一种高效、可扩展细粒度缓存管理混合存储研究

姜国松
(中国传媒大学信息与通信工程博士后流动站 北京 100024)

摘 要 混合主存储器由DRAM构成,它可用作cache来扩展非易失性存储器,相比传统的主存储器能够提供更大的存储能力。不过,要使混合存储器具有高性能和可扩展性,一个关键的挑战在于需要对缓存在DRAM中的数据的元数据(如标签)以一个细粒度的方式进行有效管理。基于这样的观察:利用DRAM缓存行的局部性,将元数据与元数据对应的数据存储在片外缓存中相同的行,使用一个小的缓冲区来只缓存最近被访问的片内缓存行,以降低细粒度DRAM缓存的开销。利用这种细粒度的DRAM高速缓存的灵活性和效率,还开发了一种自适应的策略来选择在数据迁移到DRAM时最佳的迁移粒度。在搭配了512MB的DRAM缓存的混合型存储系统中,建议使用8kB的片上缓存,这样,相比一个传统的8MB的SRAM元数据存储,即使没有考虑大的SRAM元数据存储的能源开销,也可以提升6%以内的性能,以及18%的能效节约。

关键词 缓存,标签存储,非易失性存储器,混合主存储器
中图法分类号 TP311 **文献标识码** A

Research about Efficient and Scalable Hybrid Memories at Fine-granularity Cache Management

JIANG Guo-song
(Post-doctoral Research Centers of Information and Communication Engineering, Communication University of China, Beijing 100024, China)

Abstract Hybrid main memories are composed of DRAM which can provide much larger storage capacity than traditional main memories used as a cache to scalable non-volatile memories, such as phase-change memory. However, for hybrid main memories with high performance and scalability, a key challenge is to effectively manage the metadata (e.g., tags) for data cached in DRAM in a fine- granularity. Based on this observation: storing metadata on off-chip cache line in the same row as their data corresponding to the metadata exploits DRAM row buffer locality, this paper reduced the overhead of fine-grained DRAM cache by using a small buffer to cache chip cache line which has recently been accessed. We also developed an adaptive policy to choose the best granularity when migrating data into DRAM. On a hybrid memory with a 512MB DRAM cache, our proposal using an 8KB on-chip buffer can increase the performance within 6% and save 18% better energy efficiency than a conventional 8MB SRAM metadata store, even when the energy overhead due to large SRAM metadata storage is not considered.

Keywords Cache, Tag memory, Non-volatile memories, Hybrid main memories

1 引言

随着芯片尺寸的不断缩小,未来的多处理器芯片有望将越来越多的内核集成到单一的芯片上,这就增加了主存容量的总需求。由于DRAM本身的技术限制,如果单独使用它来满足上述情况对主存的需求可能会比较困难。为了解决这个问题,最近的相关研究提出了使用DRAM作为大型非易失性存储器的高速缓存,如相变存储器(phase-change memory, PCM),其访问延时和DRAM具有可比性,但相比DRAM,预计将更具可扩展性[1]。那么,在这种混合型的主存储器中的主要问题是如何有效地在一个极大容量的DRAM中以细粒度的方式管理数据的元数据(如标签,LRU,有效位和脏位)。

对于大容量、细粒度的DRAM型cache,现有的大多数基于硬件的解决方法无非就两种:(1)要么在一个大规模的SRAM结构中为每一个缓存块存储其元数据,这限制了可扩展性,并增加了开销(例如,文献[2]);(2)要么将元数据存储在DRAM中的一个连续的区域,在访问数据时,这就要求额外的访问开销,降低了存储系统的性能[3,4]。相比大容量片上SRAM元数据结构,我们的目标是实现最小的性能下降,同时伴随着幅度较低的硬件开销。

2 混合主存储器

图1显示了一个使用DRAM作为PCM存储器的高速缓存的混合型主存的组织结构。DRAM和PCM都是由多个存储组(bank)组成,每个存储组由存储单元的行和列进行组织。每个存储组(bank)设置一个行作为存储组的缓存,它存储在本存储组中最近被访问的数据行的内容。访问某个行缓存中的数据(也就是行缓存命中)可提供比访问存储器阵列(也就

到稿日期:2012-10-08 返修日期:2013-03-09 本文受湖北省自然科学基金(2011CDC078)资助。
姜国松(1976—),男,博士,副教授,主要研究方向为网络存储、存储性能评价,E-mail:hustjgs@126.com。

是行缓存未命中)更快的速度。

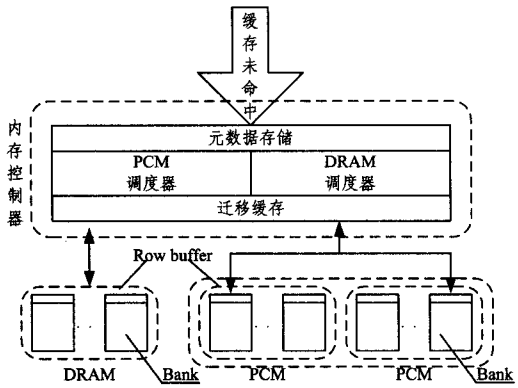


图 1 DRAM-PCM 混合型主存储器架构

混合型存储器的控制器位于芯片上,负责管理数据的存放位置、访问调度,并执行在 DRAM 和 PCM 设备之间的数据迁移。

元数据查找。跟踪位于 DRAM 缓存中的数据是否需要某种形式的元数据存储。虽然以一个大的粒度(例如,4kB)跟踪数据是可能的,但是,假如在大数据块中的大多数数据并没有被访问,这样做就可能会导致未充分利用 DRAM 高速缓存,浪费带宽,使得大粒度数据迁移的效率低下,最终导致不可取。另一方面,细粒度跟踪数据可能会导致较大的存储开销(例如,在 512MB 的 DRAM 缓存中,需要 8MB 的空间来跟踪所有的 128B 个数据块)。还有些人提出了将元数据放在 DRAM 缓存本身,与数据放在一起[3-5]。这样做虽然可以缓解由于在片上存储元数据带来的存储过载和可扩展性限制,但是,它仍然需要访问 DRAM 来获取额外的元数据,这样就增加了主存的延时和带宽消耗。

请求调度。基于检索到的元数据,数据请求要么放在 DRAM 调度器,要么放在 PCM 调度器。为了最大限度地提高吞吐量,使用一种请求准备、先来先服务的调度策略对数据请求进行调度[6,7]。

数据迁移。数据到达内存控制器后,如果它们被缓存到(或驱逐出)DRAM 高速缓存,则一个特殊的迁移请求被插入到目标调度程序,目标调度程序将它们写入到目标设备。在这短暂的瞬间状态,飞行中的数据被放置在内存控制器中的迁移缓存中,在这里它们可以以接近于片内缓存的访问速度来访问。对于每个针对 PCM 的请求,内存控制器可以发出多个数据请求,以支持大粒度的迁移。

为了实现在 DRAM 中存储元数据的优点,同时尽量减少元数据的访问延迟,我们提出了在 DRAM 中存储元数据的技术,同时使用一个新的利用小型片上元数据缓存的架构来减少对 DRAM 元数据的访问。

3 细粒度的 DRAM 缓存架构

通过文献[5]了解到元数据可以与它们对应的数据存储在 DRAM 的同一缓存行中,减少了由于访问 DRAM 中两个缓存行带来的访问延时(元数据放在一行,数据本身放在另一行,访问时需要访问 DRAM 的两行,如果元数据和数据本身放在一个缓存行中,只需要访问一次)。基于这一观察,为进一步减少元数据查询等待时间,我们在较小的片上缓存中缓存最近在 DRAM 中访问过的缓存行的元数据。关键在于,所需数据(具有时间或空间的局部性的数据)的元数据可能会被缓存在芯片上,在那里它们能以与 SRAM 标签存储器相同的访问延时被访问。

3.1 数据与元数据存储

图 2 示出了数据和元数据如何被布置在同一行中:每行使用一个缓存块(简称为元数据块)来存储此行中剩下的缓存块的元数据。在存储器访问上,请求的行索引用于检索元数据块。请求的缓存块地址用于索引元数据块来检索请求对应的元数据。我们称这种元数据组织为 tags-in-row of DRAM (TIRD),而传统的方法是在 DRAM 中将存储空间分成区,数据和元数据分别存放在不同的区域。图 3 是 TIRD 方法与传统方法进行的比较,从图中可以看出,TIRD 方法通过在元数据查找之后访问数据,减少了存储访问时间,通过缓存行的命中使延时更短。

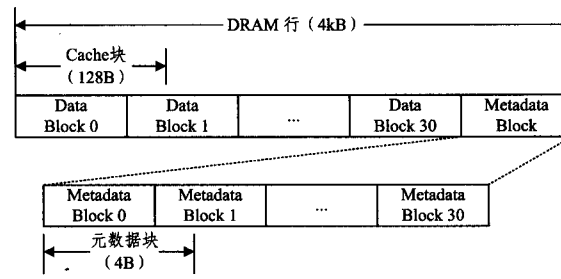


图 2 数据与元数据存储在 DRAM 同一行

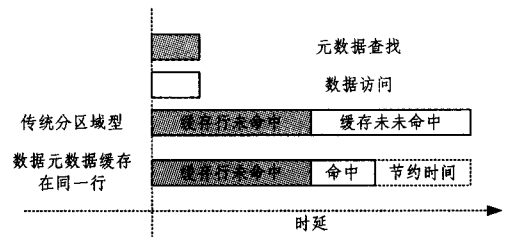
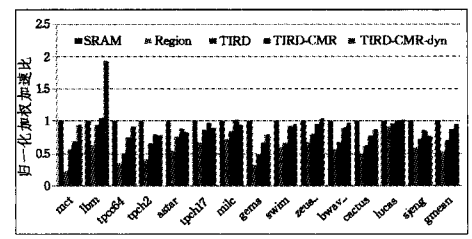
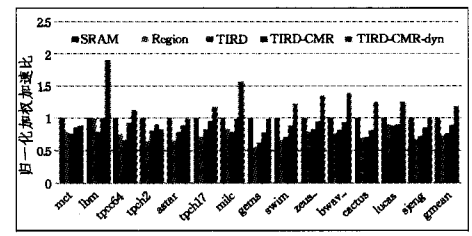


图 3 TIRD 与传统元数据存储比较



(a) 系统性能比较



(b) 系统能效比较

图 4

图 4(a)显示了各种 DRAM 高速缓存管理技术的性能,这些性能指标将贯穿整篇文章。将全部的 DRAM 用来缓存存储在片上 SRAM 中的元数据方案进行比较,一个是简单的元数据存储方案,即元数据被存储在连续的 DRAM 区域(专门存放元数据的单独区域),一个是在前面讨论了的 TIRD 优

化方案，两种方案都值得我们注意。

(1)简单、基于区域化的存储器内在标签存储机制增加了平均29%的内存访问延迟，并且需要额外的元数据的访问，相比SRAM标签存储，性能下降了48%。

(2)然而，在整个存储区域，TIRD方法提高了平均内存访问延迟19%。由于存在片外标签的查找，TIRD方案和SRAM标签存储方案之间仍然有30%的性能差距。

3.2 最近访问元数据块缓冲区

在SRAM元数据存储不要求大的硬件开销的情况下，为了帮助解决TIRD方案的性能不足，我们建议针对在缓存中最近被访问的行的元数据进行缓存，称为TIRD-CMR。TIRD-CMR背后的关键见解是在一个小的缓存中为具有较好局部性的数据缓存元数据(即标签)，缓存的元数据允许大多数的元数据访问都能以类似于SRAM的访问延时获得服务，并且只需要较低的存储开销。TIRD-CMR作为一个高速缓存进行组织，其中，条目使用DRAM行索引进行标记，数据负载中包含某个特定行的元数据块，如图5所示。

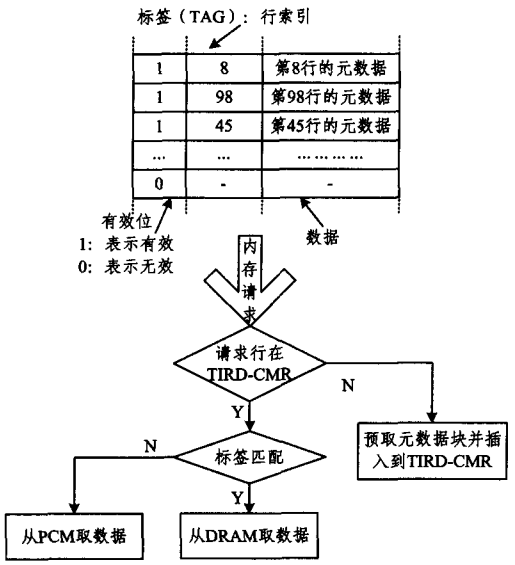


图5 TIRD-CMR组织结构及元数据查找

图5还显示了在TIRD-CMR下如何查找元数据：某个到达的内存请求的地址被用来确定其访问TIRD-CMR的行索引。如果行索引不匹配TIRD-CMR标签条目，则该行的元数据必须从DRAM获取，并将获取的数据插入到TIRD-CMR。当行索引匹配TIRD-CMR标记条目时，将TIRD-CMR条目负载中的标记数据进一步与请求块的地址标签进行比较。若标签匹配，则缓存块位于DRAM中；若标记不匹配，则缓存块位于PCM中。

3.3 实现和硬件

对于一个具有44位物理地址的系统，每个TIRD-CMR条目需要4B用于存储TIRD-CMR标记，128B用于存储元数据，1比特位用于全部针对64个条目大约8kB负载的校验。相比较而言，对于一个512MB的DRAM缓存，SRAM负载的总数要求能存储DRAM所有数据的元数据是8MB。

4 系统测试

我们开发了一个周期级内存模拟器作为一个内部x86多核模拟器的一部分，其前端是基于Pin，并通过突出的侧面执行基准测试中有代表性的部分。表1显示了用于我们研究的主要系统参数。我们的系统循环运行了100万个周期，并收集了100万个周期的结果。

表1 系统模拟参数表[1]

处理器	8核，4GHz
L1 Cache	每核私有32kB，4通道，128B块
L2 Cache	每核共享512kB，8通道，128B块
内存控制器	128/128条读/写请求队列128条迁移缓存，FR-FCFS调度器
TIRD-CMR	64-entry，direct-mapped
内存	2控制器(DRAM and PCM)，64-bit channel 1 rank with 8 banks；512MB direct-mapped，writeback，no-write-allocate DRAM cache；Block-level writeback of dirty data to PCM；Open-row policy
时间	DRAM：缓存行命中(缺失)=40(80) ns；PCM：缓存行命中(clean conflict/dirty conflict)=40(128/368) ns
功耗	Both：缓存行读(写)=0.93(1.02) pJ/bit；DRAM：读(写)：1.17(0.39) pJ/bit；PCM：读(写)=2.47(16.82) pJ/bit

对于我们的8核工作负载，在每个内核上运行一个SPEC CPU或TPC-C/H基准测试实例的单线程，这样每负载总共有8个基准测试实例，代表了许多大型CMPs系统综合的工作负载。表2描述了我们的工作负载的特征，我们的工作负载运行在一个8核的所有SRAM用于存储元数据的存储系统上，这些元数据根据最后一级缓存每一千条指令的缺失率、引用的数据量进行存储。

表2 综合多核工作负载特征

工作负载	MPKI	尺寸(MB)	平均动态迁移粒度(B)
mcf	65.18	2481	128
lbm	64.68	1703.1	700
astar	21.55	2171.6	128
tpcc64	19.44	1150.6	169
tpch2	17.97	1501.5	128
tpch17	11.72	1696.4	178
milc	11.04	1273.6	128
gems	8.84	1819.9	160
swim	8.29	1072.3	376
zeusmp	8.68	1176	361
bwaves	7.42	1506	128
cactus	4.02	783.7	881
lucas	2.15	1002.2	417
sjeng	0.85	893.9	128

5 评价

5.1 性能评价

图4(a)显示了各种使用加权加速度量[8]DRAM缓存管理技术的性能。图中对基准测试的加速度总和进行了比较，即在同一个系统上，各种元数据和数据存储在同一DRAM中的方案和单独使用SRAM来存储元数据的实方案进行了比较。

一个64条目的TIRD-CMR相比TIRD系统，性能额外地提高了22%，性能的提升主要是由于TIRD-CMR系统的能力提升，如果某行的元数据被缓存在TIRD-CMR中，则TIRD-CMR正好能以SRAM标签存储的速度访问相同的行。我们发现，62%的访问在TIRD-CMR中命中(当数据位于DRAM时是34%的命中率，当数据位于PCM时是28%的命中率)，38%的访问在TIRD-CMR中未命中，此时首先需要从DRAM访问元数据。

5.2 能效评价

图4(b)比较了不同技术动态主存储器的能源效率(每瓦

特的性能)。要注意的是,我们没有考虑内存控制器的能源消耗,因此,没有对具有较大 SRAM 存储的技术进行修正。然而分区系统和 TIRD 系统相比,使用全 SRAM 进行元数据存储的系统的能效下降大约 25%,这是由于越来越多的 DRAM 查找操作,TIRD-CMR 能够从它小的片上高速缓存来为许多那样的查找操作服务,全 SRAM 系统可以实现 11%以内的节能。

5.3 标记缓冲区大小

图 6(a)显示了性能的敏感性和 TIRD-CMR 大小对应的 TIRD-CMR 缺失率。TIRD-CMR 的性能随着 TIRD-CMR 大小的增长而增长,尽管伴随着边际收益递减,但是随着元数据缓存在 TIRD-CMR 中占据 DRAM 缓存的较大比例,TIRD-CMR 的缺失率降低。

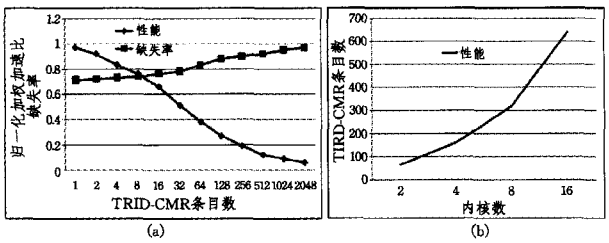


图 6 系统特征与系统性能

5.4 内核数量

图 6(b)显示了我们的 TIRD-CMR 技术在不同内核数量上如何扩展。在这项研究中,我们扩展了内核数量,保持 DRAM 的数量与内核成正比,并且计算了要达到大约全 SRAM 元数据系统性能的 6%所需要的 TIRD-CMR 条目数量。TIRD-CMR 的大小需要根据内核的数量成比例地扩展,但是 TIRD-CMR 系统的绝对存储负载仍然比一个 SRAM 标记存储系统小 3 个数量级。

5.5 细粒度的潜在效益

基于运行时的特征,以一个细粒度的方式管理 DRAM 缓存为从 PCM 到 DRAM 迁移不同数量的数据提供了机会。例如,具有高数据重用的应用程序可能会因为缓存更多的数据提高 DRAM 高速缓存的命中率而受益。在基线 SRAM 系统中,我们发现,每个迁移简单地缓存 4kB 的数据可以提高 20%的 DRAM 高速缓存的命中率,但导致性能下降了 75%,这是由于 DRAM 和 PCM 通道的带宽消耗分别增加了 55%和 140%。为了在数据的局部性和带宽消耗上取得折衷平衡,我们开发了一个简单的策略,即动态调整迁移粒度。

我们采用的机制的灵感来自 Qureshi 等人[9],其中,某些称为"leader"的缓存组遵循固定的替换策略,其余的缓存组(称为"follower")遵循在"leader"缓存组中导致最低缓存缺失率的替换策略。把主存分成 256 个行缓存组,其中 7 个为"leader"类型的行缓存组,剩下的 249 个为"follower"类型的行缓存组,7 个"leader"类型的行组采用一个固定的迁移粒度——128B,256B,512B,1kB,2kB,4kB 和不迁移。在每 10 万次循环结束时,对每个线程的,每个"leader"缓存组进行计算:(1)平均内存访问延迟(也计算包括迁移缓冲区的访问),每个未完成的请求使用延迟计数器,其总和除以请求数;(2)高速缓存块迁移的数量。

我们为下一个循环周期确定的一个线程的迁移粒度为具有最小平均请求访问延迟和最少缓存块迁移数量的"leader"类型缓存组的迁移粒度。其中最关键的思路是,"follower"缓存组应当模仿具有低访问延时和较少数据迁移的"leader"缓存组来改善系统性能。表 2 给出了我们所用策略下测量工作负载的平均粒度。

图 4(a)显示了采用我们的动态迁移粒度技术(TIRD-CMR-Dyn)的一个 64 条目 TIRD-CMR 系统的性能。相关观点如下:首先,发现采用我们的动态迁移粒度技术的 TIRD-CMR 系统可以提高 SRAM 元数据存储系统的性能 6%,这是由于采用动态迁移粒度技术的 TIRD-CMR 系统可以根据其工作负载的特性调整它的迁移粒度,某些情况下在整个测试周期内还可以不涉及数据迁移。这既改善了 DRAM 高速缓存的效率,也改善了 TIRD-CMR 的效率。我们发现,45%的 TIRD-CMR 数据访问直接在 DRAM 缓存中命中,45%的 TIRD-CMR 数据访问在 PCM 后备存储中命中,而 43%的访问缺失率在 TIRD-CMR 中,并且必须从 DRAM 中访问元数据。其次,正如图 4(b)显示,更保守地迁移提高了 DRAM 缓存的利用率,减少了带宽争用,导致能效在 SRAM 标记系统基准上提高了 18%。

虽然我们使用的 TIRD-CMR 作为一个基板能够有效地实现我们的动态迁移粒度技术,但是其它的元数据存储技术可以与我们的动态迁移粒度机制相结合。另外,这是一个早期设计的动态粒度机制,我们正在努力研究提高其效率的方法。

6 相关工作

以前的方法为减少元数据的存储开销已经使用大的高速缓存行按照千字节的顺序[3,10]管理 DRAM 缓存,或者将存在于一个大缓存块的较小的扇区作为一个位向量来管理缓存,消除了需要存储完整的标签元数据[11,12]。

这些技术仍然为所有的 DRAM 存储元数据不断增加带宽消耗,并污染了 DRAM 高速缓存,使伪共享的概率增加,并限制可扩展性。通过观察,CAT 减少了标签存储的负载,具有空间局部性的缓存块共享同样的高阶标记位,这样一个标记可以代表多个高速缓存块[13];其缓存对部分标签需要一个关联的搜索,并且缓存单元在局部被剔除的标签上必须被标记为无效。虽然 TIRD-CMR 也满足了空间局部性的访问(以行的粒度),但我们采取截然不同的方法,即在 DRAM 中存储标签,并在一个小缓冲区中缓存完整的标签信息。

同时,Loh 和 Hill 也提出了将元数据与元数据对应的数据存储在相同的行,并在一个片上的高速缓存中开发此技术,片上的高速缓存存储一个位图表示最近访问的区域(位图的位数决定了最近访问的区域在 DRAM 中固定的数量)是否存在 DRAM 中。当一个区域从元数据缓存中被淘汰时,元数据对应的数据也必须从 DRAM 中清除,所以一个大元数据缓存需要减少这类数据被淘汰的删除操作(对于 512MB 的 DRAM 缓存,作者使用了 2MB 的 SRAM 空间)[5]。他们和我们的工作相比,关键的区别是我们针对 DRAM 中一小部分的行缓存完整的元数据,而且从 TIRD-CMR 剔除的那些条目在 DRAM 缓存中既保留元数据,也保留数据。这种方式提高了 DRAM 高速缓存的利用率,避免了针对 TIRD-CMR 命中的元数据查找,减少了高成本的写回 PCM 的流量。

(下转第 108 页)

Replication for Intermittently Connected Heterogeneous Wireless Networks[C]//Proceedings of IEEE WoWMoM Workshop on AOC. 2007

[9] Balasubramanian A, et al. DTN Routing as a Resource Allocation Problem[C]// ACM SIGCOMM Conference on Computer Communications. 2007:373-384

[10] Nicolo D. Performance limits of real delay tolerant networks[C]// 5th Annual Conference on Wireless on Demand Network Systems and Services. 2008:149-155

[11] Scott K, Burleigh S. Bundle Protocol Specification [EB/OL]. IETF RFC 5050. November 2007

[12] Ramadas M, et al. Licklider Transmission Protocol Specification [EB/OL]. IETF RFC 5326. September 2008

[13] Balasubramanian A, Levine B N, et al. DTN Routing as a Resource Allocation Problem[C]// ACM SIGCOMM Conference on Computer Communications. 2007:373-384

[14] Padma H, Mundur H, Sookyoung H, et al. Routing in intermittent network topologies[C]// International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems. 2006:385-389

[15] 刘蕴络,胡佳慧,冯艳娟,等. 一种基于水声 DTN 网络的 QoS 路由算法[J]. 计算机科学,2011,38(11):37-39

[16] 卓翠敏,李鲁群. 面向 DTN 无线传感网移动节点的动态资源调度模型的研究 [J]. 计算机科学,2011,38(10):356-358

[17] 樊秀梅,单志广,张宝贤,等. 容迟网络体系结构及其关键技术研究[J]. 电子学报,2008,36(1):161-170

[18] 熊永平,孙利民,牛建伟,等. 机会网络[J]. 软件学报,2009,20(1):124-137

[19] 薛静锋,陆慧梅,石琳. 基于概率延迟的 DTN 路由算法的设计[J]. 北京理工大学学报,2008,28(8):687-691

[20] 徐昌彪,王宇,祁彦. DTN 中基于服务等级的 Push-Pull 拥塞控制研究[J]. 计算机应用研究,2010,27(10)

(上接第 82 页)

之前传统的缓存中确定数据预取大小(例如,文献[14,15]中所述)的方法在两个关键方面不同于我们的动态策略:(1)他们考虑不同的缓存粒度的收益方面,而我们也从数据移动争用上考虑了开销成本方面,在一个混合存储系统中,数据移动争用上的开销是显著影响系统性能的一个因素。(2)他们使用大型结构跟踪重用信息,使用的大型结构的大小必须与 cache 大小规模相当,然而我们的技术仅仅只需要其规模大小与未完成的内存请求数量相当的结构,远小于 DRAM 缓存的大小。

结束语 我们介绍了一个有效的体系结构来管理一个大型 DRAM 缓存。通过观察,元数据可以和它们对应的数据存储在同一缓存行中,我们设计了一个新的架构来缓存最近被使用的元数据,该架构为具有时间和空间局部性的数据访问提供了大型 SRAM 元数据存储。建立在技术上,我们还探讨了为混合存储设计的新的缓存策略,该策略确定能够导致低访问延迟和很少数据迁移操作的数据迁移粒度。我们的结果表明,在较低的存储负载下(8kB—8MB),对于大型的 SRAM 元数据存储,我们的技术和缓存策略可以实现类似的好处。即使没有考虑大型 SRAM 元数据存储系统的能源开销,由于更少的数据迁移操作,也可以节约 18%的能效。

参 考 文 献

[1] Lee B C, Ipek E, Mutlu O, et al. Architecting phase change memory as a scalable DRAM alternative[C]// Proceedings of ISCA '09. 2009:171-182

[2] Qureshi M K, Srinivasan V, Rivers J A. Scalable high performance main memory system using phase-change memory techno-logy[C]// Proceedings of ISCA'09. 2009:101-112

[3] Dong X, Xie Y, Muralimanohar N, et al. Simple but effective heterogeneous main memory with on-chip memory controller support[C]// Proceedings of SC'10. 2010:773-778

[4] Zhao L, Iyer R, Illikkal R, et al. Exploring DRAM cache architectures for CMP server platforms[C]// Proceedings of ICCD'07. 2007:253-259

[5] Loh G, Hill M D. Efficiently enabling conventional block sizes for very large die-stacked DRAM caches[C]// Proceedings of MICRO'11. 2011:123-128

[6] Rixner S, Dally W J, Kapasi U J, et al. Memory access scheduling[C]// Proceedings of ISCA'00. 2000:58-69

[7] Zuravleff W K, Robinson T. Controller for a synchronous DRAM that maximizes throughput by allowing memory requests and commands to be issued out of order[P]. U. S. patent 5630096. 1997

[8] Eyerman S, Eeckhout L. System-level performance metrics for multiprogram workloads[C]// Proceedings of MICRO' 08. 2008:147-158

[9] Qureshi M K, Lynch D N, Mutlu O, et al. A case for MLP-aware cache replacement[C]// Proceedings of ISCA'06. 2006:211-221

[10] Jiang X, Madan N, Zhao L, et al. CHOP: Adaptive filter-based DRAM caching for CMP server platforms[C]// Proceedings of HPCA'10. 2010:165-175

[11] Liptay J. Structural aspects of the System/360 Model 85, II: The cache[J]. IBM Syst. J., 1968, 7(1):15-21

[12] Seznec A. Decoupled sectored caches: conciliating low tag implementation cost and low miss ratio[C]// Proceedings of ISCA'94. 1994:19-28

[13] Wang H, Sun T, Yang Q. CAT-caching address tags-a technique for reducing area cost of on-chip caches[C]// Proceedings of ISCA'95. 1995:188-196

[14] Inoue K, Kai K, Murakami K. Dynamically variable linesize cache exploiting high on-chip memory bandwidth of merged DRAM/logic LSIs[C]// Proceedings of HPCA'99. 1999:119-128

[15] Johnson T L, Hwu W-M W. Run-TIRDe adaptive cache hierarchy management via reference analysis[C]// Proceedings of ISCA'97. 1997:134-145

[16] Mandelman J A, Dennard R H, Bronner G B, et al. Challenges and future directions for the scaling of dynamic random-access memory (DRAM)[J]. IBM J. Res. Dev., 2002, 46(2/3):187-212