doi:10.11887/j.cn.201501005 http://journal.nudt.edu.cn

异构千核处理器系统的统一内存地址空间访问方法*

裴颂文[1,2,3],吴小东[1],唐作其[4],熊乃学[5]

(1.上海理工大学 计算机科学与工程系,上海 200093;
2.中国科学院 计算机体系结构国家重点实验室,北京 100190;
3.加利福尼亚大学 电气工程与计算机科学系,加利福尼亚 92697;
4.贵州大学 计算机科学与技术学院,贵州 贵阳 550025;
5.科罗拉多科技大学 计算机科学学院,科罗拉多 80907)

摘 要:为了达到异构多核处理器能直接交叉访问对方的内存地址空间的目的,通过构建统一的三级Cache结构和数据块状态标记方法,并优化Cache块状态的修改算法,提出了异构千核处理器系统的统一内存地址空间访问方法,避免了当前独立式异构计算机系统结构下复制和传输数据块所带来的大量额外访存开销。通过采用部分Rodinia基准测试程序测试,获得了最高9.8倍的系统加速比,最多减少了90%的访存频率。因此,采用该方法能有效减少异构核心间交换数据块所带来的系统开销,提高异构千核处理器的系统性能加速比。

关键词:异构千核处理器;内存地址空间;交叉式直接访问;Cache
中图分类号:TN95 **文献标志码**:A **文章编号**:1001-2486(2015)01-028-06

An approach to accessing unified memory address space of heterogeneous kilo-cores system

PEI Songwen[1,2,3], *WU Xiaodong*[1], *TANG Zuoqi*[4], *XIONG Naixue*[5]

(1. Department of Computer Science & Engineering, University of Shanghai for Science and Technology, Shanghai 200093, China;
2. State Key Laboratory of Computer Architecture, Chinese Academy of Sciences, Beijing 100190, China;
3. Department of Electrical Engineering and Computer Science, University of California, California 92697, United States;
4. School of Computer Science and Technology, University of Guizhou, Guiyang 550025, China;
5. School of Computer Science, Colorado Technical University, Colorado 80907, United States)

Abstract: In order to access independent memory space of CPU and GPU directly from opposite directions, an effective approach to accessing unified memory address space of heterogeneous kilo-cores system is proposed, which is implemented by building a unified 3-level Cache and tagging blocks in Cache, and optimizing the algorithms of modifying the states of blocks. Therefore, the heterogeneous kilo-cores system avoids significant overhead of accessing memory instead of that in current discrete hybrid computer system equipped with GPUs by PCI-E. According to the results of experiments from partial programs of Rodinia benchmarks, a maximal speedup by 9.8x and maximal decrease of load/store instructions by 90% are gained. In conclusion, it's certified that our solution is effective to decrease overhead of transferring data among computing units in heterogeneous system and significantly enhance the whole system computing performance.

Key words: heterogeneous kilo-cores processors; memory address space; directly access from opposite directions; Cache

随着众核设计、三维芯片制造等技术的发展,单位面积晶体管的数量将会继续按照摩尔定律增长,这种趋势将会使处理器的设计延续更多的计算核心,更大的共享Cache,从而使得单片千核处理器的应用也不再遥远[1]。与此同时,异构计算机系统的研究及应用也成为学术界和工业界的热点。异构多核处理器(heterogeneous multicores)能获得比对称多核处理器(symmetric multicores)和非对称多核处理器(asymmetric multicores)更好的性能[2]。图形处理器(Graphic Processing Unit,GPU)等作为加速计算部件处理数据流或者向量数据的作用越来越受到重视,各主流处理器制造商也相继推出新型的异构多核处理器,如AMD的Fusion架构处理器[3],Intel的Xeon Phi架构处

* **收稿日期**:2014-06-10
基金项目:计算机体系结构国家重点实验室开放资助项目(CARCH201206);上海理工大学国家级项目培育基金资助项目(12XGQ07);贵阳市科技计划项目(2011101414);贵州省科技支撑项目(20123050)
作者简介:裴颂文(1981—),男,湖南邵东人,博士,硕士生导师,E-mail:swpei@usst.edu.cn

理器[4],Nvidia 的 Denver 项目[5] 和 ARM big. LITTLE[6]等,这些新型异构处理器依赖大量高性能的流处理/向量计算单元/超标量顺序处理单元作为协处理器加速浮点运算,增加多线程并发度,提高系统性能。

当前独立式 GPU 的异构计算机系统在数据传输、计算核心启动、Cache 一致性管理和数据同步方面产生大量的额外开销。Nvidia Tesla C2050 GPU 与显存通信的峰值带宽达到144GB/s,而主机和 GPU 通过PCI-E相连的峰值传输带宽却只有8GB/s。这种数据传输速率之间的巨大差异是导致独立式异构计算机访存系统性能瓶颈的重要因素。例如, memcpy 函数将一个 128K 字节的数据从主机 CPU 端传输到 GPU 端所产生的物理传输延迟占整个数据传输时间的 70%[7]。Daga 等[8]证明真正的单片异构系统计算机,如 AMD 的加速处理单元(Accelerated Processing Unit, APU),较前两类异构系统具有更好的性能。Hwu 等[9]也同样指出 GPU 端和 CPU 端数据传输的巨大开销是异构系统发展的瓶颈。因此, 随着千核处理器计算核心数量的增加,有效降低 CPU 和 GPU 之间传输数据的开销是突破异构千核处理器存储墙的重要方向,也是研究异构千核处理器系统的主要难题之一。

基于前期对异构千核高通量处理器系统模型的研究成果,提出了异构计算机系统统一的内存地址空间访问方法,通过交叉式直接访问对方内存地址空间的方法消除 GPU 和 CPU 通过PCI-E总线传输数据的开销,避免了数据块在异构系统内存空间中进行多次读写操作,降低了系统访存频率,提高异构计算机系统的计算性能。特别是对于具有大规模数据块读写操作的应用计算,具有相当明显的系统加速比。

1　异构千核高通量处理器系统的基本结构

在典型的同构千核处理器架构 Rigel[10]的基础上,采用类似的三级互联架构提出了基于通用图形处理器(General Purpose Grapsics Processing Unit,GPGPU)加速计算单元的异构千核高通量处理器(Heterogeneous Kilo-cores High-throughput Processors,HKHP)系统架构。如图 1 所示,HKHP 由三级计算单元组成,分别是 CPU/GPU 计算核心级、Quart 计算簇级和 Tile 计算簇级。4 个 CPU 计算核心组成的 CPU 簇和 12 个 GPU 计算核心组成的 GPU 簇,共同构成 Quart 计算簇。该单元内的 4 个 CPU 和 12 个 GPU 有各自独立的一级 Cache,12 个 GPU 共享二级 GPU 数据 Cache,4 个 CPU 共享二级 CPU 数据 Cache。因此,每个 Quart 共包含 16 个异构计算单元。4 个 Quart 组成一个 Tile,Quart 之间通过高速交叉网络互连,每个 Tile 含 64 个异构计算单元,并采用多端口队列机制分离访存指令和纯计算指令。每个 Quart 配置有一个 Cache 管理处理器(Cache Management Processor,CMP)负责为 Quart 中各计算核心预取和管理数据块。

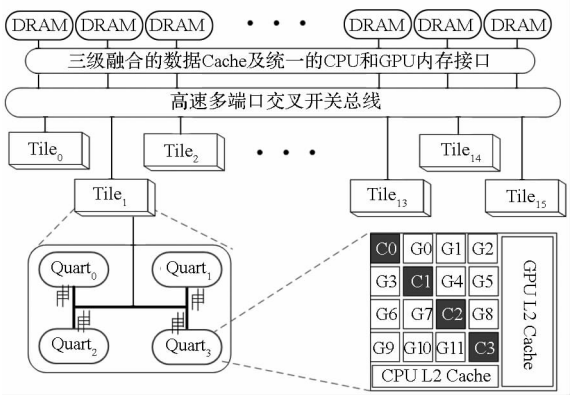

图 1　异构千核高通量处理器架构

Fig. 1　Framework of heterogeneous kilo-cores high-throughput processors

HKHP 采用分离式访存和计算指令的设计,便于重叠访存和计算指令,提高指令流水线效率以及延迟隐藏异构系统间数据传输的长时间开销。16 个 Tile 通过高速多端口交叉开关网络连接,共享三级融合的 Cache,该 Cache 既可以为 GPU 计算核心缓存数据,也可以为 CPU 计算单元缓存数据,并支持交叉式直接访问统一的物理内存。

2　统一的内存地址空间访问方法

为了避免当前 CPU 内存地址空间和 GPU 内存地址空间分离访问所带来的异构系统间数据块迁移开销庞大的缺陷,对异构千核处理器系统中的数据块访问构建统一的物理内存地址访问方法。

根据研究表明,数据块的访问和迁移是导致异构系统访存开销过大的根本原因。因此,提出的统一物理内存地址访问机制主要考量的是数据块访问在存储系统中的操作流程和传输开销,暂不考虑指令流的访存过程和开销。

2.1　统一的逻辑内存地址空间

构建的四级存储系统,如图 2 所示,各 CPU 计算核心和 GPU 计算核心分别含有私有的一级 Cache。每个 Quart 内的 CPU 计算核心共享二级 CPU 数据 Cache,GPU 计算核心共享二级 GPU 数

据 Cache。二级 CPU 数据 Cache 和二级 GPU 数据 Cache 是独立的物理 Cache;所有的 64 个 Quart 共享统一的三级数据 Cache,混合式缓存 CPU 数据块和 GPU 数据块。GPU 和 CPU 数据块在三级数据 Cache 采用类似文献[16]的 Cache 一致性目录机制实现同步和一致性管理。

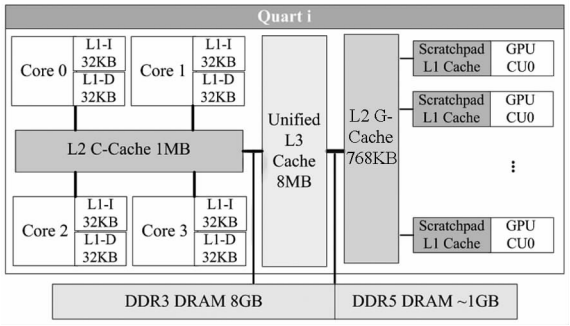

图 2 四级存储结构图

Fig. 2 4-level memory hierarchies

同时,为了细粒度刻画数据块的所有者,将额外增加 6 位标识位表示数据块的 Quart 属性,其中高 4 位表示 Tile 的编号,低 2 位表示某个 Tile 内的 Quart 编号。

三级数据 Cache 可以直接访问统一的物理内存地址空间,CPU 和 GPU 数据块都可以统一存放在统一的三级数据 Cache 里,通过数据块的状态标识位区分数据块的所有者属性。给每个数据块分配 2 位状态位,状态标识位标识数据块的操作所有者属性,数据块的状态标识位随着数据块的处理过程更新。数据块状态更新算法采用有限状态机描述各状态的动态迁移过程。数据块状态标识位功能如表 1 所示。

表 1 数据块状态标识位

Tab. 1 Tags of data blocks′ states

状态位	说明
0,0	无状态,新数据块,CPU 和 GPU 均可以无限制访问
1,0	CPU 私有数据块:数据块的所有者属于 CPU 计算核心,GPU 只能读,不能写此块
0,1	GPU 私有数据块:数据块的所有者属于 GPU 计算核心,CPU 只能读,不能写此块
1,1	共享数据块:数据块的所有者既属于 GPU 计算核心,又属于 CPU 计算核心,CPU 和 GPU 都可以对该数据块进行读写操作

2.2 统一的三级 Cache 的数据块修改策略

CPU 或 GPU 对融合的三级 Cache 中的数据块更新和同步的基本原则是:如果 CPU 或者 GPU 发起的修改操作是对本 Tile 及本 Quart 的私有三级数据 Cache 块,则执行传统的写回机制,在减少总线通信带宽的条件下保证 Cache 数据一致性。如果 CPU 或者 GPU 发起的修改操作是针对本 Quart 外的共享三级数据 Cache 块,为了保证 Cache 的严格一致性,则采用写直达法并反向同步到一级数据 Cache 和二级数据 Cache 或者 GPU 的一级数据 Cache 和二级共享数据 Cache。修改统一的三级 Cache 数据块的基本操作规则包括以下六项。

规则一:如果 CPU 修改标记为 CPU 且是本 Tile 及本 Quart 私有的三级数据 Cache 的数据块,则对三级数据 Cache 的数据块采用写回法;若 CPU 修改标记为 CPU 却不是本 Tile 及本 Quart 私有的三级数据 Cache 的数据块,则根据传统 Cache 一致性协议(Modified, Exclusive, Share, and Invalid,MESI)修改和更新 Cache 数据块。

规则二:如果 GPU 修改标记为 GPU 且是本 Tile 及本 Quart 私有的三级数据 Cache 的数据块,则对三级数据 Cache 的数据块采用写回法;若 GPU 修改标记为 GPU 却不是本 Tile 及本 Quart 私有的三级数据 Cache 的数据块,则根据传统 Cache 一致性协议 MESI 修改和更新 Cache 数据块。

规则三:如果 CPU 修改标记为共享的三级数据 Cache 的数据块,则对三级数据 Cache 的数据块采用写直达,修改数据块的状态标识到 CPU 私有状态,并反向同步到 CPU 的 L1 和 L2 数据 Cache。

规则四:如果 GPU 修改标记为共享的三级数据 Cache 的数据块,则对三级数据 Cache 的数据块采用写直达,修改数据块的状态标识到 GPU 私有状态,并反向同步到 GPU 全局 Cache 和共享 Cache。

规则五:如果 CPU 修改标记为 GPU 的私有三级 Cache 数据块,则先请求获得相应 GPU 计算单元的授权并请求数据块的有限状态机修改数据块到共享状态,然后执行规则三。

规则六:如果 GPU 修改标记为 CPU 的私有三级 Cache 数据块,则先请求获得相应 CPU 计算单元的授权并请求数据块的有限状态机修改数据块到共享状态,然后执行规则四。

基于以上六项关于修改三级 Cache 数据块的基本规则,CPU 和 GPU 发起的读写操作可以同时访问三级 Cache 和统一的物理内存,如算法 1 所示。

算法 1　三级融合 Cache 的数据块修改算法

Alg. 1　Modifying data in L3 unified Cache

```
1.  if CPU 修改(1,0) 标识的数据块地址 then
2.     { Write Back; } else if
3.     CPU 修改(1,1)标识的数据块地址 then
4.     { Write Through 统一的三级数据 Cache;
5.       update_data_status( );//更新数据块状态信息标
         识为(1,0);
6.       anti_sync( );//反向同步到 CPU 的 L1 和 L2 数
         据 Cache;} else if
7.     CPU 修改操作不命中 L3 Cache then
8.     {update_memory( );//修改内存地址块;
9.       flush_13_cache( );//将对应的内存地址块调入
         L3 Cache;
10.      lru_replace( );//执行 LRU Cache 块替换算法;}
11.  end if
12.  if GPU 修改(0,1) 标识的数据块地址 & 命中 L3
       Cache then
13.    {Write Back; } else if
14.    GPU 修改(1,1) 标识的数据块地址 & 命中 L3
       Cache then
15.    {Write Through 统一的三级 Cache;
16.      update_data_status( );//更新数据块状态信息
         标识为(0,1);
17.      anti_sync( );//反向同步到 GPU 全局 Cache 和
         共享 Cache } else if
18.    GPU 修改操作不命中 L3 Cache then
19.    {update_memory( );//修改内存地址块;
20.      flush_13_cache( );//将对应的内存地址块调入
         L3 Cache;
21.      lru_replace( );//执行 LRU Cache 块替换算法;}
22.  end if
```

面向融合的三级数据 Cache 及统一的物理内存地址空间,对于典型的标识为(1,0),(0,1)和(1,1)数据块进行修改操作时,CPU 和 GPU 分别修改数据块算法的伪代码如算法 1 所示。通过设计有效地三级 Cache 一致性访问机制,并给相应的物理内存地址的数据块分配统一的标识位和状态位,可以有效地管理融合的三级数据 Cache 和支持交叉式直接访问统一的物理内存地址空间。

3　性能评价

仿真实验主要是在 64 位 Linux 操作系统平台上用 Gem5[11] 和 GPGPU - Sim[12] 构建异构千核处理器系统,包括构建指令分离式模块、多级 Cache 目录机制、队列缓冲模块、数据块状态更新算法等。然后,采用真实的基准测试程序集 Rodinia[13] 测试系统性能加速比。通过设计和开发 Gem5 和 GPGPU - Sim 的中间件,在 Gem5 CPU 端通过 libcuda 调用计算机统一设备架构(Compute Unified Device Architecture,CUDA)GPU 线程,给 Ruby 内存系统发送 GPU 的全局和常量内存地址请求,接收来自 GPGPU - Sim 端的全局和常量数据请求,存储和管理并行线程执行(Parallel Thread Execution,PTX)代码和变量等;在 GPGPU - Sim GPU 端接收 CPU 线程上下文的通知信号,处理 PTX 代码、变量和检查点,管理 GPU 内存空间页表等。实验参数配置如表 2 所示。

表 2　全系统仿真实验参数配置表

Tab. 2　Parameters of configuring simulations

CPU	1GHz,4 × 4 × 16 共 256 个计算核心
CPU L1 数据 Cache	32KB
CPU L1 指令 Cache	32KB
CPU L2 数据 Cache	1MB (8 - way)
GPU	600MHz,12 × 4 × 16 共 768 计算单元
GPU L1 Cache	64KB
GPU L2 数据 Cache	768KB (16 - way)
L3 Unified 数据 Cache	8MB (16 - way)
CPU DDR3	8GB SDRAM
GPU GDDR5	1GB SDRAM

通过选取 Rodinia 测试集中的 Back Progagation(backprop), Breadth-first Search(bfs), Computational Fluid Dynamics Solver (cfd), Gaussian Elimination (gaussian), K-means (ks), LU Decomposition (lud), HotSpot(hs), Needleman-Wunsch(nw),系统性能加速比如图 3 所示。通过配置等价的系统参数,对比交叉式直接访问统一的内存地址空间的方法和传统的分离式内存访问方法,仿真实验结果显示:cfd 流体动力学程序、nw 和 gaussian 程序采用了较大规模的输入数据量,在 CPU 和 GPU 之间具有大量的数据流传输,因此采用交叉式直接内存访问方式后,相比于传统的分离式异构处理器 cfd 获得了 9.8 倍的计算加速比。测试 bfs,lud 程序时采用的是小规模的矩阵计算,GPU 内存可以一次存储起来,CPU 和 GPU 之间不需要大量的数据传输交换。因此,采用统一内存地址空间的直接内存访问并不能带来明显的效果,反而由于内存地址的多次转换降低了系统执行效率。对于小规模的K - means算法

尤其明显,加速比反而降低了40%。

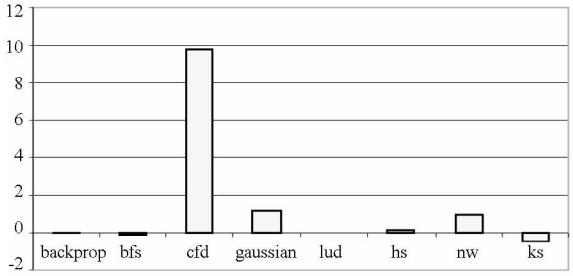

图3　执行加速比

Fig. 3　Execution Speedup

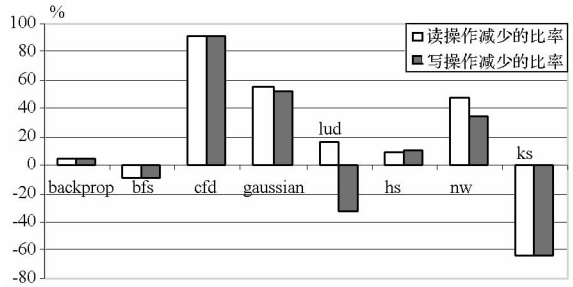

图4　读写操作的减少率

Fig. 4　The percentage of decreasing read/write operations

如图4所示,采用统一的内存地址空间交叉直接访问方法后,对于多数测试程序都能有效减少读写操作的次数,降低了访问内存的频率,提高了系统计算加速比。对比图3和图4,容易得知,系统性能加速比的提高主要得益于数据块在CPU和GPU内存之间读写次数的降低。比如,cfd程序最高降低了90%的读写频率,获得了最高的计算性能加速比。相反,ks程序却反向增加了64%的读写操作频率,严重影响了系统计算性能加速比。lud程序减少的读操作比率和增加的写操作比率基本相当,因此lud程序的计算性能基本保持不变。总之,采用统一内存地址访问方法对于高通量、大数据传输的应用具有明显的加速作用。

4　相关工作

Kelm等[14]采用软硬件协同设计的方法提出了混合式内存模型,避免了数据块的复制操作和多地址空间,减小了消息通信开销和片上的Cache目录面积。但是数据块的一致性状态转换结构非常复杂,软硬件状态转换的同步过程是Cache一致性的瓶颈。Ham等[15]提出的异构内存系统采用层次化的缓冲区桥接变相位内存(Phase Change Memory,PCM)模块和动态随机访问内存(Dynamic Random Acessing Memory,DRAM)模块,采用分离控制的方式提高系统的能效率。Power等[16]提出的CPU-GPU异构系统一致性机制是采用区域缓冲区和区域目录的结构提高异构系统的Cache一致性。Hechtman等[17]采用共享虚存的方式维持异构多核系统的Cache一致性,认为CPU Cache是按照访问时延最优化设计的,GPU Cache是按照访问吞吐率最优化设计的,很难共享二者的Cache数据块。因此,采用一个独立的目录结构来共享数据块,以避免CPU和GPU间通过访问片外内存来交换数据块的巨大开销。裴颂文等[18-19]提出的面向异构千核计算机的统一的物理内存访问框架(UPM)能协调GPU端和CPU端的数据交换,通过异构核心间交叉式直接访问对方物理内存地址空间的方式来避免显式的数据交换,从而降低数据传输的额外开销,提高存储访问效率。AMD等发起的异构系统架构(Heterogeneous System Architecture,HSA)是一个统一的计算框架,为避免CPU和GPU显式地传输数据而提出单一的逻辑地址空间访问方式——用户空间队列机制和抢占式上下文交换技术[20]。

5　结论

当前独立式GPU通过PCI-E接口与CPU相连,构成的混合式计算系统是主流的异构计算系统结构。这种异构计算结构中的主要系统性能瓶颈就是双方数据通信速率的不均衡所导致的大量数据传输开销。通过提出统一的三级Cache结构和内存地址空间访问方法,可以有效避免数据副本的传输开销,降低系统访存指令的数量,提高系统的计算性能。由于目前的仿真实验仅统一了DDR3和GDDR5内存模块的逻辑地址空间,尚未设定不同类型内存模块接口的访问带宽参数等,有待进一步优化内存模型。因此,下一步的工作将在数据块预取技术、分离并重叠执行计算指令和访存指令、多线程执行技术和功耗管理、片上网络(Network on Chip)拓扑和通信带宽模型等方面进一步优化系统性能,并与其他相关工作横向比较计算性能和访存时间。

参考文献(References)

[1] Borkar S. Thousand core chips:a technology perspective[C]// Proceedings of the 44th Annual Design Automation Conference (DAC), San Diego, California, 2007: 746-749.

[2] Chung E S, Milder P A, Hoe J C, et al. Single-chip heterogeneous computing: does the future include custom logic, FPGAs, and GPGPUs[C]//Proceedings of the 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), Atlanta, GA, 2010: 225-236.

[3] Brookwood N. AMD fusion family of APUs: enabling a superior, immersive PC experience[EB/OL]. [2014 - 06 - 10]. http://www.amd.com.

[4] Intel haswell microarchitecture[EB/OL]. Intel Corpaoration. [2014 - 06 - 10]. http://www.intel.com.

[5] Nvidia project denver[EB/OL]. Nvidia Corporation. [2014 - 06 - 10]. http://www.nvidia.com.

[6] Big. LITTLE processing[EB/OL]. ARM Corporation[2014 - 06 - 10]. http://www.arm.com.

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

[8] Daga M, Aji A M, Feng W. On the efficacy of a fused CPU + GPU processor (or APU) for parallel computing[C]// Proceedings of the 2011 Symposium on Application Accelerators in High-Performance Computing, Knoxville Tennessee, 2011: 141 - 149.

[9] Hwu W. Rethinking computer architecture for throughput computing[C]//Keynote of the 2013 International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS), Greece, 2013:1 - 29.

[10] Johnson D R, Kelm J H, Crago N C, et al. Rigel: a scalable architecture for 1000 + core accelerators[J]. IEEE Micro, 2011, 31(4):30 - 41.

[11] Binkert N, Beckmann B, Black G, et al. The gem5 simulator[J]. ACM SIGARCH Computer Architecture News, 2011, 39(2):1 - 7.

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

[13] Che S, Boyer M, Meng J Y, et al. Rodinia: a benchmark suite for heterogeneous computing[C]//Proceedings of the 2009 IEEE International Symposium on Workload Characterization (IISWC), 2009:44 - 54.

[14] Kelm J H, Johnson D R, Tuohy W, et al. Cohesion: an adaptive hybrid memory model for accelerators[J]. IEEE Micro, 2011, 31(1):42 - 55.

[15] Ham T J, Chelepalli B K, Xue N, et al. Disintegrated control for energy-efficient and heterogeneous memory systems[C]// Proceedings of IEEE 19th International Symposium on High Performance Computer Architecture (HPCA), Shenzhen, 2013:424 - 435.

[16] Power J, Basu A, Gu J L, et al. Heterogeneous system coherence for integrated CPU-GPU systems[C]//Proceedings of the 46th Annual IEEE International Symposium on Microarchitecture (MICRO), California, 2013:457 - 467.

[17] Hechtman B A, Sorin D J. Evaluating cache coherent shared virtual memory for heterogeneous multicore chips[C]// Proceedings of IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Austin, Texas, 2013:118 - 119.

[18] Pei S W, Gaudiot J L. Decoupled memory system for heterogeneous kilo-core massively-threaded throughput processor[R/OL]. [2013 - 10 - 01][2014 - 06 - 10], http://sites.uci.edu/songwenpei.

[19] Pei S W, Kim M S, Gaudiot J L, et al. Fusion coherence: scalable cache coherence for heterogeneous kilo-core system[C]// Proceedings of 2014 Annual Conference of Advance Computer Architecture (ACA2014), Shenyang, 2014:1 - 15.

[20] Heterogeneous system architecture: a technical review[EB/OL]. AMD Corporation. [2012 - 08 - 30][2014 - 06 - 10] http://developer.amd.com/wordpress/media/2012/10/hsa10.pdf.