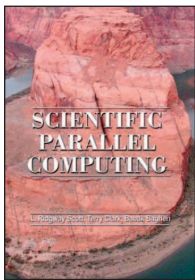




PARALLEL COMPUTING ALGORITHMS AND APPLICATIONS

By Sudarshan Raghunathan

L.R. Scott, T. Clark, and B. Bagheri, *Scientific Parallel Computing*, Princeton Univ. Press, 2005, ISBN: 978-0691119359, 416 pages.



In high-performance computing, there's often a wide chasm between low-level concepts such as memory bandwidth, cache-coherence, and nonuniform memory access, and their relevance to real-world applications, such as solving an elliptic partial differential equation with millions of degrees of freedom using a multigrid iterative solver. *Scientific Parallel Computing* is a commendable attempt at bridging the gap between these two diverse worlds in one comprehensive textbook.

The textbook consists of three sections, but the book itself isn't disjointed; references to the common theme of scientific computing applications are sprinkled throughout.

Theoretical Foundations or "What the Buzzwords Really Mean"

The first part of the book contains a systematic exposition of the fundamental concepts of parallel computing, starting with performance analysis of parallel programs, a controversial and oft-debated topic in itself, as well as practical performance constraints, such as loop dependencies and serial bottlenecks.

The authors also provide an overview of the hardware architectures commonly used for parallel computing, including different architectural

variations and their distinctions. The text then takes a brief detour into a common trick used by many languages and compilers to parallelize loops: identifying and removing the dependencies between iterations (so-called *loop-carried dependencies*). The section closes by examining the tests used to identify dependencies in an automated way, but it isn't clear to whom this technique is addressed, as it's probably most relevant to compiler writers. Moreover, it only scratches the surface: the subject is fairly complex and is dealt with in greater depth elsewhere.¹

Parallel Programming Paradigms

The book's second part covers the myriad ways in which you can write parallel programs. It focuses primarily on providing broad coverage of the techniques and how they relate to and differ from one another, but it doesn't cover most of the techniques at quite the level of depth required to write real-world applications.

The authors start by describing the two most commonly used parallel programming models—shared memory and message passing—and then the different programming issues, such as deadlocks and synchronization. Next, they detour into parallel prefix algorithms, a neat trick used to parallelize associative operators via a divide-and-conquer strategy. Although this part of the text provides several motivating examples, it feels out of place in this seg-

ment; I would have found it less disconcerting if it had appeared in the later chapters dealing with specific algorithms and applications.

A brief overview of the message-passing interface (MPI) and PThreads, which are arguably the most widely used standards, is next. But given these two standards' widespread popularity, I would have liked to see them covered at a slightly deeper level. The authors then move to IPLanguages, a programming paradigm they developed. An interesting aspect of this model is that each thread or process owns a particular location in memory; other threads and processes must explicitly request the values owned by another process. This ensures sequentially deterministic behavior while obviating the need for synchronization—a bane of shared-memory programming—and avoiding deadlocks, which plague models based on explicit message passing.

Although the authors did an admirable job of describing and comparing so many diverse programming models, standards, and languages in a 100-page span, I found a few areas in need of improvement:

- The order in which the topics are covered is a bit unclear. The text would be clearer if the authors had described all the shared-memory models and standards together, followed by a discussion of standards based on message passing, for example.
- The depth of coverage of each topic is

highly variable. If the authors' intent is simply to provide an overview of the most common parallel computing paradigms, it might be desirable to condense many of the chapters.

- The authors illustrate the programming methodologies with concise code segments, but they don't come in a ready-to-use form. The authors mention a Web site on the back cover where readers can download code, but no other references to it can be found anywhere inside the book.

However, one of the text's primary strengths is its emphasis on applications of the parallel programming models described in this part, which I detail later.

Applications

The final part of the book describes the design and parallel implementation of several scientific computing algorithms and applications. The authors start the section by dealing with the problem of parallel sparse matrix-vector multiplication. They consider the computation of the dominant eigenpair of a sparse upper-Hessenberg matrix (the Leslie matrix) using power iteration. Then, the text moves on to direct and iterative methods for solving linear systems. The direct solution of linear systems using Gaussian elimination is based on the column-wise distribution of matrices. One of this problem's most challenging aspects is the parallel solution of triangular systems because it can be a source of serial bottlenecks and load imbalance.

The iterative methods considered in this section are standard techniques, such as Jacobi and Gauss-Seidel, and the authors discuss the well-known procedure for parallelizing based on a "red-black" ordering of the unknowns. The text also briefly covers the V-cycle multigrid method, but doesn't provide

much detail about why it's particularly effective for the model problem considered or why it might not be suitable for other problems. The book describes formulation and solution approaches for nonlinear systems and initial-value problems, but it's unclear what this has to do with parallel computing because the authors merely recapitulate what can be found in any elementary textbook on ordinary and partial differential equations.²

The text then covers the parallel solution of N -body problems, a vast and very active area of research. The authors discuss several simple techniques for solving these problems, along with a detailed analysis of the associated implementation issues. Given that this particular chapter is one of the longest, I was surprised to find the briefest of mentions of fast multipole methods. Although they're very challenging to implement (particularly in parallel), multipole methods are perhaps the most popularly used techniques for particle simulations, and I hope these will be included in the book's subsequent editions.


The final application example is on parallel sorting algorithms. The authors describe data structures and algorithms specialized for sorting the positions of particles in N -body simulations.

I was very impressed with the depth and breadth of coverage of the applications and algorithms in this section, but I still have a few quibbles:

- Many of the problems discussed are fairly standard and already have several popular libraries to solve them, so it might be useful if the authors pointed these out where appropriate. For instance, ScaLapack³ could be mentioned in the context of solving dense linear systems.
- Given that most of the chapters have code fragments, it would be very use-

ful to be able to actually compile and run full examples via an accompanying Web site as discussed in the previous section. In the same regard, a lot of the presented examples use IP-Fortran, but I wasn't able to download a compiler for the IP languages.

Although the last part of the book covers the spectrum of parallel computing algorithms and applications quite well, it would have been quite useful to be able to run at least some of the examples out of the box.

Scientific Parallel Computing is a very comprehensive and well-written introductory textbook on high-performance computation that covers parallel computing fundamentals and their real-world applications in great depth. When used along with more hands-on texts,^{4,5} it can be a very effective reference to learn parallel programming's theoretical and practical aspects. 

References

1. U. Banerjee, *Loop Parallelization*, Kluwer Academic Publishers, 1994.
2. J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, Springer-Verlag, 1983.
3. L.S. Blackford et al., *ScaLapack Users' Guide*, SIAM Press, 1997.
4. K. Dowd and C. Severance, *High Performance Computing*, O'Reilly, 1998.
5. P. Pacheco, *Parallel Programming with MPI*, Morgan Kaufmann, 1996.

Sudarshan Raghunathan is a member of the technical staff at Interactive Supercomputing. His research interests include high-performance computing, multilevel iterative solvers, and adaptive methods for solving partial differential equations. Raghunathan has a PhD in computational engineering from the Massachusetts Institute of Technology. Contact him at rsudarshan@interactive.supercomputing.com.