文章编号:1007－6735(2019)01－0022－08 DOI: 10.13255/j.cnki.jusst.2019.01.004

异构内存系统全局优化的数据预取算法

裴颂文[1,2]， 赵梦涛[1]， 姬燕飞[1]
（1. 上海理工大学 光电信息与计算机工程学院，上海 200093；2. 复旦大学 管理学院，上海 200433）

摘要： 鉴于现有的数据预取算法不能满足高效能异构计算系统对动态随机存取存储器(DRAM)和非易失性存储器(NVM)相结合的新型异构存储器高效访问的要求，提出了一种模拟退火的全局优化数据预取算法(SADPA)。该算法在启发式搜索模拟退火算法的基础上，引入了随机因子，以避免局部最优，从而确定了全局优化阈值以预取 NVM 页面的有效数量。实验结果表明，该算法相对于静态阈值调整算法，平均访问延时降低了 4%，每个时钟周期内的平均指令数(IPC)增加了 10.1%；对于 cactusADM 应用，该算法相对于软硬件协同的动态阈值调整算法，系统能耗降低了 3.4%。

关键词： 异构内存系统；数据预取；模拟退火算法；全局优化
中图分类号： F 830 **文献标志码：** A

Data Prefetching Algorithm for Globally Optimizing Heterogeneous Memory System

PEI Songwen[1,2], ZHAO Mengyi[1], JI Yanfei[1]
(1. *School of Optical−Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai 200093,China*; 2. *School of Management, Fudan University, Shanghai 200433, China*)

Abstract: Due to the existing data prefetching algorithms can't meet the requirements of the novel heterogeneous memory system combining the dynamic random access memory (DRAM) with the non-volatile memory (NVM) in high energy-efficiency heterogeneous computing systems, a simulated annealing data prefetching algorithm (SADPA) was proposed. It was a heuristic search inspired simulated annealing algorithm, in which a random factor was introduced to confirm the global optimal threshold and the valid number of prefetching NVM pages. The results show that the average accessing latency of SADPA is 4% lower than that of the static threshold adjustment algorithm, and the average instruction per cycle (IPC) of the SADPA is 10.1% greater than that of the static threshold adjustment algorithm. Besides, the systemic power supported by SADPA, as for the cactusADM, is reduced by 3.4% compared with the cooperative hardware/software dynamic threshold adjustment algorithm.

Keywords: *heterogeneous memory system*; *data prefetching*; *simulated annealing algorithm*; *global optimum*

收稿日期: 2018−01−27
基金项目: 中国博士后科学基金资助项目(2017M610230)；国家自然科学基金资助项目(61775139，61332009)；上海市自然科学基金资助项目(15ZR1428600)；上海市浦江人才计划项目(PJ1407600)
第一作者: 裴颂文(1981−)，男，副教授．研究方向：异构计算机系统结构、多媒体大数据分析、分布式计算．E-mail：swpei@usst.edu.cn

处理器性能的提升对访存带宽提出了新的要求，访存延时已经成为影响处理器性能的重要瓶颈。处理器需要花费大量时间等待数据结果返回，这造成了性能的损失。数据预取正是降低或隐藏访存延时的重要手段[1]，它有效地缓解了处理器与动态随机存取存储器(dynamic random access memory，DRAM)之间存在的访存速度差异。

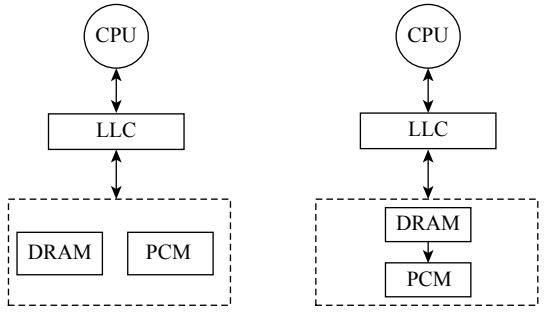
许多新兴的存储器架构利用不同存储器技术，以不同方式将 3D 堆叠存储器集成到处理器存储器中，结合了多种处理的性能和功率特性，构成异构内存架构。异构内存架构的研究主要集中在优化快速存储器作为慢速存储器的缓存，或者将快速存储器和慢速存储器放置为平级结构两个方面。设计异构内存架构的目标是既可以获得快速存储器速度上的性能优势，又拥有慢速存储器巨大的内存容量。处理器的性能以每年 60% 的速度提升，而以 DRAM 作为内存的处理器每年性能提升速度只有 10%[2]，DRAM 的性能制约了处理器性能的提升。现今已有一些存储器可以满足这些需求，包括 3D 堆叠 DRAM(3D-DRAM)以及非易失性存储器(non-volatile memory，NVM)。新型 NVM 的出现，为扩展计算机内存提供了新的途径，同时推动了传统计算机系统结构的改变。相比现存的 DRAM，新兴的 NVM 能耗更低，并且具有良好的扩展性。现有的 NVM 包括 FeRAM(铁电随机存储器)，PCM(相变存储器)，STT-RAM(自旋转移矩磁随机存储器)，MRAM(磁性随机存储器)，RRAM(阻变随机存储器)等[3]。其中 PCM 应用范围广，性能接近 DRAM，但容量远远大于 DRAM，能耗和价格远远低于 DRAM，这为新型的内存体系结构提供了良好的硬件保障；然而它访问延时较长[3]，单元存取能量消耗较高。以上为新型存储器和传统存储器结合构成新型存储结构提供了理论基础。它在大幅提升内存容量、降低成本的同时，又可以提供和 DRAM 相近的访问速度。因此，本文主要研究混合式存储系统的数据预取方法，这对于缓解存储墙问题具有重要意义。对于混合内存的数据预取有多种方法，有静态阈值的预取算法、动态阈值的预取算法。以上算法不能达到性能的最优化，因此，本文提出了一种新的数据预取算法。

1　研究背景

1.1　异构混合内存结构

Park 等[4]提出了 PCM 和 DRAM 同级的异构内存结构，DRAM 和 PCM 都用作主存储器。这种异构内存架构减少了功率的预算，它与传统的主存储器架构十分类似，在混合主内存和二级存储器之间有内存页面交换。在这种异构内存架构中，DRAM 和 PCM 被分配了统一的物理地址空间。Linux 内核可以直接管理 PCM 和 DRAM 的页面，这种情况下，在内核中对 DRAM 和 PCM 进行分区很有必要。结构如图 1(a)所示，它充分结合了 PCM 和 DRAM 的优点，采用线性统一编址[5]。

Qureshi 等[6]提出了另一种异构内存结构，考虑到 DRAM 快速的访存速度以及 PCM 的大容量，把 DRAM 作为 PCM 的缓存[7]。这种混合存储内存架构利用了 DRAM 在访问速度上的优势。PCM 上可以存储大量的信息，进而减少了对外存的访问。此种结构中 DRAM 采用 3D 芯片堆叠技术[8]，PCM 由操作系统管理，DRAM 是透明的，DRAM 发出的访问请求都到达 PCM 中。在 DRAM 上层有一个控制器，在请求到达时，这种混合结构首先在 DRAM 中查找目标数据，当查找失败时，再对 PCM 进行查找[3]。如图 1(b)所示，这种层次式的结构克服了 DRAM 容量小及 PCM 速度慢的缺点。



(a) 水平型异构内存系统结构　(b) 垂直型异构内存系统结构

图 1　异构内存系统结构

Fig.1　Structure of the heterogeneous memory system

1.2　数据预取方法介绍

鉴于数据访问的局部性原理，数据预取技术可以利用它提前预测访存地址并提前请求访存，从而达到降低或隐藏访存延时的效果[9]。数据预取(data prefetching)主要分为 3 种：

a. 基于硬件控制的数据预取(hardware

prefetching)，基于动态的分支预测，可以动态发射数据预取指令，没有额外的预取指令开销，并且提高了程序性能；

b. 基于软件控制的数据预取(software prefetching)，编译器编译或者程序员手动插入预取指令，从而预先得到数据，隐藏了延时；

c. 基于软硬件结合的混合数据预取方式(mixed data prefetching)，这种数据预取结合了上述两种预取方式，由编译器初始化标签存储器，由硬件直接从存储器中预取数据，避免了软件预取的开销，实现了性能的优化[10]。

Ramos等[11]提出了RaPP(rank-based page placement)内存管理机制，把访问级别低的页面放置在PCM中，级别高的页面放置在DRAM中，使用多级队列的方法存储页面的最近存储及访问信息。该方法使用计数器统计每个页面的读写访问次数，每访问一次计数器增加1。当页面的访问次数达到一定阈值时，把该页面提升到更高级的队列中，此时两个队列中就发生页迁移。但是，这种方法有很大的局限性，它只考虑到了页面的局部访存信息。张进宝[12]提出了基于页面热度的异构内存能效管理机制，它考虑了全局和部分的读写信息，在RaPP的基础上得到了更为理想的划分冷热页面的方法。根据页面的冷热度值，可以有效区分冷热页面，并且利用建模的方法筛选出耗能最少的页面组合，建立了异构内存上的能效迁移模型。Islam等[3]提出了平地址内存结构中的数据预取策略，这种预取策略使用了距离预取，其好处在于可以检测重复序列中的非单位步幅；使用全局历史缓冲区(GHB)结构来实现距离预取器，这种方法开销小，预取准确性更高，访存冲突较少。Yoon等[13]提出了把DRAM作为PCM缓存情况下的异构内存系统管理策略，这种策略记录行缓冲区缺失数目，并且把缺失数目多的数据放在DRAM中，避免过大的PCM访问开销。Liu等[14]提出了软硬件协同管理(HSCC)的异构内存系统，这种结构简化了硬件设计，并且在软件层提供了缓存管理技术。软硬件协同管理系统利用扩展的页表、快表缓冲(TLB)记录NVM和DRAM之间的地址映射，并且还记录了访问NVM页面的次数。HSCC系统中采用了爬山算法决定迁移到DRAM中的页面数。

2 模拟退火的全局优化数据预取算法

2.1 数据预取方法

数据预取就是一种求解最优可迁移页面数量的机制，模拟退火算法是一种全局的求解最优化问题的算法，适用于该问题的求解。这个算法的思想借鉴于固体的退火原理，固体的内能随着温度的升高而增大，粒子的运动就越接近于无序[15]，随着温度的降低，运动就趋向于有序的状态。最终，当温度恒定时粒子达到最稳定的状态。爬山算法也是启发式算法，文献[16]采用了爬山算法改进了深度优先搜索，但容易陷入局部择优的情况，模拟退火算法可以克服爬山算法的缺点。

本文采用DRAM作为PCM缓存的层次化异构内存系统结构，异构内存系统已经被证明了它的有效性与实用性。在此结构中，本文预取算法的思想是通过确定合理的预取阈值，寻找全局优化的页面迁移数量。基于模拟退火的数据预取算法可以高效地调整预取阈值，从而避免陷入局部最优，避免了资源的浪费和非必要的预取。本文提出了一种基于模拟退火的全局优化数据预取算法(simulated annealing data prefetching algorithm, SADPA)。经实验证明，这种启发式的算法可以按照一定的概率收敛于全局最优解。根据页面访问次数，当次数达到阈值时，就可以将页面从NVM预取到DRAM，可以避免不必要的页面预取。

在异构内存系统中，选择合适的时机把NVM预取到DRAM中是关键的问题，以往通常是在一个固定的时钟周期内统计预取带来的性能增益，触发相应的预取。在生存周期 T 内，性能增益 $gain$ 的计算为[14]

$$gain_t = \left(time_{\mathrm{rp}} - time_{\mathrm{rd}}\right) M_{\mathrm{rd}} + \left(time_{\mathrm{wp}} - time_{\mathrm{wd}}\right) M_{\mathrm{wd}} - M_{\mathrm{p}} time_{\mathrm{p}} \quad (1)$$

式中：$time_{\mathrm{rd}}$ 为DRAM平均读时延；$time_{\mathrm{rp}}$ 为PCM平均读时延；M_{rd} 为缓存的读次数；$time_{\mathrm{wp}}$ 为PCM平均写时延；$time_{\mathrm{wd}}$ 代表DRAM平均写时延；M_{wd} 代表缓存的写次数；M_{p} 代表预取次数；$time_{\mathrm{p}}$ 代表缓存DRAM消耗的时间。初始化预取阈值为 $threshold_{t0}$，前 T 个和现 T 个时钟周期的预取阈值分别为 $threshold_{t-1}$，$threshold_t$，对应的性能增益分别为 $gain_{t-1}$，$gain_t$。性能增益差 $delta_{\mathrm{g}}$ 的计算式为

$$delta_{\mathrm{g}} = gain_t - gain_{t-1} \quad (2)$$

2.2　算法流程

图 2 展示了 SADPA 流程，其中 *stride* 为步长。数据预取的关键在于选择合适的时机预取到需要的数据。SADPA 本质上是一种贪心算法，贪心算法是在旧解附近寻找新解，这个解很可能是局部最优解，而不是全局的。本文在贪心算法的基础上引入随机因子，可以随机对多个点进行搜索，以一定的概率来接受一个比当前解要差的解。这种方式在一定程度上可以防止陷入局部最优解，实现全局优化。

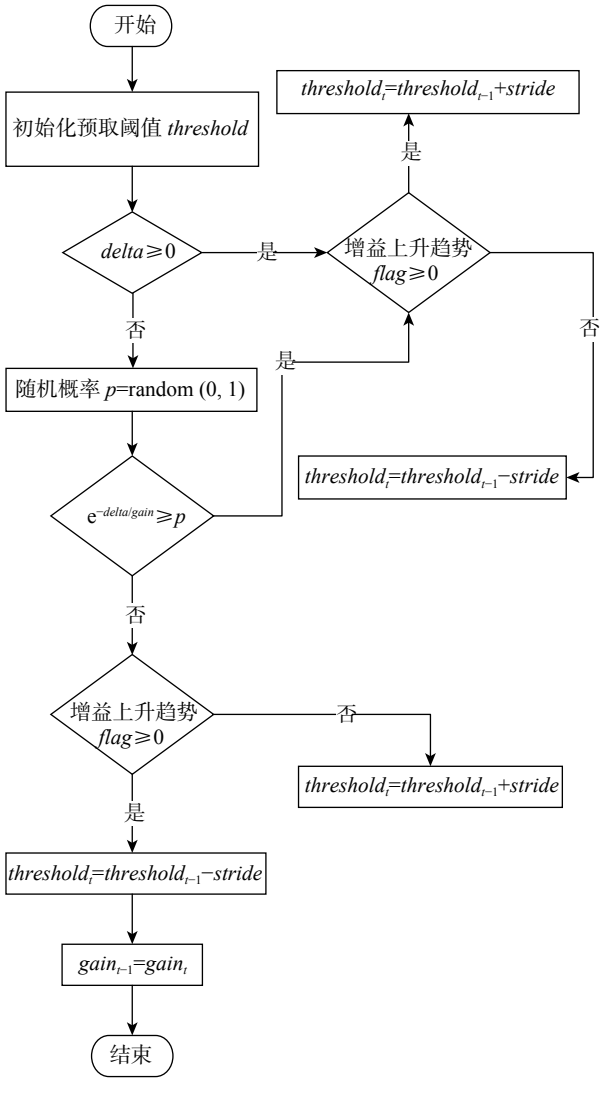


图 2　SADPA 流程图

Fig.2　Flow chart of SADPA

在 SADPA 中，每个周期内在净收益增加的方向上动态调整预取阈值，使得最终阈值收敛在净收益最大的点。本文设置阈值初始值为 7，每次减小的步幅长度为 0~2 之间的随机整数，阈值在递减的时候要保证始终大于 0。若当前增益大于之前的增益，说明选取的策略是正确的。当增益趋势为上升趋势时，阈值增加；反之阈值减小。若当前增益小于之前的增益，引入随机因子 α。α 和净增益相关，若 α 满足一定的概率，则执行和当前增益大于之前增益时的阈值调整方法，否则执行相反的阈值调整。

2.3　算法伪代码

SADPA 具体算法如下：

输入：阈值 *threshold*，当前增益 $gain_t$，先前的增益 $gain_{t-1}$，增益增减趋势标志 *flag*，可变步幅 *stride*

输出：int

```
0. /*初始化增益差值 delta 为 0，stride 取整数*/
1. delta = 0，flag = 0，stride = int（random（0，2）），threshold = 7
2. repeat
3. delta = gain_t - gain_t-1
4. if delta >= 0
5. /* 增益呈现递减趋势并且阈值大于 0 */
6. if flag < && threshold > 0
7. threshold = threshold - stride
8. else if flag > 0
9. threshold = threshold + stride
11. /* 引入随机因子，跳出局部最优解 */
10. if delta <= 0
11. /* 引入随机因子，跳出局部最优解 */
12. α = exp（-delta/gaint）
13. if α > random（0，1）
14. /* 增益呈现递减趋势并且阈值大于 0 */
15. if flag < 0 && threshold > 0
16. threshold = threshold - stride
17. else if flag > 0
19. else
20. /* 增益呈现增长趋势并且阈值大于 0 */
21. if flag > 0 && threshold > 0
22. threshold = threshold + stride
23. else if flag > 0
24. threshold = threshold - stride
25. gain_t-1 = gain_t
26. end for
```

3　实验与评估

3.1　实验平台

采用 Zsim 模拟器集成 NVMain 构建实验评估系统。Zsim 是业内广泛采用的时钟精确的系统模拟器，可以用来模拟大型异构计算系统和内存存储层次系统[17]。NVMain 可以在内存结构层面模拟新型非易失性存储器，也可以模拟 DRAM 存储器[18]。Ubuntu 14.04 的实验评估系统的硬件环境包括：处理器 Intel Core i5-6400；一级缓存 64 kB，采用

4路组相联；二级缓存256 kB，采用8路组相联；三级缓存8 MB，采用16路组相联。存储结构为DRAM和PCM混合（DRAM容量为1 GB，PCM容量为32 GB），其中DRAM和PCM的页大小为4 kB。测试实验选取了SPEC CPU 2006基准测试程序集中的astar，bzip2，cactusADM，hmmer，calculix作为预取性能和能耗分析的测试子程序[19]。

3.2 实验数据及结果分析

图3为SADPA内存访问延时分布图，周期间隔为22，最后的(88，110]是为了补全周期间隔，纵坐标为周期对应的有效延时数据的个数。在[0，22]周期内，有效延时访问记录的数量为23。在(88，110]周期内，有效延伸访问记录的数量为11。

本文还进一步对比了其他预取算法下的内存访问延时，并用多个基准测试程序进行评估。图4为平均访问延时分布图。对于astar应用，SADPA与软硬件协同的动态阈值调整算法相比，平均访问延时从27.69 ns降低到25.79 ns，降低了6.8%。因为SADPA采用了全局优化的启发式搜索方法，减少了PCM和DRAM之间的页迁移次数，从而减少了平均访问延时。对于bzip2应用，SADPA与软硬件协同的静态阈值调整算法相比，平均访问延时分布从37.14 ns降低到35.32 ns，降低了4.9%。因为SADPA可以动态调整预取阈值，确定从PCM迁移到DRAM的页面，降低了写操作延时，从而减少了平均访问延时。在不同的测试程序下，SADPA与硬件预取算法相比，平均访问延迟均有大幅降低。因为SADPA是一种基于软硬件协同的预取算法，它预取访问频率超出最优阈值的页面，而硬件预取仅根据历史访问记录信息可能会预取到一些不必要的页面，故与硬件预取算法相比，SADPA可以获得更低的访问延时。

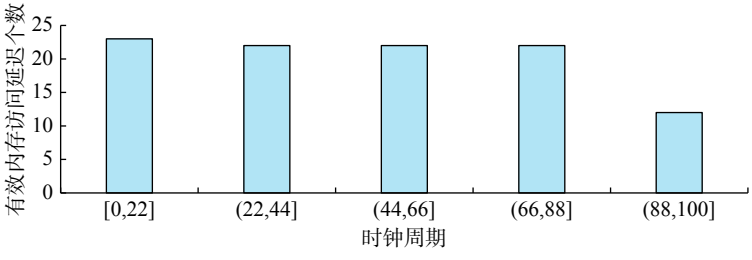


图3 内存访问延时分布图

Fig.3 Latency distribution of memory accessing

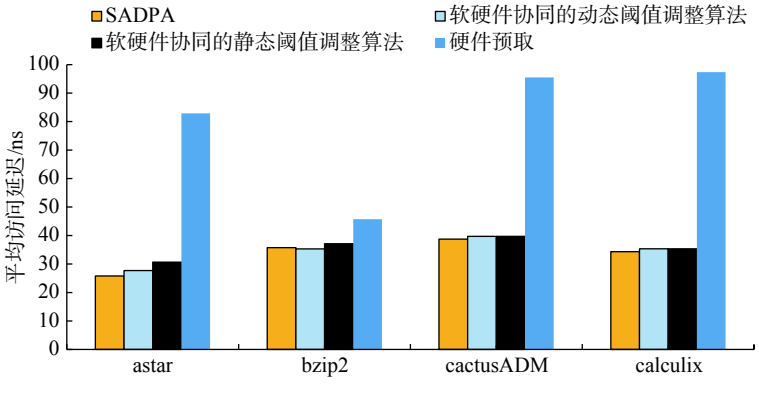


图4 平均访问延时图

Fig.4 Average latency of memory accessing

图5为各种预取算法在不同测试程序下的IPC（instructions per cycle）。SADPA与软硬件协同的动态阈值调整算法[14]相比，平均IPC增加了4.5%。因为SADPA算法是基于全局优化的动态调整阈值，相比基于局部最优的爬山算法的动态阈值调整，具有更优的全局阈值。这种方式可以提高数据页的迁移频率，因此增加了单位时间内执行的指令数，提高了系统平均IPC。SADPA与软硬件协同的静态阈值调整算法[14]相比，系统平均IPC增加了10.1%。因为静态阈值调整算法中，

阈值是固定不变的，这会带来不必要的数据页迁移，从而增加能耗、时间等开销。SADPA 和硬件预取算法相比，IPC 增加了 8.4%。因为 SADPA 算法从软件的角度出发，采用启发式搜索来动态调整阈值，优化了整个预取过程，从而提高了 IPC。

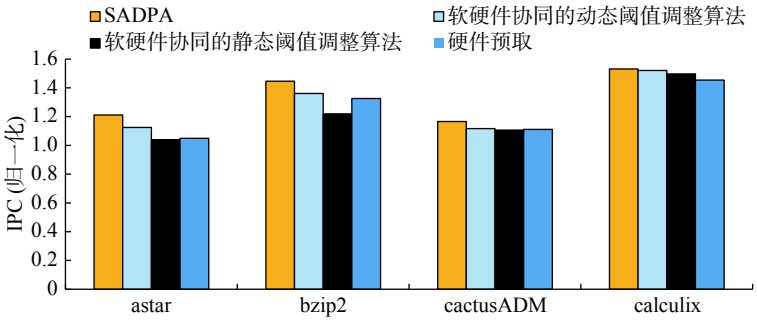


图 5　IPC 图

Fig.5　Normalized IPC

图 6 为各种预取算法在不同测试程序下的能耗分析图。以硬件预取为基准，对其他预取算法下的测试数据进行归一化处理。对于 cactusADM 应用，SADPA 与软硬件协同的动态阈值调整算法相比，能耗降低了 3.4%。因为 cactusADM 中 PCM 写占比较低，并且 DRAM 读占比较高，SADPA 选取了全局优化阈值，避免了不必要的页面预取，从而减少了能耗。对于 astar 应用，SADPA 相对于硬件预取算法能耗降低了 11%。因为 astar 是一种寻路算法，SADPA 算法考虑到全局因素，降低了 PCM 写延时长对性能的影响，因而可以获得更低的能耗。

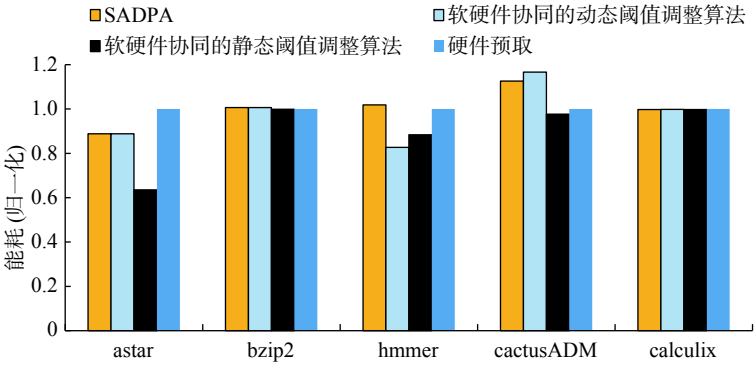


图 6　能量消耗归一化统计图

Fig.6　Normalized power dissipation

图 7 为各种预取算法在不同基准测试程序下的预取率。SADPA 比软硬件协同的动态阈值调整算法平均预取率提高了 2.5%。对于 bzip2 应用，SADPA 算法和软硬件协同的动态阈值调整算法相比，预取率从 89.01% 提升到 93.12%，预取率提高了 4.6%。因为 bzip2 的 DRAM 和 PCM 的写占比低于读占比，基于全局的启发式搜索算法可以动态调整阈值，在一定程度上能跳出局部最优解，寻找出全局优化解。得到全局优化值后，可以合理确定预取的页面，使得预取率得到提升。图中最显著的是对于 cactusADM，和硬件预取算法相比，SDAPA 算法预取率提高了 13%。因为它是一种 PCM 写占比和 DRAM 读占比相差非常大的应用，所以 SADPA 算法提高了预取率。

图 8 为各种算法在不同基准测试程序下的 DRAM cache 利用率分析图。对于 bzip2 应用，SADPA 与软硬件协同的静态阈值调整算法相比，DRAM cache 利用率从 33.15% 提高到 35.02%，提高了 5.3%。因为 SADPA 是基于全局优化的动态阈值调整预取算法，合理的数据页迁移可以避免静态阈值调整算法预取不到访问频度较高的数据页，因而提高了 DRAM cache 利用率。对于 hmmer 应

用，SADPA 算法和软硬件协同的动态阈值调整算法相比，DRAM cache 利用率从 16.35% 提高到 17.84%，提高了 9.1%。因为 SADPA 算法可以避免陷入局部最优的情况，所以提高了利用率。

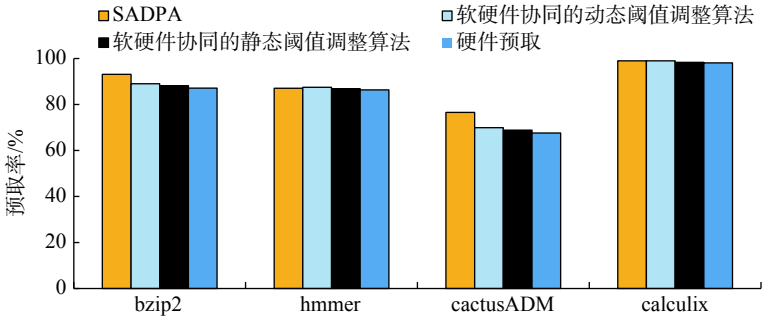


图 7 预取命中率

Fig.7 Ratio of prefetching hit

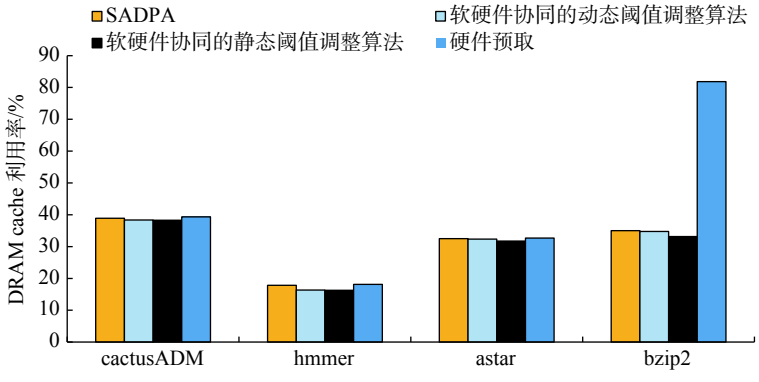


图 8 DRAM cache 利用率

Fig.8 Utilization ratio of DRAM cache

4 总结与展望

本文主要研究 DRAM-PCM 混和存储系统数据预取算法，介绍了异构内存系统的结构及发展，描述了 SADPA 的思想及实现步骤。SADPA 根据页面的访问次数与阈值判断，启发式地预测将 NVM 页面迁移到 DRAM 页面的数量。该算法可以防止陷入局部最优，寻找出全局优化的阈值。相对于局部最优的算法，SADPA 可以获得更高的性能。面向混合式存储系统的高效能数据预取机制是下一步的研究方向。近来随着机器学习的发展，本文考虑结合机器学习方法，进一步实现预取操作的能耗和预取时间之间的最优化设计。

参考文献：

[1] 郭勇, 尉红梅, 漆锋滨. 基于局部性分析数据预取在 GCC 上的实现[C]//中国计算机学会软件工程专委会 2006 年年会论文集. 长沙: 中国计算机学会, 2006: 21–23.

[2] 连瑞琦, 张兆庆, 乔如良. 指令级并行编译器的数据预取及优化方法[J]. 计算机学报, 2000, 23(6): 576–584.

[3] ISLAM M, BANERJEE S, MESWANI M, et al. Prefetching as a potentially effective technique for hybrid memory optimization[C]//Proceedings of the 2nd International Symposium on Memory Systems. Alexandria: ACM, 2016: 220–231.

[4] PARK Y, SHIN D J, PARK S K, et al. Power-aware memory management for hybrid main memory[C]// Proceedings of the 2nd International Conference on Next Generation Information Technology. Gyeongju: IEEE, 2011: 82–85.

[5] 裴颂文，吴小东，唐作其，等. 异构千核处理器系统的统一内存地址空间访问方法[J]. 国防科技大学学报, 2015, 37(1): 28–33.

[6] QURESHI M K, SRINIVASAN V, RIVERS J A. Scalable high performance main memory system using phase-change memory technology[C]//Proceedings of the 36th International Symposium on Computer Architecture.

Austin: ACM, 2009: 24–33.

[7] LEE H G, BAEK S, NICOPOULOS C, et al. An energy- and performance-aware DRAM cache architecture for hybrid DRAM/PCM main memory systems[C]// Proceedings of the 29th International Conference on Computer Design. Amherst: IEEE Computer Society, 2011: 381–387.

[8] 罗乐, 刘轶, 钱德沛. 内存计算技术研究综述[J]. 软件学报, 2016, 27(8): 2147–2167.

[9] PEI S W, ZHANG J G, XIONG N X, et al. Performance-energy efficiency model of heterogeneous parallel multicore system[C]//Proceedings of the 6th Green and Sustainable Computing Conference. Las Vegas: IEEE, 2016: 1–6.

[10] YIN H, SONG D. Temu: binary code analysis via whole-system layered annotative execution[R]. Berkeley: University of California , 2010.

[11] RAMOS L E, GORBATOV E, BIANCHINI R. Page placement in hybrid memory systems[C]//Proceedings of the International Conference on Supercomputing. Tucson: ACM, 2011: 85–95.

[12] 张进宝. 一种基于页面热度的异构内存能耗管理机制[D]. 武汉: 华中科技大学, 2015.

[13] YOON H, MEZA J, AUSAVARUNGNIRUN R, et al. Row buffer locality aware caching policies for hybrid memories[C]// Proceedings of the 30th International Conference on Computer Design. Montreal: IEEE, 2012: 337–344.

[14] LIU H K, CHEN Y J, LIAO X F, et al. Hardware/software cooperative caching for hybrid DRAM/NVM memory architectures[C]//Proceedings of International Conference on Supercomputing. Chicago: ACM, 2017: 26.

[15] IOANNIDIS Y E, WONG E. Query optimization by simulated annealing[C]//Proceedings of the 1987 ACM SIGMOD International Conference on Management of Data. San Francisco, California: ACM, 1987: 9–22.

[16] LIU H K, CHEN Y J, LIAO X F, et al. Hardware/software cooperative caching for hybrid DRAM/NVM memory architectures[C]//Proceedings of the International Conference on Supercomputing. Chicago: ACM, 2017: 26.

[17] SANCHEZ D, KOZYRAKIS C. ZSim: fast and accurate microarchitectural simulation of thousand-core systems[J]. ACM SIGARCH Computer Architecture News, 2013, 41(3): 475–486.

[18] POREMBA M, XIE Y. NVMain: an architectural-level main memory simulator for emerging non-volatile memories[C]//Proceedings of 2012 IEEE Computer Society Annual Symposium on VLSI. Amherst: IEEE Computer Society, 2012: 392–397.

[19] HENNING J L. SPEC CPU2006 benchmark descriptions[J]. ACM SIGARCH Computer Architecture News, 2006, 34(4): 1–17.

（编辑：董　伟）