Comparison between different implementations of PVM on IBM SP1 and CRAY T3D

C. Calvin
LMC - IMAG, 46 Av. Félix Viallet,
38031 Grenoble Cedex 1, France
Christophe.Calvin@imag.fr

L. Colombet
CISI-CENG/DI, 17 Rue des Martyrs,
38054 Grenoble cedex 9, France
colombe@orchidee.ceng.cea.fr

October 5, 1994

1 Introduction

PVM has been designed in order to be portable on any parallel machine, but the performance of an application written in PVM strongly depends on the quality of the implementation of PVM. In this article, we compare different implementations of PVM on the IBM SP1, with the implementation on the CRAY T3D. These comparisons have been done using communication benchmarks and a program which computes a bi-dimensional FFT. In section 2 we briefly describe the two machines, then in the next section we present the results of the communication benchmarks on the SP1 and on the T3D. The fourth section is devoted to the bi-dimensional FFT programs which allows us to test the capability of the both machines to overlap the communications by the computations.

2 Brief description of the target machines

2.1 The IBM SP1

The SP1 machine used is composed of 16 processors IBM Power1 (or RS6000) with a peak performance of 125 Mflops and a clock rate at 62.5 MHz. Thus the peak performance of the machine is equal to 2 Gflops. These processors are linked using a multi-stage network, called *High Performance Switch* (HPS), with a bi-directional bandwidth equals to 40 Mbytes/s. The routing protocol used is a buffered-wormhole routing. There exists several libraries of communication on the SP1:

- The library *EUI* has been implemented using the *MPI* specifications [8]. It is based either on the IBM's *Light Speed Protocol* (*LSP*) or the *tcp/ip* protocol on the HPS.
- The high performance version is the *EUI/H* which uses the lowest level of communication.
- *PVM* (Oak-Ridge version) [2] can be used either with the ethernet network or with the HPS using the *tcp/ip* protocol.
- An optimized implementation of *PVM* is proposed by IBM called *PVMe* [7] which uses the HPS with *LSP*.
- *MPI* is based on *EUI*.

2.2 The CRAY T3D

The Cray T3D is composed of up to 2048 processing elements (PEs) which are DEC Alpha chips with a peak performance of 150 Mflops. Each PE possesses all the necessary resources to execute program code

independently of all other PEs in the system. PEs are paired together into processing nodes that share a network interface to interconnect to the network switch. Each PE node has direct connection to 6 neighboring PE nodes. The topology of this interconnect network is a 3D torus where the data traffic is pipelined bi-directionally concurrently with a bandwidth equals to 300 Mbytes/s.
The *ShMem* package is a library to facilitate structured communication via shared memory on the Cray T3D. These routines provide a high-performance style of shared-memory programming for Cray MPP systems, but they require careful use of variables and addresses. This package is based on *remote loads* [6]

Then the Cray MPP implementation of PVM can be used with either one or both of the following two models [5]:

- Stand-alone mode, in which *PVM* is used for intra-partition (PE-to-PE) communication, based on remote loads.
- Distributed mode, in which *PVM* is used to communicate outside the Cray T3D partition.

3 Communication benchmarks

3.1 Description of the communication benchmarks

We have implemented three different benchmarks which correspond to the main communications used in scientific applications:

- An exchange between two processors, which measures the bandwidth.
- A broadcast: one processor sends a message to all the others.
- A total exchange: each processor broadcasts a message.

3.2 Exchange between two processors

The results of the exchange-test using different implementations of PVM on both machines are given on figure 1.

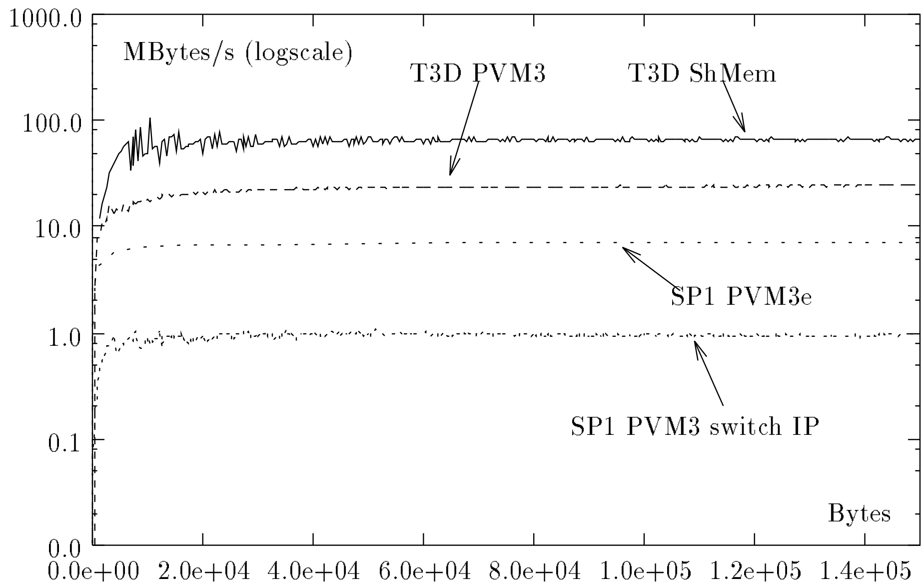
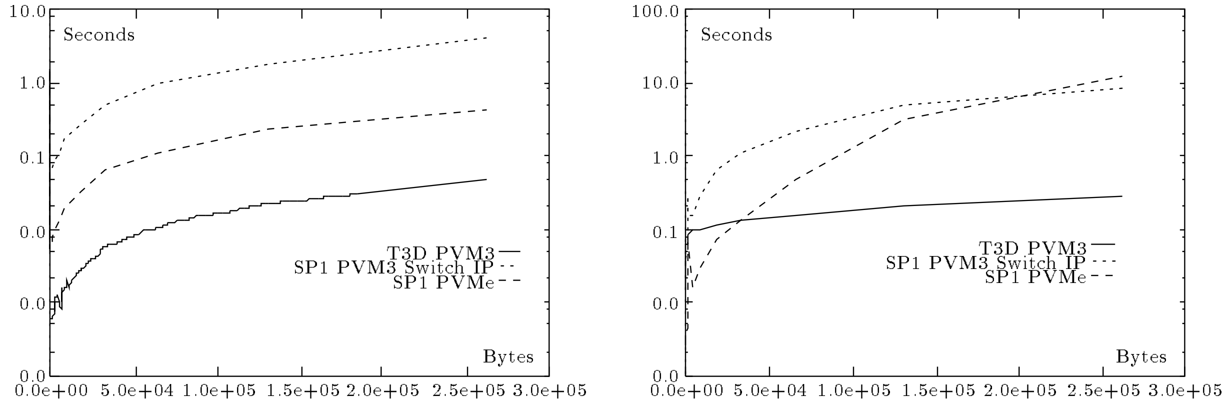


Figure 1: Performances of the exchange using different implementations of PVM on the SP1 and the T3D

The best performances are obtained using the *ShMem* library and *PVM* on Cray T3D. Regarding to the peak bandwidth, *PVMe* on the IBM SP1 is the most efficient: 7.5 Mbytes/s over 20 Mbytes/s for *PVMe* and 25 Mbytes/s over 150 Mbytes/s for *PVM3* on the T3D. We can also note that for these two precedent libraries the maximum performance is reached for quite small sizes of messages (10 Kbytes). Let us note that we can obtain very good performances with the *ShMem* package, but these ones strongly depends on the packet size.

3.3 Broadcast and total exchange benchmarks

We present on figure 2 the results of the broadcast benchmark on 16 processors and the results of total exchange on 8 processors.
For the broadcast, the results obtained, corresponds to the precedent, *ie* we have the best performance on the T3D. The performances of the broadcast depends on the bandwidth which can be obtained using the different *PVM* libraries.
Contrary to the precedent benchmarks (broadcast and exchange), the expected results for the total-exchange are not obtained. For small messages we have a very good behaviour of *PVMe*, but as the size of the messages increases, the time becomes greater than the time obtained with *PVM3* on the IP-Switch. This can be explained by the fact that when the traffic on the switch is too important, *LSP* cannot manage it, which implies timed-out on reception. On the contrary, the behaviour of this benchmark on the T3D is very good.



Broadcast on 16 processors

Total exchnage on 8 processors

Figure 2: Results of the global communication benchmarks

4 Bi-dimensional FFT

4.1 Introduction

The aim of this experiment is to test the performance of PVM using a real application and the capabilities of the machines to overlap communications by computations.
The algorithm of bi-dimensional FFT consists in computing mono-dimensional FFT on both dimensions. A parallel algorithm for computing the bi-dimensional FFT is the *Transpose Split* algorithm[4, 3]. The matrix is distributed using a row-wise allocation. After a mono-dimensional FFT has been computed on each row, a transposition of the matrix is done. Then, another mono-dimensional FFT is computed on each row of the resulting matrix. This algorithm can be improved by overlapping the computation of the local mono-dimensional FFT and the matrix transposition. A set of rows can be transposed as soon as it has been computed, and then during this communication, another FFT can be realized on another set of rows [3].

4.2 Experimental results

We summarize the experimental results on the SP1 in table 1, and on the T3D in table 2. We detail the communication and the computation times for the non-overlapped method in order to show the maximum gain that we can obtain with the overlapped one. It is clear that for the SP1, we cannot overlap the communications, because there is no process, or processor, dedicated to the management of the communications. The only gain we obtain is for very large size of messages, when the transmission time is greater than the time spent in managing the communications.

Matrix size	Total time of the non-overlapped method	Computational time	Communication time	Total time of the overlapped method
256	**0.212**	0.082	0.126	**0.241**
512	**1.462**	0.396	1.066	**1.242**
1024	**7.348**	2.818	4.53	**6.579**
2048	**22.664**	12.996	9.668	**21.011**

Table 1: Execution times in seconds of bi-dimensional FFT algorithms on the SP1 with 16 processors for various sizes of matrices

It is more difficult to conclude for the T3D, because the communication time is very small regarding to the computation time. Thus, we have implemented another method for computing the bi-dimensional FFT: the *local distributed* method. It consists in computing mono-dimensional FFT on the first dimension of the matrix and distributed mono-dimensional FFT on the second dimension [4, 3]. As for the transpose split method, we can overlap the communications by the computations during the distributed phase (see [3] for more details). In this method the computation time and the communication time are of the same order (see table 3). As we can see we obtain an overlap of the communications by the computations.

Matrix size	Total time of the non-overlapped method	Computational time	Communication time	Total time of the overlapped method
256	**0.250**	0.200	0.050	**0.240**
512	**1.080**	0.870	0.210	**1.010**
1024	**4.900**	3.870	1.030	**4.590**
2048	**20.420**	17.310	3.110	**19.310**

Table 2: Execution times in seconds of bi-dimensional FFT algorithms on the T3D with 16 processors for various sizes of matrices

Matrix size	Total time of the non-overlapped method	Computational time	Communication time	Total time of the overlapped method
256	**0.138**	0.010	0.098	**0.198**
512	**0.366**	0.180	0.186	**0.361**
1024	**1.511**	0.830	0.681	**1.211**
2048	**6.172**	3.6	2.572	**4.628**
4096	**27.542**	15.960	11.492	**18.660**

Table 3: Execution times in seconds of bi-dimensional FFT using local distributed algorithms on the T3D with 32 processors for various sizes of matrices

We present in figure 3 the evolution of the execution time of the non-overlapped transpose split method for a matrix size equal to 2048 from 16 up to 128 processors on the T3D. This figure shows the scalability of this method, since the efficiency is nearly equal to 1. This is due to the good performances of the communications on the T3D.

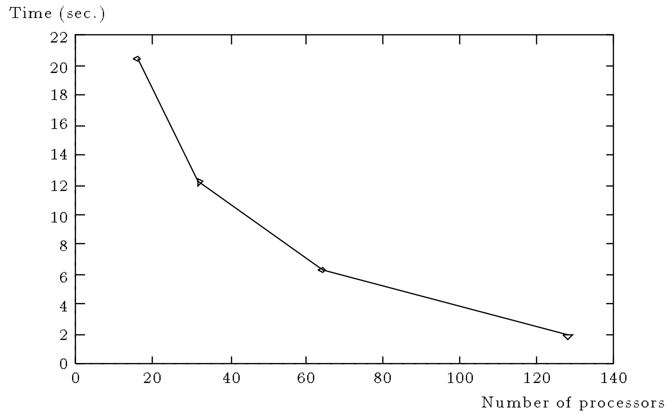


Figure 3: Evolution of the execution depending on the number of processors for $n = 2048$ on the T3D

5 Conclusion and future work

In this paper we have presented some benchmarks in order to test the quality of different implementations of PVM on the IBM SP1 and Cray T3D. These benchmarks lead us to several observations. The implementation of *PVMe* on the IBM SP1 is the most efficient, since we have obtained half of the peak bandwidth. But the reliability of this implementation is not as good as the CRAY one. It is up to the user to deal with the flow control. Although the implementation of PVM on the CRAY is less efficient ($\frac{1}{6}$ of the peak bandwidth), all the benchmarks done have the best execution time. This due to the very good capabilities of the network of the T3D.
We can also notice that on the SP1 machine, it is not possible to overlap the communications by the computations. One way to avoid this problem is to use a multi-processing model of programming, where processes would be dedicated to the communication management [1]. At this time, this is not possible on the SP1 with *PVMe*, but on the SP2 offers this possibility.
All these benchmarks allows us to have a better knowledge of the performances and of the behaviour of these two machines using PVM. The parameters and informations obtained are very useful for designing efficient algorithms.

References

[1] E. Apache, *APACHE : Algorithmique Parallèle et pArtage de CHargE*, tech. rep., LMC - LGI - IMAG, 1993.

[2] A. Beguelin, J. Dongarra, A. Geist, R. Manchek, and V. Sunderam, *A User's Guide to PVM Parallel Virtual Machine (version 3)*, tech. rep., Oak Ridge National Laboratory, May 1994.

[3] C. Calvin and F. Desprez, *Minimizing Communication Overhead Using Pipelining for Multi-Dimensional FFT on Distributed Memory Machines*, in Proceedings of the International Conference on Parallel Computing '93, F. P. D.J. Evans, G.R. Joubert and D. Trystram, eds., Advances in Parallel Computing, North Holland, 1993.

[4] C. Y. Chu, *Comparison of Two-Dimensional FFT Methods on the Hypercube*, in The Third Conference on Hypercube Concurrent Computers and Applications, G. Fox, ed., vol. 2, 1988.

[5] Cray Research Inc., *PVM and HeNCE Programmer's Manual*.

[6] ———, *ShMem user's Manual*.

[7] IBM Corporation, *IBM AIX PVMe User's Guide and Subroutine Reference*, Mar. 1994.

[8] Message Passing Interface Forum, *Document for a Standard Message-Passing Interface*, Apr. 1994.