# CPU/GPU协同并行计算研究综述

卢风顺 宋君强 银福康 张理论
(国防科学技术大学计算机学院 长沙 410073)

**摘 要** CPU/GPU异构混合并行系统以其强劲计算能力、高性价比和低能耗等特点成为新型高性能计算平台,但其复杂体系结构为并行计算研究提出了巨大挑战。CPU/GPU协同并行计算属于新兴研究领域,是一个开放的课题。根据所用计算资源的规模将CPU/GPU协同并行计算研究划分为三类,尔后从立项依据、研究内容和研究方法等方面重点介绍了几个混合计算项目,并指出了可进一步研究的方向,以期为领域科学家进行协同并行计算研究提供一定参考。

**关键词** 异构混合,协同并行计算,GPU计算,性能优化,可扩展
**中图法分类号** TP301 **文献标识码** A

## Survey of CPU/GPU Synergetic Parallel Computing

LU Feng-shun SONG Jun-qiang YIN Fu-kang ZHANG Li-lun
(College of Computer Science, National University of Defense Technology, Changsha 410073, China)

**Abstract** With the features of tremendous capability, high performance/price ratio and low power, the heterogeneous hybrid CPU/GPU parallel systems have become the new high performance computing platforms. However, the architecture complexity of the hybrid system poses many challenges on the parallel algorithms design on the infrastructure. According to the scale of computational resources involved in the synergetic parallel computing, we classified the recent researches into three categories, detailed the motivations, methodologies and applications of several projects, and discussed some on-going research issues in this direction in the end. We hope the domain experts can gain useful information about synergetic parallel computing from this work.

**Keywords** Heterogeneous hybrid, Synergetic parallel computing, GPU computing, Performance optimization

## 1 引言

当前,高性能计算机体系结构正处于变革期,各种新型体系结构不断涌现。采用通用多核微处理器与定制加速协处理器相结合的异构混合体系结构成为构造千万亿次计算机系统的一种可行途径。甚至有专家预言,今后的高性能计算平台将会成为以异构混合体系结构为主的格局。

在众多异构混合平台中,基于CPU/GPU异构协同的计算平台具有很大的发展潜力。正由于GPU所具有的强劲计算能力、高性能/价格比和高性能/能耗比,在当今追求绿色高性能计算的时代,GPU的计算优势受到越来越多的关注。除专业图形应用外,GPU已用于大量的通用计算问题,并形成了GPU通用计算研究领域,即GPGPU(General-purpose computing on graphics processing units),又称GP2U。鉴于GPU在通用计算领域的优异表现,Macedonia[1]断言GPU将成为未来计算的主流,甚至还有人将GPU的概念解释为General Computing Unit。GPU和CPU在设计思路上存在很大差异:CPU为优化串行代码而设计,将大量的晶体管作为控制和缓存等非计算功能,注重低延迟地快速实现某个操作;GPU则将大量的晶体管用作ALU计算单元,适合高计算强度(计算/访存比)的应用[2]。在协同并行计算时,CPU和GPU应各取所长,快速、高效协同地完成高性能计算任务。另外,除管理GPU计算任务外,CPU也应当承担一部分科学计算任务。以"天河一号"巨型机为例,其计算结点采用Intel Xeon E5540/E5450通用CPU和AMD ATI Radeon HD 4870x2加速GPU,计算阵列的峰值性能为214.96万亿次,加速阵列的峰值性能为942.08万亿次。如果不发挥CPU的计算能力,则相当于损失了一台200万亿次的高性能计算机。因此,需要充分挖掘CPU和GPU的计算潜能,使其达到高效协同的计算效果。

新型异构混合体系结构对大规模并行算法研究提出了新的挑战,迫切需要深入研究与该体系结构相适应的并行算法。针对CPU/GPU异构混合体系结构的高性能计算平台,研究相应的协同并行计算技术,设计并实现大型科学及工程计算问题的新型并行算法,具有重大的理论和实际意义。

## 2 CPU/GPU协同并行计算研究进展

自nVidia公司在1999年提出GPU概念以来[3],随着半

到稿日期:2010-04-13 返修日期:2010-07-15 本文受国家自然科学基金(40505023)资助。
卢风顺(1982—),男,博士生,CCF会员,主要研究方向为新型体系结构下的并行算法研究、大型数值模拟等高性能计算应用,E-mail:lufengshun@nudt.edu.cn;宋君强(1962—),男,研究员,博士生导师,主要研究方向为数值天气预报、高性能计算等。

导体技术的不断发展，芯片上集成的集体管数目不断增加，GPU 峰值性能一直以超过摩尔定律的速度增加，平均每 6 个月翻一番。GPU 具有浮点计算能力强、带宽高、性价比高、能耗低等优点，目前已被广泛用于图形处理以外的应用中，如数值天气预报[4]、地质勘探[5]、代数计算[6,7]、分子动力学模拟[8]、数据库操作[10]、频谱变换和滤波[11,12]等。特别是统一渲染架构发布以来，越来越多的科研人员（包括无任何图形 API 编程经验的科研人员）开始 GPU 非图形应用的研究，逐渐形成了新的 GPGPU 研究领域。

对于 GPGPU 领域的研究工作，文献[2,13-14]等优秀的综述已从 GPU 的发展历史、体系结构、编程模型、软件环境和成功案例等方面进行了系统阐述。本文仅从 CPU/GPU 协同并行计算的角度对国内外的研究工作进行回顾和分析。

### 2.1 协同计算概念及 GPGPU 研究分类

CPU/GPU 协同并行计算，其关键在于如何实现两者的高效“协同”。从国内外大量的研究工作来看，“协同”分为两个层次：1）CPU 仅负责管理 GPU 的工作，为 GPU 提供数据并接收 GPU 传回的数据，由 GPU 承担整个计算任务；2）除管理 GPU 外，CPU 还负责一部分计算任务，与 GPU 共同完成计算。第一层次的“协同”比较简单，CPU 与 GPU 间分工明确，但浪费了宝贵的 CPU 计算资源。如 IBM 的 Power7 处理器具有 32 个核心，因此 CPU 也具有强大的计算能力。可见，第二层次的“协同”是未来协同并行计算的发展方向。

目前，“协同并行计算”还没有统一的定义。鉴于许多文献[4,8,15-16]将 GPU 视为加速部件或协处理器，刘钦等[5]将其定义为 CPPC（co-processing parallel computing）。在异构混合并行系统中，CPU 和 GPU 都是并行计算资源，只是体系结构及计算方式不同，无须区分两者的主从关系。因此，本文将“协同并行计算”定义为“synergetic parallel computing”。

CPU/GPU 异构机群的组织形式通常是将一定数量的多核 CPU 和 GPU 封装到一个节点内（如图 1 所示），继而由若干节点互联成异构机群。将 CPU/GPU 异构机群的计算资源简单抽象为三元组 PG=[NC,CC,GC]，各分量分别表示节点数目、每个节点的 CPU 数目和 GPU 数目；如果该机群存在节点间异构特性，那么 PG 定义为[NC,CC[NC],GC[NC]]，各分量分别表示节点数目、相应节点的 CPU 数目和 GPU 数目。根据参与协同计算的 CPU 和 GPU 硬件资源规模，目前国内外 GPGPU 研究可以划分为三类：GPU 通用计算研究（PG=[1,0,$G$>0]）、CPU/GPU 协同计算研究（PG=[1,$C$>0,$C$>0]）以及 CPU/GPU 协同并行计算研究（PG=[$N$>1,$C$>0,$G$>0]）。

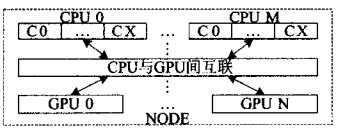


图 1 节点内 CPU/GPU 组织形式

### 2.2 GPU 通用计算研究

GPU 通用计算研究（PG=[1,0,$G$>0]），其计算规模限于单个节点，计算任务完全由 GPU 承担，因此该“协同”属于第一层次。由于 GPU 硬件以及软件开发环境的限制，早期的 GPGPU 研究[7,17]必须紧密结合 GPU 硬件细节并借助图形 API 来实现 GPGPU 程序。随着统一架构 GPU 及 CUDA[18]和 Brook+[19]等编程模型的出现，越来越多的领域专家利用 GPU 加速其具体应用问题。Liu 等[20]基于 CUDA 研究分子动力学模拟问题；Cevahir 等[21]利用 GPU 求解稀疏对称线性系统，并实现了混合精度的多 GPU 共轭梯度求解器；Chen 等[22]基于 CUDA 实现了可支持更多的元素类型的快速排序算法，并针对 GPU 体系结构进行了性能优化；Igual 等[22]基于 GPU 研究了线性代数和图像处理问题，并总结出数值算法在 GPU 上获得高性能应具备的特点。

### 2.3 CPU/GPU 协同计算研究

CPU/GPU 协同计算研究（PG=[1,$C$>0,$C$>0]），其计算任务由 CPU 和 GPU 两者共同完成，属于第二层次的“协同”，但是协同计算规模较小，仅限于单节点。方旭东[24]基于 CPU/GPU 异构平台，研究了矩阵乘、LU 分解和 Mygrid 等科学计算程序的新型并行算法，提出了 3 种任务划分模型及 4 种 GPGPU 程序优化策略；在单节点规模下，CPU/GPU 协同并行版本的矩阵乘、LU 分解和 Mygrid 等分别获得了 CPU 版本 133.8 倍、88.86 倍和 15.06 倍的性能加速。刘钦等[5]实现了非对称走时 Kirchhoff 叠前时间偏移的 CPU/GPU 协同并行计算，使其运算速度提高到单核 CPU 版本的 100～300 倍，且基于此成果开发出油气勘探地震处理 CPU/GPU 协同并行计算商业系统。

### 2.4 CPU/GPU 协同并行计算研究

CPU/GPU 协同并行计算研究（PG=[$N$>1,$C$>0,$G$>0]），涉及到 $N$ 个节点，每个节点的 $C$ 颗 CPU 和 $G$ 颗 GPU 协同完成计算任务。鉴于大型科学及工程计算问题对高性能计算资源的巨大需求，针对这些问题的 GPGPU 研究基本属于该类，这也是本文关注的重点。目前，国内外已有大量的研究机构开展多核 CPU/GPU 异构混合平台的协同并行计算研究，此处仅列举几个影响较大的项目，重点介绍其立项依据、研究内容和研究方法等。

#### 2.4.1 WRFGPU

数值天气预报的发展与高性能计算机及计算技术的不断进步密切相关，当前气象科学家通常使用包含大量处理器的机群系统进行超大规模数值模拟。但时效性要求较高的应用（如实时预报或者气候预报等）应具有较好的强可扩展性，因此需要更快而不仅是更大规模的高性能计算平台。为此，WRF 模式团队启动了 WRFGPU 项目[25]，基于新的多核 CPU 和 GPU 异构混合平台开发 WRF 模式中的细粒度并行性，提高 WRF 模式的强可扩展性。其目标包括：1）确定 WRF 模式物理和动力过程的关键内核，针对各内核的计算访存比、数据并行性、内存使用等建立数学模型；2）实现内核基准程序集，用来评测当前及未来异构高性能计算平台在天气和气候研究中的加速效果。目前的内核基准程序集包含 5 个内核：WSM5 云微过程[4]、WFOPD 标量平流[26]、WCK 化学动力求解器[27]、RRTM 长波辐射物理过程和 SWRAD 短波辐射物理过程，其中前 3 个内核已发布 CUDA 版本。

WSM5 云微过程模块的代码量仅占 WRF 模式的 0.4%，但运行时间占串行总时间的 1/4。Michalakes 等[4]研究了该计算密集模块的 CPU/GPU 协同并行计算技术，采用 MPI/CUDA 并行编程模式，每个 MPI 进程绑定一个 CPU-GPU 组合。作者基于 Illinois 大学 NCSA 机群进行了大量的数值试验，结果表明该模块性能可提高 9.4 倍，而 WRF 模式的整体

性能可提高 1.23 倍。标量平流[26]模式模拟大气标量要素场在风场驱动下的输送过程，其计算量与标量的数目成正比，如 WRF 常规运行下 5 个雾标量的计算时间占串行总时间的 10%。标量平流具有计算访存比低(0.76)、线程间数据依赖关系强和 CPU-GPU 数据传输量大等特点，为其协同并行计算提出了巨大挑战。作者采用如下措施对该内核 GPU 版本进行性能优化：1)借助三维硬件纹理(CUDA2.0 及以上版本支持)减少了计算内核数量和 CPU-GPU 数据传输次数；2)在主机端使用页锁定内存，提高了 CPU 与 GPU 间的数据传输速率，性能提高了 1.25 倍。WRF-Chem 模式支持许多化学动力求解器，文献[27]重点研究了 RADM2 模型的异构混合并行技术。在 WRF-Chem 模式中，RADM2 求解器作用于固定区域网格内的每个格点，即该模式包含大量的数据并行性；同时在 RADM2 求解器内部，线性代数操作中存在一定的指令级并行。因此，作者提出了三层并行处理技术，即 CPU 核心内的指令级并行、GPU 流处理单元上的数据并行以及多核心(多线程)或者节点间(MPI)的数据并行。

2.4.2 Folding@Home

蛋白质是生命体系中重要的功能物质，被称为生物机体的“纳米计算机”[28]。蛋白质分子由氨基酸残基组成，通过折叠成特定形状来体现其功能(如酶和抗体等)；如果折叠过程出错，则会导致癌症、疯牛病、帕金森氏症、阿兹海默症等疾病。

Folding@Home[28]是分布式分子动力学项目，通过个人和组织捐献的家庭及办公计算资源来研究蛋白质的折叠行为，目前其 GPU 版本已得到广泛部署[29]。据统计[1)]，目前 GPU 贡献的计算能力达到 3214 TFLOPS，占总计算资源的 56.3%，已超过 CPU 和 PS3 提供的计算资源的总和。

蛋白质折叠模拟可抽象为 N 体模拟问题，目前已有基于“成对相加”思想的 $O(N^2)$ 力学模型以及基于邻接表、树等数据结构的 $O(N)$ 力学模型。由于蛋白质折叠模拟涉及大量的粒子($10^3-10^6$)及积分步($10^6-10^{15}$)，因此巨大的计算需求限制了问题规模和模拟时间，最终限制了所获得的有用信息量。鉴于 GPU 与 CPU 间巨大的浮点计算性能差异，目前已有相关研究利用 GPU 来完成基于 $O(N^2)$ 模型的蛋白质折叠模拟。Elsen 等[29]利用 GPU 加速几种通用的力学模型，使其性能超过高度优化 CPU 版本的 25 倍以上，并指出 $N$ 体模拟为计算受限问题，随着 GPU 峰值性能的不断提高，其问题规模及时间尺度必定会不断增大。Friedrichs 等[31]在 GPU 上完整实现了全原子蛋白质分子动力学模拟程序，包括所有的标准力场项、积分、约束等。作者首先讨论了 GPU 版本实现所面临的算法可扩展性、访存、CPU 与 GPU 间通讯、流程控制等挑战，然后分别介绍了 ATI 和 NVIDIA 版本程序的实现细节，最后基于 ATI Radeon HD 4870 和 NVIDIA GeForce GTX 280 GPU 对该模拟程序进行数值实验，获得比传统 CPU 单核版本性能快 700 倍的超高性能。

2.4.3 MAGMA

从 Intel，AMD，IBM 和 NVIDIA 等工业界主流芯片厂商最新发布的处理器来看，异构混合体系结构将成为未来处理器以及高性能计算机系统的发展方向。目前多核 CPU 技术发展迅速，其核心数目不断增加，如 IBM 最新的超级计算机 Blue Waters 使用的 Power7 处理器具有 32 个核心，而 Intel 集成度最高的单硅 CPU 原型包括 48 个可编程 IA 处理器内核；同时，GPU 具有非常高的浮点计算性能，且 GPGPU 应用开发已具有友好的编程环境。面对混合计算环境提出的复杂挑战，最优的软件解决方案是将不同算法的优势集中到一个软件框架内，即软件本身也是混合的。基于此思想，MAGMA 项目[32]针对多核 CPU/GPU 异构混合平台，开发类似 LAPACK 的稠密线性代数(DLA)库和软件框架，从而使应用程序充分利用混合系统内各种处理器提供的计算资源。

无论针对多核 CPU 还是 GPU 平台，高效 DLA 算法的设计要求都是统一的，即算法应该具备并行度高和计算强度高等特点。对于 CPU/GPU 异构混合平台，DLA 算法的设计须同时考虑执行过程中的负载平衡问题，且计算任务的划分应密切结合各平台的优势。Tomov 等[33]利用有向无环图(DAG)来开发 DLA 算法的并行度，将算法的执行过程表示为一系列的子任务及其相互依赖关系，其中“结点”表示子任务，“边”表示子任务间的依赖关系。各子任务的粒度根据 CPU 和 GPU 的计算能力进行划分，其中大任务由 GPU 负责执行，而小任务在多核 CPU 上执行。为提高 DLA 算法的计算强度，作者修改了基于 BLAS1 库的 DLA 算法，在其最内层循环采用块矩阵操作。Tomov 等[34]提出基于“混合技术”的算法设计思想，以充分利用异构平台各处理器类型的计算优势。作者利用 CUDA 编程模型，基于 BLAS 和 LAPACK(多核 CPU)及 CUBLAS(GPU)等第三方库，设计了 Cholesky，LU 和 QR 分解等混合 DLA 算法。关于异构平台上可扩展 DLA 算法的设计，Ltaief 等[35]充分利用 DLA 算法的两级并行度，首先将其计算任务划分为块，映射到多个 CPU-GPU 组合并发执行，然后对每个块内的计算任务继续开发细粒度并行性以使 CPU 和 GPU 协同计算。数值实验表明，以 CPU-GPU 组合为计算资源单位，当计算资源线性增加时，基于 DLA 算法求解器的浮点性能也呈线性增长趋势，表现出较好的强可扩展性。

2.4.4 FEASTGPU

FEAST[36,37]是解决大规模有限元问题的高效软件包，可支持多种现代体系结构上的软件开发，其应用领域覆盖计算流体力学和计算结构力学，主要构件包括稀疏带状 BLAS、可扩展递归聚类(ScaRC)以及 FEASTGPU。稀疏带状 BLAS 是 BLAS 的一种扩展，封装了 cache 感知和平台优化的通用线性代数操作例程。ScaRC 是一种广义的求解模式，融合了区域分解和并行多重网格的优势，可提供层次式的求解器、数据和矩阵结构。FEASTGPU[15]作为局部平滑子完全工作在 ScaRC 模式内部，因此基于 FEAST 的应用程序可直接利用 GPU 提供的强大计算能力和超高内存带宽，无须对代码作任何修改(即“最小扰动”)。FEASTGPU 运行在单精度状态，且采用混合精度迭代求精法以保证迭代过程的收敛速度。

FEASTGPU 在整个 FEAST 软件包中的抽象层次及工作方式，使得混合系统的异构性完全封装到节点内，因此对 MPI 而言该并行系统是同构的。FEASTGPU 并非针对线性代数操作进行加速，而是针对局部子问题加速整个多重网格

1) 客户端统计信息：2010 年 4 月 4 日 05:01:35 更新。

求解器，避免了 GPU 计算内核的多次配置及数据传输开销。Goddeke 等[15,16,38]基于 FEAST 构建了 FEASTSolid 和 Navier-Stokes 求解器，并使用 FEASTGPU 来加速其本地求解器，分别取得了 5 倍和 12 倍的局部加速及 1.6 倍和 2.3 倍的全局加速。根据 Amdahl 定律，如果可加速部分的计算时间小于整体的 50%，那么 FEASTGPU 局部加速引起的全局性能提升将非常有限。为此，作者指出两种解决方案：1）修改原有算法，使可加速部分的比例增大；2）松弛“最小扰动”条件，将更多的计算过程迁移到 GPU 上运行。

2.4.5 其它研究项目

除上述几个代表性的项目外，国内外还有大量的相关研究，如 Zhe 等[40]较早开展了 GPGPU 研究，用格子 Boltzmann 模型（LBM）实现了并行流模拟程序；东京工业大学基于 TSUBAME 异构混合机群开展了加速计算（Accelerated Computing）研究[41-44]；国防科学技术大学成功研制出“天河一号”，开展了线性代数[45]、粒子模拟[46]、GPGPU 程序性能优化[47,48]等相关研究；中国科学院过程工程研究所及联想、曙光公司共同设计并研制出千万亿单精度峰值性能的 Mole-8.7 系统[49]，该系统主要应用于多相流、分子动力学等研究领域。

## 3 进一步研究的方向

### 3.1 面向异构混合系统的新型并行算法研究

在 CPU/GPU 异构混合平台中，CPU 和 GPU 具有不同的硬件特点和计算方式，因此基于异构混合平台进行并行算法设计时，必须密切结合其底层硬件特点，使算法充分利用混合系统中各类型处理器的性能优势。鉴于 GPGPU 研究属于新兴领域，目前大部分算法研究工作是已有算法向异构混合平台的移植，针对该平台的全新算法较少。

CPU 和 GPU 都存在存储墙[50]问题，CPU 主要通过多层次存储结构来缓解该问题，而 GPU 则使用硬件多线程技术来隐藏高开销的访存延迟。面向异构混合系统的高效并行算法应具有以下特点：1）异构感知的：根据底层硬件特点设计算法，使体系结构—算法组合发挥出最大性能；2）计算强度高：高计算强度是并行程序高计算效率的普遍要求，对 GPU 尤其重要，否则 GPU 的高浮点计算性能优势根本得不到发挥；3）CPU 与 GPU 交互开销小：包括数据传输开销及同步开销；CPU 与 GPU 间交互是协同并行计算不可避免的，应通过优化算法来减少数据传输次数和数据量以及同步开销。

### 3.2 CPU/GPU 高效协同方式研究

CPU/GPU 高效协同计算是发挥异构混合平台性能的关键因素，因此必须根据两者的计算能力和执行特点确定合理的协同方式，以保证 CPU 和 GPU 间的计算负载平衡，降低各种交互开销，进而提高程序的执行效率。研究内容包括：计算任务的划分模型、任务的调度策略以及计算任务与硬件资源的映射关系。

合理的任务划分是高效协同的基础，需综合考虑各计算资源的计算能力、计算量、通信和数据传输开销等多个因素，将整个计算任务划分成一定数量和适当粒度的子任务。任务调度将各计算子任务调度到空闲计算资源上执行，使整个异构混合系统时刻保持满负载状态，从而发挥其最大计算能力；任务调度通常由并行编程环境或资源管理系统负责，但有时领域科学家需要自己设计调度策略，如将计算任务用 DAG 图表示，“结点”表示子任务，“边”表示各子任务的后继关系。计算任务与硬件资源的映射关系决定了该计算任务的执行效率：以 GPU 为例，由于其缓存及寄存器资源非常有限，因此不同的线程组织方式对性能差异较大。

### 3.3 GPGPU 程序的性能优化技术研究

对 CPU/GPU 协同并行计算程序而言，其性能影响因素有：计算内核组织方式、线程组织方式、寄存器和缓存的使用、全局存储器访问特点、GPU-CPU 同步以及 GPU-CPU 数据传输等。因此，GPGPU 程序性能优化研究内容包括：1）研究并行程序的计算特点，设计合适的计算内核和数据流以及线程组织方式；2）研究并行程序的访存特点，有效使用 GPU 上的层次式内存，提高其计算强度；3）研究 CPU-GPU 协同方式，借助 GPU 编程环境提供的异步操作来减少各种交互开销。

为方便领域科学家开发高效的 CPU/GPU 协同并行计算程序，下面列举几条指导性建议：1）充分利用 GPU 零开销的进程切换特点，确保足够的细粒度并行性以隐藏访存延迟等开销；2）借助共享存储器实现高效的块内线程间通讯，尽量避免使用高延迟的全局同步操作（全局存储器读写）；3）研究程序的数据局部性和内存访问模式，充分利用 GPU 的内存层次，尤其是高速缓存的使用；4）设计合理的数据结构，充分利用 GPU 硬件纹理，加速大块、只读数据的访问；5）有时可根据 GPU 硬件厂商提供的指令集，研究硬件感知的优化方法，从 ALU 使用、预取带宽和线程使用三个方面优化程序。

### 3.4 异构平台上大规模并行程序的可扩展性研究

可扩展性是设计高性能计算机和并行算法所追求的一个重要目标。可扩展性主要包括体系结构可扩展性、并行机可扩展性[51]、并行算法可扩展性[52]、并行算法—机器组合可扩展性[53]和并行算法—体系结构可扩展性[54]等。新型 CPU/GPU 异构混合平台的出现为高性能计算提供了海量计算资源，同时其复杂体系结构为该平台上的可扩展并行算法设计提出了新的挑战，主要来源于平台的异构性、海量并行度、GPU 计算能力与 CPU-GPU 数据传输速度间的差距等。

目前可扩展性研究成果主要集中于同构系统，对 CPU 异构系统（CPU 峰值性能不同）的可扩展性也有少量成果发表[30,39]，因此非常有必要研究并行算法—CPU/GPU 体系结构组合的可扩展性，内容包括：1）研究 CPU/GPU 并行机体系结构特点，找到影响该组合可扩展性的因素，并提取到可扩展模型中；2）研究并降低并行算法对该组合可扩展性的负面影响；3）提出并行算法—CPU/GPU 体系结构组合的可扩展模型，为异构平台上的大规模并行程序开发提供指导意义。

**结束语** 30 多年来，随着半导体和制造工艺的飞速发展以及处理器体系结构的不断演化，单核处理器的性能一直以摩尔定律的速度不断提高；对并行程序而言，即使不进行任何修改，其性能也可随着处理器性能的提高而增长，有人将这种现象称为“免费的午餐”（free ride）[2]。但由于制造工艺、物理极限和功耗极限等因素的制约，单核处理器的发展遇到瓶颈，继而转向多核方向，以求在设计复杂度、性能以及功耗等诸多方面达到最佳平衡；同时 GPU 等处理器以其高浮点性能也受到高性能计算业界的广泛重视。为充分利用这些新兴处理器平台的性能，必须密切结合其体系结构特点开展新型并行

算法研究，这项工作具有很大的挑战性。

CPU/GPU 协同并行计算是近几年新兴的前沿课题，目前已有许多领域科学家通过协同并行计算加速相关应用，且已取得不少成果。本文综述了 CPU/GPU 协同并行计算的研究进展，并重点介绍了几个项目的立项依据、研究内容和研究方法，以期为领域科学家进行协同并行计算提供有用信息。CPU/GPU 协同并行计算研究是开放的课题，在新型并行算法研究、高效协同方式、程序性能优化和大规模可扩展算法研究等方面都值得进一步研究。

## 参 考 文 献


[1] Macedonia M. The GPU enters computing's mainstream [J]. IEEE Computer, 2003, 36(10): 106-108

[2] Owens J D, Houston M, Luebke D, et al. GPU computing [J]. Proceedings of the IEEE, 2008, 96(5): 879-899

[3] 张舒，褚艳利，等. GPU 高性能运算之 CUDA[M]. 北京：中国水利水电出版社，2009：1-13

[4] Michalakes J, Vachharajani M. GPU acceleration of numerical weather prediction [J]. Parallel Processing Letters, 2008, 18(4): 531-548

[5] 刘钦，佟小龙. GPU/CPU 协同并行计算(CPPC)在地震勘探资料处理中的应用[R]. 北京：北京吉星吉达公司，2008

[6] Bell N, Garland M. Implementing Sparse Matrix-Vector Multiplication on Throughput-oriented Processors[C]// SC2009. New York: ACM, 2009

[7] Bolz J, Farmer I, Grinspun E, et al. Sparse matrix solvers on the GPU: Conjugate gradients and multigrid [J]. ACM Transaction on Graphics, 2003, 22(3): 917-924

[8] Stone J, Phillips J, Hardy D, et al. Accelerating molecular modeling applications with graphics processors [J]. Journal of Computational Chemistry, 2007, 28(16): 2618-2640

[9] Anderson J A, Lorenz C D, Travesset A. General purpose molecular dynamics simulations fully implemented on graphics processing units [J]. Journal of Chemical Physics, 2008, 227(10): 5342-5359

[10] Govindaraju N K, Lloyd B, Wang W, et al. Fast computation of database operations using graphics processors [C] // SIGMOD 2004. New York: ACM, 2004

[11] Nukada A, Ogata Y, Endo T, et al. Bandwidth intensive 3-D FFT kernel for GPUs using CUDA[C]// SC2008. New York: ACM, 2008

[12] Govindaraju N K, Lloyd B, Dotsenko Y, et al. High Performance Discrete Fourier Transforms on Graphics Processors [C] // SC2008. New York: ACM, 2008

[13] 吴恩华. 图形处理器用于通用计算的技术、现状及其挑战[J]. 软件学报，2004，15(10)：1493-1504

[14] Blythe D. Rise of the Graphics Processor[J]. Proceedings of the IEEE, 2008, 96(5): 761-778

[15] Goddeke D, Wobker H, Strzodka R, et al. Co-processor acceleration of an unmodified parallel solid mechanics code with FEASTGPU[J]. International Journal of Computational Science and Engineering, 2009, 4(4): 254-269

[16] Goddeke D, Buijssen S H M, Wobker H, et al. GPU Acceleration of an Unmodified Parallel Finite Element Navier-Stokes Solver [C]// High Performance Computing & Simulation 2009. Logos Verlag: IEEE, 2009

[17] Larsen E S, McAllister D. Fast matrix multiplies using graphics hardware[C]// SC2001. New York: the ACM Press, 2001

[18] NVIDIA Corporation. CUDA Programming Guide Version 2.2 [EB/OL]. http://developer.download.nvidia.com/compute/cuda/2_21/toolkit/docs/NVIDIA_CUDA_Programming_Guide_2.2.1.pdf, 2009-08-12

[19] Devices A M. ATI Steam Computing User Guide[EB/OL]. http://developer.amd.com /gpu_assets/Stream_Computing_User_Guide.pdf, 2010-03-25

[20] Liu W, Schmidt B, Voss G, et al. Accelerating molecular dynamics simulations using graphics psrocessing units with CUDA[J]. Computer Physics Communications, 2008, 179(9): 634-641

[21] Cevahir A, Nukada A, Matsuoka S. Fast Conjugate Gradients with Multiple GPUs[C]// Allen G, et al., eds. ICCS 2009. Part I. LNCS 5544, 2009: 893-903

[22] Chen S, Qin J, Xie Y. A Fast and Flexible Sorting Algorithm with CUDA[C]// Hua A, Chang S-L, eds. ICA3PP 2009. LNCS 5574. 2009: 281-290

[23] Igual F D, Mayo R, Quintana-orti E S. Attaining High Performance in General-purpose Computations on Current Graphics Processors[C]// Palma J M L M, et al., eds. VECPAR 2008. LNCS 5336. 2008: 406-419

[24] 方旭东. 面向大规模科学计算的 CPU-GPU 异构并行技术研究[D]. 长沙：国防科学技术大学，2009

[25] Michalakes J, Vachharajani M. GPU Acceleration of NWP: Benchmark Kernels [EB/OL]. http://www.mmm.ucar.edu/wrf/WG2/GPU, 2009-02-25

[26] Michalakes J, Vachharajani M. GPU Acceleration of Scalar Advection [EB/OL]. http://www.mmm.ucar.edu/wrf/WG2/GPU/Scalar_Advect.htm, 2009-02-25

[27] Linford J, Michalakes J, Sandu A, et al. Multi-core acceleration of chemical kinetics for simulation and prediction[C]// SC2009. Portland, Oregon, the IEEE Press, 2009

[28] Pande lab. Folding@Home[EB/OL]. http://folding.stanford.edu, 2010-03-18

[29] Elsen E, Vishal V, Houston M, et al. N-body simulations on GPUs[C]// SC 2006. New York: ACM, 2006

[30] Chen Yong, Sun Xian-he, Wu Ming. Algorithm-system scalability of heterogeneous computing [J]. Journal of Parallel and Distributed Computing, 2008, 68: 1403-1412

[31] Friedrichs M S, Eastman P, Vaidyanathan V, et al. Accelerating Molecular Dynamic Simulation on Graphics Processing Units [J]. Journal of Computational Chemistry, 2009, 30(6): 864-872

[32] Agullo M, Demmel J, Dongarra J, et al. Numerical linear algebra on emerging architectures: the PLASMA and MAGMA projects [J]. Journal of Physics: Conference Series, 2009, 180(1)

[33] Tomov S, Dongarra J, Baboulin M. Towards Dense Linear Algebra for Hybrid GPU Accelerated Manycore Systems[R]. Tennessee: University of Tennessee Computer Science, 2008

[34] Tomov S, Nath R, Ltaief H, et al. Dense Linear Algebra Solvers for Multicore with GPU Accelerators[C]// High-level Parallel Programming Models and Supportive Environments 2010. Atlanta: IEEE, 2010

[35] Ltaief H, Tomov S, Nath R, et al. A Scalable High Performant Cholesky Factorization for Multicore with GPU Accelerators [R]. Innovative Computing Laboratory, 2009

[36] FEAST Group. FEAST: Finite Element Analysis & Solutions Tools[EB/OL]. http://www.feast.uni-dortmund.de/index.html, 2010-4-07


(下转第 46 页)

份的$(t,n)$动态门限代理签名方案。相对于一般的基于身份的门限代理签名方案，本文方案具有更高的动态属性。同时，利用CDH问题的困难性证明了本文方案在标准模型下的安全性，因此相对于随机预言模型下可证安全的方案来说，本文方案具有更高的安全性。

## 参考文献


[1] Mambo M, Usuda K, Okamoto E. Proxy signature for delegating signing operation [C]// Proceedings of the 3rd ACM Conference on Computer and Communications Security. New York: ACM, 1996: 48-57

[2] Zhang K. Threshold proxy signature schemes[C]// Proceedings of Information Security Workshop (ISW97). LNCS 1396, Springer-Verlag, 1997: 282-290

[3] Kim S, Park S, Won D. Proxy Signatures, Revisited [C]// Proceedingsof Information and Communications Security (ICICS 97). LNCS 1334. Springer-Verlag, 1997: 223-232

[4] Shamir A. How to share a secret[J]. Communications of the ACM, 1979, 22(11): 612-613

[5] Sun H M. An efficient nonrepudiable threshold proxy signature scheme with known signers[J]. Computer Communicaion, 1997, 22(8): 717-722

[6] 李继国，曹珍富. 一个改进的门限代理签名方案[J]. 计算机研究与发展，2002，39(11)：1513-1518

[7] Hwang M S, Lu J L, Lin L C. A practical(t,n) threshold proxy signature scheme based on the RSA cryptosystem[J]. IEEE Trans. on Knowledge and Data Engineering, 2003, 15(6): 1552-1560

[8] 蒋瀚，徐秋亮，周永彬. 基于RSA密码体制的门限代理签名[J]. 计算机学报，2007，30(2)：241-247

[9] Shamir A. Identity-based cryptosystems and signature schemes [C]// Blakley G, Chaum D, eds. Proceedings of Crypto 1984. New York: Springer-Verlag, 1984: 47-53

[10] Boneh D, Franklin M. Identity-based encryption from the Weil pairing[C]// Kilian J, ed. Proceedings of Crypto 2001. London: Springer-Verlag, 2001: 213-229

[11] Xu J, Zhang Z F, Feng D G. Identity Based Threshold Proxy Signature[EB/OL]. http://eprint.iacr.org/2004/250/

[12] Bao H Y, Cao Z F, Wang S B. Identity-based Threshold Proxy Signature Scheme with Known Signers [C]// Proceedings of Theory and Applications of Models of Computation. LNCS 3959. Springer-Verlag, 2006: 538-546

[13] 鲁荣波，何大可，王常吉. 对一种基于身份的已知签名人的门限代理签名方案的分析[J]. 电子与信息学报，2008，30(1)：100-103

[14] Paterson K G, Schuldt J C N. Efficient identity-based signatures secure in the standard model[C]// Proceedings of ACISP 2006. Berlin: Springer-Verlag, 2006: 207-222


---

(上接第9页)


[37] Becker C, Buijssen S H M, Wobker H, et al. FEAST: Development of HPC technologies for FEM applications[C]// Münster G, Wolf D, Kremer M, eds. High Performance Computing in Science and Engineering. Berlin: Springer, 2008

[38] Goddeke D, Strzodka R, Mohd-Yusof J, et al. Exploring weak scalability for FEM calculations on a GPU-enhanced cluster[J]. Parallel Computing, 2007(33): 685-699

[39] Pastor L, Orero J L B. An Efficiency and Scalability Model for Heterogeneous Clusters[C]// Proceedings of the 2001 IEEE International Conference on Cluster Computing. Newport Beach: IEEE, 2001

[40] Zhe F, Feng Q, Kaufman A, et al. GPU cluster for high performance computing[C]// SC2004. Washington: IEEE, 2004

[41] Ogawa S, Aoki T. GPU computing for 2-dimensional incompressible-flow simulation based on multigrid method [C]// Transactions of the Japan Society for Computational Engineering and Science. 2009: 20090021

[42] Nukada A, Matsuoka S. Auto-tuning 3-D FFT Library for CUDA GPUs[C]// SC2009. Portland: ACM, 2009

[43] Matsuoka S. Petascaling Commodity onto Exascale: GPUs as Multithreaded Massively-parallel Vector Processors-the Only Road to Exascale [C]// IEEE Cluster Computing Conference 2009. New Orleans: IEEE, 2009

[44] Matsuoka S, Aoki T, Endo T, et al. GPU accelerated computing-from hype to mainstream, the rebirth of vector computing[J]. Journal of Physics: Conference Series, 2009, 180(1): 012043

[45] 葛震. GPU加速PQMRCGSTAB算法研究[D]. 长沙：国防科学技术大学，2009

[46] 吴强. GPU加速高速粒子碰撞模拟[D]. 长沙：国防科学技术大学，2009

[47] Fang Xu-dong, Tang Yu-hua, Wang Gui-bin, et al. Optimizing stencil application on multi-thread GPU architecture using stream programming model[C]// Muller-Schloer C, Karl W, Yehia S, eds. ARCS. LNCS 5974. 2010: 234-245

[48] Ma An-guo, Cai Jing, Cheng Yu, et al. Performance Optimization Strategies of High Performance Computing on GPU[C]// Dou Y, Gruber R, Joller J, eds. APPT. LNCS 5737. 2009: 150-164

[49] Chen Fei-guo, Ge Wei, Guo Li, et al. Multi-scale HPC system for multi-scale discrete simulation—Development and application of a supercomputer with 1 Petaflops peak performance in single precision[J]. Particuology, 2009, 7: 332-335

[50] Asanovic K, Bodik R, Catanzaro B, et al. The Landscape of Parallel Computing Research: A View from Berkeley[R]. California: Electrical Engineering and Computer Sciences University of California at Berkeley, 2006

[51] Bell G. Ultracomputers: a teraflop before its time[J]. Communication of the ACM, 1992, 35(8): 26-47

[52] Gupta A, Kumar V. Scalability of Parallel Algorithms for Matrix Multiplication[C]// 1993 International Conference on Parallel Processing. New York: IEEE, 1993: 115-123

[53] Sun Xian-he, Rover D T. Scalability of Parallel Algorithm- Machine Combinations[J]. IEEE Transactions on Parallel and Distributed Systems, 1994, 5(6): 599-613

[54] Kumar V, Gupta A. Analysis of scalability of parallel algorithms and architectures: A survey[C]// International Conference on Supercomputing. Cologne: ACM, 1991: 396-405