一种可重构异构内存架构和控制器

靳晓忠[1,2],刘海坤[1,2*],赖 皓[1,2],毛伏兵[1,2],张 宇[1,2],廖小飞[1,2],金 海[1,2]
(1. 大数据技术与系统国家地方联合工程研究中心/服务计算与系统教育部重点实验室/集群与网格计算湖北省重点实验室,华中科技大学,湖北武汉 430074;2. 华中科技大学计算机科学与技术学院,湖北武汉 430074)

摘 要: 融合传统动态随机访问存储器(Dynamic Random Access Memory,DRAM)与新型非易失性内存(Non-Volatile Memory,NVM)可构建平行架构或层次架构的异构内存系统.平行架构的异构内存系统往往需要通过页迁移技术把热点数据从NVM迁移到DRAM以提高访存性能,然而在操作系统中实现热页监测和迁移会带来巨大的软件性能开销.硬件实现的层次架构由于增加了访存层次,对于访存局部性差的大数据应用反而增加了访存延迟.为此,本文提出可重构的异构内存架构,可以运行时在平行和层次架构间进行转换以动态适配不同应用的访存特性.设计了基于新型指令集架构RISC-V(Reduced Instruction Set Computing-V)的DRAM/NVM异构内存控制器,利用少量硬件计数器实现了访存踪迹统计和分析,并实现了DRAM和NVM物理页间的动态映射和高效迁移机制.实验表明,DRAM/NVM异构内存控制器可提高43%的应用性能.

关键词: 非易失性内存;异构内存系统;异构内存控制器;内存访问监测;页迁移
基金项目: 国家自然科学基金(No.62072198);湖北省自然科学基金(No.2021CFA037)
中图分类号: TP32 **文献标识码:** A **文章编号:** 0372-2112(2024)09-3038-14
电子学报 URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20221257

A Reconfigurable Heterogeneous Memory Architecture and Memory Controller

JIN Xiao-zhong[1,2], LIU Hai-kun[1,2*], LAI Hao[1,2], MAO Fu-bing[1,2], ZHANG Yu[1,2], LIAO Xiao-fei[1,2], JIN Hai[1,2]
(1. *National Engineering Research Center for Big Data Technology and System/ Services Computing Technology and System Lab/Cluster and Grid Computing Lab, Huazhong University of Science and Technology, Wuhan, Hubei* 430074, *China*;
2. *School of Computing Science and Technology, Huazhong University of Science and Technology, Wuhan, Hubei* 430074, *China*)

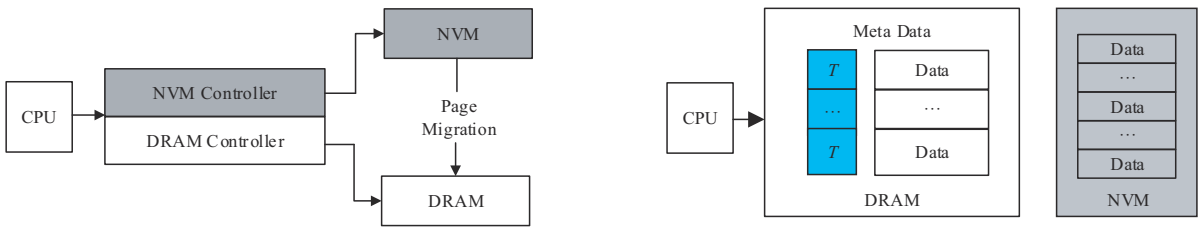
Abstract: Heterogeneous memory systems composed of traditional dynamic random access memory (DRAM) and new non-volatile memory (NVM) can be organized in a horizontal architecture or a hierarchical architecture. The horizontal DRAM/NVM architecture often requires page migration technologies to improve memory access performance. However, hot page monitoring and migration implemented in operating systems would cause significant software performance overhead. The hardware-supported hierarchical architecture even increases the memory access latency for big data applications with poor data locality due to the deeper memory hierarchy. To this end, this paper proposes a reconfigurable heterogeneous memory architecture that can be converted between horizontal and hierarchical architectures at runtime to dynamically adapt the memory access characteristics of different applications. We design a DRAM/NVM heterogeneous memory controller (HMC) based on the new instruction set architecture RISC-V (Reduced Instruction Set Computing-V). The HMC uses a few hardware counters for memory access monitoring and analyzing, and achieves dynamic address mapping and efficient page migration between DRAM and NVM pages. Experimental results show that the DRAM/NVM hybrid memory controller can improve application performance by 43%.

Key words: non-volatile memory; heterogeneous memory systems; heterogeneous memory controller; memory access monitoring; page migration
Foundation Item(s): National Natural Science Foundation of China (No.62072198);Natural Science Foundation of Hubei Province (No.2021CFA037)

收稿日期:2022-11-01;修回日期:2023-05-06;责任编辑:梅志强
*通讯作者:刘海坤

1 引言

随着大数据应用的蓬勃发展,当今的数据密集型应用对内存系统的容量和性能要求越来越高,传统计算机内存子系统面临诸多挑战.以动态随机访问存储器(Dynamic Random Access Memory,DRAM)为存储介质的传统内存系统由于其存储密度低、以及静态功耗高[1,2]等缺点,已经无法满足大数据背景下内存计算的要求.新型非易失性存储器(Non-Volatile Memory,NVM)具备存储密度高、无静态功耗、以及字节可寻址等优势,有望满足如今以数据处理为中心计算模式的内存资源需求,为促进内存计算技术的进步带来了新的契机.然而,与DRAM相比NVM也存在一定的缺陷,例如读写时延较长、读写寿命低等[3,4],使得NVM不能直接替代DRAM作为计算机主存.为了各取所长,充分利用DRAM较高的读写性能和NVM大容量的优势,目前的方案是利用大量NVM和少量DRAM构成异构内存系统.因此,如何优化这种异构内存系统的架构设计成为了如今内存计算架构研究方向的热点之一.为了在不同的计算情景下能充分发挥两种存储介质的优势,DRAM/NVM异构内存系统主要有两种架构,如图1所示.在平行架构中(图1(*a*)),NVM和DRAM共同作为主存进行管理.为了提升数据读写性能,需要将应用运行过程中频繁进行读写访问的高热度页面迁移至DRAM中,而不常访问的页面则存储在无静态功耗的大容量NVM中[4].在另一种层次化架构(图1(*b*))中,NVM作为主存介质,而将DRAM以高速缓存(cache)的形式通过硬件进行管理,从而降低NVM读写性能差带来的影响[5,6].



(*a*) Flat Architecture　　(*b*) Hierarchical Architecture

图1　DRAM/NVM异构内存架构图

平行架构使用热度检测的方式将热页频繁的在DRAM和NVM之间进行数据迁移.因为DRAM和NVM中的数据并不重复,每次迁移不仅需要将NVM中的数据写入DRAM,而且不论DRAM中的数据是否为脏数据,均需要将DRAM中的数据迁移到NVM.与之相反,在层次架构中,DRAM中存放的是NVM中数据的缓存,因此在迁移数据时仅需写回脏数据.此外,层次架构往往用简单的按需预取或者基于地址的预取,这种策略对于访存局部性不好的应用往往会造成缓存抖动,频繁的迁移会抵消缓存的收益.层次架构还存在灵活性低、硬件复杂度高、DRAM缓存利用率低等问题.总之,层次架构比较适宜于访存局部性很好的应用,而平行架构则适宜于对内存足迹大且局部性差的应用.之前的研究表明,对于某些大数据应用如Breadth First Search tree(BFS)或Minimum Spanning Forest(MSF),同样容量的异构内存采用平行或者层次架构性能差距甚至达到2倍以上[7].如图2和图3所示,我们在基于Reduced Instruction Set Computing-V(RISC-V)的Field Programmable Gate Array(FPGA)[8,9]硬件平台上使用两种架构运行不同的程序,由于开发板的内存容量有限,对各测试程序仅仅执行10^6条指令.实验测试了应用程序的每周期指令数(Instructions Per Cycle,IPC)和运行时间,结果显示不同的内存架构可使程序性能相差最大达2.5倍.单一的异构内存架构往往不能很好地适配不同应用的访存特性.因此,如何合理组织和管理异构内存,充分发挥两种架构的优势,从而最大化异构内存系统的性能,成为了异构内存系统研究的一个难点.

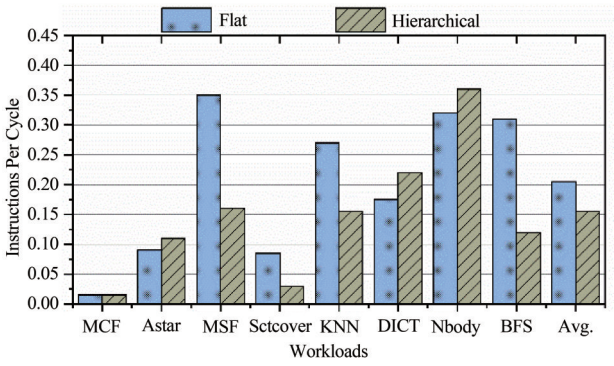


图2　层次架构和平行架构下程序的性能(IPC)

本文提出可重构的异构内存架构,可以在异构内存控制器中在线监测应用的访存特征,在运行时实现异构内存架构在并行和层次之间进行动态转换,从而适配不同应用的访存特性,提高应用执行的性能.为此,设计了基于新型指令集架构RISC-V[10]的DRAM/NVM异构内存控制器,利用少量硬件计数器实现了访存踪迹统计和分析,并实现了CPU无感的DRAM和

NVM物理页间的动态映射和高效页迁移机制. 在基于RISC-V的lowRISC-Chip硬件平台上对提出的异构内存架构进行了仿真验证,实验结果表明可重构异构内存架构可以为应用带来43%的性能提升.

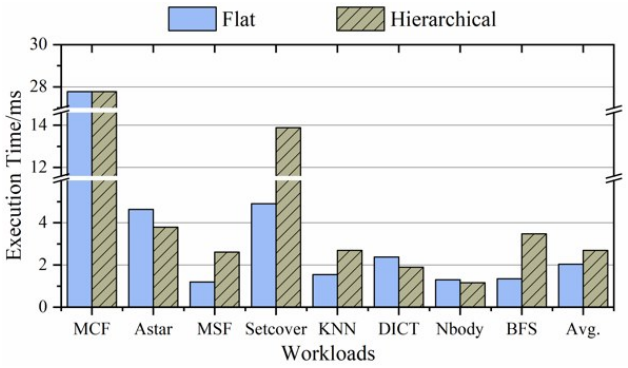


图3 层次架构和平行架构下程序的运行时间

2 相关背景

RISC-V指令集是一种全新的指令集架构,它由加州大学伯克利分校的Krste教授等人在2010年共同研发,其应用覆盖物联网设备、高性能计算机等众多领域. 从RISC-V架构出现以来,有众多研究者开发了相关硬件加速器和系统软件. LowRISC-Chip是一款基于RISC-V指令集架构的片上系统[11]. 它集成了片上缓存、片外内存、Video Graphics Array(VGA)接口显示器控制电路以及Xilinx MIG7系列内存控制器等Xilinx IP核,并采用AXI4片上互联协议实现内存系统的通信控制,使其成为了一个具备完整系统结构的FPGA硬件平台. 其具有较快的硬件仿真速度,可以提供高保真的硬件执行细节以及较好的可配置度的优点,是一款功能强大的体系结构硬件系统.

RISC-V片上系统普遍使用AXI4总线协议[12]作为与片外内存通讯的总线协议. AXI4是一种内存总线协议,基于突发(burst)传输进行设计,将传输单位设置规定为主设备(master)和从设备(slave),并将设备之间的传输形式链接分离为5个独立的通道:读地址通道、读数据通道、写地址通道、写数据通道和写响应通道.

英特尔傲腾(Optane)持久化内存技术[13]支持两种模式:层次架构(Memory Mode)和平行架构(App Direct Mode). 层次架构使用DRAM作为缓存、傲腾内存作为主存且不利用傲腾的持久化特性;平行架构下DRAM和傲腾内存都作为主存使用,并对操作系统可见,傲腾内存支持持久化特性. 如果从一种模式切换到另一种模式只能重新配置硬件使用模式并重启系统,无法实现两种架构在运行时动态转换. 单一的异构内存架构往往无法最佳适配各式各样的应用的访存特性,具有很大的局限性. 如大数据相关应用对延迟不敏感、对主存容量敏感,适合采用平行架构;互联网应用,对延迟敏感、对主存容量不敏感,适合采用层次架构.

3 基于RISC-V的异构内存控制器

本文提出一种非层次的DRAM/NVM缓存架构,如图4所示. 传统的层次架构(图4(*a*))中数据必须从NVM拷贝到DRAM才能被CPU访问,平行架构(图4(*b*))下DRAM和NVM都做主存并且冷热数据在NVM和DRAM之间周期性进行迁移(NVM和DRAM中无相同的副本),而本文设计的非层次缓存架构(图4(*c*))中,DRAM有部分作主存,而另外部分作NVM的缓存,CPU可以旁路DRAM缓存直接访问NVM中的冷数据,同时NVM中的热数据会被缓存到DRAM中. 尽管平行架构和非层次缓存架构都能充分利用DRAM和NVM的带宽,非层次缓存架构对于读密集型应用比平行架构具有更大的性能优势,因为DRAM容量满时,平行架构需要把DRAM中的部分页面写回NVM才能容纳更热的NVM页,而非层次缓存架构对于未修改的页面可以直接回收,不需要写回NVM中. 平行架构的页交换往往处于应用访存的关键路径上,并且NVM写操作的开销远大于对DRAM页的读写,因此对于读密集型应用,DRAM中的页面替换会带来更大性能开销.

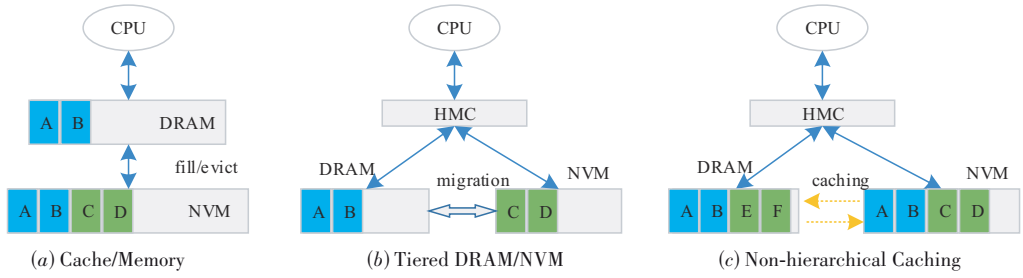


图4 DRAM/NVM的三种内存架构对比

我们基于RISC-V在异构内存控制器(HMC)中设计了非层次缓存架构,实现了CPU和操作系统不感知的异构内存架构动态重构机制,以及热点数据动态管理机制. 该架构核心要解决的问题是如何把NVM中的

热页从内存控制器迁移到更快的DRAM页,并建立NVM和DRAM物理页之间的映射关系. 迁移后原本访问NVM中热页的内存请求将被重定位到DRAM页,从而实现CPU和操作系统无感的页迁移和访存重映射,避免传统操作系统进行页迁移时带来的巨大一致性开销(如片上缓存强制flush及TLB shootdown等). 为此,我们基于RISC-V设计了支持非层次缓存架构的异构内存控制器(RISC-V based Heterogeneous Memory Controller,RVHMC). RVHMC主要由4个部分组成:(1)访存行为监测和分析模块;(2)可重构异构内存架构模块,主要包括部分DRAM和NVM物理页之间的动态重映射机制;(3)CPU和操作系统的无感页面迁移模块;(4)异构内存模拟模块.

如图5所示,访存行为动态监测和分析模块对RISC-V处理器的访存行为进行识别和收集,统计分析收集到的访存踪迹,生成页面访存热度. 根据页面信息表中的页面访存信息,本模块对页面未来的访存行为进行预测,划分出读写频度高的热页面和不常被访问的冷页面. 根据上层应用的访存行为,RISC-V处理器访问Double Data Rate(DDR)内存页面或NVM内存页面,期间异构内存会在层次架构、平行架构之间切换. 系统初始化时,地址映射表为空. 随着程序的运行,NVM页面变为热页. 此时将该热页缓存到DDR内存中,同时添加地址映射表表项. 当地址映射表满而访问新的热页时,调用替换策略,将旧页中热度最低的页面换出,将新页的相关信息存入地址映射表.

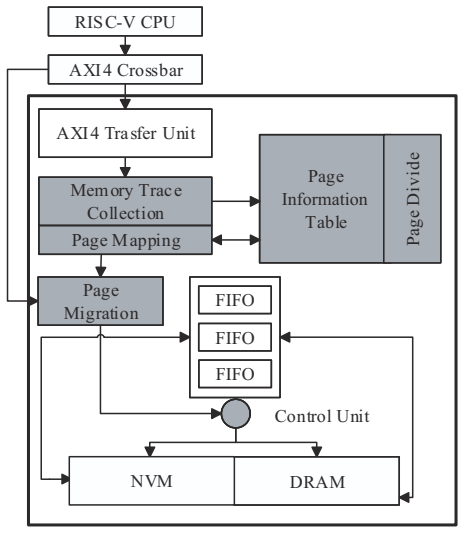


图5　基于RISC-V的异构内存控制器架构

CPU和操作系统无感页面迁移模块则是根据可重构异构架构发送的外部信号进行页面迁移. 通过设计数据迁移缓冲区First In First Out(FIFO),暂存需要迁移的热点数据. 在控制器内部设计实现逻辑模块,代替CPU发出指令,将FIFO缓冲区中的热点数据迁移至目标地址,实现CPU无感的页迁移.

由于目前没有可用的NVM模组进行实验,我们利用部分DRAM来模拟NVM的性能,设计并实现了异构内存模块. 通过对DRAM注入延迟以模拟NVM,该模块模拟实现了一个基于硬件的异构内存模拟环境,为上述三个部分提供实验支撑.

所设计的异构内存控制器基于lowRISC-Chip片上系统实现,下面将从访存行为动态监测和分析设计、可重构的异构内存管理机制设计、CPU无感的页面迁移设计以及异构内存模拟4个方面阐述RVHMC的设计与实现方案.

3.1　基于RISC-V的访存行为监测和分析

基于RISC-V的动态访存行为监测和分析模块(RISC-V based Memory Trace Analysis,RMTA)的主要功能是采集应用的访存踪迹、统计分析应用访存行为、并通过对页面未来行为的预测实现冷热页面分类. 该模块主要由3个部分组成:(1)访存踪迹收集;(2)页面访存信息表;(3)冷热页划分. 访存踪迹收集组件通过侦听RISC-V处理器总线,对RISC-V处理器的访存行为进行识别和收集,并将每个内存页面的访问频度存储到页面访存信息表中;页面访存信息表可为冷热页划分提供必要的决策信息;冷热页划分组件根据该表所存储的页面读/写频度等信息对页面未来的访存行为进行预测,从而将内存页面划分为读/写频度较高的热页面和不常被访问的冷页面.

应用访存踪迹收集组件是保障整个系统正常运行的前提,精准、高效的获取RISC-V处理器访存行为的关键在于准确识别每次Advanced eXtensible Interface4(AXI4)协议的突发握手以及每次独立的数据传输过程. 由于每次数据传输行为都对应一次AXI4突发传输,因此可设计算法监测AXI4总线状态,每当总线进入读状态或写状态时,将会触发应用访存踪迹搜集组件的访存计数功能. 在该状态下,对RISC-V处理器总线中的数据传输有效信号和数据传输就绪信号进行监测,当它们同时为高电平时,表示发生了一次有效数据传输,则使计数器自增一次.

为了完整记录页面访存信息,综合考虑数据局部性和时间局部性原理,本文设计了一个新的数据结构用于保存收集来的页面信息. 此数据结构包括6个字段,包括有效位valid,物理页面地址addr,读次数rc,写次数wc,生存时间time以及热页标志hot. 具体信息如表1所示.

应用访存过程中生成的页面访问信息均使用表1所示的数据结构缓存在片上寄存器组中,每当应用访存踪迹收集组件记录了一条新的页面信息,则通过下

表1 页面信息表项字段组成

名称	长度	描述
valid	1 bit	标识该表项是否有效,高电平有效
addr	20 bits	页面信息所属的页面地址
rc	16 bits	页面的读访问次数
wc	16 bits	页面的写访问次数
time	16 bits	页面从上次被访问到现在经历的时间
hot	1 bits	标识页面是否为热页,高电平有效

面所介绍的策略进行更新和维护. 由于利用监测RISC-V处理器总线收集应用访存踪迹将会产生大量的页面访问信息,而异构内存控制器中的可用空间有限,因而如何管理大量的页面记录信息是本系统维持高效性、鲁棒性的关键. 在地址映射方面,采用全相连映射的方式对页面信息记录进行管理. 在具体的实现中,可以使用与寄存器数量相等的多位比较器进行并发页面信息查找. 首先比对有效valid标志位,若该位为0说明本次写入的是脏数据,可以直接写入. 若该页面有效位为1,则表示该页面有效,进而对比页面地址是否相等,若相等则进行相应的页面更新操作. 当比较后发现当前寄存器组并没有寄存器的有效位为0,也不存在对应的页面信息时,说明此时片上寄存器组已满,需要进入页面淘汰流程. 由于页面信息表所存储的是页面热度信息,因此页面淘汰策略既需要具备页面存储管理的功能,也需要兼顾冷热页分类和淘汰的需要.

目前常用的cache结构页面淘汰算法有FIFO(先进先出)、Least Recently Used(LRU,最近最久未使用)算法等. LRU算法作为使用频率较高的淘汰算法,通过"如果数据最近被访问过,那么将来被访问的几率也更高"的核心预测思想,呈现出良好的性能,然而LRU往往需要频繁的交换数据,在具体实现中使用链表等移动开销小的数据结构和算法. 本系统基于lowRISC-Chip硬件平台实现,使用寄存器组存放LRU元数据信息,频繁的交换数据会造成严重的性能开销和浪费,同时硬件设计的并发性也没有被利用上. 来自罗切斯特大学的Li团队提出了一种新型的可变大小缓存,该缓存使用程序行为的统计信息来最大化缓存性能[14]. 某个数据被访问的可能性每经历一次缓存缺失则下降一些,基于这种数据淘汰思想的缓存被他们称为lease cache. 经过实验证明,lease cache的性能表现与LRU cache不相上下,在部分数据局部热度较高的应用中其表现甚至优于LRU cache. 基于上述观点,综合考虑硬件平台特性和筛选冷热页面的功能目标,本设计每经历一定数量的时钟周期就对页面信息表中最久未使用的表项进行更新,使得页面未来被访问的可能性随着时钟周期的增加(未访问该页面的前提下)而逐渐降低. 这种设计不仅充分利用了硬件平台的多器件并发性质,而且符合页面记录信息对冷热页筛选的功能,同时能达到甚至优于传统LRU cache的页面淘汰准确性以及性能.

基于页面访存信息根据访存热度对页面进行冷热页面筛选划分,其主要目标是通过页面的访问历史信息对页面未来的访存行为进行合理的预测. 本文充分考虑全局页面信息和页面访存信息存活时间,提出了一种的冷热页面筛选策略. 此策略利用FPGA硬件平台的并发优势,考虑多个页面访问因素来预测页面的访存行为,并对页面冷热度进行准确的划分. 此划分方案有以下几个主要原则:(1)页面写次数作为热度值计算的主要影响因素;(2)剩余生存时间作为热度值计算的重要影响因素;(3)页面读次数作为热度值计算的次要影响因素. 其中页面的剩余生存时间来反映了页面的活跃程度. 为了反映页面的历史热度值随时间衰减的特性,从而让最近的访存在热度计算中具有更大的权重,我们预先定义了页面的最大生存时间T(常量),即在此时间范围内页面为活跃的,否则热度值为0. 通过统计页面的生存时间即表1中的time,即可计算出页面的剩余生存时间. 页面的剩余生存时间越长,表明该页面的热度值(或活跃度)也越高.

如图6所示,每当一次AXI4突发传输结束,应用访存信息收集逻辑部件会发送一条本次突发传输的访存信息,以此触发热页划分流程. 根据发送来的内存页面地址检索页面访存信息表,将新的页面信息更新至已存在的页面信息表项中,若不存在该页面信息项,则新建表项. 读取页面访存信息表项中的相关信息,包括写次数wc,以及生存时间time,读次数rc. 基于上文的原则,热度值hotness将根据wc,rc的权重以及剩余生存时间(T–time)进行计算:

$$\text{hotness}=(\alpha\times\text{wc}+(1-\alpha)\times\text{rc})\frac{\text{time}}{T-\text{time}}$$

其中α表示写计数的权重(文中设为0.7),T表示页面的最大生存时间. 若该页面计算的热度值超过了预设阈值,则将其划分为热页,修改热页标志hot为1,否则划分为冷页. 此外,当3.3节中所介绍的页面淘汰策略生效时,说明对应的内存页面已经很长时间未被访问,同样被划分为冷页. AXI4的数据总线宽度为32 bits[15],传输一个4 KB大小的内存页面需要256个时钟周期,而计算热度大约需要20个时钟周期. 考虑到热度计算使用专门的硬件,所以计算热度的时间完全可以被下一次NVM页面访问延迟所隐藏,而不会产生额外的延迟开销.

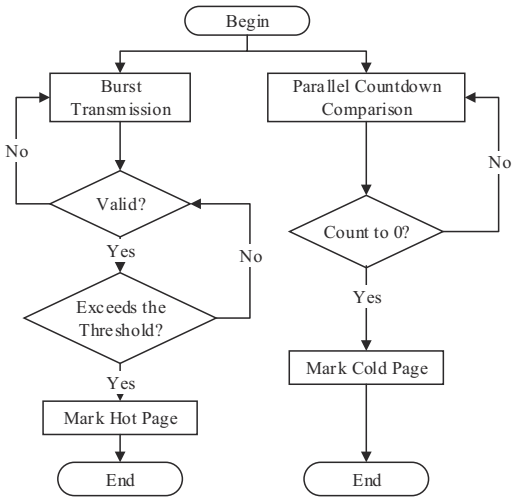


图6　冷热页面划分流程

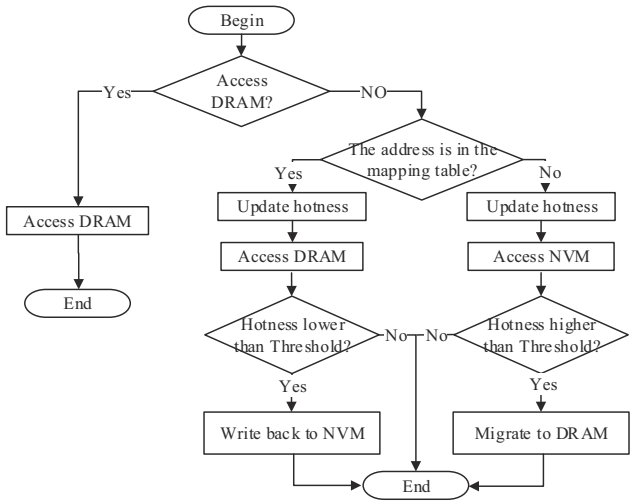


图7　访问内存总体流程

3.2　可重构异构内存架构

3.2.1　异构内存访存流程

当前对异构内存的主要研究,主要基于平行架构或者层次架构,例如英特尔傲腾持久内存若要同时支持两种异构内存架构,只能重启内存,并切换内存模式. 本文设计的可重构异构内存架构实现了内存系统在平行和层次架构之间进行动态重构,并提供了应用层的配置接口,可由应用自定义DDR/NVM的层次化配比比例,主要由4个部分组成:(1)地址映射表的构造;(2)访存地址重定向;(3)地址映射表的替换策略;(4)DDR缓存页的分配策略. 图7显示了可重构异构内存访问内存系统的总体流程.

地址映射表如图8所示,每个地址映射表表项64位. 由于32位系统中CPU对内存的寻址一般是32位,故在地址映射表表项中NVM、DDR的偏移地址为20位,页面地址为12位,页面大小为4 KB. 有效位Valid为1表示该表项有效,NVM页面缓存在DDR中;否则该表项无效. 脏数据位为1表示缓存页面被修改过. 为了辅助伪LRU算法完成地址映射表表项的替换,计数位预留了11位,还有11位表示缓存页的热度信息. 地址映射表的表项数可配置,以此实现自定义DDR/NVM层次化配比比例. 由于伪LRU算法需要的元数据信息存储开销不大,可以直接保存在异构内存控制器中.

为简化硬件逻辑,地址映射表采用全相联映射. 在查找NVM页面是否缓存在DDR页面中时,首先查找地址映射表中Valid位为1的表项,判断该页面的物理块号是否和表中某个表项相同. 如果相同则获取该表项的DDR物理块号,加上该页面偏移得到缓存该NVM页面的DDR页面物理地址. 在将DRAM作为缓存使用时,系统需要在访问NVM前查找地址映射表,但因为缓存本身可以避免访问NVM页面的高开销,大部分情况下缓存带来的性能收益大于查找地址映射表带来的延时. 此外,因为查找地址映射表和访问NVM是可以同时进行的,所以查找地址映射表不会为访问NVM带来额外的时间开销.

CPU发出访存指令,根据访存指令分离出访存的物理地址,如果这个物理地址是DDR地址范围则直接访问,该DDR页面作为平行架构和NVM一样存放数据,而不作为NVM页面的缓存. 否则该物理地址是NVM地址,需要判断是否缓存在DDR中以进行相应的处理. 如果CPU访存的是NVM地址,查找地址映射表,判断该NVM页面是否作为热页缓存在DDR中,由于是全相联映射,故遍历一遍地址映射表. 如果在表中则表明是层次架构,该页曾作为热页缓存在DDR中. 根据应用访存行为的动态感知模块的信息:

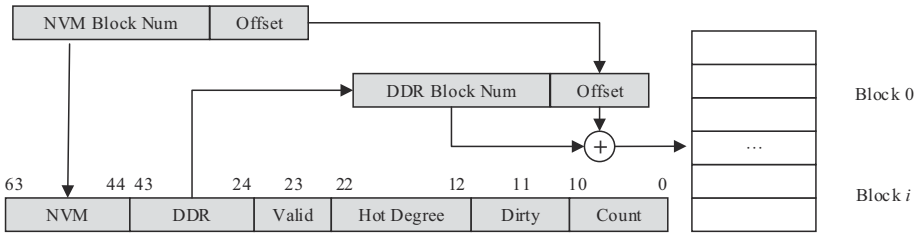


图8　地址映射表

(1)如果此时该页的热度低于阈值,那么就调用DDR-NVM直通的页迁移机制模块.将缓存的DDR地址和寻址的NVM地址传给页迁移机制模块后,DDR缓存页面的数据将被写回到NVM页面.同时移除地址映射表的对应表项,移除的方法很简单,只需要将Valid位置0,下次遍历地址映射表时不会访问这个表项.

(2)如果该页的热度不低于阈值,表明该页当前仍然是较"热"的,就更新该表项中用于类LRU算法的计数位,即页面热度.

如果CPU访存的是NVM地址,且查找地址映射表,遍历一遍未发现NVM物理块号和该页面相同的表项,则表明该NVM页目前是冷页,未缓存在DDR中.根据应用访存行为的动态感知模块的信息:(1)如果经过本次读取该页的热度高于阈值,那么就需要替换DRAM中不那么"热"的页;(2)如果该页的热度仍旧低于阈值,那么就直接读取该NVM页面.

根据伪LRU算法得到DRAM缓存中需要被替换页面DDR页面地址和NVM页面地址,调用DDR-NVM直通的页迁移机制模块.将这两个地址传给DDR-NVM直通的页迁移机制模块后,将DDR缓存页面的数据写回到NVM页面,同时修改该表项的NVM地址和计数位.之后再次调用DDR-NVM直通的页迁移机制模块将新的NVM热页面迁移到DRAM页.

传统的替换策略使用较多的是LRU算法,但是单独的LRU算法应用在内存地址映射表的替换中,存在一定的局限性.尽管在热点数据频繁访问时,LRU有很高的命中率,但是在偶发性、周期性的数据访问时,LRU的命中率则急速下降,无法保证热页一直缓存在DDR中.偶发的数据缓存到了DDR中,替换了即将访问的周期性数据,导致热页频繁地被换进换出,造成不必要的时间、能耗开销,导致系统整体的性能下降.

本文采用的替换策略是对传统LRU算法的改进.将超过设定阈值的热点页面赋予不同的优先级并放置到三个对应优先级的队列,对每个队列的页面采用LRU算法进行替换.考虑到突发性数据只是在某段集中的时间访问次数很高然后便不再访问,仅通过访问次数设置优先级会导致优先级队列中的某些突发数据在整个内存运行过程中一直占据着表项.甚至在突发数据过多时,突发数据占据了整个表项,导致表项失去了存在的意义.因此简单的将访问次数最高的数据存放到高优先级队列不妥当.

为了解决这个问题,需要调整页面的热度,添加表项在优先级队列的降级和升级机制.将高优先级的队列的"冷"了的突发数据逐次降级,先降到中优先级队列,再降级到低优先级队列,再从地址映射表移除.将周期性且"热"了的数据,从低优先级队列,先升级到中优先级队列,再升级到高优先级队列,避免了时间局部性高的热页数据频繁的换入换出,从而提高内存系统性能和缓存命中率.为此,当地址映射表表项的计数位大于等于该表项的热度的八分之一时,将计数位置0,热度位暂时减半,谨慎的降级热度.根据上层应用的访存局部性特征模块提供页面的访问次数信息,如果在表项还在队列中时被再次访问则将热度重置为该页面的历史访问次数,表明是周期性数据,否则是突发性数据,其将随着程序的运行慢慢从地址映射表移除.热度优先级队列被记录在内存控制器中,因为优先级本身是由内存控制器进行计算,将优先级队列存储在内存控制器中避免了访问内存的开销,降低了计算延迟.

3.2.2 混合架构动态转换过程

由上可见,架构动态转换过程是双向的:系统初始化时DRAM和NVM是平行架构,随着应用程序的运行,NVM中的热页将迁移到DRAM页上,逐步形成了本文提出的非层次DRAM/NVM缓存架构;而当DRAM满时,部分作为缓存的DRAM页将被收回,DRAM作为缓存的容量将变少,逐步向平行架构转化.

在异构内存架构动态切换时,仅仅需要内存控制器维护DRAM页和NVM页之间的动态映射关系,而两种页面之间的一致性维护实现也比较简单.从平行架构切换到层次架构过程中,只需在映射表中逐步建立DRAM页和NVM页间的映射关系,在系统运行过程中会有一次在地址映射表中查询DRAM缓存是否命中的开销.从层次架构切换到平行架构的过程很简单,只需将映射表中的表项逐条删除并将变脏的DRAM缓存页写回NVM页,并不会引起其它一致性维护的开销(如操作系统实现页迁移过程引起的片上缓存强制flush及TLB shootdown开销).因此,在层次架构下DRAM作为缓存的意义在于:若DRAM缓存页没有变脏,它们在被淘汰时是不用重新写回到NVM中,从而减少了数据写回NVM的开销以及对NVM的磨损.总体上讲,DRAM同时具备两种身份,一部分作为平行架构中的主存,一部分作为层次架构的缓存来加速NVM中热页访存,并且作为缓存的部分可以在平行架构和层次架构间动态切换.访问DRAM中作为缓存的部分仅仅增加一次查询映射表是否命中的开销(FPGA并行查询复杂度为$O(1)$),而作为平行架构中主存的部分访问延迟则没有变化.异构内存控制器的性能优势主要体现在两点:当访存局部性好时,可以将大量落在NVM页上的频繁访存转移到DRAM缓存页上,从而提高访存性能,并且冷数据的访问是旁路DRAM缓存的,并不会增加额外的页面拷贝开销;而当访存局部

性差且内存足迹大时,又可以切换到平行架构以提供更多内存容量.

3.3 CPU无感的页迁移机制设计

CPU无感的内存页迁移机制(下文简称页迁移机制)绕过CPU,由内存控制器发出指令控制内存中页面的迁移. 当有热点数据,可将其以页为单位,通过迁移缓冲区,完成迁移.

页迁移机制主要由FIFO缓冲区和控制内存读写模块协同实现. FIFO缓冲区是通过Xilinx的FIFOIP核定制FIFO,利用读写使能信号控制FIFO读写,可暂存迁移数据;控制内存读写模块是内存控制器内部的逻辑模块,通过控制Memory Interface Generator(MIG)核控制内存读写. 其内部共有六个不同的状态,根据信号进行跳转,与FIFO模块一起完成数据的迁移. 通过这两个模块的协同合作,可实现对异构内存的页面进行迁移. 下面将从FIFO缓冲区数据迁移实现和控制MIG读写模块设计两方面,阐述页迁移机制的实现过程.

FIFO缓冲区数据位宽为64位,深度为1 024. 包含输入的时钟和复位信号、输入数据din、输出数据dout、写使能wr_en、读使能rd_en、空标志empty、满标志full. 从内存中读取的热点数据通过din输入到FIFO中,再通过dout传出写入内存指定地址. 利用读写使能信号,控制数据在FIFO中传输,故FIFO中数据的迁移包括读、写两个部分.

首先是内存读场景,步骤1:迁移数据首先进入读地址状态,判断源地址是否属于NVM范围内;步骤2:若属于NVM范围则拉高FIFO写使能信号,使FIFO处于写状态,并记录写入的延迟,否则直接进入步骤3;步骤3:当FIFO不为满时,让读取的热点数据传入到FIFO. 步骤4:循环步骤3,直至读完整个页面.

然后是内存写场景,步骤1:进入写地址状态,判断目标地址是否属于NVM范围内;步骤2:若属于NVM范围则拉高FIFO读使能信号,使FIFO处于读状态,并记录读取的延迟,否则直接进入步骤3;步骤3:当FIFO不为空时,将数据从FIFO读出,写入目标地址;步骤4:循环步骤3,直至写完整个页面.

控制MIG读写模块是根据AXI4协议设计,可代替CPU发出访存指令,共有6个状态. 通过将不同的信号拉高进入不同的状态,页面迁移的状态机如图9所示.

(1)起始状态(IDLE):起始状态,等待外部迁移信号signal. 当signal=1时,表示有热点数据需要迁移,此时跳转到RADDR状态;当signal=0时,无热点数据需要迁移,继续等待.

(2) 发送读地址状态(Read ADDRess, RADDR):实际的读过程分为两步,首先在RADDR状态时将源地址送到线上,然后等到RDATA状态时,根据源地址读取数据. 由于要读出整个页面的数据而实际每次只能读取64位的数据,因此必须反复读取页面数据到FIFO中,直至页面读完. 外部的源页面地址信号为27位地址,需要将该数据所在的整页进行迁移. 地址前15位为页号,后12位为页内偏移. 已知Nexys4DDR板上DDR2的内存为128 MiB,地址线为27位,故其最小寻址单位为1 byte. 每次读取64 bit数据,地址增加8,继续读下一范围数据. 逐次自增,直到读完整个页面地址范围的数据.

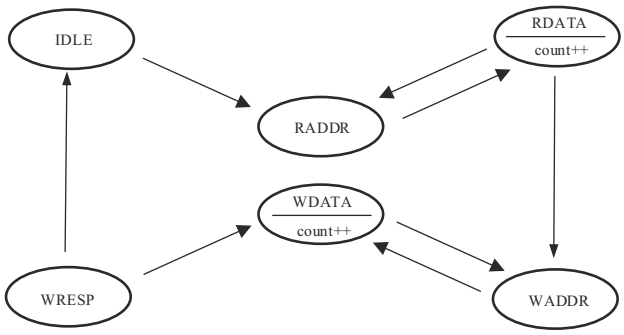


图9　页迁移机制状态机

(3) 发送读数据状态(Read DATA, RDATA):进入RDATA状态后,MIG将读出的数据写入FIFO中. read_count记录读数据的次数,每读一次就加一,同时将写次数write_count清零. 由上述知,最小寻址单位为1 byte,每次读取64 bit,页面大小为4 096 bytes则一共需要读取512次才能将整个页面读完. 因此当read_count小于511(从0开始)时,需要跳回RADDR状态,继续读取页面数据到FIFO;若read_count等于511,整个页面的数据已读到FIFO中,跳转到WADDR状态,准备将数据写入内存指定位置.

(4) 发送写地址状态(Write ADDRess, WADDR):同读地址状态一样,进入WADDR状态后则将目标地址送到写地址线上,目标地址同样也是获取前15位目标页号位后补全形成. 写完后目标地址自增加8. 不断自增目标地址并写入数据,直到将页面内的全部数据写入指定页地址范围.

(5) 发送写数据状态(Write DATA, WDATA):当进入WDATA状态. MIG将FIFO中数据读出并写入指定地址. 同样每次写入数据write_count加1,read_count清零. 若write_count小于511时,需要跳回WADDR状态,继续从FIFO中读取数据,写入到内存中;若write_count等于511,整个页面的数据已从FIFO中读出,并写入内存指定页面中,跳转到WRESP状态.

(6) 发送写回应状态(Write RESPonse, WRESP):当写数据全部传输完成,将写计数write_count及读计数read_count全部清零,表示此次页面迁移已经成功

结束.

到此为止,通过FIFO缓冲区模块和控制MIG读写模块,实现了CPU旁路的页面迁移. 由FIFO作为内存控制器中的缓冲区暂存数据,由控制MIG读写模块将热页写入内存. 两个模块相互配合,完成热点数据以页为单位的迁移.

3.4 异构内存系统模拟

之前对异构内存的研究多使用软件模拟器,但模拟器对真实环境的模拟不够准确且在实际应用中会有一定差距[16]. 此外,由于目前没有可商用的NVM模组使用,为了实现硬件管理的异构内存系统原型,本文提供一种硬件层的异构内存模拟方案.

本实验使用的FPGA开发板型号为Nexys4 DDR,其板上内存型号为镁光DDR2 MT47H64M16HR-25E,读时延RL=5(12.5 ns),写时延WL=4(10 ns). 为了模拟异构内存,对地址范围进行了逻辑划分,将板上内存一部分作为DRAM使用,一部分仿真为NVM使用,DRAM/NVM各占50%. PCM(一种非易失性内存)时延参数则参考了Lee等人[17]论文中的参数设置,其读时延为55.5 ns,写时延为150 ns. 为了模拟NVM的读写延迟,设定NVM范围内的数据读时延为DRAM的4倍,写时延为DRAM的15倍.

为了模拟NVM的延迟,我们使用延迟注入模块来模拟NVM的写入延迟. 在所设计的内存控制器中判断src_read/dst_write是否属于NVM地址范围,若在NVM范围内则延迟处理,让该地址范围的数据进行延迟注入;若不在NVM范围内,则直接读取或写入数据. Verilog HDL(硬件描述语言)中有阻塞赋值和非阻塞赋值两种赋值方式. 阻塞赋值可理解为语句的顺序执行,先执行完前面的步骤再执行后面的步骤. 因此阻塞赋值可以有一定的延迟. 本延迟模块使用的就是连续寄存器阻塞赋值,将逻辑路径延长. 每次阻塞赋值一个寄存器需要一个时钟周期,每次时钟周期为5 ns. 延迟读写即增加寄存器赋值次数,不断延长时间.

4 实验评估

本文使用基于FPGA的硬件平台对RVHMC的性能和功耗进行评估. 本系统基于RISC-V架构进行FPGA设计,硬件平台选择lowRISC-Chip.具体来说,实验套件选用Vivado 2018.3,开发板型号为XC7A100 TCSG324-1. 配套使用Digilent Nexys4 DDR型号的FPGA开发板,内存型号为MT46H 64M 16HR-25E. 实验基于Ubuntu 16.04 LTS操作系统.

4.1 性能评估

实验使用的微基准测试程序为memtester4. 0,该测试程序是Cazabon等lowRISC-Chip开发人员集成到该硬件平台的内存测试程序. 其针对lowRISC-Chip片上系统做了对应的内存空间分配与耦合,通过RISC-V交叉编译工具链即可编译为RISC-V架构的二进制机器码文件.

内存测试程序memtester4.0首先将运行累加、累乘、异或运算等计算程序对内存系统的正确性进行验证. 然后运行内存读写程序,通过对某片内存的集中访问造成读写频度较高的内存区域和读写频度较低的内存区域(一般为中间地址访问频度高,低地址与高地址访问频度低),以此模拟大部分应用的访存行为. memtester4.0访存密集区域集中在内存中段地址,对低地址以及高地址访问频度较低. 另外,将访存热点定位至NVM或DRAM上以此对异构内存控制器带来的性能提升进行测试.

实验分为两组:一组是在原生lowRISC上的平行架构(无架构重构和热页迁移)进行内存读写测试;一组是在添加了异构内存控制器RVIMC的(可重构架构)lowRISC上运行读写访存测试. 两组测试均运行memtester4.0的6个测试程序,然后对比运行时间进行性能和功耗的评估. 6个测试程序读写操作总数相同,但读写比例和访存分布不同,访存特征如表2所示.

表2 测试程序应用访存特征 单位:%

测试编号	读访问比例	写访问比例	DRAM访问比例	NVM访问比例
mt1-1	65.08	34.92	70.113	29.887
mt1-2	65.08	34.92	50.431	49.569
mt1-3	65.08	34.92	30.667	69.333
mt2-1	34.92	65.08	70.113	29.887
mt2-2	34.92	65.08	50.431	49.569
mt2-3	34.92	65.08	30.667	69.333

性能测试方面,本实验通过FPGA片上布置的计时器对应用运行时间进行时钟周期粒度的计时. 可对两种系统架构下应用的运行时间进行比较. 当测试程序开始运行时,集成在FPGA上的计时器则同步开始运行,当程序运行完毕时结束计时,由此可获得测试程序的运行时间. 由图10可见,RVHMC可为应用运行带来约43%的性能提升,主要原因在于实现的异构内存控制器可以发现NVM上的热页面,并在线迁移到DRAM上,形成了热点数据的缓存架构,大幅减少了热数据在NVM上的分布,提高了访存性能. 当测试集读写比例为65.08∶34.92时,分布在NVM上的数据比例越高时,RVHMC带来的性能提升也越高. 当测试集为写密集时(mt2),RVHMC带来的性能提升也越明显.

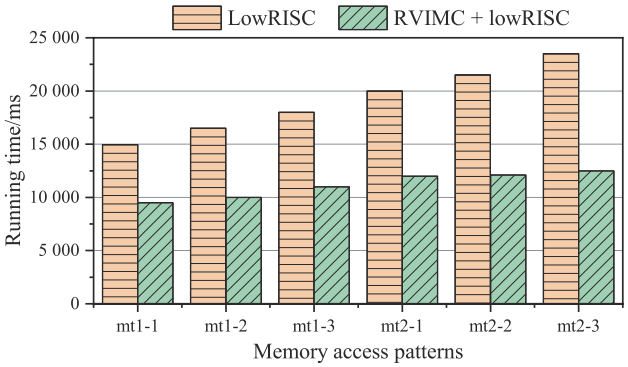


图 10　测试程序运行时间对比

在能耗方面,本实验通过 vivado 工具集提供的片上运行温度模拟程序,对应用运行时芯片的温度进行评估. 当程序运行完毕时,可反映温度变化情况,以此对比系统功耗. 由图 11 可以发现 RVHMC 可降低约 2 ℃的片上温度. 由于数据写入 NVM 的操作功耗比写入 DRAM 要高很多,RVHMC 把大量的热点数据从 NVM 迁移到 DRAM,从而降低了 NVM 的写功耗. 由图 11 可见写密集应用功耗降低的比较明显.

此外,我们在所设计的可重构异构内存架构下对 SPEC CPU2017 和 PBBS 中的部分测试程序进行了性能测试. 由于开发板的内存容量有限,各测试程序仅仅执行 10^6 条指令,其 IPC 和运行时间分别如图 12 和图 13 所示. 其中 Flat 代表没有热点数据迁移的平行架构,Hierarchical 代表纯粹的缓存架构,即所有 NVM 上的数据都会先取到 DRAM,RVHMC 代表本文设计的可重构异构内存架构,即仅仅 NVM 中判定为热页的页才会加载到 DRAM 缓存中,CPU 可以旁路 DRAM 访问 NVM 中的大多数冷页.

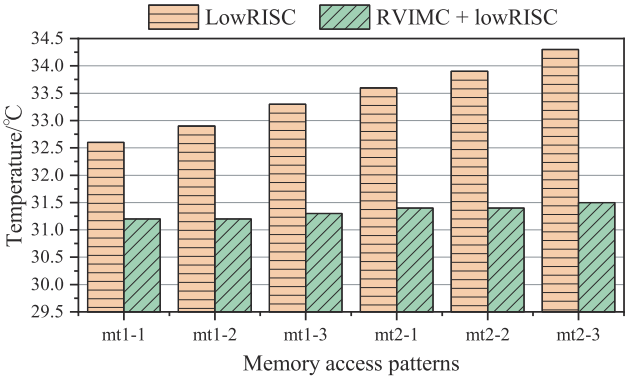


图 11　FPGA 片上温度比较图

从图 12 和图 13 可以发现,对于 MSF、Setcover、KNN、BFS 这类图应用,由于极大多数数据具有稀疏访存的特性,局部性差,平行架构由于无需把冷数据加载

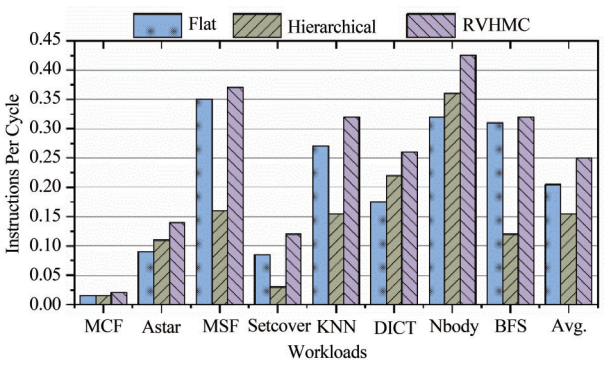


图 12　测试程序性能(IPC)对比

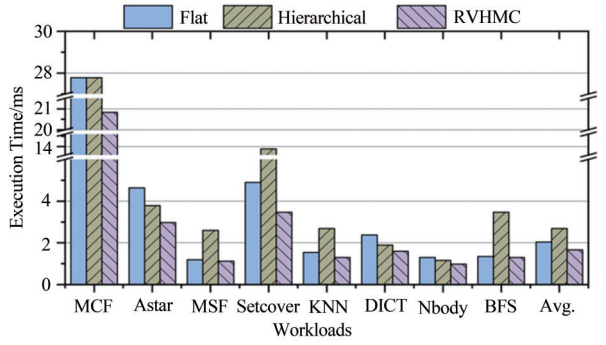


图 13　测试程序运行时间对比

到 DRAM 缓存,避免了数据移动和 DRAM 缓存置换的开销,比层次架构具有明显的性能优势. 而对于 Astar、DICT、Nbody 等具有一定的访存局部性的应用,层次架构则表现出更好的性能. 而本文提出的可重构异构内存架构无论针对那种类型的应用,都具有较好的性能表现,相对于平行架构和层级架构分别提升了 IPC 为 22% 和 61%,最高达 1.56 倍和 4 倍,验证了该设计的有效性.

此外,我们测试了不同应用程序在几种内存架构下的功耗,如图 14 所示. 由于不同应用的功耗数值差距非常大,我们对把同一应用在层次架构和可重构异构内存架构的数值相对平行架构进行了规格化处理. 由于 MSF、Setcover、KNN、BFS 这类图应用大部分数据访存比较稀疏,层次架构需要把 NVM 上的所有数据都加载到 DRAM 缓存,造成了大量不必要的数据移动,产生了更高的功耗,而对于局部性较好的应用如 Astar、DICT、Nbody,则可以利用 DRAM 缓存避免功耗较高的 NVM 写操作,从而有效降低了系统功耗. 所设计的可重构异构内存架构由于仅仅把热点数据缓存到 DRAM 中,一方面避免了不必要的页面搬移,同时也避免了大量的 NVM 写操作,其功耗相比平行架构和层次架构分别降低 28% 和 37%.

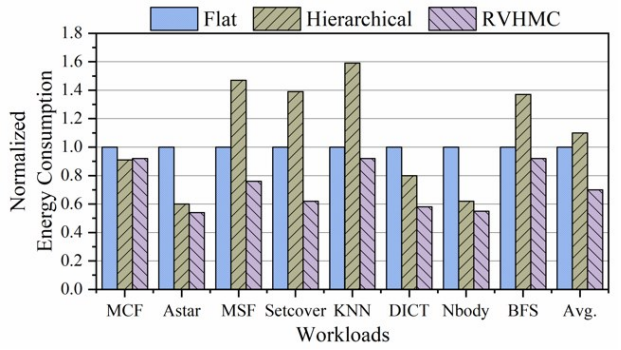


图14 测试程序功耗对比,相对平行架构进行规格化处理

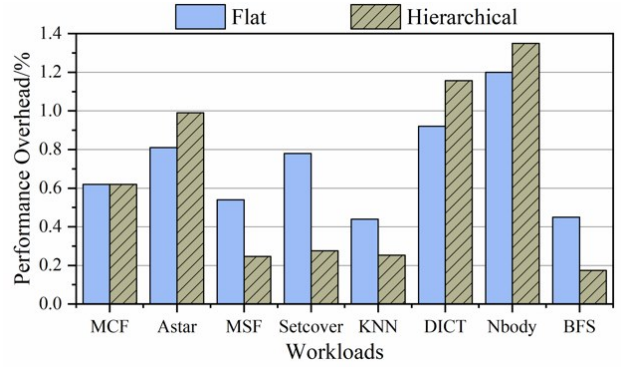


图15 相对于平行架和层次架构,可重构异构内存架构对程序的性能损耗

4.2 系统开销

我们进一步测试了所设计的可重构异构内存架构对上述测试程序带来的性能损耗,如图15所示. 性能开销主要包括:页面热度监测开销、地址映射表的查询开销、发生架构重构时DRAM中作为缓存页的映射消除开销. 相对于无地址重映射的平行架构,绝大多少测试程序的性能开销低于2%. 对于访存局部性相对较好的应用如Astar、DICT、Nbody,由于相对较多的热页监测、热页迁移和NVM-DRAM页映射维护开销,给应用程序带来的性能损耗相对较多,然而这部分的开销相对于热页的高效缓存机制带来的性能收益是微乎其微的.

接下来分析异构内存控制器需要的额外存储资源开销,主要包括用于页面热度监测的页面信息表,用于构造非层次化缓存架构的地址映射表,以及用于页迁移的FIFO队列. 页面信息表中每个页面的计数器、状态信息等共需要9字节. 由于大部分应用的工作集大小随时间窗口变化,我们只需要监测最近被访问的页面即可,对于长期未访问且热度值降为0的页面可以从本表中逐出. 为此,我们设置页面信息表的条目数为10 000,总计需要90 KB的存储空间. 地址映射表每个条目为8字节,假定DRAM中用于缓存的页面数为100 K个,则最多需要预留800 KB的存储空间用于地址映射. 用于页迁移的FIFO缓冲区数据位宽为8字节,深度为1 024,因此仅仅需要8 KB的存储空间. 因此,异构内存控制器总共需要90+800+8=898 KB的额外存储空间. 该存储开销不会因为NVM容量的增大而动态变化,是可以接受的.

5 相关工作

目前的异构内存系统主要有层次化和平行两种架构. 在层次化架构的异构内存系统中,DRAM作为NVM的缓存[5,6],DRAM缓存的地址空间对于操作系统不可见. Stealth-Persist[18]设计了新的内存控制器,选择性地利用NVM来备份缓存在DRAM中的页面,以确保这些数据可以获得持久性保证. HeMem[19]异步地管理DRAM缓存和NVM中的数据,通过CPU事件而不是扫描页表对内存访问进行采样,从而来监视应用程序内存使用情况,减小了异构内存管理的开销. AutoTiering[20]发现在同时配备DRAM和英特尔傲腾持久内存的非统一内存访问(Non-Uniform Memory Access, NUMA)系统中,放置页面时内存介质相比NUMA内存位置对性能的影响更为显著,并设计了一套内存页面的放置,回收和迁移方案. 层次化架构需要用硬件来支持DRAM缓存和NVM之间的数据替换. 当DRAM缓存缺失时,层次化异构内存系统的总访存时延将显著增大. 在平行架构的异构内存系统中,DRAM和NVM统一编址,两者均作为主存使用[3,4,7]. 但是,由于目前NVM和DRAM在读写性能、能耗等方面的差异,往往通过操作系统将频繁访问的页面从NVM迁移到DRAM中. OAM[21]通过静态代码检测自动对数据对象进行放置和动态迁移. MULTI-CLOCK[22]根据页面访问的时间局部性和频率决定页面的放置策略,从而提高访存效率并降低页迁移开销. Pei等人[23]提出了一种基于双向哈希链表的异构内存页迁移机制(THMigrator),通过降低写PCM的次数来减少异构内存系统的综合访问延时. 然而,热页迁移的优化策略需要对页面级的访存行为进行监控,增加了内存子系统的开销. T J Ham等人提出了一种多内存控制器分离的异构混合硬件架构[24],将CPU片上内存控制器作为统一的主控制器,DRAM和PCM器件分别由片外从属控制器管理. MOCHA[25]以256字节的粒度管理DRAM缓存,避免了NVM的写放大. HiNUMA[26]在数据迁移时考虑到非统一内存访问节点对延迟和带宽的影响,以降低混合内存系统中内存访问的开销. NHC[27]通过平衡平行架构中DRAM和NVM的访存带

宽来提高性能. MT²[28]利用多种硬件和软件技术来协同监控不同类型内存访问的实时带宽,并基于cgroup机制提供了一个高效的内存带宽动态调节框架. HAMS[29]将NVDIMM和超低延迟闪存(ULL-Flash)聚合到一个单一的大内存空间中并优化了数据路径,从而降低繁重的软件堆栈对NVDIMM性能的影响. 这些异构内存系统架构设计都需要特定的软硬件支持,无法在单一系统内对不同架构进行动态重构,因而缺乏对多样化应用的动态适配能力.

随着RISC-V指令集的涌现,基于RISC-V的新型计算机体系架构得到了众多的关注. Leidel等人使用RISC-V软件模拟器来构造了Stake[30],一款支持RISC-V处理内核和新型存储设备模拟的基础架构平台,但软件方式失真度高、仿真速度慢. Zhang等人基于RISC-V为异构内存系统开发设计了一套全系统硬件平台—MEG[31],通过软硬结合的方式,将RISC-V片上系统与新型存储介质HMC进行了耦合设计,但不能运行在DRAM/NVM异构内存系统上. 在目前已有的RISC-V片上系统中,仍没有面向DRAM/NVM异构内存系统的RISC-V内存控制器硬件设计. RVHMC针对新型指令集架构RISC-V,基于FPGA硬件平台设计并实现了异构内存控制系统,将RISC-V片上系统与DRAM/NVM进行了耦合设计,并为应用运行带来了较高的性能提升,同时降低了系统能耗.

6　总结与展望

RVHMC是一种基于RISC-V的异构内存控制器,其通过采集分析RISC-V片上系统的访存指令,实现页面映射、页面迁移等功能,进而基于内存页面的访存行为对内存页面进行在线迁移,并可灵活地实现平行和层次异构内存架构之间的转换,从而提升应用的访存性能. 实验表明,基于FPGA实现的异构内存控制器RVHMC可以实现高效的异构内存管理. 相较于其他实现,其显著特点是基于新兴指令集RISC-V架构,通过硬件方式实现,内存管理效率高,避免了纯软件方式带来的巨大性能开销. 未来,RVHMC可以结合RISC-V操作系统,对相关内存管理策略进行进一步的优化,实现软硬件协同的可重构异构内存管理系统.

参考文献

[1] MANDELMAN J A, DENNARD R H, BRONNER G B, et al. Challenges and future directions for the scaling of dynamic random-access memory (DRAM) [J]. IBM Journal of Research and Development, 2002, 46(2/3): 187-212.

[2] BAHAR TALUKDER B M S, KERNS J, RAY B, et al. Exploiting DRAM latency variations for generating true random numbers[C]//2019 IEEE International Conference on Consumer Electronics (ICCE). Piscataway: IEEE, 2019: 1-6.

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

[4] DHIMAN G, AYOUB R, ROSING T. PDRAM: A hybrid PRAM and DRAM main memory system[C]//Proceedings of the 46th Annual Design Automation Conference. New York: ACM, 2009: 664-669.

[5] MEZA J, CHANG J C, YOON H, et al. Enabling efficient and scalable hybrid memories using fine-granularity DRAM cache management[J]. IEEE Computer Architecture Letters, 2012, 11(2): 61-64.

[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 Annual International Symposium on Computer Architecture. New York: ACM, 2009: 24-33.

[7] 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. New York: ACM, 2017: 1-10.

[8] 王鹏, 邹彬, 刘金枝, 等. 基于Xilinx型FPGA系统单粒子效应评估方法研究[J]. 电子学报, 2022, 50(11): 2716-2721.
WANG P, ZOU B, LIU J Z, et al. Study on single event effect evaluation method based on Xilinx FPGA system[J]. Acta Electronica Sinica, 2022, 50(11): 2716-2721. (in Chinese)

[9] 田春生, 陈雷, 王源, 等. 面向FPGA的布局与布线技术研究综述[J]. 电子学报, 2022, 50(5): 1243-1254.
TIAN C S, CHEN L, WANG Y, et al. Review on technology of placement and routing for the FPGA[J]. Acta Electronica Sinica, 2022, 50(5): 1243-1254. (in Chinese)

[10] 王诲喆, 唐丹, 余子濠, 等. 开源芯片、RISC-V与敏捷开发[J]. 大数据, 2019, 5(4): 50-66.
WANG H Z, TANG D, YU Z H, et al. Open-source chip, RISC-V and agile development[J]. Big Data Research, 2019, 5(4): 50-66. (in Chinese)

[11] MOHSENI Z, REVIRIEGO P. Reliability characterization and activity analysis of lowRISC internal modules against single event upsets using fault injection and RTL

simulation[J]. Microprocessors and Microsystems, 2019, 71: 102871.

[12] MATH S S, MANJULA R B, MANVI S S, et al. Data transactions on system-on-chip bus using AXI4 protocol [C]//2011 International Conference on Recent Advancements in Electrical, Electronics and Control Engineering. Piscataway: IEEE, 2011: 423-427.

[13] YANG J, KIM J, HOSEINZADEH M, et al. An empirical guide to the behavior and use of scalable persistent memory[C]//Proceedings of the 18th USENIX Conference on File and Storage Technologies. New York: ACM, 2020: 169-182.

[14] LI P C, PRONOVOST C, WILSON W, et al. Beating OPT with statistical clairvoyance and variable size caching[C]//Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2019: 243-256.

[15] ILLANA J M. Design of an AXI-SDRAM interface IP in a RISC-V Processor[D]. Barcelona: Universitat Politècnica de Catalunya, 2020.

[16] 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(4): 15-34.
WANG X Y, LIAO X F, LIU H K, et al. Big data oriented hybrid memory systems[J]. Big Data Research, 2018, 4 (4): 15-34. (in Chinese)

[17] LEE B C, IPEK E, MUTLU O, et al. Architecting phase change memory as a scalable dram alternative[C]//Proceedings of the 36th annual international symposium on Computer architecture. New York: ACM, 2009: 2-13.

[18] ALWADI M, KOMMAREDDY V R, HUGHES C, et al. Stealth-persist: Architectural support for persistent applications in hybrid memory systems[C]//2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA). Piscataway: IEEE, 2021: 139-152.

[19] RAYBUCK A, STAMLER T, ZHANG W, et al. HeMem: Scalable tiered memory management for big data applications and real NVM[C]//Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles. New York: ACM, 2021: 392-407.

[20] KIM J, CHOE W, AHN J. Exploring the design space of page management for multi-tiered memory systems[C]// Proceedings of the 2021 USENIX Annual Technical Conference (ATC'21). Virtual Event: USENIX, 2021: 715-728.

[21] LIU H K, LIU R S, LIAO X F, et al. Object-level memory allocation and migration in hybrid memory systems[J]. IEEE Transactions on Computers, 2020, 69(9): 1401-1413.

[22] MARUF A, GHOSH A, BHIMANI J, et al. MULTI-CLOCK: Dynamic tiering for hybrid memory systems [C]//2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA). Piscataway: IEEE, 2022: 925-937.

[23] 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.
PEI S W, JI Y F, SHEN T M, et al. Migration mechanism of heterogeneous memory pages using a two-way Hash chain list[J]. Scientia Sinica (Informationis), 2019, 49(9): 1138-1158. (in Chinese)

[24] HAM T J, CHELEPALLI B K, XUE N, et al. Disintegrated control for energy-efficient and heterogeneous memory systems[C]//Proceedings of the 2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA). New York: ACM, 2013: 424-435.

[25] CHI Y, YUE J H, LIAO X F, et al. A hybrid memory architecture supporting fine-grained data migration[J]. Frontiers of Computer Science, 2024, 18(2): 182103.

[26] DUAN Z H, LIU H K, LIAO X F, et al. HiNUMA: NUMA-aware data placement and migration in hybrid memory systems[C]//2019 IEEE 37th International Conference on Computer Design (ICCD). Piscataway: IEEE, 2019: 367-375.

[27] WU K, GUO Z H, HU G Z, et al. The storage hierarchy is not a hierarchy: optimizing caching on modern storage devices with orthus[C]//Proceedings of the 19th USENIX Conference on File and Storage Technologies (FAST'21). Virtual Event: USENIX, 2021: 307-323.

[28] YI J F, DONG B C, DONG M K, et al. MT^2: Memory bandwidth regulation on hybrid NVM/DRAM platforms [C]//Proceedings of the 20th USENIX Conference on File and Storage Technologies (FAST'22). Santa Clara: USENIX. 2022: 199-216.

[29] ZHANG J, KWON M, GOUK D, et al. Revamping storage class memory with hardware automated memory-over-storage solution[C]//Proceedings of the 48th Annual International Symposium on Computer Architecture. New York: ACM, 2021: 762-775.

[30] LEIDEL J D. Stake: A coupled simulation environment

for RISC-V memory experiments[C]//Proceedings of the International Symposium on Memory Systems. New York: ACM, 2018: 365-376.

[31] ZHANG J L, ZHA Y, BECKWITH N, et al. MEG: A RISCV-based system emulation infrastructure for near-data processing using FPGAs and high-bandwidth memory [J]. ACM Transactions on Reconfigurable Technology and Systems, 2020, 13(4): 1-24.

作者简介

靳晓忠　男,1997年出生于内蒙古呼和浩特.2019年获得华北电力大学学士学位.目前正在华中科技大学攻读博士学位,研究兴趣主要包括数据去重和存储系统.

E-mail: xzjin@hust.edu.cn

刘海坤　男,1981年出生于湖北随州.华中科技大学教授,博士生导师,国家级人才入选.2012年于华中科技大学计算机系统结构专业博士毕业,目前主要从事内存型存储系统、异构计算、机器学习等方向的研究.先后主持多项国家自科基金项目、国家重点研发计划课题以及10多项企业横向合作项目.

E-mail: hkliu@hust.edu.cn

赖　皓　男,1997年出生于广东韶关.2021年获华中科技大学计算机科学与技术硕士学位,主要研究方向为计算机系统结构、FPGA.

E-mail:laihao@bigo.sg

毛伏兵　男,1985出生于湖北仙桃.硕士生导师,2018年获新加坡南洋理工大学计算机科学与工程博士学位.主要研究方向为,系统软件与体系结构、FPGA/AISC物理设计、最优化算法、机器学习等.先后参与了国家自然科学基金、十一五国家“核高基”科技重大专项和新加坡国防科技项目、校企联合项目.

E-mail: fbmao@hust.edu.cn

张　宇　男,1987年出生于湖南,华中科技大学计算机科学与技术学院教授,博导,2016年博士毕业于华中科技大学计算机科学与技术学院,主要研究高性能计算、体系结构和系统软件.

E-mail: zhyu@hust.edu.cn

廖小飞　男,1978年出生.博士,华中科技大学教授、博士生导师,国家杰出青年基金获得者,“万人计划”科技创新领军人才入选者,中国计算机学会分布式计算与系统专委会主任,华中科技大学科学技术发展院院长.主要从事大数据处理、系统软件、新型体系结构等研究工作.主持或参与多项863重点研发计划、国家自然科学基金项目.在重要期刊和会议上发表100余篇论文,获国家自然科学二等奖1项、国家科技进步二等奖1项、教育部技术发明一等奖2项、教育部自然科学一等奖1项,获2017年度CCF-IEEE CS青年科学家奖.中国电子学会会员编号:E190010157M.

E-mail: xfliao@hust.edu.cn

金　海　男,1966年出生.博士,华中科技大学教授、博士生导师,长江学者特聘教授,国家杰出青年基金、CCF“王选奖”获得者,国家“万人计划”科技创新领军人才.中国计算机学会副理事长/会士,IEEE Fellow,华中科技大学“大数据技术与系统国家地方联合工程研究中心”主任,“服务计算技术与系统教育部重点实验室”主任,十四五“先进计算与新兴软件”国家重点研发计划专家组组长,湖北省计算机学会理事长,教育部“长江学者和创新团队发展计划”创新团队学术带头人.主要从事新型体系结构、并行与分布式计算、大数据处理等研究工作.中国电子学会会员编号:E190005379S.

E-mail: hjin@hust.edu.cn