Tạp chí Tin học và Điều khiển học, T. 17, S. 3 (2001), 33–40

A COMPARATIVE STUDY ON PERFORMANCE OF MPICH, LAM/MPI AND PVM ON A LINUX CLUSTER OVER FAST ETHERNET

NGUYEN HAI CHAU

Abstract. Cluster computing provides a distributed memory model to users and therefore requires message-passing protocols to exchange data. Among message-passing protocols (such as MPI, PVM, BSP), MPI (Message Passing Interface) and PVM (Parallel Virtual Machine) are adopted as most popular protocols for distributed memory computing model. In this paper, we give a practical comparative study on the performance of MPICH 1.2.1, LAM/MPI 6.3.2 and PVM 3.4.2 implementations of the MPI and PVM protocols, on a Linux cluster over our Fast Ethernet network. We also compare some parallel applications' performance runging over the three environments.

Tóm tắt. Cụm máy tính cung cấp cho người sử dụng một môi trường tính toán theo kiểu bộ nhớ phân tán, do đó cần có các giao thức chuyển thông điệp để trao đổi dữ liệu. Trong số các giao thức chuyển thông điệp (ví dụ MPI, PVM, BSP), MPI và PVM là các giao thức được sử dụng nhiều nhất. Trong bài này, chúng tôi đưa ra sự so sánh hiệu năng của các phần mềm cài đặt các giao thức MPI và PVM: MPICH 1.2.1, LAM/MPI 6.3.2 và PVM 3.4.2 trên cụm máy tính Linux được kết nối qua mạng Fast Ethernet.

1. INTRODUCTION

In recent years, cluster computing has been growing quickly because of low cost of fast network hardware equipments and workstations. Many universities, institutes and research groups started to use low cost clusters to meet their demands of parallel processing instead of expensive supercomputers or mainframes [1, 4]. Linux clusters has been increasingly using today due to their free distribution and open source policy. Cluster computing provides a distributed memory model to users/programmer and therefore requires message-passing protocols for exchanging data. Among message passing protocols such as MPI [6], PVM [15], BSP [13]... MPI (Message Passing Interface) and PVM (Parallel Virtual Machine) are most widely adopted for cluster computing. Two implementations of MPI, MPICH [7] and LAM/MPI [5], are most widely used. MPICH comes from Argonne National Laboratory and LAM/MPI is maintained by the University of Notre Dame. PVM's implementation Oak Ridge National Laboratory (ORNL) is also popular. The software can be ported to many different platforms and acted as cluster middleware, over which parallel compilers for parallel languages such as HPF, HPC++ can be implemented.

Due to greet requirements of large parallel applications, network traffic in computer cluster is increasing heavily. Therefore performance of cluster middleware is one of important factors that affect performance parallel applications running on clusters. Since PVM, LAM and MPICH all use TCP/IP to exchange messages among nodes of a cluster, it is useful to investigate PVM, LAM and MPICH performance together with TCP/IP performance to assist one make a right choice his/her cluster configuration.

In this paper, we will practically evaluate performance of MPICH 1.2.1, LAM/MPI 6.3.2 and PVM 3.4.2 on Linux Cluster if Institute of Physics, Hanoi, Vietnam in terms of latency and peak throughput. To conduct performance tests, we use NetPIPE, a network protocol independent performance evaluation tool [12], developed by Ames Laboratory/Scalable Computing Lab, USA. We also compare performance of some parallel applications running over the three cluster middleware packages. The remaining parts of this paper are organized as follows: In Section 2, we give a brief description of computer cluster architecture and some cluster middleware. In Section 3 we describe our testing environment. Results evaluation will be given in Section 4. In the last section, we provide conclusions and future works.

2. CLUSTER ARCHITECTURE AND CLUSTER MIDDLEWARE

2.1. Cluster architecture

As shown in Fig. 1, a cluster is a type of parallel and/or distributed processing system consisting of many stand-alone computers (or nodes) connected together via network so that all of them can be seen and worked as a single virtual computer. A cluster usually contains the following components:

- Computers.
- Operating systems such as Linux, FreeBSD.
- High speed network connections and swithches such as Ethernet, Fast Ethernet, Gigabit Ethernet, Myrinet.
- Network interface cards.
- Communication protocols and services such as TCP/UDP/IP, Active Message, VIA.
- Cluster middleware, including parallel interconnection software and job scheduling software such as MPI, PVM, BSP.
- Parallel programming environments and tolls such as parallel compilers, debuggers, monitoring, benchmarking tolls such as ADAPTOR, XMPI, XPVM, XMTV, LinPACK.
- Applications, including serial and parallel ones, for example Molecular Dynamic Simulation.

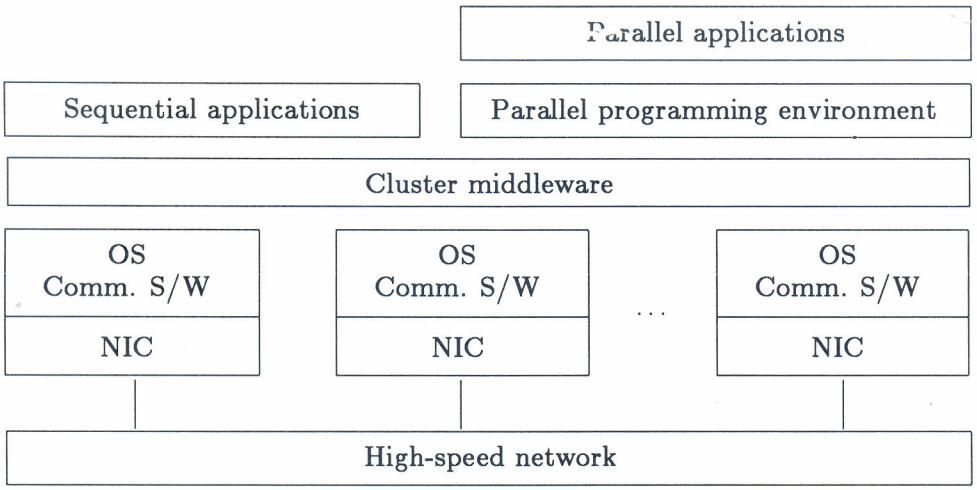

Fig. 1. Cluster architecture

Since parallel and distributed applications consume much of cluster, especially network bandwidth, 100Mbps network connections is required for cluster computing and higher bandwidth such as 1Gbps or more is highly recommended.

2.2. Cluster middleware

In this section we give a short overview of message passing packages PVM, LAM and MPICH.

The Parallel Virtual Machine was developed at Oak Ridge National Laboratory (ORNL) to handle message passing on heterogeneous distributed environment. In addition to providing a message passing mechanism, PVM provides resource management, signal hadling and fault tolerance that help build and environment for parallel processing. PVM's implementation and interface is mainly developed at ORNL. However commercial implementations of PVM are avaible.

The Message Passing Interface (MPI) Forum has been meeting since 1992 and is included high performance professional and over 40 organizations. MPI's aim is to develop a message-passing interface that meets user's demand on a common interface for parallel machines. MPI separates the interface and the implementation; therefore many vendors such as IBM, Cray Research, SGI,

Hewlett-Packard and others supported it. In addition, there are competing implementations of MPI for cluster environment, among which, MPICH anf LAM/MPI are most popular choices.

The MPI Chameleon (MPICH) began in 1993. It was developed at Argonne National Laboratory as research project to provide features making MPI simple on different types of hardware. To do this MPICH implements MPI over and architecture independent called Abstract Device Interface (ADI).

The Local Area Multi-computer (LAM or LAM/MPI) was launched at Ohio Supercomputing Facility and now maintained by University of Notre Dame. LAM is a package that provides task scheduling, signal handling and message delivery in a distributed enviroments.

The following are features summary of the three message passing packages [11].

Table 1. LAM/MPI, MPICH and PVM features

Feature	LAM/MPI	MPICH	PVM
Spawn method	user daemon	rsh	user daemon
Startup command	mpirun	mpirun	pvm
Spawn command	MPI_Spawn()	N/A	pvm_spawn()
UDP communication	default	No	default
UDP packet size	approximately 8K	N/A	< 4K (settable with pvm_setopt)
TCP communication	-c2c (mpirun option)	default	PvmDirecRoute (pvm_setopt)
TCP packet size	maximum	maximum	approximately 4K
Homogeneous mode	-O (mpirum option)	automatic	PvmDataRaw (pvm_initsend)

3. Testing Environment

Our testing environment for performance comparsion consists of 6 Intel Pentium III at 600 MHz with 64MB RAM and 4GB hard disk. Each computer connects to 3COM 10/100 Mbps auto sensing swith (24 ports) by a RealTek 10/100 auto sensing NIC and a category 5 cable. The computers are also connected back-to-back for TCP/IP verus MVIA [8] additional performance testing. In this test, we used 2 Intel Pro 10/100 NICs since MVIA did not support RealTek. All of the computers are installed RedHat Linux 6.2 with 2.2.14 kernel.

During the test, we isolated tested computers from the network to get accurate results. LAM/MPI 6.3.2, MPICH 1.2.1 and PVM 3.4.2 are installed on a fileserver computer and the other using NFS to access the software. This type of installation may cause delay when starting applications/tests but does not affect performance during the run-time since LAM/MPI, MPICH and PVM access to executable file only in launch time of applications. We use NetPIPE 2.4 [12] as network performance testing tools. NetPIPE 2.4 supports testing in TCP/IP, LAM/MPI, MPICH and PVM environments.

4. PERFORMANCE COMPARSION

In this part, we give experiment comparisons of LAM/MPI, MPICH and PVM performance in testing environment described above in point-to-point communication manner. We use two important parameters for network performance evaluation. The first one is throughput and the other one is latency. Throughput is expressed in megabit per second (Mbps) and block size is reported in bytes. This kind of unit is very common among network vendors. The throughput graphs show us maximum throughput of the network for each size of block to transfer. Network signature graph is drawn using throughput in Mbps versus total time to transfer data blocks during the test. In testing tool NetPIPE, latency is computed by the time to transfer 1 type from source to destination computer and therefore its graph representation is first point of the nerwork signature graph. In both throuhgput and signature graphs, horizontal scale is represented in logarithmic style.

During the test, we defined the following factors that affect results: TCP/IP maximum transmission unit (MTU), maximum message size of PVM, LAM/MPI and MPICH, modes of transmission of PVM, LAM/MPI and MPICH. LAM/MPI and MPICH implemented the same protocol for transfer short and long messages but their implementations are quite different. In LAM/MPI, a short message is sent to its destination together with a message's header. A long message is fragmented into several smaller packets and the first packet contains header. Sending node transmits first packet then wait for acknowledgement for receiver and continue transmitting other packets. The receiver will answer an acknowledgement to sender when it receives appropriate packets. LAM supports two modes of communicating. The first one is C2C (client-to-client) and the other is LAMD (LAM daemon). C2C allows processes to exchange data directly without notifying LAM daemon process in contrast with LAMD mode. In MPICH, a short message is sent directly without concerning whether the receiving node is expecting data or not. Thus the transmitted data may be buffered at receiver's buffer. There are two protocols are implemented in MPICH for long messages. In the first protocol, data is only transmitted to destination on receiving node's demand. In the second one, data from sender's memory is read directly by the receiver.

Fast Etherne't design is to speed up network and keep as much of the Ethernet specifications as possible, therefore its MTU is 1500, same as Ethernet's. Since TCP/IP gives best performance with MTU size 1500 as shown in Fig. 2, we set MTU at this value for all further tests.

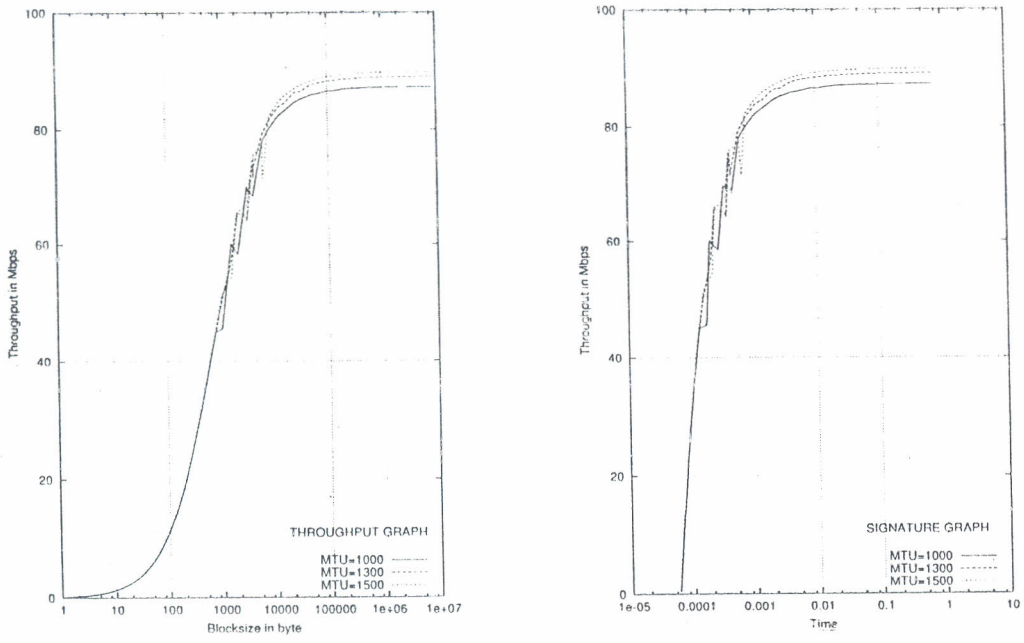

Fig. 2. TCP/IP performance verus MTU size

Since changing long/short message threshold among 64KB, 128KB and 256KB does not improve performance of LAM/MPI and MPICH, we implemented the two packages with their default parameters (LAM/MPI: 64KB, MPICH: 128KB). PVM was also implemented with its defaults. An example of performance of LAM/MPI versus short message size is shown in figures 3 and 4.

Figure 5 and Table 2 show the performance comparison for LAM/MPI, MPICH and PVM with their default implementation. LAM/MPI gives lowest latency and highest peak throughput. Latency and peak throughput of PVM and MPICH is mostly similar. The three packages work most effectively at message size from 32KB to 64KB. For larger messages, their performances are all reduced.

Since LAM/MPI can run in two modes - LAMD and C2C, we also made comparison for the two.

LAM/MPI's performance is better in C2C mode as shown in Fig. 6.

We also did the above tests with an Intel 10/100 hub (8 ports) and found that the letency of LAM/MPI, MPICH and PVM was increased by 15-16% and their peak throughput was reduced by 10% in compare with the test conducted with the switch.

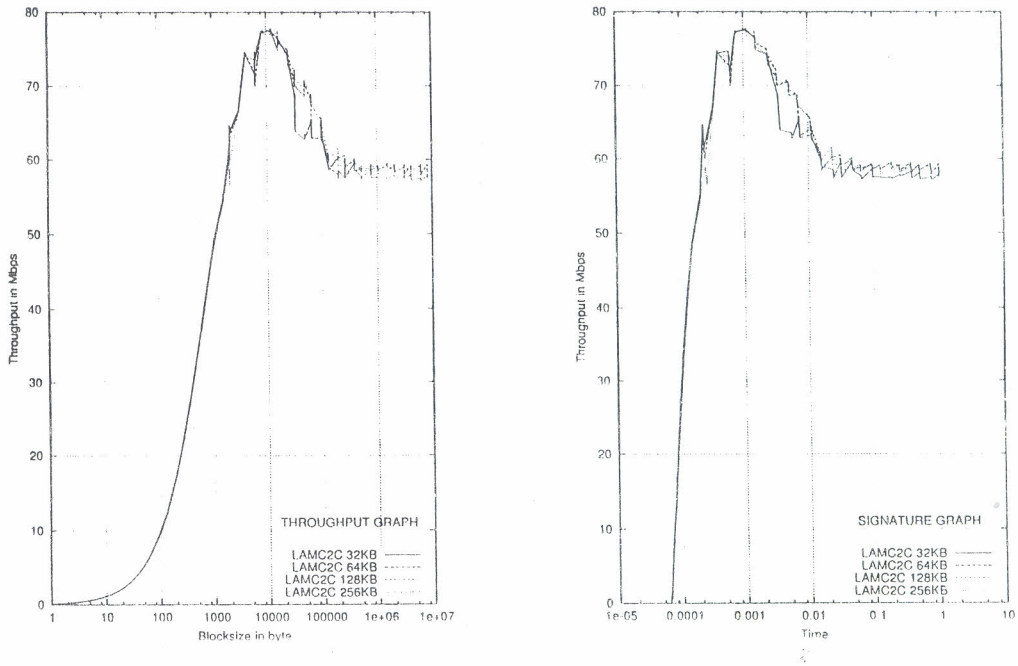

Fig. 3. LAM/MPI performance in C2C mode versus short message size

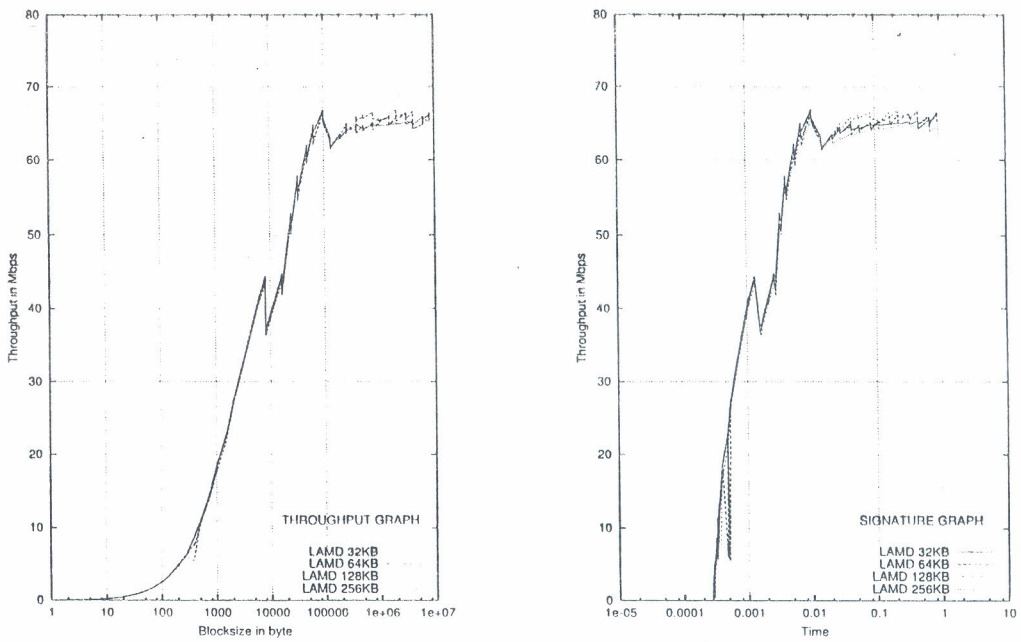

Fig. 4. LAM/MPI performance in LAMD mode versus short message size

Table 2. Performance comparison of LAM/MPICH and PVM

Criterion	LAM/PMI	MPICH	PVM
Latency (μs)	61	98	94
Peak throughput (Mbps)	77.44	75.58	75.41

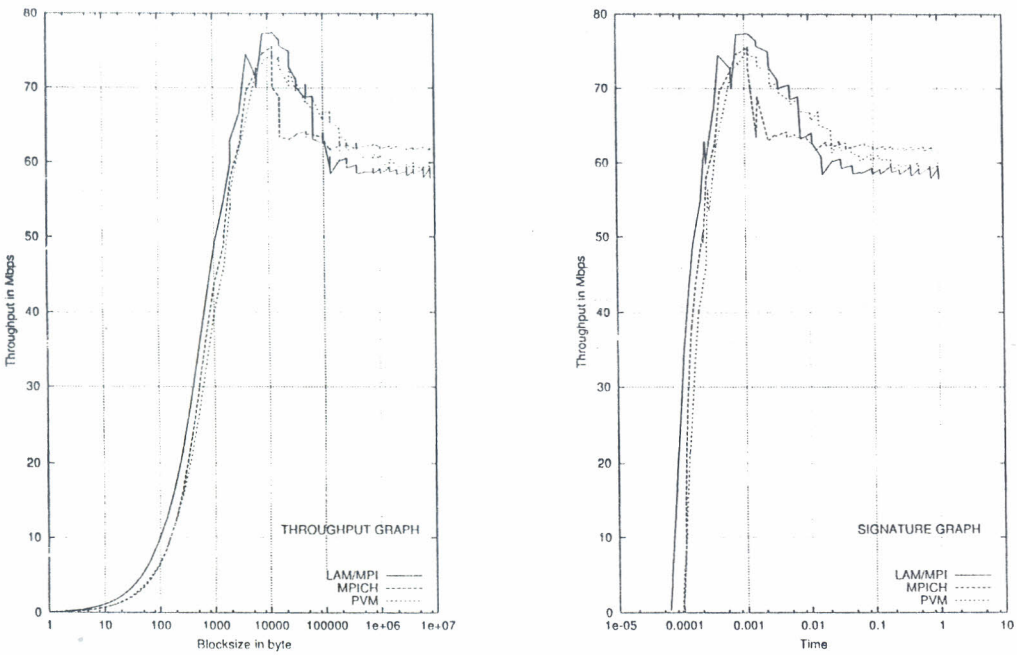

Fig. 5. Performance comparison of LAM/MPI, MPICH and PVM

In comparison MVIA and TCP/IP, we installed and did test program of MVIA 1.0a6 package. MVIA has better performance than TCP/IP. Its latency is 36 μs and peak throughput is 93.92 Mbps in comparison with 56 μs and 89.76 Mbps of TCP/IP, respectively.

Lastly, we will present performance comparison of some simple parallel applications running in LAM/MPI, MPICH and PVM over 6 nodes of the cluster. These applications are written in High Performance FORTRAN (HPF) and we use ADAPTOR 7.0 [2] as compilation system. This compiler can work over top of LAM/MPI, MPICH and PVM.

Table 3. Simple applications comparison

Application	LAM/MPI	MPICH	PVM
Pi calculation	17s	33s	23s
Matrix multiply	72s	71s	71s
Prime numbers count	16s	31s	20s

Since the PC cluster of the Institute of Physics could not be used for exclusive test, we used two computers running connected by Intel Pro 10/100 via Intel hub 10/100 for testing a real parallel application. It is a molecular dynamics simulation program written in FORTRAN 77 with library calls to MPI. We ran this application with 10000 iterations. Performance of the application running with LAM/MPI and MPICH is shown in Table 4.

Table 4. A molecular dynamics simulation's performance

Number of molecular	LAM/MPI	MPICH
H_2O: 246, Na^+: 5, Cl^-: 5	70 minutes	71 minutes
H_2O: 1230, Na^+: 25, Cl^-: 25	1220 minutes	1227 minutes

To summarize our comparisons, we have the followings:

Table 5. Overall comparison on performance of LAM/MPI, MPICH and PVM

Criterion	LAM/MPI	MPICH	PVM
Peak throughput (Mbps)	77.44	75.58	75.41
Latency (μs)	61	98	94
Throughput (Message size $\leq$ 64KB)	Very good	Good	Good
Throughput (Message size $\geq$ 64KB)	Good	Very good	Good

- Large TCP short message size increases LAM/MPI, MPICH and PVM performance slightly over Fast Ethernet.
- LAM/MPI gives best latency and peak throughput. LAM/MPI performance in C2C mode is better than that in LAMD mode.
- MPICH is better for applications exchanging large message frequently than LAM/MPI and PVM, for example applications written in HPF using REDISTRIBUTE directive frequently [2].
- VIA is much better than TCP/IP in terms of peak throughput and latency.

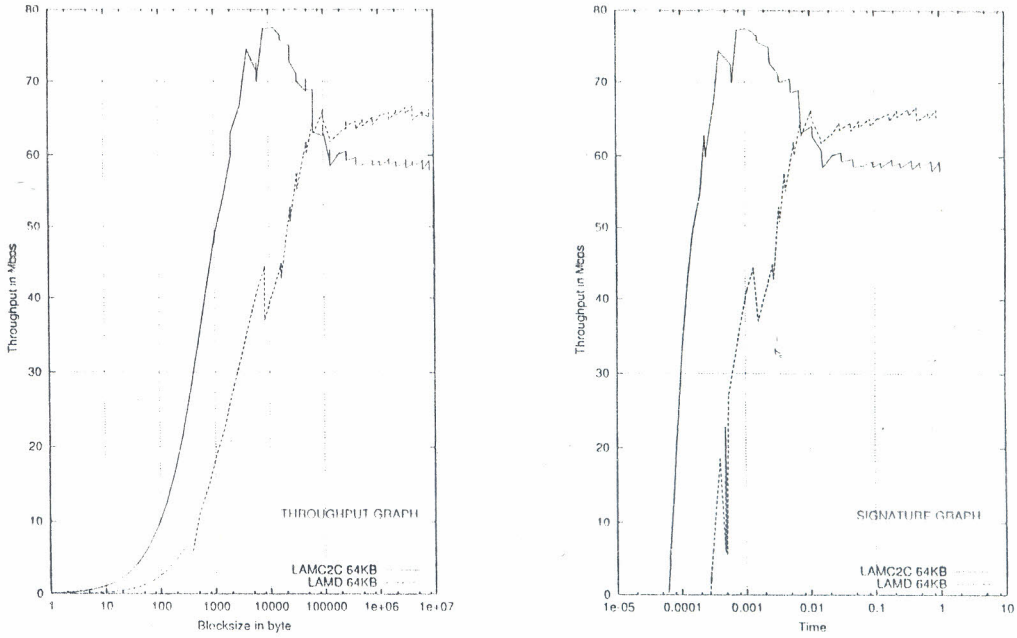

Fig. 6. LAMD and LAMC2C modes in comparison

5. CONCLUSIONS

In this paper, we presented a performance comparison LAM/MPI, MPICH and PVM - the most polular message passing packages for distributed memory computers. Since making a choice for one's

cluster is a difficult task because of the present of many software packages for cluster computing, this study may help for people who want to design and implement a Linux cluster for parallel computation in providing decision. As a result of performance testing, we has been running LAM/MPI 6.3.2 for Linux PC cluster in Institute of Physics, Hanoi, Vietnam because of its low latency and highest peak throughput in comparison with MPICH and PVM. In addition, by practice aspect, we found that LAM/MPI launches ends and clears up its parallel applications more quickly than MPICH does.

Our future works can be expressed as follows. The cluster of Institute of Physics will be used for scientific computing such as particle physics, high-energy physics and molecular dynamics simulation. Thus benchmarking the cluster with NPB [9] (NAS Parallel Benchmark) and LinPACK [16] is important. Due to great demands of parallel applications, there are many efforts to improve TCP/IP performance. However the TCP/IP improvement is in moderate progress as its delay while data passing layers of protocol stacks and seems not to meet large parallel applications' requirements. VIA (Virtual Interface Architecture) is developed recently for speed up communication ability in cluster and obtained promising results by bypassing protocol tacks to reduce data transfer delay. We will conduct performance comparison of parallel applications in LAM/MPI, MPICH and MVICH, an implementation of MPI over MVIA.

Acknowledgements. The author wishes to thank Prof. Ho Tu Bao (JAIST), Dr. Ha Quang Thuy (Vietnam National University, Hanoi) and Dr. Nguyen Trong Dung (JAIST) for their supports and advices.

REFERENCES

[1] A. Apon, R. Buyya, H. Jin, J. Mache, Cluster Computing in the Classroom: Topics, Guidelines, and Experiences, http://www.csse.monash.edu.au/~rajkumar/papers/CC-Edu.pdf.
[2] ADAPTOR - GMD's High Performance Fortran Compilation System, http:www.gmd.de/SCAI/lab/adaptor/adaptor_home.html.
[3] Foster, J. Geisler, W. Gropp, N. Karonis, E. Lusk, G. Thiruvathukal, S. Tuecke, Wide-area implementation of the Message Passing Interface, *Parallel Computing* **24** (1998) 1734–1749.
[4] K. A. Hawick, D. A. Grove, P. D. Coddington, M. A. Buntine, *Commodity Cluster Computing for Computational Chemistry*, DHPC Technical Report DHPC-073, University of Adelaide, Jan. 2000.
[5] LAM/MPI Parallel Computing, http:/www.mpi.nd.edu/lam.
[6] MPI Forum, http://www/mpi-forum.org/docs/docs.html.
[7] MPICH - A Portable MPI Implementation, http://www-unix.mcs.anl.gov/mpi/mpich.
[8] M-VIA: A High Performance Modular VIA for Linux, http://www/nersc.gov/research/FTG/via.
[9] NAS Parallel benchmark, http://www.nas.nasa.gov/NPB.
[10] P. Cremonesi, E. Rosti, G. Serazzi, E. Smirni, Performance evaluation of parallel systems, *Parallel Computing* **25** (1999) 1677–1698.
[11] P. H. Carns, W. B. Logon III, S. P. McMillan, R. B. Ross, An Evaluation of Message Passing Implementations on Beowulf Workstations, http://parlweb.parl.clemson.edu/~spmcmil/aero99/eval.htm.
[12] Quinn O. Snell, Armin R. Mikler, and John L. Gustafson, NetPIPE: A Network Protocol Independent Performance Evaluator, http://www/scl/ameslab.gov/netpipe/paper/full.html.
[13] S. R. Donaldson, J. M. D. Hill, D. B. Skillicorn, BSP clusters: High performance, reliable and very low cost, *Parallel Computing* **26** (2000) 199–242.
[14] The Beowulf project, http://www.beowulf.org.
[15] The PVM project, http://www.epm.ornl/gov/pvm.
[16] Top 500 clusters, http://www/top500clusters.org.

Received March 19, 2001

Institute of Physics, NCST of Vietnam