计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.190800115

面向访问模式的混合内存缓存替换策略

刘 伟[1,2,3] 孙童心[1] 杜 薇[1,2]
1 武汉理工大学计算机科学与技术学院 武汉 430070
2 交通物联网技术湖北省重点实验室(武汉理工大学) 武汉 430070
3 嵌入式系统与服务计算教育部重点实验室(同济大学) 上海 201804

摘 要 大数据时代催生了很多以数据为中心的技术和应用,这对计算机主存的速度、容量、能耗提出了更高的要求。为了解决传统DRAM(Dynamic Random Access Memory)内存遇到的瓶颈,由DRAM和非易失性存储NVM(Non-Volatile Memory)组成的混合内存技术受到了广泛的关注。在混合内存环境下,缓存的性能至关重要。针对混合内存环境,已有的缓存替换策略研究都是对LRU2思想的改进,虽然考虑了DRAM数据和NVM数据缺失惩罚不对称的现象,但是在面对LRU(Least Recently Used)性能差的负载时也会存在缓存抖动和污染问题,仍然存在优化空间。文中针对不同类型的负载特点,考虑了不同访问模式下DRAM与NVM数据的竞争关系,提出了一种动态可调整的缓存替换策略DLRP(Dynamic Level Replacement Policy)。该策略在面对不同类型的负载时能动态地选择最优的替换策略,在保持整体命中率较好的同时降低了NVM的缺失和写回。实验结果表明,相比WBAR策略,DLRP不仅在IPC上有平均16.5%的提升,而且在能耗和写操作数量上分别降低了5.2%和5.1%。

关键词:访问模式;缓存替换;混合内存;NVM
中图法分类号 TP333

Access Pattern-oriented Cache Replacement Strategy for Hybrid Memory Architecture

LIU Wei[1,2,3],SUN Tong-xin[1] and DU Wei[1,2]
1 Department of Computer Science and Technology,Wuhan University of Technology,Wuhan 430070,China
2 Hubei Key Laboratory of Transportation Internet of Things,Wuhan University of Technology,Wuhan 430070,China
3 Key Laboratory of Embedded Systems and Service Computing,Tongji University,Shanghai 201804,China

Abstract With the increasing demand on memory capacity and energy consumption,current DRAM based memory systems face the scalability challenges in terms of storage density and power. Hybrid memory architecture,a promising approach to large-capacity and energy-efficient main memory composed of emerging Non-Volatile Memory (NVM) and DRAM has received extensive attention. Cache plays an important role and highly affects the number of write and read to NVM and DRAM blocks. However,existing cache policies based on LRU failed to fully address the significant asymmetry between NVM operations and DRAM operations under different type of workloads. Cache trashing and scans problems can still seriously affect the performance of the system. By analyzing characteristics of different types of load and the competition between DRAM and NVM data under different access patterns,this paper proposes a dynamically adjusted level cache replacement strategy (DLRP). Experiment results show that proposed strategy improves the performance by 16.5% on average compared with a state-of-the-art cache policy (WBAR). DLRP also reduces energy consumption and NVM writes by 5.1%,5.2% against WBAR.

Keywords Aaccess pattern,Cache replacement,Hybrid memory,NVM

到稿日期:2019-08-23 返修日期:2020-02-04 本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:国家自然科学基金面上项目(61672384);湖北省自然科学面上项目(2020CFB749);教育部人文社科基金项目(16YJCZH014);中央高校基本科研业务费(WUT:2016III028,2017III028-005);嵌入式系统与服务计算教育部重点实验室(同济大学)开放基金(ESSCKF2018-05)
This work was supported by the General Program of National Natural Science Foundation of China(61672384),Natural Science Foundation of Hubei Province(2020CFB749),Humanities and Social Sciences Program of Ministry of Education(16YJCZH014),Fundamental Research Funds for the Central Universities (WUT:2016III028,2017III028-005) and Open Fund of Key Laboratory of Embedded Systems and Service Computing of Ministry of Education,Tongji University(ESSCKF2018-05).
通信作者:刘伟(wliu@whut.edu.cn)

1　引言

近年来，大数据催生的很多以数据为中心的技术和应用对计算机内存的容量、速度、能耗提出了更高的要求。传统的DRAM内存技术因存储密度小、存在易失性、刷新和静态功耗大等问题难以满足新技术对大容量内存的需求。为解决传统计算机内存面临的瓶颈，由非易失存储器NVM[1]和DRAM共同组成计算机内存的混合内存技术受到了学术界的广泛关注。相比DRAM，NVM具有非易失性、存储密度更大、静态功耗极低等优点，但NVM的读写性能不如DRAM，尤其是写延迟和写能耗相对较高，且存在寿命限制。如何充分利用NVM和DRAM两种介质各自的优势，在新的混合内存结构下进行优化，尽可能降低NVM的读写性能缺陷对系统带来的负面影响成为了目前体系结构研究的热点。目前已有大量针对混合内存技术的研究，主要包括平行结构和层次结构。平行结构能够充分利用DRAM和NVM的容量空间，并且相比层次结构，NVM的访问延迟也较低。很多学者针对混合内存环境下的内存管理做了大量研究[2-7]。例如，Lee等[2]提出了CLOCK-DWF页面替换策略，Salkhordeh等[3]在操作系统层设计了一种混合内存环境下的页面迁移策略。在内存管理领域，通过设计页面管理策略，尽可能让DRAM吸收大量的内存访问，减少NVM页面的访问，可以提高混合内存的性能。然而，在混合内存环境下设计合理的缓存替换策略，通过降低NVM数据的缺失率，同样可以减少对NVM的访问次数。

混合内存结构的出现对缓存替换策略的设计带来了新的挑战。在计算机系统中，高速缓存在整个存储体系中扮演着重要的角色，它能够有效弥补主存与处理器之间的速度差异，提高CPU的执行效率。缓存的性能直接关乎着整体系统的性能，通过研究发现：在混合内存环境下，由于DRAM数据和NVM数据存在缺失惩罚不对称的现象，传统以缓存命中率来衡量缓存性能的方法在混合内存环境下不再有效[8]。并且，由于NVM的写操作延迟较长以及能耗很高，大量对NVM的写回操作会严重损害混合内存系统的性能[9]。另外，不同访问模式下DRAM与NVM数据对于缓存空间的竞争也是影响缓存性能的关键因素。

缓存的访问模式主要可以分为4类：LRU友好型模式、缓存抖动模式、流访问模式和混合访问模式[10]。单一的缓存替换策略往往只对一种访问模式有较好的性能，如传统的LRU替换策略只有在面向LRU友好型模式时具有最理想的性能，但是在面对缓存抖动模式时性能很差。而且不同访问模式下，DRAM与NVM的竞争呈现出不同的特点，缓存替换策略也应该随之改变。混合内存环境下已有的缓存替换策略都是基于LRU思想进行的改进，通过优化策略降低NVM数据的缺失来提高缓存的性能，但没有考虑不同访问模式下具体DRAM和NVM的竞争特点。本文分析了不同的访问模式下DRAM和NVM数据具体如何竞争，针对两类不同的访问模式提出了不同的替换算法，然后采用set dueling技术[11]进行动态的选择，形成了一种动态等级替换策略DLRP。该策略在面对不同的访问模式时可以动态地选择最优的缓存替换策略，在保持整体较好的缓存命中率的同时，降低了NVM的缺失率。实验结果表明，DLRP策略相比WBAR策略在IPC、总能耗以及NVM写操作数量上均有较好的性能。

2　相关工作

缓存替换算法的研究一直是研究热点，在容量和相联度确定的情况下，缓存替换算法是影响缓存性能的主要因素。目前传统内存下的缓存替换策略已经有了大量且深入的研究，根据已有的研究，缓存的优化策略大致可以分为3类：替换算法、分区策略、死块预测。替换算法主要是从插入策略、推进策略和替换策略3个部分对缓存进行优化。分区策略的思想是将缓存划分成若干个区域，通过减弱不同数据之间的互相干扰来提高缓存的性能。假如某个数据块从插入到最后替换出缓存一直没有得到访问，则这样的块被称为死块。如果能够及时地识别、预测出死块并及时地将其驱逐出缓存则可以有效地提高缓存的性能，这样的策略称为死块预测。传统的缓存替换策略的研究基本是以提高命中率为目标，然而在混合内存环境下，命中率的提高并不意味着更好的性能。中国科学院大学的Wei等[12]首次提出一种基于混合内存平行结构的缓存管理策略(Hybrid-Memory-Aware Partition，HAP)。HAP策略在逻辑上将整个末级缓存分成DRAM数据和NVM数据两个部分，通过对采样数据的分析比较，动态调整NVM数据区域在整个缓存的占比，以使LLC中的DRAM与NVM数据的比例最优。但是HAP只是在宏观的占比上考虑了DRAM与NVM之间的竞争，并没有具体考虑不同数据之间的差异。华中科技大学的Chen等[8]认为传统的缓存命中率在混合内存架构下不再有效，并采用AMAT(Average Memory Access Time)衡量缓存的性能，提出了MALRU(Miss-penalty aware LRU-based cache replacement policy)策略。MALRU也是一种分区策略，该策略考虑了DRAM和NVM缺失惩罚的不对称性以及不同时间局部性的DRAM与NVM数据之间的竞争。该策略将缓存分成保留区和替换区两部分，把NVM块和时间局部性好的DRAM数据放在保留区，优先替换时间局部性差的DRAM块，提高了缓存的性能。但是该策略并没有考虑NVM写操作的影响以及dirty块与clean块之间的竞争。山东大学的Zhang等[9]在混合内存架构下从替换算法的角度提出了WBAR(Write-back Aware LLC replacement policy)优化策略。该策略考虑了NVM写操作对系统带来的负面影响，针对不同数据之间的特点采用不同的插入和推进策略，通过降低NVM的缺失和写回，提高了缓存的性能。但是该策略仍然是基于LRU思想的改进算法，未考虑到不同访问模式对缓存替换策略的影响。目前，在混合内存环境下的缓存替换策略的研究仍然较少，以上策略都是基于LRU思想的改进，未考虑不同访问模式的特点。不同的访问模式下，不同数据之间的竞争关系也不同，单一的替换策略的性能会有较大的差别。

3　动态等级替换策略

对于一种缓存替换策略，在面对不同的访问模式时会拥有不同的性能。例如，LRU在面对LRU友好型模式时会有

理想的性能，但对于工作集大于缓存容量的程序，使用 LRU 策略会发生缓存抖动，性能会急剧下降，并且当访问模式中存在流访问序列时会造成缓存污染问题。针对这两个问题，已有的研究发现利用 LFU(Least Frequently Used)策略来预测数据的重用概率可以有效地解决由流访问模式带来的缓存污染问题[13]，同时 LFU 策略在面对缓存抖动模式时也有较好的性能。为了验证访问模式在混合内存下的影响，本文在混合内存环境下使用 SPEC CPU2006 测试集测试了 LRU 和 LFU 两种不同的替换策略的性能。

从图 1 可以看出，在 bzip2，dealII，gobmk 3 个测试集上，混合内存环境下 LRU 策略的性能明显高于 LFU 策略；但在 gcc，lbm，milc，omnetpp，xalancbmk 等测试集上，LFU 的性能相对更好。SPEC CPU 2006 测试集大致被分成两类，一类是 LRU 策略的性能更好，另一类则是 LFU 的性能更好。因此，本文将负载划分为两类，一类是 LFU 策略表现更好的负载，这类负载主要由缓存抖动模式和流访问模式组成，主要特点是最近被访问的数据会在较长的一段时间后才会被再次访问，时间局部性较差，本文称其为 LFU 友好型负载；另一类是 LRU 策略表现更好的负载，这类负载主要由 LRU 友好型访问模式组成，其中大部分数据具有较好的时间局部性，本文称其为 LRU 友好型负载。下文将分别对这两类负载设计不同的缓存替换算法。

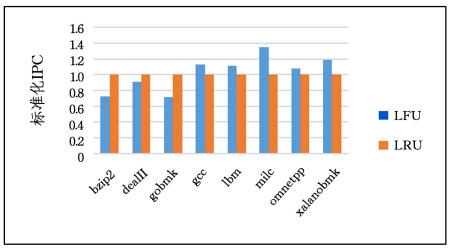

图 1 混合内存环境下 LRU 与 LFU 性能的对比

Fig. 1 Performance comparison of LRU and LFU in hybrid memory environment

3.1 LFU 友好型负载与 LRP 算法

对于 LFU 友好型负载，首先分析缓存抖动的访问模式。假设一段循环访问模式的工作集超过了缓存的容量，在传统的 LRU 缓存替换策略下会造成缓存抖动。在传统的 DRAM 内存环境下，很多学者提出了合适的策略来解决缓存抖动问题，但是在混合内存环境下，由于 NVM 和 DRAM 数据存在缺失惩罚不对称的现象，以往的策略可能导致对 NVM 内存访问过载，影响系统的整体性能。假设一段循环访问模式的工作集为 $(D_1, D_2, \cdots, D_k, N_1, N_2, \cdots, N_m)^N$，包含 DRAM 和 NVM 两种数据(工作集大于缓存容量)。经过分析，混合内存环境下理想的缓存替换策略应当尽可能保留循环体中的 NVM 数据。理想的结果有两种情况：1)循环体中有大量的 NVM 数据，缓存优先保留所有的 NVM 数据，由于空间有限因此舍弃工作集中的 DRAM 数据；2)循环体中只有少量的 NVM 数据，缓存在保留循环体中少量的 NVM 数据后，用剩余的空间来保留一部分 DRAM 数据。

为了解决该模式下 DRAM 与 NVM 的竞争问题，本文为每个进入缓存中的缓存块设置了代表替换等级的权值 RLV (Replacement Level Value)，将 DRAM 设置为最高等级 RLVMAX，代表优先被替换出缓存，将 NVM 设置为低等级 RLVMAX−1。等级高的数据优先被逐出缓存，这样缓存就可以自动地从工作集中筛选出 NVM 数据并保留下来，减少了 NVM 的缺失，并且由于 LFU 友好型负载中大部分数据具有较差的时间局部性，因此不会因为优先保留 NVM 而丢失局部性较好的 DRAM 数据，从而造成缓存的性能损失。同样由于大部分数据具有较差的时间局部性，本文设置每次插入都从末尾开始执行，避免了缓存污染的问题。但是这样的策略仍然存在一个关键问题：当需要进入新的循环时，这个策略由于总是在末尾插入新的缓存块，缓存中无法引入新的工作集，因此当循环体结束时或者工作集改变时，缓存的命中率会急剧下降。为了解决这个问题，本文借鉴了文献[11]的思想，引入了一个参数 bimodal_interval，每隔一段较长的时间后插入一个等级值设为 RLVMAX−2 的缓存块，该方式在保持较好的缓存命中率的同时解决了无法引入新的工作集的问题。本文提出的 LRP(Level Replacement Policy)算法的描述如算法 1 所示。

算法 1 LRP 缓存替换算法

```
1. Set=extractset(r->getaddr())/*r代表对LLC的请求。每当一个请求到来时，对应访问的set的bimodal_interval加1，达到临界值时，重新置为0*/
2. If Set.bimodal_interval=BIMODAL then
3.  Set.bimodal_interval=0
4. else Set.bimodal_interval++
5. endif
6. If r hits in cache block C then
7.     If C is NVM then
8.        C->Rlv=0 /* Rlv是赋予每个缓存块代表替换优先级的等级值*/
9.     Else if C is DRAM then
10.       C->Rlv=RLVMAX/2
11. end if
12. end if
13. Else r miss in cache then
14. blk *findVictim()/*从缓存的末尾开始找plv值最大的缓存块，返回该块的指针*/
15.    while(blk->Rlv<RLVMAX-1){
16.       for all block in cache set
17.          block-> Rlv++
18.    }
19.    Insert (blk)/*将原来的要替换出去的块无效化，重新设置blk参数*/
20.    If blk is NVM then
21.       blk->Rlv=RLVMAX-1
22.    Else blk is DRAM then
23.       blk->Rlv=RLVMAX
24.    end if
25.    if Set.bimodal_interval= BIMODAL then
26.       blk->Rlv=RLVMAX-2
27.    endif
28. endif
```

首先为每个缓存 set 设置 bimodal_interval 的属性，该属性的值随着访问次数的增加而增加，当 bimodal_interval 达到临界值时重新置为 0，同时将此次插入的缓存块设置为 RLVMAX-2(算法 1 中第 1－5 行)。当缓存命中时，对于 NVM 数据，本文采用 HP(Hit Priority)策略，每当命中发生时，缓存块的等级值置为 0。对于 DRAM 数据，采用 FP(Frequency Priority)策略，每当命中发生时，降低该缓存块的等级值(算法 1 中第 6－12 行)。如果请求缺失，首先从缓存末尾查找等级值最高的块作为替换块，同时判断等级值最高是否为 RLVMAX，如果不是，则该 set 中所有的缓存块的等级值加 1，直到最高值等于 RLVMAX(算法 1 中第 13－18 行)。然后，在插入新的缓存块时根据数据类型的不同设置不同的等级，DRAM 数据设置为 RLVMAX，NVM 数据设置为 RLVMAX-1(算法 1 中第 19－28 行)。

3.2　LRU 友好型负载与 TLRP 算法

不同于 LFU 友好型负载，LRU 友好型负载的主要特点是大部分数据具有较好的时间局部性，但是也存在时间局部性较差或者没有时间局部性的数据。对于这类访问模式，缓存替换策略希望尽可能筛选出工作集内时间局部性好的数据保留下来。但是，在混合内存下，问题变得更加复杂。基于前文对混合内存特点的分析，缓存仍然希望尽可能利用 NVM 数据的时间局部性，优先保留时间局部性好的 NVM 数据。从这一点来讲，DRAM 数据的替换优先级仍然应该高于 NVM 数据，这样 DRAM 总是优先比 NVM 替换出去。但是，由于并不清楚 DRAM 数据和 NVM 数据的时间局部性差异，即使 NVM 的缺失代价高于 DRAM，为了完全利用 NVM 的时间局部性而放弃时间局部性有可能更好的 DRAM 块显然是不合理的。例如，当时间局部性好的 DRAM 数据占更多的比例时，该策略会导致缓存保留大量的时间局部性差的 NVM 数据，从而影响缓存的整体性能。

针对这个问题，本文提出了 TLRP(Two-Level Replacement Policy)策略，该策略额外引入了位置等级的概念，将原本的等级分成了两个等级(基础等级和位置等级)来代表替换的优先级。基础等级与 LRP 策略中的等级类似，代表数据类型的替换优先级，位置等级代表数据在缓存的位置对应的优先级。将一个 set 的缓存均匀划分成若干个等级，越接近缓存的末尾，位置等级就越高，总的等级越高，越优先被逐出缓存。由于针对的是 LRU 友好型负载，因此 TLRP 策略设置头部插入，优先从尾部开始替换总等级最高的数据。通过引入位置等级，可以间接地表明数据对应的时间局部性特征，这样缓存既可以优先筛选时间局部性好的 NVM 数据，又能保留下时间局部性好的 DRAM 数据，然后及时地替换出时间局部性差的 NVM 数据。另外，对于 LRU 友好型负载，TLRP 将数据进行了更细的划分，考虑了脏块与非脏块的差异，研究表明脏数据的命中率明显更高，通过将脏块设置为更高的等级可以减少写回操作。

TLRP 算法与 LRP 相似，但是没有 bimodal_interval 属性，增加了位置等级的概念。TLRP 算法的描述如算法 2 所示。每当命中发生时，对 NVM 和 DRAM 数据分别采用 HP 和 FP 策略，同时将命中的缓存块移动到头部(算法 2 中第 1－7 行)。如果请求缺失，则首先从缓存末尾查找位置等级值加上基础等级值最高的块作为替换块，同时判断等级值最高是否为 RLVMAX，如果不是，则该 set 中所有的缓存块的等级值加 1，直到最高值等于 RLVMAX(算法 2 中第 8－13 行)。然后，当插入新的缓存块时根据数据类型的不同设置不同的等级，DRAM 数据设置为 RLVMAX－1，NVM 设置为 RLVMAX－2；如果是写回操作，表明这个块是脏块，那么该块的等级值在刚才的基础上再减 1，然后插入到头部(算法 2 中第 14－24 行)。

算法 2　TLRP 缓存替换算法

```
1. If r hits in cache block C then /* r 代表对 LLC 的请求 */
2.     If C is NVM then C->Rlv=0 /* Rlv 是赋予每个缓存块代表替换优先级的等级值 */
3.     Else if C is DRAM then
4.       C->Rlv=RLVMAX/2
5.     MoveToHead(C)
6.     end if
7. end if
8. Else r miss in cache then
9.     *blk =findVictim() /* 从缓存的末尾开始找 plv 加上位置等级值最大的缓存块，返回该块的指针 */
10.    while(blk->Rlv<RLVMAX){
11.    for all block in cache set
12.      block-> Rlv++
13   }
14.    Insert (blk)/*将原来的要替换出去的块无效化，重新设置 blk 参数*/
15.    If blk is NVM then
16.      blk->Rlv=RLVMAX-2
17.    Else blk is DRAM then
18.      blk->Rlv=RLVMAX-1
19.    end if
20.    if is Writeback(r) then
21.      blk->Rlv=blk->Rlv-1
22.    end if
23.    MoveToHead(blk)
24. endif
```

3.3　动态等级替换策略 DLRP

单一的缓存替换策略在面对不同的负载时性能也有较大差异。本文在针对两类负载提出各自的缓存替换算法之后，为了使缓存在面对多种不同的访问模式时能被动态地调整，本文提出了动态等级替换策略 DLRP，其采用 Set Dueling 技术[11]将前文设计的两种替换算法结合起来，形成动态可调整的替换策略。已有的研究表明，通过设置一部分缓存为样本缓存，样本缓存的性能数据可以有效地反映整体缓存的性能。本文设置两组样本缓存以及一个 PSEL 饱和计数器，一组采用 LRP 策略，另一组采用 TLRP 策略。当 LRP 组发生缺失时，PSEL 增加 1；当 TLRP 组发生缺失时，PSEL 计数器减 1。通过判断 PSEL 的正负来选择合适的策略：若 PSEL 为正，则剩余的缓存全部采用 TLRP 策略，反之则采用 LRP 策略。已有的研究[11]证明了当 r 大于 0.2 时(r 表示两个策略的相对差异，具体 r 的表达式和详细证明本文不再给出)，只需要牺

牲 32～64 个 set 就足以让 Set Dueling 机制从两个不同的替换策略中选择出合适的策略。通过在大量测试集中对 LIP 和 TLIP 进行测试可知，LRP 和 TLRP 的 r 值大于 0.2。因此本文设置的样本缓存容量为 64 个 set，交叉分成两组，每组 32 个 set，分别采用 LIP 和 TLIP 策略。使用 Set Dueling 进行动态替换策略的实现只需要一个饱和计数器而不需要额外的存储，并且其执行效果与所选择的替换策略的差别很小。这样，在面对不同的访问模式时通过牺牲少量的缓存空间就可以实现缓存替换策略的动态调整。

4 实验及结果分析

4.1 实验环境设置

本文实验平台由 Gem5 全系统模拟器[14]和 NVmain[15]内存模拟器组成。Gem5 用来模拟处理器和缓存，NVMain 用来模拟混合主存。实验系统设置了三级缓存，其中一级和二级缓存是独立的，从属于各自的 CPU，三级缓存是共享缓存。表 1 列出了本文的系统配置和实验工作负载。实验从 SPEC CPU 2006 中选取 5 个单线程负载和 3 个多程序工作负载作为本实验的测试程序。

表 1 实验环境参数

Table 1 Parameters of simulation architecture

Parameters	Configuration
Processor	1-core/4-core CMP, 2 GHz, out-of-order
L1 caches	32KB I-caches, 64 KB D-caches 2-way associative, 64B cache line, 2-cycle latency
L2 caches	256 KB, 4-way associative, 64 B cache line, 20-cycle latency
L3(shared)	2MB, 16-way associative, 64 B cache line, 30-cycle latency, write-back
Main memory	DRAM: 1GB (1 channel), NVM: 3GB (3 channel) DRAM: tRCD 5 cycles latency, tRP 5 cycles latency NVM: tRCD 22 cycles latency, tWP 60 cycles latency
Workload	cactusADM, dealII, lbm, libquantum, milc mix1: (bzip2, sphinx3, milc, soplex) mix2: (dealI, milc, xalancbmk, omnetpp) mix3: (gobmk, bzip2, omnetpp, gromacs)

4.2 实验结果

本实验选取 LRU 策略、WBAR 策略与 DLRP 策略在相同的实验环境下进行测试。实验结果分别从 IPC、内存总能耗、NVM 写操作数量 4 个指标进行对比。为方便比较，实验结果均以 LRU 策略的实验结果为标准做归一化处理。

图 2 给出了不同的测试程序下，LRU，WBAR，DLRP 3 种不同的缓存替换策略在混合内存系统下的 IPC 实验结果。从图 2 可以看出，WBAR 和 DLRP 在混合内存环境下的性能都比 LRU 优秀，而 DLRP 相比 LRU 策略平均有 16.5%的提升，相比 WBAR 有 10.5%的提升。DLRP 在 dealII 和 cactusADM 两个测试集下的性能相比 WBAR 提升得不明显，但是在 lbm，milc，libquantum 测试集中有较为明显的提升，最高提升约 14.1%。这是由于对于以 cactusADM 和 dealII 为代表的 LRU 友好型负载，WBAR 是基于 LRU 思想做的改进，本身就有较好的性能，而 DLRP 能提升的幅度也较小。但是，对于以 lbm 为代表的 LFU 友好型负载，WBAR 虽然相比 LRU 有少量的提升，但是由于 LRU 本身在面对缓存抖动模式时性能较差，因此 DLRP 相比有较为明显的性能提升。

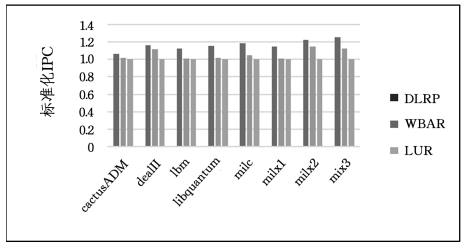

图 2 3 种替换策略的 IPC 对比

Fig. 2 IPC comparison of three replacement policies

图 3 给出了混合内存的总能耗的对比结果，DLRP 相比 LRU 平均有 10.8%的降低，相比 WBAR 平均有 5.2%的降低。但是与 IPC 相比不同的是，总能耗只有在 milc，mix1 和 mix2 3 个测试集中有较为明显的提升。而对于 lbm 和 libquantum 这两个性能提升明显的测试集，能耗降低得并不明显。为了方便说明这个问题，对图 2 和图 3 进行对比分析。从图 3 可以看出，实际上对于前 4 个测试集，DLRP 对于 NVM 的写操作降低得较少，而且由于 NVM 的写操作能耗相比 DRAM 很高，占据能耗的主要部分，因此虽然 lbm，libquantum 两个测试集的性能提升得较高，但能耗降低仍然较少。由图 3 和图 4 也可以看出，能耗的变化趋势大致与 NVM 写操作的次数成相似的趋势，而与 IPC 的变化趋势有些差异。这也意味着 NVM 的写操作数量是总能耗的主要来源，体现了混合内存环境的特点，IPC 更高并不意味着混合内存系统更好，可能存在 NVM 写操作过载引起能耗过高的问题。

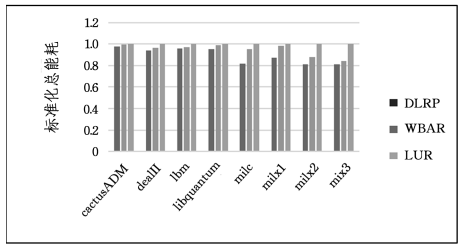

图 3 3 种替换策略的内存总能耗对比

Fig. 3 Comparison of total memory energy consumption of three replacement policies

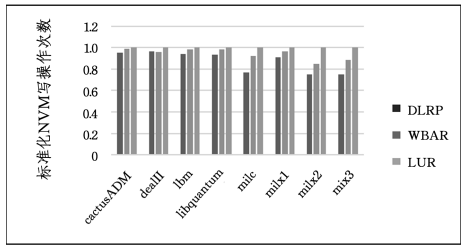

图 4 3 种替换策略的 NVM 写数量对比

Fig. 4 Comparison of number of NVM writes of three replacement policies

为了更具体地说明本文 DLRP 与 WBAR 策略的区别，实验还对比了不同测试集下 3 种策略的 NVM 写操作比例。从图 5 可以看出，对于 NVM 写操作比例这项数据，WBAR 拥有最好的表现，这是由于 WBAR 策略本身就对 NVM 数据设置了很高的优先级，WBAR 也正是通过优化该比例使得系统性

能整体提升。但是，WBAR 并没有考虑访问模式的不同，也没有优化缓存命中率，这就导致对于 lbm，libquantum，milc 这些测试集，由于整体命中率较低，整体的写操作数目仍然较高，影响了系统的性能。而本文的 DLRP 策略则考虑了访问模式的不同，在优化缓存命中率的基础上优化了 NVM 缺失率和写操作数量。虽然 NVM 写操作比例不如 WBAR，但是由于整体的 NVM 写操作数量更少，因此整体的性能更优。

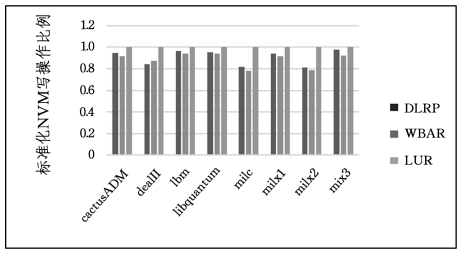

图 5　3 种替换策略的 NVM 写操作比例对比

Fig. 5　Comparison of proportion of NVM writes of three replacement policies

4.3　开销分析

由本文设置的实验系统的参数可知，每个缓存块的大小为 64B，LLC 的相联度为 16。DLRP 策略为每个缓存块设置了 3 位 RLV 代表替换优先级以及 1 位 type 代表数据类型（type 用来区分缓存块的对应是 DRAM 数据还是 NVM 数据）。另外，对于每个缓存 set，DLRP 策略需要一个 7 位的 bimodal_interval 计数值。因此，总体的硬件开销为 4/(64×8)+7/(16×64×8)，硬件开销大约占 LLC 总容量的 0.87%。

结束语　本文在混合内存环境下，针对两类不同的负载进行分析，考虑了 NVM 与 DRAM 在不同类型负载下的竞争关系，分别提出了等级替换策略 LRP 和两级替换策略 TLRP。这两种策略对不同类型的数据设置不同的替换等级，在保持整体较好的缓存命中率的同时减少了 NVM 的缺失和写回。LRP 和 TLRP 分别在面对 LFU 友好型负载和 LRU 友好型负载时有着较为出色的性能。本文采用 Set dueling 使缓存在面对不同的负载时选择最优的替换策略，形成了一种动态可调整的等级替换算法 DLRP。实验结果表明，DLRP 算法在 IPC、总能耗、NVM 写操作数量上有更好的性能。DLRP 的 IPC 相比 LRU 策略平均有 16.5% 的提升，相比 WBAR 有 10.5% 的提升；总能耗相比 LRU 平均有 10.8% 的降低，相比 WBAR 则有 5.2% 的降低；NVM 的写操作数量平均有 12.8% 的降低，相比 WBAR 则有 5.1% 的降低。

参 考 文 献

[1] MAO W，LIU J N，TONG W，et al. A Review of Storage Technology Research Based on Phase Change Memory[J]. Chinese Journal of Computers，2015，38(5)：944-960.

[2] LEE S，B H，NOH S H. CLOCK-DWF：A Write-History-Aware Page Replacement Algorithm for Hybrid PCM and DRAM Memory Architectures[J]. IEEE Transactions on Computers，2014，63(9)：2187-2200.

[3] SALKHORDEH R，ASADI H. An operating system level data migration scheme in hybrid DRAM-NVM memory architecture [C]// Proceedings of the 2016 Conference on Design，Automation & Test in Europe，2016：936-941.

[4] POURSHIRAZI B，ZHU Z. Refree：A Refresh-Free Hybrid DRAM/PCM Main Memory System[C]// 2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS). 2016：566-575.

[5] CHEN J，LIU H K，WANG X Y，et al. Large pages Supported Hierarchical DRAM/NVM Hybrid Memory Systems[J]. Journal of Computer Research and Development，2018，55(9)：2050-2065.

[6] LIU C M，YANG X，JIA G Y，et al. Hybrid Memory Page Management Strategy of Avoiding Page Migrations[J]. Journal of Chinese Computer Systems，2019，40(6)：1318-1323.

[7] LI Q，ZHONG J，LI X，et al. Memory Management Mechanism for Hybrid Memory Architecture Based on New Non-volatile Memory[J]. ACTA ELECTRONICA SINICA，2019，47(3)：664-670.

[8] CHEN D，JIN H，LIAO X，et al. MALRU：Miss-penalty aware LRU-based cache replacement for hybrid memory systems [C]// Proceedings of the Conference on Design，Automation & Test in Europe，2017：1086-1091.

[9] ZHANG D，JU L，ZHAO M，et al. Write-back aware shared last-level cache management for hybrid main memory[C]// Proceedings of the 53rd Annual Design Automation Conference. 2016：1-6.

[10] HUANG Z B，ZHOU F，MA H D. A Cache Replacement Policy Adapting to the Request Access Pattern[J]. Journal of Beijing University of Posts and Telecommunications，2016，39(3)：44-48，53.

[11] QURESHI M K，JALEEL A，PATT Y N，et al. Adaptive insertion policies for high performance caching[J]. ACM SIGARCH Computer Architecture News，2007，35(2)：381-391.

[12] WEI W，JIANG D，XIONG J，et al. HAP：Hybrid-memory-aware partition in shared last-level cache[J]. ACM Transactions on Architecture and Code Optimization，2017，14(3)：24.

[13] JALEEL A，THEOBALD K B，STEELY J S C，et al. High performance cache replacement using re-reference interval prediction (RRIP)[C]// ACM SIGARCH Computer Architecture News. 2010：60-71.

[14] BINKERT N，BECKMANN B，BLACK G，et al. The gem5 simulator[J]. ACM Sigarch Computer Architecture News，2011，39(2)：1-7.

[15] POREMBA M，XIE Y. Nvmain：An architectural-level main memory simulator for emerging non-volatile memories[C]// 2012 IEEE Computer Society Annual Symposium on VLSI. 2012：392-397.

LIU Wei，born in 1978，Ph.D，associate professor，is a member of China Computer Federation. His main research interests include in-memory computing and edge computing.