High-Speed LANs: New Environments for Parallel and Distributed Applications

Patrick Geoffray Laurent Lefèvre CongDuc Pham Loïc Prylli[†]
Olivier Reymann[†] Bernard Tourancheau Roland Westrelin

RHDAC-LIGIM, Université Lyon 1, Claude Bernard
[†]LIP, Ecole Normale Supérieure de Lyon
e-mail: `bip-team@lhpca.univ-lyon1.fr`

Abstract. As the technology for high-speed networks has incredibly evolved this last decade, the interconnection of workstations at gigabits rates and low prices has become a reality. These clusters, based on regulars workstations (e.g. PCs), can now be used in place of traditional parallel computers with no possible comparison on the prices! In this article, 3 applications (high performance computing, distributed shared memory system and parallel simulation) that were traditionally executed on expensive parallel machines are ported on a Myrinet-based cluster of PCs. The results show that the performances of these new architectures can be very close to those obtained on state-of-the art parallel computers.

Keywords: High-speed LANs, Parallel and Distributed Applications
Submitted to: Topic 8, High-Performance Computing and Applications.

1 New technologies for parallel and distributed applications

As soon as computers started to be used for problem solving tasks, this was a starting point for a never-ending quest towards more and more computational power. The availability of powerful computing environments began decades ago by putting several processing units in the same box and give them a way to communicate each other. The problem is that parallel machines were mainly built with proprietary and custom parts. Adding the high cost for software development, parallel computers could only be bought by large organizations.

For a long time, parallel computers were the solution for people with high computation needs. Nowadays, there are still some applications where parallel computers are required but the demand has decreased dramatically as parallel computers are still highly expensive. If the processing units can now be taken from the commodity market, the interconnection networks and the softwares are still highly customized. While sequential computers have always seen a dramatic cut down in their prices every year, parallel computers took the opposite direction because of the decreasing demand.

However, there are not so many solutions for having more computation power: one has to use several processors. The choice resides on how to make the parallel system: from custom or standard products. If the first choice was justified by the low-quality of products from the commodity market available at that time, this is not the case any more. There is the possibility to go a complete step farther by building parallel systems entirely from standard components. Processing units are just regular workstations or PCs that can be bought in any supermarket for a few thousands of dollars, interconnection networks are also taken from the high-speed and high-bandwidth regular market of LANs (local area network) or SANs (system area network) such as Gigabits Ethernet, ATM, and Myrinet. These architectures are often referred to as Network Of Workstations (NOW). Several research teams have launched projects dealing with NOWs used as parallel machines. Early experiments with IP-based implementations have shown disappointing performances so the goal of most of these teams is to design the software needed to make clusters built with commodity components and high-speed networks really efficient. The NOW project of UC Berkeley [1] was one of the first one. Its main contribution is the Active Messages [15] layer that provides high performance access to the network. In this sense, it is very similar to the BIP software we are developing on our cluster.

The objective of the paper is two-folds: (*i*) show experiments on different types of LANs with several communication softwares and (*ii*) compare our Myrinet-based test-bed with the BIP software to traditional parallel computers. To do so, 3 applications covering very different areas of distributed and parallel computing are ported on NOWs with a focus on Myrinet-based networks. The paper is organized as follows: Section 2 presents the high-speed environment based on a Myrinet network and the BIP communication software. Section 3 compares state-of-the-art parallel computers and our Myrinet test-bed with the NAS benchmark. Section 4, 5 and 6 presents the applications (high performance computing, distributed shared memory system and parallel simulations) that were ported on Myrinet. Conclusions are given in Section 7.

2 The high-speed LAN environment based on Myrinet

The Myricom LAN[3] target was chosen for its performance over the Gbits/s, its affordable price and its software openness (all the software and specifications are freely available for customers). There are several features that makes this kind of technology much more suitable for than a more typical commodity network based on Fast or Giga-Ethernet:

- the hardware provides a end-to-end control flow that allows the guarantee of a reliable delivery and alleviates the problem of implementing in software the reliability on top on an lossy channel. As message losses are exceptional, it is possible to use algorithms that focus on very low overheads in the normal case,
- the interface card has a general purpose processor which is powerful enough to handle most of the communication activity without interrupting the main processor. In particular, it provides efficient overlapping communication and

computation, collective operations without grabbing the main processor and one-sided communications,
- the interface card has up to one megabyte memory for buffers. As the network being as fast as the PCI bus, this memory allows to isolate transfers on the network from transfer on PCI, so that they will not be limited by the slowest one at all times.

2.1 BIP

BIP stands for Basic Interface for Parallelism. Our basic idea was to build it with a library interface accessible from applications that will implement a high speed protocol on the Myrinet network. BIP provides only a protocol with low level functionalities. Although specialized parallel applications could interface directly with it, it is intended as the base of other protocol layers like IP, and higher level APIs like the well established MPI and PVM. All the applications described later in this paper actually use our MPI implementation based on top of BIP.

Highly optimized, the raw communication performance for BIP is about $5\mu s$ latency one-way. With large messages, the bandwidth goes up to 125Mbyte/s. The MPI layer adds about $5\mu s$ for small messages. As will show the other examples of the paper, this allows applications to scale more by allowing the use of a finer grain of computation. BIP is described in more details in [13].

2.2 Other systems and perspectives

First introduced in 1997, BIP was more an incremental step in a large family of softwares than a complete new design. Especially on Myrinet, we can find many other communication systems: Active Messages from Berkeley[10], Fast Messages from Illinois [11], U-Net from Cornell University [16] and PM from RWCP (Japan) [14]. All these systems bypass the operating system to shorten the communication critical path and to limit, or avoid completely, memory copies for bandwidth improvements.

Today these techniques are well-known and people are working on keeping the same performance while becoming more general. For instance the main tasks are to allow a number of simultaneous users on the system, to scale to large numbers, to allow dynamic reconfigurations of the network and the partners involved in an application. The VIA [5] standard has been proposed by Compaq, Microsoft and Intel to fill these aims with a standard interface.

2.3 Experimentation platform

Our experimentation platform for the experiments presented in the next sections is made with 8 nodes, each with an Intel Pentium Pro 200 MHz, 64Mbytes of RAM and a 440FX chipsets, and a myrinet board with LANai 4.1 and 256Kbytes of memory. The test-bed is provided by the LHPC (Laboratoire pour les Hautes Performances en Calcul), a cooperation between ENS-Lyon and Matra Système Information.

3 Running the NAS benchmarks

The NAS parallel benchmarks (`http://science.nas.nasa.gov/Software/NPB/`) are a set of parallel programs designed to compare the performance of supercomputers. Each program try to test a given aspect of parallel computation that could be found in real applications. These benchmarks are provided both as single processor versions and parallel codes using MPI. We selected 3 different benchmarks and compiled them with MPI-BIP. They are then run on 1, 4 and 8 nodes of our Myrinet cluster described previously. Tables 1 and 2 give our measurements and comparisons with several parallel computers. Data for parallel computers come from the NAS web site.

IS (Integer Sort) sorts 8388608 keys distributed on the processors. The test uses a lot of small message communications and needs few processing power.

LU solves a finite difference discretization of the 3D compressible Navier-Stokes equations. A 2-D partitioning of the $64 \times 64 \times 64$ data grid is done. Communication of partition boundary data occurs after completion of computation on all diagonals that contact an adjacent partition. We have a relatively large number of small communications of 5 words each.

SP solves three sets of uncoupled systems of equations of size $64 \times 64 \times 64$, first in the x, then in the y, and finally in the z direction. This algorithm uses a multi-partition scheme. The granularity of communications is kept large and fewer messages are sent.

	MPI-BIP on PPro 200	IBM SP (66/WN)	Cray T3E-900	SGI Origin 2000-195	Sun Enterprise 4000
IS @ 4 proc	2.44 Mop/s	2.2	3.2	2.1	1.6
IS @ 8 proc	2.11	2.0	3.8	2.4	1.5
LU @ 4 proc	23.68 Mflop/s	59.0	67.6	96.8	36.9
LU @ 8 proc	22.73	57.2	66.4	103.3	37.3
SP @ 4 proc	10.10 Mflop/s	42.1	43.0	60.3	25.0

Table 1. Performance for NAS Benchmarks on various platforms.

	MPI-BIP on PPro 200	IBM SP (66/WN)	SGI Origin 2000-195	Sun Enterprise 4000
IS @ 4 proc	4.5	3.1	3.8	3.8
IS @ 8 proc	8.8	5.6	8.6	6.9
LU @ 4 proc	6	3.7	N/A	4.1
LU @ 8 proc	11.6	7.1	N/A	8.4
SP @ 4 proc	3.4	3.6	N/A	3.7

Table 2. Speedup for NAS benchmarks on various platforms.

The speedup values summarized in the table 2 show the very good performances of MPI-BIP and Myrinet. Super-linear speedups in our case can be explained by the large cache size and the super-scalar architecture of the PPro. Reported super-linear speedups for the Sun Enterprise show that a super-linear speedup is no exception. Regarding the favorable super-linear speedup for the LU test (compiled with `f2c`), we think that the typical working data sets used in the application do not fit in the cache in the sequential case but do in the parallel one. The behavior of MPI-BIP is excellent both for small messages and large messages. The speedup for the IS test with MPI-BIP is always better than those obtained on parallel machines. However, when a lot of computational power is needed (SP), a gap appears between traditional parallel machines and NOWs: the weak floating-point unit of the PPro 200 shows its limits!

4 High performance computing: the Thesee application

Thesee is a 3D panel method code, which calculates the characteristic of a wing in an inviscid, incompressible, irrotational, and steady airflow, in order to design new paragliders and sails. Starting from a sequential version [7], *Thesee* has been parallelized using the ScaLAPACK[2] library routines to be run on NOWs. The parallelization process has been done in a systematic manner to keep the development cost low. 3 parts have been identified.

- **Part P1** is the fill-in of the element influence matrix from the 3D mesh. Its complexity is $O(n^2)$ with n the mesh size (number of nodes). Each matrix element gives the contribution of a double layer (source + vortex) singularity distribution on facet i at the center of facet j.
- **Part P2** is the LU decomposition of the element influence matrix and the resolution of the associated linear system ($O(n^3)$), in order to calculate the strength of each element singularity distribution.
- **Part P3** is the speed field computation. Its complexity is ($O(n^2)$) because the contribution of every node has to be taken into account for the speed calculation at each node. Pressure is then obtained using the Bernoulli equation.

Each of these parts are parallelized independently and are linked together by the redistribution of the matrix data. For each part, the data distribution is chosen so as to insure the best possible efficiency of the parallel computation. The ScaLAPACK library uses a block cyclic data distribution on a virtual grid of processors. This solution provides a good load-balance because the processors receive matrix elements from different locations of the original matrix (as opposed to a classical full block decomposition). Communication overheads are reduced to a minimum by preserving the row and the column shape of the matrix (most of communication of 1D arrays can then happen without a complex index computation). Tests are run beforehand to choose the best grid shape and the best block size of the data distribution for our problem on each of the platforms. In the parallel version of *Thesee*, we used the following parameters for the data distribution of the LU factorization: the block size is 32×32 and the processor grid shape is a 1D-grid that gives the best overall computation timings.

4.1 Performance results

The tests are run with 3 different interconnection networks on 2 different platforms: (*i*) Ethernet and ATM network of SUN Sparc5 85MHz with Solaris and (*ii*) Ethernet and Myrinet network of Pentium-Pro 200MHz under Linux. On both systems, PVM and MPI were used. For IP over Myrinet, the LAM implementation of MPI is used. Otherwise it's an MPI-BIP user level implementation based on MPICH. The overhead for MPI or PVM is similar regarding the small set of primitives involved.

The efficiency of the fill-in of the influence matrix (part 1) and the speed field computation (part 3) is roughly the same on each configuration. The code for these parts is "embarrassingly" well suited for parallel execution and thus the speed-up is almost linear. On the other hand, the LU factorization that involves a lot more communications is highly dependent on the network software and hardware performances.

Comparison between Ethernet and ATM Table 3 presents the timing results for the whole computation on Ethernet and ATM with SUN Sparc 5 workstations.

		Timings(seconds) [speedups]	
	System size	PVM/IP/Ethernet	PVM/IP/ATM
4 Proc	902×902	16.3 [1.24]	13.5 [1.50]
2 Proc	902×902	18.8 [1.07]	16.2 [1.25]
Sequential	902×902	20.3	
4 Proc	1722×1722	68.3 [1.92]	61.1 [2.15]
2 Proc	1722×1722	108.9 [1.21]	95.9 [1.37]
Sequential	1722×1722	131.8	

Table 3. Timings and speedups of the whole computation (using the best block size).

As can be seen, both networks do not provide efficient communications for this kind of fine grained parallel computation where the communication/computation ratio is high. If this result could have been foreseen for the Ethernet, the ATM test shows that high-speed networks designed for long-distance communication are not well-suited to system-oriented communication. Although ATM provides more throughput, the gain obtained is small because the startup time for each communication on this network is similar to the one on Ethernet. Unfortunately, this startup time represents the larger part of the communication delay.

Comparison between Ethernet and Myrinet Table 4 presents the timings results obtained on the Myrinet test-bed along with the BIP software. Since the communication/computation ratio is rather high, we expect better performances than those obtained with ATM. The best results are achieved with MPI-BIP and we can see that IP-based implementation can not fully exploit the low latencies of the Myrinet hardware. With MPI-BIP, the low latency for a basic send communication ($9\mu s$ for this case) has an incredible impact on the speedup when compared to the Ethernet run. Super-linear speedup with 4 processors can be explained by a better cache hit ratio in the parallel version of the code. As the matrix is distributed cyclically on the processors, the computation occurs on blocked data that fits better in the cache during the LU decomposition, leading to a better use of the processor's pipeline units. One more reason is an increase in the overlapping of computation over communications in the parallel LU decomposition since the communication cost is greatly reduced with the BIP/Myrinet platform.

	Timings (seconds) [speedups]		
System size	PVM/IP/Ethernet (on SUN Sparc)	PVM/IP/Myrinet (on Pentium Pro)	MPI/BIP/Myrinet (on Pentium Pro)
Sequential 902 × 902	10.0		
2 Proc 902 × 902	9.6 [1.03]	6.7 [1.49]	5.0 [1.97]
4 Proc 902 × 902	10.2 [0.97]	4.7 [2.10]	3.1 [3.24]
Sequential 1722 × 1722	87.4		
2 Proc 1722 × 1722	56.4 [1.54]	44.3 [1.97]	38.1 [**2.29**]
4 Proc 1722 × 1722	45.9 [1.90]	28.1 [3.10]	21.3 [**4.09**]

Table 4. Timings and speedups with Myrinet (using the best block size).

5 Distributed Shared Memory System on high performance network

5.1 The DOSMOS system

The purpose of Distributed Shared Memory systems (DSM) is to implement, on top of a distributed memory architecture, a programming model allowing a transparent manipulation of virtually shared data. Thus, in practice, a DSM system has to handle all the communications and to maintain the shared data coherence. We have developed an original programming environment called DOSMOS system (**D**istributed **O**bjects **S**hared **M**em**O**ry **S**ystem). DOSMOS is an object-based DSM system which allows processes to share in a transparent way a set of objects distributed and replicated over distant processors. Moreover, DOSMOS integrates novel features:

- **DOSMOS Processes**: Basically, a DOSMOS application is composed of two types of processes: **Application processes (A.P.)** contain and execute application code; **Memory processes (M.P.)** manage the whole DSM system, i.e. they provide A.P. with the objects they request and maintain data coherence.

- **Array allocation**: DOSMOS allows to manipulate both basic type variables (integer, float, char...) and distributed arrays which can be split into several "system objects", replicated among the processors. Various splittings are provided: by row, by column, by block and by cyclic block.
- **Weak consistency protocols**: for efficiency and scalability purposes, DOSMOS gives the opportunity to duplicate shared objects. These replicas have to be kept coherent. DOSMOS implements a weak protocol: the release consistency which provides two synchronization operators: *acquire* and *release*.
- **Hierarchical structuring of the application processes**: processes can be grouped into groups and sub-groups in order to optimize the management of the data coherence.

Previously developed on top of PVM and experimented on Ethernet networks [9], DOSMOS is now available on top of MPI and experimented on MPI-BIP on top of Myrinet networks.

5.2 Gram-Schmidt Application

We based our experiments on the Gram-Schmidt application which has been completely studied on Ethernet in [4]. We propose four parallel implementations of the Gram-Schmidt application (Fig. 1) :

- Trivial version: with no splitting of matrix and use of strong consistency protocols. Accesses to shared object are sequentialized.
- Release Consistency and Object Splitting: this version takes the benefit of splitting the large matrix in small adapted shared objects.
- Synchronization 1: also based on Release Consistency protocols and matrix splitting but the matrix object is also used for synchronization of processes.
- Synchronization 2: this optimized version requires an independent object to synchronize processes.

5.3 Experiments

Figure 1 demonstrates the benefit of using a high-speed LAN for a DSM system like DOSMOS. Improvements added to the original algorithm are not linked with the application performances. Due to low latencies provided by the Myrinet network combined with MPI-BIP, applications with weak improvements provide better speedups than version highly optimized (synchro2). However, fast communications are not enough for trivial applications (*trivial version* in Fig. 1). Figure 2 shows the scalability provided by DOSMOS on top of MPI-BIP. By increasing problem size, we also increase the efficiency of applications which benefit from large bandwidth provided by the Myrinet network.

Distributed Shared Memory systems require low-latency networks to provide high performance to applications. Figure 3 shows that DOSMOS can provide high performance distributed objects when combined to low-latency communication protocols like MPI-BIP[1]. However, management of distributed objects

[1] Sequential time is computed by using one DOSMOS AP and one MP.

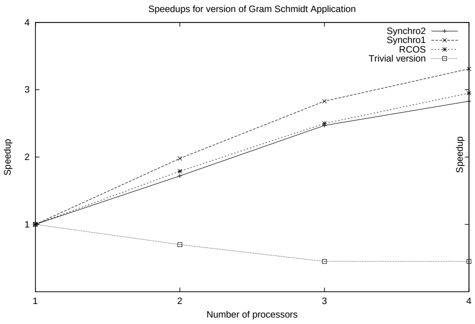

Fig. 1. Various improvements of Gram-Schmidt.

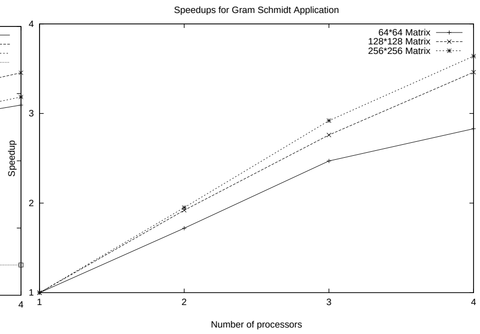

Fig. 2. Speedup of Gram-Schmidt by increasing problem size.

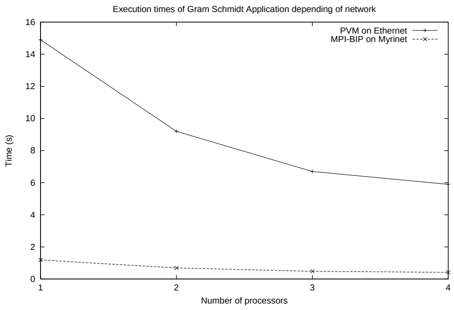

Fig. 3. Gram-Schmidt with Ethernet and Myrinet Networks.

Operation	**Latency**
Distant Read Simple Obj.	55
Distant Read Complex Obj.	140
Long Dist. Read Simple Obj.	103
Long Dist. Read Complex Obj.	340
Long Distant Acquire	95

Fig. 4. Latency in μs added by DOSMOS to shared object access.

(consistency, synchronization...) adds a latency to each distant access combined with latency of the network (Figure 4). Distant operations concern A.P. which access a shared object managed by their M.P. while Long Distant operations involve one A.P. and two M.P. to access distant objects. Access to a distant object requires a ping-pong communication between two nodes (which takes around 25μs with MPI-BIP). An additional latency is added by DOSMOS to manage complex objects (like arrays) to provide splitting and transparent access to distributed arrays. As can be ssen, the latency added by DOSMOS is kept reasonable and still compatible with the need of high performances by the applications.

By using a Distributed Shared Memory model, programming a parallel application is made easier but the added overheads usually penalizes the users to reach good performances. These experiments on a high-speed LAN show that designing a parallel application with a DSM system like DOSMOS can merge together ease of programming with high performances.

6 Parallel simulation of communication networks

With the emerging of Asynchronous Transfer Mode (ATM) technology, the performance evaluation of future large scale B-ISDN networks based on ATM is a great challenge. Parallel simulation techniques have been proposed during the last 2 decades to reduce the simulation time. These methods fall in two categories: the conservative approach [6] and the optimistic approach [8].

The parallelization process of the network model consists in spatially partitioning the entire network in n regions and to assign each region to a processor. Cells transfer from one region to another are represented by timestamped messages exchanged between the processors. The messages synchronization is performed conservatively. The system under study includes a routing algorithm with the introduction of link cost and dynamic routing functions that provide load-balancing routing features. The simulation of a routing algorithm implies to simulate (i) the mechanism that consists in constructing and updating the routing tables and (ii) the flow of cells that is handled by the network [12].

From the parallel simulation perspective, this application presents a very small granularity (processing time of an event) and requires a lot of small messages to be exchanged between processors (as opposed to more traditional high-performance computing applications where the computation part is larger). These messages are either model messages, e.g. cell exchanges between ATM switches, or synchronization messages, e.g. null-messages from the conservative kernel. Therefore the application typically requires very low latencies from the communication system. Previous tests on LANs such as Ethernet were not promising and so far only parallel computers were capable of showing interesting speedups.

6.1 Experimental results

In order to compare parallel computers and NOWs, experimental results on a Cray T3E and a Myrinet cluster are presented. The Cray is provided by the *Institut du Développement et des Ressources en Informatique Scientifique* (IDRIS) and consists of 256 processing nodes interconnected by a very low latency 3D torus. Each node on the Cray is a DEC Alpha EV5 with 128 Mo memory running at 300Mhz. The communication libraries are the native SHMEM (SHared MEMory) and MPI. The first one provides a latency of approximately $7\mu s$ while MPI shows a latency of $13\mu s$. The experimental NOW consists of Pentium Pro 200MHz interconnected by the Myrinet network described previously. The communication stack we used is native BIP and MPI-BIP. The latency is about $10\mu s$ with BIP and $17\mu s$ with MPI-BIP (for the message size of our application, 70 bytes). Table 5 and 6 summarize the results. For Myrinet, only 8 PCs have been used so far.

As can be seen, the sequential version run faster on a Pentium Pro 200MHz than on a DEC Alpha EV5 300MHz. The main advantage of using standard products is the availability of the most recent processors. As we stated before, parallel computers were interesting because they typically use customized high

np.	*time SHMEM (s)*	*speedup SHMEM*	*time MPI (s)*	*speedup MPI*
1	488	-	488	-
4	131	3.72	253	2.07
8	72	6.77	133	3.66
16	46	10.60	82	5.95
32	33	14.78	59	8.27
64	31	15.74	55	8.87
78	25	19.52	44	11.09

Table 5. Experimentations on Cray T3E.

np.	*time BIP (s)*	*speedup BIP*	*time MPI (s)*	*speedup MPI*
1	321	-	321	-
2	187	1.71	308	1.04
4	121	3.65	210	1.52
6	64	5.01	107	3
8	51	6.29	88	3.64

Table 6. Experimentations on Myrinet.

performance interconnection networks. The low latency on the Cray has a direct impact on the performances of the simulator. However, what can be seen is that results on the Myrinet cluster are close to those obtained on the Cray. Using BIP shows very interesting speedups (almost linear with 4 and 6 processors) compared to the cost of the hardware used. We have no doubt that this kind of architecture is very promising because of their excellent performance/price ratio.

7 Conclusions

Networks of workstations represent a serious alternative to expensive parallel computers. In this paper, we mainly focus on a Myrinet-based cluster with the BIP software for optimized low-level communications. Although the technology has been ready for some time, it is still used only in a confidential manner outside of the high-speed network research community. Also, even if a lot of people use clusters, a vast majority of them stay with the traditional IP-based implementations of PVM or MPI. Experiments have shown that if the hardware technology has the potential for high performance communications, the communication software layer must be well designed to fully deliver the maximum of performances to the application. One of the principal aims of this paper is to show through a number of applications the maturity of the more recent technologies, and to provide data to help them widespread to the end-users. Generalizing the use of the faster systems available will provide to the end-user community a way to reach a higher level of scalability, and to make parallel solutions usable for a wider range of applications, especially those that require fine grain decomposition.

References

1. Anderson, T. E., Culler, D. E., Patterson, D. A., the NOW Team: The Case for Networks of Workstations. IEEE Micro Magazine, February 1995.
2. Blackford, Choi, Cleary, d'Azevedo, Demmel, Dhillon, Dongarra, Hammarling, Henry, Petitet, Stanley, Walker, and Whaley: ScaLAPACK Users' Guide. SIAM, 1997. `http://www.netlib.org/scalapack/`.
3. Boden, Cohen, Feldermann, Kulwik, Seitz, Seizovic, and Su. MYRINET: A Gigabit per second Local Area Network. *IEEE-Micro*, 15:29–36, February 1995.
4. Brunie, L., Restivo, N., Reymann, O.: Programmation d'applications parallèles sur systèmes à mémoire distribuée partagée et expérimentations sur réseaux hautes performances. In *Calculateurs Parallèles*, **9**(4), pages 417–433. December 1997.
5. Buonadonna, P., Geweke, A., Culler, D.E.: Implementation and analysis of the virtual interface architecture. In *SC'98: High Performance Networking and Computing: ACM/IEEE SC'98, Orlando, USA*, november 1998.
6. Chandy, K., Misra, J.: Distribution Simulation: A Case Study in Design and Verification of Distributed Programs. Trans. on Soft. Eng., **5**(5) 440–452.
7. Giraudeau L., Perrot G., Petit S., Tourancheau B.: 3-d air flow simulation software for paragliders. TR 96-35, LIP-ENS Lyon, 69364 Lyon, France, 1996.
8. Jefferson, D. R. : Virtual Time. ACM Trans. on Prog. Lang. and Sys., **7**(3) (July 1985) 405–425.
9. Lefèvre, L.: Parallel programming on top of DSM Systems : An Experimental Study. In *Parallel Computing - Environments and Tools for Parallel Scientific Computing III*, **23**(1-2), pages 235–249. April 1997.
10. Mainwaring, A. M., Culler, D. E.: Active messages: Organization and applications programming interface. `http://now.cs.berkeley.edu/Papers/Papers/am-spec.ps`, 1995.
11. Pakin, S., Karamcheti, V., Chien, A,: Fast messages (FM): Efficient, portable communication for workstation clusters and massively-parallel processors. *IEEE Concurrency*, 1997.
12. Pham, C. D., Brunst, H., Fdida, S.: Conservative Simulation of Load-Balanced Routing in a Large ATM Network Model. In Proceedings of PADS'98, May 26-29 1998, Banff, Canada, pp142-149.
13. Prylli, L., Tourancheau B,: BIP: a new protocol designed for high performance networking on myrinet. In *Parallel and Distributed Processing, IPPS/SPDP'98*. bf 1388 of *LNCS*, pages 472–485. Springer-Verlag, April 1998.
14. Tezuka, H., Hori, A., Ishikawa, Y., Sato, M.: PM: An Operating System Coordinated High Performance Communication Library. In *High-Performance Computing and Networking'97*. **1225** of *Lecture Notes in Computer Science*, pages 708–717, April 1997.
15. von Eicken, T., Culler, D. E., Goldstein S. C., Schauser, K. E.: Active Messages: a Mechanism for Integrated Communication and Computation. Proceedings of the 19th Int'l Symp. on Computer Architecture, May 1992.
16. von Eicken, T., Basu, A., Welsh, M.: Incorporating memory management into user-level network interfaces. TR CS Dept., Cornell University, 1997.

This article was processed using the LaTeX macro package with LLNCS style