多核缓存优化技术研究综述①

闵庆豪[1,2,3], 张为华[3]

[1](复旦大学 软件学院, 上海 201203)
[2](复旦大学 上海市数据科学重点实验室, 上海 201203)
[3](复旦大学 并行处理研究所, 上海 201203)

摘 要: 随着多核技术的迅速发展, 并发处理和大批量数据操作成为主流, 而为了应对更加复杂的程序行为和愈发庞大的数据量, 缓存系统的效率也正面临着严重的挑战. 如何在复杂的多核环境中更高效的使用缓存, 提高缓存响应速度和数据吞吐量一直是体系结构领域的重要课题和研究热点. 针对多核环境中缓存的应用场景进行分析, 从缓存的效率, 内容和共享使用三个角度进行归纳和总结, 提出缓存应用的时延问题, 容量问题, 共享问题等具有针对性的问题, 并且对针对这些具体问题和情境的缓存优化技术进行总结和综述, 同时对缓存优化的一些新技术和新的优化角度进行探讨, 最后对多核缓存优化技术的发展前景进行展望.

关键词: 缓存; 性能优化; 多核架构; 替换算法优化; 程序调度

Survey of Cache Techniques for Multicore Architecture

MIN Qing-Hao[1,2,3], ZHANG Wei-Hua[3]

[1](Software School, Fudan University, Shanghai 201203, China)
[2](Shanghai Key Laboratory of Data Science, Fudan University, Shanghai 201203, China)
[3](Parallel Processing Institute, Fudan University, Shanghai 201203, China)

Abstract: With the developing of multi-core techniques, concurrent processing and big-data operation are becoming more and more popular. Facing the complexity of program behavior and the large amount of input data, the cache system needs to explore more optimization opportunities. Improving the utizlization, efficiency and throughput of cache system is always the hotspot of architectural researches. This survey analyzes the inner details of cache techniques and concludes the important issues on multi-core cache optimization, like latency problem, capacity problem and sharing problem. Furthermore, this paper introduces some typical solutions for these issues, as well as the new techniques and new optimization aspects in recent researches. Finally, a description of possible developing directions is presented at the end of this survey.

Key words: cache system; multi-core architecture; performance optimization; replacement policy optimization; program scheduling

随着处理器处理速度的提升, 处理器的运算速率和内存读写速率有着严重的不匹配, 即所谓的“内存墙”(memory wall)问题, 这使得访问内存的延时变得不可忍受. 为了更好的利用应用程序的空间局部性(space locality)和时间局部性(time locality), 缓存技术将内存中的部分数据放到读写速率快得多的临时存储器(缓存)中作为缓冲, 这样当处理器需要操作这些数据时, 就可以从缓存中调用, 从而避免访问内存带来的巨大延时, 加快数据操作的速率.

通常的处理器采用二级或三级缓存架构, 级数越大, 缓存的容量越大, 相应的访问速度也越慢. 处理器在需要调用数据时, 会先从一级缓存(最高级缓存)开始检索需要的数据, 再依次向低层缓存检索. 若所有的缓存都不命中, 即所有的缓存中都没有保存所需

① 基金项目:上海市科委科技攻关项目(13DZ1108800);国家高技术研究发展计划(863)(2012AA010901);国家自然科学基金(61370081)
收稿时间:2014-04-21;收到修改稿时间:2014-05-26

要的数据, 处理器会从内存中读取数据, 然后将数据放入缓存中. 表 1 给出了 Intel i3-2100 处理器缓存的实测速度和内存速度的比较数据.

表 1 Intel i3 处理器缓存速度和内存速度比较

	读速度	写速度	访问时延
一级缓存	98.9MB/s	49.4MB/s	1.3 ns
二级缓存	55.8MB/s	29.5MB/s	3.9 ns
三级缓存	22.0MB/s	19.0MB/s	6.4 ns
内存	15.9MB/s	16.4MB/s	54.2ns

从表 1 中的数据可以看出, 内存的访问时延非常大, 所以当前缓存技术的研究大多集中在提高缓存命中率, 尽量减少内存的直接读写等方面.

随着多核技术的迅速发展, 缓存的重要性越发不可忽视, 其组织和设计也越发需要更多的创新和发展. 事实上, 随着芯片上集成核的增多, 处理器和内存速度差异的增大, 还有应用程序日趋庞大的工作集(working set)规模, 都使得缓存对整体性能有着至关重要的影响, 同时也面临着更多的挑战. 如何加快缓存访问的读写速度, 如何更高效的利用缓存资源, 如何更合理的对缓存资源进行分配等问题, 一直是各种缓存研究的热点所在.

本文对常见的多核缓存优化技术进行分析和研究, 综述常见的缓存优化技术和理念, 并且对缓存优化技术的发展前景进行展望. 本文的组织结构如下: 首先对当前常见的多核缓存的结构和优化技术进行分析, 并且从缓存的效率, 内容和共享使用等三个方面总结出多核环境中缓存技术所需要考虑的几个主要问题, 即时延问题, 容量问题和共享问题等. 在此基础上, 对当前热点的缓存优化技术的研究进行了分类和归纳, 总结常见的缓存优化方案所使用的技术, 并且对解决以上各种问题的典型的优化技术进行简要介绍与分析. 同时, 本文还对当前缓存优化技术的一些新的优化方向和优化理念进行分析, 并将其与传统的优化技术进行比较. 最后对各种缓存优化技术进行总结, 并在此基础上对缓存优化技术未来可能的发展方向进行了展望.

1 多核缓存结构及其优化技术简介

本小节主要描述了多核缓存的基本结构, 并且对一些基础的优化重点进行分析介绍.

1.1 多核缓存的基本结构

相对于单核体系结构来说, 在多核体系结构中, 缓存资源的组织方式更加复杂. 以常见的二级缓存结构为例, 第一级缓存的读写速率最高, 容量最小, 所以通常会和处理器同时集成在一个核上, 而且为每个核所私有. 通常第一级缓存会设计成指令缓存和数据缓存相分离的结构, 以便更有效地访问缓存资源. 第二级缓存(最底层缓存)大多采用指令缓存和数据缓存共存的组织方式, 相对来说其容量更大, 所以通常会分布在各个不同的核上, 每个核可以都访问自己核上的第二级缓存资源. 但是对于其他核上的第二级缓存资源, 不同的缓存架构有不同的处理方式, 如图 1 中的示意图所示, 分为共享和私有两种架构.

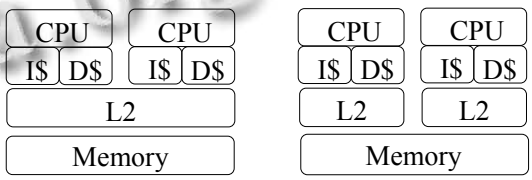

图 1 共享和私有二级缓存结构

在这两种多核缓存结构中, 私有的最底层缓存只由一个核所使用, 各个核之间只能通过共享的内存或者专用的核间通信通道相联系; 而共享的最底层缓存可以被多个核或所有核所共享, 缓存中所有的数据都可以被共享的其他核核访问和使用.

这两种架构各有优劣. 私有最底层缓存通常结构和检索过程相对较简单, 访问延时比较小, 同时由于其隔离性, 各个核之间的影响比较小; 但缺点是对于每个核来说, 可用的缓存资源相对比较少, 因而缓存的命中率较低. 共享最底层缓存则允许一个核访问全部共享的最底层缓存, 可用缓存资源比较多, 所以其命中率较高; 但是这种结构需要所有的核都可以任意访问所有共享最底层缓存的数据, 所以结构相对复杂, 从共享缓存中读写数据时会造成较大的访问时延; 同时, 各个核共同使用同样的缓存资源, 其相互影响可能会造成恶性竞争的不良后果.

1.2 多核缓存优化技术简介

正如上文所述, 私有最底层缓存和共享最底层缓存两种缓存架构不分高下, 其侧重点也各不相同, 当前的多核缓存优化技术也针对这两种架构, 以不同的侧重点和解决方案达到尽可能合理利用缓存资源的效果.

私有最底层缓存的缺点在于可用的缓存资源相对

较少, 所以在优化时会偏向于更合理的利用其他核上的缓存资源以提高整体缓存利用率, 例如 Chang 等人提出的缓存溢出技术[1]就可以在适当的时候将某个核上原本要被替换出去的数据“溢出”到相邻核的缓存中, 然后在需要的时候从相邻核的缓存中读取数据, 以增加可用缓存资源、提高数据复用性的方法提高缓存整体利用率.

共享缓存则偏向于限制单个核或单个程序无限制的缓存资源使用, 或者为某些特定情况划定受保护的缓存区块. 例如 Qureshi 等人在文献中提出的 UCP 缓存分区技术[2]就把单个核或单个程序的缓存资源使用限制在一定的分区中, 以减少某个核过度占用缓存资源的情况, 降低对其他核的影响; Sanchez 等人提出了更细粒度的以缓存行(cache line)为单位的 Vantage 分区技术[3], 进一步优化分区技术的效果; 而 Manikantan 等人提出的 NUcache 缓存分区技术[4]则针对某些命中率低的 PC, 为其划定特殊的分区以减少其被替换的机会, 从而提高其命中率.

还有些缓存设计如 Herrero 等人设计的 ElasticCC 缓存架构[5]则试图将私有最底层缓存和共享最底层缓存的优点结合起来, 使用分布式分区技术动态的划分私有缓存区块和共享缓存区块, 以达到合理使用缓存资源的目的.

综上所述, 虽然多核缓存优化技术的针对目标和侧重点各不相同, 但是其主旨都主要侧重在合理利用缓存资源, 以提高缓存命中率和减少访问时延的方式在一定程度上缓解“内存墙”问题.

2 多核环境中缓存技术的重要问题

作为体系结构研究的热点之一, 缓存优化技术试图从各个可能的方面优化缓存的整体性能和效率, 具体来说, 是减少访问时延和提高缓存的命中率两方面. 此外, 在多核体系结构中, 各个核对缓存资源的共享, 同一个核甚至同一个程序中对缓存资源的使用也需要良好的调度策略. 因而本章将主要介绍多核缓存的三个主要问题, 时延问题、容量问题和共享问题.

2.1 多核缓存的时延问题

缓存的访问时延与缓存的性能息息相关, 尤其在多核缓存相对更加复杂的架构中, 更需要谨慎的设计或合理的机制以减少访问时延.

此外, 集成了多个处理器核心的芯片上, 根据处理器和缓存资源的相对位置, 访问芯片上不同位置的缓存资源需要不同的时延, 其差距可以达到几倍甚至十几倍[6], 如此一来处理器或高层缓存与共享最底层缓存或内存的相对位置对访问时延就有着至关重要的影响. 显然访问远端资源要比访问附近的资源需要更长的时延, 所以如何有效地调度数据或程序也是其中一个关键所在.

2.2 多核缓存的容量问题

除了缓存的访问时延之外, 其命中率(或缺失率)对程序的性能有着至关重要的影响. 而影响缺失率的一个重要方面就是缓存的容量. 受到技术、成本、硬件等方面的限制, 缓存的容量本身很难进一步提升, 故而分析程序特性和其对缓存容量的需求就是一个可行的手段.

常见的缓存的容量问题有两种, 浏览访问(Scanning)和缓存抖动(Thrashing). 其中, 浏览访问实际上指的是一种浏览式的访问类型, 即程序会一直访问不同的数据, 数据复用的情况比较少. 当程序的工作集规模非常大时, 缓存的容量远远无法满足程序需要, 从而造成很大的缓存缺失率. 尤其在共享缓存中, 这种访问类型会严重影响其他程序的缓存使用, 造成更严重的缓存资源不合理使用. 而缓存摆动通常出现于工作集规模比缓存容量略大的情况下, 由于传统的基于 LRU 策略的替换算法会替换出缓存中最长时间未被使用的数据, 而程序工作集规模比缓存容量略大, 所以这些数据实际上很快就会被重新使用, 反而是刚被放入缓存的数据要很久才会被再次访问, 所以这就使得了这些程序对缓存容量非常敏感, 在缓存容量够大时有很高的命中率, 而在缓存容量不足时命中率迅速降低, 对程序性能影响很大.

2.3 多核缓存的共享问题

由于多核环境下, 每个核都在执行相同或不同的程序, 故而对于共享的缓存资源会有一定的竞争. 普通的缓存替换策略并没有考虑到这种竞争问题, 所以不合理的资源分配可能会导致缓存资源需求很大的程序频繁竞争相同一部分共享缓存资源, 导致这一部分缓存被频繁替换, 命中率很低, 而其他部分缓存资源长时间空闲的情况出现. 同时, 来自不同核的缓存访问也可能会引发类似上文提到的缓存抖动问题, 使得缓存的使用效率远远低于理想水平.

而在处理器处理多并发任务的时候, 每个核上可

能需要同时处理多个并发任务, 这些并发任务同样会竞争缓存资源, 普通的缓存架构设计也很难保证这些任务的合理调度, 从而导致缓存资源的不合理利用等问题.

甚至在同一个程序中, 也会有共享资源的竞争问题存在, 比如普通的缓存访问和预读取缓存访问就会竞争有限的缓存资源, 这同样需要一个良好的调度和控制方案才能达到更好的缓存使用效果.

2.4 多核缓存的其他问题

相对于单核来说, 多核环境的缓存结构更加复杂, 需要考虑缓存一致性, 即多个核上对于同一个内存地址的数据的缓存需要保持一致, 当一个核上的数据有改动时, 其他核上的数据也要相应的进行改动或将其数据无效化. 而为了保证这种数据一致性, 通常需要额外的机制和硬件资源来作为保证, 这些机制对多核缓存的可扩展性造成了一定的限制, 对其性能也有着一定的影响.

此外, 多核缓存结构还有着一些其他常见的具体问题, 如最底层缓存的数据复用率较低、使用物理地址索引的缓存查询需要预先经过 MMU 将程序的虚拟地址转换成物理地址等. 限于篇幅本文不再详细介绍这些问题, 而是主要关注于上文提到的时延问题、容量问题和共享问题.

3 针对性的多核缓存优化技术简介

针对前文提到的时延问题、容量问题和共享问题等具体情境, 当前也有着许多很有针对性的研究和解决方案.

3.1 针对时延问题的优化技术

为了解决多核缓存结构中的时延问题, 预读取技术是一种很常用的手段. 顾名思义, 预读取技术会在程序真正用到数据之前预先把数据读入缓存, 这样可以提高缓存的命中率, 减少访问时延. 事实上, 预读取技术很早就被使用, 而在多核环境下, 预读取技术的效果更加明显. 例如 Kamruzzaman 等人提出的 IoP 预读取技术[7]就使用了一个辅助线程在另外一个核上预读取数据, 在预读取了足够的数据之后, 将主线程迁移到这个核上继续执行, 同时辅助线程又回到原来的核上继续进行预读取操作. 这种调度技术成功的利用了多核资源进行预读取操作, 最终达到降低主线程访问时延, 加速程序性能的效果.

此外, Kim 等提出的 D-NUCA 非统一缓存架构[6]针对每个核访问不同位置的缓存资源时有不同的访问时延的问题作了研究. 在传统的统一缓存架构(UCA)中, 缓存架构是统一架构的, 因而访问任意位置的缓存资源的时延是固定的, 而在 D-NUCA 架构中缓存资源根据其位置分为不同的区域, 每个核访问不同位置的缓存资源时有不同的访问时延, 这样访问附近缓存资源的时延就会大大减少. 同时, D-NUCA 架构还提供缓存数据迁移功能, 使需要的数据尽可能处在与请求数据的核比较靠近的位置, 减少数据读取的时延, 优化缓存的效率. 而 Hardavellas 等人提出的 R-NUCA 技术[8]在此基础上通过与操作系统的协同工作提出了更智能的缓存数据存放方案, 通过判断缓存访问的模式, 进一步优化了数据存放的位置, 使得数据的存放位置更加合理, 缓存中的数据管理更加高效, 从而减少访问缓存资源所需要的时延.

3.2 针对容量问题的优化技术

由上文所述, 当程序的工作集规模和缓存容量接近时, 程序会对缓存容量非常敏感, 当缓存容量小于工作集规模时, 会出现缓存抖动的情况. 为了解决传统 LRU 替换策略的这一问题, Qureshi 等[9]提出可以在适当的时机使用和 LRU 替换策略相反的替换策略, 即在缓存替换时首先替换最新使用过的缓存行, 以保证旧的数据会被保存在缓存中; 同时在程序执行时在这两种替换策略中动态选择, 保证这两种替换策略都被用在合理的时机, 即 DIP 技术. 这一技术可以明显缓解缓存抖动所带来的性能下降等问题, 也为进一步的优化提供了基础.

Jaleel 等[10]对缓存的替换行为进行了更系统的分析, 提出了缓存的再访问间隔(Re-Reference Interval)的概念, 通过预测缓存的再访问间隔来决定是否将其保留在缓存中或者替换出去, 通过替换最长再访问间隔的缓存行达到更好的缓存资源使用效果. 在此基础上的 D-RRIP 策略可以很好的解决缓存抖动问题, 同时, D-RRIP 策略对浏览访问的访问类型也有很好的效果.

Khan 等人提出的 DCS 技术[11]使用基于采样技术的预测器把共享最底层缓存分成两部分, 曾被再次访问过的部分和未被再次访问过的部分, 同时动态的决定每一个缓存资源请求会在合适的部分中进行处理. 这种技术可以保护缓存中有用的数据, 使之尽可能被

保留在缓存中, 从而使浏览访问和缓存抖动问题中的缓存访问都偏向于在未被再次访问过的部分中处理, 不会造成频繁替换更有效的缓存数据的情况.

此外, Kurian 等[12]提出了另外一种方向的解决方案, 通过基于硬件的实时监控器检测缓存数据的局部性, 并将缓存数据进行分类, 从而在缓存替换时尽量选择将局部性较低的缓存数据替换出去, 以保证缓存数据的高利用率.

3.3 针对共享问题的优化技术

针对多核环境共享最底层缓存资源时的竞争问题, 合理的调度和任务管理是一个有效的解决方案. Jalee 等人提出的 CRUISE 机制[13]把应用程序根据其缓存访问的类型和对缓存容量的需求和敏感性分为四类, 而后通过合理的调度, 优先保证缓存需求和敏感性高的应用程序, 以保证最高的缓存使用性价比, 从而解决多核环境中的资源竞争问题.

Wu 等人则针对普通缓存访问和预读取缓存访问的缓存资源竞争问题, 提出了 PACMan 缓存管理策略[14]. 这种策略对于普通缓存访问和预读取缓存访问采取不同的处理方式, 并且在程序运行时通过调整缓存插入和替换机制动态地调整二者的相对比重, 以预读取缓存访问不会影响到程序正常的缓存使用, 达到最优的解决方案.

3.4 其他的缓存优化新技术

近年来, 除了前文中的一些比较传统的针对缓存的优化机制之外, 还有许多新技术从许多新颖的角度和切入点对缓存进行优化, 极大的拓宽了缓存优化技术的研究方向和发展空间, 如压缩缓存技术, 虚拟缓存技术, 针对缓存一致性协议的优化技术, 和针对访问粒度的优化技术等.

例如, Baek 等人提出的 ECM 技术[15], 和 Arelakis 等人的工作[16]就针对压缩缓存(compressed cache)做了分析和研究, 提出更适应压缩缓存的替换策略和缓存管理方案. Kim 等人提出的 residue cache 技术[17]则分析发现并不是缓存行中存储的所有数据都会被立即访问, 即部分命中的情况, 同时额外使用小部分剩余缓存和压缩缓存协同工作, 很好的达到了减少缓存缺失率, 提高缓存使用效率的效果.

而 Basu 等[18]使用的虚拟缓存(virtual cache)技术则试图用虚拟地址对缓存进行检索, 以优化传统的先将虚拟地址转化为物理地址, 再通过物理地址检索缓存的复杂操作. Kaxiras 等[19]则针对虚拟缓存一致性协议中频繁出现的物理地址反翻译为虚拟地址的情况进行分析, 提出为最底层缓存所使用统一的共享 TLB 和基于虚拟地址的一致性协议, 很好的减少了反翻译情况的出现.

由于多核缓存系统的缓存一致性协议随着核数的增多而越发复杂, 而基于目录的缓存一致性协议由于其目录容量的限制, 无法在目录中存放所有缓存资源的共享信息, 当这些共享信息在目录中被替换出去时, 其相应的缓存数据也需要被移出缓存, 以保证数据访问的正确性, 这进一步加重了缓存一致性协议的影响. Cuesta 等人提出的 Deactivating Coherence 技术[20]就对缓存中的共享数据和私有数据进行区分, 以合理使用缓存目录. Kurian 等[21]则在更细粒度的缓存行上对私有和共享数据进行管理, 试图在缓存中保留共享数据和空间局部性高的私有数据, 已达到缓存资源的更有效的使用. Demetriades 等人的工作[22]中也提出替换保存私有数据的缓存一致性信息以提高缓存一致性信息的有效性, 从而进一步减少一致性协议对缓存本身的影响.

而针对一致性协议的粒度, Zhao 等人提出的 Protozoa 技术[23]则针对过于细粒度的一致性协议会带来额外的通信和数据传输开销的情况, 通过动态调整一致性协议粒度的方式, 优化了一致性协议的通讯和数据传输, 提高数据的使用效果. 除了一致性协议的粒度之外, Yoon 等人提出的 DGMS 技术[24]对内存访问的粒度进行调整, 动态的改变内存访问的粒度, 以减少许多不必要的数据被读取出来放入缓存资源中, 以合理的使用缓存资源.

总体来说, 这些技术虽然是从以前鲜有人关注的新角度或针对某些应用场景所提出, 但实际上还是为了解决缓存资源更加合理使用的核心问题. 而另一方面, 这些技术和传统的缓存优化技术也可以很好的协同工作, 从而进一步优化缓存的使用效率.

4 多核缓存优化技术分析和展望

本节中, 我们对各个缓存的优化方案进行了比较与总结, 并在此基础上分析了多核缓存优化技术所面临的挑战并对将来的发展进行展望.

4.1 各缓存优化方案的比较与总结

对于缓存优化技术的优化效果, 表 2 列出了部分

缓存优化方法优化效果的比较. 但是由于各种缓存优化方法所针对的问题和解决方案各不相同, 其测试环境和测试基准也各不相同, 因此表 2 中的数据仅仅是作为参考, 不能用作直接比较的数据标准.

表 2 部分缓存优化技术优化效果比较

优化技术	优化效果
IoP	31~63% 性能提升
NUCA	50% 性能提升
R-NUCA	6~14% 性能提升
DIP	21% 缓存缺失率减少
DRRIP	10% 性能提升
DCS	5~12% 性能提升
CRUISE	接近理想缓存性能
PACMan	21.5% 性能提升

虽然以上各种缓存优化方案所针对的问题, 使用的技术和侧重点, 优化效果都各有不同, 但是总体来说, 大部分的优化技术主要集中在三个方面: 预读取, 替换算法优化和调度. 表 3 列出了上文提到的部分缓存优化方法所使用到的技术.

表 3 部分缓存优化技术分析

优化技术	预读取	替换算法优化	调度
IoP	√		
NUCA			√
DIP		√	
DRRIP		√	
DCS		√	
CRUISE			√
PACMan	√		√

从表 3 中可以看出, 这些优化技术都有着一定的针对性, 针对缓存的具体问题采用了各种可能的方法进行优化处理. 但是, 事实上多核缓存的时延问题、容量问题和共享问题等具体问题并不是独立的, 而是互相之间有所交叉, 比如 NUCA 架构的时延问题同时也是缓存资源共享问题在访问时延上的具体体现, 而容量问题同样和多个核或应用程序共享缓存资源的情境密切相关. 同样的, 这些具体问题的解决方案都有一定的针对性, 但是也都同时涉及到了其他的一些问题. 比如 Jaleel 等人提出的 D-RRIP 技术[10]可以针对浏览访问的缓存访问类型有很好的处理效果, 但是 D-RRIP 技术同样会限制浏览访问缓存的应用程序过多的占用缓存资源, 从而为共享缓存资源的其他应用程序提供了更多可用的缓存资源, 这也在某种程度上解决了多核缓存的共享问题.

因此, 多核缓存的这些具体问题和优化方法并不绝对, 相互之间都有着一定的联系, 或者说, 如果需要达到更好的优化缓存资源使用这个目标, 需要同时从多个方面进行考虑和设计, 而不仅仅是从一个方面针对某一个具体问题进行分析和处理, 这就需要一种统一而全面的缓存优化方案.

同时, 以上介绍的许多缓存优化技术都同时使用了两个或多个不同或者相反的策略, 然后在程序执行时动态的选择使用其中最合适的策略. 从这一点来说, 以上提到的许多技术都可以结合在一起使用, 比如缓存替换策略和多核调度策略可以比较容易的结合起来, 预读取技术和 NUCA 技术也可以很简单和协同工作, 这为更统一而全面的缓存优化方案提供了可能.

然而, 当下还没有这种统一而全面的缓存优化方案, 也鲜有人对此进行研究和分析, 这些技术的结合使用还需要进一步的探索.

此外, 除了这些比较传统的针对缓存的优化机制之外, 还有许多新技术从许多新颖的角度和切入点对缓存进行优化, 如虚拟缓存技术, 针对缓存一致性协议的优化技术等, 这些技术从新的角度和新的方向为缓存优化技术引入了极大的创意和前景, 也为缓存优化技术提供了更多的可能. 同时, 这些新技术完全可以和比较传统的缓存优化技术相结合, 以进一步提高缓存的使用效率.

但是值得注意的是, 这些技术目前还处于十分不成熟的阶段, 或者只是针对某些特定场景, 适用性有一定的限制, 因而还需要更进一步的研究, 才能使得这些技术更加成熟可靠.

4.2 缓存优化技术的挑战和展望

由以上分析可以看出, 虽然当下多核缓存优化技术的研究角度十分广泛, 成果也十分显著, 但是依然存在着一定的问题, 也面临着很大的挑战:

① 当下的多核缓存优化技术都有着一定的针对性, 但是缺少比较全面而系统的解决方案, 从更高的层次和更大的范围来进行考虑和分析, 从而设计出更完备更实用的优化技术;

② 多核缓存优化技术的创新还需要进一步的发展, 当下的新技术还不是十分成熟, 也没有和传统的

技术很好的结合, 所以还有很大的潜力和发展空间;

③ 许多动态缓存优化技术的核心理念都是在两个或多个策略中进行比较, 然后在程序运行时选择使用其中最合适的策略. 这种技术大多基于一种缓存集比较(Set Dueling)的机制实现, 即对于每一种备选策略, 都有固定的一些缓存集来使用, 而在程序运行时通过这些固定采用每种备选策略的缓存集的实时结果来确定那种策略是更合适的策略. 这种选择机制虽然可以很好的使用于策略的动态调整, 但是这些被预先设定采用每种备选策略的缓存集实际上消耗了一部分缓存资源, 同时也可能带来很大的硬件改动, 增加了实际开发和应用的难度和代价;

④ 目前大多数缓存优化技术都偏向于对缓存本身进行分析和优化, 而对于其他因素的考虑较少, 如操作系统参与调度, 优化缓存和处理器及内存的通讯机制等;

⑤ 当下对于多核缓存优化技术的研究大多都处于科研阶段, 许多测试用例都比较简单, 并没有在实际场景中和工业上进行验证, 因而与真正的应用和推广还有很大的距离.

因此, 我们对多核缓存优化技术的发展前景进行了展望, 提出了以下一些值得探索的新方向:

① 更全面的分析多核缓存结构中存在的问题, 从更全局的角度设计优化方案, 而不仅仅是针对某一类特定问题进行分析;

② 从新角度对多核缓存进行分析优化, 并且将这些技术和传统技术结合起来, 这些新技术度包括目前已经有的虚拟缓存技术、压缩缓存技术等技术, 和其他的如优化数据传输机制, 设计更智能的数据一致性协议等等;

③ 更合理而灵活的动态策略选择机制, 针对应用程序的特性使用针对性的方案, 可以通过操作系统或硬件提供的信息进行辅助, 已达到更好的优化效果;

④ 关注缓存之外的其他切入点, 将缓存作为硬件整体的一部分来考虑, 如优化缓存和内存数据传输范型, 提高每次数据传输中有效数据的比例, 与双通道甚至多通道的内存协同工作等等;

⑤ 从实际应用的角度出发, 提出针对实际应用场景的优化技术, 而不仅仅是在实验室中通过测试用例进行理论研究;

⑥ 与硬件厂商等具有应用和工业经验的研究人员协同工作, 提高优化技术的可行性.

5 结语

随着多核技术的迅速发展, 并发处理和大批量数据操作成为主流, 而为了应对更加复杂的程序行为和愈发庞大的数据量, 缓存系统的效率也正面临着严重的挑战. 如何在复杂的多核环境中更高效的使用缓存, 提高缓存响应速度和数据吞吐量一直是体系结构领域的重要课题和研究热点. 本文针对缓存技术中的时延问题、容量问题和共享问题等具体问题进行分析, 综述了一些针对性的解决方案, 如 NUCA 架构, 再访问间隔预测和智能调度等策略, 同时也对这些策略的使用和限制进行了总结. 此外, 本文还总结了当下缓存优化技术研究存在的问题和面临的挑战, 同时对缓存优化技术的发展前景进行了展望. 可以预见, 缓存技术的创新和发展会更加保证处理器的高性能工作, 也会为多核技术的进一步成熟发展提供强有力的支持.

参考文献

1 Chang J, Sohi GS. Cooperative caching for chip multiprocessors. IEEE Computer Society, 2006, 34(2): 264–276.

2 Qureshi MK, Patt YN. Utility-based cache partitioning: A low-overhead, high-performance, runtime mechanism to partition shared caches. Proc. of the 39th Annual IEEE/ACM International Symposium on Microarchitecture. IEEE Computer Society. 2006. 423–432.

3 Sanchez D, Kozyrakis C. Vantage: scalable and efficient fine-grain cache partitioning. ACM SIGARCH Computer Architecture News, 2011, 39(3): 57–68.

4 Manikantan R, Rajan K, Govindarajan R. NUcache: An efficient multicore cache organization based on Next-Use distance. Proc. of the 17th International Symposium on High Performance Computer Architecture (HPCA). 2011. 243–253.

5 Herrero E, González J, Canal R. Elastic cooperative caching: an autonomous dynamically adaptive memory hierarchy for chip multiprocessors. ACM SIGARCH Computer Architecture News, 2010, 38(3): 419–428.

6 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.

7 Kamruzzaman M, Swanson S, Tullsen D M. Inter-core prefetching for multicore processors using migrating helper threads. ACM SIGARCH Computer Architecture News, 2011, 39(1): 393–404.

8 Hardavellas N, Ferdman M, Falsafi B, et al. Reactive NUCA: near-optimal block placement and replication in distributed caches. ACM SIGARCH Computer Architecture News. ACM, 2009, 37(3): 184–195.

9 Qureshi MK, Jaleel A, Patt YN, et al. Adaptive insertion policies for high performance caching. ACM SIGARCH Computer Architecture News, 2007, 35(2): 381–391.

10 Jaleel A, Theobald KB, Steely Jr SC, et al. High performance cache replacement using re-reference interval prediction (RRIP). ACM SIGARCH Computer Architecture News. ACM, 2010, 38(3): 60–71.

11 Khan SM, Wang Z, Jiménez DA. Decoupled dynamic cache segmentation. Proc. of the 18th International Symposium on High Performance Computer Architecture (HPCA). 2012. 1–12.

12 Kurian G, Devadas S, Khan O. Locality-aware data replication in the last-level cache. Proc. of the 20th International Symposium on High Performance Computer Architecture (HPCA). 2014.

13 Jaleel A, Najaf-Abadi HH, Subramaniam S, et al. Cruise: cache replacement and utility-aware scheduling. ACM SIGARCH Computer Architecture News. ACM, 2012, 40(1): 249–260.

14 Wu CJ, Jaleel A, Martonosi M, et al. PACMan: prefetch-aware cache management for high performance caching. Proc. of the 44th Annual IEEE/ACM International Symposium on Microarchitecture. ACM. 2011. 442–453.

15 Baek S, Lee HG, Nicopoulos C, et al. ECM: Effective capacity maximizer for high-performance compressed caching. Proc. of the 19th International Symposium on High Performance Computer Architecture (HPCA). 2013. 131–142.

16 Arelakis A, Stenstrom P. SC2: A statistical compression cache scheme. Proc. of the 41st Annual International Symposium on Computer Architecture. ACM. 2014.

17 Kim S, Lee J, Kim J, et al. Residue cache: A low-energy low-area L2 cache architecture via compression and partial hits. Proc. of the 44th Annual IEEE/ACM International Symposium on Microarchitecture. ACM. 2011. 420–429.

18 Basu A, Hill MD, Swift MM. Reducing memory reference energy with opportunistic virtual caching. Proc. of the 39th International Symposium on Computer Architecture. IEEE Press. 2012. 297–308.

19 Kaxiras S, Ros A. A new perspective for efficient virtual-cache coherence. Proc. of the 40th Annual International Symposium on Computer Architecture. ACM. 2013. 535–546.

20 Cuesta BA, Ros A, Gómez ME, et al. Increasing the effectiveness of directory caches by deactivating coherence for private memory blocks. ACM SIGARCH Computer Architecture News, 2011, 39(3): 93–104.

21 Kurian G, Khan O, Devadas S. The locality-aware adaptive cache coherence protocol. Proc. of the 40th Annual International Symposium on Computer Architecture. ACM. 2013. 523–534.

22 Demetriades S, Cho S. Stash directory: A scalable directory for many-core coherence. Proc. of the 20th International Symposium on High Performance Computer Architecture (HPCA). 2014.

23 Zhao H, Shriraman A, Kumar S, et al. Protozoa: Adaptive Granularity Cache Coherence. Proc. of the 40th Annual International Symposium on Computer Architecture. ACM, 2013: 547–558.

24 Yoon DH, Jeong MK, Sullivan M, et al. The dynamic granularity memory system. ACM SIGARCH Computer Architecture News. IEEE Computer Society, 2012, 40(3): 548–559.