基于 CPU-GPU 异构环境的运算代价评估模型

江慧芳,蔡 达,王晓蕊

(中国矿业大学 计算机科学与技术学院,江苏 徐州 221116)

摘 要:传统性能分析模型仅针对单个处理器,未考虑异构系统中处理器之间数据的传输开销,不能有效地评估异构系统的性能。为此,提出一种运算代价评估模型。通过对计算平台硬件参数和工作负载特征属性的分层建模,结合 LogGP 模型和 Roofline 模型,估算不同执行方式的运算时间成本。依照建模的粒度粗细,通过多层建模计算消耗的能量,采用 LogGP 模型实现 CPU 与 GPU 之间的数据传输,并运用 EPCC 测试集对运算代价模型进行验证。实验结果表明,该模型对处理不同大小数据集时的性能评估具有较高的准确性。

关键词:运算代价模型;性能分析;异构系统;功耗;LogGP 模型;Roofline 模型;EPCC 测试集

中文引用格式:江慧芳,蔡 达,王晓蕊. 基于 CPU-GPU 异构环境的运算代价评估模型[J]. 计算机工程,2017,43(9):12-16.

英文引用格式:JIANG Huifang, CAI Da, WANG Xiaorui. Computation Cost Evaluation Model Based on CPU-GPU Heterogeneous Environment[J]. Computer Engineering, 2017, 43(9): 12-16.

Computation Cost Evaluation Model Based on CPU-GPU Heterogeneous Environment

JIANG Huifang, CAI Da, WANG Xiaorui

(School of Computer Science and Technology, University of Mining and Technology, Xuzhou, Jiangsu 221116, China)

[Abstract] Traditional performance analysis models are designed only for a single processor. They do not consider the cost of data transfer between different processors and cannot be applied to performance analysis of heterogeneous systems effectively. A computation cost evaluation model based on CPU-GPU heterogeneous environment is presented. It estimates the computation time cost of the two execution mode by making models of hardware parameters and workload characteristics and combining LogGP and Roofline models. In addition, it calculates consumption energy by making a model with some levels. It applies LogGP model to transfer data between CPU and GPU. Besides, and uses EPCC benchmark to verify the accuracy. Experimental results show that the improved model can provide a higher accuracy on the performance evaluation of heterogeneous system when it deals with different kinds of data sets.

[Key words] computation cost model; performance analysis; heterogeneous system; power consumption; LogGP model; Roofline model; EPCC test set

DOI:10.3969/j.issn.1000-3428.2017.09.003

0 概述

传统处理器从单核发展为多核,性能和计算能力都有显著提高。而随着功耗问题的突出,性能无法通过核心数量的继续增加而显著提升。为提高计算能力,将 CPU 与加速设备相连接的异构系统^[1-2]应运而生。近年来,GPU 作为高性能计算处理器^[3-4]被广泛应用于运算密集型应用程序,具有比 CPU 优势更明显的吞吐量,是目前最为常见的加速设备之一。以 CPU 为代表的多核模型通过更大的缓存来

提升性能,以 GPU 为代表的多线程模型则通过大量并发的线程掩蔽缓存未命中带来的性能损失^[5]。对于 CPU 或者 GPU,目前已有研究者提出性能评估模型。

性能评估是微体系结构研究的基础,用于探索微体系结构的设计空间以及预测不同工作负载运行在不同处理器上的性能。通过选定不同特性的工作负载^[6],比如运算密集型、IO 密集型或者访存密集型等,再运用性能模型来评估微体系结构的性能。性能模型分为模拟模型和分析模型 2 种。模拟模型

基金项目:国家自然科学基金(U1510115)。

作者简介:江慧芳(1992—),女,硕士研究生,主研方向为可信计算、电子商务;蔡 达、王晓蕊,硕士研究生。

收稿日期:2016-09-02 **修回日期:**2016-10-13 **E-mail:**jhf@cumt.edu.cn

对微架构的不同层次进行抽象,可以从时序到功能用不同的粒度模拟处理器的行为。其能够提供较准确的性能预测,是性能评估的主要研究方法。目前,McPAT^[7]提供了针对 CPU 的模拟器,使用了集成能耗、面积和时序的多核架构模型框架,且包含多种处理器的基本组件。而 GPGPUSim^[8]则是针对 GPU 设计的模拟器。2 种模拟器精准性较高,但效率极低,模拟现实处理器所需要的运行时间较长。另外,文献[9]指出,依赖模拟模型的性能研究会因为模拟器中可能存在的 bug 受到影响,导致不恰当的结论。而分析模型则是考虑微架构硬件参数以及工作负载的特征参数,以数学表达式的方式来描述性能模型。相比模拟模型,分析模型没有运行模拟器的开销,速度要快得多,并且更具洞察力。文献[10]提出了基于 MWP(Memory Warp Parallelism)和 CWP(Computation Warp Parallelism)的 GPU 性能分析模型,但该模型预测的内核运行时间存在误差,并且没有对 CPU 与 GPU 之间进行深入的性能比较。文献[11]提出用可同时运行的线程数量、访存指令占比以及缓存命中率作为参数来描述并行工作负载的方法,并给出了综合工作负载特性、缓存和多线程的分析模型。

然而,现有的性能评价研究较少涉及 CPU 和 GPU 组成的异构系统的性能模型。在异构系统中,需要将数据由主存传送到显存中,交由 GPU 进行运算,之后将计算结果由显存传送至主存。CPU 与 GPU 之间的数据传输过程会产生额外开销,而传统性能模型并没有考虑这部分传输开销。文献[12]以 3D 傅里叶变换运算为例,预测在今后的工作负载中节点间的通信是主要瓶颈。

CPU 与 GPU 之间的数据传输可以通过多种方式进行,文献[13]分析了 2 种方式对性能的影响:一种是异步传输;另一种是将数据块分成多个传输流使得数据传输和运算重叠进行。文献[14]提出 LogGP 模型对 CPU 和 GPU 的数据传输进行研究,分别对显示传输、隐式传输、流传输以及混合传输 4 种传输方式建立数学模型,并指出当采用某种方式时,其对应的运算时间和传输时间之间应该满足关系。文献[13-14]均没有给出运算时间的估计方法。

针对上述研究,本文使用 LogGP 模型进行 CPU 与 GPU 的数据传输,对现有模型进行改进,提出一种新的运算代价模型,并对 CPU 与 GPU 组成的异构系统的性能进行评估。

1 运算代价模型

本文提出的运算代价模型分别考虑了系统的硬件参数以及工作负载的特征参数,并从运算时间成本以及运算所消耗的能量两方面进行计算,从而评估系统性能。运算代价模型的计算公式如下:

$$Cost = \omega_1 \times Time + \omega_2 \times Energie = [Time, Energie] \omega^T \quad (1)$$

其中, ω_1, ω_2 分别为运算时间 *Time* 与运算功耗 *Energie* 的权重,且 $\omega_1 + \omega_2 = 1$ 。

运算时间成本包括片上运行时间 *ExecTime* 和数据传送时间 *TransTime*。虽然片上运行时间无法精确获得,但用 Roofline 模型^[15]可获得分别在 CPU 和 GPU 上进行浮点计算的理论峰值,依照吞吐量的峰值可以得到片上运行时间的估计。而主存与显存间的数据传送时间需要结合 LogGP 模型,对使用 GPU 运算的时间详细建模。而完成计算任务所消耗的能量,则需要综合考虑处理器制成水平、处理器中各个功能模块的耗电情况以及数据传输总线的耗电情况。

1.1 模型参数

由单个 CPU 与单个 GPU 组成的异构平台,将其抽象为通过高速总线连接的 2 个运算节点,可表示成 $Platform = \langle Node_{CPU}, Node_{GPU}, Bus \rangle$ 。而每个节点则由处理器、处理器缓存和存储器 3 个模块构成,即 $Node = \langle Processor, BW, S_{mem}, t_{mem} \rangle$ 。对于每种模块,考虑如下硬件参数:

1) 存储器。存储器大小、访存带宽与访存延迟。

2) 处理器。处理器指令集架构 (Instruction Set Architecture, ISA), 处理器的核心频率 f , 线程总数 N_{pe} , 缓存大小 S_s , 缓存的延迟 t_s 以及热设计功耗 *TDP*。处理器模块可表示为 $Processor = \langle f, N_{pe}, S_s, ISA, TDP \rangle$ 。

3) 缓存。缓存只计算片上最后一级缓存。现阶段的 CPU 一般是 L3 级缓存,而 GPU 是 L2 级缓存。

而对于工作负载的特征参数,本文模型则考虑了数据集大小 *DataSize*、数据操作类型 Π 、数据局部性参数 α 和 β 。工作负载可抽象表示为 $workload = \langle DataSize, \Pi, \alpha, \beta \rangle$ 。

1.2 运算时间计算

对于一个工作负载,本文讨论 2 种情况下的运算时间:只用 CPU 进行运算以及将运算密集部分全部传到 GPU 计算。

1) 只用 CPU 进行运算。

若运算均在 CPU 上执行,则运算时间只需考虑 CPU 片上运行时间,即:

$$Time_{CPU} = ExecTime_{CPU} \quad (2)$$

CPU 执行时间的计算公式如下:

$$ExecTime_{CPU} = \frac{TotalInstruction_{CPU}}{Performance_{CPU}} \quad (3)$$

Roofline 模型指出处理器的性能 *Performance* 受到内存带宽和计算能力顶点的限制。将设备的计算能力顶点 *ComputationRoof* 记作 *CR*, 如果设备有 N_{pe} 个运算核心,在流水线占满的情况下,每一时钟周期都会完成一次计算,则:

$$CR = f \times N_{pe} \quad (4)$$

运算密度 *AI* 可以通过操作指令数量 n_{ops} 和访存

指令数量 n_{mem} 的比值来估计。 n_{ops} 和 n_{mem} 与工作负载指令集以及处理器的指令集密切相关,将在特定指令集架构 ISA 下操作集的指令数 Π , 记作 $\Gamma(\Pi, ISA)$, 则:

$$AI = \frac{n_{ops}}{n_{mem}} = \frac{\Gamma(\Pi_{ops}, ISA)}{\Gamma(\Pi_{mem}, ISA)} \quad (5)$$

综上所述,处理器的性能计算公式如下:

$$Performance_{max} = \min(BW \times AI, CR) \quad (6)$$

令 $z = BW \times AI - CR$, 则:

$$sgn(z) = \frac{z}{|z|} = \begin{cases} 1, & BW \times AI > CR \\ 0, & BW \times AI = CR \\ -1, & BW \times AI < CR \end{cases} \quad (7)$$

$Performance_{max}$ 可表示为:

$$Performance_{max} = \frac{1}{2} [BW \times AI, CR] \cdot \left[\frac{sgn(z) + 1}{1 - sgn(z)} \right] \quad (8)$$

对于一个特定的工作负载,给定处理器对其数据的操作数总量 $TotalInstructions$ 是一定的,且满足 $TotalInstructions = \Gamma(\Pi, ISA)$, 那么当处理器性能达到最高时,片上运算时间最短。即:

$$ExecTime_{min} = \frac{TotalInstructions}{Performance_{max}} = \frac{\Gamma(\Pi, ISA)}{Performance_{max}} \quad (9)$$

综上所述,一个工作负载在 CPU 上运行的时间 $Time_{cpu}$ 的计算公式如下:

$$Time_{cpu} = \frac{2\Gamma(\Pi, ISA_{cpu})}{[BW_{cpu} \times \frac{\Gamma(\Pi_{ops}, ISA_{cpu})}{\Gamma(\Pi_{mem}, ISA_{cpu})} \times N_{pe}] \cdot \left[\frac{sgn(z) + 1}{1 - sgn(z)} \right]} \quad (10)$$

2) 运算密集部分全部传到 GPU 上执行。

GPU 的显示核心和显存都集成在显卡的板卡上,因而当运用 GPU 运算时,首先要将主存中的数据拷贝到显存中,这一阶段会通过相关总线传输。运算时间需要同时考虑 GPU 上执行时间以及运输时间。其计算公式如下:

$$Time_{GPU} = TransTime + ExecTime_{GPU} \quad (11)$$

假设从主存传到显存的速率为 BW_{CG} , 显存传回主存的速率为 BW_{GC} 。这 2 种传输一般是多个大块数据,通过 LogGP 模型对从内存到显存的传输时间 $TransTime_{CG}$, 以及从显存到内存的传输时间 $TransTime_{GC}$ 建模:

$$\begin{cases} TransTime_{CG} = L_{CG} + o_{CG} + \sum_i^{n_{CG}} (K_i - 1)G_{CG} + (n_{CG} - 1)g_{CG} \\ TransTime_{GC} = L_{GC} + o_{GC} + \sum_i^{n_{GC}} (K_i - 1)G_{GC} + (n_{GC} - 1)g_{GC} \end{cases} \quad (12)$$

传输开启延迟 L 、头字节额外开销 o 以及再次传输的等待时间 g 可以通过实验测得。这些参数对于一个计算平台来说是固定的,测得一次可以多次使用。

总线上传输数据的总时间 $TransTime$ 为:

$$TransTime = TransTime_{CG} + TransTime_{GC} \quad (13)$$

$$Overhead = L_{CG} + o_{CG} + L_{GC} + o_{GC} + g_{CG} - g_{GC} \quad (14)$$

且:

$$DataSize = \sum_i^{n_{CG}} k_i \quad (15)$$

结合式(12)~式(15)推导可得:

$$TransTime = G_{CG} \sum_i^{n_{CG}} k_i + G_{CG} DataSize + n_{CG} (g_{CG} - G_{CG}) + n_{GC} (g_{GC} - G_{GC}) + Overhead \quad (16)$$

GPU 片上执行时间的计算方式与 CPU 的一致,即:

$$ExecTime_{GPU} = \frac{TotalInstructions_{GPU}}{Performance_{GPU}} \quad (17)$$

本文模型考虑了单 CPU 和 GPU 的运算平台,传输模型较 LogGP 考虑的基于路由的模型而言,改善了 LogGP 的不足:1) 消息传递要么是 CPU 通过控制总线或者 DMA 方式由内存向 GPU 的显存传递大量数据,不存在数据重发的问题。2) 相比于路由模式,这种信号传递模式对多线程的上下文切换开销以及 Cache 未命中的情况敏感度不高。3) CPU 与内存、GPU 与显存组成的共享存储运算模型中,所考虑的运算任务是访存密集型或者运算密集型,通过 Roofline 模型综合了 2 个极限的影响。

1.3 运算功耗计算

功耗模型是根据硬件的参数和执行程序的特征来估计程序运行消耗的能量。根据精度分 3 个层次:基于热设计功耗和负载的模型,基于线程数量和缓存的模型以及基于架构的模型。前 2 种将运算平台的功耗又简单分为 2 种:没有进行运算时的基础功率 $IdlePower$ 和 CPU 或 GPU 完全占满时的最大功率 $BusyPower$ 。用 pwr 来表示运算平台的功率,则:

$$pwr(idle) = IdlePower \quad (18)$$

最大功率都可分为静态部分和动态部分,2 种模型对动态部分的计算方法不同,TDP 模型和线程数量和缓存模型分别为:

$$BusyPower_{TDP} = IdlePower + TDP/0.6 \quad (19)$$

$$BusyPower_{T\&C} = IdlePower + Performance \times EPI \quad (20)$$

综上所述,对于不同模型在不同状态下选用 $processor$ 为 CPU 或者 GPU 时的 $pwr(state, processor, model)$, 如表 1 所示。

表 1 功耗模型

功耗	TDP 模型	线程数量和缓存模型
<i>Idle</i>	<i>IdlePower</i>	<i>IdlePower</i>
<i>Busy</i>	<i>IdlePower</i> + <i>TDP</i> /0.6	<i>IdlePower</i> + <i>Performance</i> _{processor} × <i>EPI</i> _{processor}

对于一工作负载,如果在 CPU 上运行,那么消耗的能量为:

$$\begin{aligned}
 Energy_{CPU} &= ExecTime_{CPU} \times pwr(busy, CPU) \\
 &= [0, ExecTime_{CPU}] pwr(CPU)^T \quad (21)
 \end{aligned}$$

对于 GPU,除去执行指令的时间,还需要考虑传输数据时产生的开销。对于所有工作负载都在 GPU 上执行的情况:

$$\begin{aligned}
 Energy_{GPU} &= TranTime \times pwr(idle) + ExecTime_{GPU} \times pwr(busy, GPU) \\
 &= [TranTime, ExecTime_{GPU}] pwr(GPU)^T \quad (22)
 \end{aligned}$$

1.4 运算代价计算

结合运算时间模型和功耗模型,无论是用 CPU 处理或者用 GPU 处理,都用如下所示模型计算运算代价:

$$Cost = [TranTime, ExecTime] \cdot \begin{bmatrix} \omega_1 + \omega_2 pwr(idle) \\ \omega_1 + \omega_2 pwr(busy) \end{bmatrix} \quad (23)$$

运算代价模型需要的参数及其含义、以及参数的获得方式如表 2 所示。

表 2 运算代价模型参数

参数	含义	获得方式
<i>DataSize</i>	数据集大小	程序分析
<i>II</i>	循环中的指令	程序分析
$\Gamma(II, ISA)$	在特定指令架构 ISA 上的指令	程序分析
ω_1, ω_2	时间和功耗的权重	需求给定
<i>Overhead</i>	CPU-GPU 传输开销	测试程序
G_{CG}	主存向显存传输大块数据时候的间隔	测试程序
G_{GC}	显存向主存传输大块数据时候的间隔	测试程序
BW_{CPU}	主存带宽	测试程序
BW_{GPU}	显存带宽	测试程序
f_{CPU}	CPU 的频率	硬件参数
f_{GPU}	GPU 的频率	硬件参数
<i>IdlePower</i>	平台的基础功率	硬件测试
TDP_{CPU}	CPU 的频率热设计功耗	硬件参数
TDP_{GPU}	GPU 的频率热设计功耗	硬件参数
N_{CPU}	CPU 的运算模块数量	硬件参数
N_{GPU}	GPU 的运算模块数量	硬件参数

2 实验结果与分析

EPCC 测试集^[16]将程序分为 3 个等级,其中第 1

级由 13 个基础线性代数程序集中的程序组成,这部分是从另一个 Polybench/GPU 测试集中添加了 OpenACC 指令而得来的。本文使用 EPCC Level 1 集合中的 13 个程序作为样本,每个程序中都只有一个运算内核,并且每种程序集选取 3 种大小:0.5M, 2M, 10M,对运算代价模型的准确性进行测试,并分析误差产生的原因。

实验平台的硬件参数如表 3 所示。

表 3 实验硬件平台

硬件	平台 1	平台 2
CPU 型号	Core i7 4710MQ	Core i53340M
CPU 核心频率/GHz	2.5-3.5	2.7-3.4
CPU 运算核心数	8	4
CPU 访存带宽/(GB · s ⁻¹)	25.6	25.6
CPU 缓存/MB	6	3
GPU 型号	GTX 850M	GT750M
GPU 运算核心数	640	384
GPU 核心频率/GHz	0.86	0.95
GPU 访存带宽/(GB · s ⁻¹)	80	80
CPU-GPU 总线类型	PCI-E3.0	PCI-E 3.0

首先用运算代价模型评估分别在 CPU 和 GPU 上运行程序样本所产生的执行代价,然后比较两者,推荐运算代价较小的执行方式,之后在实验平台上进行实验,获得 CPU, GPU 的实际运算代价,并计算 CPU 与 GPU 的运算代价比值,判断先前的评估结论是否正确。验证结果如表 4 所示。

表 4 运算代价模型验证结果

程序	模型预测			实际 CPU 与 GPU 代价比			评价
	0.5M	2M	10M	0.5M	2M	10M	
2DCONV	•	•	•	1.04	1.55	1.68	×
2MM	•	•	•	3.96	7.95	14.42	√
3DCONV	•	•	•	0.63	0.90	1.44	×
3MM	•	•	•	23.37	31.72	46.02	√
ATAX	•	•	•	2.64	3.55	2.96	√
BICG	•	•	•	2.45	3.37	2.72	√
COR	◦	•	•	5.72	8.11	9.37	×
COV	•	•	•	5.67	8.01	8.99	√
GEMM	•	•	•	24.38	37.82	49.13	√
GESUMMV	•	•	•	0.88	1.35	1.33	×
MVT	◦	◦	•	2.93	3.49	3.06	×
SYR2K	•	•	•	6.31	9.65	14.12	√
SYRK	•	•	•	5.17	8.72	14.52	√

在表 4 中,用黑白点表示运算代价模型对程序执行方式的推荐结果。其中,黑点“•”表示推荐程序在 GPU 上执行;白点“◦”表示推荐程序在 CPU 上执行。对于实际执行代价,用实际 CPU/GPU 代价比来表示。CPU/GPU 代价比越大,说明程序在 CPU

上执行的代价越大于 GPU 上执行代价,即程序越适合在 GPU 上执行。最后,用评价来表示运算代价模型评估推荐结果和实际较优执行方式之间的一致性,即运算代价模型评估的正确性。若两者一致,则表示为√,反之则×。从表中可以看出,本文提出的运算代价评估模型具有较高的正确性。

实验程序中 2DCOV,3DCOV,GESUMMV 都是经运算代价模型分析认为应当使用 GPU 运算的,但因为生成的 GPU 程序内核过大,实际运行效率反而不如 CPU 的版本;对于 COR 程序,因为循环中包含了 sqrt 开方操作,模型错误地高估了程序的运行时间,当数据集较小时,可以对其进行并行化操作。具体的实际执行代价值如表 5 所示。

表 5 实际代价运行结果

程序	0.5M		2M		10M	
	CPU	GPU	CPU	GPU	CPU	GPU
2DCONV	0.39	0.37	1.46	0.94	7.28	4.34
2MM	8.44	2.13	73.63	9.26	1059.94	73.49
3DCONV	0.73	1.16	2.63	2.93	14.13	9.85
3MM	10.92	0.47	98.56	3.11	1095.16	23.80
ATAX	0.46	0.17	1.79	0.50	8.53	2.88
BICG	0.47	0.19	1.72	0.51	8.78	3.22
COR	9.73	1.70	91.94	11.34	1 069.43	114.11
COV	9.30	1.64	89.47	11.18	1 059.15	117.77
GEMM	13.89	0.57	135.61	3.59	1 599.25	32.55
GESUMMV	0.13	0.15	0.53	0.39	2.73	2.06
MVT	0.52	0.18	1.79	0.51	8.97	2.93
SYR2K	9.93	1.57	77.53	8.04	866.64	61.39
SYRK	9.60	1.86	75.06	8.60	851.47	58.63

3 结束语

本文结合 Roofline 模型和 LogGP 模型设计一个新的运算时间分析模型,并且分两层对运算耗能建模,最终结合两者建立一个运算代价模型。实验选取 EPCC 测试集对运算代价模型进行检验,结果表明,运算代价模型具有较高的准确性。本文设计的运算代价模型还存在可提升的空间,实验部分对功耗计算值给予了较低的权重,下一步将使用更精确的基于架构的功耗模型提高其准确性。

参考文献

- [1] 刘颖,吕方,王蕾,等.异构并行编程模型研究与进展[J].软件学报,2014,25(7):1459-1475.
- [2] 祝永志,王国仁,李丙锋,等.异构并行计算系统可扩展模型的实现[J].计算机工程,2009,35(17):97-99.
- [3] 王涛.基于 GPU 的程序分析与并行化研究[D].

郑州:解放军信息工程大学,2010.

- [4] 盛冲冲,胡新明,李佳佳,等.面向节点异构 GPU 集群的编程框架[J].计算机工程,2015,41(2):292-297.
- [5] GUZ Z, E. KEIDAR B I, KOLODNY A, et al. Many-core vs. Many-thread Machines; Stay Away from the Valley[J]. IEEE Computer Architecture Letters, 2009, 8(1):25-28.
- [6] EECKHOUT L. Computer Architecture Performance Evaluation Methods[M]. [S. l.]: Morgan & Claypool Publishers, 2010.
- [7] LIM J, LAKSHMINARAYANA N. Power Modeling for GPU Architecture Using McPAT[Z]. Argonne National Laboratory, 2013:1-17.
- [8] BAKHODA A, YUAN G L, FUNG W W L, et al. Analyzing CUDA Workloads Using a Detailed GPU Simulator[C]// Proceedings of ISPASS'09. Washington D. C., USA: IEEE Press, 2009:163-174.
- [9] NOWATZKI T, MENON J, HO C H, et al. Architectural Simulators Considered Harmful[J]. IEEE Micro, 2015, 35(6):4-12.
- [10] HONG S, KIM H. An Analytical Model for a GPU Architecture with Memory-level and Thread-level Parallelism Awareness[J]. ACM SIGARCH Computer Architecture News, 2009, 37(3):152.
- [11] GUZ Z, ITZHAK O, KEIDAR I, et al. Threads vs. Caches; Modeling the Behavior of Parallel Workloads [C]// Proceedings 2002 IEEE International Conference on Computer Design. Washington D. C., USA: IEEE Press, 2010:274-281.
- [12] CZECHOWSKI K, BATTAGLINO C, MCCLANAHAN C, et al. On the Communication Complexity of 3D FFTs and Its Implications for Exascale[C]// Proceedings of the 26th ACM International Conference on Supercomputing. New York, USA: ACM Press, 2012:216-222.
- [13] GÓMEZ-LUNA J, GONZÁLEZ-LINARES B J M. Performance Models for Asynchronous Data Transfers on Consumer Graphics Processing Units [J]. Journal of Parallel and Distributed Computing, 2012, 72(9):1117-1126.
- [14] VAN W B, MAASSEN J, SEINSTRAL F J, et al. Performance Models for CPU-GPU Data Transfers[C]// Proceedings of the 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. Washington D. C., USA: IEEE Press, 2014:11-20.
- [15] WILLIAMS S, WATERMAN A, PATTERSON D. Roofline: An Insightful Visual Performance Model for Multicore Architectures [J]. Communications of the ACM, 2009, 52(4):65-76.
- [16] GRILLO L, REYES R, SANDE F D. Performance Evaluation of OpenACC Compilers [C]// Proceedings of Euromicro International Conference on Parallel, Distributed, and Network-based Processing. Washington D. C., USA: IEEE Press, 2014:656-663.

编辑 索书志