This book is an excellent text on parallel programming in Java, especially for computational science students and faculty. If you’re short of time, simply reread the previous sentence and you’ll get the gist of this review; the rest of the following is devoted to the statement’s subtleties (including its use of italics).

As an experienced Java programmer of science and engineering applications who has authored and used many Java scientific simulations for pedagogical purposes (some of them computationally intensive), I’ve frequently asked myself the following question: “Since I have a quadcore computer, and all of my programs are sequential, do the other three cores get bored while my program runs?!” In other words, wouldn’t there be a way to parallelize my Java programs to make better use of even modestly powerful hardware?

In principle, if you create a Java program with several threads and your computer has several processors, the Java virtual machine can distribute the workload by assigning the execution of a different thread to each available processor. Concurrency utilities that can accomplish this have been included in the language since Java 5, and Java 7 takes this paradigm further by introducing a fork-join framework of thread concurrency. However, trying to build a full-fledged scientific application on these principles is simply too difficult; most scientifically-oriented programmers need a more straightforward approach.

In search of a solution, I joined a course on parallel programming that happened to be on Fortran or C programming using OpenMP and message-passing interface (MPI), and I also attended the Supercomputing 2010 conference. The main message? “If you want to do scientific parallelism, forget about Java and join us in C.” Because Java didn’t seem to be the right programming language, it looked like I’d have to revert to good old C. Then came this book!

What It Covers

The author of this book, Alan Kaminsky, is an associate professor in the Department of Computer Science at New York’s Rochester Institute of Technology (RIT). A highly experienced programmer, Kaminsky has recently designed, implemented, and tested the Parallel Java (PJ) library, which

• provides an API and middleware for creating parallel programs in pure Java;
• works on shared-memory multi-processor (SMP) parallel computers, clusters of parallel computers, and hybrid SMP clusters of parallel computers; and
• is freely available in the public domain.

My quadcore desktop is an SMP, for example, and you can build a cluster of parallel computers by assembling multiple desktop PCs and connecting them with a reasonably fast Ethernet network. If the computers in the cluster are multicore themselves, then you’ve built a hybrid cluster.

This book is heavily dependent on the PJ library and uses it to guide the reader step by step into writing parallel programs that exploit SMP, cluster, and hybrid configurations. (Neither the book nor the library address GPU programming.) Kaminsky has used his programming and teaching expertise and the object-oriented design of both Java and his library to create a highly readable and informative book.

The book is divided into five parts that cover

• the basics that everyone interested in parallel computing needs to know,
• parallel programming on SMP machines,
• cluster programming,
• programming hybrid clusters, and
• as a final bonus, three complete, fairly involved real-life examples.

These five parts span 35 short chapters, and the book ends with four appendices (the first two of which are must-reads). At the end of most chapters, the author provides references for further information, and each section is followed by a series of exercises that
you can use to reinforce the teaching or assign small student projects.

**Highs and Lows**

This book is very informative, and Kaminsky’s writing style is highly readable. Having programmed a library that implements parallelism clearly gives him deep insight into the subject. Kaminsky leads readers through a progressive process of turning sequential programs into effective and efficient parallel programs that address a variety of problems. He does this repeatedly and consistently (thus achieving persistence in students’ minds), introducing complexity as needed whenever simpler prototypes prove insufficient.

There are two especially valuable features of his approach:

- He always includes complete program listings, interspersed with comments as needed.
- He uses computational examples that hold real scientific interest for students, yet are simple enough to be easily understandable. (In my experience with parallel computing courses, this use of interesting examples is not commonplace!)

Although perhaps a bit biased toward computer science students—due most likely to his own teaching—the examples are also interesting to the scientific community: cryptography, fractal sets, Monte Carlo simulations, graph theory, cellular automata, partial differential equations, and N-body problems. These problems introduce situations that necessitate the use of various features of a parallel programming library, including patterns of parallelism, load balancing, synchronized access to shared variables, types of message-passing communications, and so on. He also covers some of the examples repeatedly in the different parts of the book to show how the same problem can be addressed using SMP, cluster, or hybrid parallelism, which is a very informative approach.

Another highlight is that, early on, Kaminsky introduces the use of metrics (and later, time models for clusters) to measure parallel program speed-up and size-up—that is, how parallelism can make your program run faster on a given problem, or successfully deal with larger problems, respectively. He then uses these metrics and time models for the various programs to show whether a given implementation of an idea will lead to the expected parallel improvement. This subsequently leads to deeper insight and better solutions.

My biggest criticism of the book stems from the same writing style I praised earlier. As I read the chapters, I frequently found the following reasoning sequence:

- A given scientific problem is stated.
- Parallel solutions are proposed and implemented.
- The solution is tested and shown not to work as expected.
- The next chapter introduces a new topic whose implementation fixes the problem.

After going through this sequence several times, I sometimes wondered: “What will be the next trick?”

Inexperienced students might get the impression that learning parallel programming consists of learning a few tricks. Also, coming to the end of the SMP part, for example, you might feel that there are more tricks that haven’t been covered. To remedy this, an instructor could provide supplemental material from a more theoretical book on parallel computing.

A second possible criticism is of a more technical nature and goes in two directions. The first points toward PJ’s C-language relatives, OpenMP and MPI. Kaminski says that the PJ library is inspired by OpenMP and MPI, but he doesn’t discuss their parallels (pun included) until the appendices. Because many readers might have previous knowledge of these well-established standards of parallelism, they could benefit from this comparison while studying the PJ approach. The second direction points toward Java itself. I was often curious to learn how the PJ library implements certain utilities (such as barriers or safe access to shared variables). You could delve into the freely available source code, but a couple of in-site explanations would have enriched the text.

This book has quickly become one of the Java programming jewels in my library. It communicates which issues are important for parallel computing, and it does so in a clear and interesting manner. It also demonstrates the elegant and natural way in which
PJ matches Java's object-oriented nature. In fact, I recommend that all serious, scientifically oriented Java programmers and students read this book as soon as possible. Given the already ubiquitous presence of desktop computers with multiple processors—and the growing availability of clusters and supercomputing facilities—this book will no doubt aid these programmers and students in building successful, multifaceted careers.

Francisco (Paco) Esquembre is an associate professor and Dean of the Faculty of Mathematics at the University of Murcia, Spain. His research interests include numerical analysis and computational methods, and the use of computer simulations and modeling for teaching science and engineering. He is an experienced (albeit sequential) programmer of computer simulations in several languages, a member of the OpenSourcePhysics project, and author of the Easy Java Simulations authoring and modeling tool. Esquembre has a PhD in mathematics from the University of Murcia. Contact him at fem@um.es.

A BREEZY LOOK AT BIOPHYSICS

By Peter Jung

While the 20th century was dubbed the century of physics, the 21st century will likely be the century of biology. With the advent of novel imaging tools with unprecedented resolutions in space and time, researchers now have direct access to cellular processes like never before, resulting in great leaps in our understanding of living-cell processes. Another important yet less visible development is biology’s transition from a descriptive to a quantitative science actively recruiting other disciplines—including physics, engineering, and computer science—in the process. In a research environment where physicists, biologists, and engineers are working jointly to understand the living state, one of the most pressing issues is how to educate students in the 21st century of biology.

When I was asked to review this book, I had two questions: How is this book different from other currently available books? What kind of student or course would best benefit from it? Before I discuss how it’s different, I want to first examine the material common to most biophysics textbooks (including this one).

The “genome” of biophysics—that is, the history of active research topics—is at the core of most textbooks. In the beginning was the nerve cell, or neuron. Nerve cells are electrically active and transmit information to other nerve cells, muscles, and so on through electric action potentials. One of the first successful and predictive biophysical models was the Hodgkin Huxley model for the generation and conduction of action potentials. Most books, including this one, contain a section on this landmark model from the 1950s, which still forms the basis of modern computational neuroscience, with some underpinning from electrochemistry and thermodynamics. Other parts of the biophysics genome included in this book are introductory sections on equilibrium thermodynamics, chemical reaction, reaction rates, energy landscapes, heat transport, diffusion, and Brownian motion.

So, what distinguishes this book from the others? At first glance, the obvious answer is weight. This book is lighter than most other books; it has approximately 270 pages, excluding appendices. This compares to 550 pages of Philip Nelson’s Biological Physics and the almost 800 pages of Physical Biology of the Cell by Rob Phillips, Jane Kondev, and Julie Theriot. Comparing Nelson’s book with this one on the next finer scale of organization, the table of contents, you can see that—apart from how it’s organized—the material is quite similar. Actually, this textbook has small sections on nonlinear dynamics, chaos, fractals, and pattern formation, which Nelson is lacking.

So why the difference in weight? The answer is detail and depth. For example, the “Nerve Conduction” chapter in the Claycomb and Tran book is approximately 15 pages; in Nelson’s book, a similar chapter is 50 pages. Comparisons of other topics yield similar results. However, this lack of detail and depth shouldn’t be seen as a detriment. Instead, this brevity of content fills a market niche,
providing a text for a light course in biophysics.

In fact, this book covers a wide range of biophysics topics, with a level of detail (including mathematical derivations) that make it well suited to a junior or senior undergraduate student, without the intimidation of 800 pages in US-letter format as in the previously mentioned book. On the other hand, the Claycomb and Tran book is no replacement for the more thorough textbooks; it’s simply too light for more inquisitive students who want a deeper understanding of the material and more detail. Furthermore, this book assumes that the reader has already taken a course in calculus-based introductory physics; in that respect, it’s unlike Russell K. Hobbie and Bradley J. Roth’s Intermediate Physics for Medicine and Biology, which assumes no knowledge of basic physics.

This is a highly suitable undergraduate textbook for a one-semester teaser course—that is, a course that gives a student a rough overview of what biophysics is all about. The text is written lucidly and provides ample problems suitable for homework assignments. I do have one suggestion for future editions: they should offer additional references to further readings. For example, chaos in the heart rhythm is covered in just a few lines, so additional references should be given for interested readers.

References

Peter Jung is a distinguished professor in the Department of Physics and Astronomy at Ohio University’s Quantitative Biology Institute. His research interests are in computational cell biology and neuroscience. Jung has a PhD in physics from University of Ulm, Germany. Contact him at jungp@ohio.edu.
IEEE
computer society

PURPOSE: The IEEE Computer Society is the world’s largest association of computing professionals and is the leading provider of technical information in the field.

MEMBERSHIP: Members receive the monthly magazine Computer, discounts, and opportunities to serve (all activities are led by volunteer members). Membership is open to all IEEE members, affiliate society members, and others interested in the computer field.

COMPUTER SOCIETY WEBSITE: www.computer.org

OMBUDSMAN: To check membership status or report a change of address, call the IEEE Member Services toll-free number, +1 800 678 4333 (US) or +1 732 981 0060 (international). Direct all other Computer Society-related questions—magazine delivery or unresolved complaints—to help@computer.org.

CHAPTERS: Regular and student chapters worldwide provide the opportunity to interact with colleagues, hear technical experts, and serve the local professional community.

AVAILABLE INFORMATION: To obtain more information on any of the following, contact Customer Service at +1 714 821 8380 or +1 800 272 6657:

- Membership applications
- Publications catalog
- Draft standards and order forms
- Technical committee list
- Technical committee application
- Chapter start-up procedures
- Student scholarship information
- Volunteer leaders/staff directory
- IEEE senior member grade application (requires 10 years
- practice and significant performance in five of those 10)

PUBLICATIONS AND ACTIVITIES

Computer: The flagship publication of the IEEE Computer Society, Computer, publishes peer-reviewed technical content that covers all aspects of computer science, computer engineering, technology, and applications.

Periodicals: The society publishes 13 magazines, 18 transactions, and one letters. Refer to membership application or request information as noted above.


Standards Working Groups: More than 150 groups produce IEEE standards used throughout the world.

Technical Committees: TCs provide professional interaction in more than 45 technical areas and directly influence computer engineering conferences and publications.

Conferences/Education: The society holds about 200 conferences each year and sponsors many educational activities, including computer science accreditation.

Certifications: The society offers two software developer credentials. For more information, visit www.computer.org/certification.

EXECUTIVE COMMITTEE

President: Sorel Reisman*
President-Elect: John W. Walz*
Past President: James D. Isakr*
VP, Standards Activities: Roger U. Fujii*
Secretary: Jon Rokne (2nd VP)*
VP, Educational Activities: Elizabeth L. Burd*
VP, Member & Geographic Activities: Rangachar Kasturi*
VP, Publications: David Alan Grier (1st VP)*
VP, Professional Activities: Paul K. Joannou*
VP, Technical & Conference Activities: Paul R. Croll*
Treasurer: James W. Moore, CSDP*
2011–2012 IEEE Division VIII Director: Susan K. (Kathy) Land, CSDP*
2010–2011 IEEE Division V Director: Michael R. Williams*
2011 IEEE Division Director V Director-Elect: James W. Moore, CSDP*

BOARD OF GOVERNORS

Term Expiring 2011: Elisa Bertino, Jose Castillo-Velazquez, George V. Cybenko, Ann DeMarie, David S. Ebert, Hironori Kasahara, Steven L. Tanimoto
Term Expiring 2012: Elizabeth L. Burd, Thomas M. Conte, Frank E. Ferrante, Jean-Luc Gaudiot, Paul K. Joannou, Luis Kun, James W. Moore
Term Expiring 2013: Pierre Bourque, Dennis J. Frailey, Atsuhiro Goto, André Ivanov, Dejan S. Milojicic, Jane Chu Prey, Charlene (Chuck) Walrad

EXECUTIVE STAFF

Executive Director: Angela R. Burgess
Associate Executive Director; Director, Governance: Anne Marie Kelly
Director, Finance & Accounting: John Miller
Director, Information Technology & Services: Ray Kahn
Director, Membership Development: Violet S. Doan
Director, Products & Services: Evan Butterfield
Director, Sales & Marketing: Dick Price

COMPUTER SOCIETY OFFICES

Phone: +1 202 371 0101 • Fax: +1 202 728 9614
Email: hq.ofc@computer.org
Los Alamitos: 10662 Los Vaqueros Circle, La Alamitos, CA 90720-1314
Phone: +1 714 821 8380
Email: help@computer.org

MEMBERSHIP & PUBLICATION ORDERS

Phone: +1 800 272 6657 • Fax: +1 714 821 4641 • Email: help@computer.org

IEEE OFFICERS

President: Moshe Kam
President-Elect: Gordon W. Day
Past President: Pedro A. Ray
Secretary: Roger D. Pollard
Treasurer: Harold L. Flescher
President, Standards Association Board of Governors: Steven M. Mills
VP, Educational Activities: Tariq S. Durrani
VP, Membership & Geographic Activities: Howard E. Michel
VP, Publication Services & Products: David A. Hodges
VP, Technical Activities: Donna L. Hudson
IEEE Division V Director: Michael R. Williams
IEEE Division VIII Director: Susan K. (Kathy) Land, CSDP
President, IEEE-USA: Ronald G. Jensen

revised 5 May 2011