h-Relation Models for Current Standard Parallel Platforms

Rodríguez C., Roda J.L., Morales D.G., Almeida F.

Dpto. E.I.O. y Computación
Universidad de La Laguna - Tenerife - Spain

Abstract. This paper studies the validity of the BSP h-relation hypothesis on four current standard parallel platforms. The error introduced by the influence of the number of processors is measured on five communication patterns. We also measure the influence of the communication patterns on the time invested in an h-relation. The asynchronous nature of many current standard message passing programs do not easily fits inside the BSP model. Often this has been criticized as the most serious drawback of BSP. Based in the h-relation hypothesis we propose an extension to BSP model valid for standard message passing parallel programs. The use and accuracy of h-relation models on standard message passing programs are illustrated using a parallel algorithm to compute the Discrete Fast Fourier Transform.

1 Introduction

The development of a reasonable abstraction of parallel machines is a formidable challenge. A simple and precise model of parallel computation is necessary to guide the design and analysis of parallel algorithms. Among the plethora of solutions proposed, PRAM, Networks, LogP and BSP models are the most popular.

Section 2 introduces the BSP model. Sections 3 presents the four representative platforms used in our study: a 10 Mbits Coaxial Ethernet Local Area Network, an UTP Ethernet LAN, an IBM SP2 and a Silicon ORIGIN 2000. The IBM Scalable POWERparallel SP2 used in the experiments is a distributed memory parallel computer with 44 processors. Processors or nodes are interconnected through a High Performance Switch (HPS). The HPS is a bi-directional multistage interconnection network. The computing nodes are Thin2, each powered by a 66MHz Power2 RISC System/6000 processor. All the algorithms were implemented in PVMe [3], the improved version of the message-passing software PVM [2]. The Origin 2000 system we have used has 64 R10000 processors (196 MHz). Origin systems use distributed shared memory (DSM). To a processor, main memory appears as a single addressable space. A four-port crossbar switch interconnects the processor, the main memory, the router board and the I/O subsystem. The router board interfaces the node with the CrayLink Interconnect fabric. Both the 10 Mb/sec. Coaxial LAN and the UTP LAN are composed of Sun Sparc workstations at 70MHz running Solaris 2.4. The UTP LAN uses a FORE-SYSTEMS ES-3810 Ethernet Workgroup Switch that interconnects all

the computers of the LAN. A theoretical 10 point-to-point Mb/sec is guaranteed. As for the Origin 2000, the PVM version used in both LANs was 3.3.11.

Instead of following the common classical approach of using a library adapted to the proposed model, like Active Messages for the LogP [1] or the Oxford BSP library for BSP [5], we tried to check the validity of the models in current Standard Message Passing libraries. The influence of the number of processors in the time spent in an h-relation is measured for the most common communication patterns in section 3. Both the LogP and the BSP models disregard the effects of the pattern. Most algorithms can be built around a small set of communication primitives such as one to all, all to one, all to all, permutations and reductions if an appropriate data layaout is used. Is it reasonable to treat everything as the general case? Section 4 measures the impact of communication patterns in the h-relation hypothesis and estimates the BSP-like gaps and latencies for the considered architectures. The asynchronous nature of many PVM/MPI programs do not easily fits inside the BSP model. Often this has been criticized as the most serious drawback of BSP. In section 5 we propose an extension to BSP, the EBSP model, for current standard message passing parallel programs. The use of the BSP and EBSP models in conventional PVM/MPI programs will be illustrated in this section using the Fast Fourier Transform. Conclusions are presented in Section 6.

2 The Bulk Synchronous Parallel Model (BSP)

The BSP model [Val90] views a parallel machine as a set of p processor-memory pairs, with a global communication network and a mechanism for synchronizing all the processors. A BSP calculation consists of a sequence of supersteps. The cost of a BSP program can be calculated simply by summing the cost of each separated superstep executed by the program; in turn, for each superstep the cost can be decomposed into: (i) local computation; (ii) global exchange of data and (iii) barrier synchronization. The communication pattern performed on a given superstep is called an h-relation if the maximum number of packets a processor communicates in the superstep is h.

$$h = max \{ in_i \ @ \ out_i / \ i \in \{0, ..., p-1\} \}$$

Where @ is a binary operator, usually the maximum or the sum. Values in_i and out_i respectively denote the number of packets entering and leaving processor i. Both the particular operation @ and the size of the unit packet depend on the particular architecture. The operation @ is closer to the maximum (@ $= max$) for machines allowing a complete parallel processing of incoming/outgoing messages. The correct interpretation for each case depends on the number of input/output ports of the network interface. The results showed on this paper correspond to take as operator @ the sum.

The two basic BSP parameters that model a parallel machine are: the gap g, which reflects per-processor network bandwidth, and the minimum duration of a superstep L, which reflects the latency to send a packet through the network as

well as the overhead to perform a global synchronization. The fundamental of the BSP model lays on the h-relation hypothesis. It states that the communication time spent on an h-relation is given by gh. Let denote by W the maximum time spent in local computation by any processor during the superstep. The BSP model guess that the running time of a superstep is bounded by the formula:

$$Time\ Superstep = W + gh + L$$

3 The Influence of the Number of Processors in the Validity of the h-relation Hypothesis for Five Different Patterns in Four Different Architectures

In the following paragraphs we will study the variation of the h-relation time on the four architectures with the number of processors under five different communication patterns: Exchange (abbreviated E), PingPong (PP), OneToAll (OA), AllToOne (AO) and AllToAll (AA). Five hundred experiments were carried out for each architecture, each pattern and each number of processors. The h-relation size varies between 6720 and 1720320 bytes. In all the tables presented in this work, time is given in seconds.

The rest of the section is dedicated to the different communication patterns Π in $\{E, PP, OA, AO, AA\}$. For each pattern Π, two tables denoted $\Pi.1$ and $\Pi.2$ are presented. Sub-column labeled $Time$ in tables $\Pi.1$ gives the times for 8 processors using the patterns Π. It is observed that the time $T_\Pi(h)$ spent on an h-relation size, follows a linear equation $T_\Pi(h) = L_\Pi + g_\Pi h$. To obtain the general linear approach to the h-relation time we have computed the least square fit of the average times for the different number of intervening processors. Tables $\Pi.2$ present the values of L_Π and g_Π for the four architectures. Columns labeled $MaxErr$ in tables $\Pi.1$ contain the maximum percentage of error computed according to the formula:

$$MaxErr = 100 * (max_i\{|Time_{\Pi,i}(h) - (L_\Pi + g_\Pi h)|/\ i \in H_\Pi\}/(min_i\{Time_{\Pi,i}(h)\})$$

Where $Time_{\Pi,i}(h)$ denotes the time spent when i processors communicate according to pattern Π. Index i varies in $H_\Pi = \{2, 4, 6, 8\}$ processors for the Exchange and PingPong patterns, and i is in $H_\Pi = \{4, 6, 8\}$ for the OneToAll, AllToOne and AllToAll patterns. An injection pattern is characterized by the existence of a subset S of processors of the total set of available processors H, such that each processor p of S sends its message to a different processor $d(p)$ of H. We say that an injection pattern is an Exchange when: $d(d(p)) = p$ for any processor p in S. An injection pattern is called a PingPong if and only if $S \cap d(S) = \emptyset$. The PingPong pattern gives place to an $h = m$-relation, where m is the message size. For the Exchange, the h-relation is $h = 2 * m$. Table E.1 shows a factor of almost 3 between the times of the Origin and the IBM SP2. The same factor appears between the UTP LAN and the COA LAN. However, the values in column $MaxErr$ prove that the IBM SP2 is among the 4 architectures the most invariant in the number of processors. It is remarkably the technological advance observed in the low value of MaxErr for the UTP LAN when compared with the

Table 1. The Exchange Pattern (E.1)

Exchange	ORIGIN 2000		IBM SP2		UTP LAN		COA LAN	
h-relation	Time	MaxErr	Time	MaxErr	Time	MaxErr	Time	MaxErr
6720	0.000080	26.43	0.000310	16.78	0.013278	26.25	0.033285	160.92
26880	0.000227	17.78	0.000839	7.48	0.046496	14.18	0.132507	133.94
107520	0.001058	4.71	0.003103	4.04	0.173081	7.57	0.573673	172.24
430080	0.004230	4.32	0.012738	0.48	0.674435	5.03	2.323600	179.46
1720320	0.017021	4.25	0.050868	0.69	2.745955	7.02	9.226720	177.65

Table 2. Values of g_E and L_E (E.2)

	ORIGIN 2000	IBM SP2	UTP LAN	COA LAN
L_E	-1.1759E-05	6.1566E-05	-9.3670E-04	-3.2527E-04
g_E	9.8942E-09	2.9675E-08	1.5923E-06	5.3772E-06

178% of the COA LAN. The negative values of the Exchange latency L_E in table E.2 are due to the unsuitability of taking the byte as packet size unit. However, this choice was determined by the goal of comparing different architectures. The PingPong times appear in Table PP.1. As for the Exchange, there is an almost perfect invariance of the IBM SP2 communication time in the number of intervening couples. For this pattern $h = m$, opposite to what occurs for the Exchange, there is no parallelism between inputs and outputs. The time ratio between the IBM SP2 and the Origin diminishes from three to a factor of two. The same decreasing is observed for the UTP LAN and COA LAN. There is a decreasing both in the time and in the MaxErr columns for the COA LAN architecture. This is due to the smaller number of collisions.

The Personalized One to All communication pattern measures the outbound node performance. From Table OA.1 follows that the OneToAll communication time is better aproached by a linear by pieces function, but, fore sake of simplicity, we approach its behavior by a single linear function. Although it no appears in the tables, for small values of h, the time slowly grows with the number of processors. This is due to the heavier influence of the latency. For larger values of h, time slightly decreases with the number of processors, due to the increasing parallelism introduced in the communications. This phenomenon is specially outstanding for the COA LAN [7]. Observe the decreasing in the g value of the COA LAN to $1.072E-06$ from $4.08E-06$ for the PingPong pattern. For that reason the OneToAll is the fastest pattern for the COA LAN architecture.

The Personalized All to One Communication Pattern measures the inbound node performance. Each processor sends a distinct message of size m to the receiver processor. This experiment was implemented by making the receiver processor call the routine *pvm_recv*() $p-1$ times, and the senders making a *pvm_send*(). As for the OneToAll, we will approach the AllToOne time by a linear function $L_{AO} + g_{AO} * m * (p-1)$ in the h-relation size: $h = m * (p-1)$.

Table 3. The PingPong Pattern (PP.1)

PingPong	ORIGIN 2000		IBM SP2		UTP LAN		COA LAN	
h-relation	Time	MaxErr	Time	MaxErr	Time	MaxErr	Time	MaxErr
6720	0.000107	54.26	0.000386	8.58	0.015953	20.08	0.027919	107.54
26880	0.000454	7.53	0.001197	10.87	0.056722	6.97	0.112579	88.11
107520	0.001945	2.29	0.004811	0.87	0.220280	3.59	0.443787	80.56
430080	0.008132	2.95	0.018752	0.24	0.850265	2.37	1.773765	80.13
1720320	0.032797	4.16	0.074292	0.52	3.376577	6.48	7.023510	80.97

Table 4. Values of g_{PP} and L_{PP} (PP.2)

	ORIGIN 2000	IBM SP2	UTP LAN	COA LAN
L_{PP}	-7.5956E-05	1.5691E-04	6.2546E-03	5.6284E-03
g_{PP}	1.9071E-08	4.3467E-08	1.9609E-06	4.0818E-06

Table 5. The OneToAll Pattern (OA.1)

OneToAll	ORIGIN 2000		IBM SP2		UTP LAN		COA LAN	
h-relation	Time	MaxErr	Time	MaxErr	Time	MaxErr	Time	MaxErr
6720	0.000248	141.77	0.000383	47.26	0.022926	39.91	0.018732	25.57
26880	0.000354	20.09	0.001022	9.73	0.046733	11.31	0.033738	13.01
107520	0.000962	37.79	0.003597	4.37	0.170056	17.36	0.118535	10.19
430080	0.005203	8.60	0.014700	2.46	0.744115	23.66	0.471055	10.21
1720320	0.020398	9.06	0.059125	2.73	2.818493	17.40	1.846068	7.28

Table 6. Values of g_{OA} and L_{OA} (OA.2)

	ORIGIN 2000	IBM SP2	UTP LAN	COA LAN
L_{OA}	-3.4072E-05	2.0023E-05	1.0634E-02	7.6290E-03
g_{OA}	1.1704E-08	3.4320E-08	1.6493E-06	1.0725E-06

Table 7. The AllToOne Pattern (AO.1)

AllToOne	ORIGIN 2000		IBM SP2		UTP LAN		COA LAN	
h-relation	Time	MaxErr	Time	MaxErr	Time	MaxErr	Time	MaxErr
6720	0.000232	142.86	0.000449	26.87	0.015997	74.77	0.021609	65.20
26880	0.000281	37.37	0.001133	14.50	0.027941	12.98	0.031973	13.64
107520	0.000745	19.97	0.003719	5.63	0.114966	5.81	0.125891	8.10
430080	0.003445	6.19	0.014139	3.21	0.469320	5.78	0.512352	6.11
1720320	0.014098	9.78	0.057669	1.40	1.814855	7.00	2.006803	4.93

Table 8. Values of g_{AO} and L_{AO} (AO.2)

	ORIGIN 2000	IBM SP2	UTP LAN	COA LAN
L_{AO}	4.0212E-05	9.1160E-05	3.5069E-03	6.9769E-03
g_{AO}	8.1586E-09	3.3386E-08	1.0521E-06	1.1619E-06

Table 9. The AllToAll Pattern (AA.1)

AllToAll	ORIGIN 2000		IBM SP2		UTP LAN		COA LAN	
h-relation	Time	MaxErr	Time	MaxErr	Time	MaxErr	Time	MaxErr
6720	0.000411	181.56	0.000694	46.40	0.040892	104.27	0.046908	286.30
26880	0.000529	76.85	0.001287	20.63	0.059168	12.76	0.221194	120.10
107520	0.001091	16.81	0.003605	5.68	0.170425	7.96	0.655296	93.53
430080	0.004102	16.36	0.014036	2.87	0.700719	6.14	2.348122	78.44
1720320	0.019053	3.70	0.055054	3.39	2.749624	4.26	9.041233	73.94

Unless for the COA LAN, for all the architectures, the inbound bandwidth g_{AO} has decreased compared with the outbound bandwidth g_{OA}. This results to be the fastest pattern for the Origin and UTP LAN architectures.

In the Personalized All to All Communication Pattern each processor has to send $p-1$ different messages of length m to the other $p-1$ processors. Processor i cyclically sends the messages, to processors $i+1, i+2, ..., i-1$. The AllToAll time can be modeled by a linear function: $L_{AA} + g_{AA}2m(p-1)$. The IBM SP2 achieves its better performance for this pattern. While the performance of the UTP LAN only doubles the COA LAN performance for a pattern free of collisions like the PingPong, it is more than three times faster for this pattern. This is a consequence of the improving achieved in the bisection bandwidth.

4 The Influence of the Communication Pattern in the h-relation

BSP states that the actual times spent on these five patterns for the same h-relation have to be similar. The influence of the communication pattern in the time spent in an h-relation is shown in Table 11. To obtain the general linear approach to the h-relation time, we have computed the least square fit of the average times $T_{average}$ of the set Π of patterns and the different number of processors:

$$T_{average}(h) = \Sigma_{k\epsilon\Pi}(\Sigma_{i\epsilon H_\Pi} Time_{\Pi,i}(h)/|H|)/|\Pi|$$

These values of L and g appear in Table 12. There is a factor of ten between the BSP-PVM values for the IBM SP2 $g = 3.44*E-08$ and the corresponding Oxford BSP library values: $g' = 35 * E - 8$, $L' = 4.62 * E - 4$ for the same machine

Table 10. Values of g_{AA} and L_{AA} (AA.2)

	ORIGIN 2000	IBM SP2	UTP LAN	COA LAN
L_{AA}	2.3919E-05	3.9166E-04	1.2658E-02	8.7903E-02
g_{AA}	1.0984E-08	3.1578E-08	1.5957E-06	5.2131E-06

Table 11. AvErr and MaxErr for all the architectures

	ORIGIN 2000		IBM SP2		UTP LAN		COA LAN	
h-relation	AvErr	MaxErr	AvErr	MaxErr	AvErr	MaxErr	AvErr	MaxErr
6720	59.82	304.00	20.58	70.37	29.82	141.41	42.59	10.13
26880	25.75	103.00	8.24	19.34	15.81	20.07	42.30	86.66
107520	26.11	108.63	12.59	33.01	13.67	5.04	43.15	49.51
430080	25.97	120.06	10.76	30.38	13.41	19.12	42.96	57.48
1720320	24.79	126.06	10.81	29.61	13.83	81.88	42.81	228.00

[4]. Columns labeled $AvErr$ and $MaxErr$ respectively show the average and maximum errors defined as:

$$AvErr = 100 * ((\Sigma_{i \epsilon \Pi} |T_i(h) - (g * h + L)|/|\Pi|)/(\Sigma_{i \epsilon \Pi} T_i(h)/|\Pi|))$$
$$MaxErr = 100 * ((max_{i \epsilon \Pi} |T_i(h) - (g * h + L)|)/(min_{j \epsilon \Pi} T_j(h)))$$

The table proves that the time invested in moving data in the IBM SP2 and the UTP LAN is more independent of the specific communication pattern than in the other two architectures.

5 Extending the BSP Model to Current Standard Message Passing Libraries: EBSP

The barrier synchronization after each step imposed by BSP, does not completely agree with the way PVM and MPI programs are written. Still, PVM and MPI programs can be divided in "Message steps" that we will call "M-steps" in roughly the same sense than BSP "supersteps". In a "M-step" a processor $i = 0, ..., p-1$, performs some local computation, send the data needed by other processors and receives the data it needs for the next M-step. Processors may be in different M-steps at a given time, since no global barrier synchronization is used. However, as in pure BSP we assume that the total number of M-steps R, performed by all the p processors is the same, and communications always occur among processors in adjacent steps $k-1$ and k (computation on any processor can be arbitrarily divided to achieve this goal). The time $t_{s,i}$ when processor i finishes its step s is bounded by the "BSP-like-time" T_s given by:

$$\begin{aligned} T_1 &= max\{w_{1,i}\} + g * max\{in_{1,i}@out_{1,i}\} + L \\ T_s &= T_{s-1} + max\{w_{s,i}\} + g * max\{in_{s,i}@out_{s,i}\} + L \end{aligned} \quad (1)$$

Table 12. The values of g and L for all the architectures

	ORIGIN 2000	IBM SP2	UTP LAN	COA LAN
L	-1.5327E-05	8.0835E-05	3.8229E-03	1.3220E-02
g	1.2192E-08	3.4454E-08	1.5107E-06	2.4318E-06

where $i = 0, 1, ..., p-1$, $s = 2, ..., R$; @ is in $\{+, max\}$; $w_{s,i}$ is the time spent in computing by processor i in M-step s; $in_{s,i}$ and $out_{s,i}$ respectively denote the number of messages sent and received by processor i in the step s. Gap and Latency values g and L can be computed as proposed in the former paragraph. Thus the "BSP-like-time", T_R gave us an upper bound approximation to the execution time of a PVM/MPI program.

A closer bound to the actual PVM/MPI time $t_{s,i}$ when processor i finishes its s-th M-step is the value $\Phi_{s,i}$ given by the **Extended BSP model (EBSP)** we propose here. Let us define for a given step s and processor i the set $\Omega_{s,i}$ of **in-partners** of i in step s as $\Omega_{s,i} = \{ j \ / \$ processor j sends a message to processor i in step s $\} \cup \{ i \}$. The **EBSP** time of an PVM/MPI program is given by the formulas:

$$\begin{array}{ll} \Phi_{1,i} = max\{w_{1,j} \ / \ j \epsilon \ \Omega_{1,i}\} + g * h_{1,i} + L & \\ \quad h_{1,i} = max\{in_{1,j} + out_{1,j} / j \ \epsilon \ \ \Omega_{1,i}\} & i = 0, 1, ..., p-1, \\ \Phi_{s,i} = max\{\Phi_{s-1,j} + w_{s,j} / j \ \epsilon \ \Omega_{s,i}\} + g * h_{s,i} + L & \quad and \\ \quad h_{s,i} = max\{in_{s,j} + out_{s,j} / j \ \epsilon \ \ \Omega_{s,i}\} & s = 2, ..., R \end{array} \quad (2)$$

The total time of a PVM/MPI program in the **ESBP** model is given by $\Psi = max\{\Phi_{R,j} / j \ \epsilon \ \{0, ..., p-1\}\}$ where R is the total number of steps. Instead a global barrier synchronization, the EBSP model implies a synchronization among partners. Formula (2) becomes the BSP-like time of formula (1) when the family of sets $\Omega_{s,i}$ is the whole set of processors $\{0, ..., p-1\}$. (This is the case, if the BSP barrier synchronization implies $\Omega_{s,i} = \{0, ..., p-1\}$ for all i.

5.1 Example: The Fast Fourier Transform

We will illustrate the BSP and EBSP models using the parallel algorithm to compute the Fast Fourier Transform described in [6]. The total time predicted by the model for this algorithm is:

$$T_{log(p)+1} = \Psi = D(N/p) + FN/p \ log(N/p) + \sum_{i=0}^{log(p-1)} (L + g(N2^{i/p}) + VN2^{i/p}) \quad (3)$$

Table 13 contains the values of the three computational constants D, F and V. The good accuracy of the h-relation model for a 524.288 floats FFT example is shown in Table IV. Columns labeled MODEL in Table 14 present the time computed according to formula (3). Under the REAL label are the actual times.

Table 13. Division (D), sequential FFT (F) and combinations (V) constants

Constants	D	F	R
ORIGIN 2000	2.3928E-07	2.8453E-07	4.8103E-07
IBM SP2	5.5161E-07	5.8598E-08	8.6916E-07
UTP-COA LAN	6.0400E-07	1.4400E-06	2.6500E-06

Table 14. FFT real times versus h-relation model time

Times	2	4	8
ORIGIN 2000	1.61	0.96	0.85
MODEL	1.55	0.86	0.56
ERROR	3.68	10.85	34.10
IBM SP2	3.36	1.90	1.27
MODEL	3.21	1.83	1.18
ERROR	1.59	3.85	6.82
UTP LAN	12.36	11.31	11.03
MODEL	11.00	8.41	7.27
ERROR	11.01	25.65	34.13
COA LAN	11.91	10.80	10.94
MODEL	12.76	12.01	11.73
ERROR	7.14	11.17	7.22

Entries in columns ERROR give the error percentage computed by the formula $ERROR = 100 * (REAL - MODEL)/REAL$.

The accuracy of the model is specially good for the IBM SP2 and acceptable for the ORIGIN 2000 and UTP LAN. The error grows with the number of processors. The curious exception is the Coaxial LAN. On this architecture, the time of the model for the two first PingPong communications ($i = 0$ and $i = 1$) is over the actual time while it is under the actual time of the third communication ($i = 2$) producing a compensation that leads to a false accuracy. In fact, this is the only architecture where the model fails its prediction of a decreasing behavior in the actual time (see columns 4 and 8 in row COA LAN).

6 Conclusions

We have studied the validity of the h-relation hypothesis on four representative different platforms: a 10 Mbits Coaxial Ethernet Local Area Network, an UTP Ethernet LAN, an IBM SP2 and a Silicon ORIGIN 2000. Current Standard Message Passing libraries have been used instead of specific BSP environments. The maximum and average errors introduced by the influence of the pattern and the number of processors, has been measured. The collective computation provided by MPI and the new version of PVM 3.4 makes feasible the use of a methodology compatible with the Bulk Synchronous Programming Model.

However, the more relaxed nature of many PVM/MPI programs does not easily fit inside the BSP model. We have proposed a new model, EBSP, that extends the BSP model to current standard message passing parallel programs. The use and accuracy of the BSP and EBSP model has been illustrated using a Fast Fourier Transform example.

Acknowledgments Part of this research has been done at C4 (CESCA-CEPBA) and at IAC in Tenerife.

References

1. Culler, D., Karp, R., Patterson, D., Sahay, A., Schauser, K.E., Santos, E., Subramonian, R., Eicken, T. LogP: Towards a Realistic Model of Parallel Computation. Proceedings of the 4th ACM SIGPLAN, Sym. Principles and Practice of Parallel Programming. May 1993.
2. Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Mancheck, R., Sunderam, V. PVM: Parallel Virtual Machine - A Users Guide and Tutorial for Network Parallel Computing. MIT Press. 1994.
3. IBM PVMe for AIX User's Guide and Subroutine Reference Version 2, Release 1. Document number GC23-3884- 00. IBM Corp. 1995.
4. Marin, J., Martinez, A. Testing PVM versus BSP Programming, VIII Jornadas de Paralelismo, Sept 1997. pp.153-160.
5. Miller, R., Reed, J.L. The Oxford BSP Library Users' Guide. Technical Report, Programming Research Group, University of Oxford. 1993
6. Roda, J., Rodriguez, C., Almeida, F., Morales, D.G. Predicting the Performance of Injection Communication Patterns on PVM. 4th European PVM/MPI Users' Group Meeting. Springer-Verlag LNCS. pp. 33-40. Cracow. Nov-97.
7. Roda, J., Rodriguez, C., Almeida, F., Morales, D.G. Prediction of Parallel Algorithms Performance On Bus-Based Networks using PVM. 6th EuroMicro Workshop on Parallel and Distributed Processing. pp. 57-63. Madrid. Jan-98
8. Valiant, L.G. A Bridging Model for Parallel Computation. Communications of the ACM, 33(8): 103-111, August 1990.