The Translator's Words 译 者 序

CUDA（Compute Unified Device Architecture，统一计算设备架构）是NVIDIA（英伟达）提出的并行计算架构，结合了CPU和GPU的优点，主要用来处理密集型及并行计算。CPU和GPU是两个独立的处理器，通过单个计算节点中的PCI-Express总线相连，GPU用来提高计算密集型应用程序中并行程序段的执行速度，CPU则负责管理设备端的资源。CUDA编程的独特优势在于开放的架构特性可以使程序员在功能强大的硬件平台上充分挖掘其并行性，既满足了计算密集型程序的需要，又实现了程序的易读性及便捷性。

对任何想要使用GPU来进行科研或技术编程的人来说，本书都将是一个珍贵的资源宝库，它全面介绍了CUDA编程接口及其用法，包含但不局限于CUDA编程模型、GPU执行模型、GPU内存模型、CUDA流和事件、多GPU编程的相关技术、CUDA库等内容。此外，本书列举了大量的范例来帮助读者入门，并提供了程序下载地址，供读者亲自运行体验。在基本掌握了本书的内容之后，你将发现CUDA C编程是一项十分简单、有趣且高效的工作。

本书第1章简要介绍了GPU和CPU构成的异构架构，对使用CUDA进行异构并行计算的原理进行了简要说明。第2章介绍了CUDA编程模型，从逻辑上解释了什么是CUDA大规模并行计算。第3章介绍了CUDA执行模型。第4章给出了CUDA内存模型，分析了全局内存的访问模式。第5章介绍了数据在共享内存中的存储方式，以及如何使用共享内存提高函数性能。第6章讲述了如何使用CUDA流实现网格级的并发。第7章介绍了计算密集型应用程序中的CUDA指令级原语。第8章介绍了CUDA函数库及其作用域，包括线性代数、傅里叶变换、随机数等。第9章讲述了多GPU编程。第10章则说明了在使用CUDA进行程序开发时的一些注意事项。

译者在翻译过程中力求忠于原著，但限于译者的能力及水平，如有纰漏，还望读者朋友们不吝赐教，译者在此表示衷心的感谢。

推 荐 序 *Foreword*

图形处理器（GPU）已经有了很大的发展。从最初能够瞬时生成图像输出的专业图形处理器，发展到显示单元，GPU 已经成为人们在进行超高速数据处理时热衷的技术手段。在过去的几年中，GPU 已经越来越多地附加到 CPU 上，用于加速异构计算中的各种计算。如今，许多桌面系统、计算集群，甚至世界上许多最大的超级计算机都配置有 GPU。图形处理器有着用于专业技术计算的大数据计算能力，它们已经在广泛的学科领域中实现了科学和工程的巨大进步，因为它们使大量计算核心并行工作，同时又将功耗预算保持在合理范围内。

所幸，GPU 编程接口也一直与时俱进。过去，要在应用功能之外使用它们要费好大一番功夫，GPU 编程人员不得不去熟练掌握那些通常只有图形程序员才能理解到位的诸多概念。如今的系统提供了一个更加便捷的手段来创建可以在其上运行的应用软件，这就是 CUDA。

CUDA 是最受欢迎的应用编程接口之一，可用于加速 GPU 上的一系列计算内核。它可以让 C 或 C++ 代码在 GPU 上合理高效地运行。CUDA 既满足了为使其良好运行而了解其架构的需要，又有为了使用和输出可读程序的编程接口，并且完美地平衡了这两项功能。

对任何想要使用 GPU 来进行科研或技术编程的人来说，该书都将是一个珍贵的资源宝库。它全面介绍了 CUDA 编程接口及其用法。首先，它描述了在异构架构上并行计算的基础知识，并介绍了 CUDA 的功能。其次，它解释了 CUDA 程序的执行方式。由于 CUDA 为程序员提供了操作手段和内存模型，因而 CUDA 程序员直接掌控了大规模并行环境。除了提供 CUDA 内存模型的细节之外，该书还提供了丰富的关于如何使用 CUDA 的信息。接下来的章节将讨论流，以及如何执行并行和重叠的内核。然后是关于调整指令级原语，关于如何使用 CUDA 库，以及关于使用 OpenACC 指令进行 GPU 编程的部分。在第 9 章之后，本书还提供了一些应用时的注意事项。此外，书中列举大量的范例来帮助读者更好地入门，读者可以下载并亲自运行体验。

实践证明，CUDA 在表现力和可编程性之间实现了绝佳的平衡。但是，以简化应用开

发为使命的人深知，路漫漫其修远兮。在过去的几年中，CUDA 的研究人员一直在努力改善异构编程工具。CUDA 6.0 引入了许多新功能，包括统一的内存和插件库，极大地简化了 GPU 编程。它们还提供了一组名为 OpenACC 的指令，这在书中也有所涉及。OpenACC 有望进一步完善 CUDA，它将减少在程序执行中所需的直接控制，与此同时提供一个更加便捷的方法来利用 GPU 的编程能力。目前看来，收效良好。无论是 OpenACC、CUDA 6.0，还是书中讲到的其他主题，都将帮助 CUDA 开发人员加快应用程序的运行，从而实现前所未有的高性能。请相信我，这本书值得占据你书架的一席之地！

祝大家编程愉快！

芭芭拉·查普曼

休斯敦大学计算机辅助创新设计系统与计算机系

自　序 Preface

多年前，当我们将产品代码从传统的C语言编程移植到CUDA C时，我们面临的困难同所有新手一样，这些困难远不是通过简单的上网搜索就能解决的。因此，我们便萌生了一个想法——假如能有一本由程序员为程序员编写的并且能够解决所有CUDA开发难题的书该有多好！为了满足这种需求，也利用我们在CUDA方面的经验，我们有了编写本书的念头。因此，本书是专门为应对高性能和科学计算集群的需求而编写的。

在学习新的框架或编程语言时，大多程序员只是随便摘取几段代码，测试一下，然后就在这个试验基础上编写他们自己的程序了。对许多软件开发者来说使用范例代码是一种常见的技巧。本书也遵循这一习惯。每一章重点论述一个主题，用精确的解释来提供基础知识，用简单但完全可行的代码示例来阐释每一个概念。学习概念和代码同步进行将使你迅速了解这些主题。本书使用配置文件驱动的方法来引导你逐步入门。

C语言并行编程与CUDA C语言并行编程两者间的主要区别在于CUDA的架构特性，比如内存与执行模型对于程序员而言是直接可见的。这将使你得以更好地掌控大规模并行GPU环境。尽管有些人仍然认为CUDA的概念太过基础，但其实了解一些底层架构的知识对于使用GPU来说也是非常必要的。实际上，即使你对于架构的了解有限，也完全能够用好CUDA平台。

并行编程是出于性能方面的考虑，它是靠配置文件驱动的。CUDA编程的独特优势在于开放的架构特性可以使程序员在功能强大的硬件平台上充分挖掘其计算性能，作为程序员的你将会充分感受到这一优势。在掌握书中通过练习教授的技巧之后，你将发现使用CUDA C编程是一项十分简单、有趣且高效的工作。

About the Authors 作者简介

程润伟（John Cheng）是一位有着丰富行业经验的研究员，研究方向为异构计算平台上的高性能计算。在进入石油天然气行业之前，他曾在金融行业工作了十余年，以其计算智能领域的专长，基于遗传算法做出了与数据挖掘和统计研究相关的许多方案，为商业项目中的技术难题提供了很多高明而有效的解决办法。作为享誉国际的遗传算法与工业工程应用方面的权威研究人员，他已参与编写了三本书。《遗传算法与工程设计》是他的第一部作品，该书于 1997 年由 John Wiley and Sons 公司出版，现今仍是全球高校的教科书。无论是学术研究还是实业发展方面，他都有了不错的成就，他擅长将晦涩复杂的学术问题化繁为简，深入浅出地给读者以阐释和启发。他已获东京技术研究所的计算智能博士学位。

马克斯·格罗斯曼（Max Grossman）与 GPU 编程模型已经打了近十年交道。他的主要研究内容为开发新型 GPU 编程模型和用于在 GPU 硬件上实现的算法。他已将 GPU 应用于地球科学、等离子体物理、医学成像和机器学习等一系列领域，并致力于发现和研究新领域的计算模式，寻求以全新方式应用它们的可能性。这些宝贵的实践经验对于指导今后在编程模型和程序框架方面的工作大有裨益。马克斯获莱斯大学的计算机科学学士学位，主要研究并行计算。

泰·麦克切尔（Ty McKercher）是 NVIDIA 公司的首席方案架构师，他领导的团队专攻跨行业的视觉计算系统架构。他通常负责在新兴技术评估期间促进客户和产品工程团队之间的沟通交流。自从 2006 年参加了在 NVIDIA 公司总部举办的第一届 CUDA 训练课程后，他一直参与 CUDA 的相关项目。也是自那时起，他也在某些世界最大、任务最为繁重的产品数据中心工作，帮助构建基于 GPU 的超级计算环境。他已获科罗拉多矿业学院数学学士学位，主要研究方向为地球物理学和计算机科学。

技术审校者简介 *About the Reviewers*

张伟，是一位在高性能计算领域有着 15 年工作经验的科研程序员。他已独立研发或与人合作研发了许多科学软件包，服务于分子模拟、计算机辅助药物设计、EM 的结构重建和地震深度成像开发等。目前，他正致力于研究如何采用诸如 CUDA 等新技术来提升地震数据的处理效率。

赵超，于 2008 年加入雪佛龙公司，目前是地球物理应用软件开发专家，负责为地球学家们设计和开发软件产品。他在加入雪佛龙之前是知识系统公司和地震微技术公司的软件开发员。他从事软件开发的研究和实业已超过 13 年，在地质学和地球物理学方面积累了丰富的经验。经过多年的学术研究，他十分乐于见到 CUDA 编程能真正广泛地应用到科研工作中，并愿为此尽其所能。赵超本科毕业于北京大学的化学专业，获罗得岛大学计算机科学专业的理学硕士学位。

Introduction 前　言

欢迎来到用 CUDA C 进行异构并行编程的奇妙世界！

现代的异构系统正朝一个充满无限计算可能性的未来发展。异构计算正在不断被应用到新的计算领域——从科学到数据库，再到机器学习的方方面面。编程的未来将是异构并行编程的天下！

本书将引领你通过使用 CUDA 平台、CUDA 工具包和 CUDA C 语言快速上手 GPU（图形处理单元）计算。本书中设置的范例与练习也将带你快速了解 CUDA 的专业知识，助你早日达到专业水平！

本书写给谁

本书适用于任何想要利用 GPU 计算能力来提高应用效率的人。它涵盖了 CUDA C 编程领域最前沿的技术，并有着以下突出的优势：

- 风格简洁
- 描述透彻
- 大量范例
- 优质习题
- 覆盖面广
- 内容聚焦高性能计算的需求

如果你是一个经验丰富的 C 程序员，并且想要通过学习 CUDA C 来提高高性能计算的专业才能，本书中建立在你现有知识之上的例题和习题，将使掌握 CUDA C 编程更加简单。仅需掌握一些 C 语言延伸的 CUDA 知识，你便可以从大量的并行硬件中获益。CUDA 平台、编程模型、工具和库将使得异构架构编程变得简捷且高效。

如果你是计算机科学领域以外的专业人士，而且想要通过 GPU 上的并行编程来最大限度地提高工作效率，并提高应用性能，那么本书正是为你量身打造的。书中的阐述清晰而

简明，专人精心设计的示例，使用配置文件驱动的方法，这些都将帮助你深入了解 GPU 编程并迅速掌握 CUDA。

如果你是教授或任何学科的研究者，希望通过 GPU 计算推进科学发现和创新，本书中将有你找到解决方案的捷径。即使你没有多少编程经验，在并行计算概念和计算机科学的知识方面也不够精通，本书也可带你快速入门异构架构并行编程。

如果你是 C 语言初学者并且有兴趣探索异构编程，本书也完全适合你，因为它不强制要求读者有丰富的 C 语言编程经验。即使 CUDA C 和 C 语言使用相同的语法，二者的抽象概念和底层硬件也是全然不同的，因而对其中之一的经验并不足以使你在学习另一个时感到轻松。所以，只要你对异构编程有浓厚的兴趣，只要你乐于学习新事物且乐于尝试全新的思维方式，只要你对技术相关的话题有深入探索的热情，本书也完全适合你。

即使你有不少关于 CUDA C 的经验，本书还是有助于知识更新、探索新工具以及了解最新 CUDA 功能。虽然本书旨在从零开始培养 CUDA 的专业人才，但它也含有许多先进的 CUDA 概念、工具和框架的概述，它们将对 CUDA 开发人员大有裨益。

本书的内容

本书讲解了 CUDA C 编程的基本概念与技术，用于大幅加速应用程序的性能，并包含了随着 CUDA 工具包 6.0 和 NVIDIA Kepler GPU 一起发布的最新功能。在对从同质架构到异构架构的并行编程模式转变进行了简要介绍之后，本书将引导你学习必要的 CUDA 编程技能和最佳的练习实践，包含但不仅限于 CUDA 编程模型、GPU 执行模型、GPU 内存模型、CUDA 流和事件、多 GPU 编程的相关技术、CUDA 感知 MPI 编程和 NVIDIA 开发工具。

本书采用一种独特的方法来教授 CUDA 知识，即将基础性的概念讲解与生动形象的示例相结合，这些示例使用配置文件驱动的方法来指导你实现最佳性能。我们对每一个主题都进行了详尽的讲解，清晰地展示出了采用代码示例形式详细操作的过程。书中不仅教授如何使用基于 CUDA 的工具，还介绍了如何以抽象编程模型为基础并凭借悟性与直觉对开发过程每一步骤的结果做出解释，从而帮助你快速掌握 CUDA 的开发流程。

每章围绕一个主题展开讲解，运用可行的代码示例来演示 GPU 编程的基本功能和技术，这之后就是我们精心设计的练习，以便你进一步探索加深理解。

所有的编程示例都是在装有 CUDA 5.0（或更高版本）和 Kepler 或 Fermi GPU 的 Linux 系统上运行的。由于 CUDA C 是一种跨平台的语言，因而书中的示例在其他平台上也同样适用，比如嵌入式系统、平板电脑、笔记本电脑、个人电脑、工作站以及高性能计算服务器。许多 OEM 供应商支持各种类型的 NVIDIA GPU。

本书的结构

本书共有 10 章，包含了以下主题。

第 1 章：基于 CUDA 的异构并行计算

本章首先简要介绍了使用 GPU 来完善 CPU 的异构架构，以及向异构并行编程进行的模式转变。

第 2 章：CUDA 编程模型

本章介绍了 CUDA 编程模型和 CUDA 程序的通用架构，从逻辑视角解释了在 CUDA 中的大规模并行计算：通过编程模型直观展示的两层线程层次结构。同时也探讨了线程配置启发性方法和它们对性能的影响。

第 3 章：CUDA 执行模型

本章通过研究成千上万的线程是如何在 GPU 中调度的，来探讨硬件层面的内核执行问题。解释了计算资源是如何在多粒度线程间分配的，也从硬件视角说明了它如何被用于指导内核设计，以及如何用配置文件驱动方法来开发和优化内核程序。另外，本章还结合示例阐述了 CUDA 的动态并行化和嵌套执行。

第 4 章：全局内存

本章介绍了 CUDA 内存模型，探讨全局内存数据布局，并分析了全局内存的访问模式。本章介绍了各种内存访问模式的性能表现，阐述了统一内存和 CUDA 6.0 中的新功能是如何简化 CUDA 编程的，以及如何提高程序员工作效率。

第 5 章：共享内存和常量内存

本章阐释了共享内存，即管理程序的低延迟缓存，是如何提高内核性能的。它描述了共享内存的优化数据布局，并说明了如何避免较差的性能。最后还说明了如何在相邻线程之间执行低延迟通信。

第 6 章：流和并发

本章介绍了如何使用 CUDA 流实现多内核并发执行，如何重叠通信和计算，以及不同的任务分配策略是如何影响内核间的并发的。

第 7 章：调整指令级原语

本章解释了浮点运算、标准的内部数学函数和 CUDA 原子操作的性质。它展示了如何使用相对低级别的 CUDA 原语和编译器标志来优化应用程序的性能、准确度和正确性。

第 8 章：GPU 加速库和 OpenACC

本章将介绍实现程序并行的 CUDA 专用函数库，包括线性代数、傅里叶变换和随机数生成等范例。本章还解释了 OpenACC 和基于编译器指令的 GPU 编程模型是如何利用更简

单的方法辅助 CUDA 挖掘 GPU 计算能力的。

第 9 章：多 GPU 编程

本章介绍了支持 P2P GPU 内存访问的 GPUDirect 技术，阐述了如何在多个 GPU 上管理和执行计算问题，还说明了在 GPU 加速计算集群上的大规模应用是如何利用 MPI 与 GPUDirect RDMA 来实现性能线性扩展的。

第 10 章：程序实现的注意事项

本章介绍了 CUDA 的开发过程和各种配置文件驱动的优化策略，演示了如何使用 CUDA 调试工具来调试内核和内存错误，通过案例教你如何将一个传统的 C 程序一步步移植到 CUDA C 中，以有助于加强你对于这一方法的理解，同时将此过程可视化，并验证了这些工具。

阅读本书前的准备

本书不对 GPU 作特殊要求，使用本书前也不需要你具备并行编程的相关经验，不过你最好会一些 Linux 的基本操作。运行本书各示例代码的理想机器环境是：安装有 CUDA 6.0 工具包的 Linux 系统，C/C++ 6.0 编译程序和 NVIDIA Kepler GPU。

尽管某些使用 CUDA 6.0 的示例可能需要 Kepler GPU，但大多数示例仍需在 Fermi 设备上运行。它们中的大多数可以使用 CUDA 5.5 来编译。

CUDA 工具包的下载

你可以从 https://developer.nvidia.com/cuda-toolkit 下载 CUDA 6.0 工具包。

该 CUDA 工具包包括了 NVIDIA GPU 编译器、CUDA 数学库以及用于调试和优化应用程序性能的工具，此外还有编程指南、用户手册、API 参考指南和其他文档，它们都将帮助你快速掌握 GPU 应用程序的开发。

规范

为了使你的阅读体验达到最佳，我们在书中使用了一些规范。对于新出现的术语和重要的词语，在其第一次被引入时，我们会对它们进行突出强调。文中出现的文件名、URL 地址和代码表示如下：

```
this_is_a_kernel_file.cu.
```

我们用以下方式表示代码：

```
// distributing jobs among devices
for (int i = 0; i < ngpus; i++)
{
   cudaSetDevice(i);
   cudaMemcpyAsync(d_A[i], h_A[i], iBytes, cudaMemcpyDefault,stream[i]);
   cudaMemcpyAsync(d_B[i], h_B[i], iBytes, cudaMemcpyDefault,stream[i]);
   iKernel<<<grid, block,0,stream[i]>>> (d_A[i], d_B[i], d_C[i],iSize);
   cudaMemcpyAsync(gpuRef[i], d_C[i], iBytes, cudaMemcpyDefault,stream[i]);
}
```

我们按如下方式介绍 CUDA 运行时的函数：

```
cudaError_t cudaDeviceSynchronize (void);
```

我们按如下方式显示程序输出：

```
./reduce starting reduction at device 0: Tesla M2070
     with array size 16777216  grid 32768 block 512
cpu reduce      elapsed 0.029138 sec cpu_sum: 2139353471
gpu Warmup      elapsed 0.011745 sec gpu_sum: 2139353471 <<<grid 32768 block 512>>>
gpu Neighbored  elapsed 0.011722 sec gpu_sum: 2139353471 <<<grid 32768 block 512>>>
```

我们用以下方式给出命令行指令：

```
$ nvprof --devices 0 --metrics branch efficiency ./reduce
```

源代码

在学习本书中的示例时，你可以选择手动输入所有代码或者使用附在本书上的源代码。所有本书中的源代码都可在 www.wrox.com/go/procudac 下载。在页面上，只要找到本书的标题（通过搜索框查找或直接查看书名列表），然后单击书中的详细信息页面上的代码下载链接，就可以获取书中所有的源代码。

在完成每章结尾的练习时，推荐你利用参考示例代码亲自编写代码。所有练习用的代码文件同样也可从 Wrox 网站下载。

勘误表

尽管我们已竭尽所能来避免书中出现文本或代码错误，然而事实上，难免百密一疏。如果你在阅读本书的过程中发现了如拼写或编码上的错误，并愿意与我们联系进行反馈，我们将不胜感激。若你能帮助勘误，这将会使其他读者免于在困惑中长久徘徊，同时，你也是在帮助我们给读者提供更加高质量的信息。

本书的勘误表可在 www.wrox.com/go/procudac 处找到。在本书的详细信息页面，点击图书勘误表链接即可。在这个页面上，你可以查看所有已提交并由 Wrox 的编辑发布的勘误表。

致　谢 *Acknowledgements*

假如没有同事们提供的诸多建议和建设性的批评意见，假如不是朋友们共同出谋划策和鼎力相助，很难想象，我们到底能否顺利给这次出书工作画上完美的句号。

我们在此要向 NVIDIA 公司致谢，感谢贵公司协办的历场 GTC 会议，也感谢提供给我们的 CUDA 技术文件，这些在很大程度上提升了本书的价值和权威性。

我们尤其要感谢 NVIDIA 公司的两位技术研发工程师——Paulius Micikevicius 博士和王鹏博士，本书的编写离不开他们的建议和帮助。同样，特别感谢 NVIDIA 公司的 CUDA 首席专家马克·埃伯索尔先生在本书的审查过程中给予的指导和反馈。

我们要向 NVIDIA 公司的高级产品经理威尔·雷米先生和产品营销员纳迪姆·穆罕默德先生表示感谢，感谢他们在整个过程中不渝的支持和鼓励。

感谢 NVIDIA 公司的石油天然气总监保罗·霍尔茨豪尔先生在本书初始阶段的大力支持。

我们万分感激历场 GTC 会议上的演讲人和发言者，感谢你们在 GPU 计算技术领域的工作和付出。我们已在本书的推荐阅读书目中记录了对你们的称赞。

（程润伟）我在工作中应用 GPU 已有多年，忝列专家行列。这些年来得到过的帮助和指导无数，对此我始终心存感激。特别是戴南浔博士和鲍照博士，他们在地震成像项目上的边界网关协议（BGP）的鼓励、支持和指导。同样，感谢周正振博士、张伟博士、格雷斯张女士和凯杨先生四位同事，他们都出类拔萃，同他们一起工作倍感愉悦。我们的团队同样十分出色，很荣幸能够成为其中的一员。在此，还想特别感谢我博士课程的导师也是国际知名教授三雄根博士，感谢他向我伸出橄榄枝，给予我在日本的大学里任教的宝贵机会，并邀我与他共同执笔完成学术专著，当年我在东京经营一家计算技术类的公司时，在最初的起步阶段三雄根博士曾给予我大力支持。与我一同攻克本书的还有泰和马克斯，很开心与他们合作，在撰写本书的同时，不得不说我也从他们身上学到了许多。还要向我亲爱的妻子乔莉和儿子里克说声谢谢，在过去的一年中，在我为出书工作忙得焦头烂额的许多个夜晚和周末，谢谢他们无条件的爱，谢谢他们的全心支持，谢谢他们的包容和耐心。

（泰·麦克切尔）25年来，我一直致力于同软件开发人员一起攻克高性能计算（HPC）这一艰巨任务。我很高兴能够在NVIDIA公司工作，给客户提供更多专业知识来实现GPU的大规模并行。在这里要感谢许多NVIDIA公司的工作人员，尤其是Paulius Micikevicius博士，他具有天才般的洞察力和精益求精的工作态度，纵使手头的项目十分繁重，修改本书的工作也绝不打折扣。当约翰邀请我一同编著一本书来讲解CUDA知识的时候，我欣然应允。NVIDIA公司的高级主管戴维·琼斯批准了我参与这本书的编写，但遗憾的是，戴维在与癌症病魔的斗争中没能坚持到最后。我们向戴维及他的家人深表哀思，即使戴维不幸过世，但他依然激励我们继续前行逐梦。另外，山克·特里维迪和马克·汉密尔顿不断的鼓励也始终是我们前进的动力。由于工作繁重，我还招募了马克斯加入这个项目。约翰和马克斯负责具体编写本书的内容，而我负责技术审校，审校的过程也让我从两位编者那里受益良多，这当真是一件乐事。最后，还应当感谢我的妻子朱迪和我的四个孩子，感谢他们给予我无条件的支持和爱——能够在为自己真心热爱的事业奋斗的同时得到如此的鼓励和鞭策，我是多么的幸运！

（马克斯·格罗斯曼）能够与诸多才华横溢的工程师与研究人员合作，能够得到优秀导师的不倦教诲，我感到无比幸运。首先要感谢莱斯大学的维韦克·萨卡教授和整个阿巴内罗研究小组。在那里，我开始涉足高性能计算且进入到CUDA的学术领域。正是维韦克教授与其他团队成员的殷切指导和无私帮助，引领我开启了探索神奇的学术研究世界的大门。另外还要感谢雷普索尔公司的毛里西奥·阿拉亚波罗和格莱迪斯·冈萨雷斯，也正是在他们的指导下所取得的宝贵经验，助我在参与本书编写时感到下笔有神。我相信，本书定能在科研和工程学领域提供实际有益的指导。最后，感谢约翰和泰邀请我参与本书的编写，感谢这段难忘的经历，无论是在CUDA领域还是写作和生活中都给予我无数宝贵经验。

没有技术编辑、项目编辑和审校者不可能有一本高质量的专业书籍的面世。诚挚地感谢我们的组稿编辑玛丽E. 詹姆斯，项目编辑马丁V. 明纳，文字编辑凯瑟琳·伯特和技术审校张伟、赵超。感谢这支颇具见地与才干的专业编辑团队为本书的成功出版所做的一切。与你们合作完成本书的编写着实愉悦之至。