Parallel programming represents the next turning point in how software engineers write software. Today, low-cost multicore processors are widely available for both desktop computers and laptops. As a result, applications will increasingly need to be parallelized to fully exploit the multicore-processor-throughput gains that are becoming available. Unfortunately, writing parallel code is more complex than writing serial code. This is where the OpenMP programming model enters the parallel computing picture. OpenMP helps developers create multithreaded applications more easily while retaining the look and feel of serial programming.

OpenMP simplifies the complex task of code parallelization, letting even beginners move gradually from serial programming styles to parallel programming. OpenMP extends serial code by using compiler directives. A programmer familiar with a language (such as C/C++) needs to learn only a small set of directives. Adding them doesn’t change the serial code’s logical behavior. It tells the compiler only which piece of code to parallelize and how to do it, and the compiler handles the entire multithreaded task.

Using OpenMP, by Barbara Chapman, Gabriele Jost, and Ruud van der Pas, appears exactly a decade after the release of the first OpenMP specification. The book has two main parts: chapters 1 through 4 introduce the OpenMP programming model, and the remaining five chapters cover advanced topics in OpenMP.

Chapter 1 describes contemporary parallel computer architectures, explains how OpenMP uses their parallelism, and compares OpenMP to the message-passing interface and Pthreads programming models using a simple example. Chapter 2 gives an overview of the OpenMP execution model, its memory model, OpenMP programming paradigms, and issues related to OpenMP programs’ correctness and performance. Chapter 3 demonstrates in detail how to write an OpenMP program, using as an example the parallelization of matrix-vector multiplication in C and Fortran. Chapter 4 introduces OpenMP directives and clauses and includes a detailed example with an explanation of each construct.

Chapter 5 gives an overview of performance issues related to OpenMP programming and explains how to overcome these problems. The main topics are loop optimizations, how to measure OpenMP performance, and how to measure OpenMP directive overheads. The authors illustrate these topics using matrix-vector multiplication as a case study. Chapter 6 uses case studies such as a CFD (computational fluid dynamics) flow solver and NAS (NASA Advanced Supercomputing) parallel benchmarks to demonstrate how to use OpenMP on large-scale parallel machines. This is followed by a detailed discussion of how to analyze OpenMP programs’ performance using state-of-the-art profiling.
tools. Chapter 7 presents common programming errors and explains how to debug OpenMP programs. The main issues the authors discuss and demonstrate are data race conditions, thread safety, memory consistency problems, and deadlock situations. Chapter 8 describes how OpenMP is implemented and integrated into existing host compilers and discusses how these implementations affect the performance of OpenMP programs. Chapter 9 offers a glimpse of OpenMP's future.

**Using OpenMP** is a well-organized text with plenty of supportive examples and illustrations. The references are recent and come from highly visible conferences and journals in the discipline. The book is an important reference tool and is suitable for use as a tutorial for novices, as a self-study book, or as technical training material for professionals.

**Ami Marowka** is an assistant professor in the Department of Software Engineering at the Shenkar College of Engineering and Design. Contact him at amimar2@yahoo.com.

**Cite this article:**