计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.20210567
59(3):568-581,2022

基于 DRAM 牺牲 Cache 的异构内存页迁移机制

裴颂文[1,2] 钱艺幻[1] 叶笑春[2] 刘海坤[3] 孔令和[4]
[1](上海理工大学光电信息与计算机工程学院 上海 200093)
[2](计算机体系结构国家重点实验室(中国科学院计算技术研究所) 北京 100190)
[3](华中科技大学计算机科学与技术学院 武汉 430074)
[4](上海交通大学计算机科学与工程系 上海 200240)
(swpei@usst.edu.cn)

DRAM-Based Victim Cache for Page Migration Mechanism on Heterogeneous Main Memory

Pei Songwen[1,2], Qian Yihuan[1], Ye Xiaochun[2], Liu Haikun[3], and Kong Linghe[4]
[1](*School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai* 200093)
[2](*State Key Laboratory of Computer Architecture (Institute of Computing Technology, Chinese Academy of Sciences), Beijing* 100190)
[3](*School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan* 430074)
[4](*Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai* 200240)

Abstract When massive data access heterogeneous memory systems, memory pages often migrate between DRAM and NVM. However, the traditional memory page migration strategy is difficult to adapt to the rapid dynamic changes among "hot" and "cold" memory pages. The "cold" pages just migrated from DRAM to NVM will become "hot" again, which results in a large number of redundant migrations, as well as false migrations. Previous related researches only focus on pages that are being migrated without paying too much attention to pages that in the migration waiting queue or that have been migrated. Therefore, this paper proposes a heterogeneous memory page migration mechanism based on DRAM-based victim Cache (VC-HMM) by adding a small capacity of victim Cache between DRAM and PCM. The "cold" pages will be migrated from DRAM to victim Cache. DRAM victim Cache can avoid redundant migrations caused by the main memory pages getting hot again in a short time. Meanwhile, some pages do not need to be written back to PCM that can reduce the number of write operations on PCM and extend the lifetime of PCM. In particular, VC-HMM can automatically update the execution parameters of migration for different workloads to increase the rationality of migration. Experimental results show that compared with other migration strategies (CoinMigrator, MQRA, THMigrator), VC-HMM reduces the average number of PCM write operations by 62.97%,

收稿日期:2021-06-07;**修回日期**:2021-09-07
基金项目:国家自然科学基金项目(61975124);上海市自然科学基金项目(20ZR1428600);上海市科技创新行动项目(20DZ2303500,20DZ2308700,19DZ2301100);上海市数据科学重点实验室开放项目(2020090600003);计算机体系结构国家重点实验室(中国科学院计算技术研究所)开放项目(CARCHA202111)
This work was supported by the National Natural Science Foundation of China (61975124), the Shanghai Natural Science Foundation (20ZR1428600), the Shanghai Science and Technology Innovation Action Plan (20DZ2303500, 20DZ2308700, 19DZ2301100), the Open Project Program of Shanghai Key Laboratory of Data Science (2020090600003), and the State Key Laboratory of Computer Architecture (ICT, CAS) (CARCHA202111).

the average access latency by 22.72%, the re-migration times by 38.37%, and the energy consumption by 3.40%.

Key words VC-HMM; heterogeneous memory system; DRAM based victim Cache; memory page migration; non-volatile memory

摘 要 当海量数据请求访问异构内存系统时,异构内存页在动态随机存储器(dynamic random access memory, DRAM)和非易失性存储器(non-volatile memory, NVM)之间进行频繁的往返迁移.然而,应用于传统内存页的迁移策略难以适应内存页"冷""热"度的快速动态变化,这使得从 DRAM 迁移至 NVM 的"冷"页面可能在短时间内变"热"从而产生大量冗余的迁移操作.当前的相关研究都仅着眼于正在执行迁移的页面而忽视了等待迁移和完成迁移的页面,且判断"冷""热"程度的标准不一,使得冗余的迁移大量产生.因此,提出了一个基于 DRAM 牺牲 Cache 的异构内存页迁移机制(VC-HMM),使用非易失性存储器中工艺较为成熟的相变存储器(phase change memory, PCM),通过在 DRAM 和 PCM 之间增加一个由 DRAM 构成的小容量牺牲 Cache 将系统主存 DRAM 中变"冷"的页面迁移到牺牲 Cache 中,以避免主存页面在短时间内再次变"热"而造成的冗余迁移.同时,还使得迁回 PCM 的部分页面不需要写回,减少 PCM 存储单元的写入操作次数,延长 PCM 的使用寿命.另外,对于不同的工作负载,VC-HMM 可以自适应设置迁移操作的参数,增加迁移的合理性.实验结果表明:与其他迁移策略(CoinMigrator,MQRA,THMigrator)相比,VC-HMM 平均减少了至少 62.97%的 PCM 写操作次数、22.72%的平均访问时延、38.37%的重复迁移操作以及 3.40%的系统能耗.

关键词 VC-HMM;异构内存系统;DRAM 牺牲 Cache;内存页迁移;非易失性存储器

中图法分类号 TP333

在大数据技术快速发展的当下,传统的随机动态存取存储器(dynamic random access memory, DRAM)逐渐暴露出其局限性,如存储密度小、带宽有限、能耗大等[1-3],揭示了传统 DRAM 无法满足越来越大的内存需求.因此,为了解决这一问题,近年来涌现的非易失性存储器(non-volatile memory, NVM)[4-7]被广泛关注,并与 DRAM 组成异构内存系统[8-11],大量研究围绕非易失性内存和异构内存展开[12-15].NVM 拥有比 DRAM 更大的存储密度,无静态能耗,即不需要反复刷新保存数据,从而成为构建主存的理想存储设备.

然而,与传统 DRAM 相比,NVM 同样具有写操作时延高、耐久性差、写操作能耗高等缺点.以相位存储器(phase change memory, PCM)为例,相比简单的电荷移动,PCM 需要更多的能量来改变相变材料(基硫族化合物)内部的原子结构.另外,PCM 的写入时延大约是 DRAM 写入时延的 10 倍.更详细的 DRAM 与部分 NVM 的特性对比如表 1 所示.因此,当 DRAM 和 NVM 构成异构内存时,应将访问频率更高的"热"内存页存放在 DRAM 中,而将"冷"内存页存放在 NVM 中,如此,系统便可以同时兼备 NVM 和 DRAM 两者的性能优势.但这同时又衍生出了新的问题,即如何进行内存页面的放置迁移操作.

Table 1 Properties of DRAM and NVM

表 1 DRAM 和 NVM 的特性

参数	DRAM	PCM	STT-RAM
读操作时延/ns	<10	10~100	2~20
写操作时延/ns	<10	20~120	5~35
耐久性/次	N/A	10^8~10^{12}	10^{12}~10^{15}
写操作功耗/(nJ·b^{-1})	~0.1	<1	1.6~5

近年来,许多相关研究都试图解决在 NVM 和 DRAM 之间进行内存页迁移的问题,如多级队列替换算法(multi-queue replacement algorithm, MQRA)[16]、CLOCK-DWF[17]、基于双向散列链表的页面迁移机制(THMigrator)[18]等.虽然它们在一定程度上优化了异构内存系统带来的问题,但也造成了冗余的页面迁移操作,从而降低系统的性能.由于不准确的迁移执行参数与预测机制,进行迁移的页面实际上可能不够"热",而有些页面则可能会过早地被迁回 NVM,这将导致 2 种设备之间频繁地往返迁移.同时页面迁移造成的不可避免的操作和资源消耗,例如页面的重新分配、页表的更新、带宽

消耗等,都将带来巨大的系统开销.因此,不必要的迁移所产生的代价可能反而超过了页面迁移带来的收益.此外,NVM 的大量数据写入操作也缩短了设备的使用寿命,并增加了访问时延.

我们通过实验观察到,当应用程序对海量数据进行处理时,主存 DRAM 中被逐出的"冷"页,可能会在短时间内再次变"热".在这种情况下,页面将进行重复的迁移操作,这将会增加 NVM 的写入次数且消耗不必要的系统资源.

定义 1. 重复迁移.对任意内存页面 x 和迁移操作序列 $M=(M_1,M_2,\cdots,M_i,\cdots,M_n)$,若当前执行的迁移操作 M_i 使得 x 的迁移总次数 $n\geqslant 2$,则迁移操作 M_i 为 x 执行了一次重复迁移.

定义 2. 冗余迁移.对任意内存页面 x 和迁移操作序列 $M=(M_1,M_2,\cdots,M_i,\cdots,M_n)$,若当前执行的迁移操作 M_i 使得 x 的迁移总次数 $n\geqslant 2$,且对页面 x 的第 i 次迁移与第 $i-1$ 次迁移之间的时间间隔 $t\leqslant\sigma$,则迁移操作 M_i 为 x 执行了冗余迁移.

其中 σ 是较小的时间阈值,由不同的内存器件和内存层次结构的差异而定.

如图 1 所示,以 CoinMigrator,MQRA,THMigrator 为例,对于基准测试 bzip2,cactusADM,omnetpp,sjeng,与总的迁移次数相比,平均有 72.50%,66.33%,69.050%,69.11%的迁移操作为重复迁移.此外,由于迁移参数的主观人为设定,一些研究中的页面"冷""热"程度判断机制可能无法准确判断并预测一个页面是否真的"热".同时,被判定为"热"页的页面可能不会立即执行迁移,由于前一个页面未完成迁移操作而导致当前页面需要等待,在这种情况下,在当前页面完成迁移时,它可能不会再与之前一样"热",并且由于"冷""热"判断标准的不同被对应迁回 NVM 的页面可能比迁移到 DRAM 的页面更"热".这样的页面迁移是不必要的,它们不仅会降低系统性能,同时也增加了系统资源的消耗.

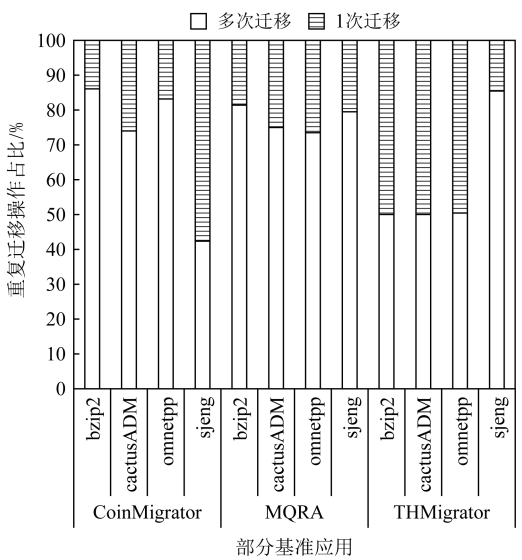

Fig. 1 The percentage of re-migration of CoinMigrator, MQRA and THMigrator

图 1 CoinMigrator/MQRA/THMigrator 的重复迁移操作占比

为了解决这些问题,本文提出了一种基于 DRAM 的牺牲 Cache(DRAM-based victim Cache, DVC)的异构内存页迁移机制(victim Cache for page migration on hybrid main memory system, VC-HMM).当执行页面迁回时,主存 DRAM 中的"冷"页将迁移到 DVC 中,而不是立即写回 PCM.在"冷"页再次变"冷"之后,迁移控制器将判断该页是否为脏页.由于 PCM 仍保留原始页面数据,故若页面不脏,则不需要写回.VC-HMM 还可以根据迁移收益情况自主地调整迁移参数,从而自适应不同的工作负载.本文工作的主要贡献有 3 个方面:

1) 提出了一种基于 DRAM 牺牲 Cache 的异构存储系统结构来消除冗余迁移.

2) 提出了适用于 VC-HMM 的基于双向散列链表的页面迁移机制进行更有效的迁移操作.

3) 提出了在不同工作负载下自适应更新迁移执行参数的方案,实现合理的迁移阈值控制.

实验结果表明:与其他迁移策略(CoinMigrator, MQRA,THMigrator)相比,VC-HMM 平均减少了至少 62.97%的 PCM 写操作次数,22.72%的平均访问时延,38.37%的重复迁移操作,以及 3.40%的系统能耗.

1 相关工作

由于 NVM 难以作为 DRAM 的直接代替,异构内存系统的提出给内存问题提供了解决的方案.而为了兼具 NVM 与 DRAM 两者的性能优势,基于异构存储系统的页面迁移机制研究显得尤为重要.本节将首先介绍一些以往有关异构内存页面迁移机制研究的相关工作,并分析它们的优点与不足.

由于 NVM 在进行写操作时具有远大于 DRAM 的访问时延和能量消耗,许多研究着眼于如何减少 NVM 的写入操作来提升系统性能.Lee 等人[17]提出了 Clock-DWF 算法将写请求频繁的页面迁移到 DRAM 中.对于一个进入内存的新页面,当访问为

写操作时,该页面将直接载入 DRAM 中,反之放置于 NVM 中.对于在 NVM 中的页面,一旦被写请求命中则该页面将迁移到 DRAM 中.如果 DRAM 容量已满,Clock-DWF 则会选择写请求命中次数最少或最长时间没有被访问的页面作为"冷"页逐出 DRAM.该机制考虑到了写操作对 NVM 的影响,但迁移的评判标准过于简单,不能正确地判断"冷""热"度从而增加迁移的次数.Kim 等人[19]提出了基于自适应分类(adaptive-classification clock)的迁移机制.通过历史的访问模式,每个页面将被设定为不同的状态来判断页面是否为读/写密集以及是否执行迁移操作,减少了不必要的迁移.

除了以上通过监控页面访问来执行迁移的机制外,使用队列结构存储页面信息的策略也在以往的研究中广泛使用.Seok 等人[20]提出了以 4 个 LRU 队列为基础的迁移模型.所有的页面被分为 4 个类别:NVM 写频繁页面、NVM 读频繁页面、DRAM 写频繁页面以及 DRAM 读频繁页面,并且根据最近的访问时间进行有序排列.根据过去的访问信息,模型能够预测未来访问的类型,更加准确地进行页面的迁移,以减少 NVM 的写操作.Zhou 等人[16]提出了多级队列迁移机制(MQMigrator)来提高迁移页面判断的准确性.多级队列迁移机制的实现基于 3 个页面属性,即最小生存周期、基于访问频率的优先级以及访问时间.每一个页面都将基于"冷""热"度与生存周期来更新节点信息并移动到对应的队列中.通过多个 LRU 队列的维护,系统能够更加准确地选择最"热"和最"冷"的页面并且监控所有的页面状态,但维护成本和时间复杂度较高.Tan 等人[21]提出了一种基于历史访问信息的统一"冷""热"度计算方式,相较于大部分研究中使用访问次数作为"冷""热"度度量标准的方法,统一的计算方式能更加公平地表现页面的访问状态.同时,迁移的阈值也能够根据每次迁移的收益来进行更新.随后,Tan 等人[22]又进一步提出减少 NVM 写入操作的机制.当一个脏页写回 NVM 时,只需要写回以行大小为单位的数据而不用写回整个页面.该机制减少了冗余写回操作的同时延长了部分 NVM 存储单元的使用寿命.

除了减少 NVM 的写入操作和提高"热"页的选择准确性之外,一些研究也提供了其他的迁移思路.Ryoo 等人[23]提出了一种基于粒度感知的迁移机制,迁移页面的粒度大小将根据不同工作负载的不同访问模式进行更新,减少不必要的迁移.为了解决迁移能耗的问题,Zhan 等人[24]提出了一种基于能耗感知的迁移机制.该机制将权衡页面继续留存在 NVM 中和迁移到 DRAM 中 2 种情况的能量代价,为系统节约不必要的迁移能耗.

在内存系统结构的研究中,Guo 等人[25]提出了新的内存系统结构 DR-HBM,减少对 NVM 的写入.在该结构中,DRAM 被分为 2 个部分:容量较小的 DRAM Cache 与容量较大的 DRAM 主存.Cache 仅仅缓存源于 NVM 的页面并且按照页面状态的不同,在迁回时不用写回 NVM 而暂时保存在 DRAM 中以便再次访问.在 Cache 中第 1 次未命中的页面不会立刻执行缓存直到再一次被请求.这样的结构设计减少了 NVM 的写入并且节约了迁移所占用的带宽.Islam 等人[26]提出了一种即时迁移的策略,通过在内存系统中添加新的硬件设备实现实时的页面迁移而不需要中断访问,为系统减少了访问的时延.

以上所有的迁移策略都对异构内存的页面迁移进行了逐步的优化升级,但同时也具有一定的局限性.大多数的迁移策略只关注迁移的页面,很少关注仍留存在 DRAM 或在迁移队列中的页面.通常在一个迁移过程中,一对"冷""热"页面将会交替迁移,从 DRAM 中选中的"冷"页将被 NVM 中的"热"页代替以节约 DRAM 的存储空间,但是由于不公平的"冷""热"度评价机制,可能会存在"冷"页面比"热"页面更"热"的情况,或"冷"页面被逐出后在短时间内再次变"热".另外,由于页面达到迁移条件后可能不会立即执行迁移,所以当完成迁移时,当前页面可能不会与之前一样"热",甚至比在它之后执行迁移的页面更加"冷".在进行页面逐出迁回时,从 DRAM 迁移回 NVM 的页面都需要进行写回操作,但是对于读频繁页面来说,页面中的数据可能并没有经过修改,重新写回 NVM 将增加 NVM 的写操作且缩短 NVM 设备的使用寿命.此外,目前存在的大部分迁移策略没有关注分析算法的时间复杂度,这也将产生额外的系统消耗.

为了进一步消除海量大数据处理带来的冗余页面迁移操作,减少 PCM 写入次数,降低访问时延并提高系统性能,本文进而通过增加新的设备并改进以往基于双向散列链表的迁移算法,提出了基于 DRAM 牺牲 Cache 的异构内存页迁移机制.

2 基于牺牲 Cache 的异构内存页迁移机制

在异构内存系统中,由于 NVM 写入操作表现出的低性能问题,在访问速度较慢的 NVM 和访问

速度较快的 DRAM 之间进行内存页面迁移显得尤为重要.针对访问时延较大的 PCM,为了消除不必要的迁移操作,减少 PCM 的写入并缩短访问的时延,本文提出了 VC-HMM,其中包括 3 个部分:基于 DVC(DRAM-based victim Cache)的异构内存系统结构、适用于 VC-HMM 的页面迁移策略以及迁移条件在不同工作负载下的自适应算法.

2.1 基于 DRAM 牺牲 Cache 的异构内存系统结构

图 2 展示了 VC-HMM 的整体结构.与传统的内存结构不同的是在 DRAM 与 PCM 之间添加了一个小容量的 Cache 作为 DRAM 主存部分的牺牲 Cache,牺牲 Cache 由随机动态存储器 DRAM 构成,并用来存储从主存 DRAM 中被逐出的"冷"页.DVC 使用访问速度较快且页面替换便利的直接映射.与垂直结构的异构内存系统中使用的 DRAM Cache 类似,DVC 中的每一页都有一个 1 b 的脏页标志来标记页面是否被修改.当 DRAM 逐出页面时,将根据页面的物理地址存储在 Cache 对应的块中.如果存储时产生冲突,则原本在块中的页面将被迁回 PCM.由于在 Cache 中的页面原本就是从 DRAM 中淘汰的"冷"页,所以就算冲突后被直接写回也能一定程度上降低再次被迁移的概率.

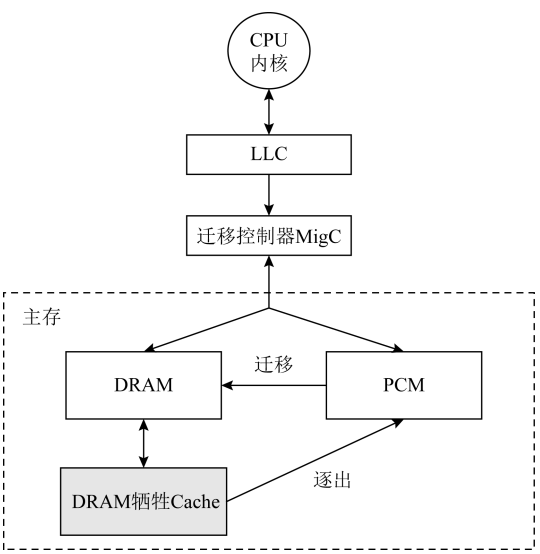

Fig. 2 DRAM-based victim Cache on hybrid main memory architecture

图 2 基于 DRAM 牺牲 Cache 的异构内存系统结构

当一个页面被判断为"热"页时,迁移控制器(migration controller, MigC)将执行迁移操作,从 PCM 迁移的页面将被拷贝到 DRAM 的空页框中,而 PCM 将继续保留此页的数据以便将来的迁回写入.通过这种方法,读操作密集的页面即使经历了迁移和迁回也不必写回 PCM,减少了 PCM 的写操作.为了实现 PCM 中原数据的保留,在 MigC 中添加了用来监视迁移页面状态的迁移页面列表(migrated page list, MPL)用于判断所请求的页面的位置.如果所请求的页面已经迁移到 DRAM 中,则 MigC 将重映射目的地址使请求进行正确访问;而如果所请求页面经过二次迁移存在于 DVC 中,请求将通过原本的目的地址在牺牲 Cache 中查找.MPL 会同时通过记录迁移的页面是否进行过写操作,从而在最后逐出 DVC 时进行写回判断,只有脏页面需要进行写回,对于其余的页面只需要删除相关的节点信息即可.

迁移控制器 MigC 基于 THMigrator 进行了改进.在 THMigrator 迁移机制中,页面节点的记录排序通过双向散列链表来完成.双向散列链表的结构能够直接快速的访问每个节点并完成增删改查的工作,这大大降低了算法的复杂度,加快了迁移的速度.每当有页面被访问时,该页的相关信息将被记录为节点插入对应表的表头,当表中页面再一次被访问时,该节点将重新被插入表头,即按照最近最少使用递减排序.通过这种排列方式进行页面搜索时能够把搜索范围控制在较小的区域内.另外,双向散列链表的结构能够使使用更加频繁的页面更早的执行迁移,而不是以达到迁移标准的时间来作为迁移准则,这将减少不必要的迁移次数.例如,页面 A 比页面 B 先达到迁移阈值进入候选迁移列表(migration candidate list, MCL),但是页面 A 此后再也没有被访问到,这时页面 B 被频繁使用,则 MCL 将把页面 B 的记录节点放在表头,从而比页面 A 更快地进行迁移.

在 VC-HMM 中双向散列链表结构依旧被使用在页面排列中.为了更好地适应 DVC 的结构,在迁移控制器中设置了 3 个双向散列链表:PCM 页面列表(PCM page list, PPL)、候选迁移列表(MCL)以及迁移页面列表(MPL).PPL 记录在 PCM 中的页面访问情况,PPL 与 MCL 中的节点结构如表 2 所示.PPL 记录 PCM 中页面的"冷""热"程度、页面编号以及生存周期.内存中所有的页面将根据其物理地址计算得出一个唯一的页号作为在表中的索引,MigC 能够根据页号直接访问页面的信息而不需要遍历查找.当一个在 PCM 中的页面首次被访问时,MigC 将该页面的信息初始化并插入 PPL 的表头,且每次访问都会将之重新插入表头."冷""热"程度将根据访问的次数逐次递增,生存周期也将随着每一次的访问被重置.生存周期用于控制一个页面存

在于列表中的时间,如果一个页面长时间不被访问,则当生存周期用尽,该页面节点将被删除.当 PPL 中的页面达到迁移阈值时,该页面节点将被插入迁移候选列表 MCL 的表头,MCL 将在页面下一次被访问时执行迁移.当有迁移还未完成时,MCL 也能够将页面按照"冷""热"程度排序使更"热"的页面优先迁移.和 PPL 相同,在 MCL 中的页面也将根据访问情况更新节点信息,并且按顺序执行迁移.每次有请求进入内存时,MigC 都将检查是否有页面达到迁移阈值或超过生命周期,超过生命周期的页面将被作为"冷"页从列表中删除以减少冗余的迁移操作.

Table 2 Page Node Structure of PPL and MCL

表 2 PCM 页面列表与候选迁移列表的页面节点结构

成员条目	说明
PageNumber	每一页拥有的唯一页号
HotnessValue	页面"冷""热"程度
LifeTime	页面生命周期

执行了迁移而进入 DRAM 的页面将被作为新的节点插入迁移页面列表(MPL)的表头,并删除在 MCL 中的旧节点,MPL 中的节点结构如表 3 所示.MPL 在 PPL 和 MCL 的基础上记录更多的页面信息帮助迁移机制的实现,1b 标签参数 IsInDRAM 记录了页面存在的位置,页面地址的映射被记录用于请求的正确访问,另外 MPL 将记录页面在 DRAM 中的读写操作次数来帮助迁移参数的动态自适应.MPL 也将随着页面的访问更新节点从而进行排序.当进行"冷"页判断时,MigC 将逆序遍历 MPL 从而查找需要进行迁移的"冷"页.

Table 3 Page Node Structure of MPL

表 3 迁移页面列表的页面节点结构

成员条目	说明
PageNumber	每一页拥有的唯一页号
HotnessValue	页面"冷""热"程度
LifeTime	页面生命周期
OSVisiblePA	未迁移时页面的初始物理地址
RealPA	迁移后页面新的物理地址
IsInDRAM	标志是否在 DRAM 中
NumberOfRead	页面读操作次数
NumberOfWrite	页面写操作次数

2.2 页面迁移与访问

VC-HMM 的迁移框架如图 3 所示,具体迁移步骤为:

① MigC 判断所访问内存页是否存在于某个表中.如果在表中,则更新节点信息并将该节点置于表头;如果不在表中,将页面节点初始化并插入 PCM 页面列表 PPL 的表头.

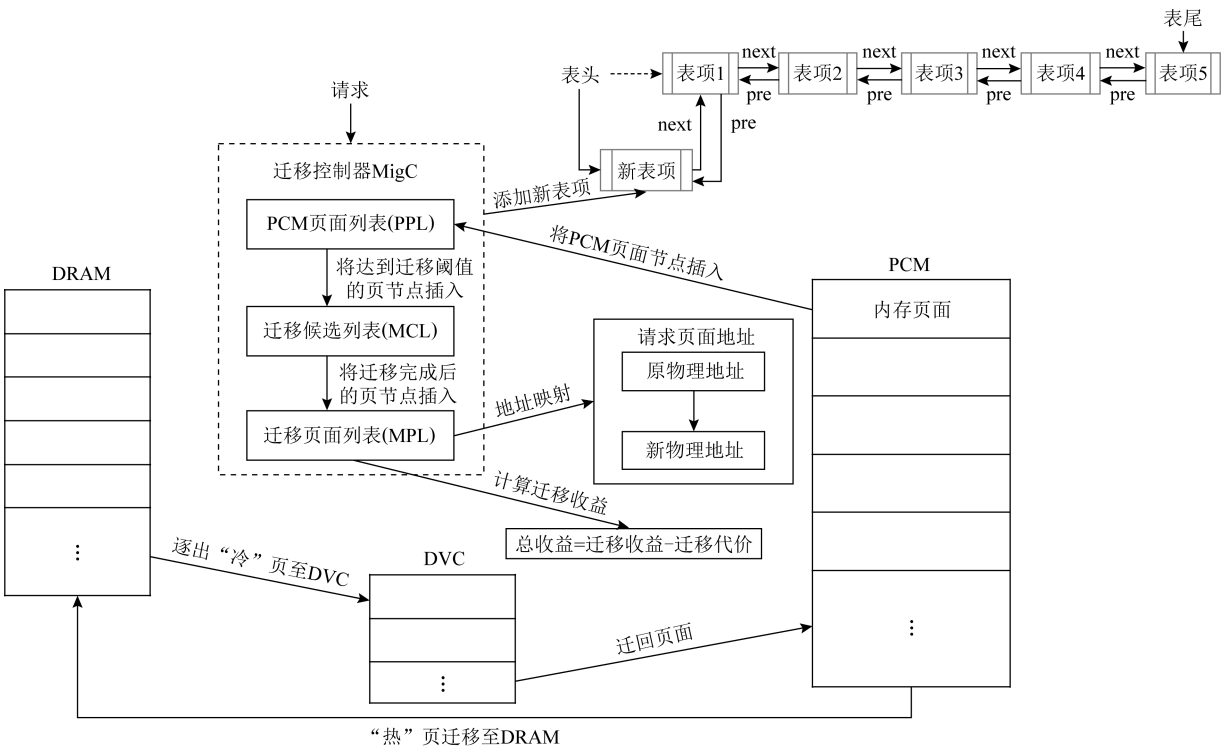

Fig. 3 Migration framework for VC-HMM

图 3 VC-HMM 迁移框架

② 如果所访问页面节点在迁移页面列表 MPL 中,则根据标志位判断该页在 DRAM 还是 DVC 中,并执行地址重映射.如果所访问页面节点在候选迁移列表 MCL 中,则检查是否有迁移正在进行,若有迁徙正在执行则等待当前迁移完成否则进行迁移操作.如果所访问页面节点在 PPL 中,则判断是否达到迁移阈值,若达到迁移阈值则将节点插入 MCL 表头.

③ 在 MCL 中按顺序进行迁移,将该页复制到 DRAM 的空页框中,将该页在 MCL 中的节点信息更新到 MPL 中并删除原节点,初始化新节点信息并置为表头.

④ 在 MPL 中从后往前检索是否有满足迁出条件的 DRAM 页面,若有满足迁出条件的 DRAM 页面,检查是否有 DRAM 到 DVC 的迁出操作正在进行,若没有正在进行的迁出操作,则进行迁移操作并更新 MPL 中对应节点信息,将对应 DRAM 页面重新置为空.若当前有迁移操作正在进行,则等待当前迁移操作结束.

⑤ 在 MPL 中从后往前检索是否有满足迁出条件的 DVC 页面,若有满足迁出条件的 DVC 页面,检查是否有 DVC 到 PCM 的页面逐出正在进行,若当前有迁移操作正在进行,则等待当前逐出结束.若没有正在进行的迁移操作,则通过 MPL 中该节点记录的读写情况判断是否需要写回 PCM 中,若不需要写回 PCM 中,则直接删除节点.最后根据该页面在 DRAM 中的读写情况计算迁移收益,若收益为正,则更新迁移参数使页面更容易迁入 DRAM 且页面的生命周期变长;若收益为负,则更新迁移条件使页面更难迁入 DRAM 且页面的生命周期变短.

⑥ 检查 PPL 和 MCL 是否有超过生命周期的页面节点,将之逐出.

迁移流程框图如图 4 所示:

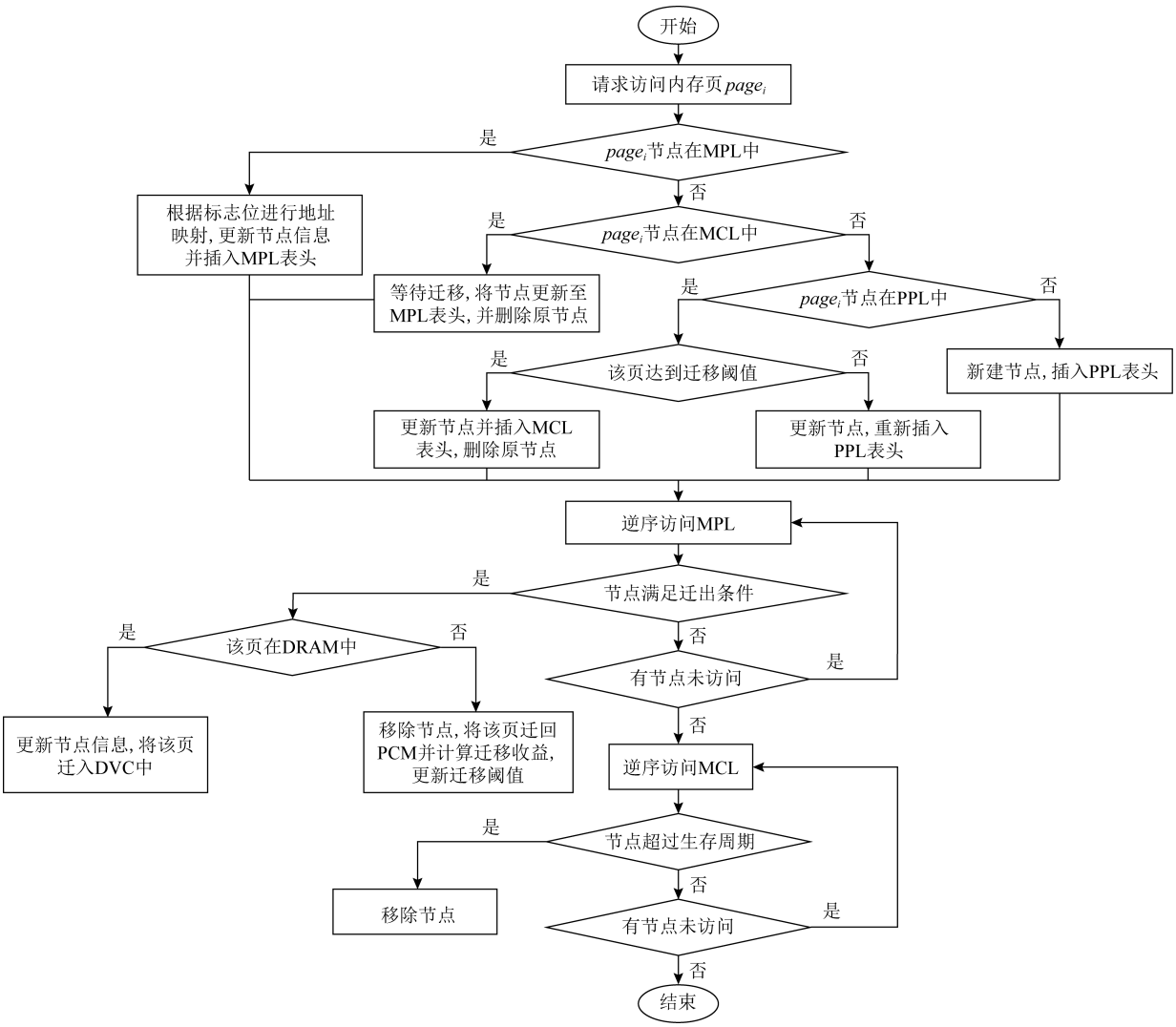

Fig. 4 Process of migration

图 4 迁移流程

页面的"冷""热"程度根据访问次数递增.迁移时,MigC 将为迁移的页面寻找 DRAM 中的空页框,如果没有可用的空页框,MigC 将在 MPL 中逆序查找并逐出最"冷"的页面到 DVC 中.基于 DRAM 牺牲 Cache 的内存页迁移机制的伪代码实现方法如算法 1 所示:

算法 1. 内存页面迁移算法.

输入:当前访问页面的页号 $page_i$、页面所在内存设备通道编号 $ChannelNumber$.

① 初始化 PPL,MCL,MPL,$Threshold$,$currentTime$;
② $Request(page_i)$;
③ if $IsLocatedInPPL(page_i)$ then
④ $UpdatePPL(page_i)$; /* 更新页面的"冷""热"程度值并重置生命周期 */
⑤ $MoveToHead(page_i)$; /* 将该页节点置为表头 */
⑥ if $page_i \rightarrow value > Threshold$ then /* "冷""热"程度值达到阈值则插入候选迁移列表表头 */
⑦ $InsertMCL(page_i)$;
⑧ $RemovePPL(page_i)$;
⑨ end if
⑩ end if
⑪ if $IsLocatedInMCL(page_i)$ then
⑫ if $LifeTime(page_i) > currentTime$ then /* 在候选迁移列表中且没有超过生命周期则进行迁移 */
⑬ $StartMigrate(page_i)$;
⑭ $RemoveMCL(page_i)$;
⑮ $InsertMPL(page_i)$; /* 从 PCM 迁移到 DRAM 的页面插入 MPL 中 */
⑯ else
⑰ $RemoveMCL(page_i)$;
⑱ end if
⑲ end if
⑳ if $IsLocatedInMPL(page_i)$ then
㉑ if $page_i \rightarrow IsInDRAM ==$ true then /* 被访问页面位于 DRAM 中 */
㉒ $Remap(page_i)$; /* 重映射地址以正确访问页面 */
㉓ $UpdateMPL(page_i)$;
㉔ $MoveToHead(page_i)$;
㉕ end if
㉖ end if
㉗ if $ChannelNumber \neq 0$ && !$IsLocatedInMPL(page_i)$ then
㉘ $InsertPPL(page_i)$; /* 第 1 次访问到的 PCM 页面插入 PPL 表表头 */
㉙ end if

与迁移操作仅针对当前所访问的页面不同,迁回和逐出操作针对每一个完成迁移的页面,可能在每一个时钟周期中发生.MigC 将在迁移判断操作结束之后分别查看 MPL 和 MCL 列表.在 MCL 中通过从后往前逆序遍历查看各页面节点是否已超过生命周期,超过生命周期的节点将被直接删除.类似地,在 MPL 中逆序查找需要迁移的 DRAM 和 DVC "冷"页,并进行迁移和迁回操作,随后针对迁回 PCM 的页面进行迁移收益核算.内存页面逐出的伪代码实现方法如算法 2 所示.通过 MPL 执行的逐出判断将在分别寻找到满足要求的一个 DRAM 和 DVC 页面后结束本次遍历.由于逐出操作面向的对象为"冷"页,故本算法中仅将生命周期作为判断"冷""热"程度的标准以简化算法的执行.

当所访问的页面已经完成迁移时,MigC 将通过 MPL 中保存的 1 b 标志位判断该页是在 DRAM 中还是在 DVC 中.如果 $IsInDRAM$ 为 true 则该页在 DRAM 中,MigC 将通过 MPL 记录的页面地址映射重新映射请求的目标地址;如果 $IsInDRAM$ 为 false 则该页在 DVC 中,请求将通过原始目的地址与 DVC 中的地址进行比对从而执行访问.

算法 2. 内存页面逐出算法.

输入:当前访问页面的页号 $page_j$.

① for $page_j = rbeginMPL()$; $page_j \neq rendMPL()$; $++page_j$ do /* 检查已迁移页面是否要逐出 */
② if $page_j \rightarrow expirationTime > currentTime$ then
③ if $page_j \rightarrow IsInDRAM ==$ true then
④ $VicfromDRAM(page_j)$;
⑤ else
⑥ $Revenue(page_j)$;
⑦ $VicfromCache(page_j)$;
⑧ $AdjustThreshold()$;
⑨ end if
⑩ end if
⑪ end for

⑫ for $page_j$=$rbeginMCL()$;
$page_j$≠$rendMCL()$;
++$page_j$ do /* 检查候选迁移列表中是否有页面超出生命周期 */
⑬ if $page_j$→$expirationTime$>$currentTime$ then
⑭ $Evictpage(page_j)$;
⑮ end if
⑯ end for

2.3 迁移参数的自适应调整

为了使迁移机制能够自适应不同的工作负载，提高迁移的合理性，MigC 将通过迁移收益自主地更新迁移参数.当一个页面最终被逐出 DVC 时，MigC 将计算这一次迁移的收益.迁移收益：

$$R_w=Write_{count}\times(Time_{w\_in\_pcm}-Time_{w\_in\_DRAM}),\quad(1)$$

$$R_r=Read_{count}\times(Time_{r\_in\_pcm}-Time_{r\_in\_DRAM}),\quad(2)$$

$$C=Time_{P\_D}+Time_{D\_DVC}+Time_{DVC\_P}\times dirty,\quad(3)$$

$$R=R_w+R_r-C.\quad(4)$$

式(1)和式(2)通过 MPL 记录的读写操作次数计算页面迁移到 DRAM 中和留存在 PCM 中的读写时延收益，$Write_{count}$表示迁移后该页面所收到的写操作次数，$Time_{w_in_pcm}$和 $Time_{w_in_DRAM}$分别表示在 PCM 和 DRAM 中进行写操作所需的时间.同理 $Read_{count}$，$Time_{r\_in\_PCM}$，$Time_{r_in_DRAM}$表示迁移后该页面所进行的读操作次数以及在 PCM 和 DRAM 中进行读操作所需的时间.

式(3)计算了迁移所需的时间代价，如果页面最终被逐出时为脏，则 $dirty=1$，反之 $dirty=0$.最终的收益通过式(4)计算，如果 $R>0$ 则表示迁移收益为正，迁移阈值将自动降低使页面更容易迁移到 DRAM 中，在 DRAM 和 DVC 中的页面的生存周期也将被延长；如果 $R<0$ 则表示迁移收益为负，迁移阈值将自动增加使页面更不容易达到迁移阈值，在 DRAM 和 DVC 中的页面的生存周期也将被缩短.

本文通过线性增值的方式自适应地调整迁移参数，设置以步长为 1 进行线性增长.生存周期的增减将应用于下一个周期执行迁移页面节点信息的初始化中，而不是更新目前迁移表中所有内存页的生存周期.实验中参数的初始数值与 THMigrator 的模型参数一致.

例如，当页面 x 被认定为“冷”页面，从而从 DVC 迁移回 PCM 时，MigC 通过计算得出该页面 x 迁移的收益.若收益 R 为正，则表示当前迁移参数合理有效，留存于 DRAM 中的页面将被大概率访问.然后，MigC 降低迁移阈值，增加其他页面迁移的机会.在下一次迁移时，提升页面初始生命周期，增加页面驻留在 DRAM 中的时间.

3 实验与分析

本文采用 GEM5[27] 和 NVMain[28-29] 构建基于异构内存的多核处理器模型系统，GEM5 模拟器是一个用于计算机系统架构研究的模块化平台.NVMain 是一个架构级的主存储器模拟器.本文通过 GEM5 与 NVMain 的结合，设计了具有 DRAM 牺牲 Cache 的水平型结构的异构内存系统，其中 DRAM 通道与 PCM 通道的比例为 1∶3，DVC 的大小为 DRAM 的 1/16，采用时钟频率为 2 GHz 的 TimingSimpleCPU，一级缓存设置为 32 KB，二级缓存设置为 256 KB.详细配置如表 4 所示，内存结构中的 1∶1/16∶3 为 DRAM 容量、DVC 容量以及 PCM 容量的比例.

Table 4 Main Configurations of GEM5 and NVMain
表 4 GEM5 和 NVMain 模拟器的主要配置

配置参数	参数值
CPU 类型	Timing Simple CPU
CPU 时钟频率/GHz	2.0
一级 Cache/KB	32
二级 Cache/KB	256
总线宽度/b	64
内存类型	NVMain Memory
内存结构	DRAM+DRAM victimCache+PCM(1∶1/16∶3)
运行模式	SE Mode

为了对 VC-HMM 进行全面分析，本文选择了 CoinMigrator、多级队列替换算法(MQRA)和基于双向散列链表迁移机制(THMigrator)作为参考.CoinMigrator 迁移策略为 NVMain 模拟器自带的随机迁移策略，每一次的迁移决策由随机数决定.MQMigrator 迁移策略为 Zhou 等人[16] 提出的页面替换算法，迁移使用多个队列来维护页面的访问信息，通过队列优先级进行“冷”“热”页面的排序和选择，但由于队列数量多，且不能随机访问每个页面节点.因此，本文提出的 VC-HMM 采用双向散列链表结构维护页面信息的数据结构，能实现页面节点的随机访问而不需要遍历链表，降低了页面信息维护的复杂度.本文采用了 SPECCPU2006[30] 中的 10 个

不同的基准应用程序,并将 CoinMigrator 的数据作为对应的标准化数据.

针对一个内存页的读写配置与能耗配置如表 5 所示,参数来源于文献[18],内存页大小设置为 4 KB.

Table 5 Configuration of the Memory for a 4 KB Memory Page

表 5 针对一个 4 KB 内存页的内存参数配置

内存	读延迟时间/μs	写延迟时间/μs	读能耗/nJ	写能耗/nJ
DRAM	3	3	3 051	3 815
PCM	3	64	4 577	22 880

实验数据通过 GEM5 和 NVMain 模拟器自身集成的 Trace 工具来产生,Trace 工具可以以文本的形式输出程序运行时的日志信息,在程序代码中通过添加一些关键数据的打印操作,使用 AddStat 方法可以将运行时的数据输出到日志信息中,通过这种输出日志的跟踪方法可以获得读写操作的数量和分布以及迁移内存页的数量等信息.实验中用于测试的基准应用均使用了 1 000 万条指令的运行.其他迁移参数的设置与前期研究[18]的配置一致.

3.1 访问时延

实验数据显示,通常情况下使用 VC-HMM 的系统的平均访问延迟比使用 CoinMigrator,MQRA,THMigrator 的系统的平均访问延迟分别低 22.72%,31.56%,23.63%,如图 5 所示.基准中的每个应用在 VC-HMM 的使用中都有较好的效果.其中,与 CoinMigrator,MQRA,THMigrator 相比,应用 sjeng 在使用 VC-HMM 时访问时延的降低最明显,分别为 53.45%,60.58%,60.47%.图 5 中应用 omnetpp 的访问时延高于 CoinMigrator,这是由于 CoinMigrator 的随机概率判断策略使得迁移不稳定,每一次的迁移并不完全一样.本文经过多次的实验,使数据尽可

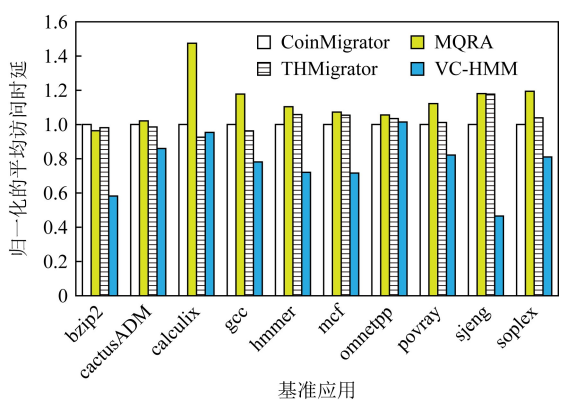

Fig. 5 Normalized average access latency

图 5 归一化的平均访问时延

能趋于稳定.目前 VC-HMM 的总体访问时延相比其他的迁移策略基本上都有明显降低.即使对于所产生的访问时延高于 CoinMigrator 策略,VC-HMM 策略的访问时延也仅高于 1.5%.

VC-HMM 的主要性能提升源于 DRAM 牺牲 Cache,DVC 将更多的页面保留在读写速度较快的 DRAM 部分.即使一个页面在迁移参数不精准的情况下进行迁移,该页面仍然可以在访问较快的 DRAM 中停留更长的时间,而不会由于"冷""热"度的快速动态变化而在 PCM 和 DRAM 之间频繁迁移.这使得更多的请求通过访问时延较短的 DRAM 进行访问,而不是访问读写时延长的 PCM.不必要迁移操作的减少也避免了带宽和其他系统资源的占用.实际上 VC-HMM 通过使已经执行的迁移变得更有效而不是单纯调整迁移阈值来消除无效迁移.

3.2 PCM 写操作频次

通过使用添加了 DVC 的异构内存系统,PCM 访问的次数也有了相应的减少.如图 6 所示,使用 VC-HMM 的系统中,PCM 的访问次数相比使用 CoinMigrator,MQRA,THMigrator 的系统平均减少了 67.38%,73.26%,62.97%.同时,对 PCM 的写操作数也明显减少.如图 7 所示,相比使用 CoinMigrator,MQRA,THMigrator 的系统,使用 VC-HMM 的系统平均减少了 81.28%,84.99%,75.89% 的 PCM 写入次数.

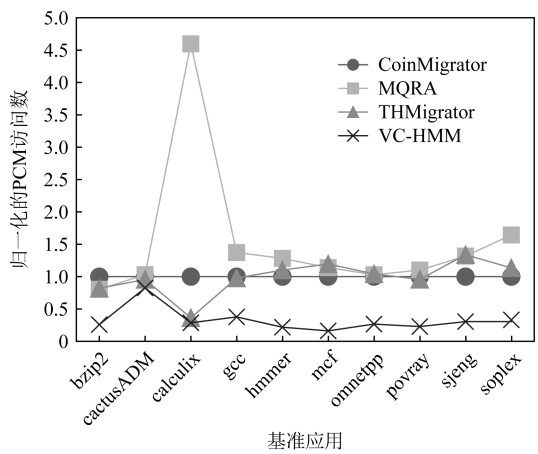

Fig. 6 Normalized PCM access times

图 6 归一化的 PCM 访问数

尽管迁移阈值可能不够精确,且由于迁移参数的人为制定无法选择和预测真正最"热"的页面进行迁移,但 VC-HMM 增加了迁移页留在 DRAM 部分的时间,使大部分的请求通过 DRAM 进行访问,冗余迁移的减少也降低了对 PCM 的读写操作次数,延长了

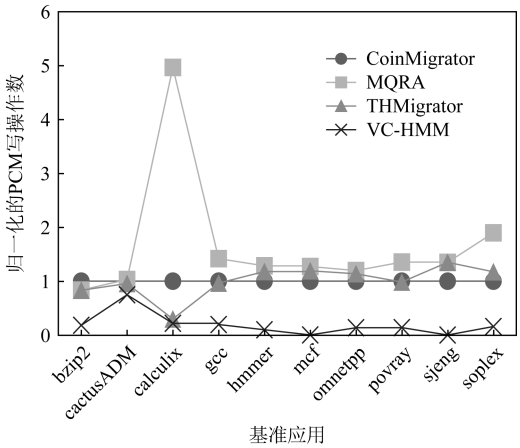

Fig. 7 Normalized PCM write times

图 7 归一化的 PCM 写操作数

PCM 的使用寿命.原页面的保留也使得读频繁的"冷"页面在迁移时不需要写回,进一步降低了 PCM 被访问的次数.

3.3 IPC

图 8 比较了使用 VC-HMM 和其他 3 种算法的系统 IPC(平均每个周期完成的指令数).这 4 种不同算法所产生的系统 IPC 差异不显著,但 VC-HMM 仍能在一定程度上提高 IPC.使用 VC-HMM 系统的 IPC 平均比使用 CoinMigrator,MQRA,THMigrator 的系统的 IPC 高 2.36%,2.88%,4.03%.在所有基准应用中,sjeng 的增长率最高,分别为 11.36%,12.72%,24.32%,说明 sjeng 应用的存储访问的局部性较高.在这种情况下,所请求的页面大部分被迁移到 DRAM 中,大大减少了请求的时间.

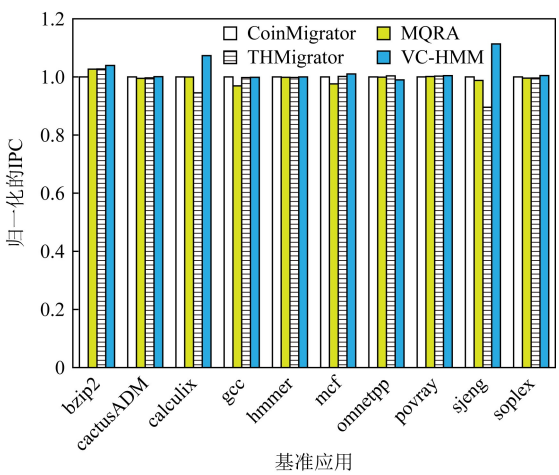

Fig. 8 Normalized IPC

图 8 归一化的 IPC

然而,即使 VC-HMM 减少了大量的 PCM 写入,使用 VC-HMM 的系统的 IPC 也没有得到显著的改善.本文分析其中一个原因是由于 VC-HMM 需要在每次迁移中进行二次迁移.迁移的页面需要从 DRAM 到 DRAM 牺牲 Cache 进行额外迁移,即使 MigC 可以自主地调整迁移参数,但二次迁移依旧需要额外的时钟周期来完成.

此外,不同应用程序的写请求频次多少、访存地址的局部性强弱会使得 VC-HMM 机制下的 IPC 值发生变化.写访问量较少且局部性较好的应用在使用 VC-HMM 时能够得到更好的性能提升.

3.4 冗余迁移操作

图 9 比较了使用 VC-HMM 的系统与使用另外 3 种算法的系统的重复迁移操作次数,与使用 CoinMigrator,MQRA,THMigrator 的系统相比,VC-HMM 平均减少了 58.80%,52.40%,38.37%的重复迁移操作.其中在基准测试应用 calculix 和 sjeng 中,均未检测到重复的内存页迁移操作.

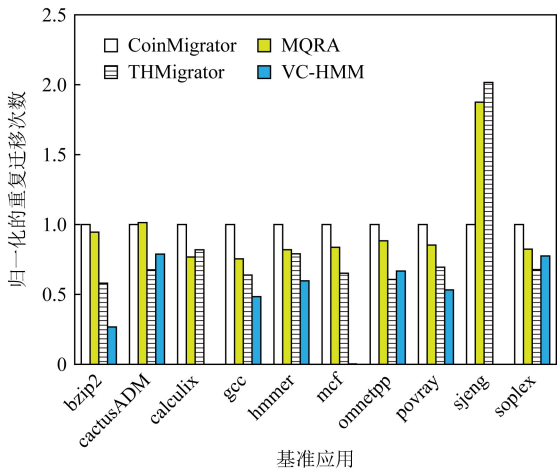

Fig. 9 Normalized re-migration times

图 9 归一化的重复迁移次数

由于当前实验阶段难以计算冗余操作中的 σ 值,本文实验采用统计重复迁移操作的次数来量化冗余迁移操作的次数.在 cactusADM 及 omnetpp 应用中,VC-HMM 的重复迁移次数高于 THMingrator,这是由于 VC-HMM 拥有迁移阈值和页面生存周期的自适应策略,对于局部性较差的应用来说,由于 VC-HMM 相比没有 DVC 设备的策略需要额外的从 DRAM 到 DVC 设备的迁移,这将会导致动态自适应算法减少页面在 DRAM 中的生存周期,从而使得页面过早迁出,增加了冗余迁移的可能性.同时,DVC 的直接映射策略也使得冲突页面过早地迁回 NVM,也增加了冗余迁移的次数.

DVC 使得完成迁移的“热”页更久地保存在访问速度较快的 DRAM 设备中,且“冷”“热”页并不成对进行迁移操作,这阻止了“冷”页的误迁回,从而减少了页面再次变“热”而导致的冗余迁移问题.在构建 DVC 时,本文使用直接映射策略作为 DVC 的映射规则,虽然直接映射策略下的访问速度较快,但也更加容易引起冲突而加快页面的迁回,导致部分冗余的迁移操作.由于迁移到 DVC 的页面为从主存 DRAM 中逐出的“冷”页.因此,虽然存在由于冲突而过早迁回的页面,但并没有对系统产生过大的影响.

3.5 系统能耗

图 10 比较了使用 VC-HMM 的系统与使用另外 3 种算法的系统的能耗情况.与使用 CoinMigrator,MQRA,THMigrator 的系统相比,VC-HMM 平均减少了 2.74%,3.16%,4.29%的系统能耗.由于系统性能的提升,VC-HMM 在降低运行时间、提高系统效率的同时减少了能耗.但同时,能耗将根据使用的 DVC 的容量大小而改变.图 11 对比了 DVC 容量为 DRAM 容量的 1/8 时所产生的能耗.与使用 CoinMigrator,MQRA,THMigrator 的系统相比,使用比先前增加 1 倍容量的 DVC 时,VC-HMM 平均增加了 24.92%,24.35%,22.97%的系统能耗.VC-HMM 在应用 hmmer 与 omnetpp 上的能耗大于其他的迁移策略.对于局部性较差的应用来说,迁移的页面也会较多,从而导致内存访问和静态维持数据所需的能耗增加,且额外的 DRAM 到 DVC 的迁移操作和 DVC 冲突问题也将增加能耗.此外,DVC 容量的增大将减少数据冲突的发生,导致所需保持的数据增加,这些数据的访问以及达到生存周期之前的数据刷新都将产生大量能耗.

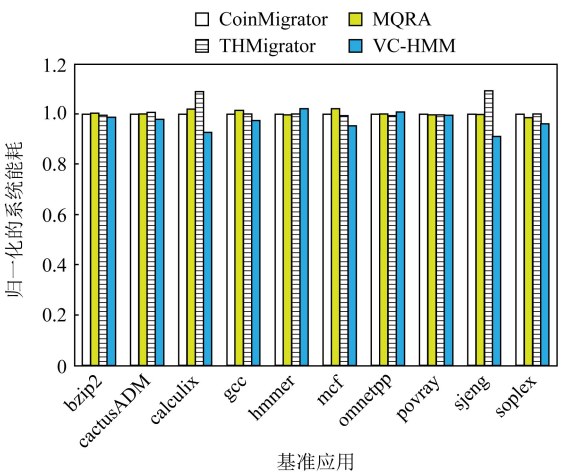

Fig. 10 Normalized energy consumption where the capacity of DVC is 1/16 of DRAM

图 10 DVC 容量为 DRAM 容量 1/16 时归一化的系统能耗

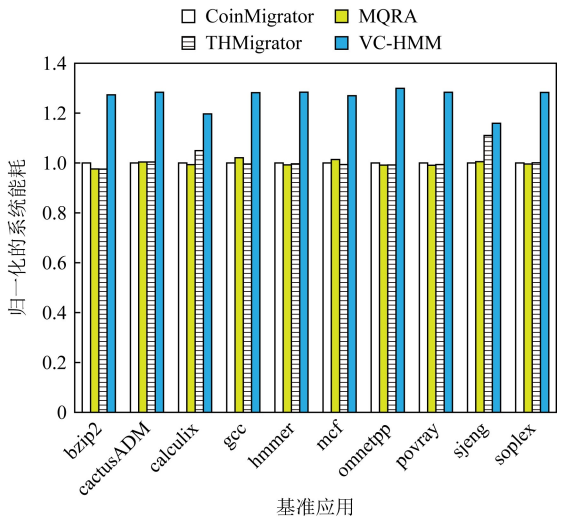

Fig. 11 Normalized energy consumption where the capacity of DVC is 1/8 of DRAM

图 11 DVC 容量为 DRAM 容量 1/8 时归一化的系统能耗

由于 DRAM 利用电容内存储电荷的多少来存储数据的特性,晶体管漏电使得 DRAM 需要周期性地充电,且访问前需要对区块进行预充,这将产生不小的系统能耗来维持 DRAM 的运行.过大的 DVC 将使得静态能耗增加,且得不到合理的使用.另外,由于异构内存的迁移使得大量的内存访问集中于 DRAM,所以 DVC 的容量大小将不同程度上影响系统能耗.

4 总 结

本文提出了一种基于 DRAM 牺牲 Cache 的异构内存页迁移机制 VC-HMM,该机制实现了比 THMigrator 模型更低的访问延迟,消除了部分无效的迁移操作.此外,VC-HMM 在局部性较强的应用中取得了较高的访存性能.

实验结果表明:与使用 CoinMigrator、多队列算法 MQRA 和基于双向散列链表的迁移机制 THMigrator 的系统相比,使用 VC-HMM 的系统的平均访问延迟降低了 22.72%,31.56%,23.63%.同时,VC-HMM 相比 CoinMigrator,MQRA,THMigrator 减少了 81.28%,84.99%,75.89%的 PCM 写操作,

并在一定程度上提升了系统 IPC.另外,在减少重复迁移操作方面,VC-HMM 平均减少了相比另外 3 种机制 58.80%,52.40%,38.37%的重复迁移操作.在能耗方面,与使用 CoinMigrator,MQRA 和 THMigrator 的系统相比,VC-HMM 平均减少了 2.74%,3.16%,4.29%的系统能耗.未来,我们计划利用机器学习算法识别和监测"热"页面,并着力于研究更智能的页迁移决策,以获得更高的异构内存综合性能.

作者贡献声明:裴颂文提出了算法思路和实验方案;钱艺幻负责完成实验并撰写论文;叶笑春提出指导意见并修改论文;刘海坤与孔令和提出指导意见.

参 考 文 献

[1] Mutlu O, Subramanian L. Research problems and opportunities in memory systems [J]. Supercomputing Frontiers and Innovations: An International Journal, 2014, 1(3): 19-55

[2] Mittal S. A survey of architectural techniques for DRAM power management [J]. International Journal of High Performance Systems Architecture, 2012, 4(2): 110-119

[3] Vetter J S, Mittal S. Opportunities for nonvolatile memory systems in extreme-scale high-performance computing [J]. Computing in Science & Engineering, 2015, 17(2): 73-82

[4] Shu Jiwu, Lu Youyou, Zhang Jiacheng, et al. Research progress on non-volatile memory based storage system [J]. Science & Technology Review, 2016, 34(14): 86-94 (in Chinese)
(舒继武, 陆游游, 张佳程, 等. 基于非易失性存储器的存储系统技术研究进展[J]. 科技导报, 2016, 34(14): 86-94)

[5] Raoux S, Xiong Feng, Wuttig M, et al. Phase change materials and phase change memory [J]. MRS Bulletin, 2014, 39(8): 703-710

[6] Burr G W, Brightsky M J, Sebastian A, et al. Recent progress in phase-change memory technology [J]. IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 2016, 6(2): 146-162

[7] Cappelletti P. Non-volatile memory evolution and revolution [C] //Proc of IEEE Int Electron Devices Meeting. Piscataway, NJ: IEEE, 2015: 10.1.1-10.1.4

[8] Pimo E, Ashok V, Logeswaran T, et al. A comparative performance analysis of phase change memory as main memory and DRAM [J/OL]. Materials Today: Proceedings, 2021 [2021-03-12]. https://www.sciencedirect.com/science/article/pii/S2214785321005642

[9] Mittal S, Vetter J S. A survey of software techniques for using non-volatile memories for storage and main memory systems [J]. IEEE Transactions on Parallel & Distributed Systems, 2016, 27(5): 1537-1550

[10] Park H, Yoo S, Lee S. Power management of hybrid DRAM/PRAM based main memory [C] //Proc of the 48th Design Automation Conf. New York: ACM, 2011: 59-64

[11] Chen An. A review of emerging non-volatile memory (NVM) technologies and applications [J]. Solid-State Electronics, 2016, 125: 25-38

[12] Chen Ji, Liu Haikun, Wang Xiaoyuan, et al. Largepages supported hierarchical DRAM-NVM hybrid memory systems [J]. Journal of Computer Research and Development, 2018, 55(9): 2050-2065 (in Chinese)
(陈吉, 刘海坤, 王孝远, 等. 一种支持大页的层次化 DRAM-NVM 混合内存系统[J]. 计算机研究与发展, 2018, 55(9): 2050-2065)

[13] Xiao Renzhi, Feng Dan, Hu Yuchong, et al. A survey of data consistency research for non-volatile memory [J]. Journal of Computer Research and Development, 2020, 57(1): 85-101 (in Chinese)
(肖仁智, 冯丹, 胡燏翀, 等. 面向非易失内存的数据一致性研究综述[J]. 计算机研究与发展, 2020, 57(1): 85-101)

[14] Zhang Ming, Hua Yu, Liu Lurong, et al. A write-optimized re-computation scheme for non-volatile memory [J]. Journal of Computer Research and Development, 2020, 57(2): 243-256 (in Chinese)
(张铭, 华宇, 刘璐荣, 等. 面向非易失内存写优化的重计算方法[J]. 计算机研究与发展, 2020, 57(2): 243-256)

[15] Wang Xiaoyuan, Liao Xiaofei, Liu Haikun, et al. Big data oriented hybrid memory systems [J]. Big Data Research, 2018 (4): 15-34 (in Chinese)
(王孝远, 廖小飞, 刘海坤, 等. 面向大数据的异构内存系统[J]. 大数据, 2018 (4): 15-34)

[16] Zhou Yuanyuan, James Philbin, Li Kai. The multi-queue replacement algorithm for second level buffer caches [C] //Proc of USENIX Annual Technical Conf. Berkeley, CA: USENIX Association, 2001: 91-104

[17] Lee S, Bahn 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

[18] Pei Songwen, Ji Yanfei, Shen Tianma, et al. Study on migration mechanism of heterogeneous memory pages using two-way Hash chain list [J]. SCIENTIA SINICA Informationis, 2019, 49(9): 1138-1158 (in Chinese)
(裴颂文, 姬燕飞, 沈天马, 等. 基于双向哈希链表的异构内存页迁移机制[J]. 中国科学: 信息科学, 2019, 49(9): 1138-1158)

[19] Kim S, Hwang S H, Kwak J W. Adaptive-classification CLOCK: Page replacement policy based on read/write access pattern for hybrid DRAM and PCM main memory [J]. Microprocessors and Microsystems, 2018, 57: 65-75

[20] Seok H, Park Y, Park K W, et al. Efficient page caching algorithm with prediction and migration for a hybrid main memory [J]. ACM SIGAPP Applied Computing Review, 2011, 11(4): 38-48

[21] Tan Yujuan, Wang Baiping, Yan Zhichao, et al. UIMigrate: Adaptive data migration for hybrid non-volatile memory systems [C] //Proc of Design, Automation & Test in Europe Conference & Exhibition. Piscataway, NJ: IEEE, 2019: 860-865

[22] Tan Yujuan, Wang Baiping, Yan Zhichao, et al. APMigration: Improving performance of hybrid memory performance via an adaptive page migration method [J]. IEEE Transactions on Parallel and Distributed Systems, 2020, 31(2): 266-278

[23] Ryoo J H, John L K, Basu A. A case for granularity aware page migration [C] //Proc of Int Conf on Supercomputing. New York: ACM, 2018: 352-362

[24] Zhan Jinyu, Zhang Yiming, Jiang Wei, et al. Energy-aware page replacement and consistency guarantee for hybrid NVM-DRAM memory systems [J]. Journal of Systems Architecture, 2018, 89: 60-72

[25] Guo Yuhua, Xiao Weijun, Liu Qing, et al. A cost-effective and energy-efficient architecture for die-stacked DRAM/NVM memory systems [C/OL] //Proc of the 37th Int Performance Computing and Communications Conf. Piscataway, NJ: IEEE, 2018 [2021-03-12]. https://ieeexplore.ieee.org/document/8711335

[26] Islam M, Adavally S, Scrbak M, et al. On-the-fly page migration and address reconciliation for heterogeneous memory systems [J]. ACM Journal on Emerging Technologies in Computing Systems, 2020, 16(1): 10:1-10:27

[27] Binkert N, Beckmann B, Black G, et al. The GEM5 simulator [J]. ACM SIGARCH Computer Architecture News, 2011, 39(2): 1-7

[28] Poremba M, Xie Yuan. NVMain: An architectural-level main memory simulator for emerging non-volatile memories [C] //Proc of IEEE Computer Society Annual Symp on VLSI. Piscataway, NJ: IEEE, 2012: 392-397

[29] Poremba M, Zhang Tao, Xie Yuan. NVMain 2.0: Architectural simulator to model (non-) volatile memory systems [J]. IEEE Computer Architecture Letters, 2015, 14(2): 140-143

[30] John L H. SPEC CPU2006 benchmark descriptions [J]. ACM SIGARCH Computer Architecture News, 2006, 34(4): 1-17

Pei Songwen, born in 1981. Received his PhD degree in computer architecture from Fudan University. Professor and PhD supervisor with the University of Shanghai for Science and Technology, Shanghai, China. Senior member of CCF, ACM and IEEE. His main research interests include intelligent computing, heterogeneous multicore system, cloud computing, and big data.

裴颂文,1981 年生.获复旦大学计算机体系结构专业博士学位,上海理工大学教授、博士生导师.CCF,ACM,IEEE 高级会员.主要研究方向为智能计算、异构多核系统、云计算和大数据.

Qian Yihuan, born in 1997. Master candidate. Her main research interests include memory migration in heterogeneous system, machine learning, artificial intelligence.

钱艺幻,1997 年生.硕士研究生.主要研究方向为异构内存系统、机器学习、人工智能.

Ye Xiaochun, born in 1981. PhD, professor. His main research interests include algorithm paralleling and optimizing, software simulation, and architecture for high-performance computer.

叶笑春,1981 年生.博士,研究员.主要研究方向为算法并行与优化、软件仿真和高性能计算机体系结构.

Liu Haikun, born in 1981. Professor and PhD supervisor. His main research interests include non-volatile memory technology, memory computing system, virtualization/cloud computing.

刘海坤,1981 年生.教授,博士生导师.主要研究方向为非易失性内存技术、内存计算系统、虚拟化/云计算.

Kong Linghe, born in 1983. Professor, PhD supervisor. His main research interests include Internet of things, 5G, big data, blockchain, and mobile computing.

孔令和,1983 年生.研究员,博士生导师.主要研究方向为物联网、5G、大数据、区块链和移动计算.