ELSEVIER

Parallel Computing 23 (1997) 783–812

PARALLEL COMPUTING

Practical aspects and experiences

Express versus PVM: A performance comparison

Ishfaq Ahmad *

Department of Computer Science, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong

Received 27 July 1994; revised 2 February 1996; accepted 25 October 1996

Abstract

Due to the increasing popularity of networked clusters of workstations and the need for portability across various parallel and distributed platforms, a number of programming environments have been proposed to develop parallel programs. Express and PVM are two such commonly used environments that are available on most commercial parallel computers as well as a variety of clusters of workstations. Programs developed under Express are portable, that is, a program developed on one hardware platform can run on another platform without any significant modification (provided Express is available on both platforms). PVM provides a similar portability and is particularly suitable for heterogeneous systems. In this paper, we make an experimental performance comparison of Express and PVM. The comparison is done by evaluation of their performance through benchmarking on three platforms: an Intel iPSC/860 hypercube parallel computer, a cluster of SUN workstations connected by an Ethernet, and a cluster of HP workstations connected by an FDDI ring. The performance measures include the timings of various communication primitives coded with Express and PVM. The results of Express and PVM on the iPSC/860 are also compared with the equivalent implementations using the NX message-passing library of the iPSC/860. To make a comparison from the applications point of view, we have also benchmarked a suite of various applications including three different versions of Gaussian elimination, and the *N*-body problem. The performance results also enable us to compare three different hardware platforms. While it is not the purpose of this study to make a qualitative judgement on Express and PVM, we highlight their usefulness and provide an overview of their programming styles and main features.

Keywords: Benchmarking; Express; Hypercube computers; Interprocessor communication; Parallel algorithms; Parallel programming; Performance evaluation; PVM

* Tel.: +852-2358-6980; fax: +852-2358-1477; e-mail: iahmad@cs.ust.hk.

PII S0167-8191(96)00087-7

1. Introduction

In the recent years, we have witnessed an unprecedented growth of parallel computing hardware platforms. Among such a large repertoire of hardware platforms, software designers desire to have the benefit of portability so that the code developed for one platform does not have to be rewritten or modified for the other. At the same time, advancements in the design of processor architecture and communication mediums have resulted in the emergence of fast workstations connected by high-speed communication networks [4,5]. These clusters of workstations also known as workstation farms are approaching the speed of some of the contemporary parallel computers. Recent studies have shown that clusters of workstations have the potential of solving very large-scale problems [6]. A number of programming environments for such platforms have recently emerged. These software environments can simultaneously exploit the potential power of diverse parallel and distributed hardware platforms and provide portability across them. They can simulate a cluster of workstations as a virtual parallel computer, and can perform communication across multiple homogeneous and heterogeneous parallel machines.

This paper compares two currently popular parallel programming environments. The first is Express which is a commercial product from Parasoft Corporation [21]. The second is PVM which is available in the public domain [24]. We call them 'environments' because they are neither operating systems nor languages, rather they allow the programs to be written using standard C or Fortran. As elaborated in Refs. [13,25], both Express and PVM are more than just message-passing libraries since they provide a broad set of tools and utilities that are vital for full exploitation of computing and networking resources.

A number of other similar environments have also been proposed, including Linda [2,11], P4 [10], PICL [15], Zipcode [23] and more recently MPI [18]. A detailed survey of software environments for networked systems can be found in [26] while an overview of recent developments for message-passing techniques for both parallel and distributed systems can be found in [18].

A key measure of the usefulness of programming environments like Express and PVM is the speed of the basic primitives used frequently in parallel programs. We have evaluated the performance of some of the basic primitives of Express and PVM running on an iPSC/860 parallel computer, a cluster of HP workstations connected by an FDDI network, and a cluster of SUN/IPX workstations connected by an Ethernet. The results from this evaluation can be useful in a number of ways.

- They help us assess these environments against each other to find out which one performs better.
- They reveal the amount of the extra overhead incurred due to portability, when compared to the host operating system such as NX.
- Algorithm developers and parallel compiler writers can benefit from these results by understanding the overhead incurred by basic communication primitives on various platforms and can, therefore, estimate the performance of various libraries [1].
- The availability of the exact timings of various communication patterns under

different platforms can assist in making better problem partitioning and scheduling decisions.

- These timings enable us to compare the performance of the iPSC/860 parallel computer to workstation clusters using different networks and shed some light on the trade-offs between performance and cost.

The rest of this paper is organized as follows. In the Section 2, we provide some background of Express and its utilities. In section 3, we provide an overview of PVM and its utilities. In Section 4, we give a summary of the hardware platforms used in our experiments. In Section 5, the performance results of the communication primitives are given. The timings of the application benchmark suite are presented in Section 6 and the Section 7 concludes this paper.

2. Overview of express

Express is developed by a group of researchers who started Parasoft Corporation. It can be used to write parallel programs on a variety of parallel machines including the CM5, NCUBE, Intel iPSC/2 and iPSC/860 hypercubes, Intel Paragon and transputer arrays [20,21]. The network version of Express allows a network of workstations to be used as a 'virtual parallel machine'. Workstations that Express can run on include DEC, HP, IBM/RS6000, SCI and Sun.

The history of Express can be traced back to the Caltech/JPL machines developed in the early eighties [12,13]. At that time, the simplest model of a parallel or distributed computation was the one in which a 'master' process takes the responsibility of creating a number of 'worker' processes which perform the computations required to produce the overall desired output. In Express, this style of programming is called the host–node programming model where the 'master' is the host while the 'worker' is the node. In order to avoid the user to write the host programs, a host-free programming model, called Cubix, was also developed which provided an I/O system for opening, reading and writing files and interfacing with the user. Later, an operating system known as 'multitasking, object-oriented, operating system' (MOOSE) was built to support multitasking, remote task creation, scheduling and a number of other housekeeping functions. Moreover, a system called 'crystal router' was developed to support more efficient concurrent communication [14]. Eventually, the combined research efforts at Caltech resulted in an integrated software package now called Express. As shown in Fig. 1, there are three implementation layers of Express.

- The lowest level consists of utilities for controlling the hardware, such as the allocation of processors, loading of programs, etc.
- The medium level provides support for problem partitioning. It also allows communication among the nodes and between the node and the control processor.
- The highest level contains facilities for node programs to perform I/O and utilities for access to the host operating system.

Since each level is logically distinct and built only on those below it, Express is portable to a variety of systems using the 'top-down' approach.

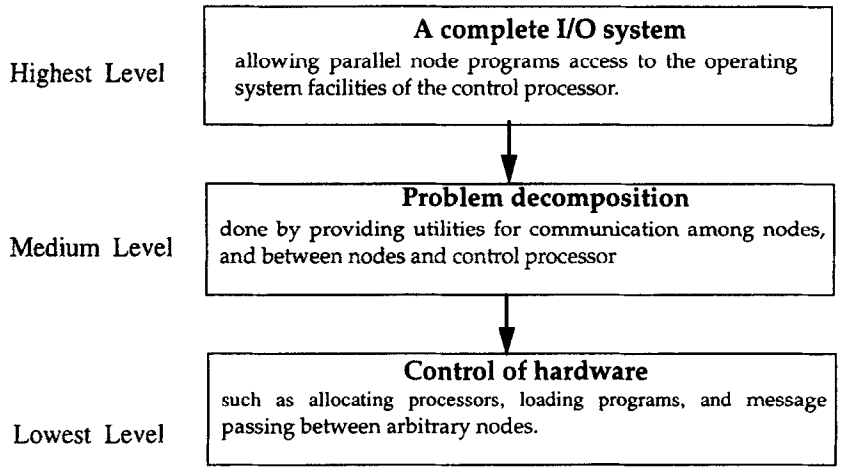


Fig. 1. Three layers of Express implementation.

A summary of the utilities provided by Express is illustrated in Fig. 2. The communication utilities include blocking and non-blocking communication among nodes, exchange, broadcast and collective communication such as reading and writing a vector. The global communication includes concatenation, global reduction operations, synchro-

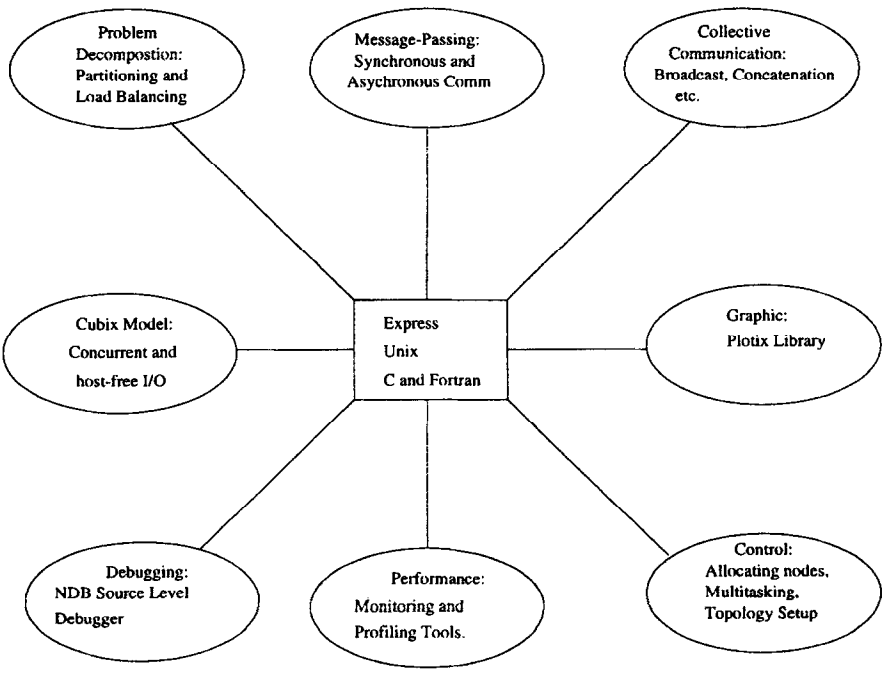


Fig. 2. The Express utilities.

nization, etc. In addition, Express provides processors control, domain decomposition tools, parallel I/O, graphics and debugging and performance analyses tools.

3. Overview of PVM

PVM is developed by researchers at Emory University, Oak Ridge National Laboratory (ORNL), University of Tennessee, Carnegie Mellon University and Pittsburgh Supercomputer Center [25]. It permits a network of heterogeneous computers to be used as a single parallel computer. PVM has evolved through many versions. The initial versions of PVM used TCP/IP sockets to implement all communication. Consequently, PVM was limited to workstation environments such as Sun3 and Sun4 SPARCstation, HP-9000 PA-RISC, IBM/RS6000, etc. However, recent versions of PVM such as the one reported in this paper have included implementation within heterogeneous parallel systems. Heterogeneous network-based computing refers to general purpose concurrent computing where:

- The hardware platform consists of a collection of computer systems of varying architectures interconnected by one or more network types such as FDDI.
- Applications are viewed as comprising several sub-algorithms, each of which is potentially different in terms of its most appropriate programming model, implementation language and resource requirements.

PVM's portability is similar to Express. The PVM computing model is illustrated in Fig. 3. Under PVM, a user defined collection of serial, parallel and vector computers appear as one large distributed-memory computer. PVM supplies the functions to automatically start up tasks on the virtual machine, and allows tasks to communicate and synchronize with each other. A task is defined as a unit of computation in PVM analogous to a UNIX process. Applications, which can be written in Fortran or C, can be parallelized by using message-passing constructs common to most distributed-memory computers. By sending and receiving messages, multiple tasks of an application can cooperate to solve a problem in parallel.

PVM supports heterogeneity at the application, machine, and network level, and can handle data conversion that may be required [16]. In other words, PVM allows application tasks to exploit the architecture best suited to their solutions. Moreover, PVM allows the virtual machine to be interconnected by a variety of different networks. PVM provides routines of packing and sending messages between tasks. The model assumes that any task can send a message to any other PVM task. The PVM communication model provides synchronous and asynchronous blocking send and receive, multicast to a set of tasks and broadcast to a user defined group of tasks. Since message buffers are allocated dynamically, the maximum size of the messages that can be sent or received is limited by the amount of available memory on a given host.

PVM supplies routines that enable a user process to become a PVM task and to become a normal process again. These routines perform functions such as adding and deleting hosts from the virtual machine, and starting and terminating PVM tasks. PVM also provides fault-tolerance: if a host fails, PVM can detect this and delete the host

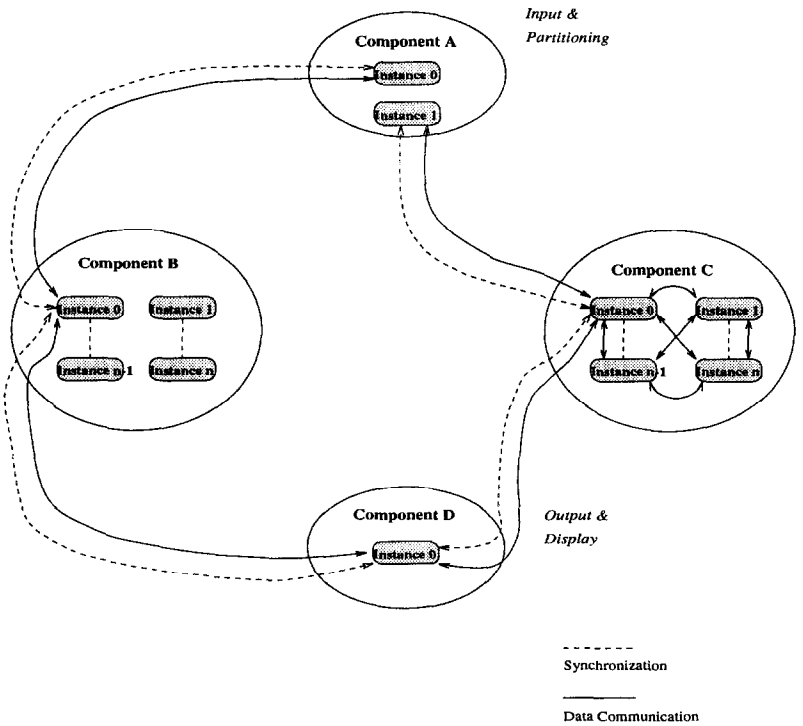


Fig. 3. The PVM computing model.

from the virtual machine. The status of host can be requested by the application and, if required, a replacement host can be added by the application.

4. Test environment

Our test environment included three platforms: an Intel iPSC/860 hypercube, an Ethernet based network of SUN workstations, and an FDDI based ring of HP workstations. All tests programs were written in Fortran. The hardware platforms are briefly described below.

Table 1
Configurations of the iPSC/860 used in the experiments

No. of nodes	32
Node CPU	i860
Clock frequency	40 MHz
Main memory/node	8 Mbytes
Express version	3.1
PVM version	3.2.0
iPSC node O/S	NX/2 (rel. 3.2)

Table 2
Configurations of SUN IPX and HP clusters

Network	SUN-IPX/Ethernet	HP/FDDI-Ring
No. of workstations	8	4
System model	4/50 SPARC IPX	735
Main memory	16 Mbytes	128 Mbytes
OS name	SUN OS 4.1.3	HP-UX 09.01
Express version	3.2.5	3.2.5
PVM version	3.2	3.2

Our first platform was the Intel iPSC/860 parallel computer at Caltech's Concurrent Supercomputing Center. This system is based on hypercube interconnection network topology and can scale up to 128 nodes. It is controlled from a host computer, called 'system resource manager' or SRM which runs System V UNIX [17]. The iPSC/860 is based on the i860 RISC microprocessor [19]. Each node has a hardware communication module, called DCM (direct connect module) that connects a node to the interconnection network [22]. Each DCM router can support up to eight channels which are bit-serial and full duplex. Table 1 gives a summary of the configuration of the iPSC/860. Some of our results using the NX library concur with those of earlier studies [7,8]. In our experiments, for small data sizes, we used 1000 repetitions for each primitive to improve the accuracy; for large messages, the number of repetitions was varied from 100 to 200.

For performance evaluation of networks of workstations, we used two platforms: The first was a cluster of 8 homogeneous SUN SPARC IPX workstations connected by 7 Ethernet segments. The second was a cluster of 4 homogeneous HP 735 workstations connected by FDDI. Three of the HP workstations are equipped with FDDI interface only. The fourth one has both FDDI and Ethernet interfaces and functions as a host-based router for the rest three to the outside network. Table 2 gives a summary of the configuration of SUN/IPX and HP clusters.

The experiments on these clusters were conducted at night and when the systems were idle. In each experiment, for small data sizes, we used about 500 repetitions for each primitive. For larger messages, the number of repetitions used was 100. In addition, each experiment was done 5 times and the average was taken across repetitions within an experiment and across experiments.

5. Communication performance

The communication tests include one-to-one communication, exchange, broadcast, global reduction operations, ring communication and complete exchange. During our experiments, we found that both Express and PVM exhibited dramatic differences in performance when compared on the three hardware platforms. In addition, we also discovered that the size of the communication data had a great impact on relative performance of all three software environments. Therefore, for each primitive, we have divided the results into two parts. The first part presents the timing results on the

iPSC/860 and the second part presents the results on the HP and IPX clusters. Furthermore, each part is presented with results on small (0 to 200 bytes), large (200 to 1000 bytes), and very large (1000 to 16000 bytes) messages. These results are provided in the following sections.

5.1. One-to-one communication

For measuring the communication speed between two nodes, we performed the standard echo test. In the echo test, the communication time between sending and receiving nodes is measured by starting a clock at the sending node and then invoking the send and receive routines to send out a message and wait for a reply. On the destination node, receive and send routines are used to echo this message back to the sending node. This process is repeated N times, and the average is taken.

The results of one-to-one communication using NX, Express and PVM on the iPSC/860 are shown in Fig. 4. The results for the IPX and HP cluster are given in Fig. 5. Fig. 4 indicates that the timings of Express and NX are relatively close for small, medium and large data sizes. PVM, on the other hand, is about 6 times slower than Express and NX for small messages. But as the message size increases, the difference between the performance of PVM and Express decreases. For example, as shown in Fig. 4(b), the difference between PVM and Express decreases steadily as the message size is increased from 200 to 1000 bytes. For very large messages, as indicated in Fig. 4(c), the timings of PVM and NX are about the same. Some timing values for various message sizes are provided in Table 3.

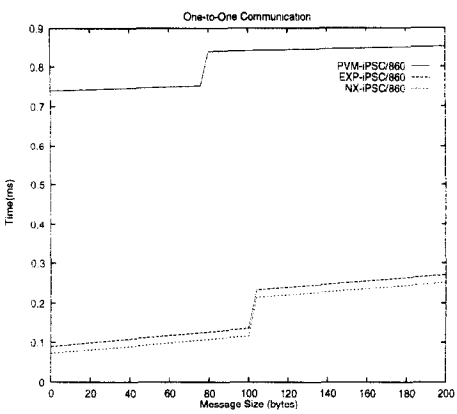
From Fig. 5, we can observe that, on the HP cluster, PVM is 4 times slower than Express. But this difference reduces to 1.5 times when the messages get larger. A similar trend can be observed using IPX workstations with Ethernet.

Notice that Express timings on the cluster of HP workstations and iPSC/860 are comparable, but PVM on the HP cluster is slower than that on the iPSC/860. Compared to the HP cluster, Express and PVM are about 2 to 3 times slower on the IPX cluster.

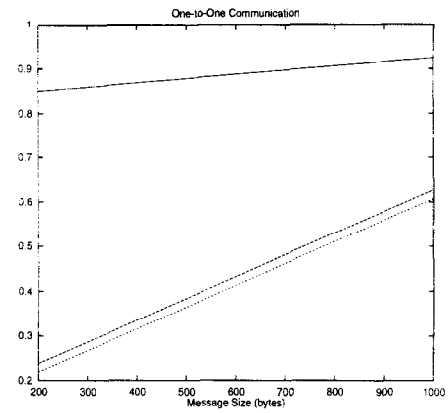
5.2. The exchange operation

The exchange function simultaneously sends a message to a node and receives a reply. The advantage of this function is that data can be read from and sent to the same or different nodes in one step. Another advantage is that data transmission in reading and writing can be overlapped. Moreover, the user is free from worrying about which node should read first and which should write. For PVM, since it does not provide the exchange operation, we implemented it by using one pvmfsend and one pvmfrecv in one node, and correspondingly, one pvmfrecv and one pvmfsend in the other node.

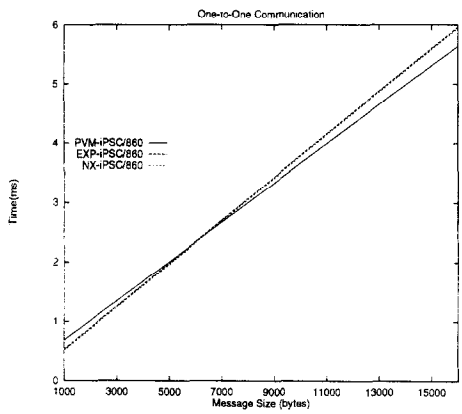
As shown in Fig. 6, on the iPSC/860, PVM is about two times slower than Express. However, as the message size increases, the difference between the timings of PVM and Express becomes smaller. From Fig. 7, we can observe the overall trend in the performance of PVM and Express on the HP and IPX clusters is similar to that of Fig. 6. For small message sizes, Express running on the IPX cluster is 4 times faster than PVM. On the HP cluster, Express is about 7 times faster than PVM. But with the larger



(a) Small messages.



(b) Large messages.



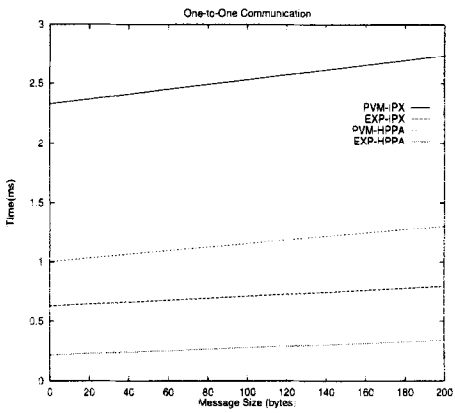
(c) Very large messages.

Fig. 4. Times for one-to-one communication using NX, Express, and PVM on the iPSC/860.

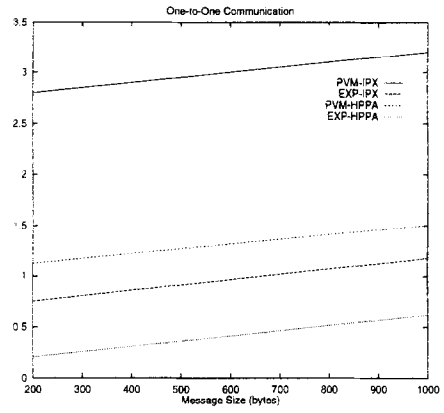
message sizes, Express is only about 2 times faster than PVM on both clusters. Express on the HP cluster is about 2 times slower than Express on the iPSC/860. Similarly, PVM on the HP cluster is 3 times slower than PVM on the iPSC/860.

5.3. The broadcast operation

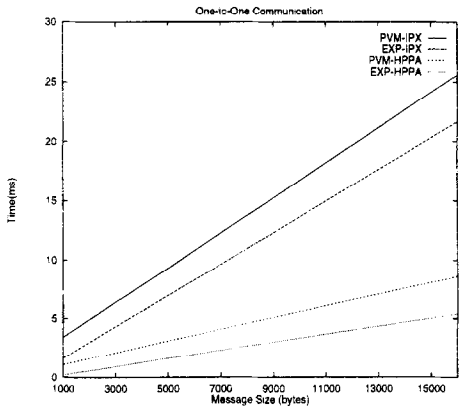
Broadcast is performed to send the same data to more than one node. It can be performed between host and nodes, or from one node to multiple nodes. It is one of the frequently used primitive and is, therefore, important for performance comparison. Both Express and PVM provide routines for broadcast. In NX, on the other hand, routines such as csend and crecv are used for broadcast purpose as well. Message can also be sent to a subcube composed of a set of nodes surrounding the source node. Express and PVM have the extra advantage of specifying arbitrarily any node(s) that should receive the broadcast message.



(a) Small messages.



(b) Large messages.



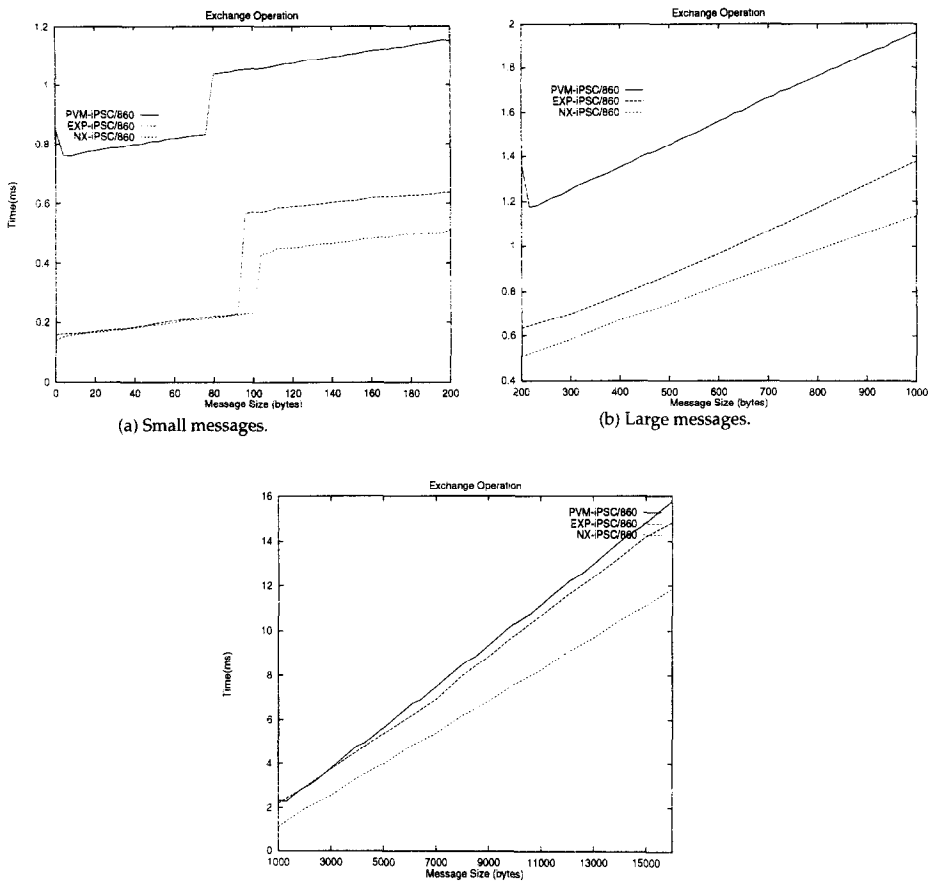
(c) Very large messages.

Fig. 5. Times for one-to-one communication using Express and PVM on the HP and IPX clusters.

The timings for the broadcast operation on the iPSC/860 using 4 processors are shown in Fig. 8. As can be noticed, the broadcast operation of PVM is inefficient and is much slower compared to both NX and Express. For small messages, broadcast

Table 3
Times required for one-to-one communication with messages of different sizes (μs)

Bytes	iPSC/860			FDDI		Ethernet	
	NX	Express	PVM	Express	PVM	Express	PVM
0	68	83	736	189.0	677	643	2225
4	74	92	741	189.7	858	647	2380
100	115	134	841	271.0	1252	754	2563
200	255	270	852	303.0	1342	820	2774
1000	566	584	929	598.0	1350	1270	3211
16000	5941	5958	5815	4570.0	9271	19352	23990



(a) Small messages.

(b) Large messages.

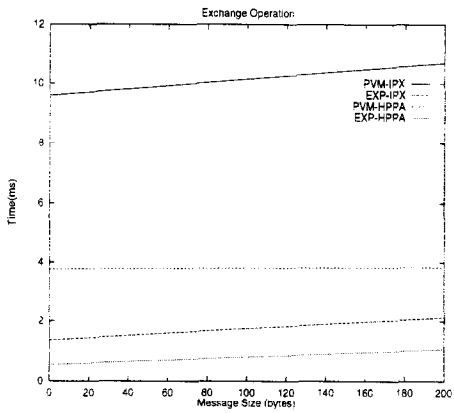
(c) Very large messages.

Fig. 6. Times for the exchange operation between nearest neighbors using NX, Express and PVM on the iPSC/860 (4 processors).

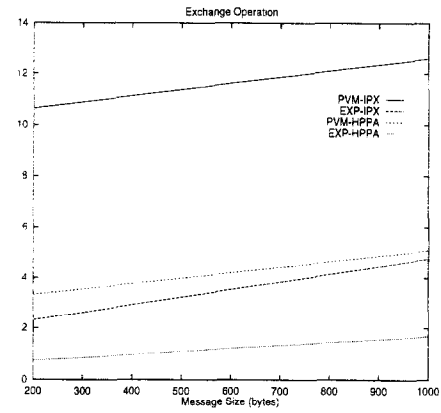
operation of PVM for 4 nodes is bout 40 times slower than that of Express and NX. However, for large messages, this difference reduces to 6–7 times.

As shown in Fig. 9, the broadcast operation of PVM on the cluster of IPX workstations is about 7 times slower than Express when the messages size is small. But for large messages, PVM is about 1.5 times slower than Express. On the HP cluster, for small message size with 4 nodes, PVM is about 4 times slower than Express. For very large messages, the timings for PVM are comparable to Express. For very large messages, PVM even outperforms Express.

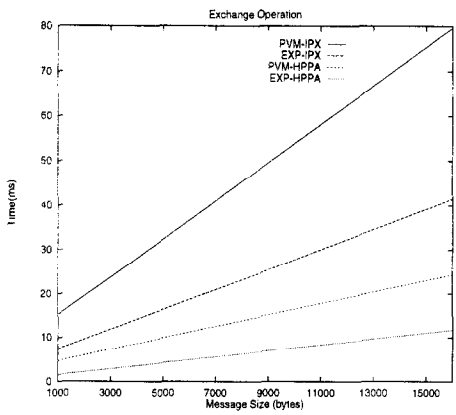
Comparing different hardware platforms, PVM on the IPSC/860 is 10 times slower than on the HP cluster when message size is small. But, this difference reduces to about 5 times for large messages. Express on the IPX performs about 10 times slower than on the iPSC/860 while on the HP it performs 2–3 times slower than on the iPSC/860, when the messages are small.



(a) Small messages.



(b) Large messages.

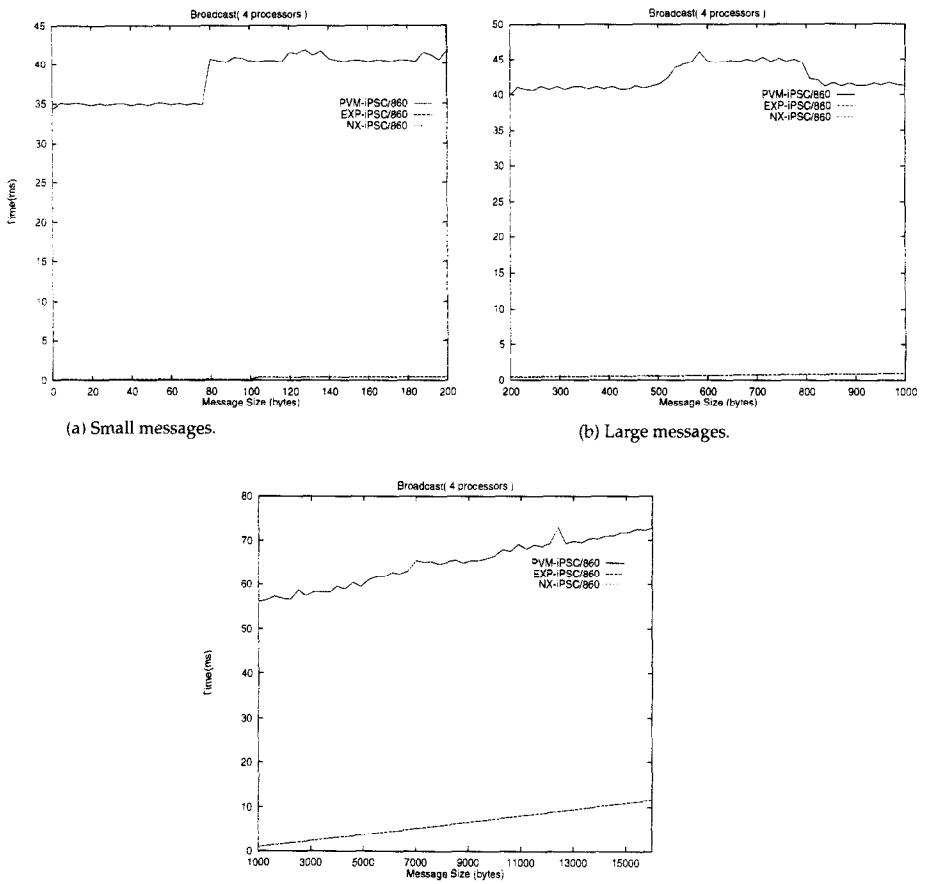


(c) Very large messages.

Fig. 7. Times for the exchange operation using Express and PVM on the HP and IPX clusters (4 workstations).

5.4. Global communication

Global communication operations are frequently required in parallel programs. In such global operations, all the nodes participate in the same operation. The reduction operations are examples that require global communication. A reduction operation takes as input a value in each processors and outputs a single value in every processors. There are various reduction operations such as *add*, *prod*, *max*, *min*, *and*, *or*, *xor*, etc. While NX provides different calls for each of these operations, Express has a unified function for performing reduction operations. But since routines for these operations are not provided by PVM, they had to be implemented. Express has an extra facility of specifying a list of nodes that should participate in the global operation. In this paper, only the results of the sum operation are given.



(a) Small messages.

(b) Large messages.

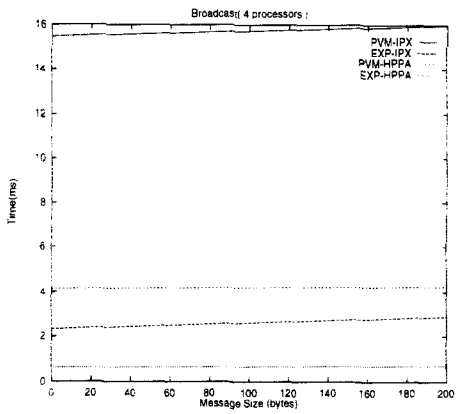
(c) Very large messages.

Fig. 8. Times for the broadcast operation using NX, Express and PVM on the iPSC/860 (4 processors).

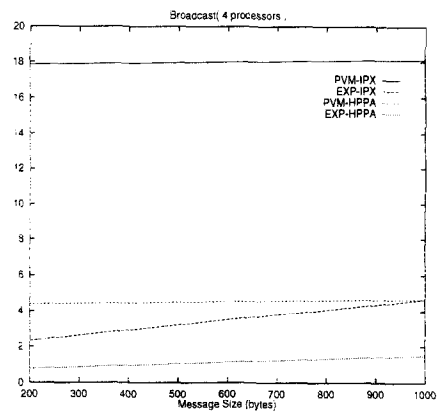
Tables 4 and 5 include times to perform the global sum on 1, 4, 16 and 26 words. The results for the iPSC/860 are repeated for 4, 8 16 and 32 processors. It can be seen that both Express and NX are much faster than PVM. We observed similar results (not reported here) for some other global reduction operations such as multiply and logical AND. The global operations of PVM do not seem to scale very well with an increase in the number of processors. Moreover, compared to iPSC/860, these operations are much slower on the clusters of HP and IPX workstations.

5.5. *The ring communication test*

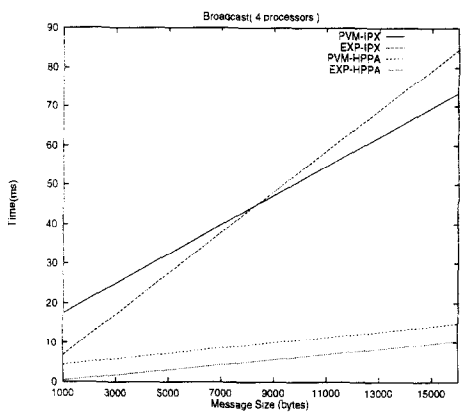
In the ring communication test, each node takes a message and circulates it through each of the other nodes in the network. This communication pattern is generated in a straight forward manner. Each node finds its 'forward' and 'backward' neighbor with whom it communicates. Each node then writes the message to its 'forward' neighbor



(a) Small messages.



(b) Large messages.



(c) Very large messages.

Fig. 9. Times for the broadcast operation using Express and PVM on the HP and IPX clusters (4 workstations).

and reads from the 'backward' neighbor. This is repeated $N-1$ times where N is the total number of nodes.

In general, the efficiency of this communication pattern depends on the underlying network topology. Determination of the 'forward' and 'backward' neighbors can be done using Express's grid utilities which can convert a physical topology to a virtual topology. On the hypercube, 'forward' and 'backward' neighbors can be easily found using gray codes for implementing the ring pattern using NX or primitives. But for the workstations cluster environment, this determination can not be done meaningfully. Hence, we can just arbitrarily determine the 'forward' and 'backward' nodes for each node, such that the message goes once through each node.

As can be seen from Fig. 10, on the iPSC/860, PVM is 1 to 2 times slower than Express and NX primitives for 4 and 8 nodes. As the message size gets larger, the

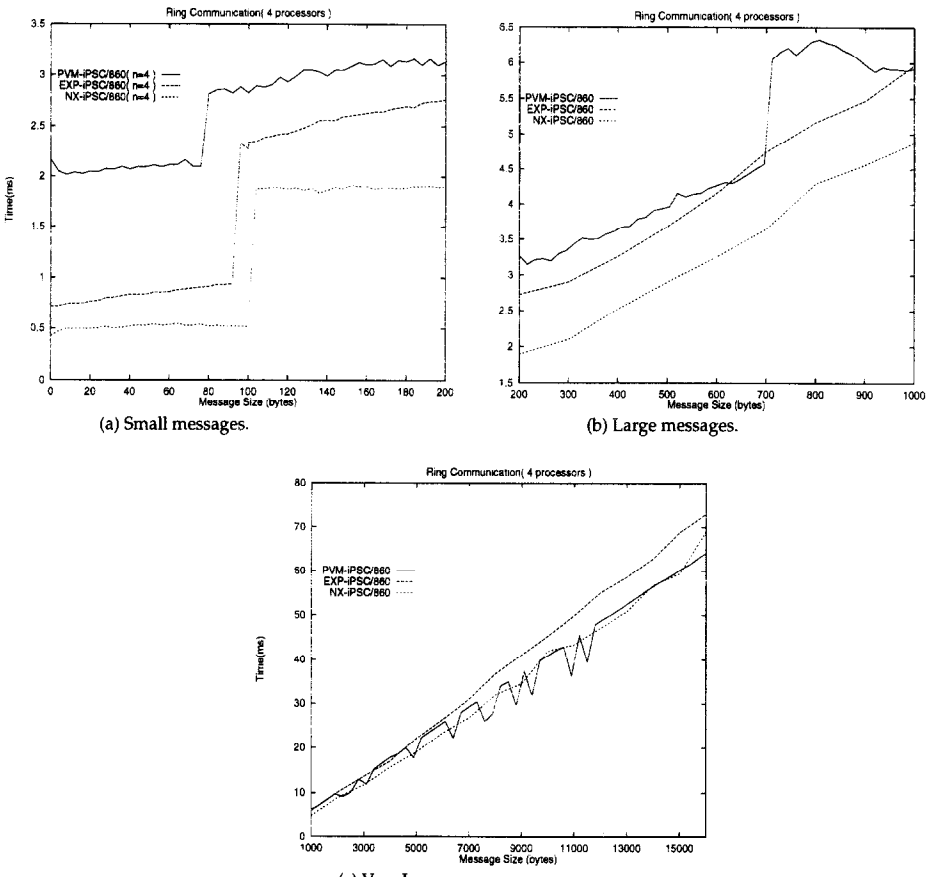
Table 4
Times for global sum (ms) on the iPSC/860

Processors	Words	NX-iPSC860	EXP.-i860	PVM-iPSC860
4	1	0.27	0.40	29.95
	2	0.27	0.41	29.76
	4	0.28	0.42	29.73
	26	0.56	1.10	33.00
8	1	0.42	0.60	47.33
	2	0.43	0.61	47.81
	4	0.43	0.64	47.99
	26	1.05	1.82	56.80
16	1	0.60	0.80	57.01
	2	0.60	0.81	55.71
	4	0.61	0.84	56.04
	26	1.72	2.48	64.82
32	1	0.77	1.03	89.78
	2	0.77	1.05	89.50
	4	0.79	1.09	86.56
	26	2.31	3.14	116.01

timings for PVM, Express and NX become almost identical. On the HP and IPX clusters, PVM is slower than Express (Fig. 11). Compared with the iPSC/860, the ring test on the HP cluster is about 2 times slower for small messages. For large messages, timings for HP cluster are comparable to those on the iPSC/860 and sometimes even smaller. The ring test on the HP cluster is 3 times faster than on the IPX cluster for small message size.

Table 5
Times for global sum (ms) on the workstation clusters

Processors	Words	Exp-HP	PVM-HP	ExpIPX	PVM-IP
4	1	4.399	7.674	3.216	22.103
	2	3.774	7.680	3.247	22.140
	4	4.114	7.806	3.272	22.118
	26	5.346	8.039	4.235	23.218
8	1	—	—	5.809	32.851
	2	—	—	6.057	33.639
	4	—	—	7.138	33.505
	26	—	—	7.901	34.838



(a) Small messages.

(b) Large messages.

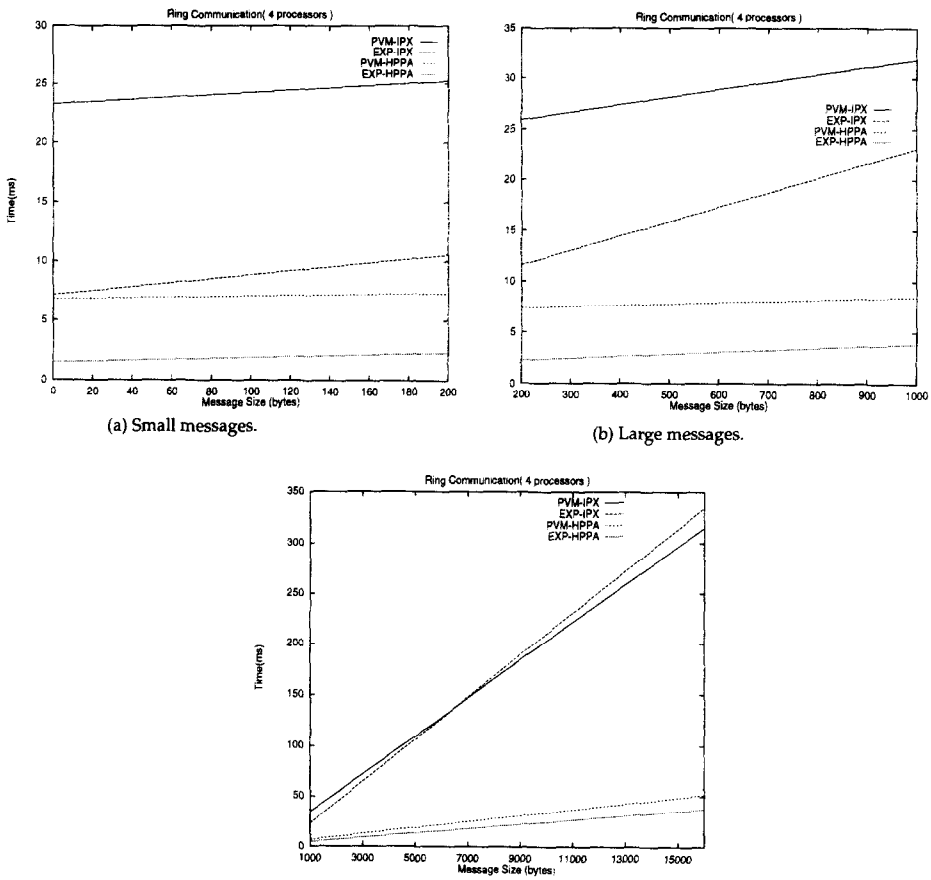
(c) Very Large messages.

Fig. 10. Times for the ring test on the iPSC/860 using NX, Express and PVM (4 processors).

5.6. The complete exchange

In the complete exchange communication pattern, which is also known as all-to-all personalized communication [9], each of the N nodes sends a different block of data to each of the remaining $N-1$ nodes. This communication pattern is equivalent to a complete directed graph. It is used in a number of algorithms including matrix transpose, matrix-vector multiply, 2-dimensional FFTs distributed table look-ups, etc. The time required to carry out the complete exchange operation is an important measure of the power of a distributed-memory parallel computer system since it is the densest communication requirement that can be implemented on a network.

The timings for complete exchange using 4 nodes using PVM, Express and NX on the iPSC/860 are given in Fig. 12. PVM is about 3 times slower than Express for small messages. For medium and large messages this difference reduces considerably. For



(a) Small messages.

(b) Large messages.

(c) Very Large messages.

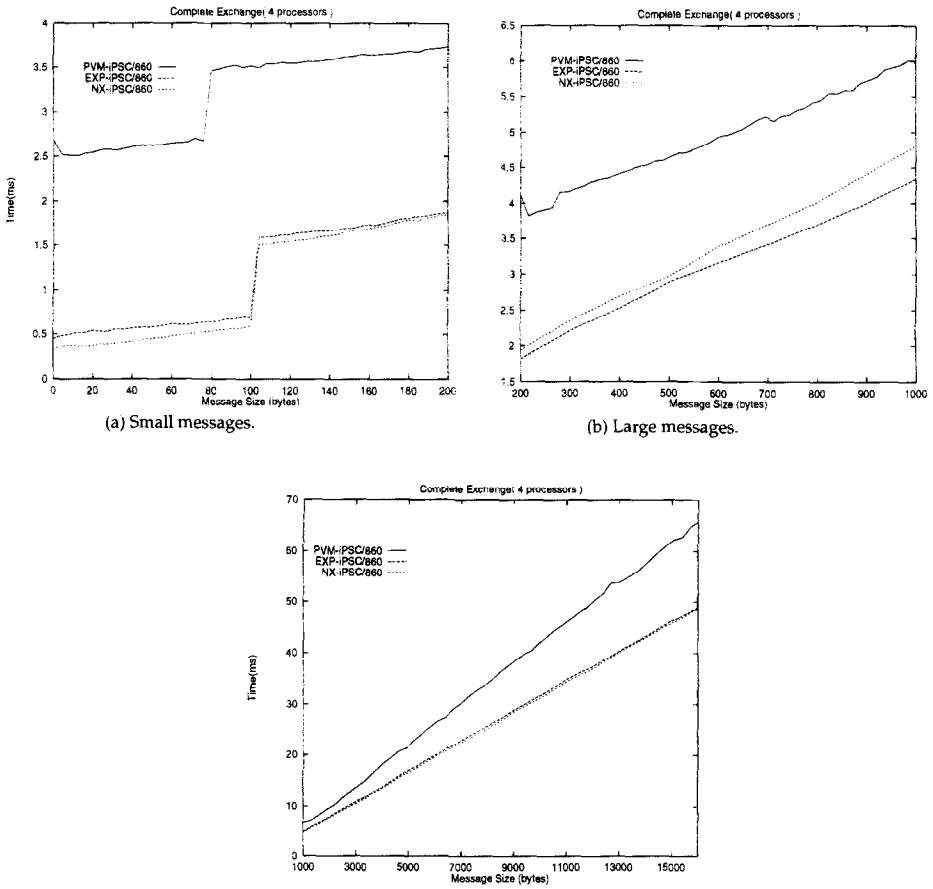
Fig. 11. Times for the ring test using Express and PVM on the HP and IPX clusters (4 workstations).

larger message sizes, timings for NX primitives and Express are about the same while PVM is about 2 times slower than the NX primitives and Express.

On the cluster of HP workstations (Fig. 13), PVM is 4 times slower than Express. However, for large messages, PVM is only 2 times slower. Comparing different platforms, for small messages, Express and PVM on the iPSC/860 are 6 times faster than on the HP cluster. For large messages, the timings difference reduces to about 2 times only. Comparing the cluster of HP and IPX workstations, for small message size, the complete exchange operation on the HP cluster is about 2 times faster than on the IPX cluster.

6. Evaluation with an application benchmark suite

To compare the performance of Express, PVM and NX with real applications, we implemented an application benchmark suite. The suite includes three different versions



(a) Small messages.

(b) Large messages.

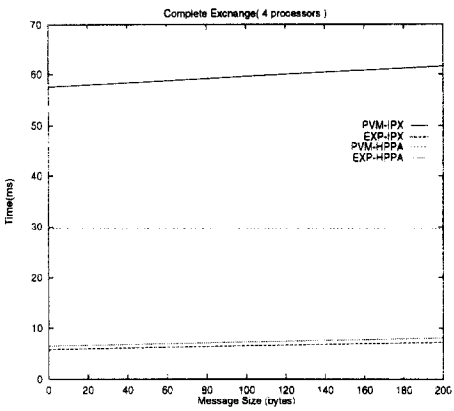
(c) Very Large messages.

Fig. 12. Times for the complete exchange test using NX, Express and PVM on the iPSC/860 (4 processors).

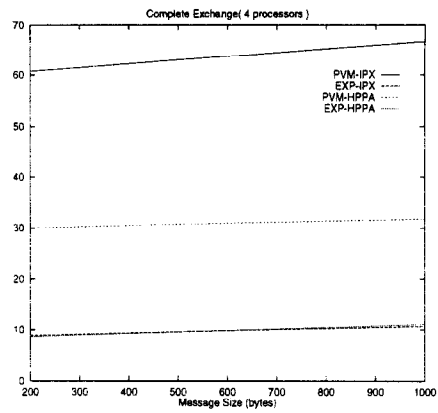
of Gaussian elimination, and the N-body problem. These applications were coded using Fortran with Express, PVM and NX primitives. The execution times were obtained with 1, 2, 4 and 8 nodes on the IPX workstations cluster, and 1, 2 and 4 nodes on the HP cluster. On the iPSC/860, we used 1, 2, 4, 8, 16 and 32 nodes. The execution time of an application was measured by taking the average across all the nodes. The results are given in the following sections.

6.1. Gaussian elimination (row–block partitioning)

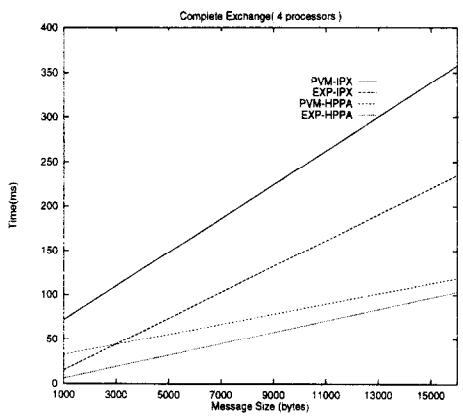
The three versions of Gaussian elimination for solving linear equations are based on partial pivoting algorithms, but with different data partitioning strategies. As a result, the algorithms used in the three versions are quite different. The first version is based on row–block partitioning, that is, an equal number of contiguous rows of the coefficient matrix are assigned to each processor. The program consists of three routines, for



(a) Small messages.



(b) Medium messages.



(c) Large messages.

Fig. 13. Times for the complete exchange test using Express and PVM on the HP and IPX clusters (4 workstations).

generating the random data, performing Gaussian elimination, and backward substitution. Only the Gaussian elimination routine was timed.

The execution times of Gaussian elimination with row–block partitioning using Express, PVM and NX primitives on different platforms are shown in Tables 6–8. These tables also include the times for serial execution of Gaussian elimination using one processor. On the iPSC/860, the serial version is implemented using NX only. On the iPSC/860, the execution times for the Express version exhibit speedup with the number of processors varied from 1 to 16. The speedup is better with a larger matrix size. However, the execution times start to increase with 32 processors. On the other hand, the NX version still yields additional speedup with 32 processors. Using the PVM version, speedup is observed with the number of processors varying from 2 to 4. However, the execution times start increasing with 8 or more processors. From Table 6,

Table 6
Timings (s) for row-block partitioned Gaussian elimination using NX, Express and PVM on the iPSC/860

Matrix size	PE = 1	PE = 2			PE = 4			PE = 8			PE = 16			PE = 32		
	NX	NX	Express	PVM	NX	Express	PVM	NX	Express	PVM	NX	Express	PVM	NX	Express	PVM
256 × 256	5.94	3.6	3.8	11.3	2.2	2.3	10.2	1.5	1.2	11.8	1.1	2.1	14.8	0.9	3.5	22.1
384 × 384	24.71	12.8	13.0	19.7	6.6	6.8	17.3	4.0	4.7	28.3	2.6	4.4	14.8	2.2	6.9	22.4
512 × 512	59.63	30.1	31.1	35.3	15.5	15.6	28.2	8.6	9.2	29.4	5.6	8.5	31.5	4.1	11.4	48.3
640 × 640	120.87	61.1	62.6	62.4	31.0	31.1	41.5	16.2	17.5	36.3	9.8	14.0	41.2	6.9	17.4	58.9
768 × 768	219.87	106.5	108.2	106.5	54.2	54.7	64.8	27.9	28.7	48.0	16.7	20.8	54.3	10.8	24.7	74.0

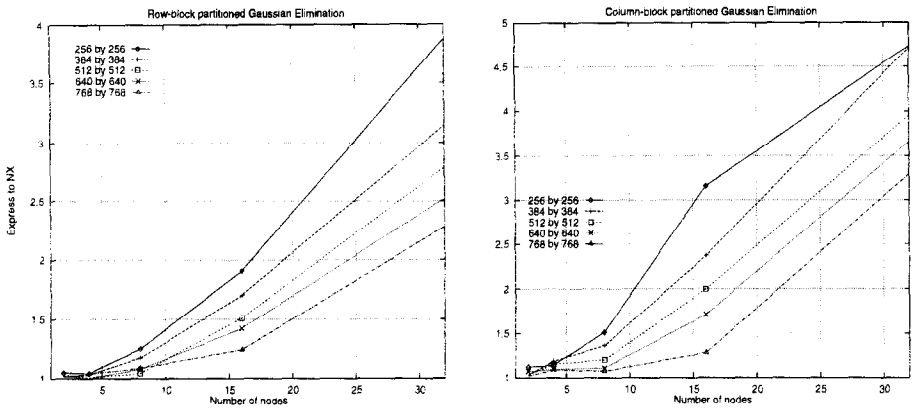
Table 7
Timings (s) for row-block partitioned Gaussian elimination using Express and PVM on the IPX cluster connected by Ethernet

Matrix size	PE = 1		PE = 2		PE = 4		PE = 8	
	Express	PVM	Express	PVM	Express	PVM	Express	PVM
256 × 256	15.26	15.97	8.38	16.44	5.24	14.53	7.10	19.16
384 × 384	51.74	53.78	25.52	41.65	15.38	34.09	15.93	33.60
512 × 512	124.87	131.82	66.09	92.46	38.66	66.86	26.89	58.64
640 × 640	247.12	255.48	126.79	157.65	62.26	99.72	61.62	88.97
768 × 768	425.91	448.82	223.47	261.07	118.87	167.52	94.33	120.38

Table 8
Timings (s) for row-block partitioned Gaussian elimination using Express and PVM on the HP cluster connected by FDDI

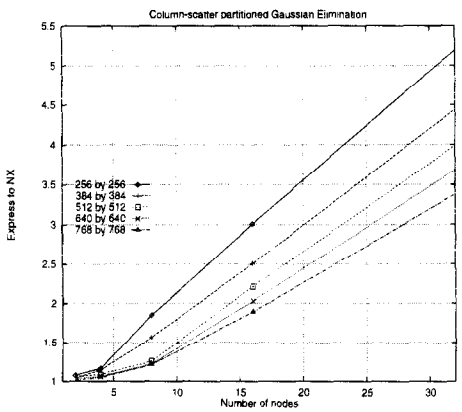
Matrix size	PE = 1		PE = 2		PE = 4	
	Express	PVM	Express	PVM	Express	PVM
256 × 256	6.01	5.95	3.07	6.63	1.81	5.52
384 × 384	22.36	21.94	10.43	18.36	5.72	14.16
512 × 512	67.43	65.86	29.02	41.32	12.18	25.56
640 × 640	115.74	113.46	52.30	70.75	25.22	43.01
768 × 768	232.90	226.97	99.04	118.81	44.65	66.70

one can observe that for small number of processors on the iPSC/860, the difference between the performance of Express and NX is insignificant. Express performs poorly with a large number of processors. The performance of PVM is even worse for the large number of processors. The main reason is that the algorithm used makes an extensive use of broadcast operations for sending the pivot row to other processors. Moreover, no optimizations are made in communication calls to exploit the hypercube topology. As a result, the algorithm uses a number of global operations which are quite slow in Express and PVM. The relative performance of this version of Gaussian elimination using Express and NX is also indicated in Fig. 14(a) in which we have plotted the ratios of



(a) Row-block partitioning.

(b) Column-block partitioning.



(c) Column-scattered partitioning.

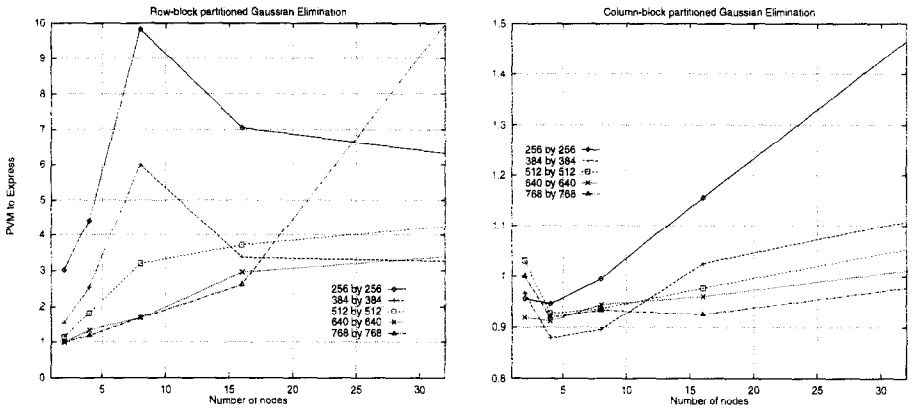
Fig. 14. Ratios of execution times of Express to NX for different Gaussian elimination algorithms on the iPSC/860.

execution times of Express to NX. The ratios of execution times of PVM to Express are plotted Fig. 15(a).

On both the IPX and HP clusters, the execution times for Express and PVM exhibit speedup with an increasing number of processors, with a reasonably large matrix size. The timings of Express version is again better than the PVM version.

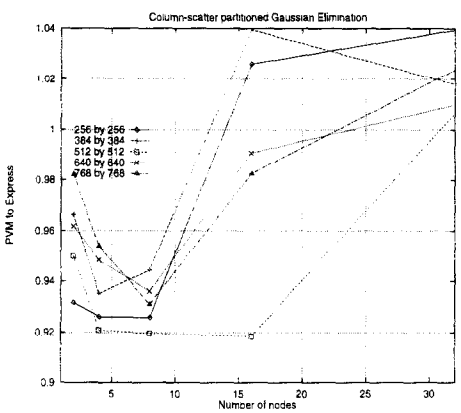
6.2. *Gaussian elimination (column–block partitioning)*

In this version of Gaussian elimination, the data is partitioned across processors in terms of blocks of columns. The execution times for this version are shown in Tables 9–11. On the iPSC/860, Express and PVM perform well for large matrices when the number of processors is between 2 and 16 but performed poorly for 32 processors. It can



(a) Row-block partitioning.

(b) Column-block partitioning.



(c) Column-scattered partitioning.

Fig. 15. Ratios of execution times of PVM to Express for different Gaussian elimination algorithms on the iPSC/860.

be noticed that, in general, the execution times of the column–block partitioned versions are larger than those of the row–block partitioned versions. This is due to the fact that in the row–block partitioned algorithm, the determination of the pivoting row is done in

Table 9
Timings (s) for column–block partitioned Gaussian elimination using NX, Express and PVM on the iPSC/860

Matrix size	PE = 2			PE = 4			PE = 8			PE = 16			PE = 32		
	NX	Express	PVM	NX	Express	PVM	NX	Express	PVM	NX	Express	PVM	NX	Express	PVM
256 × 256	6.3	7.0	6.7	3.2	3.6	3.4	1.8	2.8	2.8	1.2	3.8	4.4	1.1	5.3	7.8
384 × 384	23.1	23.9	23.1	9.8	11.6	9.9	5.0	6.8	6.1	3.2	7.6	7.8	2.6	12.3	13.6
512 × 512	46.0	47.5	46.0	23.0	26.5	24.6	11.9	14.3	13.4	6.6	13.2	12.9	5.1	20.2	21.3
640 × 640	102.6	110.2	102.7	51.5	56.5	51.7	25.8	28.6	26.9	12.1	20.8	20.1	8.3	30.3	30.6
768 × 768	178.6	182.8	182.9	90.6	98.7	90.8	46.9	50.2	47.3	24.2	31.0	28.6	13.0	42.7	41.7

Table 10
Timings (s) for column–block partitioned Gaussian elimination using Express and PVM on the IPX cluster connected by Ethernet

Matrix size	PE = 1		PE = 2		PE = 4		PE = 8	
	Express	PVM	Express	PVM	Express	PVM	Express	PVM
256 × 256	9.90	25.01	9.87	27.06	7.71	38.51	10.51	55.46
384 × 384	32.72	55.93	26.55	53.31	18.80	63.97	18.83	96.41
512 × 512	76.59	104.70	57.55	104.61	37.11	91.97	33.92	199.60
640 × 640	150.03	198.01	107.79	154.16	65.54	131.15	51.55	256.55
768 × 768	265.83	303.50	182.40	241.01	105.54	185.98	76.23	317.70

parallel. On the other hand, column–block partitioned algorithm performs this step serially. When the number of processors is large, the PVM-based column–block partitioned version takes less time than the PVM-based row-partitioned version because the broadcast operation is used less frequently in the former version. The inefficient broadcast operation of PVM, therefore, is the main cause of performance degradation when using a large number of processors.

On the IPX cluster of workstations, it is interesting to see that Express yield speedup if the number of workstations is increased even up to 8. PVM, however, does not perform well beyond 4 workstations. Both Express and PVM perform better on the HP cluster than on the iPSC/860 with 1, 2 or 4 workstations or processors. The relative performance of Express to NX and PVM to Express is indicated in Fig. 14(b) and Fig. 15(b), respectively.

6.3. Gaussian elimination (column–scattered partitioning)

In this version, the data is partitioned using cyclic distribution of the columns of the coefficient matrix. The execution times of Gaussian elimination with column–scatter partitioning using Express, PVM and NX primitives on different platforms are shown in Tables 12–14. The results indicate that, in general, this algorithm is better than the column–block partitioning version but is comparable to the row–block partitioning version. This is because the column–scatter partitioning can balance load well which

Table 11
Timings (s) for column–block partitioned Gaussian elimination using Express and PVM on the HP cluster connected by FDDI

Matrix size	PE = 1		PE = 2		PE = 4	
	Express	PVM	Express	PVM	Express	PVM
256 × 256	3.10	7.71	2.85	7.92	2.15	10.43
384 × 384	11.40	18.44	9.05	17.58	6.28	19.74
512 × 512	30.84	40.80	27.31	33.11	13.01	33.48
640 × 640	66.35	74.52	41.42	57.31	25.80	50.45
768 × 768	115.99	125.29	89.81	93.04	43.33	81.55

Table 12
Timings (s) for column-scattered partitioned Gaussian elimination using NX, Express and PVM on the iPSC/860

Matrix size	PE = 2			PE = 4			PE = 8			PE = 16			PE = 32		
	NX	Express	PVM	NX	Express	PVM	NX	Express	PVM	NX	Express	PVM	NX	Express	PVM
256 × 256	4.1	4.4	4.1	2.2	2.6	2.4	1.4	2.7	2.5	1.2	3.8	3.9	1.2	6.3	6.6
384 × 384	13.9	14.5	14.0	6.7	7.7	7.2	3.8	5.4	5.1	3.0	7.5	7.9	2.8	12.3	12.5
512 × 512	32.0	33.9	32.2	16.2	17.7	16.3	8.9	11.2	10.3	5.8	12.9	11.8	5.0	20.1	20.2
640 × 640	63.3	65.6	63.4	32.1	34.1	33.2	16.1	18.8	17.6	9.8	20.0	19.8	8.1	30.1	30.4
768 × 768	110.7	112.7	111.0	55.6	58.3	57.0	28.6	34.8	32.4	15.4	28.4	28.1	12.5	42.1	43.0

Table 13
Timings (s) for column-scattered Gaussian elimination using Express and PVM on the IPX cluster connected by Ethernet

Matrix size	PE = 1		PE = 2		PE = 4		PE = 8	
	Express	PVM	Express	PVM	Express	PVM	Express	PVM
256 × 256	7.38	12.26	6.18	25.96	6.62	35.81	9.14	55.52
384 × 384	23.95	32.96	16.83	46.28	17.36	59.60	17.79	89.87
512 × 512	58.61	67.32	34.54	74.39	28.84	92.48	35.59	137.26
640 × 640	112.26	122.60	63.04	114.93	55.74	203.08	51.02	191.80
768 × 768	192.60	203.94	102.78	169.32	68.68	147.02	72.40	255.08

Table 14
Tunings (s) for column-scattered Gaussian elimination using Express and PVM on the HP cluster connected by FDDI

Matrix size	PE = 1		PE = 2		PE = 4	
	Express	PVM	Express	PVM	Express	PVM
256 × 256	0.27	5.16	1.82	7.31	1.88	10.59
384 × 384	9.91	14.98	5.39	14.78	4.45	18.51
512 × 512	27.49	34.44	12.32	26.48	9.10	29.67
640 × 640	60.55	63.87	26.61	44.09	17.20	43.46
768 × 768	104.41	108.49	56.74	69.12	29.14	63.68

results in reduction of processor waiting times. However, it requires excessive exchanges of messages which can delay the execution. When the matrix is small and the number of processors is large, the benefit of load balancing is small compared to the extra cost of communication.

On the iPSC/860, for small matrix size, the execution times for the Express version exhibit speedup with the number of processors increased from 1 to 8. Speedup is also observed for 16 processors but only for large matrix sizes. The execution times increase with 32 processors. On the other hand, the NX version yields speedup with 32 processors. The execution times for the PVM version are comparable to Express. For large matrix size, the PVM version sometimes performs better than the Express version.

Table 15
Timings (s) for the N-body problem using NX, Express and PVM on the iPSC/860

Points	PE = 1	PE = 2			PE = 4			PE = 8			PE = 16			PE = 32		
		NX	Express	PVM	NX	Express	PVM	NX	Express	PVM	NX	Express	PVM	NX	Express	PVM
512	2.78	1.49	1.52	1.82	0.85	0.87	0.93	0.41	0.48	4.02	0.26	0.31	0.41	0.20	0.32	0.45
1 k	10.96	5.69	6.20	6.38	2.97	2.97	4.40	1.49	1.69	2.95	0.81	0.95	2.38	0.53	0.67	0.49
2k	42.71	22.25	22.99	22.72	11.20	11.36	13.85	5.42	5.90	8.24	2.89	3.35	6.94	1.61	2.23	3.28
4k	166.51	84.16	86.50	87.32	42.51	44.00	53.57	21.58	22.27	30.44	11.43	12.29	18.65	6.32	8.19	15.62
8k	657.88	331.46	342.19	342.21	167.28	171.23	195.98	84.11	86.99	114.96	42.34	44.47	62.57	22.42	24.01	36.54
16k	—	1342.9	1401.2	—	674.42	694.64	753.15	339.83	350.30	413.08	170.04	177.67	230.82	85.34	89.88	124.7

Table 16
Timings (s) for the N-body problem using Express and PVM on the IPX cluster connected by Ethernet

Points	PE = 1		PE = 2		PE = 4		PE = 8	
	Express	PVM	Express	PVM	Express	PVM	Express	PVM
512	3.51	3.94	1.87	2.06	1.09	1.06	0.57	0.62
1k	14.20	15.32	7.27	7.57	3.76	5.19	1.86	2.98
2k	58.27	59.66	28.75	29.60	14.95	19.78	7.53	9.30
4k	229.00	241.02	113.87	114.37	58.40	72.41	29.27	38.78
8k	908.17	944.21	458.32	468.84	232.92	231.67	119.96	116.03
16k	—	—	1802.05	1849.4	918.6	909.97	481.33	456.41

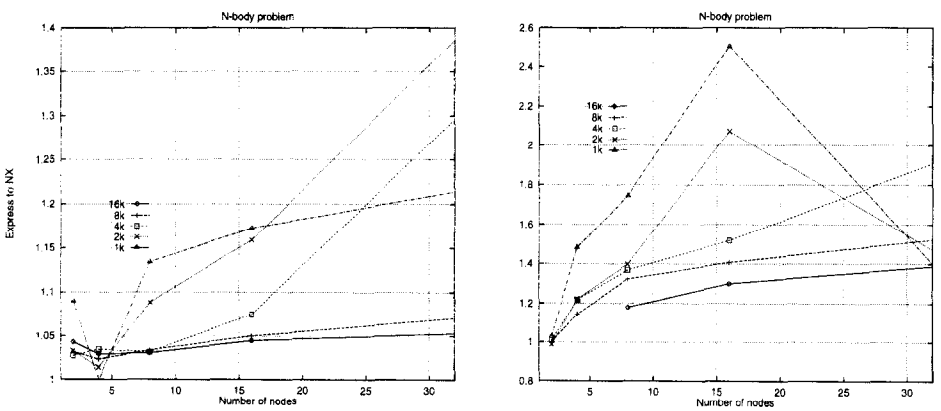
Table 17
Timings (s) for the N-body problem using Express and PVM on the IPX cluster connected by FDDI

Points	PE = 1		PE = 2		PE = 4	
	Express	PVM	Express	PVM	Express	PVM
512	1.71	2.36	0.86	1.21	0.43	0.66
1k	6.79	9.45	3.35	4.75	1.65	2.43
2k	27.09	37.96	13.21	18.97	6.60	9.51
4k	108.50	151.55	52.77	75.49	26.35	37.91
8k	433.82	606.14	209.75	299.86	105.58	150.71
16k	—	—	838.73	1233.23	467.35	601.46

On the cluster of HP and IPX clusters, the execution times for the Express and PVM decrease with increasing number of workstations. The times on the iPSC/860 and the HP cluster were comparable. For smaller matrix size, the times on the HP cluster are even better than those on the iPSC/860. The relative performance of Express to NX and PVM to Express is indicated in Fig. 14(c) and Fig. 15(c), respectively.

6.4. The N-body problem

The program for the N-body problem was written [1] using the algorithm reported in [20]. The algorithm used in this program is the simple $O(N^2)$ algorithm and not the more optimized $O(N \log N)$ approach. The execution times of the three platforms are shown in Tables 15–17. On the iPSC/860, all three versions yield speedup as the number of processors increases. For large number of bodies, the times of Express and iPSC/860 are very close. However, the PVM version is slower than the other two versions by a factor of 1 to 2 (see Fig. 16). On the HP cluster, the PVM version is in general 1.5 times slower than the Express version. On the IPX cluster, the times of both



(a) Ratios of execution times of Express to NX.

(b) Ratios of execution times of PVM to Express.

Fig. 16. Ratios of execution times of Express to NX and PVM to Express for the N-Body problem on the iPSC/860.

[1] Min-Yoa Wu thankfully provided the NX version.

Express and PVM versions are close. Compared to the iPSC/860, the HP cluster was faster but IPX is slower. Because of less communication involved, both Express and PVM scale well on the workstation clusters.

7. Conclusions

In this paper, we made an experimental performance comparison of Express and PVM. The performance study was carried out on an iPSC/860 parallel computer, a cluster of IPX workstations connected by an Ethernet and a cluster of HP workstations connected by FDDI. The summary of our results is that Express primitives are in general faster than those of PVM but marginally slower than those of NX. When the message size is small, PVM primitives are significantly slower than NX and Express on the iPSC/860. In particular, the broadcast primitive of PVM needs significant improvement. In general, PVM is also slower than Express on the IPX and HP clusters. As the message size increases, the difference between the performance of PVM and Express becomes less significant. For very large messages, PVM performs slightly better than Express. This makes Express more suitable for fine-grained applications on parallel systems and PVM more suitable for coarse-grained problems on distributed systems. The global operations of both Express and PVM do not scale very well with increasing number of processors. Comparing different platforms, FDDI is comparable to iPSC/860. However, since only 4 workstations were available to us, the scalability of FDDI could not be determined. In general, FDDI is faster than Ethernet by factors of 4 to 8. Comparing the timings of applications, we noticed that PVM outperformed Express on the IPSC/860 when the granularity of the problem was large, the number of processors was small and the message sizes were large.

The advantages of PVM include its small size, the simplicity in programming and support for complete heterogeneous supercomputing. The initial versions of PVM used TCP/IP sockets to implement all communications. Consequently, PVM was limited to distributed systems. However, recent versions of PVM such as the one reported in this paper have included implementation within the parallel systems. On the other hand, Express which was originally developed for homogeneous multiprocessors has evolved towards distributed systems. It has also tackled the problem of heterogeneity recently. Contrary to PVM, Express provides a very large number of primitives and a number of additional tools and utilities.

The choice of using Express or PVM would depend on a number of factors such as the type of application, programming paradigm, communication patterns and typical sizes of the messages in the application. The type of application may determine whether the application can be better implemented as coarse-grained or fine-grained. If it is coarse-grained, a networked hardware platform would be more useful and PVM would be a more appropriate choice. However, if the application can be implemented as fine-grained on a parallel computing machine, Express can be more useful. The advantage of Express is that by using its portable message-passing library, the development phase of an application can be carried out on a cluster of networked workstations but final experimentation can be done on the parallel machine. The programming

paradigm could be SPMD or pure MIMD with a combination of different processes. The former can be better implemented using Express's grid facilities while the latter can be done more efficiently using PVM due to its ability to dynamically create processes. Furthermore, PVM can be useful for running an application across multiple parallel machines as a combination of parallel and distributed computing paradigms. Communications patterns and sizes of the messages within a parallel program can also affect the choice of Express and PVM (for example, PVM does not provide collective communication and global operations). Finally, the major advantage of PVM over Express is that the former is available in public domain while the latter is a commercial product.

References

[1] I. Ahmad et al., Implementation and Scalability of Fortran 90D Intrinsic Functions on Distributed Memory Machines, Technical Report, Northeast Parallel Architectures Center, Syracuse University, Syracuse, NY, May 1992.
[2] S. Ahuja, N. Carriero, D. Celernter, Linda and Friends, IEEE Comput. 8 (1986) .
[3] I. Angus, G. Fox, J. Kim, D. Walker, Solving Problems on Concurrent Processors, vol. II, Prentice Hall, Englewood Cliffs, New Jersey, 1990.
[4] G. Bell, Ultracomputers a teraflop before its time, Commun. ACM 35 (8) (1992) 27–47.
[5] A. Beguelin, J. Dongarra, A. Geist, R. Manchek, V. Sunderam, Paradigms and Tools for Heterogeneous Network Computing, Supercomputing '92, Nov. 1992.
[6] A. Beguelin, J. Dongarra, A. Geist, R. Manchek, V. Sunderam, Solving computational grand challenges using a network of heterogeneous supercomputers, Proceedings of Fifth SIAM Conference on Parallel Processing, Philadelphia, PA, 1992.
[7] R. Berrendorf, J. Helin, Evaluating the basic performance of the Intel iPSC/860 parallel computer, Concurrency Pract. Exper. 4 (3) (1992) 223–240.
[8] S. Bokhari, Communication overhead on the Intel iPSC/860 hypercube, ICASE Interim Report 10 182055, NASA Langley Research Center, Hampton, VA, May 1990.
[9] S. Bokhari, Multiphase complete exchange on a circuit switched hypercube, Technical Report 91-5, ICASE, NASA Langly Research Center, Hampton, Virginia, January 1991.
[10] R.M. Butler and E. Lusk, Monitors, messages and clusters: The P4 parallel programming system, Technical Report TM-ANL 92/17, Argonne National Laboratory, 1992.
[11] N. Carriero, D. Celernter, Linda in context, Commun. ACM. 32 (4) (1989) 444–458.
[12] J. Flower and A. Kolawa, A Packet History of Message Passing Systems, Parasoft Corporation, 1992.
[13] J. Flower, A. Kolawa, Express is not just a message passing system: Current and future directions in Express, Parallel Comput. 20 (1994) 597–614.
[14] G.C. Fox, M.A. Johnson, G.A. Lyzenga, S.W. Otto, J.A. Salmon, D.A. Walker, Solving Problems on Concurrent Processors, vol. 1, Prentice Hall, 1988.
[15] G.A. Geist, M.T. Heath, B.W. Peyton, P.H. Worley, A user's guide to Picl, a portable instrumented communication library, Technical Report ORNL/TM-11616, Oak Ridge National Laboratory, Oak Ridge, TN, October 1991.
[16] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, V. Sunderam, PVM 3 Users Guide and Reference Manual, Oakridge National Labs., 1992.
[17] Intel Corporation, iPSC/2 and iPSC/860 Programmers Reference Manual, June 1990.
[18] O.A. McBryan, An overview of message-passing environments, Parallel Comput. 20 (1994) 417–444.
[19] S.A. Moyer, Performance of the iPSC/860 node architecture, Tech. Report IPC-TR-91007, Institute of Parallel Computations, University of Virginia, May 1991.
[20] Parasoft Corporation, Express, Fortran User's Guide, 1990.
[21] Parasoft Corporation, Express Introductory Guide version 3.2, 1992.
[22] P. Pierce, The NX message passing interface, Parallel Comput. 20 (1994) 463–480.

[23] A. Skjellum and A.P. Leung, Zipcode: A portable multicomputer communication library atop the reactive kernal, Proceedings of the 5th Distributed Memory Computing Conference, April 1990.
[24] V. Sunderam, PVM: A framework for parallel distributed computing, Concurrency Pract. Exper. 3 (4) (1990) 315–339.
[25] V.S. Sunderam, G.A. Geist, J. Dongarra, R. Manchek, The PVM concurrent computing system: Evolution, experiences and trends, Parallel Comput. 20 (1994) 531–545.
[26] L. Turcotte, A survey of software environments for exploiting networked computing resources, Technical Report, Engineering Research Center for Computational Field Simulations, Mississippi State University, January 1993.