During the past decade, parallel computing has entered the mainstream of information technologies. The use of parallel hardware is no longer restricted to traditional supercomputing applications, but spans a broad range of problems from the design of commercial packaging to the analysis of portfolios for financial institutions. Of the different kinds of parallel architecture marketed today, the most widely deployed is probably the symmetric multiprocessor, or SMP, in which two or more processors share a common memory.

Systems of this kind were first built during the 1980s. At that time, each vendor developed a custom set of language extensions that permitted the development and execution of applications, mostly written in Fortran, on their own range of hardware. Early attempts to provide a uniform programming model across the different systems did not come to fruition, and the shared memory machines were increasingly replaced by a new breed of parallel architecture, in which the processors no longer shared memory. Ever since such distributed memory systems have become widely used, the relative merits of shared memory and distributed memory programming models have been hotly debated. This debate continues today.

In the meantime, however, much has changed with regard to SMPs and their programmability. First, there has been considerable convergence in how multiprocessors with shared memory are designed and built. Second, the popularity of this kind of architecture created a mass market ripe for the take-up of a variety of commercial parallel applications. All that was needed to encourage the development of appropriate codes was a standard application programming interface (API) for SMPs.

Late in 1997, just such an API appeared. OpenMP version 1.0 for Fortran 77 was announced by an influential consortium of vendors in October 1997. A year later, the corresponding API for C and C++ was released. OpenMP was based upon familiar shared memory parallel programming concepts, and it soon gained widespread recognition. Just two years later, it is supported by all the major vendors of hardware and software of parallel computing.

This special issue contains 11 articles based on presentations of OpenMP-related work at the first European Workshop on OpenMP (EWOMP’99) held in Lund, Sweden, September 30–October 1, 1999. EWOMP ’99 was intended to serve as a forum for the exchange of experiences and information related to the use of OpenMP in high-performance computing. It was attended by researchers and practitioners.
from academia as well as industry, and the topics discussed ranged from OpenMP application development strategies, via tool support and benchmarking efforts, to proposals for extensions to the OpenMP language. The 15 contributed presentations were complemented by five invited talks that highlighted the role of OpenMP in industry.

We have divided the 11 articles into five different subject areas—sometimes with overlapping borders—presented below.

(1) **Application experiences**

In *Development and Performance of a Mixed OpenMP/MPI Quantum Monte Carlo Code*, by Smith and Kent, the authors present their experiences in porting an existing MPI application to a mixed OpenMP/MPI version that would be better suited for future SMP clusters. Gebremedhin and Manne use OpenMP and an SGI Origin 2000 in *Scalable Parallel Graph Coloring Algorithms* to validate theoretical results on parallel algorithm speedup. The last article in this section, *HPF to OpenMP on the Origin2000: A Case Study*, by Brieger, presents the author’s experiences when porting an existing HPF application to OpenMP.

(2) **Comparisons with other approaches**

OpenMP sometimes directly competes with other models for parallel programming and in *SPMD OpenMP vs MPI for Ocean Models*, by Wallcraft, the author compares an SPMD coding style of OpenMP with MPI. In *OpenMP versus Threading in C/C++*, by Kuhn, Petersen and O’Toole, the authors compare OpenMP with another approach using shared memory, Pthreads.

(3) **Tools and implementations**

Even though it is argued that the shared memory programming paradigm of OpenMP is easier to use than message-passing, there are still many problems that can arise, especially with respect to performance. Adhianto *et al.* present, in *Tools for OpenMP Application Development: The POST Project*, an infrastructure to develop OpenMP applications with a main emphasis on tool support in the programming process. In *OdinMP/CCp—A Portable Implementation of OpenMP for C*, by Brunschen and Brorsson, a simple and portable implementation of OpenMP for C is presented.

(4) **Language extensions**

OpenMP is not static. The architecture review board actively encourages performance studies that demonstrate the usefulness of new language constructs. In *NanosCompiler: Supporting Flexible Multilevel Parallelism Exploitation in OpenMP*, by González *et al.*, the authors use an experimental compiler infrastructure to evaluate the possibility of using multilevel parallelism. Similarly, in *Flexible Control Structures for Parallelism in OpenMP*, by Shah *et al.*, the authors propose work queues as a new worksharing construct.

(5) **Performance evaluations**

The final two papers deal with performance evaluation. In *Porting and Performance Evaluation of Irregular Codes using OpenMP*, by Hisley *et al.*, the authors evaluate the suitability of OpenMP for irregular applications. Berrendorf and Niecken present, in *Performance Characteristics for OpenMP Constructs on Different Parallel Computer Architectures*, a performance evaluation of the OpenMP compiler and run-time system itself on a range of platforms.
We are proud to be able to present such a diverse cross-section of the activities that are taking place with and around OpenMP. We would like to thank all the individuals involved in making this special issue, especially all the authors and the referees that helped us in maintaining the high quality.

MATS BRORSSON  
Royal Institute of Technology, Sweden

BARBARA CHAPMAN  
University of Houston, Texas and  
University of Southhampton, U.K.

Copyright © 2000 John Wiley & Sons, Ltd.  