



OpenACC for Programmers: Concepts and Strategies

OpenACC

高性能并行编程

概念与策略

[美] 苏妮塔·钱德拉塞克兰 (Sunita Chandrasekaran)

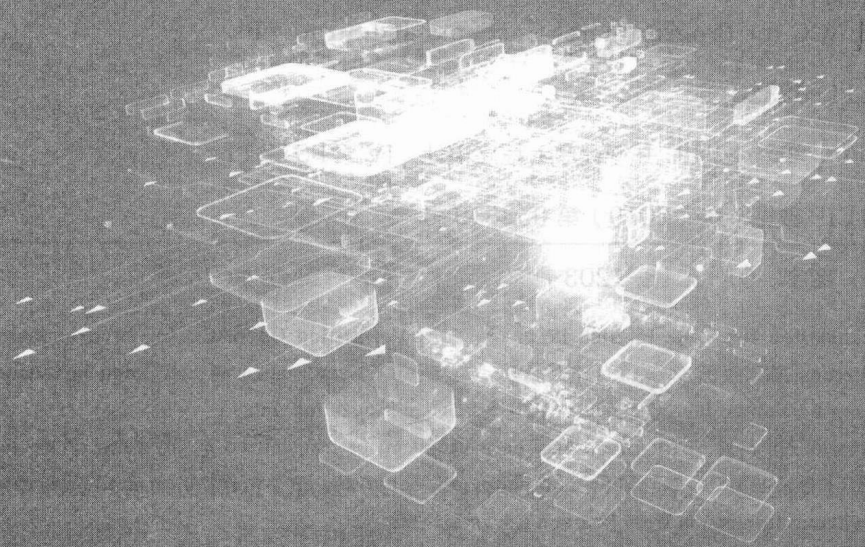
[德] 吉多·杰克兰德 (Guido Juckeland)

编

闫林 关志光 朱明星 等译



机械工业出版社
China Machine Press



OpenACC for Programmers: Concepts and Strategies



OpenACC 高性能并行编程

概念与策略

[美] 苏妮塔·钱德拉塞克兰 (Sunita Chandrasekaran)

[德] 吉多·杰克兰德 (Guido Juckeland)

编

闫林 关志光 朱明星 赵艳华 杨晓晨

徐志国 雷娟 范璟玮 李艳文

译



机械工业出版社
China Machine Press

图书在版编目 (CIP) 数据

OpenACC 高性能并行编程: 概念与策略 / (美) 苏妮塔·钱德拉塞克兰等编; 闫林等译.
—北京: 机械工业出版社, 2019.3

(高性能计算技术丛书)

书名原文: OpenACC for Programmers: Concepts and Strategies

ISBN 978-7-111-62323-6

I. O… II. ①苏… ②闫… III. 并行程序—程序设计 IV. TP311.11

中国版本图书馆 CIP 数据核字 (2019) 第 053619 号

本书版权登记号: 图字 01-2018-1203

Authorized translation from the English language edition, entitled OpenACC for Programmers: Concepts and Strategies, ISBN: 9780134694283, edited by Sunita Chandrasekaran and Guido Juckeland, published by Pearson Education, Inc., Copyright © 2018 Pearson Education, Inc.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc.

Chinese simplified language edition published by China Machine Press, Copyright © 2019.

本书中文简体字版由 Pearson Education (培生教育出版集团) 授权机械工业出版社在中华人民共和国境内 (不包括香港、澳门特别行政区及台湾地区) 独家出版发行。未经出版者书面许可, 不得以任何方式抄袭、复制或节录本书中的任何部分。

本书封底贴有 Pearson Education (培生教育出版集团) 激光防伪标签, 无标签者不得销售。

OpenACC 高性能并行编程: 概念与策略

出版发行: 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: 陈佳媛

责任校对: 殷虹

印刷: 北京诚信伟业印刷有限公司

版次: 2019 年 4 月第 1 版第 1 次印刷

开本: 186mm × 240mm 1/16

印张: 15.25

书号: ISBN 978-7-111-62323-6

定价: 79.00 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) 88379426 88361066

投稿热线: (010) 88379604

购书热线: (010) 68326294

读者信箱: hzit@hzbook.com

版权所有·侵权必究

封底无防伪标均为盗版

本书法律顾问: 北京大成律师事务所 韩光 / 邹晓东

Praise 赞誉

作为新一代的信息技术，高性能计算、云计算、物联网、新型网络等已经成为数字经济的重要支撑。OpenACC 自 2011 年推出以来，在高性能计算领域得到了广泛应用。本书由业内多名专家从各个方面对 OpenACC 进行了阐述，不仅讲述了应该怎么做，而且还深入探讨了这么做的原因。无论是对于初学者还是专业人士，本书都具有很好的参考价值。

张载龙教授

南京邮电大学

本书是介绍大规模并行编程 OpenACC 的经典书籍之一。书中阐述了 OpenACC 的基本原理、重要概念、关键技术，教授了如何使用 OpenACC 开发工具和性能检测工具。书中有基于多种场景的、完整的代码示例，也有许多案例练习。无论您是已经掌握 OpenACC 基础知识、需要提升 OpenACC 能力的专业人员，还是 OpenACC 的初学者，本书由浅入深的教学，以及它的全面性和实践性都能让您受益匪浅。

孙振鹏

最佳国际实践管理联盟主席

EXIN 国际信息科学考试学会亚太区总经理

本书将异构大规模并行编程的复杂机制用严谨而通俗的语言阐述出来，读者无须学习新的编程语言和加速器硬件知识，就能迅速掌握 OpenACC，能够实现快速开发且不会破坏原代码。本书还能帮助读者克服常见的性能、互操作性和可移植性的挑战，助力大家在多个处理器之间高效从容地分配任务。本书值得编程人员拥有！

边江

美国计算机行业协会 (ComTIA) 中国区首席代表

在云计算和大数据时代，掌握异构环境下的大规模并行编程将成为程序员必备的能力之一。本书系统介绍了 OpenACC 编程的设计思路、方法、过程、步骤、产出物和注意事项，既有理论，又有实践演练，可帮助广大读者方便地学习 OpenACC 程序是如何编译的，并系统地掌握 OpenACC 编程的最佳实践，以从容应对时代的挑战！

盛国军

海尔电器集团首席信息官

随着大数据和人工智能的场景应用日益广泛，并展示出清晰明了的价值，如何实现技术能力的高效获取变得越来越重要。建立已广泛存在的异构系统和计算能力的协同机制，最大化共享和复用价值，成为一种关键能力。本书为我们打开了一扇大门，感谢 19 位专家和中国译者们的辛勤努力，让我们有机会快速学习和实践，快速构建异构系统的大规模海量计算能力，从而更快更好地实现场景价值。

徐斌

碧桂园集团首席信息官

在计算需求几何级增长的背景下，由传统编程向大规模并行编程的转型是一个漫长、充满风险的艰辛过程。本书用浅显的语言，帮助大家掌握异构环境下的大规模并行编程原理、方法和技巧，帮助我们主动防范并降低各种风险，少走弯路。本书还通过各种编程练习，帮助大家成为真正的 OpenACC 专家。

史震

东旭集团副总裁兼首席信息官

本书是并行编程领域的一部力作。19 位非常优秀的 OpenACC 专家基于自己丰富的理论和实践经验，系统地讲授了 OpenACC 代码编程的原理、原则、语法结构和实战应用技巧及方法，帮助大家掌握编辑器、编译器、调试器和性能分析工具以提高效率。本书是本领域不可多得的佳作，值得拥有和学习！

朱怀敏

37 互娱首席技术官

本书是 OpenACC 领域影响力巨大的专著。书中对 OpenACC 是什么、为什么需要 OpenACC 以及 OpenACC 能够用来做什么都有翔实的介绍。19 位顶尖专家的真知灼见在书中随处可见。相信本书的出版将成为国内 OpenACC 生态圈中的一件盛事，并为众多 OpenACC

爱好者、研究者以及其他圈内人士提供巨大帮助。

王津银

优维科技 CEO，互联网技术专家

使用 OpenACC 无缝对接 CUDA Fortran、CUDA C/C++、cuBLAS 等成熟生态资源，可以大幅提升大规模并行编程效率，提高移植性和性能。本书对编程人员很有参考和学习价值！

熊普江

腾讯布道师

十几年前，我在攻读高性能计算博士学位的时候就已经接触和使用了 OpenMP，今天再回顾这个领域，很高兴看到过去几年中多个编程工具和框架百花齐放。特别是在当前大数据和人工智能开发应用需求非常旺盛的情况下，在如何更高效便捷和更稳定地使用 GPU 等硬件资源方面，本书深入浅出地给出了介绍，并详细阐述了使用 OpenACC 的实战经验。本人最感兴趣的部分是底层设备如何交互，在如何开发多任务并行方面，本书也有所涵盖，读后受益良多！

左鑫博士

埃森哲企业内部大数据人工智能分析专家

随着人工智能的普及，越来越多的人开始认识到利用 GPU 的大型并行计算能力是提高 AI 计算的有效方式。然而，如何让自己的代码能够有效地在 GPU 上进行并行运算，却使很多数据科学家等非计算机专业人员望而却步。OpenACC 是由克雷公司、CAPS 公司、英伟达公司和 PGI 公司共同开发的并行计算编程标准。它可大大简化并行计算的复杂度，真正使并行计算“平民化”。

本书译者非常准确地将原书内容呈现给国内广大读者，这必将对并行计算在中国的数据科学家、科研工作者、软件工程师等广大从业人员中的普及起到重要作用。

王晓光

埃森哲新技术总管

推荐序 *Foreword*

在 20 世纪,用于科学和技术编程的大多数计算机都由一个或多个通用处理器(通常称为 CPU)组成。从工资单处理到工程和科学计算,处理器都能轻松应对。处理器能够高效地执行算术运算、在内存中移动数据以及在各种操作之间自由切换。处理器是桌面和个人计算机、便携式计算机的计算引擎,它们处理各种工作负载的能力同样适用于处理文字、计算 π 的近似值、在 Web 上搜索和访问文档、播放音频文件以及维护许多不同类型的数据。计算机处理器的发展,就是一部不断追求更快速度的传奇史:为了构建能够在任何给定时间内执行更多数据操作的系统,计算机硬件制造商设计了越来越复杂的处理器。典型 CPU 的组件包括:执行简单算术和逻辑运算的算术逻辑单元(ALU),管理各种计算机组件并向 ALU 发送指令的控制单元(CU),以及运算时用于存储程序指令和数据的缓存和高速存储器。目前大多数计算机都具有多级缓存,包括少量的快速存储器到大量的慢速存储器。

应用程序开发人员和用户不断要求更强的计算能力,不管他们的目标是更真实地建立对象模型、在更短的时间内分析更多的数据,还是更快地实现高分辨率显示。例如,计算能力的增强使得天气预报取得长足进步,预报员能够预测未来几天甚至几周的天气,汽车制造商能够生产节能汽车。为了满足这一需求,计算机供应商努力缩小处理器各种组件的尺寸,以便配置更多的晶体管,晶体管是实际负责执行计算的微型设备。但是随着晶体管变得更小更密集,它们也变得越来越热。在某些时候,很明显,如果要获得更快的处理速度,就需要采用新的方法。

由此多核处理系统诞生了!在多核处理系统中,处理器的实际计算逻辑(或核)被复制成多份。每个核通常拥有自己的 ALU 和 CU,但可能与其他核共享一个或多个级别的缓存和其他内存。核与核之间能够以各种不同的方式连接,通常会共享一些硬件资源,尤其是内存。时至今日,几乎所有的便携式计算机、桌面计算机和集群都是基于多核处理器构建的。

多核处理器中的每个核都能够独立执行传统单核处理器执行的所有指令(比如加法、乘

法和分支指令)。因此,可以使用各个核同时运行不同的程序,或者将各个核协同起来提升单个应用程序的性能。在多核上并行运行的应用程序要取得实际的性能提升,取决于应用程序如何充分利用各个核的能力,以及如何有效管理核与核之间的协同。对于创建多核程序的应用程序开发人员来说,挑战层出不穷。理想的状态是,每个核都会不断地为整体结果做出贡献。为了(接近)实现这一点,工作负载需要在多个核之间均匀分配,并且可能由于一个核需要另外一个核产生的数据,因此要对工作负载进行组织,以尽量减少每个核的等待时间。最重要的是,程序员必须尽量避免大量的顺序代码,或只让一个核处于活动状态的区域。这个观点在阿姆达尔定律(Amdahl's law)中得以证实。该定律指出,无论程序的并行部分有多快,总体计算的速度提升受到顺序代码部分的限制。为了实现这一点,在某些情况下可能需要从头开始重新设计应用程序。

其他许多计算机被嵌入在电话系统、玩具、打印机和其他电子设备中,并越来越多地用于从洗衣机到冰箱等家电中。这些专用计算芯片通常用来执行某个特定功能或一组功能,并且刚好具有完成该工作所需要的硬件。一般来说,它们所能执行的也只有这些任务了。随着对实现更复杂操作的需求的不断增长,如今这些设备中有一部分也采用了专用多核处理器,使其计算能力得以增强,适用范围得以扩大。

虽然计算机游戏的概念自 20 世纪 60 年代就已出现,但家用游戏机十年之后才首次推出,直到 20 世纪 80 年代才开始普及。家用游戏机所采用的专用芯片,也是特别为它们设计的。不管是过去还是现在,游戏机设备市场都非常巨大,为此技术人员投入了大量的精力来研发处理器,这些处理器特别擅长快速构建显示在屏幕或其他输出设备上的图像。与此同时,图形处理单元(GPU)顺应游戏机市场而生,并发展成为非常强大的计算设备。游戏机专用处理器和 GPU 在满足特定用途方面,即支持计算机游戏方面,表现得非常专业,它们能够运行各种内容千差万别的游戏。换言之,它们虽然不是通用计算机,但并没有被高度优化为只能执行某个非常特殊的指令序列。尤其是 GPU,它被设计为尽可能平滑且真实地呈现一系列图像。当一个游戏场景因游戏玩家的输入而产生(高速生成并显示一系列图像)时,会涉及大量的物理知识。例如,可以模拟草的运动以确定各个叶柄在(虚拟的)风中如何摆动,并可以计算阴影效果以提供逼真的体验。因此,为游戏设计的硬件可能适用于某些类型的技术计算并不足为奇。我们很快就会看到,事实的确如此。

一些超大规模应用,例如天气预报、化学及药物、经济及金融、航空和数字电影,需要大量的计算能力。新的对硬件速度有特别要求的计算用途层出不穷。所构建的用于支持超大规模应用的系统,被称为高性能计算(High-Performance Computing, HPC)集群。高性能计算集群由一组被称为节点的计算机组成,节点之间通过高速网络连接。此类系统中的许多节

点（并非所有节点）使用与桌面系统基本相同的技术构建而成。当多核处理器进入桌面和 PC 市场后，它们也被配置为 HPC 平台的节点。目前几乎所有的 HPC 平台都具有多核节点。

几十年来，HPC 系统的开发人员和运营人员一直站在硬件革新的最前沿，他们在这—领域取得的进步构成了本书主题的背景和动机。IBM 的 Roadrunner 超级计算机（2008 年安装在美国能源部洛斯阿拉莫斯国家实验室，LANL）是首个在基准测试（Linpack TOP500，广泛用于评估运行科学应用程序代码的系统的速度）中持续性能可以达到 1 petaflop/s（每秒 1 千万亿次浮点计算）的计算系统。它的节点就是通常所说的混合架构：它们不仅在节点中引入双核处理器，还将 Cell 处理器连接到多核处理器。如此设计的理念是，Cell 处理器可以比多核更快地执行代码的某些部分。但是要在 Cell 上执行，必须专门为它编写代码，还必须将数据从多核内存传输到 Cell 内存，然后返回计算结果。由于 Cell 的可用内存非常少，这一切被证实很难完成。

工业和公共机构大型数据中心不断增长的计算服务成本，尤其是计算机的电力消耗成本，已经引起人们的关注。像 Cell 这样的专用核，有望以非常合理的运营成本为适当的应用程序代码提供更高的计算效率。具有这些特性的核越来越多地被称为加速器。在美国能源部洛斯阿拉莫斯国家实验室，实验人员遇到一个在混合节点中部署加速器的重大挑战。应用程序代码必须经过重大修改才能充分利用 Cell 技术。此外，数据和代码的传输成本必须通过代码速度提升来分摊。

Titan（2013 年安装在美国能源部橡树岭国家实验室）是一个具有里程碑意义的计算系统，运行速度高达 20 pflop/s（峰值每秒 2 亿亿次计算），拥有超过 18000 个节点，它比 Roadrunner 更强大。它的每个混合节点本身就是一个强大的计算系统，配置有 16 核 AMD 处理器和一个 NVIDIA Tesla K20 GPU。由此可见，图形处理单元已经进入高性能计算领域，通常是科学和技术计算领域。设备市场一直关注其产品功耗，GPU 能够以相对低的功耗提供特别高的性能。然而，与 Cell 处理器一样，应用程序也需要修改才能够从 GPU 中受益。因此，提供适当的编程模型以帮助进行必要的适配至关重要。为支持 Titan 用户而开发的编程模型是本书的主题。

今天，我们正处于 HPC 系统节点设计创新的时代。地球上许多最快的机器都采用了 Titan 开创的理念，因此 GPU 是最常用的硬件加速器。一些新兴系统在每个节点中使用多个 GPU，它们之间有时具有非常快的数据传输能力。在其他的研究成果中，所实施的技术能使多核 CPU 在不进行数据传输的情况下，直接与 GPU 共享内存和数据。尽管在有效使用内存方面仍然存在许多挑战，这一进步将减少一些最大、最严重的编程难题。也许更重要的是，很多小型 HPC 系统，以及桌面和便携式计算机系统，现在都配备有 GPU，用户可以成功地

将它们用于科学和技术计算。GPU 当然是为服务游戏行业而设计的，如果没有在 Titan 中的成功应用，很难想象 GPU 还能成功地用于科学和技术计算。反过来，如果没有能满足科学应用开发社区需求的可用编程模型，在科学和技术计算领域也不可能使用 GPU。

其他类型的节点架构也陆续被设计出来，它们具有类似的性能、可编程性和能源效率。特别是众核（manycore）处理的思想得到了很大的推崇。众核处理器本身就是为并行计算而设计的。换句话说，与多核平台不同，众核处理器并非用于满足通用的顺序计算需求。因此，每个核可能无法提供特别高的性能：它们提供的总体计算能力，是通过聚合大量的核并协同部署它们以解决问题。为实现这一目标，抛弃了多核硬件的一些复杂架构。这样就可以释放空间，以添加更多更简单的核。根据这个定义，GPU 实际上是基于众核思想设计的，尽管它通常以其初始目的为特征。其他硬件开发人员正在将设计背后的基本思想（大量旨在协同工作的核，同时并不期望支持应用程序的整体通用性）用于创建其他类型的众核硬件。基于不同类型的核，可能采用不同的机制来聚合众多的核。HPC 领域已经出现大量类似的系统，关于这方面的创新还在继续。

Titan 及其后续平台和其他众核系统的用户面临的最大问题都与内存有关。在 GPU 和其他众核系统中，每个核的内存容量相对较小，在大多数现有平台中，存储在多核主机平台上的数据和代码必须通过相对较慢的通信网络拷贝到 GPU。更糟糕的是，数据移动消耗大量电力，因此需要尽量减少数据移动。如上所述，最近的创新解决了这个问题，降低了创建代码的复杂性，这些代码在执行速度和功耗方面都非常高效。当前的趋势表明，HPC 中的计算节点会越来越强，并行桌面和便携式计算机也因此可能会越来越强。伴随着这个趋势的是，在配置的核类型、新型内存和内存组织以及集成组件的新策略方面，会体现出更多的异构性。尽管这些进步不会降低硬件的复杂度，但有望降低使用加速器创建高效代码的难度。它们还将增加 OpenACC 适用的系统范围。

Barbara Chapman 博士

石溪大学应用数学和统计学、计算机科学教授

布鲁克海文国家实验室数学和计算机科学主管

译者序 *The Translator's Words*

随着大数据和人工智能时代的到来，云计算成为基础设施，会像水和电一样无处不在。目前支持异构、众核的超级计算机架构已成为云计算的主流发展方向，多种支持异构并行编程的模型也应运而生，而 OpenACC 无疑是其中的佼佼者。

OpenACC 是一种由用户驱动的、基于指令的、性能可移植的高级并行编程模型。它可以方便快捷地将代码移植到各种异构 HPC 硬件平台上。OpenACC 支持 C/C++/Fortran 等多种编程语言，以及 x86、POWERCPU、XeonKNL 和 NVIDIA GPU 等多种硬件架构。OpenACC 一经问世便广受好评，在全球范围内得到快速推广。

国内许多科研单位和研发组织都在积极引入、学习和实践 OpenACC，遗憾的是目前国内相关权威书籍较少。在亚马逊网站热卖的、五星级评价的《OpenACC for Programmers: Concepts and Strategies》（即本书英文版），无疑是 OpenACC 领域最具权威的专著之一。本书汇聚了全球学术界、公共研究机构和产业界的 19 位资深专家的前沿观点，有理论、有经验教训分享、有实践案例演练、有前沿探索并且在大多数章节的结尾都提供了有针对性的练习，通过演练帮助读者深入理解对应的知识点。本书内容深入浅出，易学易懂。我们希望通过翻译此书，将 OpenACC 最先进的理念、方法、策略和技术推广给更多中国读者，包括大规模编程领域的科学家、科研工作者、软件工程师，也包括刚刚入行的大学生、研究生。

除了我之外，参与本书翻译的人员还有关志光、朱明星、赵艳华、杨晓晨、徐志国、雷娟、范璟玮、李艳文。我们都在软件研发领域有着多年理论研究和编程实践经验。为了确保译稿质量，我们追根溯源，查阅了大量国内外相关资料，译者们亲密合作，反复审校译稿，付出了极大努力！此书大部分工作都是译者们利用个人休息时间完成的，大家得到了家人的

理解和无微不至的支持，向我们的家人表示由衷的感谢！

在本书翻译的过程中我们得到了中兴通讯有限公司袁飞老师和王渊老师的鼎力帮助，在此深表感谢！同时也非常感谢华章公司关敏编辑的协调与支持！

我们的水平有限，错疏难免，读者可通过邮箱 33zzqq@163.com 反馈意见。

闫林

2018 年 12 月 10 日

前言 *Preface*

欢迎阅读本书。这本书汇聚了来自学术界、公共研究和工业界的 19 位资深作者的前沿观点。作者的共同目标是汇编一系列章节，对使用 OpenACC 进行并行编程做系统介绍。我们设计这些章节是为了提供借鉴，以便在课堂中使用。因此，亲爱的读者，你也许是一名报名参加这个可能令人生畏的并行编程课程学习的学生。请放心，这门课你选对了。计算机设备不再属于非并行类型，并行编程比以往更加重要。

本书的组织结构

我们的目标是循序渐进地介绍 OpenACC 是如何表达并行性的，以避免让你承受过多的学习压力。以下是本书的组织方式。

- 前 3 章介绍了 OpenACC 背后的概念和 OpenACC 开发工具。
- 第 4 ~ 7 章带你了解第一个真实世界的 OpenACC 程序，并揭示 OpenACC 程序编译背后的魔力，从而引入更多概念。
- 第 8 ~ 10 章涵盖高级主题，例如 OpenACC 的替代方案、底层设备交互、多设备编程和任务并行性。
- 第 11 章和第 12 章探讨了 OpenACC 实现潜在新语言特性的各种研究领域。

大多数章节在最后包含一些练习来回顾章节内容。这些解决方案以及章节中使用的代码示例可在 https://github.com/OpenACCUserGroup/openacc_concept_strategies_book 在线获得。此 URL 还提供了每章的幻灯片，供授课教师参考。

Acknowledgements 致 谢

还有许多贡献者无法在名单中一一列出，没有他们，本书将无法完成。出版这本书的想法最初来自于长期担任 OpenACC 主席的 Duncan Poole。除了在线资料之外，他还希望提供老式的印刷资料，以便学生和感兴趣的读者可以使用本书揭开 OpenACC 并行编程的神秘面纱。当 Duncan 无法进一步实现此想法时，他将火炬传递给了我们，结果就是出现在大家手中的这本书。

我们将永远感谢那些为让火焰生生不息而提供帮助的人士：

- 感谢 NVIDIA 的 Pat Brooks 和 Julia Levites，是他们帮助我们与出版社联系，并对需要内部知识的问题提供了解答。
- 感谢我们的编辑 Laura Lewin 和 Sheri Replin，产品经理 Rachel Paul，文字编辑 Betsy Hardinger，是他们帮助我们安全地穿越重重迷雾，完成了本书的创作。
- 感谢我们的章节审稿人：Mat Colgrove、Rob Faber、Kyle Friedline、Roberto Gomperts、Mark Govett、Andreas Hertel、Maxime Hugues、Max Katz、John Larson、Junjie Li、Sanhu Li、Meifeng Lin、Georgios Markomanolis、James Norris、Sergio Pino、Ludwig Schneider、Thomas Schwinge、Anne Severt 和 Peter Steinbach。

如果没有助理们为贡献者提供帮助，其中一些章节也不可能完成。在此非常感谢 Lingda Li、Masahiro Nakao、Hitoshi Murai、Mitsuhisa Sato、Akihiro Tabuchi 和 Taisuke Boku！

最后还有我们的贡献者们。在这个疯狂的旅程中，是你们与我们一起走过，从来没有让我们失望，并且按时交付内容。当然不会忘记感谢你们！

再次感谢所有人！

Sunita Chandrasekaran 和 Guido Juckeland

贡献者简介 *About the Contributors*



Randy Allen 是 Mentor Graphics 嵌入式系统部门的高级研究主管。他的职业生涯涉及研究、高级开发和以应用程序性能优化为中心的创业项目。Allen 博士为大部分 HPC 编译器项目的开发提供咨询或直接参与其中。他是 Catalytic 公司（专注于 DSP 的 MATLAB 编译）的创始人，也是 Forte Design Systems（高层次综合）的联合创始人。他撰写或参与撰写了 30 多篇与编译器有关的论文，涉及高性能计算、模拟、高层次综合和编译器优化，并参与撰写了《Optimizing Compilers for Modern Architectures》一书。Allen 博士在哈佛大学获得化学学士学位，在莱斯大学获得数学科学博士学位。



James Beyer 是 NVIDIA GPU 软件组织的一名软件工程师。他目前是 OpenMP 加速器小组委员会的联合主席，也是 OpenMP 语言委员会和 OpenACC 技术委员会的成员。在加入 NVIDIA 之前，James 是 Cray 编译器优化团队的成员。在 Cray 期间，他帮助编写了 OpenACC 初始规范。他还是 Cray OpenMP 和 OpenACC 运行时团队的成员。他在明尼苏达大学获得 CS/CE 博士学位。



Sunita Chandrasekaran 是特拉华大学生物信息学与计算生物学中心（Center for Bioinformatics & Computational Biology, CBCB）的助理教授和附属教员。她是 MIT 出版社出版的《Programming Models for Parallel Computing》，以及 2016 年 Elsevier 出版的《Parallel Programming with OpenACC》的合著者。她的研究领域包括探索高级编程模型及其语言扩展，构建编译器及运行时实现，并对实现及其与标准规范的一致性进行验证和检查。她是 OpenMP、OpenACC 和 SPEC HPG 社区的成员。Chandrasekaran 博士在新加坡南洋理工大学（NTU）因基于 FPGA 构建一个高级软件栈而获得计算机科学与工程博士学位。



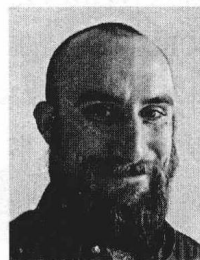
Barbara Chapman 是石溪大学应用数学和统计学、计算机科学教授，也是高级计算科学研究所（Institute for Advanced Computational Science）的一员。她还在布鲁克海文国家实验室（Brookhaven National Laboratory）主管计算机科学和数学研究。她从事并行编程接口及相关实现技术的研究已有 20 多年，并参与了多项制定并行编程社区标准的项目，包括 OpenMP、OpenACC 和 OpenSHMEM。她的团队构建了 OpenUH 编译器，该编译器支持对所提议的扩展和实现技术进行实际检验。Chapman 博士与人合著了 200 多篇论文和两本书。她在坎特伯雷大学获得数学一等荣誉学士学位，在英国贝尔法斯特女王大学获得计算机科学博士学位。



Robert Dietrich 在德累斯顿工业大学学习信息系统技术并于 2009 年毕业。作为初级研究员，他的研究重点和毕业论文是高性能计算环境下的 FPGA 编程。毕业后，作为助理研究员，他从事在诸如 Score-P 和 Vampir 等知名性能工具中支持硬件加速器和协处理器的工作。他的研究方向主要集中在可扩展异构应用程序的编程和分析。



甘霖是清华大学计算机科学与技术系博士后研究员，无锡国家超级计算中心主任助理。他的研究方向包括基于混合平台（例如 CPU、FPGA 和 GPU）的地球科学应用程序的 HPC 解决方案。甘霖拥有清华大学计算机科学博士学位，并获得 ACM Gordon Bell 奖（2016 年），清华大学-浪潮集团计算地球科学青年人才奖（2016 年），以及 FPL 2015 年度最佳论文奖。



David Gutzwiller 是 NUMECA-USA（位于加利福尼亚州旧金山）的一名软件工程师兼高性能计算负责人。David 在辛辛那提大学完成航空航天工程研究生学位后，于 2009 年加入 NUMECA。在研究生期间，他的研究课题主要集中在涡轮机械部件的自动化结构设计和优化。自从加入 NUMECA 以来，David 一直致力于改进 FINE/Turbo 和 FINE/Open CFD 求解器，以便在大规模并行、异构环境中使用。David 与行业用户合作，充分利用领先的超级计算机构建了智能驱动的设计和优化框架。



Oscar Hernandez 是橡树岭国家实验室（Oak Ridge National Laboratory）计算机科学和数学部（Computer Science and Mathematics Division）的工作人员。他致力于为 NCCS 和 OLCF 下一代领先级机器开发编程环境。他的研究主要集中在编程语言及编译器、静态分析工具、性能工具集成、并行语言（尤其是 OpenMP 和加速器导语）优化技术。他是 ORNL 在 OpenACC

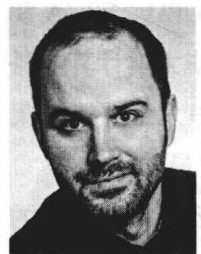
和 OpenMP ARB 标准组织中的代表，并与 SPEC/HPG 进行合作。



Adrian Jackson 是爱丁堡大学爱丁堡并行计算中心 (Edinburgh Parallel Computing Centre, EPCC) 的研究架构师。他是 EPCC 的英特尔并行计算中心 (Intel Parallel Computing Centre) 的负责人，专门从事基于巨量资源和新型计算硬件的应用程序优化。他还积极参与高性能计算的支持和培训，在 EPCC 提供的 HPC 硕士学位课程中负责 HPC 架构课程，并在英国各地举办了一系列关于并行计算各个方面的培训课程。



Guido Juckeland 刚刚在德国 HZDR (Helmholtz-Zentrum Dresden-Rossendorf) 成立了计算科学小组 (computational science group)。他负责与 HZDR 的科学家和 IT 专家一起研究 IT 工作流端到端的设计和实现。他的研究重点是提升硬件加速器、应用程序性能监控及优化的可用性和可编程性。他是 SPEC 高性能组 (High Performance Group, HPG) 的副主席，OpenACC 技术和市场委员会的活跃成员，也为 OpenMP 工具工作组做出了贡献。Guido 在德国德累斯顿工业大学因致力于基于轨迹的硬件加速器性能分析而获得计算机科学博士学位。



Jiri Kraus 在 HPC 和科学计算方面拥有 8 年以上的经验。作为 NVIDIA 的高级开发工程师，他是 GPU HPC 应用程序的性能专家。在 NVIDIA 尤利希应用实验室和动力加速及设计中心 (Power Acceleration and Design Center, PADC)，Jiri 与来自 Jülich 超级计算中心以及其他欧洲机构的开发人员及科学家合作。他的一个工作重点是多 GPU 编程模型。在加入 NVIDIA 之前，Jiri 曾在 Fraunhofer SCAI (位于圣奥古斯丁) 致力于多核 CPU 和 GPU 集群科学与技术应用程序的并行化和优化。他拥有德国科隆大学数学文凭 (辅修计算机科学)。



Jeff Larkin 是 NVIDIA 开发者技术小组的一名软件工程师，专门从事 HPC 应用程序到加速计算平台的移植和优化。此外，Jeff 参与了 OpenMP 和 OpenACC 规范的制定和推广，他还撰写了许多书籍中的章节、发表博客文章、录制视频和参与研讨会，提倡使用基于导语的并行编程。Jeff 和他的妻子、儿子住在田纳西州诺克斯维尔。在加入 NVIDIA 之前，他是橡树岭国家实验室 (ORNL) Cray 卓越超级计算中心 (Cray Supercomputing Center of Excellence) 的成员，在那里他与许多应用程序开发团队合作，其中包括两个 Gordon Bell 获奖团队。他在田纳西大学获得计算机科学硕士学位，在福尔曼大学获得计算机科学学士学位。