

基于 PDRAM 混合内存架构的 Linux 内存管理算法

李 功, 陈 岚, 郝晓冉

(中国科学院 微电子研究所, 北京 100029)

摘 要: 近些年来, 相变存储器 (PRAM) 凭借其良好的特性极有可能替代动态存储器 (DRAM), 发展为下一代主流内存技术. 由于 PRAM 在延时和寿命方面的性能退化, 因而采用了位于 Linux 同一物理地址空间的一大块 PRAM 和一小块 DRAM 作为主内存. 现有的 Linux 内核并不是针对混合内存架构而开发的, 无法降低 PDRAM 的内存功耗, 因而, 采用了基于 PDRAM 混合内存架构的 Linux 内存管理算法, 针对不同页帧进行管理, 并根据虚拟地址空间页面的一般访问性质进行选择分配内存页帧, 动态迁移内存页帧, 将频繁读为主的数据存储在 PRAM, 频繁写为主的数据存储在 DRAM, 从而在只增加 0.4% 的内存延时情况下, 降低 50% 左右的内存功耗, 并且没有影响 PRAM 的寿命.

关键词: 相变存储器; 混合内存; Linux 内核; 内存管理; GEM5 模拟器

中图分类号: TP316

文献标识码: A

文章编号: 1000-7180(2014)05-0014-07

Linux Memory Management Algorithm Based on Hybrid Memory Architecture PDRAM

LI Gong, CHEN Lan, HAO Xiao-ran

(The Institute of Microelectronics of Chinese Academy of Sciences, Beijing 100029, China)

Abstract: In recent years, phase-change RAM (PRAM) will most likely become the next generation of mainstream memory technology instead of dynamic RAM (DRAM) because of its good properties. In order to minimize the performance degradation on latency and endurance caused by PRAM, we use PRAM with a small size DRAM and both PRAM and DRAM are mapped into single physical main memory address space in Linux. However, the typical Linux kernel is not develop for hybrid memory architecture, which cannot reduce the PDRAM power consumption. Therefore, we use the memory management algorithm based on hybrid memory architecture PDRAM, which not only can manage different page frames, but also can allocate page frames according to the general access characteristics of pages on virtual address space and migrate page frames dynamically. It can store frequently read data in PRAM and frequently written data in DRAM, which can reduces around 50% power consumption of main memory increased by only 0.4% latency and increase the lifetime of PRAM.

Key words: Phase-Change RAM; Hybrid Memory; Linux Kernel; Memory Management; GEM5 Simulator

1 引言

物联网智能终端具有体积小、重量轻、计算趋向复杂、能量有限等基本特征, 以电池为能量来源的智能终端的续航时间短, 将会成为未来物联网智能终端迫切需要解决的问题. 研究表明, 当 DRAM 负载

达到最大值时, 其功耗可以占据系统总功耗的 50%, 不符合绿色环保的计算概念^[1]. 因此, 使用新型存储架构, 优化存储机制, 达到降低内存功耗的目的.

长期以来, 计算机系统主内存的任务一直由 DRAM 来担当. DRAM 具有易失性, 需要不断充电

收稿日期: 2013-06-20; 修回日期: 2013-08-22

基金项目: 中国科学院战略性先导科技专项 (XDA06000000)

刷新来保持其中所存储的数据.同时,随着集成电路关键尺寸的减小,DRAM 存储器件在尺寸收缩、功耗控制和存储密度方面的发展遇到了极大的困难.因此,国内外研究人员开始考虑利用新型非挥发存储器件来代替 DRAM 构建计算机主内存系统,如磁性存储器(MRAM)、铁电体存储器(FRAM)、相变存储器(PRAM).在访问延时、读写功耗、耐久性、存储密度和成本方面的综合考虑下,PRAM 成为最有可能取代 DRAM 地位的存储器件,其特性^[2]如下:

- (1)非易挥发存储,掉电后数据存储期限可以长达 10 年之久;
- (2)按位寻址,每一位都可单独作为操作数;
- (3)读功耗相对低,写功耗相对高,无静态功耗;访问延时相对较高;
- (4)存储密度高,存储单元的大小只有 $4F^2$;
- (5)有限存储寿命,过多的写入操作($10^8 \sim 10^9$ 次)可能会导致失效.

现有移动智能终端大多搭载了安卓操作系统,它是一个以 Linux 内核为底层基础的半开源操作系统.传统的 Linux 内核内存管理算法是针对同质主内存架构而开发的,并不能充分利用新型混合内存架构的优点.本文提出一种基于 PDRAM 混合内存架构的 Linux 内存管理算法,该算法主要依据以下两个原则:将以频繁读为主的数据存储在 PRAM 中,这样可以发挥 PRAM 读功耗低的优点;将以频繁写为主的数据存储在 DRAM 中,这样可以发挥 DRAM 写功耗低、写延时小以及无使用寿命限制的优点.

2 混合内存体系架构

本文提出的基于 Linux 内存管理算法的混合主内存架构如图 1 所示.与传统同质主内存架构类似,32 位 Linux 内核的线性地址空间总长度为 4 GB.线性地址空间在 3 GB 处划分,各个用户进程都有长度为 3 GB 的独立用户空间和长度为 1 GB 的共享内核空间. Linux 内核通过页表将虚拟地址空间映射到物理地址空间.主物理内存和磁盘之间可以进行页面交换,以保持尽可能多的空闲物理内存.与传统同质主内存体系架构相比,混合内存架构中的主物理内存由位于同一物理地址空间的 PRAM 与 DRAM 两部分组成.其中,位于低地址空间的大部分物理内存被 PRAM 所取代,这样可以增加内存的存储密度,降低内存的读功耗.位于高地址空间的一小部分 DRAM 内存,可以减少因 PRAM 读写速

度和寿命有限而导致的内存性能退化.

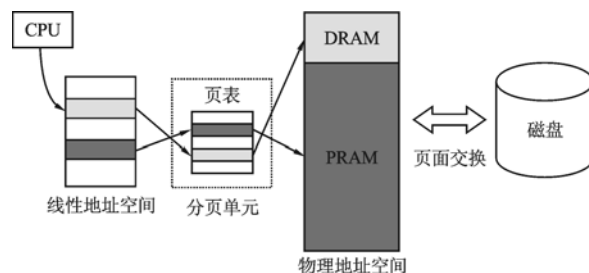


图 1 混合主内存体系架构

3 内存管理算法

3.1 初始化混合物理内存

传统同质主内存架构的物理内存被划分为结点.其中,UMA 系统的物理内存只被划分为一个结点(结点 0),这个结点关联到系统中的每个处理器; NUMA 系统的物理内存被划分为多个结点(结点 0、结点 1、结点 2 等),各个结点分别关联到系统中的每个处理器.各个结点的物理内存又被划分为三个内存域,分别是 DMA(4 kB~16 MB)、NORMAL(16~896 MB)和 HIGHMEM(剩余内存).各个内存域最终被划分成大小为 4 kB 的页帧,并与硬件中的物理内存页框一一映射.然而, Linux 内核并不能区分 PRAM 和 DRAM 不同存储空间页帧的分配、迁移、回收和换入换出等操作.

如果将 PRAM 页帧和 DRAM 页帧映射到不同的结点上,那么 Linux 内核就可以根据标志位选择分配 DRAM 结点的页帧或 PRAM 结点的页帧.然而,多结点的概念只适用于 NUMA 系统并不适用于 UMA 系统的智能终端,不具有通用性.因此,本文提出的 Linux 混合内存页帧管理算法会为各个结点由枚举结构创建 4 个内存域.其中,原有的内存域 DMA、NORMAL、HIGHMEM 来管理 PRAM 页帧,新添加的内存域 DRAM 来管理 DRAM 页帧.借助此枚举结构,各个内存域之间形成了一种优先次序.如果在首选分配的内存域中没有空闲的内存页帧, Linux 内核就会按照代价由低到高的优先次序(DRAM | HIGHMEM → NORMAL → DMA)进行页帧分配,如图 2 所示.

混合内存架构大部分由 PRAM 构成,本文提出的基于混合内存 PDRAM 的 Linux 内存管理算法在分配 PRAM 页帧时,依旧采用传统的页帧分配算法,由伙伴系统从相关的内存域的伙伴列表中得到足够的空闲内存块.在必要情况下,会通过不断二分高分配阶空闲内存块来满足分配请求.如果依旧不

能在相应的内存域找到足够的空闲页帧,则通过守护进程缩减缓存和页帧回收来获得空闲内存块.本文提出的基于混合内存 PDRAM 的 Linux 内存管理算法在分配 DRAM 页帧时,首先会尝试在内存域 DRAM 中查找适合大小的空闲页帧,如果失败,则通过守护进程缩减缓存和页帧回收获得适当的空

闲内存块.如果仍然失败,则尝试在 PRAM 存储空间查找足够的空闲页帧并把内存域 DRAM 中冷 DRAM 页帧迁移到该空闲 PRAM 页帧.如果最后依旧失败,则会导致系统崩溃.虽然迁移操作增加了 PRAM 的写入次数和动态功耗,但是,可以避免把即将频繁写操作的数据分配到 PRAM 存储空间.

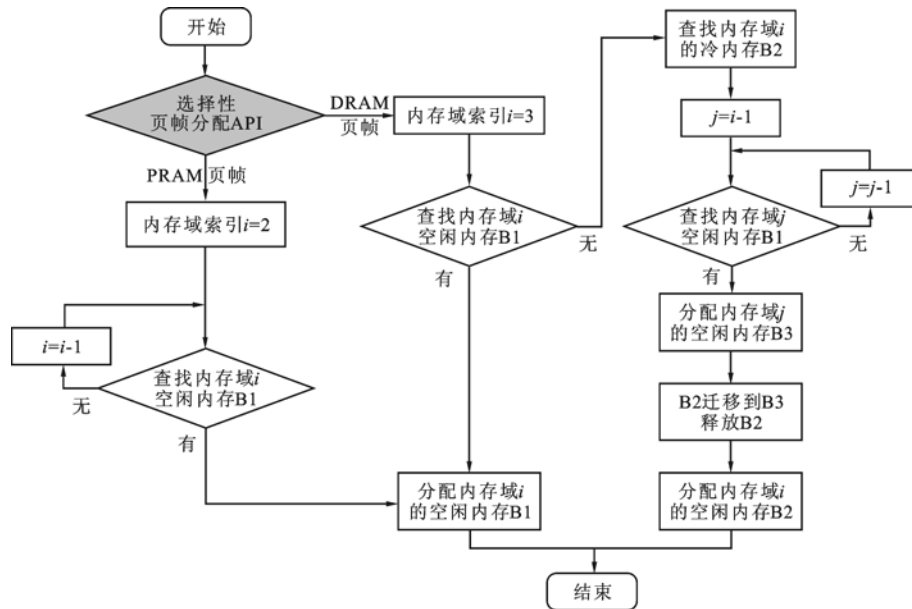


图 2 混合内存页帧分配流程图

3.2 页帧的选择分配

在传统同质物理内存架构中,各个内存页帧的特性总是一致的, Linux 内核不关注页帧具体被分配到主内存的哪个地址范围内.在本文提出的混合内存架构中,页帧分配到 DRAM 存储空间或 PRAM 存储空间将直接影响主内存的性能. PRAM 读操作的延时和功耗优于写操作的延时和功耗.如果 PRAM 页帧频繁进行写操作,那么主内存的动态功耗和延时将会显著地增加,而且过多的写操作势必会减少主内存的使用寿命.因此,本论文提出的基于 PDRAM 混合内存架构的 Linux 内存管理选择分配算法如图 2 灰色部分所示,该算法主要依据如下两条原则:频繁读为主的数据分配到 PRAM 存储空间,这样可以发挥 PRAM 读功耗低的优点;频繁写为主的数据分配到 DRAM 存储空间,这样可以发挥 DRAM 写功耗低、写延时小的优点.

尽管很难预测各个页帧未来的访问模式,但是依旧可以根据虚拟地址空间页面的一般访问性质来分配两种不同的页帧.一般的, Linux 进程的虚拟地址空间被划分为五个线性的内存区域,其访问模式如表 1 所示.其中,堆栈段数据读写对称的访问模式适合储存到 DRAM 空间,其他段数据读多写少的

访问模式适合储存到 PRAM 空间.因此,本文提出的基于 PDRAM 混合内存架构的 Linux 内存管理选择分配算法,将堆栈段的页面优先分配到内存域 DRAM,其他段的页面优先分配到内存域 HIGH-MEM.

表 1 虚拟内存区的访问模式^[9]

内存区	存储内容	读操作	写操作
代码段	当前运行的二进制代码	频繁	无操作
数据段	存储全局变量和静态变量	频繁	不频繁
堆段	动态产生的数据	频繁	频繁
栈段	局部变量和实现函数调用	频繁	频繁
内核段	参数列表	较频繁	不频繁
	命令行参数		
	环境变量		

当程序访问虚拟地址空间的一个页面但是其并未被映射到物理地址空间的某一个页帧时,处理器就会产生一个缺页异常.如果 Linux 内核判断此次异常是无效的,那么 Linux 内核通常会转储内存、终止相关的进程,然后向用户报告.如果此次异常是有效的,那么 Linux 内核会分配空闲的物理内存页帧并尝试按需调页、按需分配、页面交换,或者写时复

制使得页帧允许被访问并映射. 内存描述符参数 `mm_struct` 描述了产生缺页异常进程的内存区信息. 例如, 代码段开始和结束的虚拟内存地址分别通过 `start_code` 和 `end_code` 标记. 参数 `address` 描述了当前进程产生缺页异常的虚拟空间地址. 这样, 当处理器产生按需调页、按需分配、页面交换或者写时复制的缺页异常时, 通过比较产生缺页异常的虚拟空间地址 `address` 和内存描述符 `mm_struct` 中各个内存区的起止地址, 就可以判断出这个虚拟地址是属于堆栈段还是其它段, 并为其标志相应内存修饰符, 从而分配 PRAM 页帧或 DRAM 页帧, 其流程如图 3 所示.

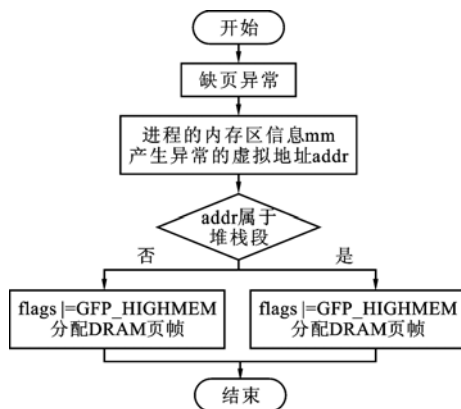


图3 页帧选择分配算法流程图

3.3 页帧的动态迁移

页帧选择分配算法是一种静态的算法, 无法动态调度不符合虚拟地址空间页面的一般访问性质的页帧. 例如, 以频繁读为主的堆栈段数据仍然分配 DRAM 页帧, 以频繁写为主的其他段数据仍然分配 PRAM 页帧. 如果 PRAM 页帧频繁进行写操作, 那么主内存的动态功耗和延时将会有显著的增加, 而且过多的写操作势必会减少主内存的使用寿命. 因此, 本文提出的基于混合内存 PDRAM 的 Linux 内存管理动态迁移算法, 流程图如图 4 所示, 该算法基于著名的局部性原理^[4]: 最近不读(写)的页帧在未来也不会需要频繁进行读(写)操作; 最近读(写)次数多的页帧在未来也会需要频繁进行读(写)操作.

冷 DRAM 页帧迁移: 通过传统的最近最少使用(LRU)算法, 将最近最少读写的 DRAM 页帧迁移到 PRAM 存储空间, 由于未来该页帧也不会频繁进行读写操作, 因此, 迁移后该页帧的访问行为不会大量增加 PRAM 的功耗. 每个内存域包含了两个双向 LRU 链表: 活动(active)链表, 链接经常被访问的处于活跃状态的页帧; 非活动(inactive)链表, 链接不经常被访问的页帧. 随着时间的推移, LRU 链

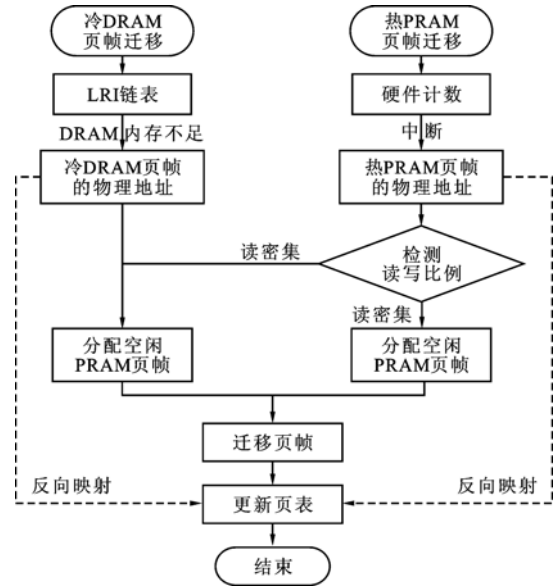


图4 页帧动态迁移算法流程图

表通过传统的最近最少使用算法形成了一种热平衡, 经常被访问的页帧位于活动链表的开头, 而较少访问的页帧位于非活动链表的开头. 当 DRAM 内存空间不足时, 本文提出的基于混合内存架构 PDRAM 的 Linux 内存管理动态迁移算法首先查找并分配一个空闲 PRAM 页帧, 将位于内存域 DRAM 非活动链表末尾的冷 DRAM 页帧迁移到该 PRAM 页帧, 并释放这个冷 DRAM 页帧, 最后通过传统的反向映射算法更新与这个冷 DRAM 页帧相关联的所有页表.

热 PRAM 页帧迁移: 通过硬件实时监控内存的读写次数来预测各个页帧未来的访问模式, 将读密集的 PRAM 页帧迁移到另一个空闲的 PRAM 页帧, 这样可以依旧发挥 PRAM 读功耗低的优点. 将写密集的 PRAM 页帧迁移到 DRAM 存储空间, 这样可以发挥 DRAM 写功耗低、写延时小和无寿命限制的优点. 此外, 达到写操作阈值的 PRAM 页帧会被释放, 并且不会立即被分配进行读写操作, 还可以延长 PRAM 页帧的寿命. 本论文提出的基于混合内存架构 PDRAM 的 Linux 内存管理动态迁移算法, 由硬件对每个内存页帧进行读写计数, 如果写操作达到阈值, 则计算该页帧读写操作的比例, 并产生中断. 根据 PRAM 与 DRAM 的读写功耗可以得出零功耗迁移读写比例, 即迁移前后页帧读写功耗保持不变. 如果大于等于此比例值, 则查找并分配一个空闲 PRAM 页帧, 将产生中断的热 PRAM 页帧迁移到该空闲 PRAM 页帧, 并把这个热 PRAM 页帧释放到空闲 LRU 链表末尾, 最后通过反向映射算

法更新与这个热 PRAM 页帧相关联的所有页表. 如果小于此比例值, 则查找并分配一个空闲 DRAM 页帧, 将产生中断的热 PRAM 页帧迁移到该空闲 DRAM 页帧, 并把这个热 PRAM 页帧释放到空闲 LRU 链表开头, 最后通过反向映射算法更新与这个热 PRAM 页帧相关联的所有页表.

4 仿真结果

本文选用 GEM5 开源模拟器^[5], 主要关注内存的模拟, 不考虑排队延时和资源竞争, 因此只对包含 CPU 模型、总线模型和内存模型的简单系统进行模拟. 其中, CPU 模型采用 AtomicSimpleCPU 模型, 总线模型仅仅实现数据包无延时转发的功能. 为了实现混合主内存架构, 需要在原有模型基础上, 在同一物理地址空间添加一个新的内存模块, 分别模拟 PRAM 和 DRAM 两块内存. 此外, 还需要在内存模块之上添加一个内存监控模块, 它可以对 PRAM 和 DRAM 两块内存的读写次数进行监控, 进而得到内存的访问功耗和延时.

GEM5 模拟的硬件环境如表 2 所示, DRAM 和 PRAM 的访问功耗和延时如表 3 所示, 各个参数的选取会直接影响仿真结果. 其中, 写操作阈值为 1 000 即一旦页帧写操作计数值每达到 1 000 就会产生中断; 根据内存功耗得出的零功耗迁移读写比例为 5:2. 本论文在同环境下对 DRAM 同质内存架构、不包含选择分配和动态迁移内存管理算法的 PDRAM 混合内存架构、只包含选择分配内存管理算法的 PDRAM 混合内存架构、包含选择分配和动态迁移内存管理算法的 PDRAM 混合内存架构这 4 种情况进行仿真, 并选用 7 个 MBench 基准测试程序^[7] (Basicmath、Bitcount、Qsort、Susan、String Search、Jpeg、Typeset) 对内存功耗和延时进行分析和验证. 其中, 基准测试程序读操作平均为 5 亿次、写操作平均为 3 600 万次, 呈现出读操作次数很多、写操作次数较少的内存访问模式.

表 2 GEM5 模拟环境

参数	值
处理器	ARMv7 2.0 GHz
内存	176 MB (PRAM) 80 MB (DRAM)
操作系统	修改版 32 位 Linux 3.9.1
写操作阈值	1 000
读写比例阈值	5 : 2

表 3 DRAM 和 PRAM 特性^[6]

	DRAM	PRAM
读功率	200 mW	80 mW
写功率	200 mW	500 mW
读延时	50 ns	50 ns
写延时	50 ns	150 ns

DRAM 同质内存架构、不包含选择分配和动态迁移内存管理算法的 PDRAM 混合内存架构、只包含选择分配内存管理算法的 PDRAM 混合内存架构、包含选择分配和动态迁移内存管理算法的 PDRAM 混合内存架构的归一化内存功耗如图 5 所示.

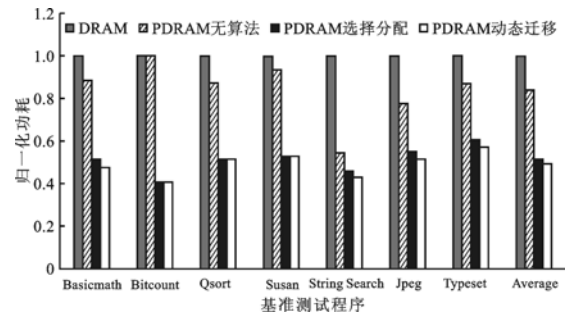


图 5 归一化功耗

当 DRAM 同质内存架构位于低地址空间的大部分物理内存被 PRAM 所取代时, 尽管 PRAM 的写功耗高于 DRAM 的写功耗, 但是由于 PRAM 的读功耗低于 DRAM 的读功耗, 而且在此区域依旧呈现读操作次数多、写操作次数少的内存访问模式, 因此不包含选择分配和动态迁移内存管理算法的 PDRAM 混合内存架构可以平均降低 15% 的内存功耗. 其中, Bitcount 测试程序发生在 PRAM 存储空间的读操作次数 (305 万次) 很少, 因而仅仅内存功耗降低了 0.2%. 相反的, String Search 发生在 PRAM 存储空间的读操作次数 (4 864 万次) 很多、写操作次数 (10 万次) 很少, 因而内存功耗降低了 46.0%.

只包含选择分配内存管理算法的 PDRAM 混合内存架构, 在不包含选择分配和动态迁移内存管理算法的 PDRAM 混合内存架构的基础上, 大量增加了发生在 PRAM 存储空间的读操作次数 (减少了发生在 DRAM 存储空间的读操作次数), 因此比不包含选择分配和动态迁移内存管理算法的 PDRAM 混合内存功耗平均减少了 34.7%, 即平均降低 49.7% 的内存功耗. 其中, Bitcount 测试程序发生在 PRAM 存储空间的读操作次数 (7 亿次) 有最为明显

增多,因而比不包含选择分配和动态迁移内存管理算法的PDRAM混合内存功耗减少了58.9%,即降低59.1%的内存功耗.相反,String Search测试程序发生在PRAM存储空间的读操作次数(6 047万次)、写操作次数(13万次)无明显改变,因而比不包含选择分配和动态迁移内存管理算法的PDRAM混合内存功耗仅仅减少了8.6%,即降低54.6%的内存功耗.

包含选择分配和动态迁移内存管理算法的PDRAM混合内存架构,在只包含选择分配内存管理算法的PDRAM混合内存架构的基础上,同时减少了发生在PRAM存储空间的读写操作次数(增加了发生在DRAM存储空间的读写操作次数),导致迁移后增加的读操作功耗和减少的写操作功耗,形成了动态平衡,而且达到写阈值发生的迁移操作平均仅为34万次,并不会对内存功耗产生影响,因此比只包含选择分配内存管理算法的PDRAM混合内存功耗平均只减少了1.0%,即平均降低50.7%的内存功耗.其中,Bitcount测试程序发生在PRAM存储空间的读操作次数(7亿次)无明显变化,写操作次数减少了188万次,发生在DRAM存储空间的读操作次数增加了302万次,写操作次数增加了256万次,因而与只包含选择分配内存管理算法的PDRAM混合内存功耗相比无明显变化,即降低59.1%的内存功耗.

这四种架构的归一化内存延时如图6所示.其中,DRAM同质内存架构平均延时约为27.1秒、PDRAM混合内存架构平均延时分别约为27.5秒、28.2秒、27.2秒.

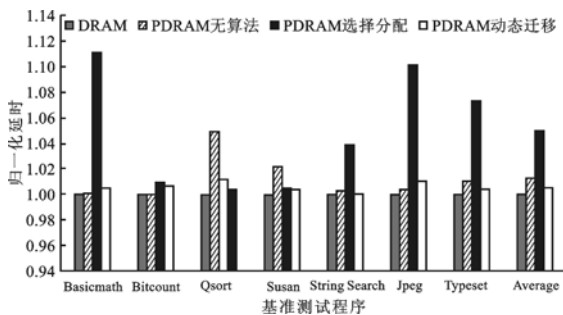


图6 归一化延时

当DRAM同质内存架构位于低地址空间的大部分物理内存被PRAM所取代时,由于PRAM的写延时,高于DRAM的写延时而PRAM的读延时基本等于DRAM的读延时,因此PDRAM混合内存架构必定会增加内存的延时.不包含选择分配和动态迁移内存管理算法的PDRAM混合内存架构

平均增加1.3%(约0.47秒)的内存延时.其中,Qsort测试程序发生在PRAM存储空间的写操作次数(2 214万次)很多,远高于其他测试程序,因而内存延时增加了5.0%(约2.0秒).相反的,Basicmath测试程序发生在PRAM存储空间的写操作次数(17万次)很少,因而内存延时仅仅增加了0.1%(约0.02秒).

只包含选择分配内存管理算法的PDRAM混合内存架构,在不包含选择分配和动态迁移内存管理算法的PDRAM混合内存架构的基础上大量增加了发生在PRAM存储空间的读操作次数(减少发生在DRAM存储空间的读操作次数),同时也增加了发生在PRAM存储空间的写操作次数(减少发生在DRAM存储空间的写操作次数),增加的写操作次数使得内存延时比不包含选择分配和动态迁移内存管理算法的PDRAM混合内存架构的内存延时平均增加了3.7%(约0.73秒),即平均增加5.0%(约1.2秒)的内存延时.其中,Qsort测试程序发生在PRAM存储空间的写操作次数明显减少了1 579万次,因而比不包含选择分配和动态迁移内存管理算法的PDRAM混合内存架构的内存延时减少了3.8%(约1.5秒),即增加1.2%(约0.5秒)的内存延时.相反的,Basicmath测试程序发生在PRAM存储空间的写操作次数明显增加了2 178万次,因而比不包含选择分配和动态迁移内存管理算法的PDRAM混合内存架构的内存延时增加了11.1%(约2.1秒),即增加11.2%(约2.2秒)的内存延时.

包含选择分配和动态迁移内存管理算法的PDRAM混合内存架构,在只包含选择分配内存管理算法的PDRAM混合内存架构的基础上减少了发生在PRAM存储空间的读写操作次数(增加发生在DRAM存储空间的读写操作次数),而且达到写阈值发生的迁移到操作平均仅为34万次,并不会对内存延时产生影响,因而平均只增加了0.5%(约0.1秒)的内存延时.其中,Qsort测试程序发生在PRAM存储空间的写操作次数减少了354万次,因而比只包含选择分配内存管理算法的PDRAM混合内存架构的内存延时减少了0.8%(约0.3秒),即增加0.4%(约0.2秒)的内存延时,相反的,Basicmath测试程序发生在PRAM存储空间的写操作次数明显减少了2 134万次,因而比只包含选择分配内存管理算法的PDRAM混合内存架构的内存延时减少了11.0%(约2.1秒),即增加0.2%(约

0.1 秒)的内存延时。

综上所述,不包含选择分配和动态迁移内存管理算法的 PDRAM 混合内存架构,虽然内存延时没有显著增加,但是不能显著降低内存功耗。只包含选择分配内存管理算法的 PDRAM 混合内存架构,虽然可以有效降低内存功耗但是内存延时却显著增加。而包含选择分配和动态迁移内存管理算法的 PDRAM 混合内存架构,可以有效降低内存功耗的同时,保证内存延时的增加在可接受范围之内。

5 结束语

随着内存技术的不断发展,PRAM 将会逐步替代 DRAM 成为下一代内存技术。为了更好地管理混合内存系统,充分发挥 DRAM 和 PRAM 两种内存存储器件的优点,本文提出了基于混合内存 PDRAM 的 Linux 内存管理算法,经过验证可以针对 PRAM 和 DRAM 页帧进行管理,选择分配和动态迁移算法还可以在只增加了 0.5% 的内存延时的情况下降低了 50% 左右的内存功耗,并且没有影响 PRAM 的寿命。然而,各个参数的选取会直接影响性能,例如写操作阈值和读写比例阈值,因此选取最优参数从而最大化提升性能将成为下一步的研究重点。

参考文献:

- [1] Ibrahim H, Calvin L. A comprehensive approach to DRAM power management[C]//IEEE International Symposium on High-Performance Computer Architecture, Taxes, 2008.

- [2] Wikimedia Foundation, Inc. Phase-change memory [EB/OL]. [2013-05-19]. http://en.wikipedia.org/wiki/Phase-change_RAM.
- [3] Park Y, Park S K, Park K H. Linux kernel support to exploit phase change memory[C]//Linux Symposium. Canada;Ottawa,2010.
- [4] Mauerer W. Professional linux kernel architecture[M]. New Jersey, USA, John Wiley & Sons, Inc. 2008: 107-205.
- [5] Binkert N, Beckmann B, Black G. The GEM5 simulator[J]. SIGARCH Computer Architecture News, 2011,39(2):1-7.
- [6] Jung K S, Park J W, Weems C C, et al. A superblock-based memory adapter using decoupled dual buffers for hiding the access latency of non volatile memory [C]//Proceedings of the World Congress on Engineering and Computer Science. USA, San Francisco,2011.
- [7] Guthaus M R, Ringenberg J S, Ernst D, et al. MiBench: A free, commercially representative embedded benchmark suite[J]. IEEE International Workshop, 2001,12(4):3-14.

作者简介:

李 功 男,(1989—),硕士研究生。研究方向为物联网智能终端体系架构。

陈 岚 女,(1968—),博士,研究员,博士生导师。研究方向为移动物联网系统架构及核心芯片、SoC/IP 核验证、评测、低功耗设计技术。

郝晓冉 女,(1980—),博士,助理研究员。研究方向为物联网智能终端。