Towards a Portable Cluster Computing Environment Supporting Single System Image

Tatsuya Asazu† Bernady O. Apduhan‡ Itsujiro Arita‡

Department of Artificial Intelligence
Kyushu Institute of Technology
Iizuka, 820-8502 Japan

† *asazu@mickey.ai.kyutech.ac.jp*
‡ {*bob, arita*}*@ai.kyutech.ac.jp*

Abstract

Cluster computing environments, e.g. DSE, NOW, created out of commodity-off-the-shelf computer and networking hardware have gained wide popularity as a cost-effective solution to high performance computing. Considering the recent trends in cluster computing technology and for practical usability, the need to provide a single-system image is important where users can freely use these cluster computing systems without knowing the underlying system architecture.

In this paper, we cite some related works and present our approach. We describe the modified software organization of DSE, discuss the experiments and results of parallel applications on different UNIX-based platforms. Finally, we give our concluding remarks and future directions

1. Introduction

The research and development of network-based or cluster computing systems has burgeoned in recent years. Cluster computing environments constructed from *commodity-off-the-shelf* (COTS) hardware, i.e., workstations or PCs, connected by high-speed networks, have gained wide popularity as a compelling alternative to expensive parallel supercomputers or multiprocessors with highly customized hardware and software, establishing the paradigm of commodity supercomputing [1].

This significant low cost increases the availability of high performance computing platforms whose design enables an easy upgrade to improvements in computers and networking hardware technology. Aside from favorable price/performance ratio, cluster computing environments have a large choice of hardware and software vendors, as well as a number of operating systems and development environments.

A recent trend in cluster computing environments is to provide the popularly known *Single-System Image* (SSI) that would allow all nodes to offer a unified access to resources [2]. The notion of single-system image which can be offered at the hardware, operating system, applications and subsystem layers promises to provide a number of advantages as well as services to users.

In line with these advances in cluster computing technology motivated further by improvement of DSE (Distributed Supercomputing Environment), a cluster of workstations was developed in our laboratory [4]. The goal of this work is to study the intricacies of various UNIX-based operating systems to provide support for DSE to offer an adequate degree of portability, architecture and operating system independence for shared memory parallel processing on UNIX-based cluster environments without compromising efficiency for generality. Likewise, we attempt to expose the effects of these operating systems on the system performance, and how this is reflected in the behavior of applications performance. Furthermore, this study is aimed at gaining some insights into the limitations and potential of UNIX-based OS functions to reduce the time to implement and develop practical single-system image functionalities.

The following Sections are organized as follows: Section 2 describes some related works and our approach. Section 3 gives an overview of DSE and describes its modified software organization. Section 4 describes the experiment set-up, experiments, and results. Section 5 gives the summary and concluding remarks with a discussion on the implications of the results, and cites future works.

2. Related Works

PVM [5] and MPI [6] are development environments which provided degree of portability, architecture and operating system independence for message passing systems. GLUnix and Solaris MC are among those that provide a single-system image at the operating system level [2]. GLUnix is a global layer unix at the topmost layer in the existing OS of the heterogeneous computers in the network which provide the abstraction of a single, serverless file system [7]. Likewise, Solaris MC is built as a globalization layer on top of the existing kernel preserving the existing Solaris ABI/API and runs existing Solaris 2.x applications and device drivers without modification. Solaris MC provides a single-system image, making the cluster appear like a single machine to the user, to applications, and to the network [8].

We envisioned to provide similar functions to the above-mentioned works, but will significantly differ on targeted goals. Our goal is to develop a portable shared memory based cluster computing environment with SSI support. We shall utilize the compatible functions of UNIX-based OS to reduce the development time and effort to implement practical and flexible SSI functions.

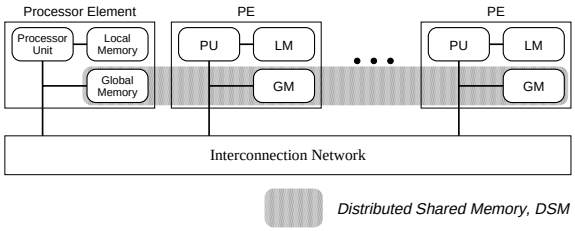

Figure 1. The DSE System Model

3. DSE and Software Organization

DSE is a shared memory based cluster computing environment developed on a network of SunOS-based workstations connected by a local area network, shown in Figure 1. The DSE software mainly consist of two components: the DSE kernel and DSE process [4]. An earlier attempt to implement the DSE kernel and DSE process into one UNIX process to reduce the communication cost in between has decreased the portability and modularity of DSE, and require dynamic linking. Likewise, the attempt to speed up communication processing by optimizing TCP/IP processing has lead to more dependency on the network protocols [9].

3.1. New DSE Software Organization

The overhead due to OS system calls and protocol processing seems inevitable since DSE is implemented at the UNIX user level. Also, since DSE adapted the shared memory model, communication frequency for fine-grain granularity is high. And so it is of great importance to reduce this overhead. Eventually, we re-evaluated DSE and improved its software organization in an effort to provide portability and more modularity to facilitate system improvement, shown in Figure 2.

The notion of putting the DSE kernel and DSE process into one UNIX process is again adapted but takes a new implementation approach, i.e., without using dynamic linking, and implement the DSE kernel as a parallel processing library, shown in Figure 3. Likewise, the implementation of DSE kernel and the DSE process into one UNIX process is achieved in the parallel application by linking this parallel processing library with the parallel API library.

This implementation method simplifies the system composition, does not directly use dynamic loading, and thus improves modularity. Moreover, we utilized the asynchronous I/O mode interruption in context switching of DSE kernel and the DSE process, as was used in older DSE implementation.

The improved system modularity provides the feasibility of using multi-threading technique, eliminates dependency on a specific communication protocol which will facilitate exploring and utilizing the raw performance of high-speed networks.

4. Experiments and Results

In this Section, we evaluate the performance of the improved DSE with parallel applications, e.g. Gauss-Seidel Method, DCT-II, Othello and Knight's Tour Games, on three platforms, shown in Table 1.

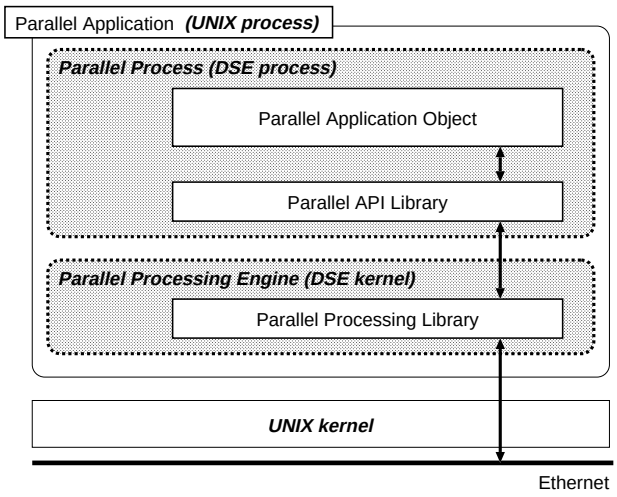

Figure 2. The DSE Software Organization

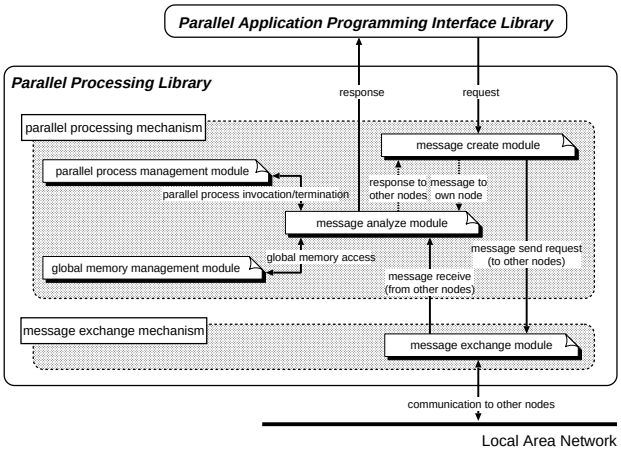

Figure 3. The Parallel Processing Library

4.1. Gauss-Seidel Method

SunOS over Sun Workstations

The dimension of the simultaneous equation varied from 8 to 192. Figure 4 shows the execution time as the number of processors is increased. Figure 5 shows higher speed-up rates with 4 processors and slows down with greater number of processors. This graph can be divided into two. The speed-up rate decreases when N-dimension is 8, 16, and 32, while it shows improvement with 4 or 6 processors when N-dimension is 64 or more, and decreases thereafter. In the former case, this implies that with N-dimension, efficient parallel processing cannot be obtained. While in the latter case, the speed-up rate improved when processors are 4 or 6, because the problem size is large.

The experiment's environment can be cited as one of the reasons why the speed-up did not improve with more than 4 (or 6) processors. Table 1 shows the actual number of computers (SparcStation10) we utilized in this experiment. A virtual cluster environment is constructed by starting two or more DSE kernels on

Table 1. Experiments environments

	Machine	OS
Platform 1	Sun SparcStation10 × 6	SunOS 4.1.4-JL
Platform 2	RS/6000 × 8	AIX 4.1.4
Platform 3	PC-AT (PentiumII 450MHz) × 4	GNU/Linux (kernel 2.1.106)

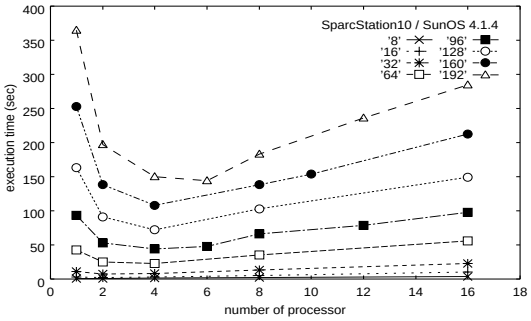

Figure 4. Gauss-Seidel Method on SunOS over SparcStation10

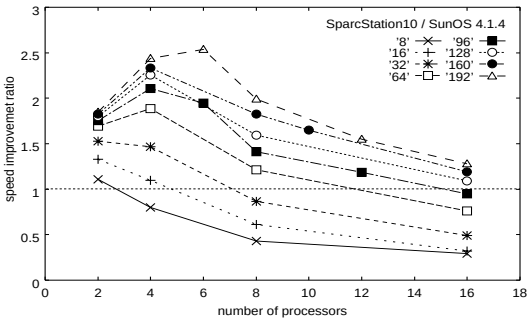

Figure 5. Speed-up of Gauss-Seidel Method on SunOS over SparcStation10

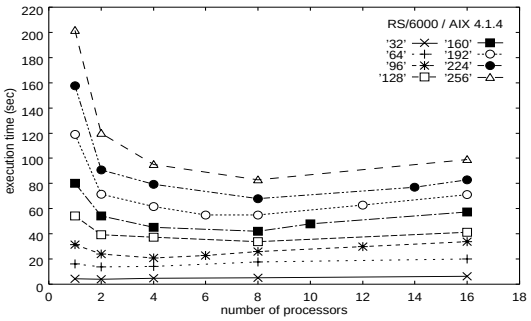

Figure 6. Gauss-Seidel Method on AIX over RS/6000

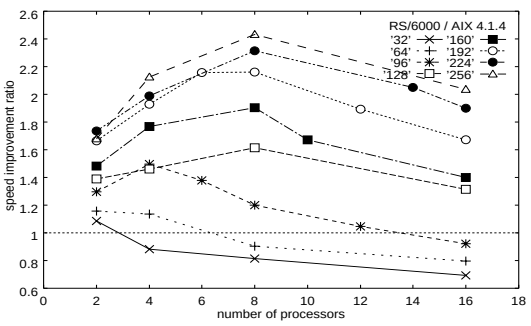

Figure 7. Speed-up of Gauss-Seidel Method on AIX over RS/6000

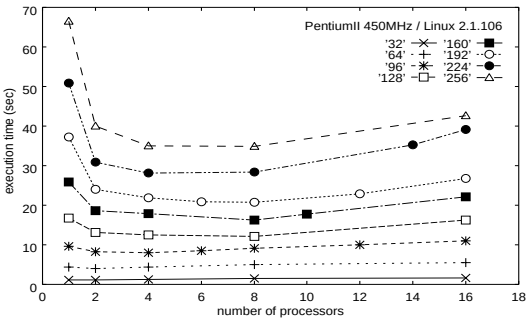

Figure 8. Gauss-Seidel Method on Linux over PC-AT

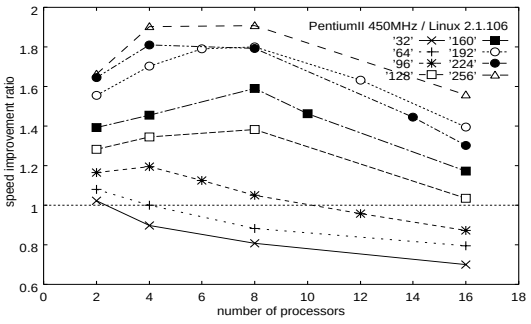

Figure 9. Speed-up of Gauss-Seidel Method on Linux over PC-AT

each computer when the required number of processors is more than six. For example, two DSE kernels start on each computer when the number of processors is 12. Therefore, when starting two or more DSE kernels on one machine, the machine load increases in proportion to this number. This causes the decrease in performance when the number of processors exceeds six.

AIX over IBM RS/6000 and Linux over PC-AT

The same performance patterns were observed when experiments (with N-dimension from 32 to 256) were conducted on AIX over IBM RS/6000 and Linux over PC-AT Pentium II platforms. These are shown in Figures 6, 7, 8, and 9, respectively.

4.2. DCT-II

Discrete Cosine Transformation-Two Dimension (DCT-II) is an image compression method in which the source image is divided into some independent partial processing, and every pixel block of $N \times N$ can be processed in parallel.

SunOS over Sun SparcStation10

Figure 10 shows the result of performing DCT-II on an image of size 256×256 pixel, with block size varied from 8×8, 16×16, 32×32, 64×64 at 65% compression rate. Here the sequential processing is executed with one processor and parallel processing is executed with two or more processors.

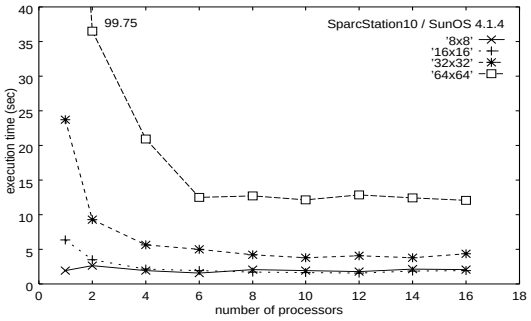

Figure 10. DCT-II on SunOS over SparcStation10

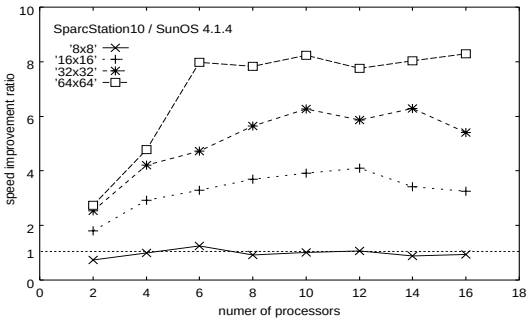

Figure 11. Speed-up of DCT-II on SunOS over SparcStation10

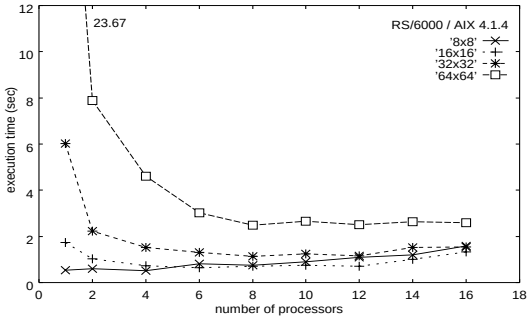

Figure 12. DCT-II on AIX over RS/6000

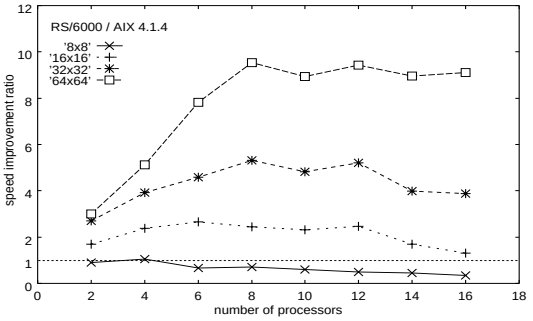

Figure 13. Speed-up of DCT-II on AIX over RS/6000

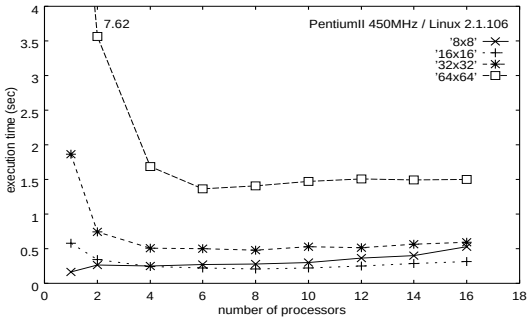

Figure 14. DCT-II on Linux over PC-AT

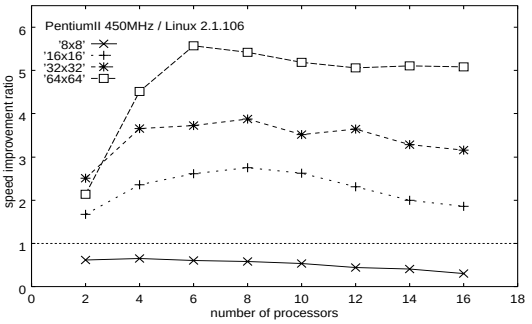

Figure 15. Speed-up of DCT-II on Linux over PC-AT

Likewise, Figure 11 shows the speed-up ratio with respect to sequential processing. It can be observed that speed-up improves in all block sizes when the number of processors is increased except in block size 8. This can be attributed to the small computation granularity and longer communication processing time with block size 8. Decrease in communication frequency and increase in block size (16, 32, 64) exhibits good speed-up ratio.

AIX over IBM RS/6000 and Linux over PC-AT

Similar performance behaviors were observed, shown in Figures 12, 13, 14, 15, when the same DCT-II experiments were conducted on AIX over IBM RS/6000 and Linux over PC-AT PentiumII machines, respectively.

4.3. Othello Game

The Othello game is a typical search problem application common in artificial intelligence research.

SunOS over Sun SparcStation10

In this experiment, we verify the execution speed of the Othello game written in parallel program at different depths. In Figure 16 with depths 3, 4, and 5, no speed-up improvement is observed due to the influence of communication frequency as the number of processors is increased. However with higher depths, speed-up improved where the effect of parallelism can be observed.

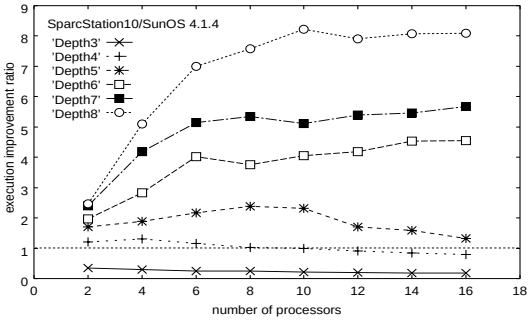

Figure 16. Speed-up of Othello Game on SunOS over SparcStation10

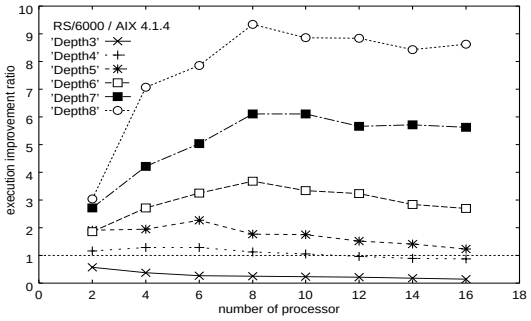

Figure 17. Speed-up of Othello Game on AIX over RS/6000

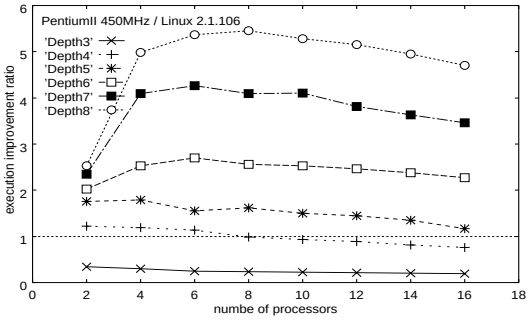

Figure 18. Speed-up of Othello Game on Linux over PC-AT

AIX over IBM RS/6000 and Linux over PC-AT

Similar performance patterns were observed, as shown in Figures 17 and 18, when the same Othello game experiments were conducted on AIX over IBM RS/6000 and Linux over PC-AT PentiumII machines, respectively.

4.4. Knight's Tour Problem

Knight's Tour problem is also a search problem whose task is to find the route which a knight passes all masses on the surface of an $N \times N$ chess board only once.

SunOS over Sun SparcStation10

The purpose of this experiment is to investigate the influence on the parallel program execution when the number of processors is increased with varying computation granularity. Here, the relation between the computation granularity and execution speed is examined by changing the computation granularity in the chess board of 5×5.

In Figure 19, we observed that the case of 80 jobs is most efficient while the case of 648 jobs is the least efficient. In the case of 256 and 648 jobs, the execution speed improves until the number of processors becomes 6, but the speed decreases proportionately as the number of processors exceeds 6. This decrease can be attributed to the large number of divisions in the problem where communication frequency is also large. Moreover, the experiment environment, like the bus type Ethernet where occurrence of packet collision increases when communication frequency between nodes increases. The execution speed does not improved in the case of 26 jobs which can be due to the small number of divisions of the problem even with increase on the number of processors. In the case of 80 jobs, the execution speed improves although it tends to stay constant as the number of processors is increased from 8 to 16.

AIX over IBM RS/6000 and Linux over PC-AT

Similar performance patterns were observed, shown in Figures 20 and 20, when the same Knight's Tour experiments were conducted on AIX over IBM RS/6000 and Linux over PC-AT PentiumII machines, respectively.

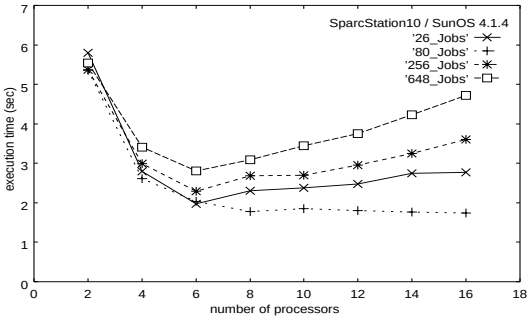

Figure 19. Knight's Tour Problem on SunOS over SparcStation10

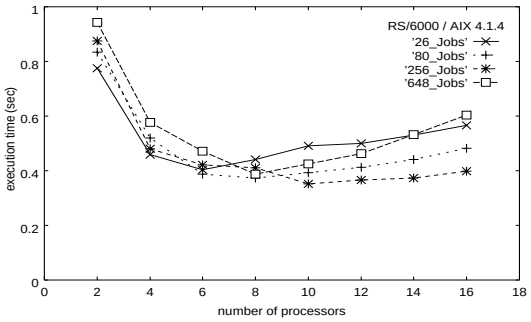

Figure 20. Knight's Tour Problem on AIX over RS/6000

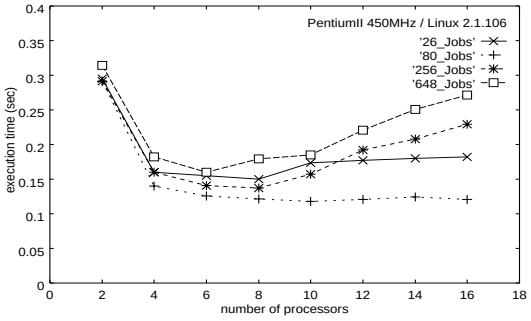

Figure 21. Knight's Tour Problem on Linux over PC-AT

5. Concluding Remarks

In this paper, we presented the modified DSE software organization which provides more modularity to reduce modules' improvement time, and portability for wider usability. We performed some parallel applications on different UNIX-based cluster environments with different problem sizes and number of processors. With the modified DSE software organization, experiment results reveal substantial enhancement to DSE system performance (Note: Due to space limitation, performance of older versions can be found in [3],[4],[9]). Furthermore, the results show similar performance patterns in all environments exhibiting scalability and ensuring portability.

This is a work in progress and we plan more exploration and study of the intricacies of UNIX-based operating systems. And to carry out experiments on other UNIX-based platforms in order to further assess the portability function and gain substantial knowledge as to how to efficiently realize a portable cluster computing environment with single-system image support.

References

[1] Buyya, R., "Cluster Computing: The Commodity Supercomputing", *Journal of Software - Practice and Experience*, John Wiley & Sons, Inc., January 1999.

[2] Buyya, R., "Single System Image: Need, Approaches, and Supporting HPC Systems", *Proc. Fourth International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'97)*, Las Vegas, Nevada, USA, CSREA Press, 1997.

[3] B.O. Apduhan, T. Sueyoshi, Y. Namiuchi, T. Tezuka, I. Arita, T. Fujiki, "Experiments and Analysis of Distributed Supercomputing in a Distributed Workstation Environment", *SUPERCOMPUTER*, ASFRA bv, Edam, The Netherlands, Vol. VIII, No. 6, pp. 90-100, November 1991.

[4] T. Tezuka, K. Ryokai, B.O. Apduhan, and T. Sueyoshi, "Implementation and Evaluation of a Distributed Supercomputing Environment on a Cluster of Workstations", *Proc. of 1992 International Conference on Parallel and Distributed Systems*, pp. 58-65, 1992.

[5] Geist, A., et al, "PVM Parallel Virtual Machine - A User's Guide and Tutorial for Networked Parallel Computing", *The MIT Press*, 1994.

[6] MPI Forum, "MPI: A Message-Passing Interface Standard", *International Journal of Supercomputer Applications and High Performance Computing*, Vol. 8, 1994.

[7] Ghormley, D.P., et al, "GLUnix: A Global Layer Unix for a Network of Workstations", *Journal of Software - Practice and Experience*. (To appear)

[8] Khalidi, Y.A., et al, "Solaris MC: A Multi Computer OS", *Proc. 1996 USENIX Technical Conference*, pp. 191-203, San Diego, CA, January 22-26, 1996.

[9] K. Ryokai, T. Tezuka, B.O. Apduhan, and T. Sueyoshi, "Improvement of Communication Processing Using Thread and Signal in a High-Performance Computing Environment on a Cluster of Workstations", *Proc. of 8th International Joint Workshop on Computer Communication*, pp.G-4-2-1 $\sim$ G-4-2-8, 1993.

[10] D.E. Culler, et al, "Parallel Computing on the Berkeley NOW", *Proc. 9th Joint Symposium on Parallel Processing*, 1997.