A Performance Evaluation of P-EDR in Different Parallel Environments*

M. Acacio, J.M. García and P.E. López-de-Teruel
Dpto. de Ingeniería y Tecnología de Computadores
University of Murcia
Campus de Espinardo, s/n 30080 Murcia (Spain)
{meacacio, jmgarcia, pedroe}@ditec.um.es

Abstract *This paper presents an exhaustive performance analysis of the P-EDR algorithm in some parallel machines available nowadays. EDR algorithm is a recent result of our Research Group. It constitutes an extension of the well-known Parzen method, which can be applied when dealing with uncertainty. Management of uncertainty implies heavy computational loads in terms of both calculus and storage, so a parallel version of the algorithm is more adequate to solve this problem in a practical time, especially for samples of large sizes. P-EDR represents our parallel implementation of EDR. Our tests go from an expensive and powerful MPP (as an IBM SP2 constitutes) to a cost-effective solution, as clusters of PCs represent. In these tests we have run the P-EDR algorithm in two distinct clusters of PCs each one with a different type of interconnection network: a low-cost 100 Mbit/s Fast Ethernet and a 1.28 Gbit/s Myrinet. Results obtained confirm parallel computing on networks of PCs as an attractive alternative to MPPs for cost/performance reasons.*

Keywords: Parzen Method with Uncertainty, Parallel Algorithm Evaluation, Cluster of PCs and MPPs, MPI, Distributed Memory.

1 Introduction

As the power of computers grows, the complexity of the problems that can be solved also increases. The application of high-performance computing for numerical intensive problems brings new issues that do not appear in standard computing. New algorithms and codes are required in order to exploit effectively the use of these novel computer architectures. Computational capability is now available to develop and study mathematical models for many different fields that are expensive, difficult, impractical or even impossible to consider by means of other methodologies. This is the case of many statistical problems. Up to now, some of these problems have been only studied theoretically, because available computers were not powerful enough to cope with them in practice. One of these problems is density estimation, all about when it is treated in several mathematical frameworks. This is the problem in which we are interested, but with a novel approach, the treatment of uncertainty, which though makes it computationally intensive, it is also true that the available computer power nowadays can make it affordable.

Parzen method is a well-known statistical technique used in non-parametric density estimation from observed samples. The application of this technique is very diverse: Pattern matching, machine learning, function approximation, and many other inference tasks. It can be applied in many different fields related to statistics, such as biomedicine, engineering, sociology, and so on. Nevertheless, one of the limitations of Parzen method is that it works only with exact data.

The EDR (Empirical variable Deconvolution by Regularization) [12] constitutes an adaptation of the Parzen method which can be applied when dealing with uncertainty (inaccurate samples). Treating uncertainty entails more demanding requirements, in terms of memory

* This work has been partially supported by the Spanish CICYT under grant TIC97-0897-C04-03

and amount of computation. This constitutes a very serious problem when datasets are large, which is unfortunately the case in practical applications.

Parallel computing has made possible to overcome this drawback. In [8], we presented P-EDR, a parallel version of our EDR algorithm for a message-passing parallel machine. We gave a detailed description of an efficient implementation of this algorithm. This implementation performed an efficient treatment of memory, in order to minimize communications between processes. We expected that our approach could obtain good scalability and therefore high speed-up values. Preliminary performance tests confirming these predictions were also presented.

This paper presents an exhaustive evaluation of the former P-EDR implementation in some of the parallel environments available nowadays, from an expensive and powerful MPP to a cost-effective solution, as a network of PCs represents.

MPPs and clusters of PCs have been tested in order to compare the performance differences between these environments. IBM SP2 represents MPPs category. And, two distinct clusters of PCs with two different types of networks were used: a 100 Mbit/s Fast Ethernet cluster of PCs and a 1.28 Gbit/s Myrinet of PCs [4].

Our results show how the performance gap between MPPs and NOWs is becoming narrower. Other investigations [1, 6, 11] also show how since the arrival of fast interconnection networks (as 1.28 Gbit/s Myrinet), cluster of PCs seem competitive with commercial MPPs, although at a fraction of the cost.

We have structured the paper as follows: First, we briefly introduce how P-EDR has been implemented. Next, we describe the parallel workbench environments employed in our tests. The results of these tests are then exposed and a comparison between MPPs and cluster of PCs is made. Finally, we present the conclusions of our work and some future ways.

2 The P-EDR implementation

In this section, we describe how the P-EDR algorithm has been implemented. A more exhaustive explanation can be found in [8].

As we mentioned before, EDR is a new iterative algorithm for non-parametric density estimation and P-EDR (that stands for *P*arallel *E*mpirical *D*econvolution by *R*egularization) constitutes its parallel implementation.

Our parallel algorithm has been designed in the SPMD [7] style. It is well known that message passing with SPMD constitutes the programming paradigm for distributed memory machines. It has been implemented in a message-passing parallel environment using the MPI library and C programming language. MPI [5] constitutes the current standard for writing parallel programs that are based on the message-passing model.

Distributed resources are optimally exploited, making the system capable of estimating densities from large databases of uncertain samples. An efficient treatment of the distribution of this structure among processors, together with a low communication cost scheme is intended in order to obtain a high scalability parallel algorithm.

Initially, P-EDR uses a database of N elements, which represents the original means and deviations (the N input samples with uncertainty). Then, the algorithm generates, after the necessary iterations, N means and deviations (referred as μ and σ). From these values, the new density estimation function ($p(x)$) can be calculated.

At each iteration the new values of means and deviations are recalculated. In order to obtain these values, a table of NxN elements must be filled in. This table stores the temporal values participating on the calculation of such means and deviations. This algorithm structure is ideal to take advantage of parallelism. We parallelized the algorithm in the following manner. Means and deviations can be calculated without the need of communications when a distribution by columns of this table between the processes is done. Each process is assigned a set of columns of the table, and it has to obtain the values of means and deviations for such columns. However, some communications are needed due to the fact that every process must know the values of means and deviations worked out by the rest of processes from the previous iteration to compute the new ones, and also to make the convergence test. Therefore, when a process obtains its local values, it must transmit them to the rest of processes. Here, there is a first synchronization point of the algorithm.

Furthermore, the convergence test can also take advantage of parallelism. This test is based on the calculus of the *L1* norm, the value that

measures the difference between two successive estimations of *p(x)*. It is obtained splitting a sufficiently wide interval (that captures the whole set of examples, with some margin at the extremes) into *I* small subintervals, and subsequently performing a numerical integration by the trapezoid method. Thus, these subintervals can be distributed between all the processes, so each of them calculates a local value of the *L1* norm. Nevertheless, calculation of the *L1* norm requires means and deviations to have been obtained, because these values are obviously used in the estimate of this rule. Moreover, the parallel calculus of the *L1* norm must be initiated when every process has obtained the values of means and deviations corresponding to the columns of the table it has assigned and it has sent them to the rest of processes. Once each process has computed the numerical value of the integral in the subintervals that it has associated, the global *L1* norm is obtained as the sum of the local values calculated by every process. Therefore, also here, there is a second point of synchronization.

Figure 1 summarizes the data-flow in the P-EDR implementation.

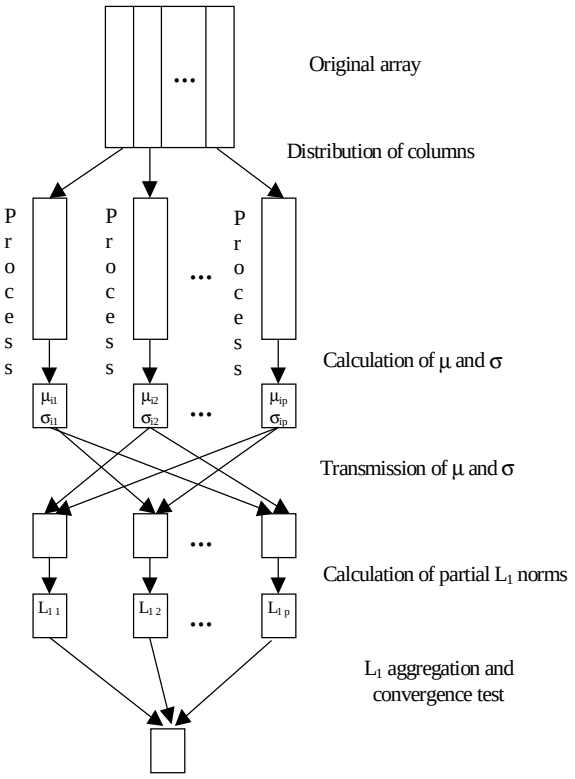

Figure 1. Data-flow in the P-EDR implementation

3 Our workbench environments

This paper carries out an exhaustive evaluation of the P-EDR algorithm. We have tested the parallel version of the algorithm for a great variety of database sizes, from small sizes (with less than 32 MB of main memory to execute the algorithm) to large databases (with a minimum of 128 MB of main memory needed). Besides, these tests have been done using a variable number of processors (from 1 to a maximum of 8), but always running one process in each processor.

Our tests have been done on distributed memory environments. Distributed memory is based on the fact that every processor access only its own memory, so programming is more complex, as data needed by a given processor may be located at the address space of a different one, and must be ordered and transferred through messages. Thus, it must be emphasize data locality in order to minimize communication among processes to achieve a good performance. However, it has two important advantages over shared memory systems: First, scalability, meaning that the system can be enlarger up to a higher number of processors than shared memory systems; and second, greater performance/cost ratio.

In spite of the great speed-up reached, high performance machines (MPPs) are very expensive. Most multiprocessor designs include custom-made components, which not only increase the costs, but also diminish its availability in the market. Supercomputers are beyond the economic of many scientific laboratories and universities that could need the calculation power they offer.

Recently, there has been a trend to use a Network of Workstation (NOWs) as a more cost-effective solution to high performance computing [2]. Our research efforts are focused in this important area.

The main advantage of this approach is that NOWs can be constructed from commercial off-the-shelf hardware. Lately, the evolution of CPU technology has brought high-end Personal Computers (PCs) to performance levels in the range of workstations at a very competitive cost, so they are already a very cost effective alternative to workstations as processing nodes in NOW platforms [13]. The major limitation of PC clusters is the high communication overhead in exchanging messages between nodes. Unlike a multiprocessor that has a custom low la-

tency/high bandwidth network, in a PC cluster a LAN with a lower bandwidth is used to connect the nodes. Therefore, we have used fast networks in the cluster of PCs, concretely a 1.28 Gbit/s Myrinet and a 100 Mbit/s Fast Ethernet.

First of all, we have run P-EDR on an IBM SP2. IBM SP2 constitutes a high performance supercomputer that employs the distributed memory paradigm. Consequently, message passing is used in this machine as the parallel programming model. The IBM SP2 we have used in our tests consists of 12 + 32 processors (42 thin160 and 2 wide), 12 GB total main memory, 128 KB cache memory, 494 GB disk and 27.41 Gflop/s theoretical peak. In this environment we could use two different versions of MPI: MPICH (from Argonne National Laboratory and Mississippi State University) [10] and IBM MPI-F. The latter promises to obtain better performance results, as it has been specifically adapted to this supercomputer [9]. We have employed IBM MPI-F in our tests.

We could not miss the opportunity of evaluating P-EDR using a cluster of PCs. Clusters of PCs could be seen as loosely coupled parallel machines, so it is imperative the use of a message passing programming model. We use Linux as the operating system on each PC. Two different versions of MPI have been used with Linux, one for each of the two clusters we have evaluated. MPICH [10] for the 100 Mbit/s cluster of PCs, and BIP from University of Lyon [3] for the 1.28 Gbit/s cluster of PCs.

In order to test the algorithm in such different environments, we took advantage of MPI portability. P-EDR source code does not need any change, we use exactly the same code for MPI-F, MPICH and BIP versions. This constitutes one of the most important characteristics of MPI.

As we have mentioned before, the bottleneck that we find when using a cluster of PCs for parallel computing is the interconnection network. Actually, this problem has been eased with the introduction of high-speed networks, as 155 Mbit/s ATM or 1.28 Gbit/s Myrinet.

We carried out the execution of P-EDR in two clusters of PCs with two different networks. First, it has been used a low cost network, a 100 Mbit/s Fast Ethernet (using a 3Com 905-network adapter), in a cluster of PCs made up of 7 Pentium 200 MHz processors, each one with 64 MB main memory and 256 KB cache memory. Then a 1.28 Gbit/s Myrinet, which represents a medium-high cost network, has been tested with a cluster of PCs made up of 6 Pentium II 350 MHz processors, each one also with 64 MB main memory and 512 KB cache memory.

Both sequential program and parallel MPI program have been compiled on each environment with the –O2 optimization compiler option.

Table 1 summarizes the main characteristics of the different environments used.

	IBM SP2	**Myrinet cluster**	**Fast Ethernet cluster**
Node type	Thin160	Pentium II 350 MHz	Pentium 200 MHz
Memory	256 MB	64 MB	64 MB
Cache	128 KB	512 KB	256 KB
MPI	IBM MPI-F	BIP 0.95c	MPICH 1.0.13
Operating System	AIX 4.2.1	Linux 2.0.35	Linux 2.0.36
Nodes	28	6	7

Table 1. Description of the workbench environments

4 Results

In this section we present a detailed analysis of the results obtained when running our P-EDR algorithm implementation on parallel environments introduced in previous section. Execution times and speed-up reached when executing P-EDR implementation in these environments are showed. Different database sizes have been used in our tests: small sizes (with 500 and 1000 elements), a medium size database of 1500 items, and large databases of 2000 and 2500 elements which need 128 MB main memory to complete their executions (in the sequential version).

P-EDR algorithm has been run on each test with a value of h of 0.5 and an accuracy factor of 0.001 (for more details, see [8]).

Finally, we present a comparison between IBM SP2 and clusters of PCs from results obtained.

4.1 The IBM SP2 results

We have employed in our test the IBM SP2 of the CESCA-CEPBA, at Barcelona. Parallel program has been inserted inside the Parallel Queue, and executed with different number of nodes (from 2 to 8). In order to corroborate the good scalability of P-EDR implementation, we have also used 16 and 28 nodes in our tests.

Table 2 shows the times obtained (in seconds) when running P-EDR algorithm on the IBM SP2. Figure 2 shows graphically the evo-

lution of the speed-up with respect to the number of processors.

Speed-ups reached are very significant especially when large-size databases are used (nearly linear for 2500 elements). The good scalability expected from our P-EDR algorithm is confirmed because of the speed-ups obtained when using 16 nodes in the executions (almost 13 for 2500 elements), and 28 (more than 19 for 2500 elements) nodes are using in the calculus.

	Size				
Nodes	**500**	**1000**	**1500**	**2000**	**2500**
1	5,00	20,00	44,00	79,00	123,00
2	3,00	11,50	25,00	44,00	69,00
3	2,67	8,00	17,34	28,34	44,00
4	2,25	6,00	13,00	21,75	33,75
5	2,00	5,20	10,80	18,60	28,20
6	1,83	5,00	9,34	15,17	23,34
7	1,57	4,57	8,28	13,86	20,28
8	1,25	4,37	7,25	12,00	18,00
16	1,62	3,06	4,75	6,94	9,62
28	1,38	3,03	3,46	4,93	6,43

Table 2. Execution times (seconds) on an IBM SP2

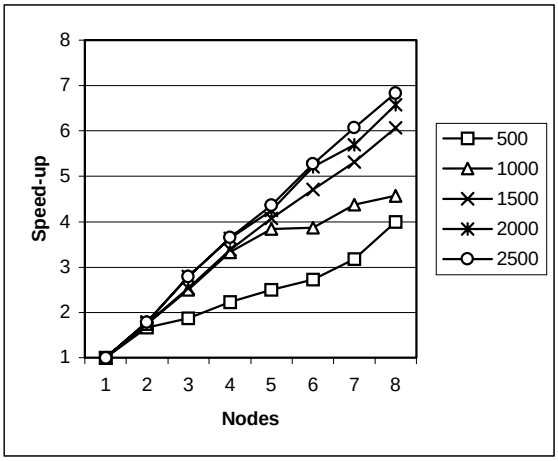

Figure 2. Speed-ups on an IBM SP2

4.2 The results on Cluster of PCs with Fast Ethernet

The 100 Mbit/s cluster of 7 PCs nodes used in our tests belongs to our Research Group. It constitutes a low-cost and broadly available solution for parallel computing.

Table 3 shows the times obtained (in seconds) when executing P-EDR algorithm on the 100 Mbit/s Fast Ethernet cluster of PCs described in section 3. Figure 3 shows graphically the evolution of the speed-up with respect to the number of processors.

Important speed-up values are reached in this environment, almost 6 when using 7 processors for 2500 elements. These values vary in a very narrow band, comprised between 4,3 (for 500 items samples) and 5,9 (for 2000 and 2500 items samples).

	Size				
Nodes	**500**	**1000**	**1500**	**2000**	**2500**
1	12,87	51,77	116,84	211,60	330,53
2	7,23	27,54	68,02	118,62	185,03
3	5,54	20,80	45,57	81,94	125,73
4	4,27	15,69	34,61	61,38	96,86
5	3,86	12,95	28,26	49,50	78,13
6	3,39	11,16	24,12	42,30	65,64
7	3,01	9,53	21,50	35,94	56,18

Table 3. Execution times (seconds) on the Fast Ethernet cluster of PCs

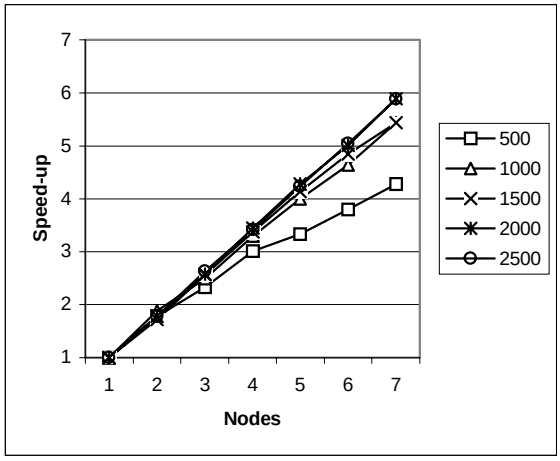

Figure 3. Speed-ups on the cluster of PCs with Fast Ethernet

4.3 The results on Cluster of PCs with Myrinet

We have also run P-EDR on a 1.28 Gbit/s cluster of PCs with 6 nodes. This working environment belongs to the Computer Science Department of the University of Castilla-La Mancha.

Table 4 shows the times obtained (in seconds) when executing P-EDR algorithm on the 1.28 Gbit/s Myrinet cluster of PCs described in section 3. Figure 4 shows graphically the evolution of the speed-up with respect to the number of processors.

In this case, we also obtain important speed-up values (approximately 4,5 when using 6 processors).

As it can be observed from figure 4, speed-up values reached when running P-EDR on this 1.28 Gbit/s Myrinet are very near. More pre-

cisely, we could place them inside a very narrow band, between 4,25 and 4,55 values.

	Size				
Nodes	**500**	**1000**	**1500**	**2000**	**2500**
1	6,04	24,16	54,30	97,30	155,18
2	3,15	12,53	28,19	50,33	79,58
3	2,78	10,53	24,34	43,37	67,76
4	2,05	8,09	18,16	32,27	50,29
5	1,66	6,54	14,67	26,07	40,66
6	1,42	5,60	12,30	21,70	34,10

Table 4. Execution times (seconds) on the cluster of PCs with Myrinet

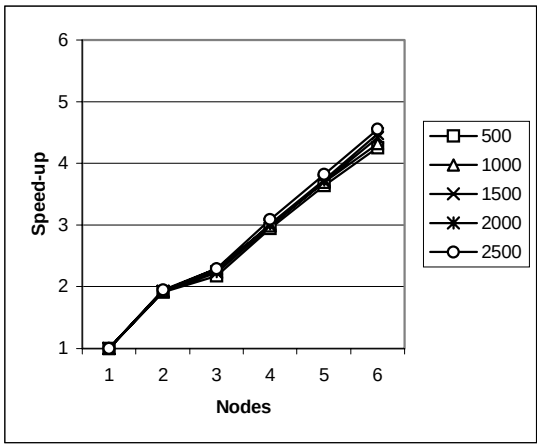

Figure 4. Speed-ups on the cluster of PCs with Myrinet

4.4 Comparison

In terms of speed-ups, our 100 Mbit/s cluster of PCs constitutes a tough opponent to IBM SP2. It outperforms the IBM SP2 speed-up for small-size databases (4.3 vs. 3.2 for 500 items), but not for medium/large-size databases, although in this case the speed-up obtained was not very far from IBM SP2 one (5.8 vs. 6.1 for 2500 items).

In terms of execution times, 100 Mbit/s Fast Ethernet cluster of PCs, which consists of Pentium 200 MHz computers, does not obtain so good results. In this case, sequential program completes its execution in about 330 seconds (with a database of 2500 items), whereas, for example, the IBM SP2 executes this sequential version employing 123 seconds. But, of course, it is sure that the use of more powerful PCs would reduce drastically these differences.

1.28 Gbit/s Myrinet cluster of PCs constitutes a serious alternative to the IBM SP2, in both execution time and speed-up terms. We can observe how it obtains better speed-ups than IBM SP2 when small-size databases are used (4.2 vs. 2.7 for 500 items and 4.3 vs. 3.87 for 1000 items). For medium-size databases (1500 items) they reach practically the same speed-up values (4.4 vs. 4.7). However, for large-size databases (2000 and 2500 items) IBM SP2 represents the best option. In terms of execution times, they obtain very similar results when small-size databases are used, but as we increase database sizes differences between these environments appear.

5 Conclusions and future work

Results obtained in our workbench environments confirm the good scalability expected from P-EDR. P-EDR speed-up on an IBM SP2 was very significant (nearly linear, when databases of 2500 elements are used). Even if we used a large number of nodes (16, 28 nodes) the reached speed-up continues growing significantly (almost 13 for 16 nodes and more than 19 for 28 nodes, also for 2500 elements databases). These results demonstrate that the parallel program scales very well (as we predicted in [8]), despite the overhead introduced by significant communications in each iteration.

For 100 Mbit/s Fast Ethernet cluster of PCs, the speed-up obtained was greater than 5.8 (for 7 processors and with 2500 elements databases), that is not so far from the IBM SP2 result. With the 1.28 Gbit/s Myrinet cluster of PCs better speed-ups than IBM SP2 ones are obtained for 500 and 1000 sizes. For 1500 size both IBM SP2 and 1.28 Gbit/s Myrinet cluster of PCs obtain the similar results. And for 2000 and 2500 sizes, the IBM SP2 outperforms to 1.28 Gbit/s Myrinet cluster of PCs.

With respect to execution times, the differences between the IBM SP2 and our 100 Mbit/s cluster of PCs are important. It is sure that this gap could be reduced using current Pentium II or Pentium III processors, as we could see with the 1.28 Gbit/s Myrinet cluster of PCs. This cluster was equipped with powerful 350 MHz Pentium II PCs. They were able to run the sequential program in 155 seconds (always with a database of 2500 items), very near to the 123 seconds need by the IBM SP2.

Summarizing, this paper presents an exhaustive evaluation of the P-EDR algorithm. The evaluation has been carried out using three different parallel programming environments, with different costs and performance. IBM SP2 constitutes the most expensive option, but it obtains the best results when large databases are used (in both speed-up and execution time). Also, we have tested a cluster of Pentium II 350 MHz PCs with a 1.28 Gbit/s Myrinet network,

that represents a medium-high cost solution, and obtains results very near the MPP when 500, 1000 and 1500 databases are employed (in terms of both speed-up and execution time). Finally, a cluster of Pentium 200 MHz PCs with a cheap and broadly available 100 Mbit/s Fast Ethernet has been measured. It obtains good speed-ups, but due the limitations of the processors used (Pentium 200 MHz), it can not compete with the IBM SP2 results in execution times.

Results obtained demonstrate how with much cheaper equipment we are able to solve certain parallel programs with very acceptable performances.

In brief we will replace these old-fashioned Pentium 200 MHz processors with modern Pentium II processors. An interesting future work would be measuring P-EDR in such new cluster with Fast Ethernet in order to study the differences between a 100 Mbit/s Fast Ethernet and a 1.28 Gbit/s Myrinet for this application.

Acknowledgments

This research has been performed using the facilities of the Centre de Computació i Comunicacions de Catalunya (CESCA-CEPBA).
The authors would like to thank the Computer Science Department of the University of Castilla-La Mancha for its important help when using their 1.28 Gbit/s Myrinet cluster of PCs, especially to Pedro García and Francisco J. Alfaro for their helpful comments and suggestions when configuring BIP.

References

[1] M. Acacio, O. Cánovas, J. M. García and P. E. López-de-Teruel. *An Evaluation of Parallel Computing in PC Clusters with Fast Ethernet.* In Procc. of the ACPC 99, LNCS 1557, pp. 570-571, 1999.

[2] T.E. Anderson, D.E. Culler and D.A. Patterson. *A Case for NOW.* IEEE Micro, vol. 15, nº 1, pp. 54-64, 1995.

[3] BIP Messages. http://lhpca.univ-lyon1.fr/bip.html

[4] N.J. Boden, et al. *Myrinet: A Gigabit-per-second Local Area Network.* IEEE Micro, vol. 15, nº 1, pp. 29-36, 1995.

[5] W. Gropp, E. Lusk and A. Skjellum. *Using MPI: Portable Parallel Programming with the Message Passing Interface*. The MIT Press, 1994.

[6] K. Langendoen, R. Hofman and H. Bal. *Challenging Applications on Fast Networks*. In Procc. of the 4th Int. HPCA, IEEE Computer Society Press, pp. 68-79, 1998.

[7] B. Lester. *The Art of Parallel Programming*. Englewoods Cliff, New Jersey, Prentice Hall, 1993.

[8] P.E. López-de-Teruel, J.M. García, M. Acacio and O. Cánovas. *P-EDR: An Algorithm for Parallel Implementation of Parzen Density Estimation from Uncertain Observations*. In Procc. of the 2nd Int. Merged IPPS/SPDP, IEEE Computer Society Press, 1999.

[9] J. Miguel, A. Arruabarrena, R. Beivide and J.A. Gregorio. *Assessing the performance of the new IBM SP2 communication subsystem.* IEEE Parallel and Distributed Technology, Vol. 4, nº4, pp. 12-22, 1996.

[10] *MPICH-A Portable Implementation of MPI*. http://www.mcs.anl.gov/Projects/mpi/mpich.

[11] S. Nagar, D. Seed, A. Sivasubramaniam. *Implementing Protected Multi-User Communication for Myrinet.* In Procc. of 2nd Int. Workshop CANPC'98, LNCS Vol. 1362, Springer-Verlag, pp. 31-44, 1998.

[12] A. Ruiz, P.E. López-de-Teruel and M.C. Garrido. *Kernel Density Estimation from Heterogeneous Indirect Observations*. Proceedings of the Learning'98, pp. 86-96, 1998.

[13] T. Sterling. *The Scientific Workstation of the Future May Be a Pile of PCs*. In Comm. of ACM, vol. 39, nº 9, pp. 11-12, 1996.