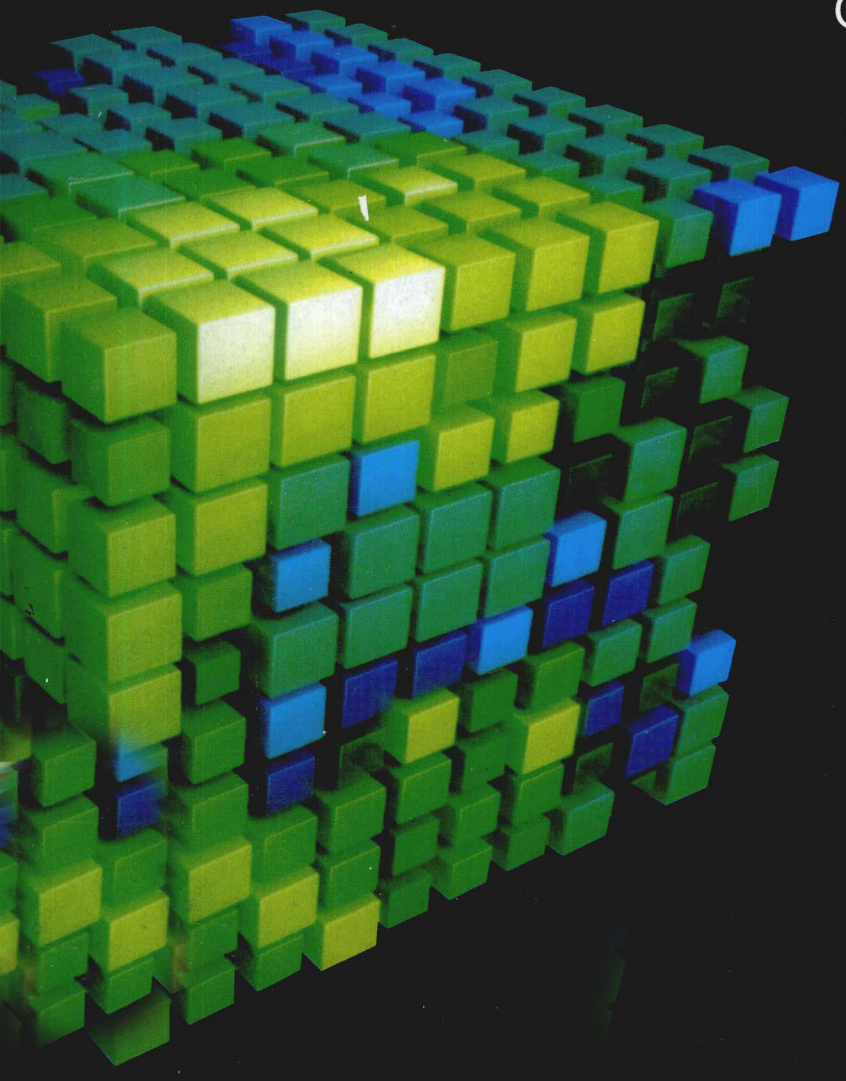
HZ Books
华章科技

PEARSON

GPU高性能编程CUDA实战

CUDA By Example

an Introduction to General-Purpose GPU Programming



（美） Jason Sanders
Edward Kandrot 著

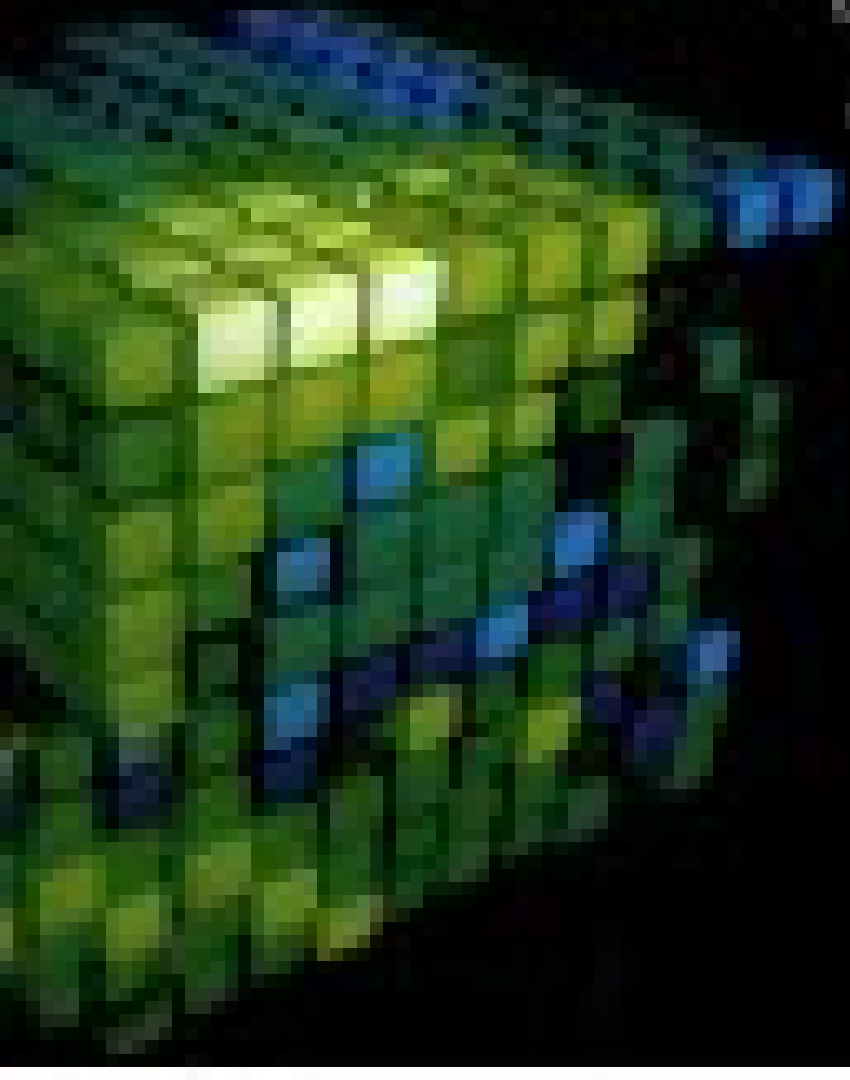
聂雪军 等译

机械工业出版社
China Machine Press

GPU高性能编程 CUDA实战

CUDA By Example

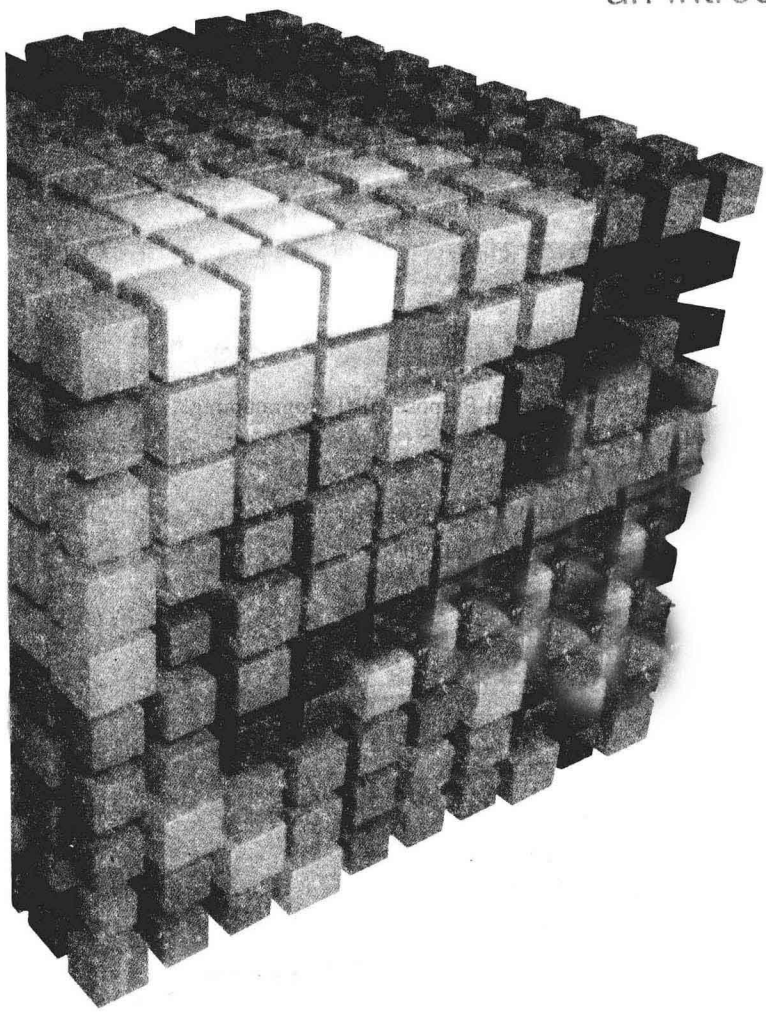
an Introduction to General-Purpose GPU Programming



GPU高性能编程 CUDA实战

CUDA By Example

an Introduction to General-Purpose GPU Programming



（美）Jason Sanders
Edward Kandrot 著
聂雪军 等译

机械工业出版社
China Machine Press

CUDA是一种专门为提高并行程序开发效率而设计的计算架构。在构建高性能应用程序时，CUDA架构能充分发挥GPU的强大计算功能。本书首先介绍了CUDA架构的应用背景，并给出了如何配置CUDA C的开发环境。然后通过矢量求和运算、矢量点积运算、光线跟踪、热传导模拟等示例详细介绍了CUDA C的基本语法和使用模式。通过学习本书，读者可以清楚了解CUDA C中每个功能的适用场合，并编写出高性能的CUDA软件。

本书适合具备C或者C++知识的应用程序开发人员、数值计算库开发人员等，也可以作为学习并行计算的学生和教师的教辅。

本书版权登记号：图字：01-2010-5159

图书在版编目（CIP）数据

GPU高性能编程CUDA实战/（美）桑德斯（Sanders, J.）著；聂雪军等译. —北京：机械工业出版社，2011.1

书名原文：CUDA by Example: an Introduction to General-Purpose GPU Programming

ISBN 978-7-111-32679-3

Ⅰ. G…　Ⅱ. ①桑…　②聂…　Ⅲ. 图像处理—程序设计　Ⅳ. TP391.41

中国版本图书馆CIP数据核字（2010）第235229号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）
责任编辑：秦　健
北京市荣盛彩色印刷有限公司印刷
2011年1月第1版第1次印刷
186mm × 240mm · 13.25印张
标准书号：ISBN 978-7-111-32679-3
定价：39.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991；88361066
购书热线：（010）68326294；88379649；68995259
投稿热线：（010）88379604
读者信箱：hzjsj@hzbook.com

译 者 序

并行计算已成为突破摩尔定理局限性的重要研究方向，而GPU强大的并行计算能力也因此吸引了全球广泛的研究兴趣。然而，在实现通用并行计算时，GPU计算模式存在着一些限制。首先，GPU的设计初衷是为了加速应用程序中的图形绘制运算，因此开发人员需要通过OpenGL或者DirectX等API来访问GPU，这不仅要求开发人员掌握一定的图形编程知识，而且要想方设法将通用计算问题转换为图形计算问题。其次，GPU与多核CPU在计算架构上有着很大不同，GPU更注重于数据并行计算，即在不同的数据上并行执行相同的计算，而对并行计算中的互斥性、同步性以及原子性等方面支持不足。这些因素都限制了GPU在通用并行计算中的应用范围。

CUDA架构的出现解决了上述问题。CUDA架构专门为GPU计算设计了一种全新的结构，目的正是为了减轻GPU计算模型中的这些限制。在CUDA架构下，开发人员可以通过CUDA C对GPU编程。CUDA C是对标准C的一种简单扩展，学习和使用起来都非常容易，并且其最大的优势在于不需要开发人员具备图形学知识。

本书的主要内容是介绍如何通过CUDA C来编写在GPU上运行的并行程序。本书首先介绍了CUDA架构的应用背景，并给出了如何配置CUDA C的开发环境。然后，本书通过矢量求和运算、矢量点积运算、光线跟踪、热传导模拟、直方图统计等示例详细介绍了CUDA C的基本语法和使用模式。在这些示例中还穿插介绍了GPU的各种硬件特性及适用环境，例如常量内存、事件、纹理内存、图形互操作性、原子操作、流以及多GPU架构等。

这些示例的构思以及分析过程都很容易理解，它们也是本书最具价值的部分。读者在阅读这些内容时要反复思考，做到融会贯通，举一反三。只要掌握了这些简单的示例，更复杂的问题也能迎刃而解。本书适合所有程序员阅读，只需具备基本的C语言知识即可。最后，本书还给出了CUDA C的其他一些参考资源。

参与本书翻译工作的主要有李杨、吴汉平、徐光景、童胜汉、陈军、胡凯等。由于译者的时间和水平有限，翻译中的疏漏和错误在所难免，还望读者和同行不吝指正。

聂雪军

2010年于武汉

序

主流芯片制造厂商（例如NVIDIA）最近一系列的动作已经表明了，未来的微处理器系统和大型高性能计算系统都将是异构的。在这些异构系统中将同时使用两种不同的技术，二者在不同系统中所占的比例是不同的。

- **多核CPU技术**：CPU核的数量将不断增长，因为人们希望将越来越多的部件封装到芯片上，同时又要避免功耗墙、指令集并行性墙、内存墙等。
- **专用硬件和大规模并行加速器**：例如，近年来NVIDIA公司推出的GPU在浮点运算上的性能已经超过了标准CPU的性能。而且，GPU编程也正变得和多核CPU编程一样简单。

在未来的设计中，这两种技术在系统中所占的比例并不是固定的，并且随着时间的推移很可能发生变化。显然，在未来的计算机系统中，无论是笔记本电脑还是超级计算机，都将包含各种异构的部件。事实上，这种系统的浮点计算能力已经达到了P级（即每秒钟执行10^{15}次浮点运算）。

但是，在采用混合处理器的新计算环境中，开发人员面临的问题和挑战仍然很艰难。软件基础架构中的关键部分要赶上这种变革的步伐也同样很困难。在一些情况中，软件的性能无法随着处理器核数量的增加而增加，因为越来越多的时间消耗在数据移动而不是计算上。在其他情况中，软件通常是在硬件出现数年后才发布性能调优后的版本，因此在发布时就已经过时了。还有一些情况，例如在最新的GPU上，软件根本无法运行，因为编程环境已经发生了太大的变化。

本书解决了软件开发中面临的核心问题，它介绍了近年来在编写大规模并行加速器中最具创新性和功能最强大的解决方案之一。

本书介绍了如何使用CUDA C来编写程序，不仅给出了详细的示例，而且还介绍了程序的设计流程以及NVIDIA GPU的高效使用方式。本书介绍了并行计算的一些基本概念，包括简单的示例和调试方法（涵盖逻辑调试和性能调试），此外还介绍了一些高级主题以及在编译和使用应用程序时需要注意的问题，并通过编程示例来进一步强化这些概念。

如果读者需要开发基于加速器的计算系统，那么本书是首选。本书深入分析了并行计算，并为其中的多种问题提供了解决方案。本书对于应用程序开发人员、数值计算库开发人员，以及学习并行计算的学生和教师来说尤为有用。

我很高兴从本书中学到了不少东西，并且我相信你也会有同样的体会。

——Jack Dongarra

田纳西大学杰出教授，美国橡树岭国家实验室杰出研究人员

前　言

本书介绍了如何利用计算机中图形处理器（Graphics Process Unit, GPU）的强大计算功能来编写各种高性能的应用软件。虽然GPU的设计初衷是用于在显示器上渲染计算机图形（现在仍然主要用于这个目的），但在科学计算、工程、金融以及其他领域中，人们开始越来越多地使用GPU。我们将解决非图形领域中的问题的GPU程序统称为通用GPU程序。值得高兴的是，虽然你需要具备C或者C++的知识才能充分理解本书的内容，但却不需要具备计算机图形学的知识。任何图形学的基础都不要！GPU编程只是使你进一步增强现有的编程技术。

在NVIDIA GPU上编写程序来完成通用计算任务之前，你需要知道什么是CUDA。NVIDIA GPU是基于CUDA架构而构建的。你可以将CUDA架构视为NVIDIA构建GPU的模式，其中GPU既可以完成传统的图形渲染任务，又可以完成通用计算任务。要在CUDA GPU上编程，我们需要使用CUDA C语言。在本书前面的内容中可以看到，CUDA C本质上是对C进行了一些扩展，使其能够在像NVIDIA GPU这样的大规模并行机器上进行编程。

我们为经验丰富的C或者C++程序员编写了本书，这些程序员通常较为熟悉C语言，因此能很轻松地阅读或者编写C代码。本书不仅将进一步增强你的C语言编程能力，而且还能作为使用NVIDIA的CUDA C编程语言的一本快速入门书籍。你既不需要具备任何在大规模软件架构上工作的经验，也不需要有编写过C编译器或者操作系统内核的经历，此外也无需了解ANSI C标准的细枝末节。本书并没有花时间来回顾C语言的语法或者常用的C库函数，例如malloc()或者memcpy()，我们假设你对这些概念已经非常熟悉了。

虽然本书的目的并不是介绍通用的并行编程技术，但你在书中仍将学习到一些通用的并行编模式。此外，本书并不是一本详细介绍CUDA API的参考书，也不会详细介绍在开发CUDA C软件时可以使用的各种工具。因此，我们强烈建议将本书与NVIDIA的免费文档结合起来阅读，例如《NVIDIA CUDA Programming Guide》和《NVIDIA CUDA Best Practices Guide》等。然而，你不用费工夫去收集所有这些文档，因为我们将介绍你需要的所有内容。

不会费太多的周折，CUDA C编程领域欢迎你的到来！

致　谢

任何一本技术书籍的完成都离不开许多人的帮助，本书也不例外。作者要感谢许许多多的人，在这里将列举其中一些。

感谢Ian Buck，NVIDIA GPU计算软件的高级主管，他在本书编写的各个阶段，从本书的构思到最终完成，都给予了极大的帮助。我们还要感谢Tim Murray，他是一位始终保持微笑的审阅者，感谢他保证了本书的技术准确性和可读性。感谢插图设计师Darwin Tat，他在非常紧张的时间中为本书制作了精美的封面和插图。最后，我们要感谢John Park，他帮助本书完成了在出版流程中涉及的复杂法律步骤。

如果没有Addison-Wesley工作人员的帮助，本书将仍然只是作者的一个转瞬而逝的想法而已。感谢Peter Gordon、Kim Boedigheimer、Julie Nahil，感谢他们的耐心、专业，最终使得本书顺利出版。此外，Molly Sharp和Kim Wimpsett的编辑工作帮助本书从一堆充满错误的文档转换成为读者们现在看到的文字。

如果没有其他撰稿人的帮助，读者将无法阅读本书的一些内容。特别是，Nadeem Mohammad收集了第1章中介绍的CUDA案例分析，Nathan Whitehead无私地提供了本书的示例代码。

此外，还要感谢最先阅读本书草稿的一些人，感谢他们提供了有帮助的反馈，他们包括Genevieve Breed和Kurt Wall。在本书的编写过程中，许多NVIDIA的软件工程师都提供了宝贵的技术帮助，包括Mark Hairgrove，他通读了本书，并指出了所有不一致的地方——包括技术上的、拼写上的和语法上的。Steve Hines、Nicholas Wilt和Stephen Jones审阅了一些章节的CUDA API，并补充了作者忽略的许多细节。还要感谢Randima Fernando，感谢他帮助我们启动了本书的编写工作，感谢Michael Schidlowsky在他的书中对Jason表达的谢意。

最后还要感谢我们的家人。我们要感谢我们的家庭，他们陪伴我们经历了每一件事情，并最终完成本书。我们要特别感谢我们两个人的父母，Edward、Kathleen Kandrot、Stephen、Helen Sanders。感谢我们的兄弟，Kenneth Kandrot和Corey Sanders。感谢你们给予的无尽支持。

作者简介

Jason Sanders是NVIDIA公司CUDA平台小组的高级软件工程师。他在NVIDIA的工作包括帮助开发早期的CUDA系统软件，并参与OpenCL 1.0规范的制定，该规范是一个用于异构计算的行业标准。Jason在加州大学伯克利分校获得计算机科学硕士学位，他发表了关于GPU计算的研究论文。此外，他还获得了普林斯顿大学电子工程专业学士学位。在加入NVIDIA公司之前，他曾在ATI技术公司、Apple公司以及Novell公司工作过。

Edward Kandrot是NVIDIA公司CUDA算法小组的高级软件工程师。他在代码优化和提升性能等方面拥有20余年的工作经验，参与过Photoshop和Mozilla等项目。Kandrot曾经在Adobe公司、Microsoft公司工作过，他还是许多公司的咨询师，包括Apple公司和Autodesk公司。

目　录

第1章
为什么需要CUDA

就在前不久，并行计算（Parallel Computing）还被认为是一种“神秘的”技术，属于计算机科学的专业领域之一。然而，这种观念在近几年发生了巨大转变。在计算机业界，并行计算已不再是一个冷门领域，几乎每个程序员都要学习并行计算的知识才能胜任计算机科学领域的工作。当你拿起这本书时，或许还没有意识到并行编程在当前计算机业界有着怎样的重要地位，也没有意识到并行编程在未来数年中将起到如何重要的作用。在本章中，我们将分析近年来硬件领域的发展趋势，以及它们对软件开发的推动作用。我希望通过这些介绍使读者们意识到，并行计算的革命已经开始了，通过学习和使用CUDA C，我们能够在由CPU处理器和GPU构成的异构平台上编写出高性能的应用程序。

1.1　本章目标

通过本章的学习，你可以：

- 了解并行计算正在发挥越来越重要的作用。
- 了解GPU计算和CUDA的发展历程。
- 了解一些通过使用CUDA C而获得成功的应用程序。

1.2　并行处理的历史

近年来，计算机业界正在迅速迈进并行计算时代。在2010年，几乎所有的客户计算机都采用了多核处理器。从入门级的双核低端上网本，到8核或者16核的工作站计算机，并行计算已不再是超级计算机或者大型机的专属技术。此外，一些电子设备，例如手机和便携式音乐播放器等，都开始集成并行计算功能，以提供比早期产品更强大的功能。

随着时间的推移，软件开发人员将看到越来越多的并行计算平台和技术，并需要基于这些平台和技术为不断成熟的用户群提供崭新且丰富的体验。命令提示符的界面已经过时了，现在流行的是多线程图形界面。只能接打电话的手机也已经过时了，现在流行的是能同时播放音乐、浏览网页和提供GPS服务的手机。

中央处理器

在30多年前，要想提升客户计算设备的性能，主要手段之一就是提高处理器的时钟频率。在20世纪80年代早期出现的第一台个人计算机，其中央处理器（CPU）的运行时钟频率为1MHz。30年后，大多数桌面处理器的时钟频率都在1GHz和4GHz之间，这比当初个人计算机的时钟频率要快1 000倍。尽管提高CPU时钟频率并不是提升计算性能的唯一方法，但却是一种相对稳定的提升方法。

然而，近年来计算机制造商们却不得不开始寻求其他的替代方法来提升计算性能。由于在集成电路元器件中存在的各种严重限制，人们已经无法在现有的架构上通过提高处理器时钟频率来提升性能。随着功耗与发热的急剧升高以及晶体管的大小已接近极限，研究人员和制造商们开始寻求其他的方式。

除了个人计算机外，超级计算机在过去数十年里也借助相同的方式获得了极大的性能提升。超级计算机中的处理器与个人计算机中CPU一样，在时钟频率上都得到了极大提升。然而，除了在单个处理器上提升性能外，超级计算机的制造商们还通过增加处理器的数量来提升性能。在高速的超级计算机中，几十个、几百个甚至几千个处理器同时运行是很常见的情形。

当人们在探索如何提升个人计算机的性能时，超级计算机中性能提升方式引出了一个很好的问题：为什么不在个人计算机中放置多个处理器，而不是仅提升单个处理器核的性能？这样，在不需要提高处理器运行频率的情况下，个人计算机的性能就能获得持续的提升。

在2005年，当面对竞争日趋激烈的市场以及越来越少的可行方式时，业界一些领先的CPU制造商们开始提供带有两个计算核的处理器。在接下来的几年中，他们延续了这种发展趋势，不断推出3核、4核、6核以及8核的中央处理器。这种趋势也称为多核革命（Multicore Revolution），它标志着在个人计算机上开始发生重大的转变。

当前，要购买一台单核CPU的桌面计算机已经比较困难了。即使在低端、低能耗的中央处理器中，通常都包含有两个或多个计算核。一些业界领先的CPU制造商已经宣布在未来将计划推出12核和16核的CPU，这进一步证明了并行计算已经给人们带来了不可忽视的好处。

1.3　GPU计算的崛起

与中央处理器传统的数据处理流水线相比，在图形处理器（Graphics Processing Unit，GPU）上执行通用计算还是一个新概念。事实上，在计算领域中，GPU本身在很大程度上就是一个新概念。然而，在图形处理器上执行计算却并非新概念。

1.3.1　GPU简史

在前面介绍了中央处理器在时钟频率和处理器核数量上的发展历程。与此同时，图形处理技术同样经历了巨大的变革。在20世纪80年代晚期到90年代早期之间，图形界面操作系统（例如Microsoft公司的Windows）的普及推动了新型处理器的出现。在20世纪 90年代早期，用户开始购买配置2D显示加速器卡的个人计算机。这些显卡提供了基于硬件的位图运算功能，能够在图形操作系统的显示和可用性上起到辅助作用。

在专业计算领域，Silicon Graphics在整个20世纪80年代都致力于推动3D图形在各种市场上的应用，包括政府与国防等应用领域以及科学与技术的可视化技术等，此外还提供了各种工具来制作炫目的电影特效。在1992年，SG公司发布了OpenGL库，这是一个对该公司硬件进行编程的接口。SG公司试图将OpenGL作为一种标准化的、与平台无关的3D图形应用程序编写方法。并行计算和GPU的演变情况是类似的，这些技术进入到消费者应用程序中只是时间早晚的问题。

20世纪90年代中期，在消费者应用程序中采用3D图形技术的需求开始快速增长，这为两类非常重要的应用程序创造了基础条件。首先，许多第一人称游戏，例如Doom、Duke Nukem 3D和Quake等，都要求为PC游戏创建更真实的3D场景。虽然最终3D图形技术融入几乎所有的计算机游戏中，但第一人称射击类型游戏在初期对3D图形技术在消费者应用程序中的普及起到

了极大的推动作用。其次，一些制造商，例如NVIDIA公司、ATI Technologies公司以及3Dfx Interactive公司等，都开始发布一些普通消费者能够买得起的图形加速器卡，以便吸引更广泛的关注。这些事件都促使3D图形技术成为在未来几年中占据重要地位的技术之一。

NVIDIA公司GeForce 256显卡的发布进一步提升了消费者图形硬件的功能。GeForce 256第一次实现了在图形处理器上直接运行变形与光效（Transform and Lighting）等计算，因此在某些应用程序中能实现更强的视觉效果。由于变形和光效已经成为OpenGL图形流水线功能（Graphics Pipeline）的一部分，因此GeForce 256标志着越来越多的图形流水线功能将开始直接在图形处理器上实现。

从并行计算的角度来看，NVIDIA在2001年发布的GeForce 3代表着GPU技术上的最重要突破。GeForce 3系列是计算工业界的第一块实现Microsoft DirectX 8.0标准的芯片。该标准要求在硬件中同时包含可编程的顶点着色功能（Vertex Shading Stage）和像素着色功能（Pixel Shading Stage）。开发人员也正是从GerForce 3系列开始第一次能够对GPU中的计算实现某种程度的控制。

1.3.2　早期的GPU计算

GPU中的可编程流水线吸引了许多研究人员来探索如何在除了OpenGL或者DirecX渲染之外的领域中使用图形硬件。早期的GPU计算使用起来非常复杂。由于标准图形接口，例如OpenGL和DirectX，是与GPU交互的唯一方式，因此要在GPU上执行计算，就必然受限于图形API的编程模型。基于这个原因，研究人员开始探索如何通过图形API来执行通用计算，然而这也使得他们的计算问题在GPU看来仍然是传统的渲染问题。

在2000年早期，GPU的主要目标都是通过可编程计算单元为屏幕上的每个像素计算出一个颜色值，这些计算单元也称为像素着色器（Pixel Shader）。通常，像素着色器根据像素在屏幕上的位置（x，y）以及其他一些信息，对各种输入信号进行合成并计算出最终的颜色值。这些信息包括输入颜色、纹理坐标、或其他可以传递给着色器的属性。由于对输入颜色和纹理的计算完全是由程序员控制的，因此研究人员注意到，这些输入的“颜色”实际上可以是任意数据。

因此，如果输入值实际上并不是表示颜色值，那么程序员可以对各个像素着色器进行编程从而对输入值执行任意计算。计算结果将交回GPU作为像素的最终“颜色”，尽管这些颜色值只是程序员通过GPU对输入数据进行计算的结果。研究人员可以获得这些计算结果，而GPU永远都不会知道其中的过程。事实上，GPU能够执行除渲染之外的任务，只要这些任务看起来像是一个标准的渲染任务。虽然这种技术非常聪明，但使用起来却非常复杂。

由于GPU有着很高的计算吞吐量，因此最初从这些实验中得到的结果表明GPU计算有着非常光明的应用前景。然而，这种编程模型对于开发人员来说存在着非常大的局限性。首先，该

模型有着严格的资源限制，因为程序只能以颜色值和纹理单元等形式来输入数据。此外，程序员在将计算结果写入内存的方式以及位置上同样存在着严格限制，如果在算法中需要写入到内存的任意（分散）位置，那么将无法在GPU上运行。而且，我们也无法预测所使用的GPU能否处理浮点数据，如果不能处理浮点数据，那么大多数科学计算都将无法使用GPU。最后，当出现问题时，例如程序计算得到错误结果，程序无法终结，或者使计算机挂起，那么没有任何一种方便的方法能够对GPU上执行的代码进行调试。

除了这些限制因素之外，如果程序员希望通过GPU来执行通用计算，那么他们还需要学习OpenGL或者DirectX，因为这些接口仍然是与GPU交互的唯一方式。这不仅意味着要将数据保存在图形纹理中并调用OpenGL或者DirectX函数来执行计算，而且还意味着要使用特殊的图形编程语言来编写这些计算，这些语言也称为着色语言（Shading Language）。因此，研究人员在开始使用GPU的强大计算功能之前，首先需要考虑严格的资源限制和编程限制，然后还要学习计算机图形学和着色语言，这种负担对于研究人员来说过于沉重，因此GPU计算在早期并没有被广泛的接受。

1.4　CUDA

直到在GeForce 3系列发布五年之后，GPU计算才开始逐渐成为主流技术。在2006年11月，NVIDIA公布了业界的第一个DirectX 10 GPU，即GeForce 8800 GTX。GeForce 8800 GTX也是第一个基于NVIDIA的CUDA架构构建的GPU。CUDA架构专门为GPU计算设计了一种全新的模块，目的是减轻早期GPU计算中存在的一些限制，而正是这些限制使得之前的GPU在通用计算中没有得到广泛应用。

1.4.1　CUDA架构是什么

在之前的图形处理架构中，计算资源划分为顶点着色器和像素着色器，而CUDA架构则不同，它包含了一个统一的着色器流水线，使得执行通用计算的程序能够对芯片上的每个数学逻辑单元（Arithmetic Logic Unit，ALU）进行排列。由于NVIDIA希望使新的图形处理器能适用于通用计算，因此在实现这些ALU时都确保它们满足IEEE单精度浮点数学运算的需求，并且可以使用一个裁剪后的指令集来执行通用计算，而不是仅限于执行图形计算。此外，GPU上的执行单元不仅能任意地读/写内存，同时还能访问由软件管理的缓存，也称为共享内存。CUDA架构的所有这些功能都是为了使GPU不仅能执行传统的图形计算，还能高效地执行通用计算。

1.4.2　CUDA架构的使用

NVIDIA并不局限于通过集成CUDA架构的硬件来为消费者同时提供计算功能和图形功能。尽管NVIDIA在芯片中增加了许多功能来加速计算，但仍然只能通过OpenGL或者DirectX来访

问这些功能。这不仅要求用户仍然要将他们的计算任务伪装为图形问题，而且还需要使用面向图形的着色语言（例如OpenGL的GLSL或者Microsoft的HLSL）来编写计算代码。

为了尽可能地吸引更多的开发人员，NVIDIA采取了工业标准的C语言，并且增加了一小部分关键字来支持CUDA架构的特殊功能。在发布了GeForce 8800 GTX之后的几个月，NVIDIA公布了一款编译器来编译CUDA C语言。这样，CUDA C就成为了第一款专门由GPU公司设计的编程语言，用于在GPU上编写通用计算。

除了专门设计一种语言来为GPU编写代码之外，NVIDIA还提供了专门的硬件驱动程序来发挥CUDA架构的大规模计算功能。现在，用户不再需要了解OpenGL或者DirectX图形编程结构，也不需要将通用计算问题伪装为图形计算问题。

1.5 CUDA的应用

自从CUDA C在2007年首次出现以来，许多企业都尝试以CUDA C为基础来构建应用程序，并获得了极大的成功。基于CUDA C编写的代码比之前的代码在性能上提升了多个数量级。而且，与传统在CPU上运行的应用程序相比，在NVIDIA图形处理器上运行的应用程序的单位成本和单位能耗都要低很多。下面就给出CUDA C以及CUDA架构在一些方面的成功应用。

1.5.1 医学图像

在过去20年中，乳腺癌患者的数量持续在增长。在众多研究人员的不懈努力下，对这种可怕疾病的预防与治疗工作取得了重大进展。人们的研究热点在于如何提前发现乳腺癌，从而有效地防止在辐射和化疗中产生的严重副作用，外科手术造成的永久性后遗症，以及由于治疗无效而导致的死亡等。因此，研究人员一致在努力找到某种快速，精确并且影响最小的方式来找出乳腺癌的早期症状。

乳房X线照片（Mammogram）是当前识别早期乳腺癌的最好技术之一，然而这种技术在实际应用中存在一些严重的限制。该项技术需要拍摄两张或者多张图像，并且由一位熟练的医生来分析图像以找出潜在的肿瘤。此外，X射线还会造成反复地辐射患者的胸腔。在经过仔细研究后，医生通常要求进一步的、并且更为具体的成像——有时甚至需要进行活体检查，从而判断癌症的可能性。诊断中的误报（False Positive）不仅会导致昂贵的后续诊断工作，而且在最终报告出来之前，会给患者造成过度的压力。

超声波成像技术比X射线成像技术要更为安全，因此医生通常将其与乳房X线照片结合起来使用，以辅助乳腺癌的治疗与诊断。然而，常规的乳房超声波技术仍然有其局限性。因此，人们研发了TechniScan医疗系统。TechniScan采用了一种三维的超声波成像方法，但这种解决方案却由于一个非常简单的问题而无法投入实际使用：计算量的限制。简单来说，在将收集