CN 43-1258/TP
ISSN 1007-130X

计算机工程与科学
Computer Engineering & Science

第 41 卷第 7 期 2019 年 7 月
Vol. 41, No. 7, Jul. 2019

文章编号:1007-130X(2019)07-1168-08

通过部分页迁移实现 CPU-GPU 高效透明的数据通信*

张诗情,杨耀华,沈　立,王志英
(国防科技大学计算机学院,湖南 长沙 410073)

摘　要:尽管对集成 GPU 和下一代互连的研究投入日益增加,但由 PCI Express 连接的独立 GPU 仍占据市场的主导地位,CPU 和 GPU 之间的数据通信管理仍在不断发展。最初,程序员显式控制 CPU 和 GPU 之间的数据传输。为了简化编程,GPU 供应商开发了一种编程模型,为"CPU+GPU"异构系统提供单个虚拟地址空间。此模型中的页迁移机制会自动根据需要在 CPU 和 GPU 之间迁移页面。为了满足高性能工作负载的需求,页面大小有增大趋势。受低带宽和高延迟互连的限制,较大的页面迁移延迟时间较长,这可能会影响计算和传输的重叠并导致严重的性能下降。提出了部分页迁移机制,它只迁移页面的所需部分,以缩短迁移延迟并避免页面变大时整页迁移的性能下降。实验表明,当页面大小为 2 MB 且 PCI Express 带宽为 16 GB/s 时,部分页迁移可以显著隐藏整页迁移的性能开销,相比于程序员控制数据传输,整页迁移有平均 98.62%倍的减速,而部分页迁移可以实现平均 1.29 倍的加速。此外,我们测试了页面大小对快表缺失率的影响以及迁移单元大小对性能的影响,使设计人员能够基于这些信息做出决策。

关键词:"CPU+GPU"异构系统;数据通信;页迁移
中图分类号:TP392.02　　**文献标志码**:A
doi:10.3969/j.issn.1007-130X.2019.07.004

Efficient and transparent CPU-GPU data communication through partial page migration

ZHANG Shi-qing, YANG Yao-hua, SHEN Li, WANG Zhi-ying
(School of Computer, National University of Defense Technology, Changsha 410073, China)

Abstract: Despite the increasing investment in integrated GPUs and next-generation interconnect research, discrete GPUs connected by PCI Express still dominate the market, and the management of data communication between CPUs and GPUs continues to evolve. Initially, the programmers control the data transfer between CPUs and GPUs explicitly. To simplify programming, GPU vendors have developed a programming model to provide a single virtual address space for "CPU + GPU" heterogeneous systems. The page migration engine in this model transfers pages between CPUs and GPUs on demand automatically. To meet the needs of high-performance workloads, the page size tends to be larger. Limited by low bandwidth and high latency interconnections, larger page migration has longer delay, which can reduce the overlap of computation and transmission and cause severe performance degradation. We propose a partial page migration mechanism that only transfers the requested part of a page to shorten the migration latency and avoid performance degradation of the whole page migration when the page becomes larger. Experiments show that the proposed partial page migration can well hide the performance overheads of the whole page migration when the page size is 2MB and the PCI Express bandwidth is 16GB/sec. Compared with data transmission controlled by the programmers, the whole page migration degrades the performance by 98.62 on average, while the partial page migration upgrades the performance

* 收稿日期:2018-10-19;修回日期:2018-12-11
通信地址:410073 湖南省长沙市国防科技大学计算机学院
Address: School of Computer, National University of Defense Technology, Changsha 410073, Hunan, P. R. China

by 1.29 on average. Additionally, we examine the impact of page size on TLB miss rate and the impact of migration unit size on execution time, enabling designers to make informed decisions based on this information.

Key words: heterogeneous "CPU + GPU" system; data communication; page migration

1 引言

图形处理单元 GPU(Graphic Processing Unit)已被广泛应用于现代高性能计算系统。由于数据处理和传输之间巨大的性能差距,CPU 与 GPU 之间的数据通信成为异构"CPU+GPU"系统中的关键问题。统一内存(Unified Memory)创建了一个由 CPU 和 GPU 共享的托管内存池,桥接了 CPU 与 GPU 之间的鸿沟[1]。系统软件在 GPU 和 CPU 之间自动迁移统一内存中的数据[2]。统一内存带来至少 2 个优点:首先,简化了编程和内存模型。其次,通过在 CPU 和 GPU 之间按需迁移数据,统一内存可以在 GPU 上提供访问本地数据的性能。

在统一内存中,数据通常以页面为单位进行迁移。页面大小在多个方面影响性能。当采用小页面(例如 4 KB)时,由于每次要迁移的数据量小,页迁移延迟较低,因此迁移和计算的重叠率很高。但是,使用小页面可能会降低转换检测缓冲区 TLB(Translation Lookaside Buffer)命中率并生成更多的页表查询操作,导致更大的地址转换开销[3]。当使用大页面(例如 2 MB)时,页迁移延迟大,而地址转换开销较低[4,5]。同时,由于应用程序数据访问的不连续性,被迁移的大页面在短时间内只被访问了一小部分。当页面大小为 2MB 时,我们对 10 个基准测试进行了实验,结果显示被迁移页面的被访问比例小于 5%。这种不必要的数据传输会导致带宽浪费,还会阻塞后续的数据迁移。

整页迁移机制使用整个页面作为迁移单元和内存管理单元,很难在大小页面各自的利弊之间进行权衡。因此,有必要优化页迁移机制本身来综合不同页面大小的优点。新的迁移机制应当在优化地址转换的同时限制迁移延迟,并且满足大规模应用的 CPU-GPU 通信需求。

为了解决这些问题,本文提出了一种透明的部分页迁移机制,即只迁移被请求的部分页,而不是其所在的整个页面。为了实现部分页迁移,我们在页面状态列表中添加部分有效的状态,同时在页表项中记录已经迁移到 GPU 的页内范围,本文称之为有效范围。实验表明,部分页迁移相对于整页迁移获得了显著的性能改进。其中单个有效范围的部分页迁移实现了 55.94 倍的加速,多个有效范围的部分页迁移实现了 93.81 倍的加速。此外,与程序员控制传输机制相比,部分页转移对程序员来说更简单和方便。

本文的贡献总结如下:

(1)提出了一种透明的部分页迁移机制,可以根据需要自动迁移页面的被请求部分,并在不修改程序代码和运行时库的情况下应用。

(2)提出了两种部分页迁移的实现机制,即单个有效范围的部分页迁移和多个有效范围的部分页迁移。

(3)评估了不同页面大小的快表(TLB)失效率和部分页迁移对迁移单元大小的性能敏感性,结果表明,相对于整页迁移机制,部分页迁移机制获得了显著的性能改进。

2 相关工作

CPU 和 GPU 之间数据通信的发展经历了 3 个阶段:(1)CPU 和 GPU 分别拥有自己的私有地址空间。在 GPU 执行计算之前,程序员必须显式地将数据从 CPU 复制到 GPU。这种程序员控制传输模型增加了编程的复杂性,并使得数据传输和数据处理串行化[6,7]。(2)引入了统一虚拟地址,例如 Intel 的 Haswell、AMD 的 Berlin 处理器、NVIDIA 的 Fermi 架构和 ARM 的 Cortex 内核[8,9]。虽然统一虚拟地址提供了一些编程方便性,但由于数据通信的低带宽和高延迟,对于提高整体性能几乎没有帮助。而且它不能自动迁移数据。(3)统一内存是基于统一虚拟地址引入的,例如 Intel 的 Haswell、ARM 的 HSA 和 NVIDIA 的 Tesla[1,8,10]。数据可以在主机内存和设备内存之间自动迁移。统一内存简化了编程并具有提高性能的潜力。

当前统一内存中的数据通信机制是整页迁移[4,11]。当页面变大时,使用整页迁移机制导致统一内存管理存在一些性能风险,如引言部分第 3 段所述。

Lustig 等人[12]在 GPU 中利用 F/E 位实现了一种细粒度的同步方法,通过重叠已传输数据的计算与剩余数据的传输来提高性能。但是,它需要修改应用程序的源代码,并对 API 进行扩展,对程序员来说不透明。而本文提出的方法是根据需要自动迁移页面的所需部分,无需修改应用程序和扩展 API。子页面是一种精细的页面管理理念,类似于本文中的部分页,但目前它的应用场景和目的与本文的部分页不同,主要用于减少数据规模或增加并行性,从而加速计算[13]。本文的部分页迁移是为了避免仅在请求部分页面时迁移整个页面,以减少迁移延迟和带宽浪费。此外,一些研究提出了页面放置机制和迁移阈值[14]。本文的迁移机制是按需迁移被请求的部分页而不预设阈值,这一机制基于以下结论:设置必要的硬件计数器来跟踪系统中的所有页面,以区分其访问计数的开销,超过了这一改进相对于简单的"首次请求即迁移"机制得到的性能提升[15]。

3 部分页迁移

3.1 基本理念

为了减小页面变大时整页迁移所带来的迁移延迟和带宽浪费,本文提出了一种新的迁移方法,可以保持较大的页面,同时调整每次迁移的数据量。这种迁移方法就是部分页迁移,即只迁移页面的被请求部分,并记录已迁移到 GPU 本地内存的数据范围。与整页迁移相比,部分页迁移使迁移单元可调整,并且每次迁移的数据都是被请求的部分,而不是整个页面。这种方式可以有效缩短迁移延迟,降低阻塞后续请求的概率,并提高计算和传输的重叠率。在整页迁移中,页面在系统中只有一个副本。在部分页迁移中,页面可以部分迁移到 GPU 本地内存,而其余部分仍在 CPU 端。为了实现部分页迁移,需要修改页面状态管理和迁移过程。

3.2 部分有效状态

在整页迁移中,有 3 种页面状态:无效(not-valid)、CPU 中有效(valid-in-CPU)和 GPU 中有效(valid-in-GPU)。无效意味着页面尚未从硬盘传输到异构系统内存。CPU 中有效表示页面处于 CPU 内存中,GPU 不能在其本地内存中直接访问该页面。GPU 中有效表示页面已被迁移到 GPU 本地内存。当请求无效的页面时,它可以通过页缺失处理从硬盘传输到 CPU 或直接传输到 GPU,并将状态更改为 CPU 中有效或 GPU 中有效。为了简化描述,我们在下面的章节中不考虑无效状态。图 1a 显示了整页迁移的页面状态转换图。

为了执行部分页迁移,我们添加了部分有效(partial-valid)的页面转换状态,记录页面的有效范围并根据页面迁移操作实时修改它。部分有效意味着页面已经有一部分从 CPU 内存迁移到 GPU。部分页迁移的页面状态转换图如图 1b 所示。与整页迁移相比,部分页迁移的状态转换过程具有以下变化:(1)当页面状态为 CPU 中有效时,如果有来自 GPU 的请求,系统会将此页面的被请求部分迁移到 GPU 端,将页面状态更改为部分有效并修改有效范围。(2)当页面状态为部分有效时,如果有来自 GPU 的请求,系统将继续迁移页面的被请求部分到 GPU 内存并修改有效范围。如果整个页面都已被迁移,页面状态将被更改为 GPU 中有效。(3)当页面状态不是 CPU 中有效并且有来自 CPU 的请求时,整个页面将被迁移到 CPU,页面状态被更改为 CPU 中有效。

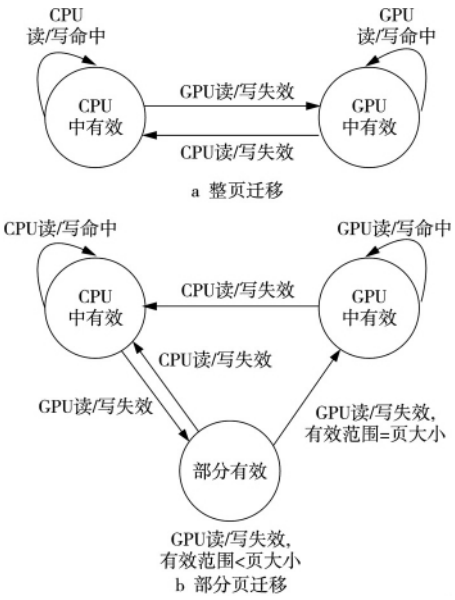

Figure 1 Page state transition graphs

图 1 页面状态转换图

3.3 部分页迁移过程

为了支持部分页迁移,我们还需要对迁移过程的管理进行修改。部分页迁移过程如图 2 所示。首先,来自 GPU 核的数据请求将所请求的地址和数据长度发送到 GPU 内存管理单元 GMMU。接着,GMMU 查询相应页面的状态和有效范围,根据不同的页面状态,对 GPU 的数据请求进行相应的处理。(1)如果页面状态为部分有效且被请求的

数据在有效范围内,或者页面状态为 GPU 中有效,则 GMMU 允许 GPU 直接在本地访问页面。(2)如果页面状态为部分有效并且所请求的数据范围不在有效范围内,或者页面状态为 CPU 中有效,则 GMMU 为该请求分配缓冲器条目并向 CPU 发送部分迁移请求。然后,页面的被请求部分从 CPU 迁移到 GPU。在部分页迁移完成后,GMMU 更新页面的状态和有效范围,并允许 GPU 直接在本地访问页面。

在部分页迁移中,添加页面状态和维护有效范围,可以使 GMMU 快速确定是否需要迁移所请求的数据并在需要时向 CPU 发送迁移请求。在同一个周期内,不同的核可能生成多个迁移请求。当缓冲区满时,如果出现新的迁移请求,它将阻塞并等待空闲的缓冲区条目。

部分页迁移步骤如下所示:

①GPU 核产生数据请求;

②GMMU 查询对应页面的状态和有效范围,如果页面状态为 GPU 中有效或部分有效且请求范围在有效范围内,则跳转至步骤⑥;

③GMMU 为请求分配迁移缓冲区条目;

④GMMU 向 CPU 发送迁移请求;

⑤CPU 向 GPU 迁移被请求的部分页;

⑥GMMU 修改对应页的状态和有效范围;

⑦GMMU 向核发送消息使其在本地访问数据。

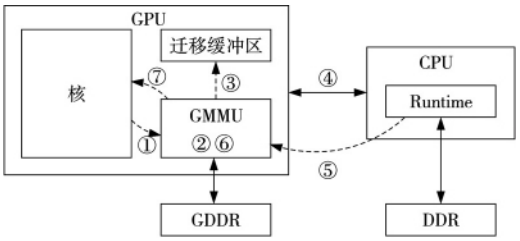

Figure 2 Process of partial page migration

图 2 部分页迁移过程

4 迁移机制

4.1 单一有效范围的部分页迁移

要将部分页迁移应用于异构系统,最简单的方法是向页面状态列表添加部分有效状态,同时由页表项记录并维护一个有效范围,包括有效范围起始处的页面偏移量和有效范围的长度。当新请求的范围与页面有效范围不连续时,将间隙与请求部分一起迁移以保持连续的有效范围。因此,页面迁移时有效范围始终保持连续。

如果不考虑 CPU 产生的请求和无效页面状态,则单个有效范围的部分页迁移机制的工作方式如图 3a 所示。(1)如果页面状态为 GPU 中有效,则 GPU 可以直接访问本地内存中的数据,页面状态和有效范围保持不变。(2)如果页面状态为 CPU 中有效并且存在可用的缓冲区条目,则为请求分配缓冲区条目并等待迁移。如果没有可用的缓冲区条目,则请求将被阻止,直到获得空闲条目。迁移请求完成后,GMMU 会将页面状态从 CPU 中有效修改为部分有效,并更新有效范围的起始页面偏移量和长度,然后允许 GPU 直接访问本地内存中的数据。(3)如果页面状态是部分有效,GMMU 将检查所请求的范围是否在有效范围内。如果是,则处理过程与页面状态为 GPU 中有效时相同;如果不是,则处理过程将与页面状态为 CPU 中有效时相同,且若更新后的有效长度等于页面大小,则 GMMU 将页面状态修改为 GPU 中有效。

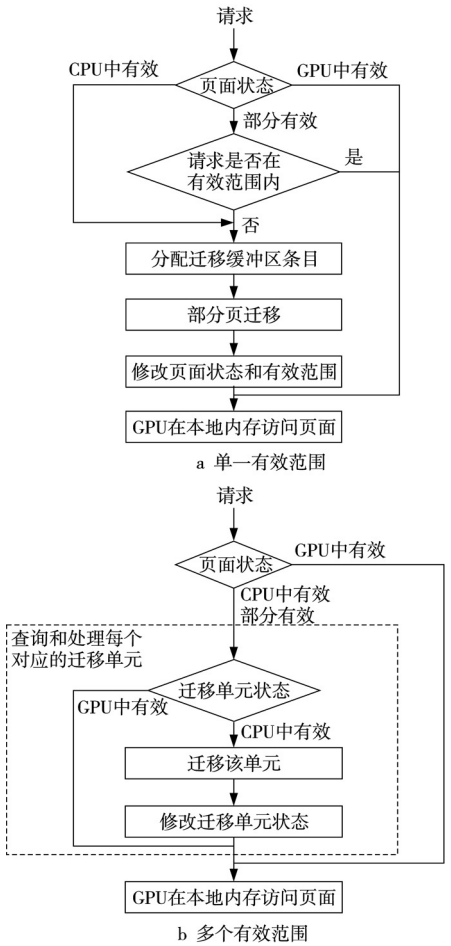

Figure 3 Flow chart of partial page migration

图 3 部分页迁移流程图

单一有效范围的部分页迁移是一种对整页迁移改动较少的部分页迁移实现方式。但是,这种实现有一个明显的缺陷:当页面较大时,如果出现较

大的偏移间隙,系统将会迁移大量未被请求数据,以保持有效范围的连续性。

4.2 多个有效范围的部分页迁移

使用单个连续有效范围无法避免过大迁移长度的可能性。为了解决这个问题,我们提出了第2种实现机制,多个有效范围的部分页迁移,在单个页面中启用多个不连续的有效范围。设定一个阈值,当新请求的范围与页面有效范围不连续时,若间隙长度小于阈值,将间隙与请求部分一起迁移,以保持连续的有效范围,否则记录为另一个有效范围。

为了实现多个有效范围的部分页迁移,我们需要记录多个有效范围。数据请求根据有效范围分成可能具有不同长度和状态的迁移单元。迁移单元的状态可能是无效、CPU中有效或GPU中有效。(1)如果页面状态为GPU中有效,则GPU在本地访问数据。(2)如果页面状态为在CPU中有效:首先,GMMU计算所请求的数据范围对应于哪些迁移单元。其次,GMMU检查每个被请求的迁移单位的有效状态。如果它在CPU中有效,则GMMU向CPU发送一个迁移请求。迁移完成后,GPU将页面状态更改为GPU中有效并在本地访问此迁移单元。直到所有请求的迁移单元都在GPU中有效,该请求才能完成。最后,GMMU根据迁移单元状态更新页面有效范围和页面状态。(3)如果页面状态是部分有效的,则该过程类似于(2),唯一的区别是迁移单元可能在GPU中有效,那么GPU可以在本地访问它。多个有效范围的部分页迁移流程如图3b所示。

5 实验和分析

部分页迁移引入了一些新的系统参数,包括页面大小、迁移单元大小和CPU-GPU互连带宽等。实验侧重于部分页迁移本身的参数以及(1)理想"传输+执行"重叠;(2)程序员控制传输;(3)整页迁移;(4)部分页迁移之间的性能比较。理想"传输+执行"重叠代表计算和迁移完全并行时没有通信开销的理想情况。程序员控制传输代表计算和迁移完全串行的情况,即GPU运行所需的所有数据在程序开始运行之前由程序员从CPU迁移到GPU。

5.1 环境设置

我们修改gpgpu-sim v3.x中的GTX480配置参数,使它们更接近当前的GPU配置[16];修改了模拟器的内存管理部分(包括页表、TLB、迁移缓存等),增加了CPU-GPU数据通信模块。部分参数如表1所示。目前广泛使用的PCI Express 3.0×16的带宽为16 GB/s[17],PCI Express 4.0×16的带宽预计为32 GB/s[18]。因此,我们以16 GB/s和32 GB/s作为当前和未来带宽的代表进行测试。计时模型中的部分页迁移的额外开销包括查询和修改页面状态的开销。

Table 1 Simulator configuration

表1 模拟器配置参数

项目	参数值
模拟器	GPGPU-Sim 3.x
GPU架构	NVIDIA GTX-480 Fermi-like
GPU核	15 CUs @ 1.4 GHz
一级缓存	16 KB/CU
二级缓存	Memory Side 128 KB/DRAM Channel
快表项数(per CU)	128-entry,4-port,supporting hit under miss
时钟频率/MHz	Core:IC:L2:DRAM 700:700:700:1024
GPU GDDR5	12-channels,384 GB/s aggregate
CPU-GPU互连带宽	16 GB/s or 32 GB/s
迁移缓存条目数	128 Entries/Memory Partition

10个基准测试来自不同的基准测试集或被应用于不同的领域。BFS、MUM和NN来自Rodinia测试集,CP来自Parboil测试集。它们在应用程序属性上有所不同,例如线程的组织、GPU上利用的内存空间、数据量和计算量等[19]。通过对这些应用进行测试,可以显示部分页迁移对具有不同属性的应用程序的影响。

5.2 单一有效范围的部分页迁移

首先,我们测试了页面大小为2 MB时单个有效范围的部分页迁移的性能,所有数据在初始化时都在CPU内存中。单一有效范围的部分页迁移、程序员控制传输和整页迁移的性能比较结果如图4所示。由于整页迁移的执行时间过长,在图4中只给出20以内的标准化时间。

当不采用额外的优化机制时,整页迁移的性能最差,因为2 MB页面的迁移延迟太长,不能重叠,并且大部分被迁移的数据都未得到使用。对于LIB和WP,采用2 MB页面的整页迁移的性能相对可以接受。这是因为它们所请求的数据在内存中分布密集,占用了被迁移页面中相对较大的比例。

当带宽为16 GB/s时,单个有效范围的部分页

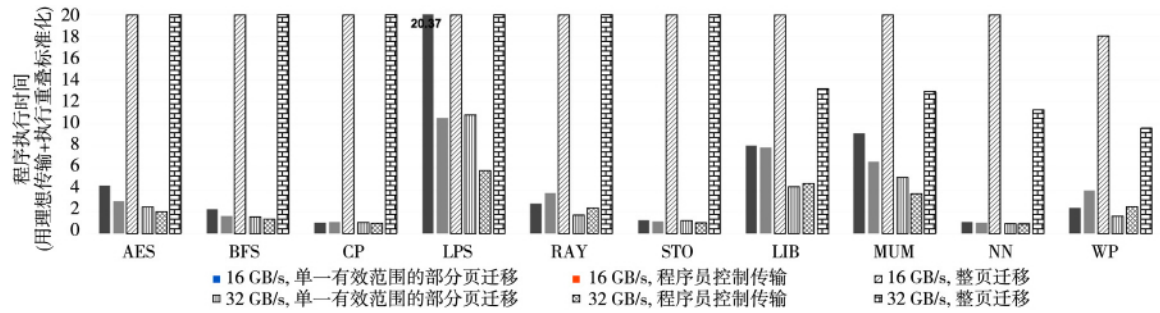

Figure 4 Performance comparison of single valid range partial page migration, programmers controlled transfer and whole-page migration

图 4 单一有效范围的部分页迁移、程序员控制传输和整页迁移的性能比较

迁移的性能远远优于整页迁移的。在许多情况下,与程序员控制传输相比,单个有效范围的部分页迁移可以获得接近甚至更好的性能。但是,对于 LPS 和 MUM,单一有效范围的部分页迁移的性能比程序员控制转移的性能差得多。这种现象证明了单一有效范围的缺陷。与程序员控制传输相比,整页迁移有平均 98.62%倍的减速,而单一有效范围的部分页迁移只有平均 30%的减速。也就是说,单个有效范围的部分页迁移相对于整页迁移获得了平均 55.94 倍的加速。

当带宽从 16 GB/s 增加到 32 GB/s 时,单一有效范围的部分页迁移和程序员控制传输之间的平均性能差距从 30%下降到 10%,且相对于整页迁移获得了平均 70.64 倍的加速。

5.3 多个有效范围的部分页迁移

(1)迁移单位大小。

迁移单元大小对性能具有显著的影响,为了简化实现并分析需要支持的有效范围数量,将每组实验中迁移单位的长度设为一个固定值。实验使用 2 MB 页面和 16 GB/s 带宽,更改迁移单元的大小并确保它大于每个时钟周期可传输的字节数,以避免带宽浪费。

如图 5 所示,对于不同大小的迁移单元,各测试程序的性能改变的总体趋势是一致的。随着迁移单元变大,性能下降,并且下降率与迁移单元大小不成比例。这是由于当迁移单元较小时被计算隐藏的延迟随着迁移单元的增大不再被隐藏。可以看出,对于被请求的数据在内存中密集分布的测试程序(如 LIB),迁移单元可以取得相对较大,以实现更高的带宽利用率。但是,对于被请求的数据在内存中稀疏分布的应用,采用较大的迁移单元会浪费时间来移动不必要的数据,对性能有不利影响。

此外,我们统计分析了应用执行后页面有效范围,结果表明,页面内通常存在 2 个或 3 个不连续的有效范围,因此多个有效范围的部分页迁移中支持的有效范围数可以限制为 4 或 8。

(2)TLB 失效率。

TLB 缓存最近使用的虚拟地址及其地址转换,有助于降低地址转换延迟开销。然而,显著增加的 GPU 内存容量、有限的 TLB 条目数和单指令多线程执行模型使得 GPU 的 TLB 失效率难以降低。上述实验结果表明,部分页迁移可以解决大页面整页迁移的问题,使页迁移机制对大页面有效。可以推测,较大的页面使得 TLB 可以利用有限的条目覆盖较大的存储空间,从而降低 TLB 失效率并改善性能。我们测试了页面大小分别为 4 KB和 2 MB 时部分页迁移的 TLB 失效率。测试

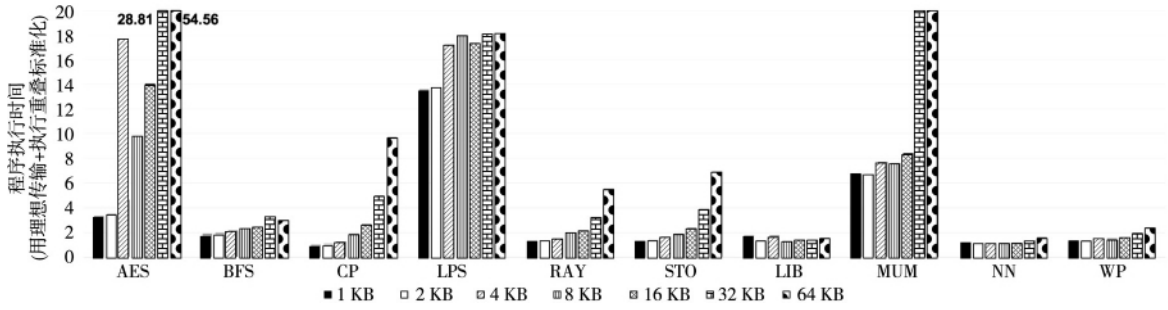

Figure 5 Effect of migration unit size on the performance of partial migration with multiple valid ranges when the bandwidth is 16 GB/s

图 5 带宽为 16 GB/s 时迁移单元大小对多个有效范围的部分迁移性能的影响

所用的 TLB 具有 256 个共享条目,GPU 存储器容量为 1 536 MB。结果表明,当页面大小为 2 MB 时,平均 TLB 失效率为 64.76%,明显低于页面大小为 4 KB 时的 89.72%。

(3)性能比较。

本组实验使用 1 KB 迁移单位作为多个有效范围的部分页迁移的代表,页面大小为 2 MB。实验结果如图 6 所示。当带宽为 16 GB/s 时,多个有效范围的部分页迁移的性能远远优于整页迁移的。在许多情况下,与程序员控制传输相比,它可以获得接近甚至更好的性能。与图 5 展示的单一有效范围的部分页迁移相比,多个有效范围的部分页迁移表现更好。以 LPS 为例,当采用单一有效范围的部分页迁移时,其执行时间接近程序员控制传输的 2 倍。但是,当采用多个有效范围的部分页迁移时,与程序员控制传输的性能差距仅为 30%左右。这一变化证明多个有效范围的部分页迁移可以弥补单个有效范围的部分页迁移的缺陷。与程序员控制转移相比,整页迁移有平均 98.62%倍的减速,而多个有效范围的部分页迁移获得了平均 1.29 倍的加速。也就是说,多个有效范围的部分页迁移获得了相对于整页迁移的 93.81 倍加速。

当带宽从 16 GB/s 增加到 32 GB/s 时,与程序员控制的传输相比,整页迁移有平均 98.72%倍的减速,而多个有效范围的部分页迁移获得了平均 1.22 倍的加速。也就是说,多个有效范围的部分页迁移获得了相对于整页迁移的 94.99 倍加速。

通过设置有限数量的有效范围(根据实验结果,页面最多有 2～3 个不连续的请求范围),部分页迁移不仅比程序员控制传输更简单方便,而且还具有性能优势。这是因为部分页迁移限制了迁移单元的大小,所以迁移延迟更短,可以通过计算隐藏。但是,与理想“传输+执行”重叠相比,对于请求数据在内存中稀疏分布或请求时间间隔较短(如 LPS 和 MUM)的应用,仍存在一些性能差距,因为传输时间不能被完全隐藏。

6 结束语

本文研究了整页迁移中页面增大(从 4 KB 到 2 MB)对应用程序性能的影响。测试结果显示,较大的页面会导致严重的性能下降。当页面较大时,迁移延迟变长,计算和传输的并行度降低,从而导致性能变差。本文提出的部分页迁移机制可以在使用大页面的情况下缩短迁移延迟。本文采用 2 种机制实现了部分页迁移,即单一有效范围的部分页迁移和多个有效范围的部分页迁移。实验结果显示,当页面大小为 2 MB 时,若带宽为 16 GB/s,与程序员控制传输相比,整页迁移有平均 98.62%的减速,而多个有效范围的部分页迁移获得了平均 1.29 倍的加速。实验结果表明,在大多数情况下,部分页迁移相对于整页迁移具有显著的性能优势。部分页迁移不仅能满足高性能工作负载对大页面的需求,而且还可以限制页面大小增加时可能出现的性能下降。

参考文献:

[1] Lindholm E, Nickolls J, Oberman S, et al. NVIDIA Tesla: A unified graphics and computing architecture[J]. IEEE Micro, 2008, 28(2): 39-55.

[2] Di Carlo S, Gambardella G, Martella I, et al. Fault mitigation strategies for CUDA GPUs[C]// Proc of 2013 IEEE International Test Conference (ITC), 2013: 18.

[3] Power J, Hill M D, Wood D A. Supporting x86-64 address translation for 100s of GPU lanes[C]// Proc of 2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA), 2014: 568-578.

[4] Zheng T, Nellans D, Zulfiqar A, et al. Towards high performance paged memory for GPUs[C]// Proc of 2016 IEEE International Symposium on High Performance Computer Archi-

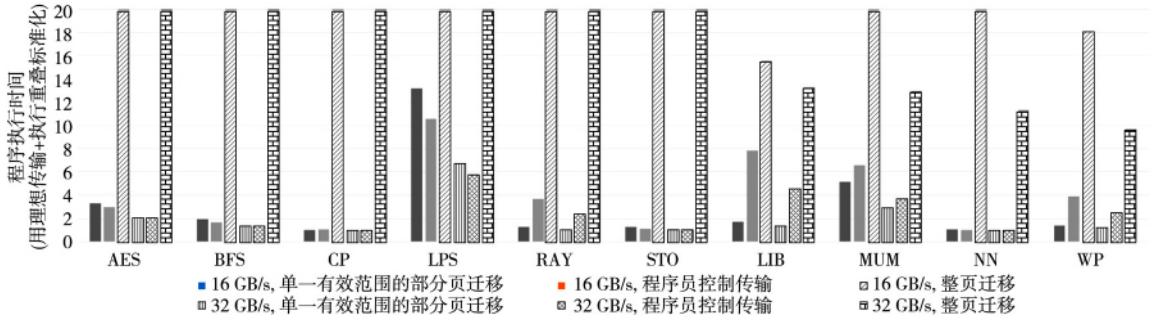

Figure 6 Performance comparison among multiple valid range partial page migration, programmers controlled transfer and whole page migration

图 6 多个有效范围的部分页迁移、程序员控制传输和整页迁移的性能比较

tecture (HPCA),2016:345-357.

[5] Landaverde R,Zhang T,Coskun A K,et al. An investigation of unified memory access performance in CUDA[C]// Proc of IEEE High Performance Extreme Computing Conference (HPEC),2014:1-6.

[6] Kirk D. NVIDIA CUDA software and GPU parallel computing architecture[C]// Proc of ISMM,2007:103-104.

[7] The top 10 innovations in the new NVIDIA Fermi architecture, and the top 3 next challenges [EB/OL]. [2009-09-30]. https://www.nvidia.com/content/PDF/fermi_white_papers/D. Patterson _Top10InnovationsInNVIDIAFermi.pdf.

[8] Hammarlund P,Martinez A J,Bajwa A A,et al. Haswell:The fourth-generation Intel core processor[J]. IEEE Micro,2014,34(2):6-20.

[9] Ghorpade J,Parande J,Kulkarni M,et al. GPGPU processing in CUDA architecture[J]. Advanced Computing,2012,3(1):105.

[10] Rogers P,Fellow A. Heterogeneous system architecture overview[C]// Proc of 2013 IEEE Hot Chips 25 Symposium,2013:1-41.

[11] Kim Y,Lee J,Kim D,et al. ScaleGPU:GPU architecture for memory-unaware GPU programming[J]. IEEE Computer Architecture Letters,2014,13(2):101-104.

[12] Lustig D,Martonosi M. Reducing GPU offload latency via fine-grained CPU-GPU synchronization[C]// Proc of 2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA),2013:354-365.

[13] Cao Y, Chen L,Zhang Z. Flexible memory:A novel main memory architecture with block-level memory compression [C]// Proc of 2015 IEEE International Conference on Networking,Architecture and Storage (NAS),2015:285-294.

[14] Agarwal N,Nellans D,Stephenson M,et al. Page placement strategies for GPUs within heterogeneous memory systems [C]// Proc of ACM SIGPLAN Notices,2015:607-618.

[15] Agarwal N,Nellans D,Connor M O,et al. Unlocking bandwidth for GPUs in cc-numa systems[C]// Proc of 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA),2015:354-365.

[16] GPGPUSim 3.x manual [EB/OL]. [2017-06-13]. http://gpgpu-sim.org/manual/index.php/Main_Page.

[17] Ajanovic J. PCI Express 3.0 overview[C]// Proc of 2009 IEEE Hot Chips:21 Symposium,2009:1-61.

[18] PCI Express 4.0 electrical previews[EB/OL]. [2018-50-01]. http://pcisig.com/sites/default/files/files/01_05_PCIe_4_0_Electrical_Previews_FROZEN.pdf.

[19] Bakhoda A, Yuan G L,Fung W W,et al. Analyzing cuda workloads using a detailed GPU simulator[C]// Proc of 2009 IEEE International Symposium on Performance Analysis of Systems and Software(ISPASS),2009:163-174.

作者简介:

张诗情(1994-),**女,四川眉山人,硕士生,研究方向为并行编程和优化技术。E-mail**:zhangshiqing12@nudt.edu.cn

ZHANG Shi-qing, born in 1994, MS candidate, her research interests include parallel programming, and optimization techniques.

杨耀华(1994-),**男,山东临沂人,硕士生,研究方向为高性能处理器和优化技术。E-mail**:yangyaohua16@nudt.edu.cn

YANG Yao-hua, born in 1994, MS candidate, his research interests include high performance processor, and optimization techniques.

沈立(1975-),**男,四川成都人,博士,教授,CCF 会员**(12903S),**研究方向为计算机体系结构、系统结构虚拟化和动态编译优化。E-mail**:lishen@nudt.edu.cn

SHEN Li, born in 1975, PhD, professor, CCF member (12903S), his research interests include computer architecture, system architecture virtualization, and dynamic compilation optimization.

王志英(1956-),**男,山西长治人,博士,教授,CCF 会员**(E2000055595),**研究方向为高性能计算机体系结构、异步微处理器设计和计算机系统安全。E-mail**:zywang@nudt.edu.cn

WANG Zhi-ying, born in 1956, PhD, professor, CCF member(E2000055595), his research interests include high performance computer architecture, asynchronous microprocessor design, and computer system security.