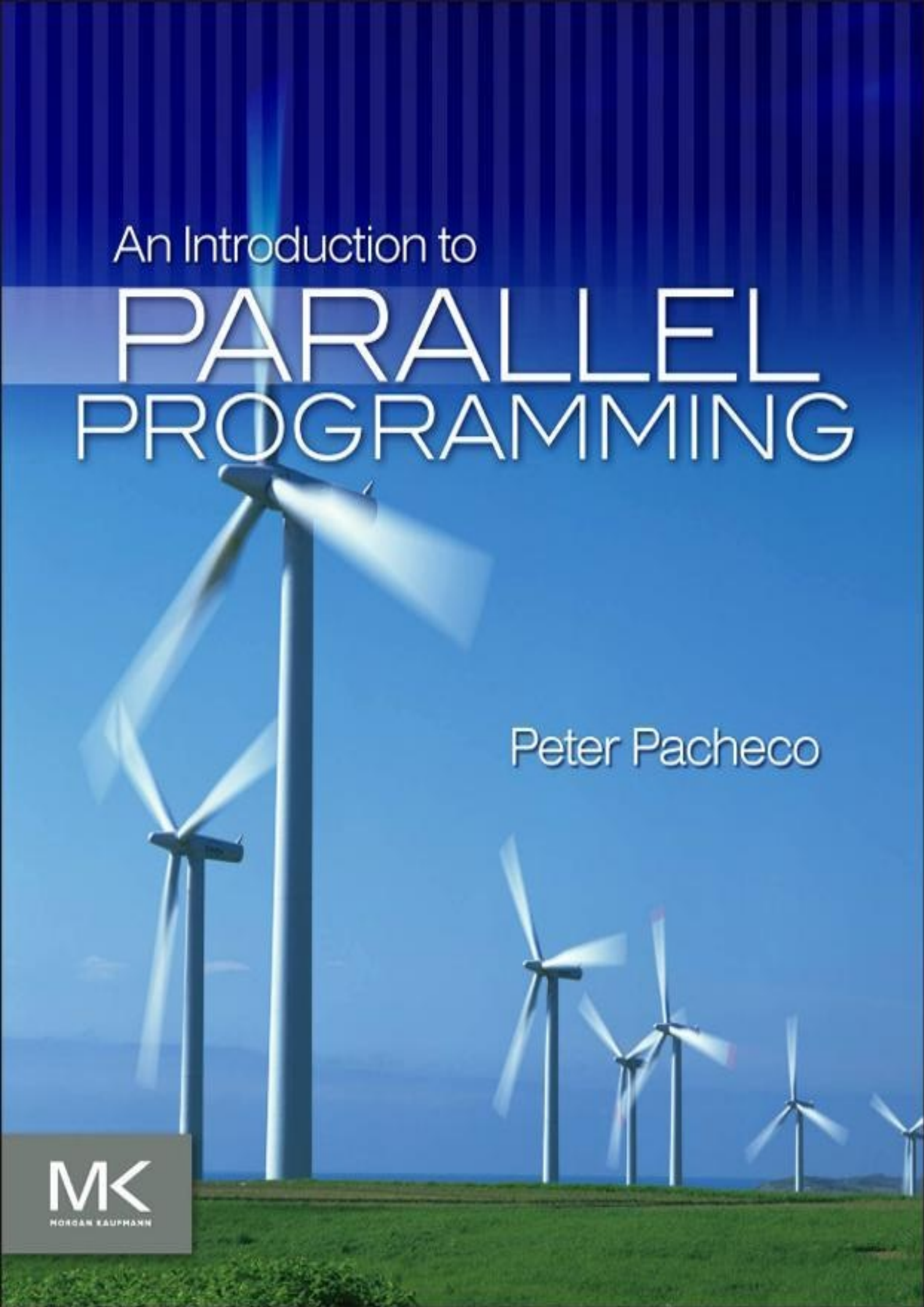

An Introduction to
PARALLEL
PROGRAMMING
Peter Pacheco
MK
MORGAN KAUFMANN

An Introduction to Parallel Programming

Peter S. Pacheco

University of San Francisco

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

ELSEVIER

Morgan Kaufmann Publishers is an imprint of Elsevier

Table of Contents

In Praise of *An Introduction to Parallel Programming*

With the coming of multicore processors and the cloud, parallel computing is most certainly not a niche area off in a corner of the computing world. Parallelism has become central to the efficient use of resources, and this new textbook by Peter Pacheco will go a long way toward introducing students early in their academic careers to both the art and practice of parallel computing.

Duncan Buell
Department of Computer Science and Engineering
University of South Carolina

An Introduction to Parallel Programming *illustrates fundamental programming principles in the increasingly important area of shared-memory programming using Pthreads and OpenMP and distributed-memory programming using MPI. More important, it emphasizes good programming practices by indicating potential performance pitfalls. These topics are presented in the context of a variety of disciplines, including computer science, physics, and mathematics. The chapters include numerous programming exercises that range from easy to very challenging. This is an ideal book for students or professionals looking to learn parallel programming skills or to refresh their knowledge.*

Leigh Little
Department of Computational Science
The College at Brockport, The State University of New York

An Introduction to Parallel Programming *is a well-written, comprehensive book on the field of parallel computing. Students and practitioners alike will appreciate the relevant, up-to-date information. Peter Pacheco's very accessible writing style, combined with numerous interesting examples, keeps the reader's attention. In a field that races forward at a dizzying pace, this book hangs on for the wild ride covering the ins and outs of parallel hardware and software.*

Kathy J. Liszka
Department of Computer Science

University of Akron

Parallel computing is the future and this book really helps introduce this complicated subject with practical and useful examples.

Andrew N. Sloss, FBCS
Consultant Engineer, ARM
Author of ARM System Developer's Guide

Recommended Reading List

For students interested in furthering their understanding of parallel programming, the content in the following books supplements this textbook:

Programming Massively Parallel Processors
A Hands-on Approach
By David B. Kirk and Wen-mei W. Hwu
ISBN: 9780123814722

The Art of Multiprocessor Programming
By Maurice Herlihy and Nir Shavit
ISBN: 9780123705914

Parallel Programming with MPI
By Peter Pacheco
ISBN: 9781558603394

The Sourcebook of Parallel Computing
Edited by Jack Dongarra et al.
ISBN: 9781558608719

Parallel Computer Architecture
A Hardware/Software Approach
By David Culler, J. P. Singh and Anoop Gupta
ISBN: 9781558603431

Engineering a Compiler, Second Edition
By Keith D. Cooper and Linda Torczon
ISBN: 9780120884780

Copyright

Acquiring Editor: Todd Green
Development Editor: Nate McFadden
Project Manager: Marilyn E. Rash
Designer: Joanne Blank

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

Notices

Knowledge and best practice in this field are constantly changing. As new research and experience broaden our understanding, changes in research methods, professional practices, or medical treatment may become necessary.

Practitioners and researchers must always rely on their own experience and knowledge in evaluating and using any information, methods, compounds, or experiments described herein. In using such information or methods they should be mindful of their own safety and the safety of others, including parties for whom they have a professional responsibility.

To the fullest extent of the law, neither the Publisher nor the authors, contributors, or editors, assume any liability for any injury and/or damage to persons or property as a matter of products

Library of Congress Cataloging-in-Publication Data

Pacheco, Peter S.

An introduction to parallel programming / Peter S. Pacheco.

p. cm.

ISBN 978-0-12-374260-5 (hardback)

1. Parallel programming (Computer science) I. Title.

QA76.642.P29 2011

005.2'75–dc22

2010039584

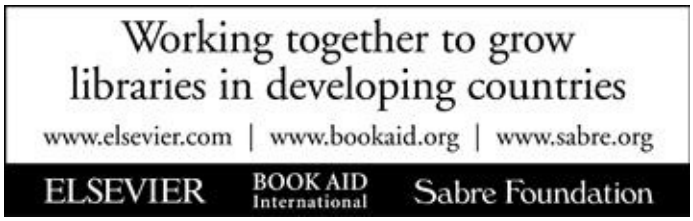
British Library Cataloguing-in-Publication Data

A catalogue record for this book is available from the British Library.

For information on all Morgan Kaufmann publications, visit our web site: *www.mkp.com* or *www.elsevierdirect.com*

Printed in the United States

12 13 14 15 10 9 8 7 6 5 4 3 2



Dedication

To the memory of Josephine F. Pacheco

Preface

Parallel hardware has been ubiquitous for some time now. It's difficult to find a laptop, desktop, or server that doesn't use a multicore processor. Beowulf clusters are nearly as common today as high-powered workstations were during the 1990s, and cloud computing could make distributed-memory systems as accessible as desktops. In spite of this, most computer science majors graduate with little or no experience in parallel programming. Many colleges and universities offer upper-division elective courses in parallel computing, but since most computer science majors have to take numerous required courses, many graduate without ever writing a multithreaded or multiprocess program.

It seems clear that this state of affairs needs to change. Although many programs can obtain satisfactory performance on a single core, computer scientists should be made aware of the potentially vast performance improvements that can be obtained with parallelism, and they should be able to exploit this potential when the need arises

An Introduction to Parallel Programming was written to partially address this problem. It provides an introduction to writing parallel programs using MPI, Pthreads, and OpenMP—three of the most widely used application programming interfaces (APIs) for parallel programming. The intended audience is students and professionals who need to write parallel programs. The prerequisites are minimal: a college-level course in mathematics and the ability to write serial programs in C. They are minimal because we believe that students should be able to start programming parallel systems *as early as possible*.

At the University of San Francisco, computer science students can fulfill a requirement for the major by taking the course, on which this text is based, immediately after taking the "Introduction to Computer Science I" course that most majors take in the first semester of their freshman year. We've been offering this course in parallel computing for six years now, and it has been our experience that there really is no reason for students to defer writing

parallel programs until their junior or senior year. To the contrary, the course is popular, and students have found that using concurrency in other courses is much easier after having taken the Introduction course.

If second-semester freshmen can learn to write parallel programs by taking a class, then motivated computing professionals should be able to learn to write parallel programs through self-study. We hope this book will prove to be a useful resource for them.

About This Book

As we noted earlier, the main purpose of the book is to teach parallel programming in MPI, Pthreads, and OpenMP to an audience with a limited background in computer science and no previous experience with parallelism. We also wanted to make it as flexible as possible so that readers who have no interest in learning one or two of the APIs can still read the remaining material with little effort. Thus, the chapters on the three APIs are largely independent of each other: they can be read in any order, and one or two of these chapters can be bypass. This independence has a cost: It was necessary to repeat some of the material in these chapters. Of course, repeated material can be simply scanned or skipped.

Readers with no prior experience with parallel computing should read Chapter 1 first. It attempts to provide a relatively nontechnical explanation of why parallel systems have come to dominate the computer landscape. The chapter also provides a short introduction to parallel systems and parallel programming.

Chapter 2 provides some technical background in computer hardware and software. Much of the material on hardware can be scanned before proceeding to the API chapters. Chapters 3, 4, and 5 are the introductions to programming with MPI, Pthreads, and OpenMP, respectively.

In Chapter 6 we develop two longer programs: a parallel *n*-body solver and a parallel tree search. Both programs are developed using all three APIs. Chapter 7 provides a brief list of pointers to additional information on various aspects of parallel computing.

We use the C programming language for developing our programs because all three APIs have C-language interfaces, and, since C is such a small language, it is a relatively easy language to learn—especially for CC++ and

Java programmers, since they are already familiar with C's control structures.

Classroom use

This text grew out of a lower-division undergraduate course at the University of San Francisco. The course fulfills a requirement for the computer science major, and it also fulfills a prerequisite for the undergraduate operating systems course. The only prerequisites for the course are either a grade of "B" or better in a one-semester introduction to computer science or a "C" or better in a two-semester introduction to computer science. The course begins with a four-week introduction to C programming. Since most students have already written Java programs, the bulk of what is covered is devoted to the use pointers in C.[1] The remainder of the course provides introductions to programming in MPI, Pthreads, and OpenMP.

We cover most of the material in Chapters 1, 3, 4, and 5, and parts of the material in Chapters 2 and 6. The background in Chapter 2 is introduced as the need arises. For example, before discussing cache coherence issues in OpenMP (Chapter 5), we cover the material on caches in Chapter 2.

The coursework consists of weekly homework assignments, five programming assignments, a couple of midterms, and a final exam. The homework usually involves writing a very short program or making a small modification to an existing program. Their purpose is to insure that students stay current with the course work and to give them hands-on experience with the ideas introduced in class. It seems likely that the existence of the assignments has been one of the principle reasons for the course's success. Most of the exercises in the text are suitable for these brief assignments.

The programming assignments are larger than the programs written for homework, but we typically give students a good deal of guidance: We'll frequently include pseudocode in the assignment and discuss some of the more difficult aspects in class. This extra guidance is often crucial: It's not difficult to give programming assignments that will take far too long for the students to complete. The results of the midterms and finals, and the enthusiastic reports of the professor who teaches operating systems, suggest that the course is actually very successful in teaching students how to write parallel programs.

For more advanced courses in parallel computing, the text and its online

support materials can serve as a supplement so that much of the information on the syntax and semantics of the three APIs can be assigned as outside reading. The text can also be used as a supplement for project-based courses and courses outside of computer science that make use of parallel computation.

Support Materials

The book's website is located at `http://www.mkp.com/pacheco`. It will include errata and links to sites with related materials. Faculty will be able to download complete lecture notes, figures from the text, and solutions to the exercises and the programming assignments. All users will be able to download the longer programs discussed in the text.

We would greatly appreciate readers letting us know of any errors they find. Please send an email to peter@usfca.edu if you do find a mistake.

---

[1] Interestingly, a number of students have said that they found the use of C pointers more difficult than MPI programming.

Acknowledgments

In the course of working on this book, I've received considerable help from many individuals. Among them I'd like to thank the reviewers who read and commented on the initial proposal: Fikret Ercal (Missouri University of Science and Technology), Dan Harvey (Southern Oregon University), Joel Hollingsworth (Elon University), Jens Mache (Lewis and Clark College), Don McLaughlin (West Virginia University), Manish Parashar (Rutgers University), Charlie Peck (Earlham College), Stephen C. Renk (North Central College), Rolfe Josef Sassenfeld (The University of Texas at El Paso), Joseph Sloan (Wofford College), Michela Taufer (University of Delaware), Pearl Wang (George Mason University), Bob Weems (University of Texas at Arlington), and Cheng-Zhong Xu (Wayne State University).

I'm also deeply grateful to the following individuals for their reviews of various chapters of the book: Duncan Buell (University of South Carolina), Matthias Gobbert (University of Maryland, Baltimore County), Krishna Kavi (University of North Texas), Hong Lin (University of Houston—Downtown), Kathy Liszka (University of Akron), Leigh Little (The State University of New York), Xinlian Liu (Hood College), Henry Tufo (University of Colorado at Boulder), Andrew Sloss (Consultant Engineer, ARM), and Gengbin Zheng (University of Illinois). Their comments and suggestions have made the book immeasurably better. Of course, I'm solely responsible for remaining errors and omissions.

Kathy Liszka is also preparing slides that can be used by faculty who adopt the text, and a former student, Jinyoung Choi, is working on preparing a solutions manual. Thanks to both of them.

The staff of Morgan Kaufmann has been very helpful throughout this project. I'm especially grateful to the developmental editor, Nate McFadden. He gave me much valuable advice, and he did a terrific job arranging for the reviews. He's also been tremendously patient with all the problems I've encountered over the past few years. Thanks also to Marilyn Rash and Megan Guiney, who have been very prompt and efficient during the production

process.

My colleagues in the computer science and mathematics departments at USF have been extremely helpful during my work on the book. I'd like to single out Professor Gregory Benson for particular thanks: his understanding of parallel computing—especially Pthreads and semaphores—has been an invaluable resource for me. I'm also very grateful to our system administrators, Alexey Fedosov and Colin Bean. They've patiently and efficiently dealt with all of the "emergencies" that cropped up while I was working on programs for the book.

I would never have been able to finish this book without the encouragement and moral support of my friends Holly Cohn, John Dean, and Robert Miller. They helped me through some very difficult times, and I'll be eternally grateful to them.

My biggest debt is to my students. They showed me what was too easy, and what was far too difficult. In short, they taught me how to teach parallel computing. My deepest thanks to all of them.

About the Author

Peter Pacheco received a PhD in mathematics from Florida State University. After completing graduate school, he became one of the first professors in UCLA's "Program in Computing," which teaches basic computer science to students at the College of Letters and Sciences there. Since leaving UCLA, he has been on the faculty of the University of San Francisco. At USF Peter has served as chair of the computer science department and is currently chair of the mathematics department.

His research is in parallel scientific computing. He has worked on the development of parallel software for circuit simulation, speech recognition, and the simulation of large networks of biologically accurate neurons. Peter has been teaching parallel computing at both the undergraduate and graduate levels for nearly twenty years. He is the author of *Parallel Programming with MPI*, published by Morgan Kaufmann Publishers.