面向大数据高通量计算的 CPU/GPU 并行优化技术研究

扶聪

2018 年 3 月

中图分类号 TQ028.1

UDC分类号： 540

面向大数据高通量计算的 CPU/GPU 并行优化技术研究

作 者 姓 名 扶聪

学 院 名 称 计算机学院

指 导 教 师 翟岩龙

答辩委员会主席

申 请 学 位 工学硕士

学 科 专 业 计算机科学与技术

学 位 授 予 单 位 北京理工大学

论 文 答 辩 日 期 2018 年 3 月

Synthesis and Application on textile of the Shape Memory Polyurethane

Candidate Name:	Fu Cong
School or Department:	****
Faculty Mentor:	Prof. YanLong Zhai
Chair, Thesis Committee:	Prof. **
Degree Applied:	****
Major:	****
Degree by:	Beijing Institute of Technology
The Data of Defence:	*, ****

面向大数据高通量计算的CPU/GPU并行优化技术研究

北京理工大学

研究成果声明

本人郑重声明：所提交的学位论文是我本人在指导教师的指导下进行的研究工作获得的研究成果。尽我所知，文中除特别标注和致谢的地方外，学位论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京理工大学或其它教育机构的学位或证书所使用过的材料。与我一同工作的合作者对此研究工作所做的任何贡献均已在学位论文中作了明确的说明并表示了谢意。

特此申明。

作者签名：_____　　签字日期：_____

关于学位论文使用权的说明

本人完全了解北京理工大学有关保管、使用学位论文的规定，其中包括：① 学校有权保管、并向有关部门送交学位论文的原件与复印件；② 学校可以采用影印、缩印或其它复制手段复制并保存学位论文；③ 学校可允许学位论文被查阅或借阅；④ 学校可以学术交流为目的，复制赠送和交换学位论文；⑤ 学校可以公布学位论文的全部或部分内容（保密学位论文在解密后遵守此规定）。

作者签名：_____　　导师签名：_____

签字日期：_____　　签字日期：_____

摘要

本文……。（摘要是一篇具有独立性和完整性的短文，应概括而扼要地反映出本论文的主要内容。包括研究目的、研究方法、研究结果和结论等，特别要突出研究结果和结论。中文摘要力求语言精炼准确，硕士学位论文摘要建议 500~800 字，博士学位论文建议 1000~1200 字。摘要中不可出现参考文献、图、表、化学结构式、非公知公用的符号和术语。英文摘要与中文摘要的内容应一致。）

关键词： 形状记忆；聚氨酯；织物；合成；应用（一般选 3 ～ 8 个单词或专业术语，且中英文关键词必须对应。）

Abstract

In order to exploit …….

Key Words: shape memory properties; polyurethane; textile; synthesis; application

目录

摘要 I

Abstract II

第 1 章　绪论 1

1.1　研究背景 1

1.1.1　大数据高通量仿真计算的研究背景和意义 1

1.1.2　CPU/GPU 异构计算研究背景和意义 2

1.2　CPU/GPU 异构编程简介 3

1.2.1　CUDA 平台简介 3

1.2.2　OpenCL 简介 4

1.2.3　OpenAcc 简介 5

1.3　GPU 结构简介 6

1.3.1　CPU 和 GPU 设计区别 6

结论 8

参考文献 9

附录 A　*** 10

附录 B　Maxwell Equations 11

攻读学位期间发表论文与研究成果清单 12

致谢 13

作者简介 14

第 1 章　绪论

本章主要分为五部分，分别介绍了大数据高通量仿真和 CPU/GPU 并行计算的研究背景和意义，CPU/GPU 异构编程平台和 GPU 物理特性，最后介绍了本篇论文的主要研究工作和创新点。

1.1　研究背景

互联网技术的飞速发展颠覆了人们的传统生活方式，几乎每一个用户个体每天都会产生数据，如何利用好这些海量数据成为各行各业急需解决的问题。大数据应用正融入生活的方方面面，大数据计算用于建模与仿真学科也是必然的发展趋势。传统的单 CPU 处理器结构计算能力有限，不适用于大数据仿真中，使用 CPU/GPU 异构计算才能快速处理大数据仿真。大数据仿真中 CPU/GPU 异构计算的优化是本文研究的主要内容。

1.1.1　大数据高通量仿真计算的研究背景和意义

建模与仿真技术是以相似理论、模型理论、系统技术、信息技术，以及建模与仿真应用领域的有关专业技术为基础，以计算机系统、与应用相关的物理效应设备及仿真器为工具，根据对系统仿真的目标建立并利用模型对系统进行研究、分析、试验、运行和评估的一门综合性、交叉性技术。建模与仿真技术和计算机科学一起，正成为继理论研究和实验研究之后的第三种认识、改造客观世界的重要手段。大数据技术的出现给仿真技术带来了新的需求与新的挑战，将大数据技术与仿真技术进行结合逐渐成为新的发展趋势。美国国防部高级计划研究局于 2012 年发起 1 亿美元的 XDATA 计划，用于支持国防信息数据处理和分析的软件和技术。军用仿真领域早在 90 年代就已经面临了大数据问题，美英 1997 年联合举行的 STOW97 战争综合演练仿真，由 500 台计算机构成，超过 3700 个仿真实体参与，在三个独立 48 小时阶段仿真中共计产生了大约 1.5TB 的数据。此后越来越多的军用仿真应用不断出现，高解析模型甚至实装模型不断运用到仿真中，仿真产生的数据也在快速增长。但是由于计算能力、计算方法等各种原因，仿真采集的绝大部分数据被丢弃了，仿真数据的使用多处于仿真结果检查或者简单统计分析的级别，没有起到军事分析、决策和预测的作用。国内外

仿真专家近年不断探讨大数据与仿真的结合方式，试图寻求新的建模与仿真方法。有学者提出大数据时代基于仿真的工程科学还需要发展仿真范式, 实现密集计算与密集数据的集成，以实现无组织复杂系统因果规律的发现。数据密集方法可由数据从整体上分阶段发现涌现性、演化机制下的结果, 而计算密集方法在部分时段或部分区域上满足了相似性研究的需要，为实现整体上的可预测性，即通过模型运行来揭示相应复杂性系统的运行规律，必须将数据密集与计算密集集成起来。这就是大数据高通量仿真要解决的核心问题之一。数据量较少的仿真实验很多情况下是不足以描述事物发展趋势的，海量的实验数据可以更好的覆盖到实体各方面运行和结果参数，计算出其中的因果关系。传统的单 CPU 处理器计算能力有限，不具备快速计算大数据仿真的能力。高性能异构集群协同计算方法是解决数据密集且计算密集型大数据高通量仿真的首选方法。随着 GPU 和 Intel 至强 Phi 等协处理器的不断发展，异构系统成为高性能计算的重要方式。将 GPU 与大数据处理框架集成来解决数据密集且计算密集型大数据计算是近年的研究热点。

1.1.2 CPU/GPU 异构计算研究背景和意义

GPU 最初是被设计用于显卡中图形计算的，帮助 CPU 快速处理复杂的图像计算。由于其卓越的并行计算能力，现在 GPU 不再只作用于显卡中，越来越多地作为协处理器帮助 CPU 处理高并行的大数据计算。在一些专业的领域中，比如机器学习和生物研究中，CPU/GPU 异构计算的应用十分广泛。从 CPU 和 GPU 性质上看，CPU 适合处理计算量比较小的逻辑运算，而 GPU 由于其内部有很多个小的计算单元非常适合处理高度并行的复杂计算。单 CPU 处理器结构的计算能力不足以满足海量数据的计算需求，因此业界普遍把 GPU 作为协处理器帮助 CPU 处理高并行的大数据计算。特别是在大数据高通量的仿真程序中，大量的实验参数和仿真计算必须依赖 CPU/GPU 异构计算才能快速得出实验结果。虽然 GPU 编程语言，如 CUDA、OpenAcc 和 C 语言语法规则差别不是很大，语法也比较简单，但是 GPU 编程对于普通程序员来说仍是一个巨大的挑战。设计者只有清楚地了解 GPU 内部结构和 CPU/GPU 通信特点才能编写出高效地 GPU 程序。首先，设计者需要人为判断哪些计算是数据量大，并行程度高，需要放到 GPU 中计算的。其次 CPU 和 GPU 的内存是隔离的，在相应的处理器上计算时必须先把需要的数据拷贝到相应的内存中，这需要程序员手动的设计 CPU 和 GPU 之间的数据交换过程。每一次 GPU kernel 的调用都涉及到数据在 CPU 和 GPU

内存的交换，所以在有多个 kernel 的 GPU 程序中，数据在 CPU 内存和 GPU 内存之间的反复拷贝将成为限制程序运行速度的一大障碍。与此同时，我们也发现编译制导方式的 CPU/GPU 异构编程虽然能够很大程度的降低 GPU 编程难度，但是由于 CPU 与 GPU 体系结构之间的差异和大数据高通量计算的数据密集特点，要实现 CPU/GPU 高性能协同计算，难度仍然不小。正如在 OpenAcc 官方网站中介绍的，由于使用了不同的编译制导语句控制 CPU 与 GPU 之间的数据通信，导致同一个算法性能相差 30 倍。这主要是因为 GPU 内存带宽能够达到 100GB/s~250GB/s，平均为 CPU 内存带宽的 8 倍，但是访问 GPU 显存却需要 400~800 个周期，是访问内存十几倍，PCI-E 总线的实际带宽也只能达到 8GB/s 左右，CPU/GPU 之间的 I/O 瓶颈问题已经是公认的阻碍 CPU/GPU 协同计算性能的关键问题，这一问题对大数据高通量仿真的影响将更加明显。因此本论文拟研究面向大数据高通量仿真的 CPU/GPU 异构计算数据通信模型与优化方法，以提高集群的大数据高通量计算能力。

1.2 CPU/GPU 异构编程简介

大数据时代需要更加快速、更高并行的计算方式，使得 GPU 编程越来越普遍。业界普遍使用的 GPU 编程语言有 CUDA、OpenCL 和 OpenAcc，本节将对 GPU 编程语言做出简单介绍。另外，GPU 编程语言仅仅提供了操作 GPU 的计算语法，如何根据 GPU 物理结构去设计 GPU 计算来充分优化 GPU 计算将在后续章节介绍。

1.2.1 CUDA 平台简介

CUDA(Compute Unified Device Architecture)，是显卡厂商 NVIDIA 推出的运算平台, 一种通用并行计算架构，支持 Linux 和 Windows 平台。该架构仅适用于 NVIDIA 显卡，已应用于 GeForce、ION、Quadro 以及 Tesla GPU 上, 使 GPU 能够快速解决复杂的计算问题。它包含了 CUDA 指令集架构以及 GPU 内部的并行计算引擎, 开发人员现在可以使用 C 语言来为 CUDA 架构编写程序, 所编写出的程序可以在支持 CUDA 的处理器上以超高性能运行。CUDA3.0 已经开始支持 C++ 和 FORTRAN。从 CUDA 体系结构的组成来说，包含了三个部分：开发库、运行期环境和驱动。开发库是基于 CUDA 技术所提供的应用开发库。目前 CUDA 的 1.1 版提供了两个标准的数学运算库——CUFFT （离散快速傅立叶变换）和 CUBLAS（离散基本线性计算）的实现。这两个数学运算库所解决的是典型的大规模的并行计算问题，也是在密集数据计算中非

常常见的计算类型。开发人员在开发库的基础上可以快速、方便的建立起自己的计算应用。此外，开发人员也可以在 CUDA 的技术基础上实现出更多的开发库。运行期环境提供了应用开发接口和运行期组件，包括基本数据类型的定义和各类计算、类型转换、内存管理、设备访问和执行调度等函数。基于 CUDA 开发的程序代码在实际执行中分为两种，一种是运行在 CPU 上的宿主代码（Host Code），一种是运行在 GPU 上的设备代码（Device Code）。不同类型的代码由于其运行的物理位置不同，能够访问到的资源不同，因此对应的运行期组件也分为公共组件、宿主组件和设备组件三个部分，基本上囊括了所有在 GPGPU 开发中所需要的功能和能够使用到的资源接口，开发人员可以通过运行期环境的编程接口实现各种类型的计算。由于目前存在着多种 GPU 版本的 NVidia 显卡，不同版本的 GPU 之间都有不同的差异，因此驱动部分基本上可以理解为是 CUDA-enable 的 GPU 的设备抽象层，提供硬件设备的抽象访问接口。CUDA 提供运行期环境也是通过这一层来实现各种功能的。目前基于 CUDA 开发的应用必须有 NVIDIA CUDA-enable 的硬件支持，NVIDIA 公司 GPU 运算事业部表示：一个充满生命力的技术平台应该是开放的，CUDA 未来也会向这个方向发展。由于 CUDA 的体系结构中有硬件抽象层的存在，因此今后也有可能发展成为一个通用的 GPGPU 标准接口，兼容不同厂商的 GPU 产品。CUDA 相比于其他 GPU 编程语言更加底层，需要程序员手动设计每次 GPU kernel 调用时的数据交换过程，这样可以使设计者根据需求充分利用 GPU 性能，同时也是对程序员的一大挑战，不合理的 GPU 编程设计将会带来巨大的性能丢失。

1.2.2 OpenCL 简介

OpenCL 是第一个面向异构系统通用目的并行编程的开放式、免费标准，也是一个统一的编程环境，便于软件开发人员为高性能计算服务器、桌面计算系统、手持设备编写高效轻便的代码，而且广泛适用于多核心 CPU、GPU、Cell 类型架构以及数字信号处理器等其他并行处理器，在游戏、娱乐、科研、医疗等各种领域被广泛使用。OpenCL 由一门编写 kernels 的语言和一组用于定义并控制平台的 API 组成，提供了基于任务分割和数据分割的并行计算机制，支持 Windows 和 Linux 操作系统。OpenCL 1.1 标准支持三维矢量和图像格式等新数据类型，支持处理多 Host 指令以及跨设备的缓冲区，并且通过链接 OpenCL 和 OpenGL 的事件，高效共享图像和缓冲区，大大改进了与 OpenGL 的互操作性。OpenCL 对图像格式的处理更加快速高效，也是图形学

中常用的 CPU/GPU 异构编程语言之一。OpenCL 2.0 在已有标准上首次引入共享虚拟内存的支持，CPU 和 GPU 内核可以直接共享复杂的，包含指针的数据结构，避免了程序员不熟悉异构编程造成的冗余数据交换，大大提高了并行编程的灵活性。支持动态并行，GPU 内核不需要与主句 CPU 交互情况下进行内核排队，实现灵活的调度方式，减少数据在 CPU 内存和 GPU 内存之间的拷贝，降低 CPU 处理器的负担。该标准改进图像支持，内核可以同时读写同一对象。新增对安卓系统的支持，安卓客户端可以安装驱动扩展将 OpenCL 作为共享对象进行操作。OpenCL 框架由平台 API，运行时 API 和 OpenCL 编程语言组成。平台 API 用于宿主机程序发现 OpenCL 设备的相关函数和为 OpenCL 应用创建上下文的相关函数。运行时 API 是用来管理上下文创建命令队列以及运行时的一些其他操作。OpenCL 语言是用来编写 GPU 内核计算代码的编程语言，与 c 语言相似。目前主流的芯片厂商都已推出支持 OpenCL 的芯片，各大品牌的手机平板几乎全都能支持 OpenCL。移动端的 GPU 通用计算必然成为被广泛使用的技术，推动移动计算进入一个全新时代。

1.2.3 OpenAcc 简介

OpenAcc 是另一种面向 CPU/GPU 异构结构并行编程 GPU 编程语言。2011 年 11 月，Cray、PGI、CAPS 和英伟达 4 家公司联合推出 OpenACC 1.0 编程标准，适用于 NVIDIA 和 AMD 显卡，支持 Windowshe 和 Linux 系统。OpenAcc 与 CUDA 和 OpenCL 相比，不需要程序员手动设计每一块数据在 CPU 和 GPU 内存之间的拷贝细节，而是以一个或者多个在 GPU 上执行的 kernel 为单位，以编译制导语句的方式告诉编译器在当前 kernel 的数据拷贝方式。OpenAcc 大大降低了 GPU 编程的门槛，很多数据的拷贝都是由编译器帮助程序员去完成的。程序设计者一般只需规划好哪些代码是在 GPU 上执行和相应的编译制导语句就可以写出效率较高的 OpenAcc 程序。PGI 编译器对 OpenAcc 的支持非常完善，GCC 编译器也支持大部分 OpenAcc 特性。使用 CUDA 重构 C 语言代码时，需要重写程序，OpenAcc 并行化的方式不是重写程序，而是在串行 C/C++ 或 Fortran 代码上添加一些编译制导标记。支持 OpenACC 的编译器能够看懂这些标记，并根据标记含义将代码编译成并行程序。对英伟达 GPU 来说，编译器将 C/C++/Fortran 代码翻译成 CUDA C/C++ 代码，然后编译链接成并行程序。对 AMD GPU 来说，中间代码是 OpenCL。可见 OpenAcc 语言相比 CUDA 和 OpenCL 更加高级一点，不需要程序设计者具体编写底层的数据拷贝语句，只需要程序员利用编

译制导语句告诉编译器每个 kernel 的执行方式和数据拷贝。程序耗时最多的是循环模块，OpenAcc 把每个并行计算量很大的循环结构，看成是一个可以在 GPU 上执行的 kernel。循环结构中的迭代计算被分散到 GPU 上不同线程执行，GPU 拥有多个核心可以同时快速运行多个线程，大大降低了 CPU 负担。经上述章节介绍可以看出，无论选取何种 GPU 编程语言进行代码编写，都需要设计者规划那些计算需要在 GPU 上执行和每次 GPU kernel 调用 CPU 内存和 GPU 内存之间的数据交换。本文的主要研究方向也是如何优化 CPU 内存 GPU 内存之间的数据交换来优化 GPU 程序。

1.3 GPU 结构简介

GPU 作为协处理器，帮助 CPU 处理图形计算和一些其他的高并行的复杂计算，降低 CPU 负担，提高整个系统运行速度。NVIDIA 和 AMD 是现在最大的两个显卡供应商，本节就 NVIDIA 显卡的物理结构做出简单介绍，并且分析一些在 CPU/GPU 异构计算中的一些瓶颈。

1.3.1 CPU 和 GPU 设计区别

CPU 和 GPU 的设计初衷打不相同，分别针对了两种不同的应用场景。CPU 作为中央处理器需要很强的通用性来处理各种不同数据类型，同时又要有处理判断循环等逻辑的控制结构，这使得 CPU 的内部结构异常复杂。而 GPU 的设计初衷就是解决高度统一的、相互无依赖的大规模数据和不需要被打断的计算环境。图 1 是来自 NVIDIA 文档关于 CPU 和 GPU 结构的描述图，其中绿色是计算单元，橙色是控制单元，蓝色是存储单元。从图中可以看出 GPU 内部有大量的计算单元和超长的流水线，只有非常简单的控制逻辑并且 cache，可见 GPU 的设计初衷就是用来处理大规模数据计算的。相比 GPU，CPU 内部计算单元相对较少，被缓存占据了大量空间，拥有复杂的控制逻辑单元和诸多优化电路。所以 CPU 的结构非常适合作为中央处理器来处理很多复杂的控制逻辑，同时大块的缓存也加速了对硬盘的读写，但是由于计算单元有限不擅长计算大规模的并行计算。同时，GPU 拥有很多的核心，可以快速并行运行多个线程，处理循环时，在不破坏数据依赖的情况下可以把每一轮的迭代计算分散到不同线程上。GPU 拥有很少的 cache，GPU 缓存的目的是和 CPU 一样缓存后续可能使用到的数据，而是缓存线程需要的数据。当有多个线程需要一个数据块时，GPU cache 会合并这些请求，从 DRAM 中缓存这些数据到 cache，然后再把 GPU 缓存中的

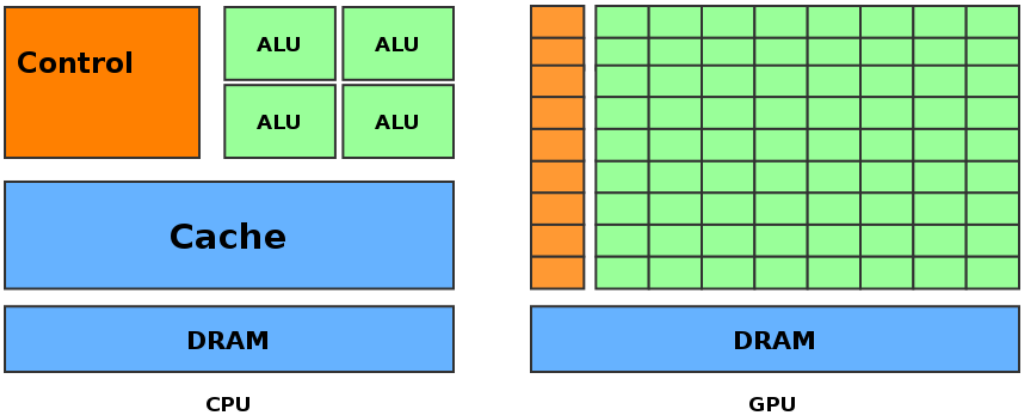

图 1.1　CPU 和 GPU 结构比较

数据发送到相应线程。从上述描述可以看出，CPU 作为中央处理器处理通用计算，负责和操作系统、硬盘、网络硬件等交互，拥有复杂的逻辑控制单元可以快速处理每条指令。而 GPU 作为协处理器处理，是用来帮助 CPU 处理专用大规模数据的并行计算，不需要与操作系统和其他硬件交互。

结论

本文采用……。（结论作为学位论文正文的最后部分单独排写，但不加章号。结论是对整个论文主要结果的总结。在结论中应明确指出本研究的创新点，对其应用前景和社会、经济价值等加以预测和评价，并指出今后进一步在本研究方向进行研究工作的展望与设想。结论部分的撰写应简明扼要，突出创新性。）

参考文献

附录 A　***

附录相关内容…

附录 B Maxwell Equations

因为在柱坐标系下，$\bar{\bar{\mu}}$ 是对角的，所以 Maxwell 方程组中电场 **E** 的旋度

所以 **H** 的各个分量可以写为：

$$H_r = \frac{1}{\mathbf{i}\omega\mu_r}\frac{1}{r}\frac{\partial E_z}{\partial\theta} \tag{B–1a}$$

$$H_\theta = -\frac{1}{\mathbf{i}\omega\mu_\theta}\frac{\partial E_z}{\partial r} \tag{B–1b}$$

同样地，在柱坐标系下，$\bar{\bar{\epsilon}}$ 是对角的，所以 Maxwell 方程组中磁场 **H** 的旋度

$$\nabla\times\mathbf{H} = -\mathbf{i}\omega\mathbf{D} \tag{B–2a}$$

$$\left[\frac{1}{r}\frac{\partial}{\partial r}(rH_\theta) - \frac{1}{r}\frac{\partial H_r}{\partial\theta}\right]\hat{\mathbf{z}} = -\mathbf{i}\omega\bar{\bar{\epsilon}}\mathbf{E} = -\mathbf{i}\omega\epsilon_z E_z\hat{\mathbf{z}} \tag{B–2b}$$

$$\frac{1}{r}\frac{\partial}{\partial r}(rH_\theta) - \frac{1}{r}\frac{\partial H_r}{\partial\theta} = -\mathbf{i}\omega\epsilon_z E_z \tag{B–2c}$$

由此我们可以得到关于 E_z 的波函数方程：

$$\frac{1}{\mu_\theta\epsilon_z}\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial E_z}{\partial r}\right) + \frac{1}{\mu_r\epsilon_z}\frac{1}{r^2}\frac{\partial^2 E_z}{\partial\theta^2} + \omega^2 E_z = 0 \tag{B–3}$$

攻读学位期间发表论文与研究成果清单

[1] 高凌. 交联型与线形水性聚氨酯的形状记忆性能比较 [J]. 化工进展, 2006, 532 − 535.（核心期刊）

致谢

本论文的工作是在导师……。

作者简介

本人…。