的实现效果，用评价指标（Evaluation Metrics）衡量[15]。

（2）实现方式：Cache 划分实施的粒度、方向以及实施的位置等（操作系统、硬件等）。

（3）划分策略：产生 Cache 划分决策所依据的策略，是划分机制的核心。LRU 替换策略可看作是最基本的划分策略[15]。

（4）策略指标：划分策略的量化依据、驱动划分策略产生划分决策[15]。理想上，策略指标应与评价指标相同，但实际中评价指标往往不可测或开销太大，因此需选择一些易测量的策略指标来近似评价指标[15]。策略指标与评价指标的相关性越高越有效。

（5）监测系统：划分策略所依据的统计数据的收集系统是 Cache 划分机制的基础。监测系统所收集数据的准确性决定着 Cache 划分机制的效率。

（6）搜索算法：基于划分策略，Cache 划分机制需对监测系统收集到的统计数据空间进行搜索，以确定策略指标最优的划分决策。搜索算法的准确性对 Cache 划分决策的效率有很大影响。此外划分决策搜索是 NP 问题[12]，需考虑算法的开销。广泛采用的搜索算法是贪心法及其变种[12]。

动态 Cache 划分机制一般按周期启动，每个划分周期开始时唤醒划分搜索算法，以上周期监测系统收集的统计数据为输入，查找令策略指标最优的划分，作为本划分周期的划分决策。动态 Cache 划分实际上是一种预测机制，隐含了在接连的两个划分周期内负载的行为相对稳定这一假设。

2.2　实现方式

按划分方向，片上多核 Cache 划分机制的实现方式可分为页划分（Page Partitioning）、组划分（Set Partitioning）和路划分（Way Partitioning）[2] 3 种，如图 2 所示。页划分以页作为划分的粒度，给 CMP 上不同的核或应用分配相应的 Cache 页数目；组划分以一个或多个 Cache 组为单位，给 CMP 上不同的核或应用分配相应的 Cache 组数目；路划分则以 Cache 路为单位，将 Cache 路分配给同时执行的线程或应用。路划分多为硬件实现，而组划分、页划分既可软件实现[16]，也可硬件实现[16]。

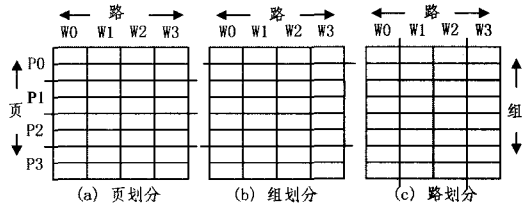

图 2　CMP 中 Cache 划分的不同实现方式

目前针对路划分的研究最广。路划分包括列划分[2,25]（Column Caching）和自由路划分[2,12]两种。列划分[25]通过设置一个位向量将特定线程或应用的数据限制在指定列中，位向量的一位代表一列（即通常所说的路），不同数据可映射到相同列、不同列或两者混合。不同于列划分，自由路划分[26]只限制特定数据使用的路数目，并不指定路的位置。路划分方式多懒惰进行[12,26]，即在实际发生 Cache 失效时才替换、逐出数据。因此 Cache 划分行为可与失效处理重叠，对访问延迟不产生影响。

2.3　监测系统

目前关于 Cache 划分的研究中采用路划分实现方式的最多，且已经具有相对成熟的硬件和理论支持，因此本小节主要介绍基于路划分的监测系统。监测系统可分为在线或线下两种。常用的线下监测方法是将负载中的每个线程都在其独占

Cache的条件下单独执行一次，以获得独立执行历史。在线监测系统有ATD(Auxiliary Tag Directory，冗余Tag)[12]、影子Tag(Shadow Tag)[27]、LRU计数器[14,16,28]和在线采样周期[10]等。

应用在执行时，存在程序段行为[29,30]，同一程序段内应用的行为和各种统计数据相对稳定，程序段之间则差异明显。线下监测系统无法适应负载的动态变化和多线程间的相互干扰，而在线监测系统收集负载的实时行为信息，可更好地适应应用的动态程序段行为。

Cache划分研究中常采用失效率或失效次数作为策略指标近似IPC，原因在于无法用一种开销适当的在线监测方法获得所有可能的Cache空间分配选项下不同应用的IPC变化；但采用ATD和硬件LRU栈计数器，可获得LRU替换策略下应用独占Cache的所有路时的统计数据栈距离直方图SDH[31](Stack Distance Histogram)，如图3所示。据SDH可推算出在所有可能的Cache容量分配选项下应用的失效率或失效次数[32]。

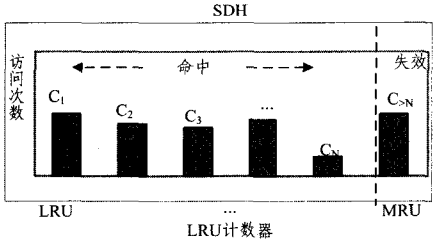

图3 栈距离直方图示意

LRU等传统逐出策略都有一个栈属性，Cache中的每组都可看成是一个LRU栈[30]，Cache块按最后访问周期排序。访问的栈距离被定义为访问的Cache块被访问时在LRU栈中的位置。如图3所示，构建N路组相联Cache的SDH，需$N+1$个LRU计数器，$C_1, C_2, \cdots, C_N, C_{>N}$。每个Cache访问按以下规则更新LRU计数器：若访问命中LRU栈中的第i个位置，递增C_i；若失效，则递增$C_{>N}$。利用SDH，具有相同组数的容量较小的Cache的失效次数可通过容量较大Cache的SDH方便地计算出来：假设$K<N$，则K路组相联相同组数的Cache的失效次数可计算如下：

$$misses = C_{>N} + \sum_{i=K+1}^{N} C_i \quad (1)$$

相当一部分Cache划分机制采用SDH计算使负载的总失效次数最小化L2 Cache划分[12,14,28]。SDH的收集方式有动态和静态两种，可通过软件[33,34]或硬件[14,16]LRU栈及栈计数器来统计。动态Cache划分采用硬件LRU栈及计数器收集SDH[14,28]。为进行准确的Cache划分，应得到所有应用LRU策略下独占Cache时的SDH。但多核及多线程中，单纯使用硬件LRU计数器[14]只是基于原本的Tag收集信息，应用间的相互干扰会影响收集到的单个应用的SDH的准确度。

ATD[12]解决了此问题，使单个应用的SDH与其它应用的影响隔离。ATD在原有Tag阵列之外额外复制一个冗余Tag阵列，通过在这个冗余Tag阵列上实施任何欲进行比较的策略，但不进行数据操作，可获得采用备选策略时Cache的统计数据，包括IPC、失效次数等。CMP的Cache划分研究中，为避免应用间的相互干扰影响SDH的准确性，可为每核设置一个采用LRU策略的ATD，每核的所有或部分访问流也访问该核的ATD。

Qureshi提出对ATD采用DSS[12](Dynamic Set Sampling)方法来减少硬件代价。DSS方法令ATD只复制Cache Tag的部分组来近似整个Cache行为。Qureshi等[12]用概率方法从理论上证明只需采样32组，ATD就可达到90%以上的预测准确率。

2.4 片上多核Cache划分机制现状

我们按实现方式分类，对CMP上的Cache划分机制的相关研究进行分析和探讨。表1给出了目前几种主流Cache划分机制的特点比较。

表1 CMP上Cache划分机制比较

划分机制	优化目标	策略指标	监测方法	实现方式	搜索算法
硬件LRU计数器[14,28,35]	性能	失效率	硬件LRU计数器	硬件自由路划分	改进的贪心法支持非凸曲线
UCP[12]	性能	利用度	ATD、硬件LRU计数器	硬件组级自由路划分	支持非凸曲线的前瞻算法
公平划分[23]	公平性	五个公平性指标	静态历史信息、失效次数计数器	硬件路划分	贪心法
PDAS[11]	保证基本QoS下优化性能	失效率、IPC加权的失效率	ATD、硬件LRU计数器	硬件路划分	贪心法
MLP感知的动态划分[32]	性能	MLP加权的失效次数	ATD、硬件LRU计数器、MSHR和HSHR	硬件路划分	贪心法
核ID划分[5]	性能等	未考虑	静态历史信息、OS信息	组划分，粒度为Cache片	OS控制
影子Tag划分[25]	性能	失效次数	影子Tag	硬件路划分	贪心法
CCP[10]	保证QoS下优化性能	失效率	在线采样、硬件LRU计数器	硬件路划分多时分划分	迭代法
OS控制的硬件配额[26]	多种	多种	未考虑	硬件自由路划分	OS控制
统一资源分配[36]	性能	失效率等	在线监测系统	未考虑	爬山算法
CQoS[24]、QoS感知的划分[37]	QoS，优先级	MPI、IPC	失效率计数器、IPC计数器	为每个优先级设置阈值	动态调整阈值
OS控制的页划分[16]	多种	多种	多种	页划分，页重着色	多种

2.4.1 基于路划分的Cache

Suh等[14,28,35]最先研究CMP中集中共享Cache的动态Cache划分，率先采用硬件LRU计数器作为监测方式统计SDH。但他们的机制只基于原有Tag收集信息，CMP上同时执行的应用之间的干扰会影响SDH的准确度。文献[28]假定失效率关于Cache容量是凸曲线，搜索算法采用贪心法[13]。

在Suh等[14,28,35]提出的LRU计数器的基础上，UCP[12](Utility-Based Cache Partitioning)为CMP上的每个核增加一个ATD，提出了以利用度为策略指标的CMP共享Cache的动态划分机制。ATD可克服应用间相互干扰对统计信息的影响。此外，为减少硬件开销，UCP[12]还首次并采用了DSS技术。搜索算法上，UCP[12]改进贪心法提出前瞻算法，搜索

时考虑了增加多路时的失效率变化，可克服非凸函数不收敛的问题。

公平划分[23]机制(Fair Partitioning)研究 Cache 共享的公平性问题，提出 5 个公平性策略指标，这 5 个指标的基本组成元素是线程与其它线程同时执行时及其单独执行时的失效率或失效次数。公平划分[23]指出优化公平性一般会带来性能的优化，但优化性能不一定会使公平性变优，原因在于一些优化性能的措施正是通过牺牲公平性来提升性能的。

PDAS[11](Performance Driven Adaptive Sharing)将共享 L2 Cache 资源组织成为物理分布式，分成多个 Cache 片，不同 Cache 片与不同核物理上紧密耦合，片间采用环连接。优化目标是在保证每个线程满足一个最低性能的基础上最大化总体吞吐率。划分时，每核先独立确定自身的需求，集中的仲裁器通过环连接取得所有核的需求，据此进行划分。划分搜索算法先保证每核都至少分得一个最小 Cache 空间来提供"基本公平性"，然后在此基础上搜索性能最高的分配决策。

统一资源分配机制[36](Coordinated Partitioning)统一分配 CMP 中的多种共享资源，包括 Cache 和访存带宽等。采用在线监测系统收集应用在各种资源分配下的性能，然后通过机器学习算法自动生成应用的性能-资源分配预测模型，据此模型进行划分空间搜索。

CQoS[24]是一种保证 QoS 的 Cache 划分管理框架，对具有不同延迟敏感度、不同局部性特性和不同需求的访存流设置和实现不同的优先级。QoS 感知的[37]机制(QoS-aware Partitioning)则实现了多种优先级资源管理策略，在满足低优先级应用 QoS 约束的前提下在执行时动态给高优先级应用分配更多 Cache 和带宽资源。

MLP 感知的动态 Cache 划分[32](MLP-aware DCP)在 UCP[12]的基础上考虑了失效的 MLP 开销，对每次失效按实际访问延迟赋予一个 MLP 权值。核 ID 划分[5]用核 ID 号划分 CMP 的分布式共享 L2，用一个 OS 控制的硬件表记录核与 L2 片间的对应关系。Dybdahl 等[27]采用影子 Tag 持续动态评估每核增或减一个 Cache 路时对总体性能的影响，并据此动态调整共享/私有 Cache 划分。CCP[10](Cooperative Cache Partitioning)引入多时分划分 MTP，将 Cache 的优先级支持和公平性改善等问题转化为已被很好解决的时分共享资源管理问题。OS 控制的硬件配额(Quota)机制[26]在 OS 和体系结构之间设置一个接口，由 OS 决定硬件配额，而体系结构为 OS 提供支持，OS 可利用这些支持有效管理 Cache。

有些研究试图从理论上解释基于路划分的 Cache 划分机制的作用原理。文献[32,38]给出一种模型来解释采用路划分的 Cache 划分机制的适用场合，而文献[15]则对 Cache 划分机制进行了全面考察，认为不存在对所有应用都最优的划分策略，同时最优的 Cache 划分随所采用策略指标的不同而不同。

2.4.2 基于页划分的 Cache 划分

OS(操作系统)控制的动态页划分[16]机制(OS Controlled Page Partitioning)通过 OS 级存储地址映射采用软件方法同时支持静态和动态 Cache 划分。当前的大容量 Cache 的容量都已大于页的容量，页号与 Cache 的索引之间有一些重叠的位，称为页着色位。页着色位的不同值代表不同的颜色，Cache 的空间被按页的颜色分成一些相互独立的区域。给不同线程/应用分配不同的页颜色，就可在线程/应用间划分 Cache 空间。该机制的划分控制及实现方式通过修改 OS 实现的。OS 控制的动态页划分是一种动态 Cache 划分实现方式的框架，可在此基础上进一步考虑多种优化目标(性能、公平性和 QoS)、策略指标、监测方式(在线采样、IPC、失效次数计数器等)和搜索算法。

2.5 分析和总结

从以上的介绍中可看出，之前的研究多集中在基于路划分的实现方式上，原因包括两方面：一是路划分的实现不涉及地址的变换，且具有 ATD 这样高效简单的硬件支持(ATD 及 LRU 计数器)；另一方面是传统 Cache 中，Cache 数据到组及页的映射采用模 N 函数，由数据地址的低位地址唯一确定，因而组划分方式和页划分中，若想更改 Cache 块使用的组或页，则需进行地址的重映射，重映射的复杂度较高，需要体系结构和操作系统的协同支持。

但路划分方式只能以路为单位，划分粒度较大，受限于 Cache 相联度的大小，且只能开发 Cache 组内资源使用的不平衡。事实上，应用的访问在不同 Cache 组间的分布差异也非常显著[16]。图 4 给出了 SPEC2006 测试程序 perlbench 的不同 L2 Cache 组间 Cache 访问次数的分布图。实验中我们将 1MB L2 的 1024 组按地址顺序分为 128 个区域，分别统计了各个区域的 L2 访问次数。图 4 的结果显示 L2 Cache 访问在各个区域间的分布很不均匀且没有明显的规律，有些应用某一时段只访问 L2 Cache 的某几组或某些页，存在 Cache 的某些组或页被频繁访问，甚至发生抖动，但是其它组或页却闲置的情况。

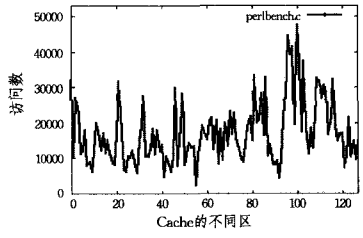

图 4 SPEC2006 程序 perlbench 的 Cache 访问在不同组间的分布图

这种组间分布不均衡的现象不仅造成了 Cache 空间的浪费，还有可能产生片上的局部热点，导致芯片失效。这种组间访问、页间访问的不均匀，为进一步开发 Cache 资源的利用，提供了机会。但在以往以路为粒度的研究中，多假设 Cache 组间访问均匀或者近似均匀，因此无法开发这方面的性能。开发组间、页间的不均匀，涉及到地址转换的问题，以往有些研究[39,40]用硬件实现了这种转换。完全由硬件实现地址转换，开销大且复杂性高。我们认为，从操作系统层面进行或协助进行页或组划分的控制，可以降低实现的开销和复杂性[39-42]。因此，鉴于基于路划分的 Cache 划分机制已经比较成熟，而基于组、页划分的 Cache 划分研究较少，我们认为未来 Cache 划分机制的重点应是软硬件协同控制的组划分或页划分机制。

3 片上多核 Cache 共享机制

Cache 共享机制研究内容涵盖广泛。本节先概要描述 Cache 共享机制的研究重点，然后依据不同机制的研究侧重点分类介绍这些机制。

3.1 Cache 共享机制概述

CMP 中的 Cache 共享(Cache Sharing)机制研究多核多线程下 Cache 空间的共享使用。研究的切入点是数据类别,将负载的数据或指令根据访问者数目及读写特性等分为不同类型,并采用不同组织方式、放置和查找策略等。

Cache 共享机制的优化目标一般是性能[17-22],或在优化性能的同时兼顾公平性或 QoS[17,18]。研究内容有数据分类、组织方式、放置策略(Placement Policy)、迁移策略(Migration Policy)、查找策略(Search Method)、溢出策略(Spill Policy)和复制策略(Replication Policy)等。

(1)数据分类:Cache 共享机制构建各种策略的理论基础。最常见的分类分为私有、共享两类及对这两类的细化。

(2)组织方式:不同数据类别采用不同的组织方式,基础组织方式一般采用分布式 NUCA(Non-Uniform Cache Architectures)结构,不同 Cache 片(体)的访问延迟非一致。

(3)放置策略:在数据分类并采用不同组织方式的基础上,确定不同数据类别的放置位置。放置位置决定着不同数据类别的访问延迟。

(4)复制策略:是否、何时及如何复制共享数据。复制共享数据可减少访问延迟和片上通信,但消耗 Cache 空间,可能导致额外的 Cache 失效。

(5)溢出策略:当 Cache 片逐出某个块时,是否及如何将该块保存到其它 Cache 片中。

(6)迁移策略:NUCA 将经常访问的数据迁移到离访问核较近的位置,可减少平均访问延迟。迁移策略决定是否、何时及如何迁移数据。

(7)查找方法:在不同放置、复制、溢出和迁移策略下,查找数据所需查找的位置、顺序、方向也不同,从而影响访问延迟。

3.2 片上多核 Cache 共享机制现状

不同共享机制的研究侧重点各有不同,如 VR[17](Victim Replication),VM(Victim Migration)[18],CC[2,20](Cooperative Caching),ASR[19](Adaptive Selective Replication)和 CMP-NuRAPID[21]重点研究了复制策略,CC[2,20],CMP-NuRAPID[21]和 DSR[22]等都支持溢出并给出了相应的溢出策略,而 VM[18],CMP-NuRAPID[21],Nahala[43,44]和 SP-NUCA[45]等机制则支持共享数据的迁移以减少空间浪费和延迟。表 2 比较了多种共享机制的数据分类和研究点。

表 2 CMP 上 Cache 共享机制比较

	数据分类	支持复制	支持溢出	支持迁移	片间一致性
VR[17]	未分类	是	否	否	需要
VM[18]	未分类	是	否	是	需要
ASR[19]	共享只读、其它	是	否	否	需要
CC[2,20]	共享、私有	是	是	否	需要
CMP-NuRAPID[21]	私有和共享只读、读写	有限	是	是	不需
DSR[22]	私有、共享	否	是	否	不需
Nahalal[43,44]	共享热块、其它	否	否	是	不需
SP-NUCA[45]	私有、共享	否	否	是	不需
混合 LLC[47]	共享、私有	是	否	否	需要
R-NUCA[41]	指令、私有和共享	是	否	否	不需

本节按主要研究点,即复制、溢出及迁移分类介绍这些共享机制。有些机制的研究点有多个,本文将其归类到其最主要的研究点中。

3.2.1 基于复制的共享机制

VR[17]基于 Tile 式 CMP,在采用目录协议的 L2 共享 Cache 上增加了一定条件下将本地 L1 逐出块复制在本地 L2 片中的策略。复制的逐出块是主节点为其它 Tile 的块的本地副本(Replica)。因此 VR 中 L2 片的空间由逐出块的本地副本和全局 L2 块共享。块的主节点由物理地址唯一确定,所有请求核都可将 L2 数据载入到本地 L1 中。当 L1 的块被作废或写回请求逐出时,如共享 Cache 一样处理;当因替换被逐出时,尽量在本地 L2 片中保存逐出块的副本,以减少后续对该块的访问延迟。

VR[17]中复制浪费空间,不适合多程序负载这类共享较少的负载。VM[18]在 VR[17]基础上,通过迁移私有块减少空间浪费。VM 为每个 L2 片增加一组 VM Tags。当块的副本在其它节点创建时,主节点不再保存该块,只在 VM Tags 中记录该块的信息。Cache 块按地址静态映射到 VM 中的主节点。主节点以两种方式保存数据块:一是正常保存方式,包括数据和 Tag;另一种则可能只在 VM Tags 中保存该块的 Tag,但不保存数据。VM 和 VR 都需要复杂的协议来维持 L2 间的一致性。

ASR[19]动态监测负载行为来控制 CMP L2 中共享数据的复制,只在复制的收益(降低 L2 的命中延迟)大于开销(L2 失效增加产生的延迟)时才进行复制。ASR 将复制分为多个离散的复制级别,动态监测复制的收益和开销,分段逼近最优级别。将计算最优复制级别的问题转化为确定复制是应该增加(H)、减少(L)或保持不变(C)的局部问题。不同复制级别采用不同复制概率随机确定是否复制写回块,复制集中在共享只读块上。

3.2.2 基于溢出的共享机制

CC[2,20]基于 CMP 片上最后一级 Cache(Last Level Cache,LLC),每个核私有本地 LLC,将所有私有 LLC 整体看成一个大聚合 LLC,采用一系列管理策略实现空间共享。CC 支持"干净"数据的 Cache 片到 Cache 片传输,可实现私有 LLC 间的"干净"共享数据的复制;CC 采用复制感知的替换策略,优先替换存在副本的数据块,且选择以一定概率随机将 LLC 逐出块(都是私有数据)溢出到其它 LLC 去;对不活跃数据,CC 采用 N-Chance 转发[46]实现全局替换,减轻溢出带来的 LLC 间冲突。CC 采用集中式目录提供复制和溢出所需的一致性支持,但集中式目录可能成为瓶颈,限制了 CC 的可扩展能力。

CMP-NuRAPID[21]将 NUCA 结构 NuRAPID[6]扩展到 CMP 上,沿用距离相联[6]和间接 Tag 的组织模式,采用私有 Tag 阵列、共享数据阵列的结构。此外,对不同数据类别采用不同策略:对共享只读数据采用有限复制,只在共享读次数达到一定阈值时才复制数据,可减少容量消耗;对共享读写数据采用原位通信,不进行复制,只支持迁移,避免一致性开销;对私有数据用容量借用支持溢出,当私有数据超出单核所属容量时,可放在容量需求较小的邻近核中,以实现不同核间 Cache 空间的共享。

DSR[22](Dynamic Spill-Receive)实现动态溢出,每个私有 Cache 用 Set Dueling 技术监测确定自身是应充当溢出 Cache 还是接收 Cache,二者只能居其一,每核只需增加一位溢出位,可扩展性好。

3.2.3 基于迁移的共享机制

Nahalal[43,44]提出一种根据数据是共享还是私有建立分开的共享和私有区域的 CMP Cache 布局拓扑。Nahalal 分析指出存在部分共享块，对其的访问很密集，占共享块访问的绝大部分，即共享热块效应。减少对共享热块的访问延迟对 CMP 整体性能会有很大影响，且只需很小的空间就可保存共享热块。Nahalal 把整个芯片布局成 3 层的形式：共享数据放置在芯片中心相对较小的共享区域内；区域四周是各核；私有数据放在最外围。这种布局方式可使所有核快速访问共享数据，且同时维持私有数据和核的临近性。Nahalal 不支持共享数据复制，避免一致性操作。Nahalal 中数据刚载入时，放置在请求核的私有区域，经过来自不同核的多次访问后，可迁移到中心共享区域。

SP-NUCA[45]将静态 NUCA[4] LLC 划分为私有和共享部分，对私有数据和共享数据采用不同放置策略。私有数据只能放置在本地 LLC 体中，位置取决于地址和请求核；共享数据则可放置在整个 LLC 中，位置只取决于地址。数据刚载入时，默认为私有数据，放置在请求核的私有体中；当有其它核访问该块时，状态改为共享并迁移到共享体中，保持共享状态直到被逐出。SP-NUCA 中共享片和私有片其实是重叠的，共享片和私有片的区分体现在组级。类似于 SP-NUCA，混合 LLC(Hybrid LLC)组织方式[47]也将本地 Cache 片分为私有片和共享片两部分。

Reactive NUCA[41]基于 Tile 分布式共享 L2 NUCA，引入固定中心簇(Fixed-Center Clusters)的概念，簇内 Tile 间访问速度为临近核访问速度，可快速访问。Reactive NUCA[41]支持 3 种固定中心簇：size-1，size-16 和 size-4。size-1 和 size-16 采用标准地址交叉放置和查找方法，size-4 簇采用轮转交叉算法[41]。Reactive NUCA 将 Cache 访问分为指令、私有数据和共享数据 3 种不同类型：指令放在 size-4 固定中心簇，簇内轮转交叉放置，簇间复制；私有数据放在请求核的 L2 片中，采用 size-1 固定中心簇，标准地址交叉放置；共享数据放在 size-16 固定中心簇，标准地址交叉放置，位置由地址静态确定。Reactive NUCA 以页为粒度区分不同数据类别，放置策略通过 OS 和一个布尔逻辑索引重映射算法实现，保证可修改块在 Cache 中只有唯一块，避免 L2 间硬件一致性机制。

3.3 分析与总结

在 Cache 共享的研究中，复制、溢出以及迁移策略是研究的重点。这些策略的共同点是它们能否真正带来益处(如提高吞吐率、降低失效率)与 CMP 上执行的应用的类型及相应的 Cache 访问行为密切相关。如 VR[17]中，复制带来性能提升的条件是必须有对复制数据块足够多的访问，使得复制减少的访问时间足以抵消其额外占用 Cache 空间导致的额外失效次数。因此 VR[17]更适用于数据共享较多的应用[17]。Nahalal[43,44]机制的提出则源于多线程应用存在“共享热块”效应。

因此 Cache 共享机制的有效性与具体的应用背景、应用类型以及应用中 Cache 访问数据的类型密切相关。无论是通用多核处理器还是专用多核处理器，Cache 共享机制的选择以及新共享机制的提出都应建立在对应用的访存行为类型分析的基础上。

结束语 本文从 Cache 划分机制和 Cache 共享机制两方面介绍了片上多核 Cache 资源管理机制。对于片上多核 Cache 划分机制，我们认为软硬件协调的页划分应是未来片上多核 Cache 划分机制的研究重点。而片上多核 Cache 共享机制的效率则与应用的行为模式密切相关，Cache 共享机制的选择与研究应始终以目标应用的类型和行为特征为基础。

参 考 文 献

[1] 屈文新，樊晓桠，张盛兵. 多核多线程处理器存储技术研究进展[J]. 计算机科学，2007，34(4)：13-16

[2] Chang J. Cooperative caching for chip multiprocessors [D]. Madison，Wisconsin，USA：University of Wisconsin at Madison，2007：385-396

[3] Subramanian R，Smaragdakis Y，Loh G H. Adaptive caches：effective shaping of cache behavior to workloads [C]// Proc. of the 39th Annual IEEE/ACM Int Symp on Microarchitecture. Orlando，Florida，USA：IEEE，2006

[4] Kim C，Burger D，Keckler S. An adaptive，non-uniform cache structure for wire-dominated on-chip caches [C]// Proc. of the Int Conf on Architectural Support for Programming Languages and Operating Systems. San Jose，California：ACM，2002：211-222

[5] Liu C，Sivasubramaniam A，Kandemir M. Organizing the last line of defense before hitting the memory wall for cmps [C]// Proc. of the 10th Int Symp on High Performance Computer Architecture. Madrid，Spain：IEEE，2004：176-185

[6] Chishti Z，Powell M，Vijaykumar T. Distance associativity for high-performance energy-efficient non-uniform cache architectures [C]// Proc. of the 36th Annual Int Symp on Microarchitecture. San Diego，CA，USA：IEEE，2003：55-66

[7] Albonesi D H. Selective cache ways：on-demand cache resource allocation [C]// Proc. of the 32nd Annual ACM/IEEE Int Symp on Microarchitecture. Haifa，Israel：IEEE，1999：248-259

[8] Qureshi M K，Thompson D，Patt Y N. The v-way cache：demand based associativity via global replacement [J]. SIGARCH Computer Architecture News，2005，33(2)：544-555

[9] Dybdahl H. Architectural techniques to improve cache utilization [D]. Trondheim，Norway：Norwegian University of Science and Technology，2007

[10] Chang J，Sohi G S. Cooperative cache partitioning for chip multiprocessors [C]// Proc. of the 21st Annual Int Conf on Supercomputing(ICS'07). Seattle，Washington：ACM，2007：242-252

[11] Yeh T Y，Reinman G. Fast and fair：data-stream quality of service [C]// Proc. of the 2005 Int Conf on Compilers，Architectures and Synthesis for Embedded Systems(CASES). San Francisco，California，USA：ACM，2005：237-248

[12] Qureshi M K，Patt Y N. Utility-based cache partitioning：a low-overhead，high performance，runtime mechanism to partition shared caches [C]// Proc. of the 39th Annual IEEE/ACM Int Symp on Microarchitecture. Orlando，Florida，USA：IEEE，2006：423-432

[13] Stone H S，Turek J，Wolf J L，et al. Optimal partitioning of cache memory [J]. IEEE Trans on Computers，1992，41(9)：1054-1068

[14] Suh G E，Rudolph L，Devadas S. Dynamic partitioning of shared cache memory [J]. The Journal of Supercomputing，2004，28(1)：7-26

[15] Hsu L R，Reinhardt S K，Iyer R，et al. Communist，utilitarian，

and capitalist cache policies on cmps: caches as a shared resource [C]// Proc. of the 15th Int Conf on Parallel Architectures and Compilation Techniques. Seattle, Washington, USA: ACM, 2006: 13-22

[16] Lin J, Lu Q, Ding X, et al. Gaining insights into multicore cache partitioning: bridging the gap between simulation and real systems [C]// Proc. of the 14th Int Symp on High-performance Computer Architecture (HPCA 14). Salt Lake City, Utah: IEEE, 2008: 367-378

[17] Zhang M, Asanovic K. Victim replication: maximizing capacity while hiding wire delay in tiled chip multiprocessors [C]// Proc. of the 32nd Annual IEEE/ACM Int Symp on Computer Architecture. Madison, Wisconsin, USA: IEEE, 2005: 336-345

[18] Zhang M, Asanovic K. Victim migration: dynamically adapting between private and shared cmp caches[R]. MIT, 2005

[19] Beckmann B M, Marty M R, Wood D A. ASR: adaptive selective replication for cmp Caches [C]// Proc. of the 39th Annual IEEE/ACM Int Symp on Microarchitecture. Orlando, Florida, USA: IEEE, 2006: 443-454

[20] Chang J, Sohi G S. Cooperative caching for chip multiprocessors [C]// Proc. of the 33rd Annual Int Symp on Computer Architecture. Boston, MA, USA: IEEE, 2006: 264-276

[21] Chishti Z, Powell M, Vijaykumar T. Optimizing replication, communication, and capacity allocation in cmps [C]// Proc. of the 32th Int Symp on Computer Architecture. Madison, Wisconsin, USA: IEEE, 2005: 357-368

[22] Qureshi M K. Adaptive spill-receive for robust high-performance caching in cmps [C]// Proc. of the 15th Int Symp on High-performance Computer Architecture. Raleigh, North Carolina, USA: IEEE, 2009: 45-54

[23] Kim S, Chandra D, Solihin Y. Fair cache sharing and partitioning in a chip multiprocessor architecture [C]// Proc. of PACT 2004. Antibes, Juan-les-Pins, France: IEEE, 2004: 111-122

[24] Iyer R. CQoS: a framework for enabling qos in shared caches of cmp platforms [C]// Proc. of ICS'04. Malo, France: ACM, 2004: 257-266

[25] Chiou D, Jain P, Rudolph L, et al. Application-specific memory management for embedded systems using software-controlled caches [C]// Proc. of Design Automation Conference. Los Angeles, CA, USA: ACM, 2000: 416-419

[26] Rafique N, Lim W T, Thottethodi M. Architectural support for operating system-driven cmp cache management [C]// Proc. of PACT 2006. Seattle, Washington, USA: ACM, 2006: 2-12

[27] Dybdahl H, Stenstrom P. An adaptive shared/private nuca cache partitioning scheme for chip multiprocessors [C]// Proc. of HPCA 2007. Phoenix, Arizona: IEEE, 2007: 2-12

[28] Suh G E, Devadas S, Rudolph L. A new memory monitoring scheme for memory-aware scheduling and partitioning [C]// Proc. of HPCA 2002. Cambridge, Mass: IEEE, 2002: 117-128

[29] Sherwood T, Perelman E, Hamerly G, et al. Automatically characterizing large scale program behavior [C]// Proc. of ASPLOS-X. San Jose, California: ACM, 2002: 45-57

[30] Sherwood T, Perelman E, Hamerly G, et al. Discovering and exploiting program phases [J]. IEEE MICRO, 2003, 23(6): 84-93

[31] Mattson R L, Gecsei J, Slutz D R, et al. Evaluation techniques for storage hierarchies [J]. IBM Journal of Research and Development, 1970, 9(2): 78-117

[32] Moreto M, Cazorla F J, Ramirez A, et al. MLP-aware dynamic cache partitioning [C]// LNCS 4917: Proc of HiPEAC'08. Goteborg, Sweden: Springer, 2008: 337-352

[33] Sugumar R A, Abraham S G. Set-associative cache simulation using generalized binomial trees [J]. ACM Trans on Computer System, 1995, 13(1): 32-56

[34] Li X, Negi H S, Mitra T, et al. Design space exploration of caches using compressed traces [C]// Proc. of ICS'04. Malo, France: ACM, 2004: 116-125

[35] Suh G E, Rudolph L, Devadas S. Dynamic cache partitioning for simultaneous multithreading systems [C]// Proc. of the Int Conf on Parallel and Distributed Computing and Systems. Richardson, Texas, USA: IASTED, 2001: 16-127

[36] Bitirgen R, Ipek E, Martnez J F. Coordinated management of multiple interacting resources in chip multiprocessors: a machine learning approach [C]// Proc. of the 41st Annual IEEE/ACM Int Symp on Microarchitecture. Lake Como, Como, Italy: IEEE, 2008: 318-329

[37] Iyer R, Zhao L, Guo F, et al. QoS policies and architecture for cache/memory in cmp platforms [J]. SIGMETRICS Performance Evaluation Review, 2007, 35(1): 25-36

[38] Planas M M, Cazorla F, Ramirez A, et al. Explaining dynamic cache partitioning speed ups [J]. IEEE Computer Architecture Letters, 2007, 6(1): 1-4

[39] Chaudhuri M. PageNUCA: Selected Policies for Page-grain Locality Management in Large Shared Chip-multiprocessor Caches [C]// Proc. of the 15th Int Symp on High-performance Computer Architecture. Washington, DC, USA: IEEE Computer Society, 2009, 227-238

[40] Jin L, Lee H, et al. A flexible data to L2 cache mapping approach for future multicore processors[C]// Proc. of the 2006 Workshop on Memory System Performance and Correctness. New York, NY, USA: ACM, 2006: 92-101

[41] Hardavellas N, Ferdman M, Falsafi B, et al. Reactive NUCA: near-optimal block placement and replication in distributed caches [C]// Proc. of the 36th Annual IEEE/ACM Int Symp on Computer Architecture. Austin, TX, USA: IEEE, 2009: 184-195

[42] Awasthi M, Sudan K, et al. Dynamic Hardware-assisted Software-controlled Page Placement to Manage Capacity Allocation and Sharing within Large Caches [C]// Proc. of the 15th Int Symp on High-performance Computer Architecture. Washington, DC, USA: IEEE Computer Society, 2009: 250-261

[43] Guz Z, Keidar I, Kolodny A, et al. Nahalal: cache organization for chip multiprocessors [J]. IEEE Computer Architecture Letters, 2007, 6(1): 21-24

[44] Guz Z, Keidar I, Kolodny A, et al. Utilizing shared data in chip multiprocessors with the nahalal architecture [C]// Proc. of the 20th Annual ACM Symp on Parallelism in Algorithms and Architectures. Munich, Germany: ACM, 2008: 1-10

[45] Merino J, Puente V, Prieto P, et al. SP-NUCA: a cost effective dynamic non-uniform cache architecture [J]. SIGARCH Comput. Archit. News, 2008, 36(2): 64-71

[46] Dahlin M, Wang R, Anderson T E, et al. Cooperative caching: using remote client memory to improve file system performance [C]// Proc. of the 1st OSDI. Monterey, California, USA: USENIX, 1994: 267-280

[47] Zhao L, Iyer R, Upton M, et al. Towards hybrid last level caches for chip multiprocessors [J]. SIGARCH Comput. Archit. News, 2008, 36(2): 56-63