In Praise of *Programming Massively Parallel Processors: A Hands-on Approach*

Parallel programming is about performance, for otherwise you'd write a sequential program. For those interested in learning or teaching the topic, a problem is where to find truly parallel hardware that can be dedicated to the task, for it is difficult to see interesting speedups if its shared or only modestly parallel. One answer is graphical processing units (GPUs), which can have hundreds of cores and are found in millions of desktop and laptop computers. For those interested in the GPU path to parallel enlightenment, this new book from David Kirk and Wen-mei Hwu is a godsend, as it introduces CUDA, a C-like data parallel language, and Tesla, the architecture of the current generation of NVIDIA GPUs. In addition to explaining the language and the architecture, they define the nature of data parallel problems that run well on heterogeneous CPU-GPU hardware. More concretely, two detailed case studies demonstrate speedups over CPU-only C programs of 10X to 15X for naïve CUDA code and 45X to 105X for expertly tuned versions. They conclude with a glimpse of the future by describing the next generation of data parallel languages and architectures: OpenCL and the NVIDIA Fermi GPU. This book is a valuable addition to the recently reinvigorated parallel computing literature.

David Patterson
Director, The Parallel Computing Research Laboratory
Pardee Professor of Computer Science, U.C. Berkeley
Co-author of *Computer Architecture: A Quantitative Approach*

Written by two teaching pioneers, this book is the definitive practical reference on programming massively parallel processors—a true technological gold mine. The hands-on learning included is cutting-edge, yet very readable. This is a most rewarding read for students, engineers and scientists interested in supercharging computational resources to solve today's and tomorrow's hardest problems.

Nicolas Pinto
MIT, NVIDIA Fellow 2009

I have always admired Wen-mei Hwu's and David Kirk's ability to turn complex problems into easy-to-comprehend concepts. They have done it again in this book. This joint venture of a passionate teacher and a GPU *evangelizer tackles the trade-off between the simple explanation of the concepts and the depth analysis of the programming techniques. This is a great book to learn both massive parallel programming and CUDA.*

Mateo Valero
Director, Barcelona Supercomputing Center

The use of GPUs is having a big impact in scientific computing. David Kirk and Wen-mei Hwu's new book is an important contribution towards educating our students on the ideas and techniques of programming for massively-parallel processors.

Mike Giles
Professor of Scientific Computing
University of Oxford

This book is the most comprehensive and authoritative introduction to GPU computing yet. David Kirk and Wen-mei Hwu are the pioneers in this increasingly important field, and their insights are invaluable and fascinating. This book will be the standard reference for years to come.

Hanspeter Pfister
Harvard University

This is a vital and much needed text. GPU programming is growing by leaps and bounds. This new book will be very welcomed and highly useful across inter-disciplinary fields.

Shannon Steinfadt
Kent State University

Programming Massively Parallel Processors

A Hands-on Approach

Programming Massively Parallel Processors

A Hands-on Approach

David B. Kirk and Wen-mei W. Hwu

ELSEVIER

AMSTERDAM • BOSTON • HEIDELBERG • LONDON
NEW YORK • OXFORD • PARIS • SAN DIEGO
SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO

Morgan Kaufmann Publishers is an imprint of Elsevier

MORGAN KAUFMANN

Morgan Kaufmann Publishers is an imprint of Elsevier.
30 Corporate Drive, Suite 400, Burlington, MA 01803, USA

This book is printed on acid-free paper.

Library of Congress Cataloging-in-Publication Data
Application Submitted

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library.

ISBN: 978-0-12-381472-2

For information on all Morgan Kaufmann publications,
visit our Web site at www.mkp.com or www.elsevierdirect.com

Printed in United States of America
10 11 12 13 14 5 4 3 2 1

Contents

Preface

WHY WE WROTE THIS BOOK

Mass-market computing systems that combine multicore CPUs and many-core GPUs have brought terascale computing to the laptop 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 unprecedented level of scale, 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.

We started with a course now known as ECE498AL. During the Christmas holiday of 2006, we were frantically working on the lecture slides and lab assignments. David was working the system trying to pull the early GeForce 8800 GTX GPU cards from customer shipments to Illinois, which would not succeed until a few weeks after the semester began. It also became clear that CUDA would not become public until a few weeks after the start of the semester. We had to work out the legal agreements so that we can offer the course to students under NDA for the first few weeks. We also needed to get the words out so that students would sign up since the course was not announced until after the preenrollment period.

We gave our first lecture on January 16, 2007. Everything fell into place. David commuted weekly to Urbana for the class. We had 52 students, a couple more than our capacity. We had draft slides for most of the first 10 lectures. Wen-mei's graduate student, John Stratton, graciously volunteered as the teaching assistant and set up the lab. All students signed NDA so that we can proceed with the first several lectures until CUDA became public. We recorded the lectures but did not release them on the Web until February. We had graduate students from physics, astronomy, chemistry, electrical engineering, mechanical engineering as well as computer science and computer engineering. The enthusiasm in the room made it all worthwhile.

Since then, we have taught the course three times in one-semester format and two times in one-week intensive format. The ECE498AL course has become a permanent course known as ECE408 of the University of Illinois, Urbana-Champaign. We started to write up some early chapters of this book when we offered ECE498AL the second time. We tested these chapters in our spring 2009 class and our 2009 Summer School. The first four chapters were also tested in an MIT class taught by Nicolas Pinto in spring 2009. We also shared these early chapters on the web and received valuable feedback from numerous individuals. We were encouraged by the feedback we received and decided to go for a full book. Here, we humbly present our first edition to you.

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 use pervasive terascale computing hardware to achieve breakthroughs. We assume that the reader has at least some basic C programming experience and thus are more advanced programmers, both within and outside of the field of Computer Science. We especially target computational scientists in fields such as mechanical engineering, civil engineering, electrical engineering, bioengineering, physics, and chemistry, who use computation to further their field of research. As such, these scientists are both experts in their domain as well as advanced programmers. The book takes the approach of building on basic C programming skills, to teach parallel programming in C. We use C for CUDA™, a parallel programming environment that is supported on NVIDIA GPUs, and emulated on less parallel CPUs. There are approximately 200 million of these processors in the hands of consumers and professionals, and more than 40,000 programmers actively using CUDA. The applications that you develop as part of the learning experience will be able to be run by a very large user community.

HOW TO USE THE BOOK

We would like to offer some of our experience in teaching ECE498AL using the material detailed in this book.

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 naïve parallel matrix multiplication code in a couple of hours.

Phase 2: The next phase is a series of 10 lectures that give students the *conceptual* understanding of the CUDA memory model, the CUDA threading model, GPU hardware performance features, modern computer system architecture, and the common data-parallel programming patterns needed to develop a high-performance parallel application. These lectures are based on Chapters 4 through 7. 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 scan through this period.

Phase 3: Once the students have established solid CUDA programming skills, the remaining lectures cover computational thinking, a broader range of parallel execution models, and parallel programming principles. These lectures are based on Chapters 8 through 11. (The voice and video recordings of these lectures are available on-line (http://courses.ece.illinois.edu/ece498/al).)

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 course that it is prominently positioned in the course and commands nearly 2 months' focus. It incorporates five innovative aspects: mentoring, workshop, clinic, final report, and symposium. (While much of the information about final project is available at the ECE498AL web site (http://courses.ece.illinois.edu/ece498/al), 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 recruit several major 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, 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 ECE498AL 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. We the instructors attempt to facilitate the collaborative relationship between students and their mentors, making it a very valuable experience for both mentors and students.

The 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/she can simply attend the presentations, participate in the discussions, and join one of the project teams. Students are not graded during the workshops, in order to keep the atmosphere nonthreatening and enable them to focus on a meaningful dialog with the instructor(s), teaching assistants, and the rest of the class.

The workshop schedule is designed so the instructor(s) 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 min so there is time for feedback and questions during the class period. This limits the class size to about 36 presenters, assuming 90-min lecture slots. All presentations are preloaded into a PC in order 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 instructor(s) and TAs must make a commitment to attend all the presentations and to give useful feedback. Students typically need most help in answering the following questions. First, are the projects too big or too small for the amount of time available? Second, is there existing work in the field that the project can benefit from? Third, are the computations being targeted for parallel execution appropriate for the CUDA programming model?

The 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; (2) The best CUDA parallel code in terms of performance. This version is the main output of the project; (3) A version of CPU sequential code that is based on the same algorithm as version 3, using single precision. 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 precision 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 1 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.

The 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/her as individuals, so that different grades can be assigned to individuals in the same team. 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 the Publisher's Web site www.elsevierdirect.com/9780123814722.

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

David B. Kirk and Wen-mei W. Hwu

Acknowledgments

We especially acknowledge Ian Buck, the father of CUDA and John Nickolls, the lead architect of Tesla GPU Computing Architecture. Their teams created an excellent infrastructure for this course. Ashutosh Rege and the NVIDIA DevTech team contributed to the original slides and contents used in ECE498AL course. Bill Bean, Simon Green, Mark Harris, Manju Hedge, Nadeem Mohammad, Brent Oster, Peter Shirley, Eric Young, and Cyril Zeller provided review comments and corrections to the manuscripts. Nadeem Mohammad organized the NVIDIA review efforts and also helped to plan Chapter 11 and Appendix B. Calisa Cole helped with cover. Nadeem's heroic efforts have been critical to the completion of this book.

We also thank Jensen Huang for providing a great amount of financial and human resources for developing the course. 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 contributed 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. Laurie Talkington and James Hutchinson helped to dictate early lectures that served as the base for the first five chapters. Mike Showerman helped build two generations of GPU computing clusters for the course. 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 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 also thank Thom Dunning of the University of Illinois and Sharon Glotzer of the University of Michigan, Co-Directors of the multiuniversity 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 Govindaraj, 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. Michael Giles reviewed the semi-final draft chapters in detail and identified many typos and inconsistencies.

We 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

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