Journal of Parallel and Distributed Computing **59**, 101–106 (1999)
Article ID jpdc.1999.1579, available online at http://www.idealibrary.com on IDEAL®

Special Issue on Software Support for Distributed Computing

Guest Editors' Introduction

Ishfaq Ahmad

Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong

and

Francis Lau

Department of Computer Science and Information Systems, The University of Hong Kong, Hong Kong

Rapid advances in communications technology and the proliferation of inexpensive PCs and workstations have created a wide avenue for distributed computing to move into mainstream computing. Distributed computing is no longer confined to research laboratories but is available to the mass community of computer users. The shift of status would not have been possible if it were not for the immense effort that researchers and developers have put into creating the necessary supporting software—networking protocols, distributed operating systems, network file systems, remote procedure calls, message passing interfaces, and most recently, object-oriented middleware. The advent of the Internet and the Web has also added fuel to the shift. The Internet as one gigantic distributed system is no longer a far-fetched idea of the visionaries; it is now being used by some to try to solve really tough problems—to search for extraterrestrial intelligence [1], to find the largest prime number [2], and to crack the RC5 block cipher [3], just to name a few. Many wide-area distributed systems that are in the making, the Globe system [6] for example, have worldwide scalability as one of their main goals.

But distributed computing is not about just IP and HTTP, nor is it about setting up an Ethernet network. Distributed computing, as it always has been, is about achieving high performance at low cost; it is a kind of "lowly parallel computing" that dwindling costs and increasing power of the hardware have brought about [5]. Easy as it might sound, distributed computing is still struggling to reach the goal of high performance that is commensurate with the amount of hardware invested. To the researchers, distribution is a long-standing problem.

We used to focus a lot on absolute performance, but when machines became faster, the focus seemed to have shifted; it did not, however, completely go away. Today, communication latency is a prime concern affecting performance in the design of high-performance distributed systems, especially when transparency is much demanded at application or user levels and when the system extends beyond a local area.

0743-7315/99 $30.00

A new class of distributed computing systems, called clusters, have emerged in recent years. Cluster proponents have made a very clear distinction between clusters and traditional distributed systems. What ultimately would make clusters unique and distinctive is the single system image (SSI) that these systems present to the application programmers and the users [4, 5]. For SSI to succeed, I/Os across the system have to be unified to create a single I/O space, and the same for processes, and processes have to be readily migratable at runtime. All this sounds wonderful, only if one can do it right. But doing it right means being able to overcome a lot of latency issues. Is complete transparency too costly a feature to have?

Distributed shared memory (DSM) suffers also from the latency problem. DSM is nice, but getting it to work really efficiently is difficult. Cache memory, which has become so much faster than main memory, has added another dimension to the problem. If the performance-conscious programmer has to be aware of where all the memory is, we are losing the advantages of what programming based on DSM is supposed to give us. Is the all-message-passing model a better alternative?

Apart from communication latency, there are other age-old problems that have remained as pressing as they have ever been: how to assign tasks to processors, how to balance workloads, how to overcome hardware and software faults, how to deal with heterogeneity, etc. Software supports implementing solutions to these problems must not be overlooked, even though these days people's attention seems to have been drawn to all these "trend wars"—such as that among middleware platforms (CORBA, DCE, DCOM, and Java RMI)—that are going on at higher levels. File systems, system mobility, modular construction, security, networking above TCP/IP, storage management, predictability of performance, and other components and features at the operating system level either are in a state of flux or still fall short of reaching those performance and transparency goals of distributed computing. Continued effort in researching on these components and features is much needed.

The purpose of this special issue is to provide an overview of recent developments and issues being researched by the parallel and distributed community and to identify challenges and future avenues in this promising area of software support for distributing computing. For the special issue, we received 34 submissions. Each paper was reviewed by three to six reviewers. We selected nine papers for inclusion in the special issue. These papers provide a panoramic view spanning a wide range of issues, ranging from scheduling and mapping and communication to middleware and applications. A short review of the accepted papers is given in the following.

Assignment of tasks and determining the order of the tasks assigned to each machine of a distributed system have long been recognized to be challenging problems. Efficient scheduling and assignment schemes are more crucial in a heterogeneous environment, where not only do the speeds and characteristics of the machines differ, but also their interconnection network links may have diverse bandwidths. In contrast to static methods, where the characteristics of tasks are known a priori, dynamic methods are required to deal with tasks as they arrive. The first paper by Maheswaran, Ali, Siegel, Hensgen, and Freund, entitled "Dynamic Mapping of a Class of Independent Tasks on Heterogeneous Computing Systems," proposes two types of mapping heuristics. First, for on-line mapping the paper proposes two heuristics and compares them with three existing heuristics. Second, for batches of tasks the paper proposes another heuristic and compares it with two existing heuristics. The proposed heuristics outperform the existing

heuristics in terms of tasks' makespan. The experimental study gives interesting insights on the effects of task characteristics on the performance of these heuristics.

Parallel computing on a distributed platform needs software support at the operating system level. This support includes transparent dependency maintenance while preserving semantic correctness and matching of interacting tasks to processors. Transparent dependent maintenance, in turn, includes support for communication, file access, memory access, process creation/termination, and preserving task precedence. Task matching needs to obtain load balancing subject to reduction of communication overhead. The paper by Cruz and Park, entitled "Toward Performance-Driven System Support for Distributed Computing in Clustered Environments," talks about such a system support environment. Named DUNES, their system can complement a commodity operating system with essential functionalities for parallel processing in a transparent manner. One of the attractive features of their system is the process of caching that reduces the communication overhead. The experimental study, conducted using a parallel iterative algorithm for solving linear equations, includes results obtained from a UNIX-based system with heterogeneous machines. Three load-balancing schemes are implemented that improve the job completion time by significant margins.

PC- and workstation-based clusters using a LAN are now widely adapted as common platforms for high-performance applications. A major portion of present research is geared toward making such platforms more effective and reliable. For this purpose, highly efficient middleware layers are needed that can exploit the communication performance offered by modern networks to parallel applications. In their paper, "ARTS of PEACE—A High Performance Middleware Layer for Parallel Distributed Computing," Buttner, Nolte, and Schroder-Preikschat describe an object-oriented platform using a middleware layer that provides high-level operations and utilizes the network with low overhead. This work is an extension of their previously proposed and implemented parallel operating system known as PEACE. The ARTS framework enhances the efficiency and functionalities of PEACE through high-performance middleware support. Functions such as basic communication, multicasts, and global reductions directly benefit from the middleware layer and incur low communication overhead. ARTS, which is available on PC clusters, is qualitatively compared with the CORBA system.

Harnessing distributed shared memory has baffled researchers for quite some time. One of the major problems in distributed shared memory is the data race that arises when multiple tasks try to access the same variable. Data races can lead to run-time errors and unexpected results and can be more cumbersome to detect than other programming errors. It is desirable to have algorithms that can detect data races on-the-fly while the program is running. However, most of the existing algorithms are not able to detect all data races on-the-fly. The second problem is that the overhead of on-the-fly detection of data races can be prohibitively high. The paper entitled, "Toward Integration of Data Race Detection in DSM Systems," authored by Itzkovitz, Schuster, and Zeev-Ben-Mordehai discusses a new on-the-fly data race detection scheme. The main contribution of the proposed scheme is its low overhead that is due to the matching of detection and data object granularities. This approach has additional advantages that help in improving the performance. Based on this scheme, a distributed protocol is designed that announces data races as they occur. The protocol is evaluated on a cluster of eight Pentium processors

running Windows-NT using a set of applications from NAS benchmarks. The paper also provides a useful overview of existing work on this topic.

The next paper entitled, "On Choosing a Task Assignment Policy for a Distributed Server System," written by Harchol-Balter, Crovella, and Murta addresses the problem of task assignment in a distributed system, where each incoming task is assigned to a host and each host machine processes its assigned tasks in a first-come-first-served fashion. The problem is finding a task assignment policy to maximize the performance such as reducing the task response time. Even though this problem has been extensively studied and numerous task assignment policies have been proposed in the past, this paper examines the influence of task size variability in choosing the best task assignment policy. Using both simulation and analysis, the authors evaluate a number of policies and conclude that no single policy is the best under all circumstances. This finding leads to a number of additional observations, prompting the authors to propose a new policy that outperforms the other policies.

Fast dissemination and access of information in large distributed systems has become one of the necessities of our daily lives, fueled largely by the unprecedented growth of the Internet. Reducing the data access times for multimedia information is considered to be an important performance objective. Creating replicas of frequently accessed objects across a read-intensive network can result in large bandwidth savings which, in turn, can lead to reduction in user response time. The set of sites at which an object is replicated constitutes its replication scheme. The paper, "Content Replication in a Distributed and Controlled Environment," by Li investigates the problem of placing multiple replicated servers in a distributed system of Web servers acting as information providers. The problem of finding an optimal set of servers for replication is shown to be an NP-complete problem which is then solved with a number of heuristics, including a greedy algorithm, a genetic approach, and local search. Comparisons are made with an optimal algorithm.

One of the strengths of network-based computing systems is their higher cost–performance ratio compared to that of expensive parallel machines. However, extracting a higher performance from a networked computer system is not a straightforward task. It is a well-known fact that communication overhead is one of the major hurdles. In their paper, "Adaptive Communication Algorithms for Distributed Heterogeneous Systems," Bhat, Prasanna, and Raghavendra present a framework for providing application-level communication support. The framework allows the development of communication schedules for collective communication patterns such as the total exchange. The communication schedule is developed using a heuristic that guarantees the completion time to be within twice the optimal. Simulating a heterogeneous system, the paper provides extensive results and illustrative examples.

There is an increasing demand for using a cluster of workstations for computationally intensive jobs. One such application is the ISODATA classification algorithm, an iterative method that performs clustering and grouping of objects based on their similarity. Since the algorithm is computationally intensive, an efficient distributed version can speed up the problem solving time. The distributed algorithm can execute a supervisor–worker model in which one workstation working as a supervisor divides the problem into tasks and assigns them to other workstations acting as workers. However, an effective implementation of this programming model requires orchestrating various activities such as partitioning, scheduling, and

distribution of workload. The paper entitled, "D-ISODATA: A Distributed Algorithm for Unsupervised Classification of Remotely Sensed Data on a Network of Workstations," authored by Dhodhi, Saghri, Ahmad, and Ul-Mustafa, proposes a distributed version of the ISODATA algorithm that yields a high speedup using up to eight workstations.

The final paper entitled, "Alternatives to Coscheduling a Network of Workstations," authored by Nagar, Banerjee, Sivasubramaniam, and Das addresses the problem of scheduling jobs on a network of workstations. The type of scheduling considered in the paper is at the CPU-level considering message and interrupt handling. The paper provides a good survey of existing techniques and examines nine scheduling techniques out of which five are newly proposed. Each strategy is evaluated in terms of throughput, response time, CPU utilization, and fairness. The proposed scheduling mechanisms are integrated into the MPI programming environment.

ACKNOWLEDGMENTS

We thankfully acknowledge all the authors of submitted papers. We thank Professor Kai Hwang for his support for the special issue and for continued guidance during the editorial process. Special thanks are due to the reviewers who provided useful comments and suggestions that helped us make the selection of papers. Professor Ahmad's work was supported by the Hong Kong Research Grants Council under Grant HKUST6076/97E. The reviewers were Imtiaz Ahmad, Shoukat Ali, Mark Allen Ehab Al-Shaer, David A. Bader, Amnon Barak, Noah Beck, Prashanth Bhat, Jun Bi, J. P. Black, Lotzi Boloni, Tracy Braun, Tim Brecht, Peter A. Buhr, Edward Chan, Sriram V. Chandrasekaran, Guihai Chen, Shiping Chen, William Chu, Yeh-Ching Chung, Alan Clement, Mark Crovella, John Cruz, Sajal Das, Xing Du, Guy Edjlali, H. El-Rewini, Mor Harchol-Balter, Yong He, Gernot Heiser, Debra Hensgen, Thomas Hou, Lim Joo Hwee, Mehmet Karaata, Iffat H. Kazi, Jai-Hoon Kim, James Kohl, Ravindra Kuramkote, Ricky Y. K. Kwok, Siet Leng Lai, Clarence Lau, Byoungjoo Lee, Xiaola Lin, Bo Li, Keqin Li, Zhiyuan Li, Qin Lu, Steve MacDonald, Rajib Mall, T. A. Marsland, Thierry Monteil, Jogesh K. Muppala, Cristina Murta, Matt W. Mutka, Joseph K.-Y. Ng, Joerg Nolte, Stephan Olariu, Taesoon Park, L. M. Patnaik, Jim Plank, Jerry Potter, C. S. Raghavendra, Rajeev Raje, Krithi Ramamritham, Binoy Ravindran, Donna Reese, Vincent Rodin, Sumit Roy, Samuel H. Russ, Michael J. Quinn, Wolfgang Schroeder-Preikschat, J. P. Sheu, Chris Sheu, Behrooz A. Shirazi, Jonathan Simonson, Ambuj Singh, Ron Sass, Hemal V. Shah, Anand Sivasubramaniam, Nenad Stankovic, Boleslaw Szymanski, Domenico Talia, Wei Tao, David Taylor, Cho-Li Wang, Wenping Wang, Xiaoge Wang, Stefan Weber, Jie Wu, Chengzhong Xu, Zhiwei Xu, Tao Yang, C. K. Yuen, Zhensheng Zhang, and Albert Y. Zomaya.

REFERENCES

1. http://setiathome.ssl.berkeley.edu/.
2. http://www.mersenne.org/prime.htm.
3. http://www.distributed.net/rc5.

4. K. Hwang, H. Jin, E. Chow, C.-L. Wang, and Z. Xu, Designing SSI clusters with hierarchical checkpointing and single I/O space, *IEEE Concurrency* **7** (1999), 60–69.
5. G. F. Pfister, *In Search of Clusters*, 2nd ed., Prentice–Hall, Englewood Cliffs, NJ, 1998.
6. M. van Steen, P. Homburg, and A. S. Tanenbaum, Globe: A wide-area distributed system, *IEEE Concurrency* **7** (1999), 70–78.

ISHFAQ AHMAD received a B.Sc. degree in electrical engineering from the University of Engineering and Technology, Lahore, Pakistan, in 1985. He received his M.S. degree in computer engineering and Ph.D. degree in computer science, both from Syracuse University in 1987 and 1992, respectively. At present, he is an associate professor in the Department of Computer Science at the Hong Kong University of Science and Technology. His research interests are in the areas of parallel programming tools, scheduling and mapping algorithms for scalable architectures, video compression, and interactive multimedia systems. He is director of the Multimedia Technology Research Center at HKUST, where he and his colleagues are working on a number of research projects related to information technology, in particular in the areas of video coding and interactive multimedia systems in a distributed environment using high-performance computing. He has served on the program committees of numerous international conferences and has guest-edited several journals. He serves on the editorial board of *Cluster Computing*, *IEEE Concurrency*, and *IEEE Transactions on Circuits and Systems for Video Technology*. He is a member of the IEEE Computer Society.

FRANCIS LAU has a B.Sc. degree from Acadia University, Canada, and an M.Math. and a Ph.D. degree from the University of Waterloo, Canada. He joined the Department of Computer Science and Information Systems at the University of Hong Kong in 1987, where he is now an associate professor and a leader of the Systems Research Group. One of the current projects of the group is to develop a Java-based platform for cluster computing. Dr. Lau's main research interests are in parallel and distributed computing, object-oriented programming, and operating systems. Dr. Lau is an active member of the IEEE Computer Society, where he is now serving as the 1999 Vice President for Chapters' Activities.