Performance Impact of Process Mapping on Small-Scale SMP Clusters -A Case Study Using High Performance Linpack

Tau Leng Rizwan Ali Jenwei Hsieh Victor Mashayekhi Reza Rooholamini
Dell Computer Cooperation
{Tau_leng; Rizwan_Ali; Jenwei_Hsieh; Victor_Mashayekhi; Reza_Rooholamini@dell.com}

Abstract

Typically, a High Performance Computing (HPC) cluster loosely couples multiple Symmetric Multi-Processor (SMP) platforms into a single processing complex. Each SMP uses shared memory for its processors to communicate, whereas communication across SMPs goes through the intra-cluster interconnect. By analyzing the communication pattern of processes, it is possible to arrive at a mapping of processes to processors to ensure optimal communication paths for critical traffic. This critical traffic refers to the communication pattern of the program, which can be characterized by either frequency or size (or both) of the messages. To find an ideal mapping, it is imperative to understand the communication characteristics of the SMP memory system, intra-cluster interconnection, and the Message Passing Interface (MPI) program running on a cluster.

Our approach is to study the ideal mapping for two classes of interconnects: 1) standard, high-volume Ethernet interconnects, and 2) proprietary, low-latency high-bandwidth interconnects. In the first installment of our work presented in this paper, we have focused on the ideal mapping for the first class.

We configured a 16-node dual-processor cluster, interconnected with Fast Ethernet, Gigabit Ethernet, Giganet, and Myrinet. We used High Performance Linpack (HPL) benchmark to demonstrate that re-mapping of processes to processors (or changing the order of processors used) can affect the overall performance. The mappings are based on the HPL program analysis obtained from running a MPI profiling tool. Our results suggest that the performance of HPL using Fast Ethernet as the interconnect can be improved from 10% to 50% depending on the process mapping and the problem size. Conversely, an ad hoc mapping can adversely affect the cluster performance.

1. Introduction

Cluster of commodity computers is the fastest growing choice for building cost-effective high-performance parallel computing systems. The rapid advance of microprocessor technologies and high-speed interconnects have facilitated many successful deployments of this type of computing environment. Researchers in previous studies reported that the cluster interconnect is a crucial component which can significantly impact the performance of parallel applications [2]. Other empirical studies on commodity cluster configurations suggested that Small-scale Symmetric Multi-Processors (SMPs) make ideal platforms for building High Performance Computing (HPC) clusters [4]. For instance, a two-way SMP has sufficient processing power, with a bounded contention on the system, memory, and I/O busses, is the most commonly selected platform today.

Using SMPs as the compute nodes introduced two types of communications paths to the cluster for any one pair of processors: 1) shared memory of a single SMP platform, and 2) message passing across SMP platforms. Communicating through shared memory can take advantage of the very high throughput of SMP memory subsystem without involving additional communications' resources such as PCI buses. However, when the communication traffic is intensive and/or the message sizes are large, memory contention can occur and deteriorate the performance significantly. Moreover, the context switching performed by the OS, when the processes communicate with each other within the same node, creates overheads on the program execution.

On the other hand, communicating across SMP platforms or through the cluster interconnect will release the memory contention, as well as have better utilization of system resources and load balance the communication overheads. Yet, a slow interconnect can decrease the cluster performance considerably, if the critical traffic of the program goes through internodes communication paths. This critical traffic refers to the communication pattern of the program, which can be characterized by either frequency or size (or both) of the messages. By analyzing the communication pattern of processes, it is possible to arrive at a mapping of processes to processors to ensure optimal communication paths for critical traffic. To find an ideal mapping, it is imperative to understand the communication characteristics of the SMP memory system, intra-cluster interconnection, and the Message Passing Interface (MPI) program running on a cluster.

Previous studies on communications between processes in message-passing applications showed the importance of arranging the mapping for better performance in a heterogeneous-network distributed system [5]. Duato et al proposed a "cluster-based" method

0-7695-1573-8/02/$17.00 (C) 2002 IEEE

to characterize the communications between processes generated by message-passing application [6]. A technique that provides a model consisting of partitions of the processes was also presented in the research. The information obtained by the method is useful when mapping processes to processors in heterogeneous environments where the communication bandwidth available is not the same for each group of processors.

For our study, we applied an experimental approach and used a message-passing analysis tool to obtain the communication information during the run time of the program. We configured a 16-node dual-processor cluster, interconnected with Fast Ethernet, Gigabit Ethernet, Giganet, and Myrinet, and used High Performance Linpack (HPL) benchmark as an example. We set up three mappings of processes to processors (or arrange the order of processors used) to study the effects of HPL performance on the cluster when using different mapping. The three mapping principles, Round robin, Frequency-major, and Size-major, that we designed are based on the HPL program profile analysis obtained from running VAMPIR, a MPI tracing and profiling tool.

In the next section, we start with an introduction of cluster interconnects which are used for our experiments. Section 3 introduces the HPL benchmark and the TOP500, which lists the most powerful 500 computer systems. Section 4 and section 5 describes our experimental environment, and explains our methodology and the approaches we take for the study. In section 6, we show the experiment results and analysis. Section 7 is the conclusion.

2. The Interconnects

The interconnect that networks the individual compute nodes convert a group of computers into a single system, an HPC cluster, which is capable of executing parallel jobs. The performance of the interconnect is an important factor which effects the cluster performance significantly when the applications running on the cluster are communication centric. Ethernet is the most popular networking configuration today. However, due to its inefficiency, the conventional TCP/IP protocol is not necessarily suitable for a dedicated system area network, like a HPC cluster interconnect, to accomplish the parallel and communication intensive jobs. Nevertheless, familiarity and the cost factor make Fast Ethernet and Gigabit Ethernet the more popular and cost-effective choices for many applications.

Giganet [1] and Myrinet [3] are two of the leading cluster interconnect technologies for HPC clusters. Both provide low-latency, high-bandwidth, end-to-end communication between two processes in the cluster. Giganet is a connection-oriented interconnect based on a hardware implementation of Virtual Interface Architecture (VIA) and Asynchronous Transfer Mode (ATM) technologies. Myrinet is a connection-less interconnect which leverages packet switching technologies from experimental Massively Parallel Processors (MPP) networks. Giganet achieves this by a hardware implementation of VI Architecture [], which gives a user process direct access to the network interface, avoiding immediate copies of data and by-passing the operating system (OS) in a fully protected fashion. Myrinet is not VI Architecture compliant, but it offers similar advanced mechanisms for efficient communication through its Grand Message (GM) message-passing system.

The following table shows the specification of the four interconnects we used for our experiments.

Inter-connect	Fast Ethernet	Gigabit Ethernet	Giganet	Myrinet -2000
Network Interface Card	Intel 100Mb	Broadcom 1000Mb	cLAN-1000 64/32-bit, 33MHz PCI	Myrinet-2000 64/32-bit 66MHz PCI
Switch	3Com 4300	Foundry FastIron-III	cLAN 5300	Myrinet-2000
Link speed	100 Mb	1 Gb	1.25 Gb	2 Gb
Topology	Non-blocking	Non-blocking	Fat Tree	Clos network

3. HPL and Top500

Linpack is a commonly applied benchmark in the High Performance Computing arena. It uses a number of linear algebra routines to measure the time it takes to solve dense linear equations in double precision (64 bits) arithmetic using the Gaussian elimination method. The measurement obtained from Linpack is in number of floating-point operations per second (FLOPS). In the early 80's the original version of Linpack had a fixed problem size of approximately 100x100, which was no longer useful when this size could be easily held in the cache of a microprocessor. HPL is one of the several newer Linpack implementations developed to overcome scalability problem. Today, Linpack is the benchmark used for measuring a system's performance as well as ranking supercomputers on the Top500 list. Starting from 1993, Top500 has maintained a list of 500 computer systems twice a year that have the most powerful computing capability, measured by number of FLOPS, installed in the world.

When running HPL, it is important to select a suitable grid size or "problem size" in order to get the best performance of the system. The recommended problem

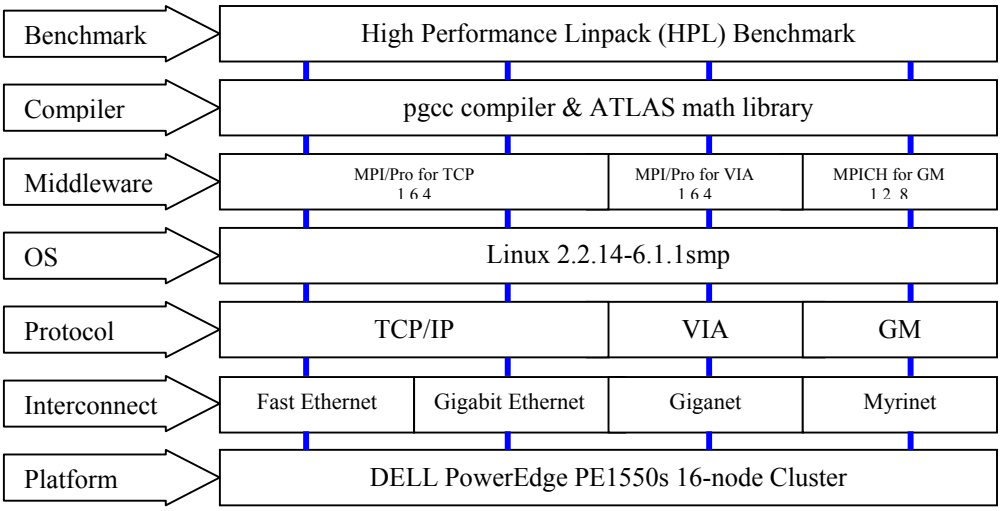

Figure 1. Architectural stack of the test environment. HPL benchmark is compiled with Portland Group's pgcc compiler. The Linux is RedHat 6.2 distribution with kernel version 2.2.14-6.1.1smp.

size is approximately no more then 80% of the total amount of memory in the system as the problem size. If too large a problem size is used, swapping may occur and the performance will decrease greatly.

4. Experimental Environment

Our testing environment is based on a cluster consisting of 16 Dell PowerEdge 1550 servers interconnected with Fast Ethernet, Gigabit Ethernet, Giganet, and Myrinet. Each PowerEdge 1550 has two Intel Pentium III processors running at 1 GHz with 256KB L2 cache, 512MB 133 MHz, Sync, CL3, ECC of SDRAM memory and dual 64-bit 66 MHz PCI slots for the interconnects. The FSB and memory are both at 133 MHz speed. The benchmark program HPL is executed on the same server platform, Linux kernel (RedHat's Linux distribution), and compilers (Portland Group's pgcc compiler) with ATLAS (Automatically Tuned Linear Algebra Software) math library. Figure 1 shows the architectural stack of our test environment.

5. The Methodology

Latency and Bandwidth. First, to obtain a "clue" about the communication paths' performance, we use "perf" benchmark, a point-to-point communication-measuring program, to obtain the latency and bandwidth information of the four interconnects and the shared memory communication performance of the cluster compute node. The results for the four interconnects were obtained by using two nodes in the cluster, and the shared memory latency and bandwidth test was based on the TCP / VIA versions of MPI/Pro and GM version of MPICH. Both the latency and bandwidth charts as shown in Figure 2 indicate that the communication performance ranks from Shared Memory (MPICH for GM protocol) as being the best all the way thru to Fast Ethernet which is the worst. Note that, although in theory, the shared Memory communication should have the greatest throughput compared to the inter-node communications, overheads such as OS context switching and the memory contention that occur when the two processes communicate through the memory bus, make the communication throughput lower than the fast interconnects. The bottom chart in Figure 2 shows a drop-off for shared memory communications at 64KB message length, which is the MPI communication buffer size. Other factors, which might influence the performance of shared memory communications, include the shared memory support from the OS and the MPI implementation.

Analyzing HPL program behavior using VAMPIR. To understand the communication pattern of HPL, our next step is to analyze the HPL program by using VAMPIR, a commercial MPI program profiling and visualization tool from PALLAS GmbH. VAMPIRtrace, also available from PALLAS, is an instrumented MPI

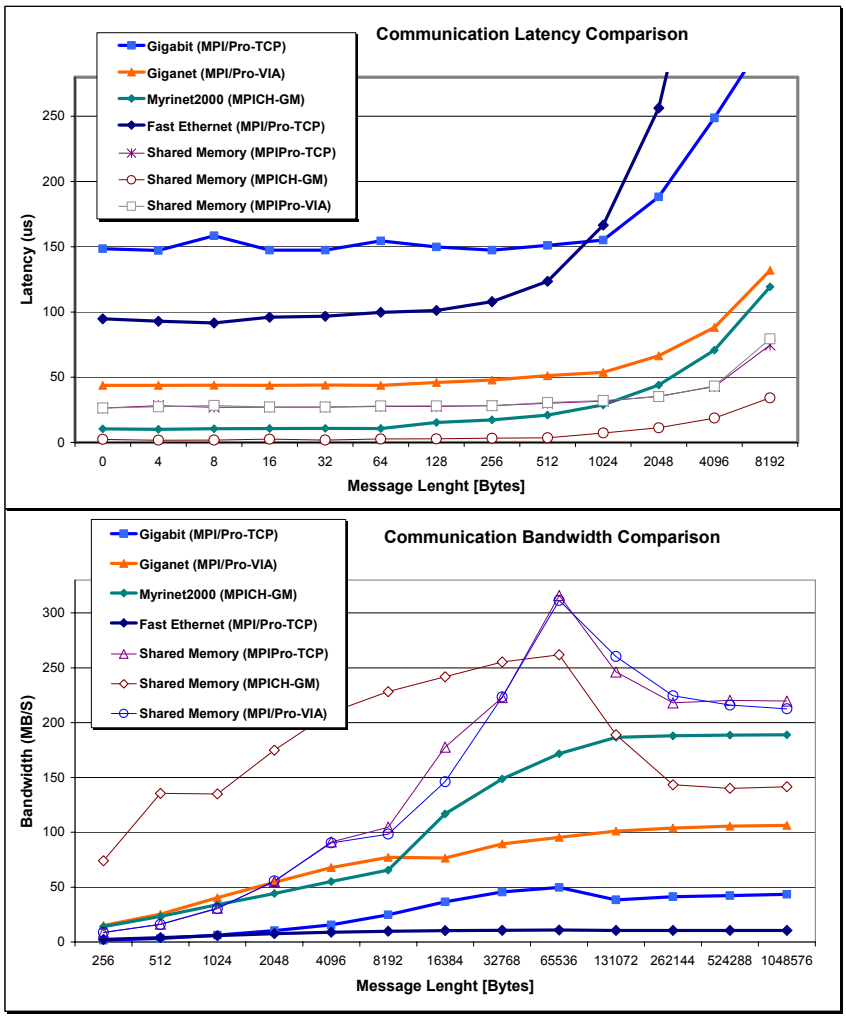

Figure 2. The latency and bandwidth of four interconnects and the shared-memory associated with the MPI and protocols.

library. In order to "visualize" the MPI program behavior, instrumentation is necessary before or during the compiling / linking of the understudied program. In our experiment, the instrumentation is done by linking the HPL with VAMPIRtrace library. The VAMPIRtrace library is then interposed between the HPL codes and the MPI library by means of the MPI profiling interface. Running the instrumented codes produce a merged trace file for HPL program that can be viewed by using VAMPIR profile visualization tool.

Figure 3 shows the "timeline" of the VAMPIR trace file obtained from executing the instrumented HPL program using 2000x2000 grids problem size on our cluster interconnected with Fast Ethernet. The 16 processes, process 0 to process 15, were running on a cluster of 16 processors. From the left to the right, the "timeline" chart shows the program activities, which include the application's local computation, MPI activities, and the VAMPIR trace code of the instrumented HPL program run. The green area (or light gray in a gray scale picture) showed in the timeline chart is the local computation on processors, while the red area (or dark gray) is the MPI activity among processes. The blue area, which is the part of executing VAMPIR trace codes for generating the trace file, cannot be seen in the chart. This implies the overhead created by instrumented VAMPIR trace library is very small and can be ignored.The black lines showed in the chart represent the message passing paths.

Figure 3. The time-line of HPL 16-processor run.

Detailed information, such as the source and destination, the message size, duration, and the type of communication can be obtained by clicking on the particular line and by other display features of VAMPIR.

From Figure 3, it can be observed that the most frequent communications are performed between neighboring processes, i.e., process 0 and 1, process 2 and 3, process 4 and 5, and so on. This implies that most of the message-passing actions take place in pairs between processes for the 16-processor run. However, most of the messages for those neighboring communications are small and range from 4 bytes to around 80 bytes. Infrequent but larger messages, around 600 KB to 800 KB, are communicated between processes 0 and 2, processes 1 and 3, processes 4 and 6, processes 5 and 7 and so on.

Similar pattern can be seen in 32-processor run. Figure 4 is the "timeline" of the same HPL program ran on the same cluster but utilizing 32 processors. The black blocks shown in the chart aggregated by black lines or communication paths indicate the message passing activities are very intensive among a group of four processes. That is, most of the communications of the 32-processor HPL run are performed among processes 0 to 3, 4 to 7, 8 to 11, and so on. Again, the messages for these communications are small, and those infrequent but larger messages are communicated between process 0 and 4, process 1 and 5, process 2 and 6, and so on.

In summary, the MPI communication pattern of HPL is concentrated on neighboring processes, which is in pair for 16 processes or in a group of four for 32 processes with small message size. Large messages, on the other hand, are communicated between non-neighboring or non-grouping processes. For small and frequent messages, low latency communication is required to expedite the program execution. As for the large messages, a high bandwidth communication path is required to move data efficiently and avoid performance bottleneck.

Mapping processes to processors. Rearranging the processor order used for running HPL program will change the mapping of processes to processors. The principle for the arrangement is to use the shared memory communication as much as possible when the cluster interconnect is slower. As for the clusters using faster interconnect, the critical communications should be arranged in such a way that the message sending and receiving processes are running on separate nodes. In a cluster environment, parallel programs are executed through MPI operations. For example, MPICH and MPI/Pro operate by spawning processes on the compute nodes based on a machine file and parameters given to the mpirun command. The machine file lists each of the nodes in the cluster by hostname. When a cluster invokes the mpirun command, mpirun scans the machine file and spawns processes on each node listed in the file.

Based on the latency and bandwidth test results in the previous section and the HPL program analysis above, we adopt three types of mapping principle for our testing, Round robin, Frequency-major, and Size-major. The Round robin arrangement is the default mapping used by most of the MPI and is a mapping of processes to processors in the order that process 0 to 15 will run on the first processors of node 0 to node 15, and process 16 to 31 will run on the second processors on node 0 to node 15.

0-7695-1573-8/02/$17.00 (C) 2002 IEEE

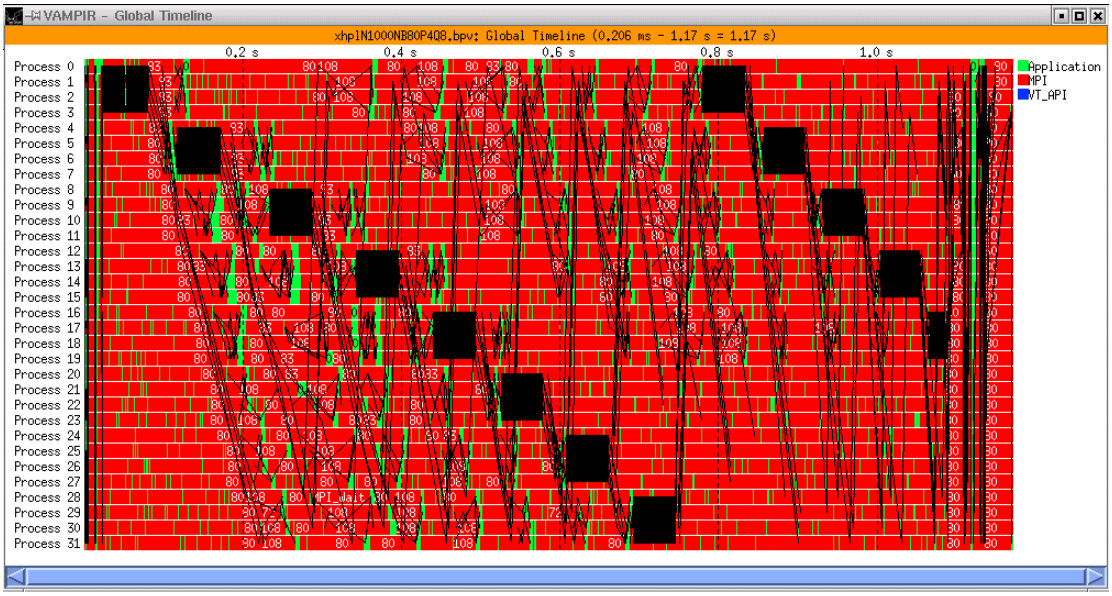

Figure 4. The time-line of HPL 32-processor run.

The Frequency-major arrangement is focusing on those communication intensive paths among processes, and the Size-major is looking at the large messages communication paths. Both of the arrangements are intended to use shared memory communication as much as possible for either frequent and small messages or infrequent but large messages when running HPL. The following is the three mapping arrangements for 8x2 and 16x2 configurations.

8x2 Configuration

Round robin	Frequency-major	Size-major
node 0	node 0	node 0
node 1	node 0	node 1
node 2	node 1	node 0
node 3	node 1	node 1
…	node 2	node 2
node 7	node 2	node 3
node 0	node 3	node 2
node 1	node 3	…
node 2	…	node6
node 3	…	node7
…	node 7	node 6
node 7	node 7	node 7

16x2 Configuration

Round robin	Frequency-major	Size-major
node 0	node 0	node 0
node 1	node 0	node 1
node 2	node 1	node 2
node 3	node 1	node 3
node 4	node 2	node 0
node 5	node 2	node 1
…	node 3	node 2
node 15	node 3	node 3
node 0	node 4	…
node 1	node 4	node 12
node 2	node 5	node 13
node 3	node 5	node 14
node 4	node 6	node 15
node 5	node 6	node 12
…	…	node 13
node 14	node 15	node 14
node 15	node 15	node 15

6. Experimental Results and Analysis

We conducted three sets of HPL experiments using the three processes-to-processors mapping principals, Round robin, Frequency-major, and Size-major respectively. For each set of the experiments, we run HPL on 8x2 and 16x2 cluster configurations using the four interconnects. Both of the configurations are set for running HPL with various problem sizes. For the 8x2 runs, the problem size ranges from 2000x2000 grids (0.03 GB or 0.8% of the total memory) to 20000x20000 grids (2.98 GB or 74% of the total memory). On the 16x2 runs, the problem sizes are from 2000x2000 grids (0.03 GB or 0.4% of the total memory) to 28000x28000 grids (5.98 GB or 73% of the total memory). The results of each run are measured in GFLOPS.

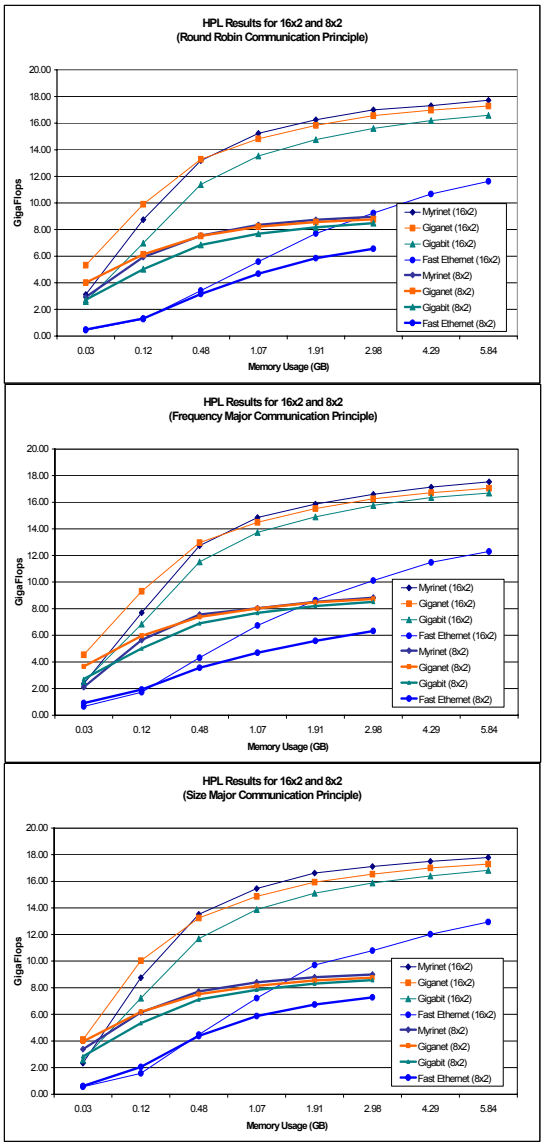

Figure 5, 6, & 7. The HPL results for 16x2 (thin lines) and 8x2 (thick lines) configurations. The processes order corresponded to the processors of the cluster are based on Round Robin, Frequency-major, and Size-major arrangements, from the top to the bottom respectively.

Figure 5 shows the HPL results for both 8x2 and 16x2 configurations using Round Robin arrangement of mapping processes to processors. The thin lines in the figure are the results for 16x2 runs, and the thick lines are 8x2's results. From the left to the right, the problem size or the memory usage is from small (0.03 GB) to large (5.94 GB). Figure 6 and 7 show the results of same experiments, but using the rearranged machine files with Frequency-major and Size-major principles respectively. In Figure 6, we can see that the results of both 8x2 and 16x2 for the four interconnects are closer than those in Figure 5. That is because these rearrangements help improve the performance of Fast Ethernet and Gigabit Ethernet configurations. This is also applicable for the results of Size-major runs shown in Figure 7.

Figure 8, 9, 10 and 11 show the performance differences among the results of the three mapping principles for the four interconnects respectively. The percentages shown in the bar charts are the performance improvement/deterioration of Frequency-major and Size-major runs using Round robin as the base (100%). The performance improvements for the Fast Ethernet are obvious. For 8x2, starting from small problem size runs (0.8% of memory usage) to large problem size runs (74% memory usage), the improvements are roughly 80% to 10%. As for the 16x2, the improvements are roughly 50% to 10% for the memory usage between 0.4% and 73%. Similar improvements can be seen on Gigabit Ethernet configuration but with smaller percentages.

Based on all the HPL results, it can be observed that the more the memory used, the better the HPL performed, which is particularly true for the slow interconnect, like Fast Ethernet. When running large problem sizes, the performance differences among the four interconnects become smaller. This also implies that the faster interconnect, like Myrinet, is reaching its peak performance sooner than slower ones. In other words, for slow interconnect clusters such as Fast Ethernet, more memory spaces and critical traffic going through shared memory are needed to compensate the low communication capability so as to obtain better HPL performance.

7. Conclusions

Rearranging the order of processors for executing parallel programs according to the communication behavior could improve the cluster performance. In this paper, we analyzed the HPL program communication pattern and designed two processes-to-processors mapping principles, Frequency-major and Size-major. The mapping principles were designed in favor of shared memory communications, and the outcomes were used to compare with those obtained from the default processes arrangement, Round robin. As a result, most of the critical traffic of HPL went through shared memory communication paths, which benefited the slower interconnect configurations and improved their performance. In particular, the improvement due to the Size-major arrangement was significant, while the Frequency-major showed modest performance increase because of the messages on these path are fairly small. Conversely, if the mapping principles designed were geared toward internodes communications, we could see performance improvement for faster interconnects.

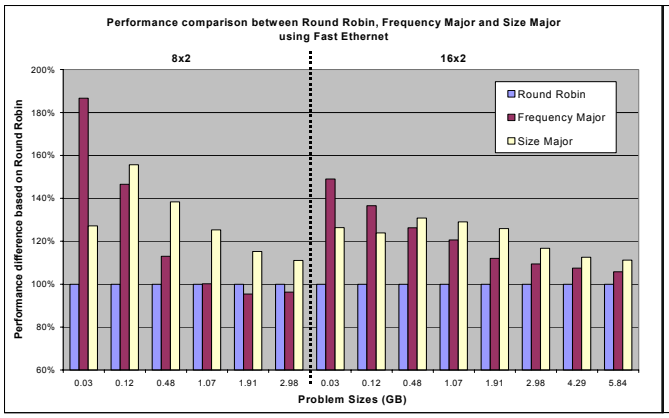

Figure 8. The performance differences (percentages using Round Robin as the base) among the results of three mappings running on Fast Ethernet configuration.

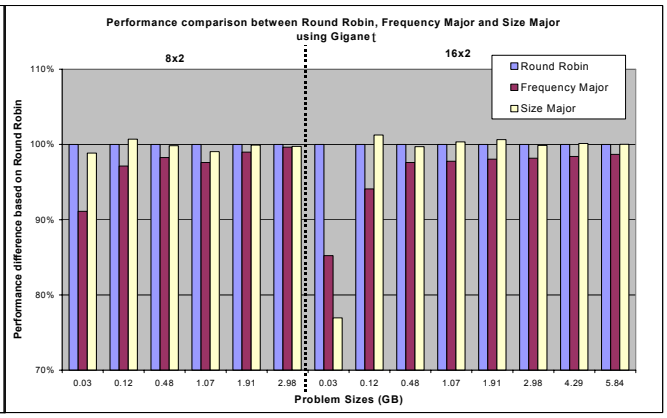

Figure 10. The performance differences (percentages using Round Robin as the base) among the results of three mappings running on Giganetconfiguration.

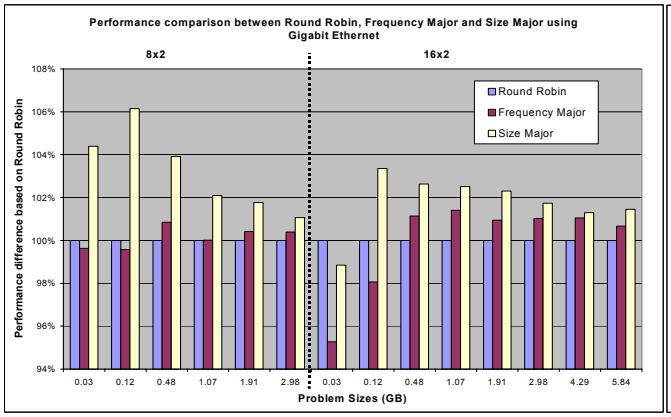

Figure 9. The performance differences (percentages using Round Robin as the base) among the results of three mappings running on Gigabit Ethernet configuration.

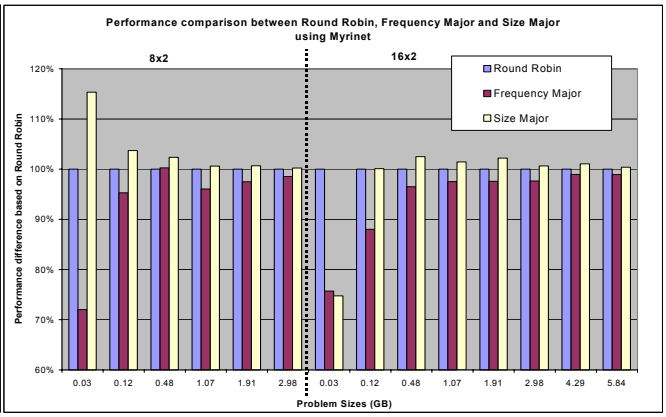

Figure 11. The performance differences (percentages using Round Robin as the base) among the results of three mappings running on Myrinet configuration.

The methodology presented in this paper can be applied on any application. The degree of impact on cluster performance will depend upon the program message-passing behavior, the performance characteristics of the interconnect and the shared memory, and how well the processes-to-processors mappings are arranged.

References

[1] Compaq, Intel, and Microsoft. "Virtual Interface Architecture Interface Specification, Version 1.0". *http://www.viarch.org*, December 1997

[2] J. Hsieh, T. Leng, V. Mashayekhi, and R. Rooholamini. "Architectural and Performance Evaluation of Giganet and Myrinet Interconnects on Cluster of Small-Scale SMP Servers", In the *Proceedings of Super-Computing '00*, Dallas, TX, Nov. 2000.

[3] Myrinet, Inc. "The GM Message Passing System", *http:// www.myri.com*, 1999.

[4] J. Hsieh, T. Leng, V. Mashayekhi and R. Rooholamini. "Impact of Level 2 Cache and Memory Subsystem on the Scalability of Clusters of Small-Scale SMP Servers", in the *Proceedings of International Conference on Cluster Computing*, Cluster'00, November 2000.

[5] M. G. Norman, P. Thanisch, "Models of Machines and Computation for Mapping in Multicomputers", *ACM Computing Surveys*, Vol 25, No. 3, September 1993.

[6] J. M. Orduna, V. Arnau, J. Duato, "Characterization of Communications between Processes in Message-Passing Applications", in the *Proceedings of International Conference on Cluster Computing*, Cluster'00, November 2000.