HZ BOOKS
华章教育

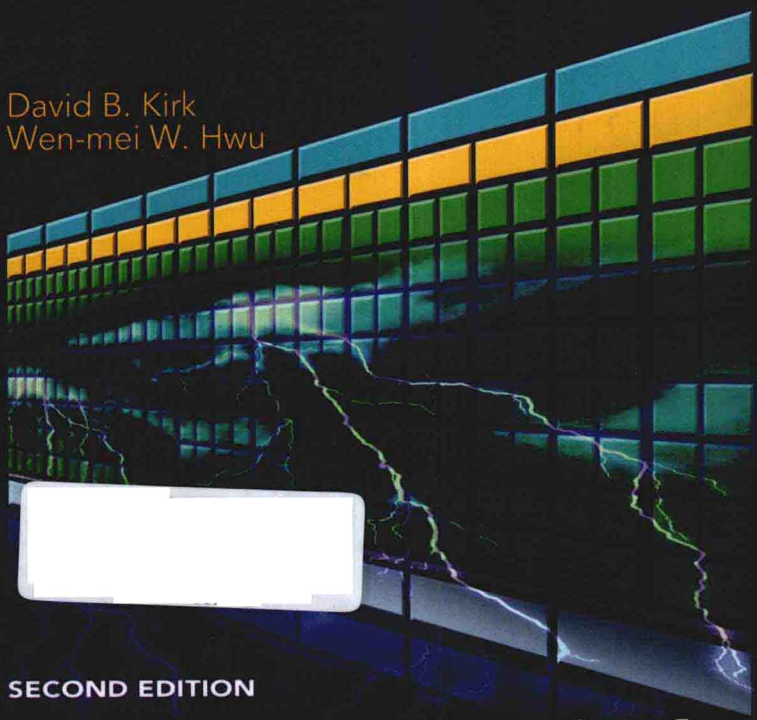
ELSEVIER
爱思唯尔

经 典 原 版 书 库

大规模并行处理器程序设计

（美） David B. Kirk Wen-mei W. Hwu 著

（英文版·第2版）



SECOND EDITION

Programming Massively Parallel Processors

A Hands-on Approach

机械工业出版社
China Machine Press

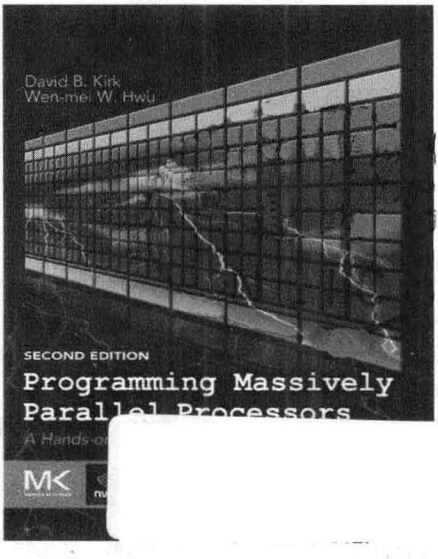
MK MORGAN KAUFMANN
NVIDIA

经 典 原 版 书 库

大规模并行处理器程序设计

（英文版·第2版）

Programming Massively Parallel Processors A Hands-on Approach (Second Edition)



（美）David B. Kirk　Wen-mei W. Hwu 著

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

大规模并行处理器程序设计（英文版·第2版）/（美）柯克（Kirk, D. B.），胡文美（Wen-mei W. Hwu）著. —北京：机械工业出版社，2013.3
（经典原版书库）
书名原文：Programming Massively Parallel Processors: A Hands-on Approach，Second Edition

ISBN 978-7-111-41629-6

I. 大… II. ① 柯… ② 胡… III. 并行程序—程序设计—英文 IV. TP311.11

中国版本图书馆 CIP 数据核字（2013）第 036145 号

本书版权登记号：图字：01-2013-0599

Elsevier (Singapore) Pte Ltd.
3 Killiney Road
#08-01 Winsland House I
Singapore 239519
Tel: (65) 6349-0200
Fax: (65) 6733-1817
First Published 2013

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：迟振春
蓟城市京瑞印刷有限公司印刷
2013 年 3 月第 1 版第 1 次印刷
170mm×242mm · 32.25 印张
标准书号：ISBN 978-7-111-41629-6
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) 88378991 88361066 投稿热线：(010) 88379604
购书热线：(010) 68326294 88379649 68995259 读者信箱：hzjsj@hzbook.com

出版者的话

文艺复兴以降，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的传统，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭示了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短的现状下，美国等发达国家在其计算机科学发展的几十年间积淀和发展的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起到积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章公司较早意识到“出版要为教育服务”。自1998年开始，我们就将工作重点放在了遴选、移译国外优秀教材上。经过多年的不懈努力，我们与Pearson，McGraw-Hill，Elsevier，MIT，John Wiley & Sons，Cengage等世界著名出版公司建立了良好的合作关系，从他们现有的数百种教材中甄选出Andrew S. Tanenbaum，Bjarne Stroustrup，Brain W. Kernighan，Dennis Ritchie，Jim Gray，Afred V. Aho，John E. Hopcroft，Jeffrey D. Ullman，Abraham Silberschatz，William Stallings，Donald E. Knuth，John L. Hennessy，Larry L. Peterson等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及珍藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力襄助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专程为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近两百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍。其影印版“经典原版书库”作为姊妹篇也被越来越多实施双语教学的学校所采用。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证。随着计算机科学与技术专业学科建设的不断完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都将步入一个新的阶段，我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

华章网站：www.hzbook.com
电子邮件：hzjsj@hzbook.com
联系电话：（010）88379604
联系地址：北京市西城区百万庄南街1号
邮政编码：100037

华章教育
华章科技图书出版中心

To Caroline, Rose, and Leo
To Sabrina, Amanda, Bryan, and Carissa
for enduring our absence while working on the course and the book

Preface

We are proud to introduce the second edition of *Programming Massively Parallel Processors: A Hands-on Approach*. Mass-market computing systems that combine multicore computer processing units (CPUs) and many-thread GPUs have brought terascale computing to laptops and petascale computing to clusters. Armed with such computing power, we are at the dawn of pervasive use of computational experiments for science, engineering, health, and business disciplines. Many will be able to achieve breakthroughs in their disciplines using computational experiments that are of an unprecedented level of scale, accuracy, controllability, and observability. This book provides a critical ingredient for the vision: teaching parallel programming to millions of graduate and undergraduate students so that computational thinking and parallel programming skills will be as pervasive as calculus.

Since the first edition came out in 2010, we have received numerous comments from our readers and instructors. Many told us about the existing features they value. Others gave us ideas about how we should expand its contents to make the book even more valuable. Furthermore, the hardware and software technology for heterogeneous parallel computing has advanced tremendously. In the hardware arena, two more generations of graphics processing unit (GPU) computing architectures, Fermi and Kepler, have been introduced since the first edition. In the software domain, CUDA 4.0 and CUDA 5.0 have allowed programmers to access the new hardware features of Fermi and Kepler. Accordingly, we added eight new chapters and completely rewrote five existing chapters.

Broadly speaking, we aim for three major improvements in the second edition while preserving the most valued features of the first edition. The first improvement is to introduce parallel programming in a more systematic way. This is done by (1) adding new Chapters 8, 9, and 10 that introduce frequently used, basic parallel algorithm patterns; (2) adding more background material to Chapters 3, 4, 5, and 6; and (3) adding a treatment of numerical stability to Chapter 7. These additions are designed to remove the assumption that students are already familiar with basic parallel programming concepts. They also help to address the desire for more examples by our readers.

The second improvement is to cover practical techniques for using joint MPI-CUDA programming in a heterogeneous computing cluster. This has been a frequently requested addition by our readers. Due to the cost-effectiveness and high throughput per watt of GPUs, many high-performance computing systems now provision GPUs in each node. The new Chapter 19 explains the conceptual framework behind the programming interfaces of these systems.

The third improvement is an introduction of new parallel programming interfaces and tools that can significantly improve the productivity of data-parallel programming. The new Chapters 15, 16, 17, and 18 introduce OpenACC, Thrust,

CUDA FORTRAN, and C++AMP. Instead of replicating the detailed descriptions of these tools from their user guides, we focus on the conceptual understanding of the programming problems that these tools are designed to solve.

While we made all these improvements, we also preserved the first edition features that seem to contribute to its popularity. First, we kept the book as concise as possible. While it is very tempting to keep adding material, we want to minimize the number of pages readers need to go through to learn all the key concepts. Second, we kept our explanations as intuitive as possible. While it is extremely tempting to formalize some of the concepts, especially when we cover the basic parallel algorithms, we strive to keep all our explanations intuitive and practical.

Target Audience

The target audience of this book is graduate and undergraduate students from all science and engineering disciplines where computational thinking and parallel programming skills are needed to achieve breakthroughs. We assume that readers have at least some basic C programming experience. We especially target computational scientists in fields such as mechanical engineering, civil engineering, electrical engineering, bio-engineering, physics, chemistry, astronomy, and geography, who use computation to further their field of research. As such, these scientists are both experts in their domain as well as programmers. The book takes the approach of building on basic C programming skills, to teach parallel programming in C. We use CUDA C, a parallel programming environment that is supported on NVIDIA GPUs and emulated on CPUs. There are more than 375 million of these processors in the hands of consumers and professionals, and more than 120,000 programmers actively using CUDA. The applications that you develop as part of the learning experience will be able to run by a very large user community.

How to Use the Book

We would like to offer some of our experience in teaching courses with this book. Since 2006, we have taught multiple types of courses: in one-semester format and in one-week intensive format. The original ECE498AL course has become a permanent course known as ECE408 or CS483 of the University of Illinois at Urbana-Champaign. We started to write up some early chapters of this book when we offered ECE498AL the second time. The first four chapters were also tested in an MIT class taught by Nicolas Pinto in the spring of 2009. Since then, we have used the book for numerous offerings of ECE408 as well as the VSCSE and PUMPS summer schools.

A Three-Phased Approach

In ECE498AL the lectures and programming assignments are balanced with each other and organized into three phases:

Phase 1: One lecture based on Chapter 3 is dedicated to teaching the basic CUDA memory/threading model, the CUDA extensions to the C language, and the basic programming/debugging tools. After the lecture, students can write a simple vector addition code in a couple of hours. This is followed by a series of four lectures that give students the *conceptual* understanding of the CUDA memory model, the CUDA thread execution model, GPU hardware performance features, and modern computer system architecture. These lectures are based on Chapters 4, 5, and 6.

Phase 2: A series of lectures covers floating-point considerations in parallel computing and common data-parallel programming patterns needed to develop a high-performance parallel application. These lectures are based on Chapters 7–10. The performance of their matrix multiplication codes increases by about 10 times through this period. The students also complete assignments on convolution, vector reduction, and prefix sum through this period.

Phase 3: Once the students have established solid CUDA programming skills, the remaining lectures cover application case studies, computational thinking, a broader range of parallel execution models, and parallel programming principles. These lectures are based on Chapters 11–20. (The voice and video recordings of these lectures are available online at the ECE408 web site: *http://courses.engr.illinois.edu/ece408/*.

Tying It All Together: The Final Project

While the lectures, labs, and chapters of this book help lay the intellectual foundation for the students, what brings the learning experience together is the final project. The final project is so important to the full-semester course that it is prominently positioned in the course and commands nearly two months' focus. It incorporates five innovative aspects: mentoring, workshop, clinic, final report, and symposium. (While much of the information about the final project is available at the ECE408 web site, we would like to offer the thinking that was behind the design of these aspects.)

Students are encouraged to base their final projects on problems that represent current challenges in the research community. To seed the process, the instructors should recruit several computational science research groups to propose problems and serve as mentors. The mentors are asked to contribute a one- to two-page project specification sheet that briefly describes the significance of the application, what the mentor would like to accomplish with the student teams on the application, the technical skills (particular type of math, physics, or chemistry courses) required to understand and work on the application, and a list of web

and traditional resources that students can draw upon for technical background, general information, and building blocks, along with specific URLs or FTP paths to particular implementations and coding examples. These project specification sheets also provide students with learning experiences in defining their own research projects later in their careers. (Several examples are available at the ECE408 course web site.)

Students are also encouraged to contact their potential mentors during their project selection process. Once the students and the mentors agree on a project, they enter into a close relationship, featuring frequent consultation and project reporting. The instructors should attempt to facilitate the collaborative relationship between students and their mentors, making it a very valuable experience for both mentors and students.

Project Workshop

The main vehicle for the whole class to contribute to each other's final project ideas is the project workshop. We usually dedicate six of the lecture slots to project workshops. The workshops are designed for students' benefit. For example, if a student has identified a project, the workshop serves as a venue to present preliminary thinking, get feedback, and recruit teammates. If a student has not identified a project, he or she can simply attend the presentations, participate in the discussions, and join one of the project teams. Students are not graded during the workshops, to keep the atmosphere nonthreatening and enable them to focus on a meaningful dialog with the instructors, teaching assistants, and the rest of the class.

The workshop schedule is designed so the instructors and teaching assistants can take some time to provide feedback to the project teams and so that students can ask questions. Presentations are limited to 10 minutes so there is time for feedback and questions during the class period. This limits the class size to about 36 presenters, assuming 90-minute lecture slots. All presentations are preloaded into a PC to control the schedule strictly and maximize feedback time. Since not all students present at the workshop, we have been able to accommodate up to 50 students in each class, with extra workshop time available as needed.

The instructors and teaching assistants must make a commitment to attend all the presentations and to give useful feedback. Students typically need the most help in answering the following questions: (1) Are the projects too big or too small for the amount of time available? (2) Is there existing work in the field that the project can benefit from? (3) Are the computations being targeted for parallel execution appropriate for the CUDA programming model?

Design Document

Once the students decide on a project and form a team, they are required to submit a design document for the project. This helps them think through the project steps before they jump into it. The ability to do such planning will be important to their later career success. The design document should discuss the background

and motivation for the project, application-level objectives and potential impact, main features of the end application, an overview of their design, an implementation plan, their performance goals, a verification plan and acceptance test, and a project schedule.

The teaching assistants hold a project clinic for final project teams during the week before the class symposium. This clinic helps ensure that students are on track and that they have identified the potential roadblocks early in the process. Student teams are asked to come to the clinic with an initial draft of the following three versions of their application: (1) the best CPU sequential code in terms of performance, with SSE2 and other optimizations that establish a strong serial base of the code for their speedup comparisons and (2) the best CUDA parallel code in terms of performance—this version is the main output of the project. This version is used by the students to characterize the parallel algorithm overhead in terms of extra computations involved.

Student teams are asked to be prepared to discuss the key ideas used in each version of the code, any floating-point numerical issues, any comparison against previous results on the application, and the potential impact on the field if they achieve tremendous speedup. From our experience, the optimal schedule for the clinic is one week before the class symposium. An earlier time typically results in less mature projects and less meaningful sessions. A later time will not give students sufficient time to revise their projects according to the feedback.

Project Report

Students are required to submit a project report on their team's key findings. Six lecture slots are combined into a whole-day class symposium. During the symposium, students use presentation slots proportional to the size of the teams. During the presentation, the students highlight the best parts of their project report for the benefit of the whole class. The presentation accounts for a significant part of students' grades. Each student must answer questions directed to him or her as individuals, so that different grades can be assigned to individuals in the same team. We have recorded these presentations for viewing by future students at the ECE408 web site. The symposium is a major opportunity for students to learn to produce a concise presentation that motivates their peers to read a full paper. After their presentation, the students also submit a full report on their final project.

Online Supplements

The lab assignments, final project guidelines, and sample project specifications are available to instructors who use this book for their classes. While this book provides the intellectual contents for these classes, the additional material will be crucial in achieving the overall education goals. We would like to invite you to

take advantage of the online material that accompanies this book, which is available at

Finally, we encourage you to submit your feedback. We would like to hear from you if you have any ideas for improving this book. We would like to know how we can improve the supplementary online material. Of course, we also like to know what you liked about the book. We look forward to hearing from you.

Acknowledgements

There are so many people who have made special contributions to the second edition. We would like to first thank the contributing authors of the new chapters. Yuan Lin and Vinod Grover wrote the original draft of the OpenACC chapter. Nathan Bell and Jared Hoberock wrote the original draft of the Thrust chapter, with additional contributions on the foundational concepts from Chris Rodrigues. Greg Ruetsch and Massimiliano Fatica wrote the original draft of the CUDA FORTRAN chapter. David Callahan wrote the C++AMP Chapter. Isaac Gelado wrote the original draft of the MPI-CUDA chapter. Brent Oster contributed to base material and code examples of the Kepler chapter. Without the expertise and contribution of these individuals, we would not have been able to cover these new programming models with the level of insight that we wanted to provide to our readers.

We would like to give special thanks to Izzat El Hajj, who tirelessly helped to verify the code examples and improved the quality of illustrations and exercises.

We would like to especially acknowledge Ian Buck, the father of CUDA and John Nickolls, the lead architect of Tesla GPU Computing Architecture. Their teams laid an excellent infrastructure for this course. John passed away while we were working on the second edition. We miss him dearly. Nadeem Mohammad organized the NVIDIA review efforts and also contributed to Appendix B. Bill Bean, Simon Green, Mark Harris, Nadeem Mohammad, Brent Oster, Peter Shirley, Eric Young and Cyril Zeller provided review comments and corrections to the manuscripts. Calisa Cole helped with cover. Nadeem's heroic efforts have been critical to the completion of this book.

We would like to especially thank Jensen Huang for providing a great amount of financial and human resources for developing the course that laid the foundation for this book. Tony Tamasi's team contributed heavily to the review and revision of the book chapters. Jensen also took the time to read the early drafts of the chapters and gave us valuable feedback. David Luebke has facilitated the GPU computing resources for the course. Jonah Alben has provided valuable insight. Michael Shebanow and Michael Garland have given guest lectures and offered materials.

John Stone and Sam Stone in Illinois contributed much of the base material for the case study and OpenCL chapters. John Stratton and Chris Rodrigues contributed some of the base material for the computational thinking chapter. I-Jui "Ray" Sung, John Stratton, Xiao-Long Wu, Nady Obeid contributed to the lab material and helped to revise the course material as they volunteered to serve as teaching assistants on top of their research. Jeremy Enos worked tirelessly to ensure that students have a stable, user-friendly GPU computing cluster to work on their lab assignments and projects.

We would like to acknowledge Dick Blahut who challenged us to create the course in Illinois. His constant reminder that we needed to write the book helped keep us going. Beth Katsinas arranged a meeting between Dick Blahut and NVIDIA Vice President Dan Vivoli. Through that gathering, Blahut was introduced to David and challenged David to come to Illinois and create the course with Wen-mei.

We would also like to thank Thom Dunning of the University of Illinois and Sharon Glotzer of the University of Michigan, Co-Directors of the multi-university Virtual School of Computational Science and Engineering, for graciously hosting the summer school version of the course. Trish Barker, Scott Lathrop, Umesh Thakkar, Tom Scavo, Andrew Schuh, and Beth McKown all helped organize the summer school. Robert Brunner, Klaus Schulten, Pratap Vanka, Brad Sutton, John Stone, Keith Thulborn, Michael Garland, Vlad Kindratenko, Naga Govindaraju, Yan Xu, Arron Shinn, and Justin Haldar contributed to the lectures and panel discussions at the summer school.

Nicolas Pinto tested the early versions of the first chapters in his MIT class and assembled an excellent set of feedback comments and corrections. Steve Lumetta and Sanjay Patel both taught versions of the course and gave us valuable feedback. John Owens graciously allowed us to use some of his slides. Tor Aamodt, Dan Connors, Tom Conte, Michael Giles, Nacho Navarro and numerous other instructors and their students worldwide have provided us with valuable feedback.

We would like to especially thank our colleagues Kurt Akeley, Al Aho, Arvind, Dick Blahut, Randy Bryant, Bob Colwell, Ed Davidson, Mike Flynn, John Hennessy, Pat Hanrahan, Nick Holonyak, Dick Karp, Kurt Keutzer, Dave Liu, Dave Kuck, Yale Patt, David Patterson, Bob Rao, Burton Smith, Jim Smith and Mateo Valero who have taken the time to share their insight with us over the years.

We are humbled by the generosity and enthusiasm of all the great people who contributed to the course and the book.

David B. Kirk and Wen-mei W.Hwu

Contents