面向大数据的异构内存系统

王孝远，廖小飞，刘海坤，金海
华中科技大学计算机学院，湖北 武汉 430074

摘要
受限于DRAM和新型非易失性存储器（non-volatile memory，NVM）的缺陷，单纯的DRAM 或者NVM 难以满足大数据应用对内存系统容量以及功耗提出的高要求。因此如何将DRAM和NVM组合成异构内存系统并进行高效的管理、准确的评估，是当今学术界和工业界面临的主要挑战。从体系结构、系统软件、编程模型以及应用等方面对面向大数据的异构内存系统进行分析与研究，提出了一系列异构内存系统的优化方法，如层次化异构内存架、片上缓存管理、访存调度、能耗管理、虚实地址转换和面向对象的内存分配与迁移机制等，并实现了原型系统进行验证。

关键词
内存计算；异构内存；大数据；非易失性存储器

中图分类号：TP31 文献标识码：A doi: 10.11959/j.issn.2096-0271.2018037

Big data oriented hybrid memory systems

WANG Xiaoyuan, LIAO Xiaofei, LIU Haikun, JIN Hai
School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China

Abstract
Big data applications put more pressure on memory system in the aspect of capacity and power consumption. However, limited by the shortcomings of DRAM and non-volatile memory(NVM), memory consists of single medium like DRAM or NVM is not competent for the requirements of big data applications. Thus, how to effectively design, efficiently manage and accurately evaluate the hybrid memories consist of DRAM and NVM are the major challenges that the academia and industry face today. The challenges of hybrid memory systems for big data processing from the perspective of computer architecture, system software, programming model and application were analyzed, and several solutions and optimizations were correspondingly provided, such as on-chip cache management, parallel processing, memory access scheduling, energy management, virtual-to-physical address translation, object-level memory allocation and migration mechanisms. Meanwhile, a number of prototypes to validate the effectiveness and efficiency of these proposals were developed.

Key words
in-memory processing, hybrid memory, big data, non-volatile memory

1 引言

随着大数据应用的蓬勃发展，计算模式已经从传统的以计算为中心转变为以大规模数据处理为中心。大数据处理的需求不仅仅表现为要处理的数据规模大，而且要求能快速实时地响应。数据的快速增长和以数据为中心的计算模式给现有的计算机存储系统带来了新的挑战。传统以外存为主体的存储模式需要频繁地在内存和外存之间交换数据，这使得大数据处理的大部分时间都耗费在数据移动过程中，内外存之间过低的I/O带宽成了大数据处理系统的性能瓶颈。为了提高大数据处理的效率，内存计算模式应运而生。内存计算模式要求计算机系统能够提供大容量的内存，从而尽可能多地把需要处理的数据缓存在内存中，从而消除磁盘I/O的性能瓶颈。

然而，传统的动态随机访问存储器（dynamic random access memory，DRAM）因其存储能耗大、存储密度小、可扩展性有限、刷新和静态功耗高等缺点，已经无法满足应用越来越大的内存需求。近些年涌现的非易失性存储器（non-volatile memory，NVM），如相变存储器（phase change memory，PCM）、磁性随机存储器（magnetic random access memory，MRAM）等，具有可随机访问、存储密度大、可扩展性强、无刷新和闲置功耗等特点，成为替代DRAM提供大容量内存的理想存储器。但NVM具有访问时延高、写次数有限、写功耗大等缺点（见**表1**），因此直接使用NVM替代DRAM是不可取的。

为了充分利用NVM容量大和DRAM读写性能好的优势，并且最大限度地避免各种存储介质的缺陷，学术界提出使用DRAM-NVM异构内存架构，通过混合使用小容量DRAM和大容量NVM，将DRAM作为NVM内存的缓存，或者将频繁访问的数据放置于DRAM中来提升系统性能，降低NVM高访问时延的缺陷。DRAM-NVM异构内存系统的设计与优化成了研究的热点[1-9]。

目前主要有层次和平行两种异构内存架构。DRAM-NVM层次化异构内存架构如**图1（a）**所示，在这种架构下，DRAM作为大容量NVM内存的缓存，其组织形式类似于传统的片上缓存，CPU请求的数据在最后一级片上缓存缺失后，首先查找DRAM缓存，若数据在DRAM缓存中缺失，则访问NVM内存，并将缺失的数据块从NVM内存复制到DRAM缓存中。DRAM-NVM平行架构如**图1（b）**所示，在这种架构下，NVM和DRAM均作为内存使用，由操作系统统一管理。

表1 DRAM 与 NVM 的性能参数对比

参数	单元面积（F^2）	读操作时延/ns	写操作时延/ns	耐久性/次	写操作功耗/(nJ·b^{-1})	空闲功耗	数据保持/年
DRAM	6~8	<10	<10	N/A	~0.1	高	刷新
PCM	4~8	10~100	20~120	10^8~10^{12}	<1	低	>10
自旋力矩存储（STT-RAM）	16~60	2~20	5~35	10^{12}~10^{15}	1.6~5	低	>10
阻变式存储器（RRAM）	4~14	10~50	10~50	10^8~10^{10}	~0.1	低	10
铁电随机存储器（FeRAM）	15~34	5~10	5~10	10^{12}~10^{14}	<1	低	10

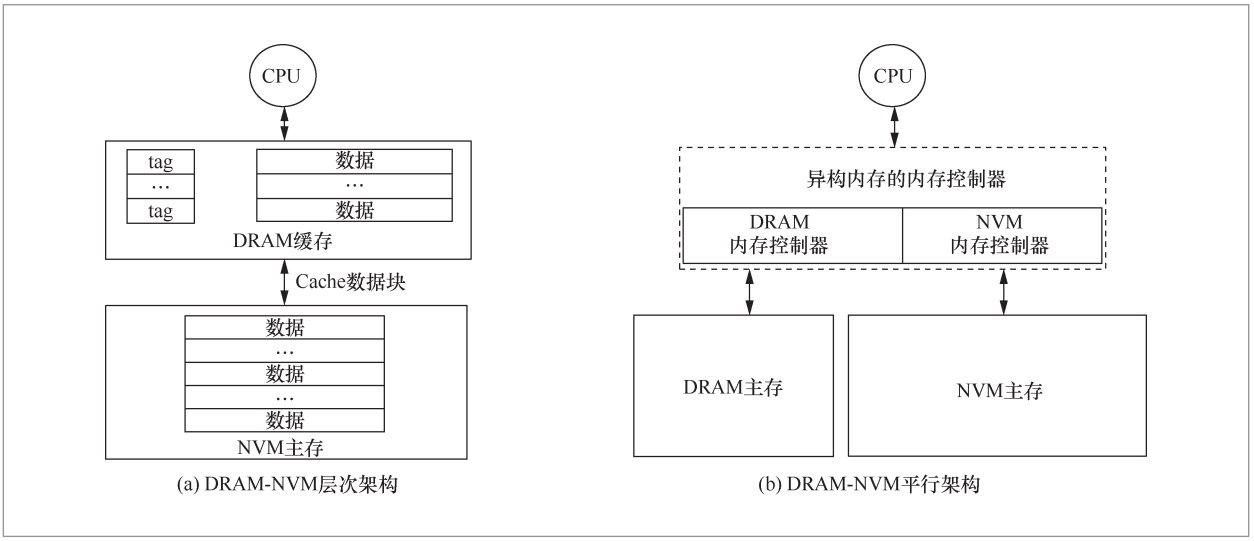


图1 异构内存架构

层次架构的异构内存系统中，DRAM等快速存储器作为NVM的缓存，缓存操作时间开销较小，但是对于局部性较差的应用，可能会引起DRAM缓存频繁换入换出，从而导致性能降低。平行架构的异构内存系统中，为了提高异构内存的性能，常用的优化策略是热页迁移[4,7,9-11]，即对应用页面信息进行监测，将频繁访问和频繁写的NVM页框迁移到DRAM中，但是迁移操作的时间开销较大，并且监测开销较大。

以上两种方案各有利弊，但是无论采取何种方案，最终目的都是尽可能同时发挥DRAM与NVM的自身优势，从而改善异构内存各方面的性能，更好地为大数据应用服务。由此可见，异构内存给现有计算机设计带来了新的挑战，这些挑战主要集中在体系结构层面、系统软件层面、编程模型层面（如**图2**所示）。

具体说来，上述3个层面主要面临以下挑战。

● 在体系结构层面，如何根据NVM和DRAM的特性以及不同应用的访存特征，设计可动态适配的存储架构，是异构内存系统设计面临的主要挑战。

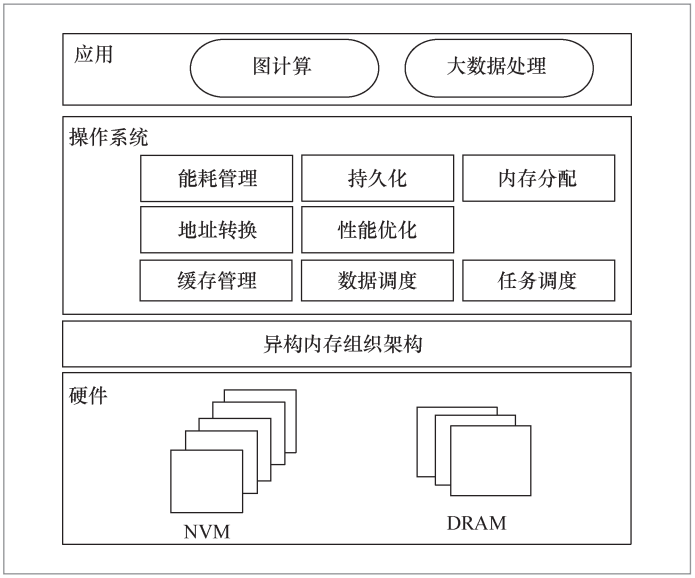


图2 异构内存研究挑战

● 在系统软件层面，由于NVM和DRAM在性能、能耗和耐久性方面的差异，异构内存资源管理面临的挑战是：如何对异构内存中不同存储介质进行差异化管理和调度；如何应对可用地址空间增大而引起的旁路转换缓冲（translation lookaside buffer，TLB）缺失率和地址转换开销增大的问题；如何根据应用的特征决

定哪类数据组织形式更适合大规模事务数据的处理；如何结合新的内存特性，设计高效、低传输量的数据可靠性保证机制。

- 在编程模型层面，异构内存给程序员带来的主要挑战是：如何设计统一的用户接口与编程抽象、执行模型与相应的编译器。

本文将概述DRAM/NVM异构内存的发展现状，总结目前异构内存的体系结构、系统软件、编程模型等层面的研究进展，并分析现有研究的优点与不足，提出针对性的解决方案。最后，以大规模图计算系统为例，给出异构内存系统针对应用层面的优化方向。

2 国内外发展现状

为了克服异构内存中各介质的缺点，并充分发挥各自的优点，现有的研究主要是集中在异构内存系统架构、异构内存管理机制、异构内存持久化和异构内存缓存管理等方面。

2.1 异构内存系统架构

为了研究NVM与DRAM组成的异构内存系统架构以及两种介质的容量配比问题，Qureshi M K等人[1]把DRAM设计成NVM的一个小容量缓存区，并利用操作系统页表管理NVM，基于此页表，文中使用了一系列的优化策略来减少对NVM的写操作次数，从而延长了NVM的使用寿命。这些策略主要有3种：延迟写机制，该机制的原理是在发生缺页的时候，首先从磁盘读入DRAM，当DRAM里的页面需要逐出的时候，根据页面是否被修改的信息来决定页面是否被写入NVM；缓存行粒度写回机制，该机制的原理是只将DRAM中被标记为脏的缓存行写回NVM中，从而减少对NVM的不必要的写操作；细粒度磨损均衡机制，该机制的原理是通过细粒度地管理逻辑扇区和物理扇区之间的映射关系，达到磨损均衡的目的。实验证明，DRAM容量与NVM容量的比在3%左右时，可以更好地利用DRAM与NVM组成的异构内存系统。当然，细粒度的映射机制也带来了较大的空间开销，tag的查询、比较也增加了系统时延，降低了系统性能。

通过将DRAM和NVM放在同一个地址空间内，Zhang W等人[12]实现了DRAM与NVM的平行架构的异构内存系统。他们的想法是以页面为粒度对应用程序的页面进行分类，将频繁修改的页面放入DRAM，将不频繁修改的页面放入NVM。具体实现机制是，首先所有的页面都加载到NVM中，随着程序的运行，对页面进行引用计数，当某个页面的计数达到一定阈值时，则会被迁移到NVM中。在进行页面计数的过程中，文章采用了多级队列（multi-queue）的管理策略：使用多个最近最少使用（least recently used，LRU）队列，每个队列代表不同的引用等级，根据页面的引用计数值对页面进行分类，引用等级越高的队列中的页面越容易被迁移到DRAM中。与此同时，也会把之前在DRAM中但具有较少引用计数值的页面回迁到NVM中。但是该方案计数和队列维护的开销巨大，页面迁移的操作也会给系统带来影响。

通过利用应用的空间局部性原理，Mladenov R[2]设计了拥有小容量DRAM缓存和大容量NVM的异构内存系统。该系统将DRAM设计成NVM的小容量缓存，对系统的访存请求的处理进行优化。对于读请求，系统优先访问DRAM缓存，如果在DRAM缓存中命中，则直接返回DRAM缓存中的数据；若DRAM缓存未命中，则接着访问NVM，并把对应的NVM数据缓存到DRAM中。对于写请求，若DRAM未满，

则直接写入DRAM缓存；若DRAM已满，则基于LRU算法进行DRAM缓存页面的替换，并把替换下来的页面写回到NVM中。为了进一步减少页面替换对性能的影响，该系统会定期检测DRAM的缓存状态，并在内存空闲时，根据LRU替换策略进行页面替换，从而预留DRAM空间用于响应后来的DRAM写请求。该系统有效地弥补了NVM访问速度慢的不足，更有利于充分发挥NVM的优势。

为了进一步提升DRAM缓存的优势，Loh G H等人[13]以一个Cache line（64 byte）的粒度对DRAM进行管理，而在DRAM与NVM之间采用组相连的方式进行映射。同时，为了降低tag查询开销对系统的影响，该系统将元数据和数据放在同一个存储阵列的行（bank row）中，从而在DRAM命中时快速访问数据，有效地缩短了DRAM命中时的串行查找开销。然而，在DRAM缓存未命中的时候，该系统需要4次访存：访问DRAM中的元数据，判断DRAM缓存命中与否；访问NVM并获取相应数据块；将获取的数据块加载到DRAM缓存中；访问DRAM缓存，获取相应数据。由此可见，该系统缺点也很明显：由于DRAM缺失代价太高，该系统不适用于局部性较差的应用；由于DRAM管理的粒度较小，tag的存储开销较大（9%左右），降低了DRAM实际可用容量；硬件改动较大，可扩展性差。

2.2 异构内存管理机制

为了降低异构内存的系统功耗，Park H等人[3]提出3种优化策略。

- DRAM能耗动态监测机制，以DRAM行（row）为粒度对DRAM中的数据进行监测，系统维护每一个DRAM row的监测值，并定期衰减此检测值，一旦检测值为0，则对相应的数据进行写回操作，从而减少DRAM刷新能耗。
- DRAM bypass机制，该系统认为对数据的第一次访问并不能说明数据是热数据，当此数据再次被访问的时候才会被判定为热数据，才会被迁移到DRAM。
- 脏数据保持机制，因为程序的局部性原理，脏数据接下来被访问的概率较高，所以尽可能让脏数据在DRAM中保持较长时间，从而降低DRAM换入换出的开销，并且在写回时进行写操作合并策略，降低对NVM的写频次，更进一步地降低系统功耗。该方案降低了系统的功耗，但是定期的页面写回操作对系统性能有一定的影响，并且策略的实现复杂度较高。

通过对各个Linux段的访存模式分析，Park Y等人[14]提出了基于段的页面放置机制。Linux把线性地址空间分为很多个段，每一个段中的数据具有相似的访存特征。比如文本段包含只读的运行代码，访存模式为只读，而堆栈段包含返回地址、参数和本地变量等经常被读写的数据，因此堆栈段的访存模式是频繁读写的模式。根据这些信息，在程序运行之前，针对性地把读写频繁的段的数据放置在DRAM中，而把只读的段的数据放在NVM中，从而提升异构内存系统性能。

通过对多级队列算法进行改进以及对队内元素进行动态排序，Ramos L E等人[4]提出了一种基于排序策略的异构内存页面放置（rank-based page placement，RaPP）机制。与参考文献[12]类似，RaPP的理念也是尽量将写入操作较为频繁的页面放入DRAM，而将写入操作不频繁的页面放入NVM。不同的是，首先，RaPP通过修改内存控制器监测页面的写操作频次以及页面写密集程度；其次，RaPP根据监测信息对页面进行动态排序；最后，RaPP在发生页面替换时，优先替换队列中排名较为靠前的页面。虽然该方案缓解了页面统计的

开销问题，但并未从根本上解决问题，页面迁移操作也给系统带来了不小的性能影响。

通过对堆中对象的访存特征的分析，Wei W等人[5]提出一种软硬件结合的异构内存页面放置框架——二维页面放置机制（2-phase page placement framework，2PP）。2PP的原理主要分以下两步：

- 离线分析，通过离线运行应用程序，获取并分析应用程序的访存特征（读写比例、程序计数器对象特征等），从而指导页面的初始放置；
- 运行时调整，通过在内存控制器中对页面进行读写监测，周期性地计算页面的读写比例，从而筛选出要迁移的页。

2.3 异构内存持久化

通过使用一种基于影子内存（shadow memory）的数据持久化方式，Liu M X等人[15]提出了一个将程序执行和数据持久化操作完全解耦的NVM持久内存框架——DUDETM。DUDETM的原理是，将系统执行需要使用的数据全部从NVM缓存到位于DRAM的影子内存，系统对持久数据的访问均在影子内存中完成，不直接访问NVM。当需要对数据进行持久化处理时，DUDETM采用本地更新的方式将数据写入影子内存，同时将对应的日志信息写入DRAM缓冲区。待事务提交之后，后台进程会异步地将日志数据持久化到NVM中，并根据日志数据更新位于NVM中的持久数据。相比于不具有数据持久化功能的内存事务系统，DUDETM仅牺牲了7.4%~24.6%的系统吞吐量就实现了数据的持久化操作。但是作为层次化的架构，系统的性能很大程度上受限于DRAM缓存容量。

为了能够充分发挥DRAM/NVM异构内存性能，同时保证数据强制一致性，Xu J等人[6]设计了一个新颖的内存文件系统——非易失内存加速的日志文件系统（non-volatile memory accelerated log-structured file system，NOVA）。NOVA通过日志来保证用户对文件系统元数据、文件数据操作的原子性，并且提供保证数据一致性的Atomic_mmap接口使应用可以直接访问NVM。在提高系统访问性能方面，NOVA扩展了已有的Log-structured文件系统技术，充分发挥NVM可快速随机访问数据的特征，并且为每一个索引节点（inode）都配置了独立的日志结构，以保证访问的高并行性。此外，NOVA还将复杂的元数据结构存储在DRAM中，以加速数据查找。

2.4 异构内存缓存管理

Wang Z等人[16]提出了一种基于NVM的写回操作感知的缓存动态管理机制（writeback-aware dynamic cache management for nvm-based main memory system，WADE）。针对NVM的读写时延不同，WADE把片上共享高速末级缓存（last level cache，LLC）分成两个列表：一个仅包含频繁写回的Cache line的列表，一个记录其他Cache line的列表。通过替换非频繁写回的Cache line，WADE可以显著地减少对NVM内存的写操作频次。但是WADE并不适合应用在异构内存环境中，原因如下。

- 异构环境中有4种访存模式：DRAM读、DRAM写、NVM读、NVM写，在LLC中维护4个列表，开销比较大。
- 未考虑DRAM与NVM之间的性能差异，因为在异构内存中，DRAM与NVM之间的性能差异要大于NVM读操作与NVM写操作之间的性能差异，所以发生缓存替换时，应当充分考虑DRAM与NVM的特性。

Wei W等人[17]提出一种面向异构内

存的缓存管理（hybrid memory-aware partition in shared last-level cache，HAP）机制。HAP机制把整个共享缓存分成两个部分，一部分用来缓存NVM页面的数据，一部分用来缓存DRAM页面对应的数据，通过对采样数据的分析比较，动态调整NVM缓存区在整个缓存的占比，以达到LLC中DRAM 与NVM缓存块最优的配比。HAP机制只有5种配比策略，因此并不能保证每次调整都是最优配比。因为HAP机制使用简单的LRU替换算法，发生LLC缺失的时候并未考虑替换块的类型，所以会导致因替换而影响缓存块的最优配比，造成性能的降低。

3 异构内存关键技术

为了更好地发挥异构内存的优势，同时也为了给上层应用提供高效、可靠的数据访存和资源调度与管理等系统级支撑，主要从异构内存全系统模拟器、异构内存体系结构、异构内存系统软件、异构内存编程模型、面向大数据的异构内存应用等方面进行研究。

3.1 异构内存全系统模拟器

目前市场中还没有商用的NVM产品，DRAM-NVM异构内存相关的研究主要是基于模拟器完成的。目前主流的模拟器主要有两种：软件模拟器和硬件仿真器。

3.1.1 软件模拟器

软件模拟器基于Zsim[18]和NVMain[19]的异构内存模拟器进行研究。Zsim是基于Intel Pin[20]工具集的系统模拟器，它使用Intel Pin工具集获取应用程序访存虚拟地址，并将虚拟地址重放到模拟的系统架构中（包括CPU、片上高速缓存、片上互联网络等）模拟应用执行和访存特征。此外，Zsim还支持使用其他内存模拟器（如NVMain、DRAMSim2[21]等）模拟其物理内存。NVMain是精确的内存模拟器，可模拟DRAM和NVM等物理内存的访问行为和物理组件。

但当前Zsim模拟器缺少TLB以及操作系统的模拟模块，通过Intel Pin工具集获取访存虚拟地址后，直接用该虚拟地址访问Cache模块和内存模块。不同应用虚拟地址存在大量的重叠，因此当前的Zsim不能准确模拟多进程混合的访存行为。为了解决这个问题，笔者在当前Zsim模拟器中添加了操作系统内存管理模块、页表和TLB模拟模块，操作系统内存管理模块主要负责为虚拟页分配物理页[7]，页表、TLB模拟模块主要负责将虚拟地址转换成物理地址。修改后的Zsim模拟器可精确地模拟单进程、混合多进程的运行和访存行为。具体架构如**图3**所示。

此外，为了给DRAM-NVM异构内存系统管理优化提供基本的平台支持，笔者基于Zsim和NVMain开发了DRAM-NVM层次化异构内存架构和DRAM-NVM统一编址的异构内存架构。为了进一步提升异构内存系统性能，笔者基于DRAM-NVM统一编址的异构内存架构，提供了以下两种内存管理策略。

- 行缓存局部性感知（row buffer locality aware，RBLA）[22]的页迁移策略：考虑行缓冲区命中时，NVM读写访问时延接近DRAM读写访问时延，该策略将行缓冲区局部性差的NVM内存页迁移到DRAM中，将行缓冲区局部性好的NVM页放置于NVM中，尽量降低NVM行缓冲区缺失带来的巨大性能和能耗开销。
- 基于多级队列（multi-level queue，MQ）的热页迁移策略：采用多级队列算

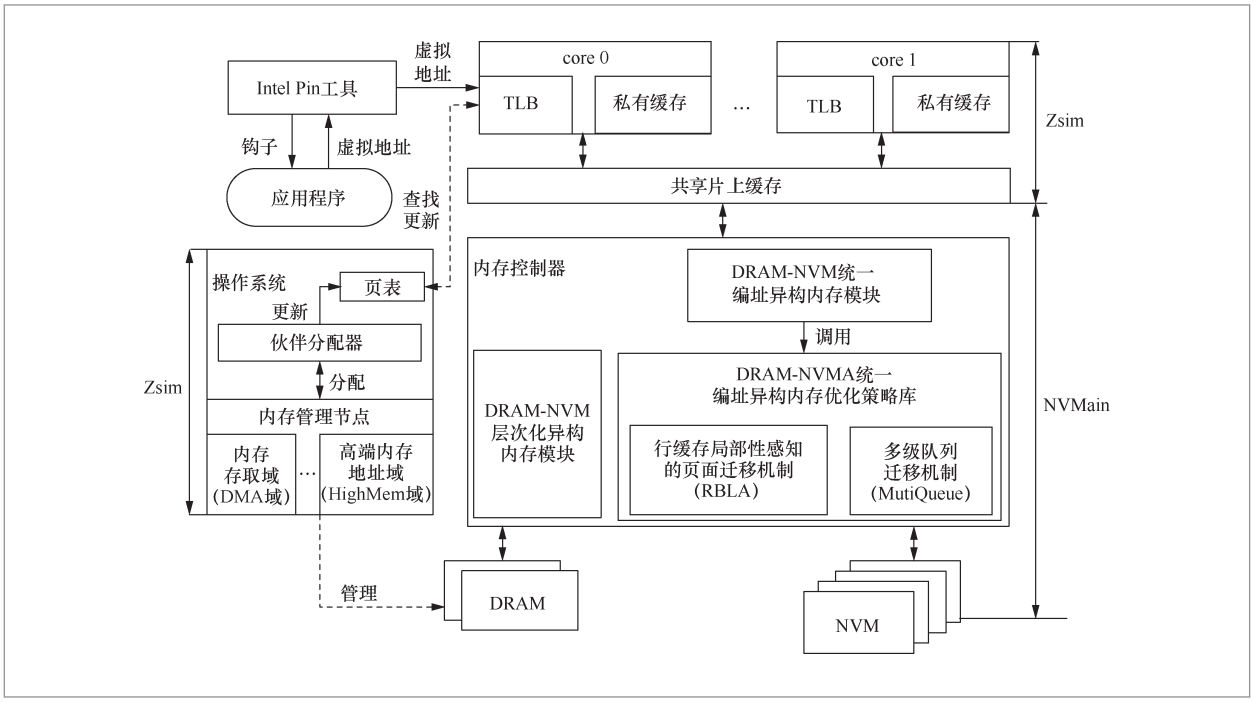


图 3 基于 Zsim 和 NVMain 混合模拟器的异构内存平台架构

法，根据页面访问频度将内存页划分为冷页面和热页面，将访问频繁的热页面由NVM迁移到DRAM，从而提升系统性能。

3.1.2 硬件仿真器

现有的软件模拟方法模拟速度较慢、无法准确地模拟复杂和规模较大的工作负载，而且对网络通信、虚拟化和分布式应用的兼容性较差，笔者提出了一种基于DRAM的性能仿真器——异构内存仿真器（hybrid memory emulator，HME[23]），以模拟即将到来的NVM技术的性能和能耗特性。HME利用商业级CPU中的非统一内存访问架构（non-uniform memory access architecture，NUMA）中可用的功能来模拟两种内存：快速、本地DRAM和其他NUMA节点上的较慢的远程NVM。通过在远程NUMA节点上注入不同的内存访问时延，HME可以模拟范围广泛的NVM时延和带宽。

为了帮助程序员和研究人员评估NVM对应用程序性能的影响，笔者还提供了一个高级编程接口，用于从NVM或DRAM中分配内存。

在商用硬件中没有编程接口直接控制内存访问时延，因此笔者采用了一种软件方法来模拟NVM读时延。其基本思想是在合适的时间间隔内为应用程序注入软件生成的额外时延。笔者通过对应用程序的内存访问进行监控，并通过内存访问行为对滞后时间进行建模，以逼近一段时期内的NVM读时延。

一般情况下，数据写入可以归类为写直达法和写回法两种方式。在写直达法中，数据立即通过缓存写入内存中。在写回法中，数据首先写入缓存，并在以后将相应的缓存行逐出时写入内存。虽然内存写入操作通常不在应用程序执行的关键路径上，但是，它仍会占用内存总线带宽，也就

可能造成内存读取请求的响应被推迟，因此内存写入对系统的影响不应被忽视。笔者通过特殊的性能监控硬件来监控写直达操作和写回操作，并通过构建内存读写次数与写时延之间的关系来模拟NVM写时延，从而实现了在不增加读请求响应时间的同时，准确地模拟写操作时延。

在带宽模拟方面，笔者通过限制最大可用DRAM带宽来模拟NVM带宽。带宽节流是通过在商品英特尔至强处理器中提供性能计数器来控制DRAM能通过的最大访存指令实现的。笔者还建立了能源消耗模型来计算NVM的能耗。由于缺少用于计算主存储器所消耗能量的硬件级特征，笔者选择了一种统计方法对NVM能耗进行建模。不像DRAM，NVM不产生静态功耗，故只需要计算NVM读写能耗。记录一段时间内NVM的读写访存次数，通过建立简单的模型来模拟NVM单位时间内的能耗。

笔者使用NUMA体系结构在双插槽服务器上模拟混合内存系统。每个插槽都包含三级 CPU。NUMA体系结构既支持时延较低的本地DRAM访问操作，也支持时延较高的远程Socket上的DRAM访问操作。因此，笔者使用远程节点上的DRAM来模拟时延较高和带宽较低的NVM。

如**图4**所示，笔者在远端Socket中选择一个核心（core），用来计算应向本地Socket上的应用程序注入的附加时延。笔者关闭远端Socket上的其他core以避免对计算核的性能干扰。HME使用英特尔至强处理器性能监视单元（uncore PMU）中本地代理（home agent，HA）来记录正在本地节点0上运行的应用程序对远程节点1上的DRAM内存读取访问的次数（LLC 缺失）和写访问次数。HME通过核间中断（inter-processor interrupt，IPI）向节点0的核心注入额外的时延。这样，HME增加了远程内存访问时延，以模拟NVM时延。

3.2 异构内存体系结构

由于目前市场上没有商用的异构内存

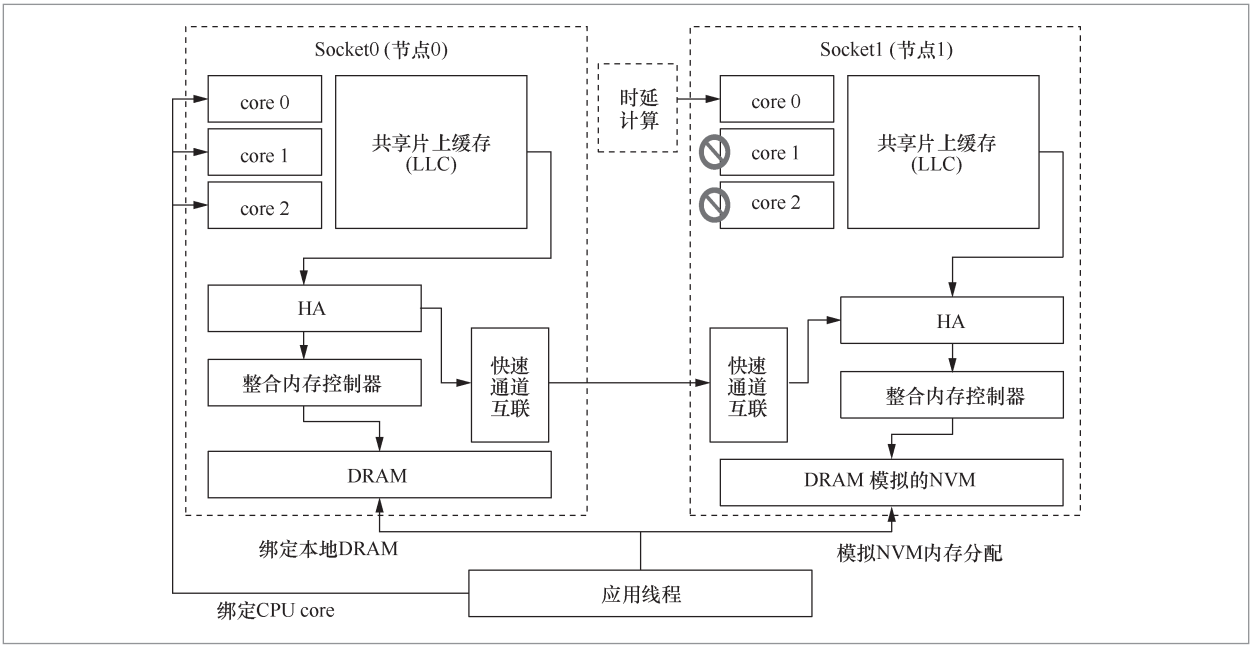


图 4 HME 硬件仿真器系统架构

产品，笔者在模拟器中进行异构内存体系结构的研究，主要研究包括软硬件协同管理的层次化异构内存架构、面向异构内存架构的片上缓存管理和异构内存并行处理等方面。与此同时，由于在不同的应用场景下，异构内存系统中两种架构各有优劣，笔者在模拟器中实现了软件定义的可重构异构内存体系架构。

3.2.1 软硬件协同管理的层次化异构内存架构

Demand-Based数据管理机制是当前内存计算环境下DRAM-NVM层次化异构内存系统主要采用的数据管理机制，但这种数据预取机制没考虑NVM读写性能不对称的特征，且存在硬件复杂度高、存储开销大、软件灵活性低、DRAM缓存利用率低等挑战，并不适用于DRAM-NVM异构内存系统。

基于DRAM-NVM层次化异构内存系统，笔者提出了基于效用的（utility-based）软硬件协同管理的层次化异构内存架构（hardware/software cooperative caching for hybrid DRAM/NVM memory architectures，HSCC[7]）。如**图5**所示，HSCC在页表和TLB中建立了NVM内存到DRAM缓存的映射，消除了复杂的硬件开销和巨大的存储开销，同时将多级数据预取机制的实现提升到软件层次，增加了系统的可扩展性。HSCC实现了基于效用的多级数据预取机制，该机制仅将NVM热页面预取到DRAM缓存，提升了DRAM缓存利用效率。此外，HSCC在基于效用的多级数据预取机制的基础上，提出了HSCC-Dyn和HSCC-MQ两种优化策略：HSCC-Dyn使用爬山算法动态调整数据预取阈值，充分利用DRAM缓存的同时，尽量减少DRAM缓存回收造成的性能开销；HSCC-MQ采用多级队列算法动态更新NVM页面访问热度，提升了NVM页面热度监测的准确度。

3.2.2 面向异构内存架构的片上缓存管理

传统基于提高缓存命中率的策略在传统DRAM内存中展现了良好的性能。然而，

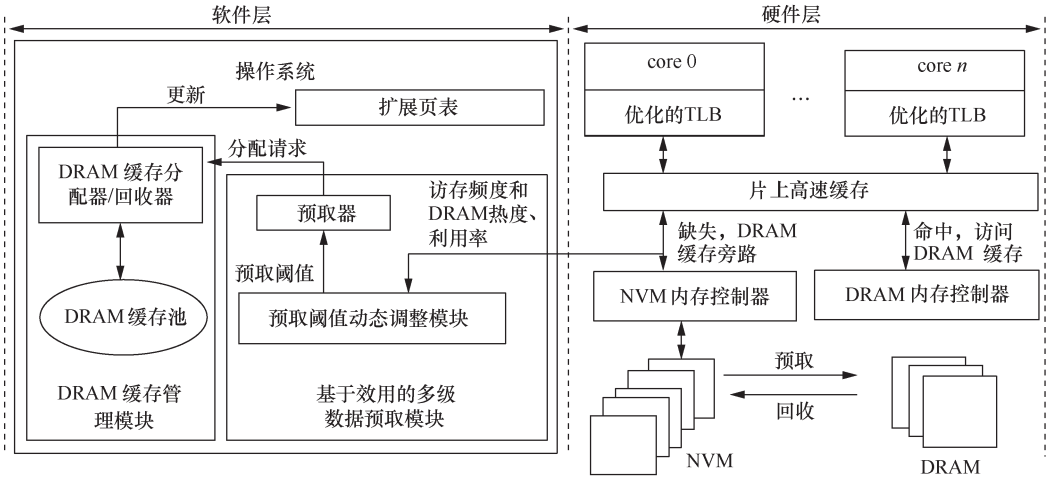


图 5 软硬件协同管理的层次化异构内存架构（HSCC）总体架构

这些策略在混合内存环境下性能不高。这是因为DRAM与NVM的访问时延存在着多倍的差异，不再像传统DRAM内存环境下那样有着相对比较单一的访存时延开销。单纯地考虑提升命中率不能反映整体的访存开销，因此这些算法已经不能完全适应混合内存环境新的特性。混合内存环境下的LLC替换策略还需要将LLC缓存缺失时的访存时延开销这一变量纳入缓存替换决策的考虑范畴。因此，笔者考虑LLC的命中开销、DRAM缓存块的缺失代价以及NVM缓存块的缺失代价，并采用平均访存时延作为评价混合内存下缓存性能的指标。

笔者提出了一种新型缓存替换策略——异构内存环境下的缺失代价感知的缓存替换策略（miss-penalty aware LRU-based cache replacement for hybrid memory systems，MALRU[8]）。MALRU的主要思想是增加高时延缓存块留存在LLC中的概率，降低这些高时延缓存块缺失时带来的昂贵访存开销。MALRU的整体构架如图6所示，MALRU通过设置一个保留指针，将LLC在逻辑上分为一个替换区间和一个保留区间。当发生LLC缺失时，MALRU会从LRU开始，在替换区间内寻找第一个DRAM缓存块，并将其替换出去。如果在优化区间内没有找到DRAM缓存块，则将LRU位置的NVM缓存块替换出去。这种策略相当于增加了NVM缓存块留存在LLC中的概率，提高了NVM缓存块的命中率，从而减少了因为高时延的NVM缓存块被替换出去带来的昂贵的访存开销。同时，保留区间内的DRAM缓存块和NVM缓存块不会被替换出去。这是因为这些缓存块均为被频繁访问的缓存块，保护其不被替换出去能够避免这些缓存换被频繁地换进换出而导致的缓存颠簸。

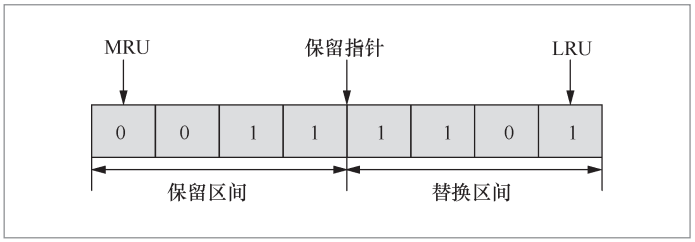


图6 MALRU

3.2.3 异构内存并行处理

并行程序在异构内存系统中运行时，一些运行较慢的线程可能会减缓其他线程的运行速度。原因在于：所有线程的访存请求被分发到不同的内存通道。NVM的写请求时延过长，访问NVM的请求需要较长的完成时间，因此NVM写请求数量多的线程需要更多的时间来完成内存访问操作，该线程将因此成为运行最慢的线程。该运行最慢的线程显然将拖慢其他线程的运行进程，从而导致整个程序的性能下降。在异构内存环境下更好地调度线程的访存请求，可以缓解这种性能的损失。笔者提出了一种针对异构内存环境下并行程序的访存调度算法（如图7所示）。该算法在运行时监测每个线程的访存请求数量，然后根据访存请求为每个线程计算调度优先级，最后执行调度策略。

为了能够在异构混合内存环境下对多线程程序进行调度，必须对多线程程序的访存请求进行监测，包括记录每个访存请求是由哪个处理器发出的、每个访存请求所请求的数据的物理地址，这是对整个系统的基本要求，通过对这些访存请求进行分析，确定内存控制器的调度算法。

在记录了每个线程访存信息之后，如何通过这些信息确定内存控制器的调度算法，减小NVM访存时延高所带来的系统性能下降，提高程序整体性能，是进一步要研究的问题。该难点在于如何在监测的访存请求的基础上制定一个合适的调度优化

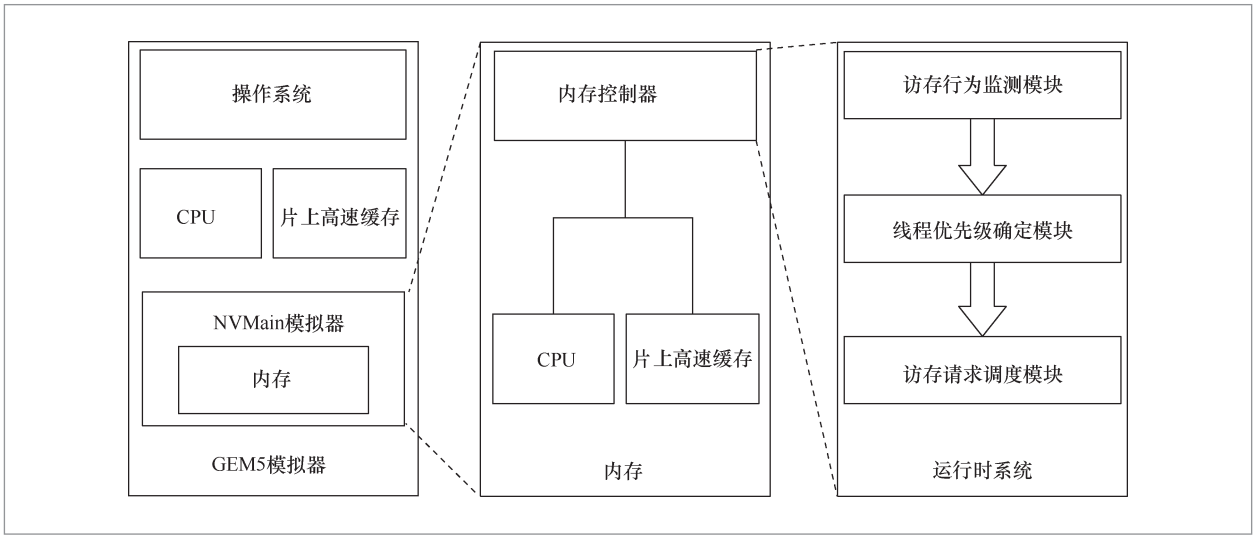


图 7 面向多核的异构内存调度策略整体功能模块

算法，尽可能使运行时调度开销与程序性能优化之间达到平衡。

如**图7**所示，在内存子系统中，异构内存控制器负责对访问DRAM和NVM的请求做优化调度，在异构内存控制器中主要包含访存行为监测模块、线程优先级确定模块、访存请求调度模块3个模块。访存行为监测模块主要在程序运行期间对程序每个线程的访存行为进行检测，获取确定调度策略的依据信息；线程优先级确定模块会根据访存行为为每个线程计算出一个调度权值；访存请求调度模块会在已有的FRFCFS调度算法的基础上优先调度权值高的线程。

3.3 异构内存系统软件

为了研究新型异构内存管理系统，基于传统内存介质和新型非易失性内存的访存特性和介质属性，研究新型异构内存管理机制，将内存和存储资源结合起来进行统一管理，实现异构内存体系架构下系统软件的基础。

3.3.1 能耗管理

随着计算机系统软硬件的发展，计算机内存系统的能耗问题日益突出。诸如PCM等新型非易失性存储器依靠介质的电阻特性存储数据，为解决现代计算机系统的能耗问题提供了契机，但这些存储器存在写功耗大的缺陷。为了改善异构内存系统的能耗问题，笔者提出了一种新的能耗管理策略——异构内存能耗管理机制（EMH[9]）。

如**图8**所示，EMH策略根据内存页面访问特征信息划分冷页面和热页面，并经过一系列的排序和筛选得到若干个冷页面和热页面，最后采用能耗模型预测每个页面的能耗值，找到可使系统能耗效率最大化的冷热页面组合，迁移配对的冷热页面，从而提升系统的能耗效率。具体来说，EMH主要有以下3个方面的设计。

- 统计页面全局和局部访存信息的数据结构设计。在异构内存管理中，分配页面的依据就是页面的访存历史信息，需要考虑页面从被访问开始到当前时间的全局和局部访存信息。为了准确地记录这些页面访存

信息，笔者设计了一个能统计页面全局和局部访存信息的数据结构。

● 基于页面全局/局部写密集度和时间局部性的冷热页面决策算法设计。由于采用了本文设计的记录访存信息的数据结构，系统运行时全局和局部的页面读/写信息都被记录下来，充分参考这些信息，从全局角度和局部角度出发，利用时间局部性制定了一个冷热页面决策算法。此算法准确地对页面的未来行为进行预测，从而更为准确地选出写操作频繁的热页面和系统不经常访问的冷页面。

● 基于数据局部性的异构内存能耗迁移模型设计。本文基于数据局部性对异构内存能耗以页面为单位进行数学建模，通过该模型可以计算出迁移某页面组合后所带来的节能效益，对计算出的节约能耗值进行排序，挑选出使系统能耗降低最大化的页面组合，更好地达到降低异构内存整体能耗的目标。

3.3.2 异构内存全系统检查点

在异构内存环境下，为了减轻预复制对其他应用的内存访问干扰、降低检查点的开销，笔者对不同应用在混合内存上的访存行为进行了研究，发现多数应用访问内存Bank（一组存储颗粒的集合）的频繁程度随着时间变化有着较大的变化，并且即使在同一时间段内，不同Bank的访存频繁程度也有较大差异。根据这个特征，可以利用应用访问Bank的空闲时间完成预复制，减少检查点对应用访存的干扰，降低检查点操作的开销。

基于上述思想，笔者在混合内存上实现了一个面向应用内存访问优化、降低检查点开销的预复制检查点系统——Shadow。Shadow利用预复制检查点数据复制顺序灵活可变且对内存访问时延不敏感的特征，优化了预复制检查点的访存行为。Shadow系统需要通过软件和硬件的协同工作来实现，其总体设计原则为将复杂、高开销的预复制检查点流程控制、Bank访问频繁程度预测等功能放在软件层实现；对于简单且需要底层支持的应用Bank访问信息采集、单个预复制事件数据复制、冲突预复制操作处理等功能，通过在传统内存控制器上添加新的部件，在硬件层实现。其整体结构如图9所示。具体来讲，分别在软件与硬件层面实现了以下主要功能。

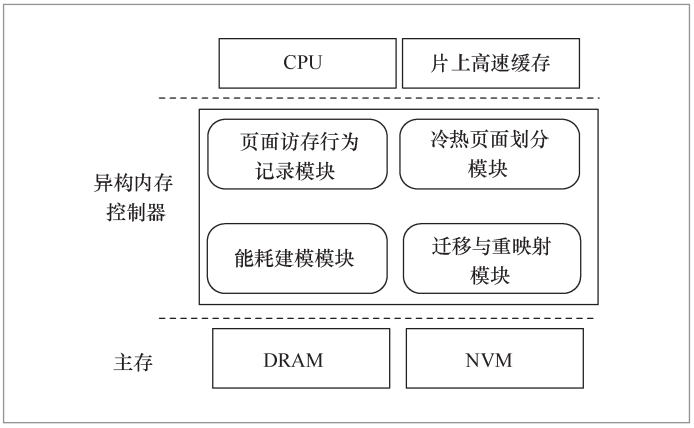


图 8 EMH 系统模块

在系统软件层面，主要实现了以下功能。

● 预复制检查点流程控制：该模块用于实现预复制检查点的基本功能，包括跟踪应用数据的修改、生成检查点数据复制事件、控制检查点操作的起止、维护检查点操作元数据等。

● Bank访问频繁程度预测：该模块根据硬件收集的应用Bank访问信息，通过预测算法计算得出未来应用对各个Bank访问的频繁程度，为预复制检查点流程控制模块生成新的检查点复制事件提供参考。

在硬件层面，在内存控制器上添加了以下功能部件。

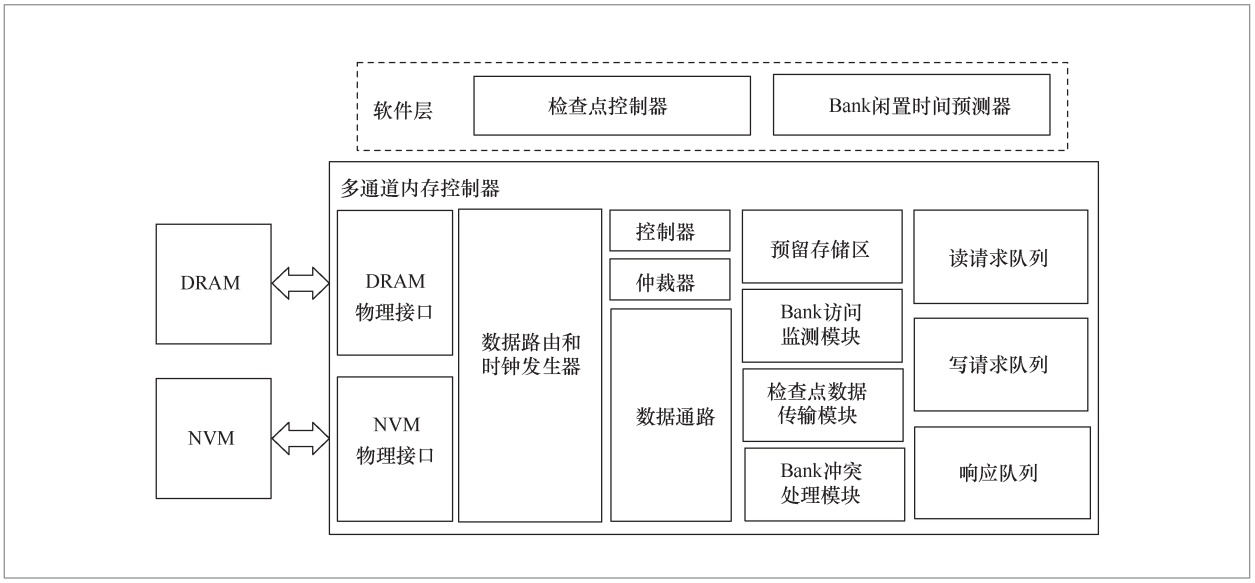


图 9 Shadow 系统架构

● 应用Bank访问信息采集：该模块用于监测到达内存控制器的应用访存请求，记录应用对各个Bank的访问情况，为软件中的预测模块提供输入数据。

● 预复制事件数据复制：该模块从软件中的检查点控制模块接收单个预复制事件的源地址、目的地址，在内存控制器中完成检查点数据的复制，使检查点数据无须经过CPU。

● 冲突预复制操作处理：该模块监测到达内存控制器的应用访存请求，判断应用是否与预复制检查点发生Bank冲突，并通知预复制事件数据复制模块暂停或重启复制过程。

● 检查点数据寄存器：该模块用于存储检查点执行过程中产生的关键数据，包括Bank访问记录、检查点事件信息、缓存的检查点数据请求，软硬件各功能模块通过寄存器中的数据进行数据交流。

3.3.3 大页管理

随着应用内存需求、计算机系统内存容量的持续增加，虚实地址转换逐渐成为系统性能瓶颈。现有的计算机普遍采用TLB来加快虚实地址转换的速度。由于每一次CPU访存操作都需要优先访问TLB，如果TLB命中，则直接使用获取的物理地址访问缓存或者内存中的数据，如果TLB缺失，则需要进行查询系统页表的操作。由于现有Linux采用四级页表的机制，访问页表导致多达4次的访存开销，这将严重影响系统性能。同时，由于TLB位于系统的关键路径上，考虑到时延和能耗等相关因素的影响，TLB硬件的容量多年来发展缓慢。在不改变现有TLB硬件特性的情况下，使用大页机制成为解决TLB瓶颈问题的关键。而另一方面，随着新型存储介质NVM的出现，计算机系统内存容量得以显著增加，但由于NVM自身存在缺陷，导致主流的研究主要集中在DRAM/NVM异构内存系统中，在这些研究中，为了提升异构内存的性能，数据迁移成为异构内存优化的一个重要手段。然而，因为现有的大页管理机制不能很好地支持页面迁移，所以这些方式在

异构环境下性能不高。

笔者提出了一种异构内存环境下大页管理机制（如图10所示）。该机制的主要思想是，对DRAM和NVM采用不同的粒度进行管理，通过使用Split TLB机制来加快DRAM和NVM的地址转化速度，并在内存控制器中使用位图来标识页面迁移的状态，从而在逻辑上保证大页的完整性。同时为了减少页面监测的开销，采用轻量级的页面监测机制，即对内存页面分两步进行检测：首先，监测大页的冷热程度，并筛选出最热的N个大页；接着对筛选出来的N个大页进行细粒度的监测，从而筛选出最热的细粒度页面。此外，为了优化DRAM中热页面的访问，笔者设计了一套DRAM页面重映射机制。

3.4 异构内存编程模型

在平行化异构内存系统中，只有将应用程序的数据放置在合适的内存介质中，才能充分利用NVM和DRAM各自的优势。现有的研究主要着重于利用页面迁移的机制来获取更好的性能和更优的能耗效率。然而这些机制都依赖于在线的页面监测，而这种监测机制需付出很大的性能开销，并且以页面为粒度进行数据迁移的操作会浪费一定的DRAM带宽，并降低DRAM的利用率。

笔者提出了一种基于对象的内存分配和迁移机制（OAM）。OAM利用离线访存分析工具获取对象的访存模式，并用性能/能耗模型来指导应用程序的数据分配，并为有需要的数据对象提供迁移操作。基于OAM中提供的用于平行化异构内存系统内存分配和数据迁移的编程接口，应用程序的源码可通过OAM提供的静态代码工具自动修改。当程序开始执行后，操作系统根据OAM提供的分配建议以及当前系统中DRAM的可用量共同决定对象的放置。如图11所示，OAM主要包括离线分析（offline instrumentation，OFI）模块和动态分配（online allocation，

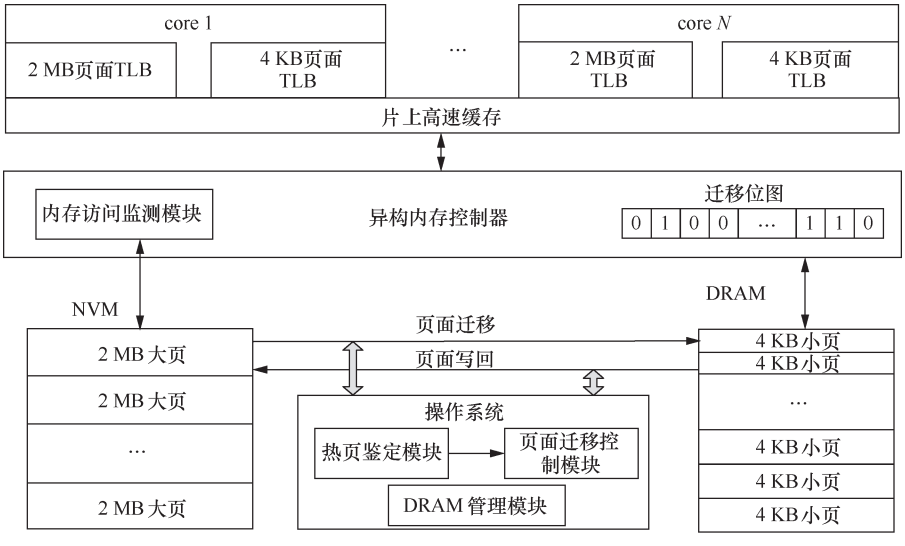


图 10 异构内存大页管理机制

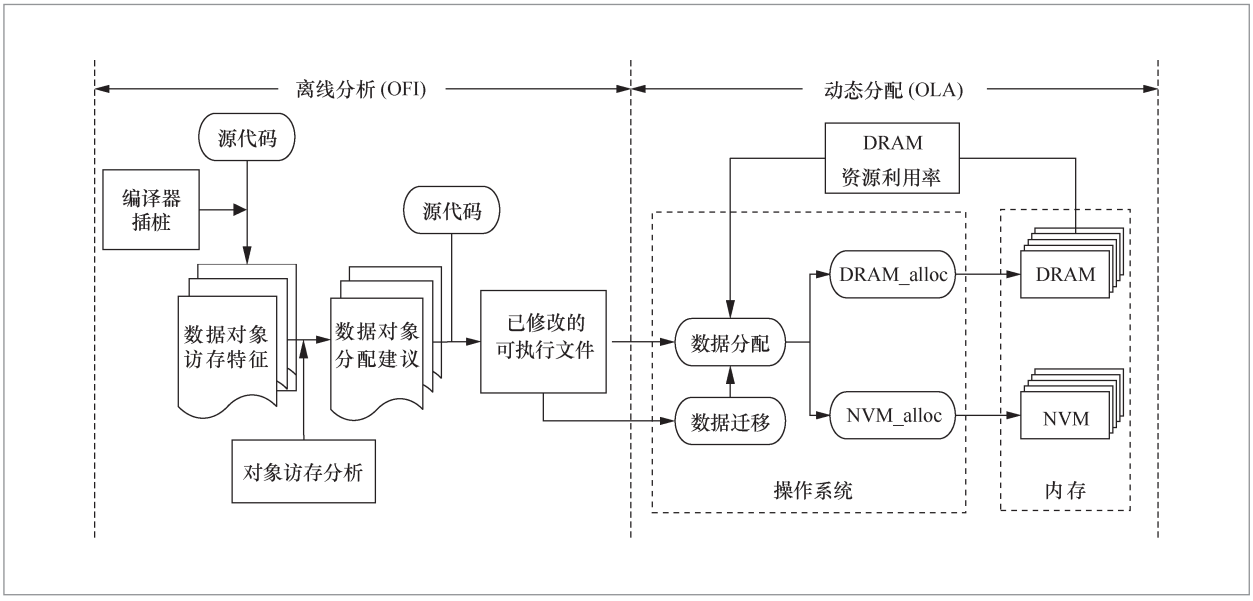


图 11 基于对象的内存分配和迁移机制总体架构

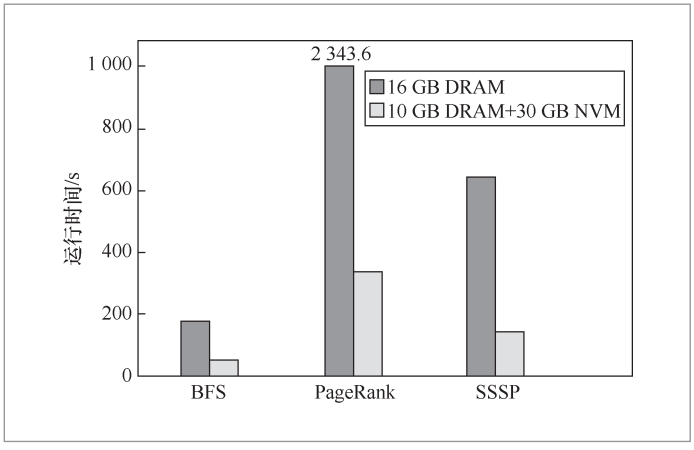


图 12 各种图计算应用下性能对比

OLA）模块两个模块，并且通过提供适用于本系统的编程接口以及基于底层虚拟机（low level virtual machine，LLVM）实现的自动化修改程序源代码的机制，将这两部分有效地结合。

3.5 面向大数据的异构内存应用

大数据应用需要越来越大的内存，而新兴的NVM内存，因其优异的特性成为内存扩展更好的选择。相比于DRAM，NVM具有较低的带宽和较高的时延，但也具有大容量和成本低廉的优势。笔者量化分析了相同成本下，异构内存环境带来的性能提升。由先前的工作可知，DRAM内存的成本约为NVM内存成本的5倍，使用Twitter的数据集测试和对比了同种成本情况下多种图计算应用的性能。如**图12**所示，相同成本情况下，异构内存系统中程序的性能明显优于纯DRAM系统，其中宽度优先搜索（breadth first search，BFS）算法、网页排序（page rank）算法和单源最短路径（single-source shortest path，SSSP）算法分别获得了3.47倍、4.52倍、6.96倍的性能提升。这是因为16 GB的DRAM内存，图数据无法全部加载到内存中处理，部分数据需要存储在磁盘中，而相同的成本，采用异构内存技术，可以拥有更大的内存，有机会使数据全部在内存中处理，而无需与磁盘进行数据交换。

笔者进一步测试了相同成本下两种内存配置不同迭代次数的性能。如**图13**所示，随着迭代次数增多，程序执行时间相应变长，采用异构内存获得的加速比更大。这是因为，在较小的DRAM内存条件下，每次迭代都和磁盘发生数据交换，而在异构内存中，只有在第一次数据加载到内存中的时候，才有数据交换，其他时候数据都是在内存当中的，迭代次数较多时，异构内存减少的磁盘数据交换更多，节省的时间也就越多。

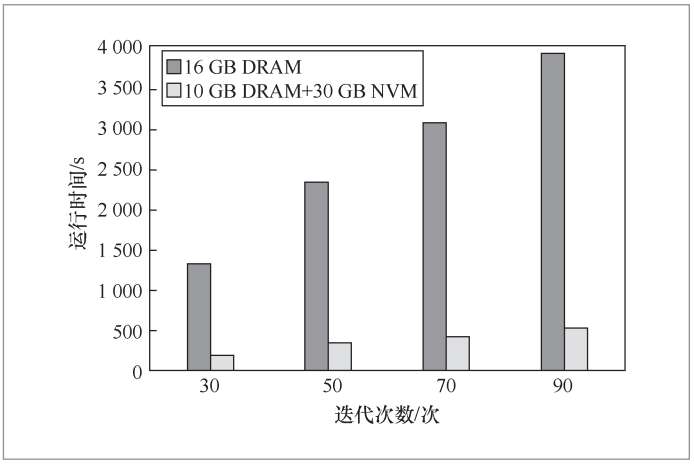


图 13 PageRank 不同迭代次数下性能对比

最后，笔者测试了在不同的数据集下，不同的图算法在两种内存环境下的性能提升。如**图14**所示，对于不同的应用程序、不同的数据集，采用异构内存获得的性能各不相同，但是总体来说，相同成本下，采用异构内存可以获得一定的性能提升。笔者未考虑固态硬盘（solid state drive，SSD）和磁盘的开销，因为有了足够大的NVM内存，可以不用SSD和磁盘来存放数据。把SSD或者磁盘开销考虑进去，程序性能提升会更大。

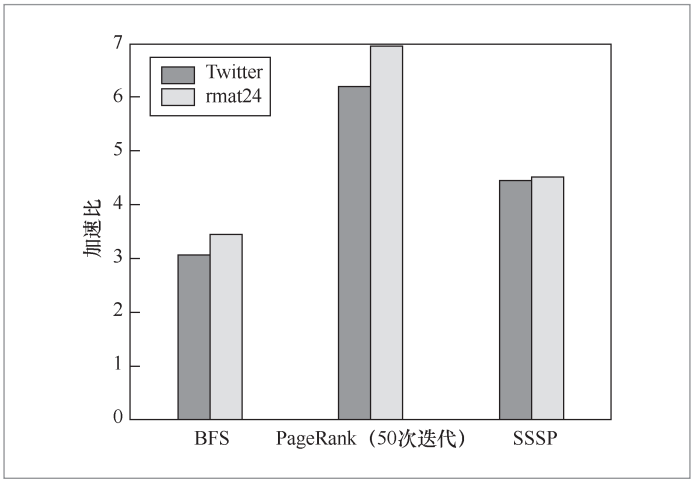


图 14 不同数据集各种图计算应用下性能对比

此外，针对图计算应用的特性，笔者提出一种面向图处理的异构内存管理系统。该系统主要思想有以下两部分。

- 异构内存图数据放置策略：在图数据中，由于图属性数据访存不规则，缓存命中率较低，所以将其放置在DRAM中，以减少数据访问时延；而图结构数据一般以数组形式存放，访存局部性较好，缓存命中率高，因此将其放置在NVM中；对于元数据等一些其他数据结构，数据量相对较小，可以缓存在缓存中，也可以放置在NVM中。
- 异构内存多核图处理优化策略：在同步执行模式下，多核并行处理图数据因为DRAM与NVM内存访存差异，各个核数据处理完成时间不同，任务快速完成的核会处于空闲状态，核利用率低，程序整体性能下降。优化策略为：在进行图数据划分和任务分配的时候，NVM中子图划分得小一些，DRAM中子图划分得大一些，使绑定到两种内存介质上的核处理数据完成时间尽量保持一致。

4 结束语

与DRAM相比，NVM的读写时延较大、写能耗较高、寿命有限，因此如何有

效地引进、高效地管理和准确地评估异构内存是当今研究面临的挑战。面对以上挑战，本文从异构内存系统架构、内存管理机制、缓存替换管理、内存请求调度、持久化以及应用等方面对面向大数据的异构内存系统进行分析与研究，并提出了面向大数据的异构内存系统的优化方案。

可以看出，随着对异构内存研究的深入，DRAM与NVM组成的异构内存取代纯DRAM内存指日可待，尤其是Intel、Micron等大公司对NVM的研究推进，加速了异构内存的商业化进程。笔者相信异构内存将会给整个计算机系统架构设计带来新的变革。

参考文献:

[1] QURESHI M K, SRINIVASAN V, RIVERS J A. Scalable high performance main memory system using phase-change memory technology[C]//The 36th Annual International Symposium on Computer Architecture, June 20-24, 2009, Austin, USA. New York: ACM Press, 2009: 24-33.

[2] MLADENOV R. An efficient non-volatile main memory using phase change memory[C]//The 13th International Conference on Computer Systems and Technologies, June 22-23, 2012, Ruse, Bulgaria. New York: ACM Press, 2012: 45-51.

[3] PARK H, YOO S, LEE S. Power management of hybrid DRAM/PRAM based main memory[C]//The 48th Design Automation Conference, June 5-10, 2011, San Diego, USA. New York: ACM Press, 2011: 59-64

[4] RAMOS L E, GORBATOV E, BIANCHINI R. Page placement in hybrid memory systems[C]//The International Conference on Supercomputing, May 31-June 4, 2011, Tucson, USA. New York: ACM Press, 2011: 85-95

[5] WEI W, JIANG D, MCKEE S A, et al. Exploiting program semantics to place data in hybrid memory[C]// The 2015 International Conference on Parallel Architecture and Compilation, October 18-21, 2015, San Francisco, USA. Piscataway: IEEE Press, 2015: 163-173.

[6] XU J, SWANSON S. NOVA: a log-structured file system for hybrid volatile/non-volatile main memories[C]//The 14th USENIX Conference on File and Storage Technologies, February 22-25, 2016, Santa Clara, USA. San Francisco: USENIX Association, 2016: 323-338.

[7] LIU H K, CHEN Y J, LIAO X F, et al. Hardware/software cooperative caching for hybrid DRAM/NVM memory architectures[C]//International Conference on Supercomputing (ICS '17), June 14-16, 2017, Chicago, USA. New York: ACM Press, 2017.

[8] CHEN D, JIN H, LIAO X, et al. MALRU: miss-penalty aware LRU-based cache replacement for hybrid memory systems[C]//Design, Automation & Test in Europe Conference & Exhibition (DATE), March 27-31, 2017, Lausanne, Switzerland. Piscataway: IEEE Press, 2017: 1086-1091.

[9] ZHANG J, LIAO X, JIN H, et al. An optimal page-level power management strategy in PCM-DRAM hybrid memory[J]. International Journal of Parallel Programming, 2017, 45(1): 4-16.

[10] DHIMAN G, AYOUB R, ROSING T. PDRAM: a hybrid PRAM and DRAM main memory system[C]//The 46th Annual Design Automation Conference, July 26-31, 2009, San Francisco, USA. Piscataway: IEEE Press, 2009: 664-669.

[11] 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.

[12] ZHANG W, LI T. Exploring phase change memory and 3D die-stacking for power/thermal friendly, fast and durable memory architectures[C]//The 18th International Conference on Parallel Architectures and Compilation Techniques, September 12-16, 2009, Raleigh, USA. Piscataway: IEEE Press, 2009: 101-112.

[13] LOH G H, HILL M D. Efficiently enabling conventional block sizes for very large die-stacked DRAM caches[C]//The 44th IEEE/ACM International Symposium on Microarchitecture, December 3-7, 2011, Porto Alegre, Brazil. Piscataway: IEEE Press, 2011: 454-464.

[14] PARK Y, PARK S K, PARK K H. Linux kernel support to exploit phase change memory[C]//Linux Symposium, July 13-16, 2010, Ottawa, Canada. [S.l.:s.n.], 2010: 217-224.

[15] LIU M X, ZHANG M X, CHEN K, et al. DudeTM: building durable transactions with decoupling for persistent memory[C]//The 22nd International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '17), April 8-12, 2017, Xi'an, China. New York: ACM Press, 2017: 329-343.

[16] WANG Z, SHAN S C, CAO T, et al. WADE: writeback-aware dynamic cache management for NVM-based main memory system[J]. ACM Transactions on Architecture and Code Optimization, 2013, 10(4): 1-21.

[17] WEI W, JIANG D J, 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): 1-25.

[18] SANCHEZ D, KOZYRAKIS C. ZSim: fast and accurate microarchitectural simulation of thousand-core systems[C]//The 40th Annual International Symposium on Computer Architecture, June 23-27, 2013, Tel-Aviv, Israel. New York: ACM Press, 2013: 475-486.

[19] POREMBA M, ZHANG T, XIE Y. NVMain 2.0: a user-friendly memory simulator to model (non-)volatile memory systems[J]. IEEE Computer Architecture Letters, 2015, 14(2): 140-143.

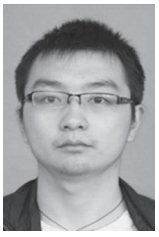
[20] LUK C K, COHN R, MUTH R, et al. Pin: building customized program analysis tools with dynamic instrumentation[J]. SIGPLAN Not, 2005, 40(6): 190-200.

[21] ROSENFELD P, COOPER-BALIS E, JACOB B. DRAMSim2: a cycle accurate memory system simulator[J]. IEEE Computer Architecture Letters, 2011, 10(1): 16-19.

[22] YOON H, MEZA J, AUSAVARUNGNIRUN R, et al. Row buffer locality aware caching policies for hybrid memories[C]//2012 IEEE 30th International Conference on Computer Design (ICCD), September 30-October 3, 2012, Montreal, Canada. Piscataway: IEEE Press, 2012: 337-344.

[23] DUAN Z H, LIU H K, LIAO X F, et al. HME: a lightweight emulator for hybrid memory[C]//Design, Automation & Test in Europe Conference & Exhibition (DATE), March 19-23, Dresden, Germany. [S.l.:s.n.], 2018: 1375-1380.

作者简介



王孝远(1990-),男,华中科技大学计算机学院博士生,主要研究方向为内存计算、操作系统等。

廖小飞(1978-),男,华中科技大学计算机学院教授,主要研究方向为内存计算、系统软件、分布式系统等。

刘海坤(1981-),男,华中科技大学计算机学院副教授,主要研究方向为内存计算、虚拟化技术等。

金海(1966-),男,华中科技大学计算机学院教授,主要研究方向为大数据处理、计算机系统结构、信息安全等。

收稿日期: 2018-04-20

基金项目: 国家重点研发计划基金资助项目(No.2017YFB1001603);国家自然科学基金资助项目(No.61672251, No. 61732010, No.61628204)

Foundation Items: The National Key Research and Development Program of China(No.2017YFB1001603), The National Natural Science Foundation of China(No.61672251, No. 61732010, No.61628204)