Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®
Parallel Computing 29 (2003) 1505–1508

PARALLEL COMPUTING

www.elsevier.com/locate/parco

Guest editorial

Parallel and distributed scientific and engineering computing

The scientific and engineering computing domains play a key role in shaping future research and development activities in academia and industry. In the not too distant future, every researcher in science and engineering fields will have to understand parallel and distributed computing. With hyperthreading in Intel processors, hypertransport links in next generation AMD processors, multicore silicon in today's high-end microprocessors from IBM, emerging cluster and grid computing, parallel (and distributed) computing have moved into the mainstream of computing. To fully exploit these advances, researchers must start to write parallel or distributed scientific and engineering software and algorithms to cope with large and complex problems with very tight timing schedules.

We would like to use this special issue to report the recent important advances in the area of parallel and distributed scientific and engineering computing. There were 33 paper submissions, not only from the Asian Pacific, but also from Europe and North America. All submissions were reviewed by at least three reviewers on relevance and technical contents on basis of papers. It was extremely difficult to select the presentation in the special issue because there were many excellent and interesting submissions. In order to allocate as many papers as possible and keep the high quality of the special issue, we finally decided to accept 13 papers. We believe all of these papers and topics will not only provide novel ideas, new results, work in progress and state-of-the-art techniques in this field, but also stimulate the future research activities in the area of parallel and distributed computing for science and engineering applications.

This special issue is mainly divided into three sections, namely programming and system support, advanced numerical computation and high performance applications, respectively.

Part 1: Programming and system support

In the first paper, Iwamoto et al. propose and evaluate the receiving message prediction method for high performance message passing. This method is independent of underlying computer architecture and message passing libraries. They also

0167-8191/$ - see front matter
doi:10.1016/j.parco.2003.08.001

propose the algorithms for the message prediction, and evaluate them from the viewpoint of the success ratio and speed-ups. The application of the method to the MPI libraries achieves a speed-up of 6.8% for the NAS Parallel Benchmarks, and the static and dynamic selection of prediction methods based on profiling results improve the performance.

Sun et al. present a distributed object model called MOIDE (multithreading object-oriented infrastructure on distributed environment) for solving irregularly structured problems in the second paper. The model creates an adaptive computing infrastructure for developing and executing irregular applications on distributed systems. A runtime system is developed to implement MOIDE-based computing. Applications including N-body problem, ray tracing, and conjugate gradient are developed to demonstrate the advantages of the model.

The popularity of Java and recent advances in compilation and execution technology for Java are making the language one of the preferred ones in the field of high performance scientific and engineering computing. A framework to characterize object access patterns, along three orthogonal dimensions of distributed Java Virtual Machine is proposed by Fang et al. in the third paper. Several benchmark applications have been tested on their distributed JVM. The authors report the performance results and give an in-depth analysis of the effects of the proposed adaptive solutions as well.

Chan et al. in the fourth paper describe the graph-oriented programming (GOP) model and environment for building and evaluating parallel applications. The GOP model provides higher level abstractions for message-passing parallel programming and the software environment offers tools which can make it easier for programmers to parallelize, write, and deploy scientific and engineering computing applications. The authors also describe the evaluation of the environment implemented on top of MPI with a sample parallel scientific application program.

In the last paper of the section, Shen et al. present a Multi-Storage I/O System (MS-I/O) that can not only effectively manage various distributed storage resources in the system, but also provides novel high performance storage access schemes. Additionally the authors present a User Access Pattern data structure which is associated with each dataset that can help MS-I/O easily make accurate I/O optimization decisions.

Part 2: Advanced numerical computation

One of the fundamental tasks of numerical computing is to solve large and sparse linear systems. The first two papers in this section deal with this problem. In the first paper, Amestoy et al. consider the direct solution of general sparse linear systems based on a multifrontal method. The approach combines partial static scheduling of the task dependency graph during the symbolic factorization and distributed dynamic scheduling during the numerical factorization to balance the work among the processors of a distributed memory computer. The performance analysis on an IBM SP3 with 16 processors per SMP node and up to 128 processors shows that they can significantly reduce both the amount of inter-node communication and the solution time.

In another paper, Leo Chin Sim et al. propose a new high-speed computation algorithm for solving a large matrix system using the Gauss-LU algorithm on a MIMD-SIMD Hybrid System consisting of a combination of cluster of workstations (COWs) and SIMD systems working concurrently to produce an optimal parallel computation. This algorithm basically performs the "divide and conquer" approach by breaking down the large $N \times N$ matrix system into manageable 32×32 matrices for fast computation.

Shen and Zhang present a fully parallel algorithm for constructing a block independent set for general sparse matrices in a distributed environment in the third paper of the section. The block independent set is used in the construction of parallel multilevel preconditioners in solving large sparse matrices on distributed memory parallel computers. The authors compare a few implementations of the parallel multilevel ILU preconditioners with different block independent set construction strategies. Numerical experiments indicate that the parallel block independent set algorithm is effective in reducing both the parallel multilevel preconditioner construction time and the size of the last level reduced system.

A parallel library of efficient algorithms for model reduction of large-scale systems is discussed by Benner et al. in the fourth paper. The authors survey the numerical algorithms underlying the implementation of the chosen model reduction methods, then employ Newton-type iterative algorithms for the solution of the major computational tasks. Experimental results report the numerical accuracy and the parallel performance of the proposed algorithms on a cluster of Intel Pentium II processors.

The last paper by Chen et al. describes the context, design, and recent development of the LAPACK for clusters (LFC) project. It has been developed in the framework of Self-Adapting Numerical Software (SANS) since such an approach can deliver the convenience and ease of use of existing sequential environments bundled with the power and versatility of highly tuned parallel codes that execute on clusters. Accomplishing this task is far from trivial as the authors argue in the paper by presenting pertinent case studies and possible usage scenarios.

Part 3: High performance applications

In this section, there are three papers to address high performance applications in image compositing, air quality simulation and computational fluid dynamics, respectively.

The first paper by Takeuchi et al. presents an improvement on the binary-swap (BS) method, an efficient image compositing algorithm for sort-last parallel rendering. The proposed compositing method uses three acceleration techniques, namely the interleaved splitting, multiple bounding rectangle, and run-length encoding to balance the compositing workload among processors, to better exploit sparsity of the image, and to reduce the cost of communication.

The aim of the second paper by Martín et al. is to provide a high performance air quality simulation using the STEM-II program, a large-scale pollution modeling application. Performance results are presented for a SGI Origin2000 multiprocessor,

a Fujitsu AP3000 multicomputer and a cluster of PCs. Experimental results show that the parallel versions of the code achieve important reductions in the CPU time needed by each simulation, allowing users to obtain results with adequate speed and reliability for the industrial environment.

The last paper analyzes the effects of various performance enhancement strategies on the parallel efficiency of an overset grid Navier–Stokes CFD application running on an SGI Origin2000 machine. Specifically, the roles of asynchronous communication, grid splitting, and grid grouping strategies are presented and discussed by Djomehri et al. Details of a sophisticated graph partitioning technique for grid grouping are also provided. Results indicate that performance depends critically on the level of latency hiding and the quality of load balancing across the processors.

Acknowledgements

Of course, the proposed division into sections is one of the possible partitioning strategies. The represented areas, as well, are not an exhaustive representation of the world of current high performance scientific and engineering computing. Nonetheless, they represent the rich and many-faceted knowledge, that we have the pleasure of sharing with the readers. We would like to thank the authors for their excellent contributions and patience in assisting us. Finally, the fundamental work of all reviewers on these papers is also very warmly acknowledged.

Laurence T. Yang *
Department of Computer Science
St. Francis Xavier University
P.O. Box 5000
Antigonish, NS
Canada B2G 2W5
Tel.: +1-902-867-5546; fax: +1-902-867-2448
E-mail: lyang@stfx.ca

Yi Pan
Department of Computer Science
Georgia State University
Atlanta, GA 30303
USA
E-mail: pan@cs.gsu.edu

Minyi Guo
Department of Computer Software, University of Aizu
Aizu-Wakamatsu City, Fukushima 965-8580
Japan
E-mail: minyi@u-aizu.ac.jp

* Corresponding author.