

文章编号:1007-5321(2020)02-0103-07

DOI:10.13190/j.jbupt.2019-091

一种面向边缘计算的混合内存系统

孙浩^{1,2}, 陈岚¹, 郝晓冉¹, 刘晨吉^{1,2}, 倪茂¹

(1. 中国科学院微电子研究所, 北京 100029; 2. 中国科学院大学, 北京 100049)

摘要: 针对物联网智能终端的低功耗需求,提出了一种基于内存控制器扩展的低功耗混合内存系统. 使用动态随机存储器和相变存储器构成混合内存结构,通过在内存控制器中添加迁移控制模块对混合内存进行管理. 设计了一种改进的双队列算法,筛选出相变存储器中写请求较多的内存页面,并通过地址映射模块和迁移控制模块将写请求较多的页面从相变存储器迁移到动态随机存储器中,规避相变存储器写操作的缺陷,从而实现对低功耗混合内存系统的性能优化. 仿真结果表明,与动态随机存储器构成的内存系统相比,混合内存系统的功耗延时积平均降低了43.9%,在面向边缘计算的应用场景中具有一定的可行性.

关键词: 边缘计算; 混合内存; 内存控制器; 页面迁移

中图分类号: TP333.1

文献标志码: A

A Hybrid Memory System for Edge Computing

SUN Hao^{1,2}, CHEN Lan¹, HAO Xiao-ran¹, LIU Chen-ji^{1,2}, NI Mao¹

(1. Institute of Microelectronics, Chinese Academy of Sciences, Beijing 100029, China;

2. University of Chinese Academy of Sciences, Beijing 100049, China)

Abstract: In order to satisfy the low power requirements of IoT devices, a low power hybrid main memory system composed of dynamic random access memory and phase change memory is proposed. The hybrid main memory is managed by the hybrid memory control modules which are added to the memory controller. An improved dual queue algorithm is proposed to filter out memory pages with more write requests from phase change memory. Because of the poor write performance of phase change memory, the page migration module migrates the selected pages from phase change memory to dynamic random access memory. And then the updated page address is recorded in the address mapping table for subsequent memory access. Experimental results show that compared with the traditional main memory composed entirely of dynamic random access memory, the hybrid main memory reduces the power-delay product by 43.9% on average.

Key words: edge computing; hybrid memory; memory controller; page migration

在云计算场景下,物联网终端的数据需要传输到云中心,在云端完成数据处理.但是从边缘向云端传输数据存在延时较高、带宽资源不足等问题,无

法满足实时应用场景的需求.边缘计算技术可以将云端的智能处理能力延伸到物端,提升物联网终端的数据处理效率,减少对云端的依赖,因此物联网终

收稿日期:2019-05-28

基金项目:国家物联网与智慧城市重点专项对接项目(Z181100003518002);北京市科技专项项目(Z171100001117147)

作者简介:孙浩(1993—),男,博士生.

通信作者:陈岚(1968—),女,研究员,博士生导师, E-mail: chenlan@ime.ac.cn.

端设备需要具有更强的数据处理和存储能力^[1]. 随着数据的存储从云向边缘转移, 给物联网终端的存储能力带来极大挑战. 由于物联网终端需要具备长的续航能力, 而传统的动态随机存储器(DRAM, dynamic random access memory) 由于其功耗和密度上的劣势, 已经难以满足边缘计算技术的需求.

新型非易失性存储器的出现, 为内存提供了新的解决方案. 一些研究者针对混合内存架构开展了相关研究, 扩展内存架构 SSDRAM^[2] 利用大容量的固态硬盘作为内存扩展, 以运行时库的方式提供应用程序接口, 使用户能够透明访问大容量扩展内存. 基于访存行为地址映射机制的混合内存系统^[3] 利用内存系统物理地址重映射机制实现对混合内存的管理. 与 DRAM 相比, 相变存储器(PCM, phase change memory) 具有高存储密度、低功耗、非易失等优势, 可作为物联网终端内存的候选方案. 而且 PCM 可以按位读写, 是最有可能替代 DRAM 的新型非易失性存储器. 但是 PCM 读写不对称, 与 DRAM 相比, PCM 的写延时较长, 导致内存访问时间增加, 系统性能下降. 在功耗方面, 对 PCM 进行写操作比读操作的功耗高. 另外, PCM 能承受的写操作次数有限, 在使用寿命上有劣势.

由上述分析可知, 在物联网终端中单独使用 DRAM 或者 PCM 作为内存都有不足之处. 为了满足边缘计算技术中的低功耗需求, 笔者使用 DRAM 和 PCM 构建了混合内存架构, 在内存控制器中添加扩展模块, 采用改进的双队列算法对混合内存页面进行管理, 通过对页面进行迁移, 规避了 PCM 写操作的劣势, 充分发挥 DRAM 和 PCM 两种介质的优势. 仿真结果验证了混合内存架构下算法的有效性.

1 内存控制器扩展

为克服 PCM 写功耗高和写次数有限的缺点, 可将 DRAM 和 PCM 统一编址, 构成平面混合内存结构. 针对混合内存系统, 需要设计新的内存管理策略, 规避 PCM 写操作的缺陷, 充分发挥混合内存结构的优势. 研究人员针对 DRAM 和 PCM 构成的平面混合内存结构的页面管理算法开展了相关研究. Bock 等^[4] 通过软硬件配合的方式设计了一种多页面并发迁移机制, 但是该方案的实现需要同时兼顾操作系统和硬件结构, 实现较为复杂. Ramos 等^[5] 使用内存控制器来统计混合内存页面的访问频率,

并建立地址映射表. 根据页面访问频率对其进行分类, 访问频率越高, 页面级别越高, 将高级别的页面从 PCM 迁移到 DRAM 中. 但是该方案使用的多队列算法不能迅速筛选出 PCM 中需要迁移的写操作密集的页面. 面向混合内存结构的末级缓存管理策略^[6-7], 通过对传统页面置换算法进行改进, 可有效减少 PCM 上的写操作, 但是同时带来缓存命中率下降等问题, 严重影响系统性能.

现有的研究大多面向通用计算机系统, 对于面向边缘计算的嵌入式系统, 硬件和软件资源都非常有限, 所以一个开销小且实现简单的方案更适合嵌入式应用. 面向边缘计算的物联网终端主要针对嵌入式应用, 而嵌入式系统具有面向特定应用及资源受限的特征. 在嵌入式程序中, 大部分写操作通常集中于少数变量, 若这部分写操作发生在 PCM 中, 不仅功耗较高, 同时也会加快 PCM 的磨损, 影响 PCM 的使用寿命. 因此, 如果能够将这部分写操作转移到 DRAM 上, 可以有效避免 PCM 的缺陷, 改善系统整体性能.

为了减小对整个系统的影响, 使混合内存系统能够迅速融入传统计算机系统结构中, 笔者所做的工作主要通过内存控制器实现. 通过对内存控制器进行扩展, 集成混合内存系统管理模块, 实现对混合内存系统的控制和优化. 基于硬件结构实现 DRAM 与 PCM 的页面管理机制能够在对操作系统透明的前提下完成系统性能的优化. 如图 1 所示, 在内存控制器中加入地址映射模块、迁移控制模块和迁移算法模块.

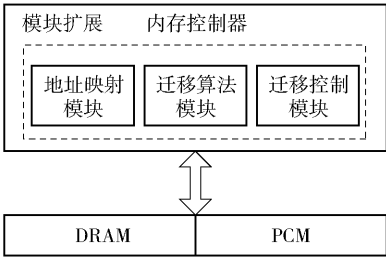


图 1 混合内存系统架构

1.1 地址映射模块

地址映射模块由内存控制器维护的地址映射表构成. 从总线传入的访存请求, 首先根据访存地址判断是否被迁移过, 若没有经过迁移, 则按照正常访存流程进行; 若访存地址被迁移过, 则会查找内存控制器中的地址映射表, 访问映射后的新地址. 地址映射模块可确保页面被迁移后, 当请求再次访问被

迁移的页面时能够得到正确的数据。

1.2 迁移控制模块

迁移控制模块控制页面迁移的时机。当内存页面满足迁移条件时,由迁移控制模块启动迁移机制。在页面迁移和交换过程中,会先将页面放入寄存器中,此时若有请求需要访问正在迁移的页面,可直接对寄存器中的页面进行操作,这样可以有效减少迁移操作带来的延时。迁移控制模块将页面从 PCM 迁移到 DRAM 中,需要从 DRAM 选取一个页面交换回 PCM 中。为了防止将已经从 PCM 迁移到 DRAM 中的页面再替换回 PCM,需要在页面初次迁移时,对页面进行标记。在 DRAM 中选择交换页时,若页面未被标记过,则可直接交换;若页面已被标记,则选择相邻的未被标记的页面作为交换页。

1.3 迁移算法模块

对 PCM 中的页面进行写操作功耗较高、延时较大,而且会影响 PCM 的寿命。为了避免对 PCM 中的页面进行过多写操作,需要将频繁被写的页面从 PCM 迁移到 DRAM。为了提升系统性能,该模块需要设计合理的页面迁移算法。

2 多队列算法

基于排序的页面放置策略(RaPP, rank-based page placement)使用内存控制器实现了多队列(MQ, multi-queue)算法^[5],其基本思想是:通过维护多个最近最少使用(LRU, least recently used)队列来筛选访存热点页面。算法通过统计 PCM 中页面的访问频率,判断页面是否应该被迁移。当页面的访问请求次数增加,达到升级阈值,则将页面从当前队列删除,升级到更高一级的队列头部。当页面被升级到迁移队列时,则触发页面迁移操作。因此,迁移队列的设置决定了迁移的阈值,迁移队列设置越高,迁移阈值就越高。为了防止高队列数据永远不被淘汰,当数据在指定的时间内没有被访问,需要降低优先级,将数据从当前队列删除,加入低一级队列的头部。多队列算法原理如图 2 所示。

RaPP 策略是针对通用处理器等高性能应用场景而设计的混合内存管理机制,算法较为复杂,占用硬件资源较多。高性能与嵌入式场景所运行的应用程序访存特征有较大差别,因此需要针对嵌入式应用特征对 RaPP 策略进行改进。为了给物联网嵌入式应用找到合适的算法方案,使用了嵌入式测试用例^[8-9]对 RaPP 策略进行了分析研究。PCM 写操作

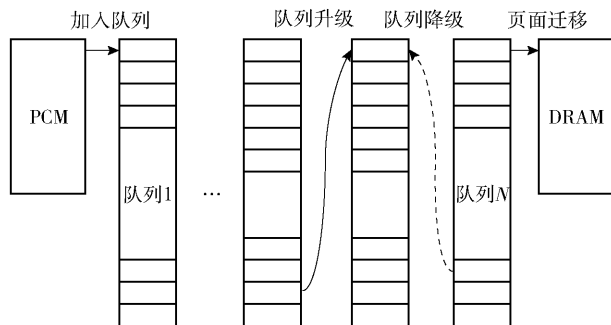


图 2 多队列算法原理

的功耗和延时性能较差,是影响混合内存结构性能的主要因素,所以测试统计了不同迁移阈值条件下发生在 PCM 上的写操作次数。

测试先将迁移队列设置为第 9 个队列,页面需要从第 1 个队列经过 8 次升级,直到被升级到第 9 个队列时才会被迁移。这种情况下,页面迁移阈值过高,不能及时筛选出访存热点页面,使 PCM 上写操作数较多。通过更改迁移队列设置,可以降低迁移阈值,使 PCM 中的访问频率较高的页面更容易被迁移到 DRAM 中。随着迁移阈值的降低,PCM 中的写操作次数下降。

当迁移队列设置为第 5 个队列时,PCM 上的写次数已经达到相对较少的水平。继续降低迁移阈值,则会导致迁移频率的增加,而 PCM 上的写次数降低却不明显。这是因为降低页面迁移阈值,虽然可以使 PCM 上的写操作数降低,但是会使多队列算法中的页面降级机制的降级空间变小,从而使页面降级机制效果变差。如在极端情况下,PCM 中的页面一旦被访问就会被迁移到 DRAM 中,这样会使页面迁移过于频繁,而页面迁移操作本身也会降低系统性能。由于缺少降级机制的筛选,被迁移的页面也不一定是真正的访存热点页面。另外,现有的降级机制需要遍历页面的生存时间参数,消耗较多的资源,所以现有的多队列算法并不适用于物联网嵌入式应用。因此,所设计的迁移算法需要在资源受限的条件下既能够迅速筛选出 PCM 中写请求密集的页面,又要防止冗余的页面迁移。

3 改进的双队列算法

提出了一种改进的双队列(IDQ, improved dual queue)算法。改进的双队列算法维护 2 个队列:一个队列为历史记录队列,该队列使用先进先出

(FIFO, first input first output)策略管理;另一个队列为等待迁移队列,该队列使用最近未使用(NRU, not recently used)策略管理.改进的双队列算法不再关注读请求访问的页面,专注于写请求所访问的页面,从而提升页面迁移的效率.当PCM中的页面被写请求访问后,首先会被加入历史记录队列中.若页面一定时间内未被再次访问,则被淘汰出FIFO队列,队列的长度决定了页面的生存时间.历史记录队列中的页面在一定时间内被再次访问后,会被移动到等待迁移队列.当等待迁移队列中的页面达到设定的迁移阈值,会触发迁移操作,由迁移控制模块迁移到DRAM中,并记录在地址映射模块.若等待迁移队列中的页面一定时间内未被再次访问,则放弃迁移,将该页面移出队列.改进的双队列算法中,已经进入等待迁移队列的页面不再进行升级,仅增加一个标记位记录访问次数.

等待迁移队列的淘汰机制使用NRU算法.队列中的每一个页面增加一个标记位.当页面从历史记录队列中放入等待迁移队列时,标记位被置0.页面再次被写请求访问后,标记位置为1.当标记位为1的页面被写请求访问,则交给迁移控制模块进行迁移.如果NRU队列已满,则淘汰搜索标记位为0的页面.若所有标记位都为1,则将全部页面标记位刷新为0.

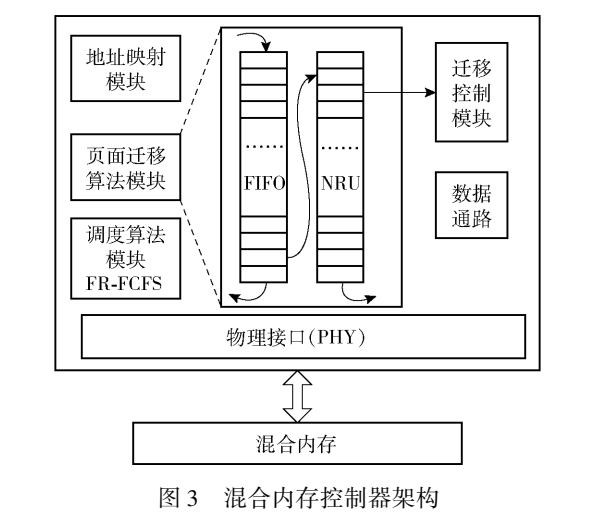
在多线程算法中,为实现降级机制,设置了内存页面的生存时间,每当有页面访问时,页面的生存时间都会进行更新.进行访存操作时,会检查每个队列的生存时间参数.如果超过设定阈值,则会进行页面降级操作.这一过程需要遍历页面的生存时间参数,降低了算法效率,硬件结构实现难度大,而且在嵌入式应用场景下不能有效实现页面降级.所以,利用FIFO队列和NRU队列的淘汰机制替代多线程算法的页面降级机制.混合内存控制器架构及页面迁移算法原理如图3所示.

在混合内存控制器中,页面迁移算法模块使用改进的双队列算法,当页面达到迁移阈值时,即交给迁移控制模块进行迁移.页面迁移完成后,将已迁移的页面记录在地址映射模块.

4 算法测试结果与分析

4.1 测试条件

测试用例使用嵌入式基准测试集MiBench^[8]和多媒体基准测试集MediaBench II^[9].选取



mpeg2dec、mpeg2enc、cjpeg、djpeg、FFT、stringsearch、basicmath和patricia进行测试(分别编号为A~H),覆盖了物联网嵌入式应用中的视频、图像、通信等场景.测试平台使用gem5模拟器^[10],利用NVMain^[11]模拟内存器件的性能.表1列出了在NVMain中配置的DRAM和PCM的延时和功耗参数^[11-12].其中tRCD表示行寻址到列寻址的延迟时间,tCL表示发送列地址到内存后的响应延迟,tRAS表示行激活命令与预充电命令之间的最小时间间隔,tWR表示数据从行缓冲保存到内存中所需的时间.功耗参数取自NVMain模拟器对内存操作所建立的模型^[11].

表 1 内存仿真参数

DRAM		PCM	
延时/ns	功耗/nJ	延时/ns	功耗/nJ
tRCD:18	Eop:1.08	tRCD:44	Eop:0.002
tCL:15	Ewr:1.02	tCL:7.5	Ewr:1.68
tRAS:42	Erd:3.40	tRAS:25.5	Erd:0.08
tWR:15	Eref:38.55	tWR:300	

由于90%的物联网嵌入式应用所需的CPU频率低于1GHz^[13],所以在gem5中将CPU主频设置为1GHz.DRAM和PCM容量比例设置为1:3,内存总容量设置为4GB.

在嵌入式处理器指令系统选取方面,因为RISC-V开源指令系统具有低功耗、低成本、可模块化、面积小、速度快等优点,适合应用于物联网、边缘计算等新兴领域^[14],已成为当下的研究热点,因此在gem5中选用RISC-V指令系统.

4.2 测试与分析

由图4和图5可知,使用混合内存架构后,内存

功耗与纯 DRAM 结构相比平均下降 44.8%, 其原因是充分利用了 PCM 不需要刷新的优势, 大幅降低了内存系统的静态功耗。但是由于 PCM 写延时较大, 在降低功耗的同时导致系统延时较纯 DRAM 结构平均增加 31.3%。

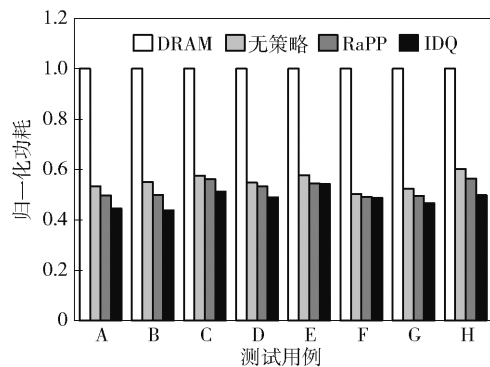


图4 功耗对比

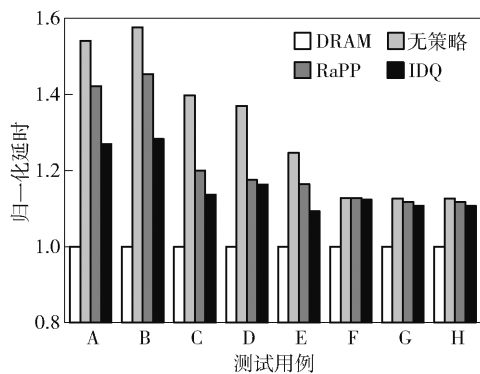


图5 延时性能对比

为了减小系统的性能损失, 将多队列算法应用于混合内存控制器中, 参数设置参考 RaPP 策略^[5]。由于多队列算法将访存较多的内存页面从 PCM 迁移至 DRAM, 减少了对 PCM 的访问次数, 从而降低 PCM 对系统性能的影响。与纯 DRAM 结构相比, 使用了多队列算法的内存系统功耗平均下降 47.6%。系统延时平均增加 22.2%。

提出的改进双队列算法在多队列算法的基础上, 将超时页面淘汰工作改为由 FIFO 算法和 NRU 算法完成, 降低了算法复杂程度, 减少了生存时间参数遍历带来的额外延时开销, 使系统性能进一步提升。另外, 改进的双队列算法仅统计 PCM 中写操作密集的页面, 减少了冗余的页面迁移。使用所提出的改进双队列算法, 与纯 DRAM 结构相比, 内存系统功耗平均下降 51.5%, 系统延时增加了 16%。

与传统的 DRAM 内存结构相比, 所提出的混

合内存系统虽然可以平均降低功耗 50% 以上, 但是由于 PCM 自身缺陷仍然无法完全克服混合内存对系统性能造成的损失。为了能够综合评估混合内存系统性能, 采用功耗延时积(功耗与延时的乘积)对混合内存系统性能进行评价。功耗与延时是内存系统性能的 2 个关键参数。单纯用功耗或者延时来评价混合内存系统都是不准确的, 因为混合内存系统在功耗降低的同时, 会因为 PCM 较高的写延时导致内存系统延时上升。所以, 使用功耗延时积来评价混合内存系统在功耗和延时两方面的综合性能。功耗延时积越小, 表示混合内存系统的整体性能越好。

如图 6 所示, 在考虑混合内存带来的性能损失后, 本文所提出的方案与 DRAM 内存相比, 功耗延时积平均降低 43.9%, 证明了使用 IDQ 算法管理的混合内存结构在综合性能上具有一定优势。

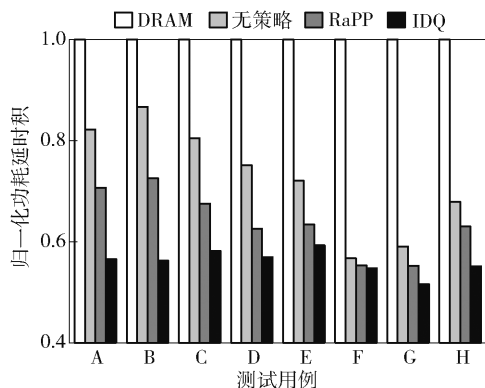


图6 功耗延时积性能

上述分析结果均在特定的内存总容量(4 GB)和内存比例(DRAM: PCM = 1:3)下得到。嵌入式应用需求的多样性导致了其对内存系统需求的多样性, 本文对不同内存比例、不同内存总容量下, 混合内存系统的性能进行了比较分析。

在总内存容量为 4 GB 的条件下, 分别测试 DRAM 与 PCM 比例为 1:3、1:7 和 1:15 时的内存系统功耗、系统延时以及功耗延时积, 实验结果如图 7 所示。

测试结果表明, 减少 DRAM 所占比例, 可以降低内存系统功耗, 但是增加了系统延时。这是因为减少 DRAM 容量可以有效降低 DRAM 的刷新功耗, 但是由于更多内存访问使用了 PCM, 使系统性能有所下降。因此在实际应用中, 在实时性要求较高的场景, 可以提高 DRAM 所占的比例以保证系统性能; 在对功耗性能要求较高的场景, 可以通过减少

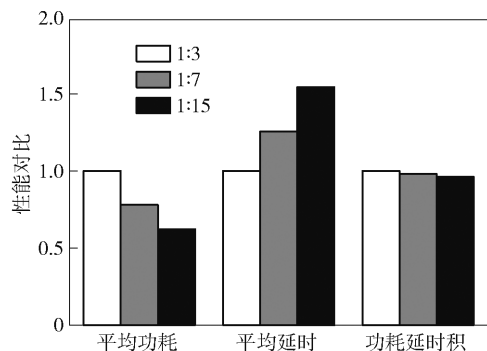


图7 不同混合比例性能对比

DRAM 在混合内存中所占比例使内存系统功耗进一步降低。

测试了不同总内存容量下内存系统的功耗,设置混合内存的比例为 DRAM: PCM = 1: 3。实验结果如图 8 所示。

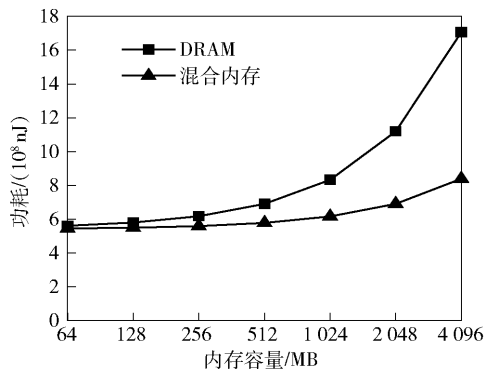


图8 不同内存容量性能对比

测试结果表明,随着总内存容量增大,混合内存结构在降低内存系统功耗方面的优势也随之增大。在内存容量为 4 GB 时,混合内存系统功耗仅为传统 DRAM 内存功耗的 50% 左右;当内存总容量下降到 64 MB 时,混合内存功耗与 DRAM 内存功耗基本保持一致。其原因是随着内存容量的减小,DRAM 内存刷新功耗的占比也随之减小,使得混合内存存在降低功耗方面的效果随之减弱。因此,混合内存结构在大内存容量需求的应用场景中优势更加明显。

5 结束语

针对物联网嵌入式应用场景提出了一种混合内存系统,设计了改进的双队列算法进行页面迁移。该算法能够迅速筛选出 PCM 中写操作密集的页面,同时还避免了冗余的迁移。研究表明,与传统的完全由 DRAM 构成的内存系统相比,系统功耗延时积平均降低 43.9%。此外,对不同内存混合比例

和不同内存总容量下混合内存系统的功耗与性能进行了深入分析。分析结果表明,通过调整 DRAM 内存所占比例,可以实现对系统性能和功耗水平的灵活调整,以满足不同应用场景需求。

参考文献:

- [1] Shi Weisong, Cao Jie, Zhang Quan, et al. Edge computing: vision and challenges[J]. IEEE Internet of Things Journal, 2016, 3(5): 637-646.
- [2] 王力玉, 陈岚, 郝晓冉, 等. 利用 SSD 和 DRAM 构建的扩展内存架构[J]. 西安电子科技大学学报, 2017, 44(3): 144-150.
Wang Liyu, Chen Lan, Hao Xiaoran, et al. Memory-extended architecture based on the SSD and DRAM[J]. Journal of Xidian University, 2017, 44(3): 144-150.
- [3] 王强, 陈岚, 郝晓冉. 一种基于访存行为地址映射机制的混合内存系统[J]. 小型微型计算机系统, 2014, 35(6): 1201-1206.
Wang Qiang, Chen Lan, Hao Xiaoran. Hybrid memory system using memory access-aware remapping mechanism[J]. Journal of Chinese Computer Systems, 2014, 35(6): 1201-1206.
- [4] Bock S, Childers B R, Melhem R, et al. Concurrent migration of multiple pages in software-managed hybrid main memory[C]//IEEE International Conference on Computer Design. New York: IEEE Press, 2016: 420-423.
- [5] Ramos L E, Gorbato E, Bianchini R. Page placement in hybrid memory systems[C]//Proceedings of the 25th International Conference on Supercomputing. New York: ACM Press, 2011: 85-95.
- [6] Jia Gangyong, Han Guangjie, Xie Hongtian, et al. Hybrid-LRU caching for optimizing data storage and retrieval in edge computing-based wearable sensors[J]. IEEE Internet of Things Journal, 2019, 6(2): 1342-1351.
- [7] Chen Di, Jin Hai, Liao Xiaofei, et al. MALRU: miss-penalty aware LRU-based cache replacement for hybrid memory systems[C]//Design, Automation & Test in Europe Conference & Exhibition (DATE). New York: IEEE Press, 2017: 1086-1091.
- [8] Guthaus M R, Ringenberg J S, Ernst D, et al. MiBench: a free, commercially representative embedded benchmark suite[C]//Proceedings of the IEEE Workshop on Workload Characterization. New York: IEEE Press, 2001: 3-14.
- [9] Fritts J E, Steiling F W, Tucek J A, et al. MediaBench II video: expediting the next generation of video systems

- research[J]. *Microprocessors & Microsystems*, 2009, 33(4): 301-318.
- [10] Binkert N, Beckmann B, Black G, et al. The gem5 simulator [J]. *Acm Sigarch Computer Architecture News*, 2011, 39(2): 1-7.
- [11] 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.
- [12] Ved S, Awasthi M. Exploring non-volatile main memory architectures for handheld devices[C]//*Design, Automation & Test in Europe Conference & Exhibition (DATE)*. New York: IEEE Press, 2018: 1528-1531.
- [13] Samie F, Bauer L, Henkel J. IoT technologies for embedded computing: a survey[C]//*International Conference on Hardware/Software Codesign and System Synthesis*. New York: IEEE Press, 2016: 1-10.
- [14] Gautschi M, Schiavone P D, Traber A, et al. Near-threshold RISC-V core with DSP extensions for scalable IoT endpoint devices[J]. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2017, 25(10): 2700-2713.

(上接第 86 页)

- [10] Uzunalioglu H. Probabilistic routing protocol for low earth orbit satellite networks[C]//*1998 IEEE International Conference on Communications*. New York: IEEE Press, 1998: 89-93.
- [11] 曹志刚, 王京林, 晏坚. LEO 卫星网络快照序列路由算法优化[J]. *宇航学报*, 2009, 30(5): 2003-2007. Cao Zhigang, Wang Jinglin, Yan Jian. Optimization of sequent snapshots routing algorithm in LEO satellite networks[J]. *Journal of Astronautics*, 2009, 30(5): 2003-2007.
- [12] Li J, Lu H, Wang Y. Temporal netgrid model based routing optimization in satellite networks[C]//*2017 IEEE International Conference on Communications*. New York: IEEE Press, 2017: 1-6.
- [13] Yarr N, Ceriotti M. Optimization of inter-satellite routing for real-time data download[J]. *IEEE Transactions on Aerospace and Electronic Systems*, 2018, 54(5): 2356-2369.
- [14] Yang L, Sun J. Multi-service routing algorithm based on GEO/LEO satellite networks[C]//*2016 International Conference on Network and Information Systems for Computers*. New York: IEEE Press, 2016: 80-84.