Assessing the Performance of the New IBM SP2 Communication Subsystem

J. Miguel[1], R. Beivide[2*], A. Arruabarrena[1], J.A. Gregorio[2*]

[1] Departamento de Arquitectura y Tecnología de Computadores
Universidad del País Vasco UPV/EHU
Apdo 649, 20080 San Sebastián (Spain)

{miguel, acparfra}@si.ehu.es

[2] Departamento de Electrónica y Computadores
Universidad de Cantabria
Avda. de los Castros s/n
39005 Santander (Spain)
{beividr, monaster}@atc.unican.es

Abstract

IBM has recently launched an upgrade of the communication subsystem of its SP2 parallel computer. This change affects the communication hardware (high-performance switch and interface adapters) as well as the communication software (MPI implementation). In order to characterize to what extent the execution times of parallel applications will be affected by these changes, a collection of benchmarks has been run on a SP2 with the old communication subsystem and on the same machine after being upgraded. These benchmarks include point to point and collective communication tests as well as complete parallel applications. The performance indicators are the latency and throughput exhibited by the basic communication tests, and the execution time in the case of real applications.

Keywords

Communication subsystem, message passing networks, massively parallel computers, performance evaluation, IBM SP2.

1 Introduction

A long time has passed since the high-performance computing community realized that the highest computation speeds at reasonable cost can only be reached via massive parallelism. During the eighties and early nineties a sort of "building euphoria" led to the

---

[*] This work has been done while R. Beivide and J.A. Gregorio were at the Department of Electrical and Computer Engineering, University of California, Irvine, as visiting researchers.

design and use of massively parallel processors (MPPs) with thousands or even tens of thousands of processing elements. However, nowadays the community seems to think that a few hundred processors represent an upper limit on the feasibility of parallel computing systems. Additionally, and mainly due to the lack of stability in the supercomputing sector, constructors are making relatively conservative decisions regarding the hardware and software elements of MPPs, as the only way of guaranteeing that their systems will survive in the market for a reasonable period. IBM's SP2 parallel computer, built around workstation-based hardware and software, represents one of the most successful approaches to MPPs, and seems capable of surviving in these difficult times [HP96].

Most current MPPs (including the SP2) have been shown to be suitable for running coarse grain parallel applications that interchange very large data structures. In many cases, these systems are used simply to increase the throughput of sequential jobs of multiple users sharing a machine. However, the big challenge for the SP2 and similar machines (distributed memory parallel computers) is to be efficient when running communication-demanding, medium and fine grain parallel applications, reducing their execution times.

Nowadays, and following the principle of allowing the production of portable software, MPPs are programmed using conventional imperative languages, enhanced with communication libraries such as PVM and MPI to implement message passing and synchronization among processes [Gei94, MPI94]. In this environment, the efficiency of parallel applications is maximized when the workload is evenly distributed among processors and the overhead introduced in the parallelization process is minimized: the cost of communication and synchronization operations must be kept as low as possible. In order to achieve this, the interconnection subsystem used to support the interchange of messages must be fast enough to avoid becoming a bottleneck.

Traditionally, latency and throughput are the two parameters used to indicate the performance of an interconnection network for MPPs. While throughput increases from one generation of MPPs to the next, a significant reduction of latency seems to be a tougher problem for designers and industry. Unfortunately, the performance of parallel applications is very sensitive to latency and, while an increase in throughput can help processing long messages, it does not help significantly when interchanging reasonably-sized messages of tens to thousands of bytes—i.e., several orders of magnitude smaller than those which are required to reach the maximum achievable throughput in current MPPs.

Most of the current attempts to measure and characterize the different components of latency in modern MPPs lead to the same conclusion: the interconnection network itself (routers, wires) is over-dimensioned compared to the nodes' ability to send and receive messages. In other words, the largest components of latency are *not* in the network. Network interfaces and communication software cause most of the message passing overhead. These two elements (I/O hardware and software) must be greatly improved to reduce message latency in MPPs and, thus, to achieve better performance when running parallel applications on them. The importance of this issue is being highlighted by several research and development projects, including [Cul96, Myr96].

Recently IBM has developed a new version of the high-performance switch which constitutes the interconnection network of the SP2. The upgrade affects the switch itself, and also the I/O adapters connecting the processors to the switch. Before that, IBM upgraded the operating system from AIX 3 to AIX 4. The latter includes a new version of MPI, specially tailored for the SP2; with AIX 3, MPI was available as an additional layer of software over MPL, the native IBM message passing library. Both changes are aimed to increase parallel application performance.

In this paper we report the experimental data obtained from running a collection of benchmark applications on a SP2, first with the old and after that with the new communication subsystem. This information is used to make a performance comparison between two SP2 computers whose only difference is the communication subsystem. We don't try to report an exhaustive evaluation of the new subsystem, nor to provide an exact characterization of latency components—that will be done in future papers. However, it is important to allow the community to know the preliminary experimental data, especially for those in search of a machine to run parallel applications, and those designing new MPPs (or re-designing existing ones).

This paper is structured as follows. In the next section a brief introduction to the IBM SP2 parallel computer is made, including the (limited) information available about the recent change in its communication subsystem. Then we describe the collection of programs that have been used to test the machine (Section 3). The results of running these programs are analyzed in Section 4. The paper ends with some conclusions (Section 5) and acknowledgements (Section 6). The reader can find in the References the access path to most of the parallel code used to elaborate this paper.

2 An Overview of the IBM SP2

The SP2 is a distributed memory parallel computer where processors (nodes) are interconnected through a communication subsystem. IBM offers several alternatives for this subsystem. It is possible to provide communication among processors using standard networking technology, such as Ethernet, FDDI or ATM. However, for high-end SP2 systems

IBM offers a *high-performance switch*, which provides better characteristics for parallel computing. Figure 1 shows the high-level system structure of the SP2 [Age95].

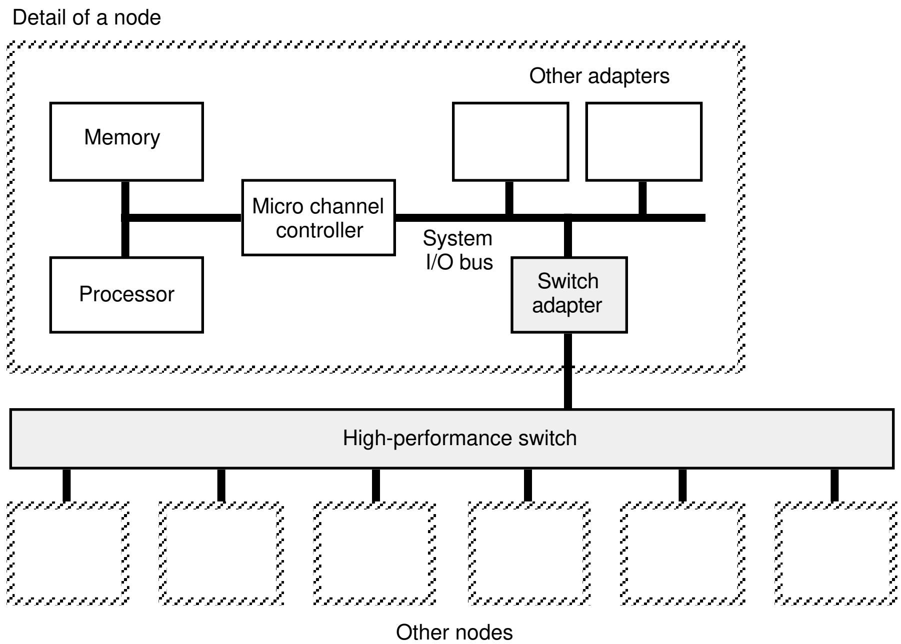

Figure 1. The SP2 system.

IBM also offers several alternatives for the nodes. We had access to the SP2 of C4 (see the acknowledgements section) composed of 32 THIN2 nodes, each of them powered by a 66 MHz POWER2 RISC System/6000 processor attached to 256 MB of memory by means of a 128-bit wide bus. These nodes have 32 KB of instruction cache, 128 KB of data cache and 2 MB of L2 cache. A Micro Channel controller governs the I/O bus which connects the processor/memory subset to devices such as disks, Ethernet networks and the high-performance switch, by means of the appropriate adapters.

The SP2 communication subsystem is composed of the high-performance switch, plus the adapters that connect the nodes to the switch. Both elements need to be carefully designed to allow them to cooperate without introducing bottlenecks. The adapter contains an onboard microprocessor to offload some of the work of passing messages from node to node, and some memory to provide buffer space. DMA engines are used to move information from the node to the adapter's memory, and from there to the switch link.

The high-performance switch is described by IBM as "an any-to-any packet-switched, multistage or indirect network similar to an Omega network". An advantage of this network is that the bisection bandwidth increases linearly with the size of the system (in contrast with direct networks such as rings, meshes and tori), so that it guarantees system scalability. The core of the network is a crossbar chip offering 8 bi-directional ports, that can be used to build small SP2 systems. For larger systems, boards composed of two stages with 4 of these chips each (for a total of 32 bi-directional ports) are used. These systems always have at least one stage more than necessary, in order to provide redundant paths (at least 4) between any pair of nodes. The reader can find in [Age95, Stu95] sketches of configurations of systems with 16, 48, 64 and 128 nodes.

A *frame* is a building block for SP2 systems. Each frame contains a switch board and up to 16 nodes. The SP2 available at C4 has two frames of 16 nodes each. One of the frames is used to run sequential and batch jobs, while the second one is available for running parallel programs. This is the one we used for the experiments reported in this paper. Originally the system had a communication subsystem with the following characteristics:

- The adapters were built around an Intel i860 processor with 8 MB of RAM. The theoretical peak transfer ability of the adapter was 80 MB/s, although the achievable maximum was 52 MB/s, due to overheads associated with accessing and managing the Micro Channel.
- The links to the high-performance switch provided 40 MB/s peak bandwidth in each direction (80 MB/s bi-directional), with a node-to-node latency of 0.5 μs for systems with up to 80 nodes.

Recently this system has been upgraded. The information about the new system available when writing this report was very limited, but we were able to confirm that the new adapters are built around a PowerPC 601 processor, allowing the doubling of the peak transfer bandwidth—reaching 160 MB/s. The new switch offers 300 MB/s peak bi-directional bandwidth, with latencies of less than 1.2 μs for systems with up to 80 nodes [Gar96].

Regarding the software environment, each SP2 node runs a full version of AIX, IBM's version of UNIX. It includes all the UNIX features, plus specific tools and libraries for programming and executing parallel programs. Our first tests on C4's SP2 were carried out with version 3 of AIX, which included MPL, an IBM-designed library for parallel programming using the message-passing paradigm [Sni95]. MPL is actually an interface that can be implemented over several communication subsystems. In particular, it can run over a software layer designed to make the best use of the high-performance switch. Alternatively, MPL might also run over IP (and, thus, over almost any communication media, including the switch). In this environment, an MPI library (MPICH, see [Bri95]) was also available, but it was an additional software layer over MPL.

With the introduction of AIX version 4, IBM decided to adopt MPI as the "native" language for programming SP2 systems, replacing the non-standard MPL. This way, applications programmed with MPI could run with less overhead (compared to the previous version). The introduction of the new switch was accompanied by a new version of the MPI implementation (although AIX itself remained without significant changes), in order to take advantage of the characteristics of the new hardware. MPL is still available to ease migration and to ensure the usability of old code. Both MPI and MPL can be used from Fortran 77 and C programs.

In this paper we will report the results obtained running the experiments and bechmarks described in section 3 with the following three configurations of C4's SP2 (Table 1):

	Communication subsystem	**Software**
o-v3	Old switch and adapters	AIX v3. MPICH implementation of MPI (on top of MPL)
o-v4	Old switch and adapters	AIX v4. Native version of MPI
n-v4	New switch and adapters	AIX v4. Native version of MPI

Table 1. SP2 configurations under test.

3 Experiments and Benchmarks

In order to make a fast—but fair—comparison between the two versions of the SP2, we selected and ran a small collection of test programs; most of them have already been used by other researchers, while some others have been prepared by us. When selecting these programs our aim was to perform a progressive evaluation of the machines, starting with simple, point to point communications and then going through collective communications, numeric kernels and fine-grain, non-numeric applications. The measurements obtained from each experiment have provided information about the latency and throughput of the communication channels and about the complete network. This information can be used as a first-order indicator for predicting how a change in the communication subsystem will affect the execution times of parallel applications. In the next subsections the selected benchmarks are briefly described.

3.1 Point to Point Communication

A first group of tests are based on the code provided by Dongarra to characterize point to point communications using MPI [DD95]. Two processors, 0 and 1, engage in a sort of ping-pong, with processor 0 in charge of the measurements. This processor reads the value of a wall-time clock before invoking a MPI_Send operation and then blocks in a MPI_Recv (meanwhile, processor 1 performs the symmetric operations). Once the latter operation finishes at processor 0, the clock is read again. Thus, the delay of a two-message interchange (one in each direction) has been measured; the latency is computed as one half of this time. The achieved throughput is also computed, considering the latency and the message size. These operations are done a few times to avoid warm-up effects, and then another 1000 times to average results. The message size is provided as an input parameter.

The program measures minimum, maximum and average latency and throughput. We have considered average values, because they are more representative of the performance the user can obtain from the machine; other authors [XH96] consider minimum values for latency because they are supposed to be free from the influence of the operating system and other users. In any case, minimum and average values are very close in most cases.

3.2 Collective Communications

Several MPI collective operations have been the object of detailed measurement, similar to that taken with point to point communication. We will pay special attention to two of them, broadcast and reduction, because the experiments performed with the others exhibit very similar behavior. In addition, a random traffic test has been performed. In all these experiments several messages might compete for the network resources as well as for accessing a common destination. Next we describe the details of the tests.

The **broadcast** test involves all the 16 processors performing a broadcast from processor 0 (the root of the operation) to all of them (including itself). The test performs 1000 iterations, again to average values. In each iteration, all the processors perform a broadcast (MPI_Bcast), with processor 0 designated as root, and then barrier-synchronize (MPI_Barrier). To compute the broadcast delay, the root measures the time from the moment the broadcast starts until the time the barrier finishes, and then subtracts a pre-computed barrier delay. The obtained throughput figures indicate the amount of information received by any of the participants.

The **reduction** test is like the previous one, but using MPI_Reduce instead of MPI_Bcast. In this case, the throughput figures consider the amount of information sent by any of the participants. MPI offers a collection of pre-defined operations (MPI_MAX, MPI_MIN, MPI_SUM, etc.) to be performed in the reduction, and also allows the user to define new operations. For this test, a user-defined, null operation, has been performed.

The last test of this group, **random-2 sets**, separates the available processors into two groups: those with even rank and those with odd rank. An even processor sends data to anyone of the odd processors, randomly chosen, which responds immediately. The delay of a two-way interchange, and the achieved throughput, is computed like in the point to point case.

3.3 Parallel Applications

In order to assess the behavior of the SP2 when running complete applications, we have selected four benchmarks (MG, LU, SP and BT) from the NAS Parallel Benchmarks (NPB) version 2 [Bai95], a shallow water modelling code (SWM) from the ParkBench suite of benchmarks [WF95], and a parallel simulator (PS) developed by our group [Mig95]. Next we briefly describe these programs.

Version 2 of the NPB can be obtained in source code form (in contrast with version 1), written in Fortran 77 plus MPI. The programs have been compiled and executed without any change in the source code [*]. Each benchmark can operate with several input problem sizes (number of grid points). The NPB specify three classes (A, B and C) depending on the problem size. In our experiments we used Class B (i.e., medium-size problems).

MG uses a multigrid method to compute the solution of the three-dimensional scalar Poisson equation. **LU** is a simulated computational fluid dynamics application which uses symmetric successive over-relaxation to solve a block lower triangular-block upper triangular system of equations resulting from an unfactored implicit finite-difference discretization of the Navier-Stokes equations in three dimensions. **SP** and **BT** are simulated computational fluid dynamics applications that solve systems of equations resulting from an approximately factored implicit finite-difference discretization of the Navier-Stokes equations. BT solves block-triangular systems of 5x5 blocks, while SP solves scalar pentadiagonal systems resulting from full diagonalization of the approximately factored scheme [Bai95].

SWM is a parallel algorithm testbed that solves the non-linear shallow water equations on a rotating sphere using the spectral transform method. It has been programmed, using the message-passing paradigm, in Fortran 77 plus MPI, and forms part of the ParkBench benchmark suite. The developers are Patrick H. Worley (Oak Ridge National Laboratory) and Ian T. Foster (Argonne National Laboratory). This benchmark includes several input files to select the problem size and the algorithms to use. We have used the medium-size problem, which requires approximately 1 GB of workspace when run with 64 bit precision, and 1000 Gflop. The default parallel algorithms are distributed Fourier transform and distributed Legendre transform.

PS is a parallel discrete-event simulator developed by our group to evaluate the characteristics of message passing networks with 2-D torus topology and cut-through flow control. The parameters of the simulator are basically the problem size (in terms of number of switching elements), the load of the network (in terms of a percentage of the maximum theoretical bandwidth: that of the network bisection) and the number of time steps to

[*] The appropriate *makefiles* are distributed with the code.

simulate. For the results reported in this paper, a torus of 32x32 routers is simulated for 40000 cycles. The load varies from 5% to 90%. This means that the number of messages needed to perform the simulation varies from 2.5 million to nearly 9 million. Processes are organized in a (logical) 4x4 torus. A process always communicates with its four logical neighbors. Communication does not follow any particular temporal pattern. Messages are short: just 32 bytes. The code has been written in C plus MPI.

4 Performance Evaluation

Latency and throughput are the basic parameters that characterize the applications' view of the performance of a communication subsystem. Both in conjunction determine the adequacy of a given system to execute a given type of parallel application. First, message latency imposes restrictions on the granularity of applications and, second, throughput imposes a limit on the maximum amount of information processes can interchange per unit of time. For this reason, in the following sections we will analyze the results of the above described benchmarks in terms of these two parameters, in order to show how the upgrade in the SP2 communication subsystem has affected them.

In this preliminary attempt to evaluate the performance improvement achieved with the new SP2 communication subsystem, we do not intend to perform an exhaustive analysis of each and every MPI function. The results discussed here are just a minimum kernel, with the aim of offering a first overview of the potential performance changes an application might experience.

4.1 Point to Point Communication

A first approach to assess the communication performance of a parallel computer is to measure the minimum time required to send a message between two processes located on different nodes. For this reason, many performance evaluation studies concentrate on point to point communication, in order to establish an initial comparison point among different platforms [Cul96, DD95, Hoc94, XH96].

Table 2 and Figure 2 show the results obtained running the point to point test described in the previous section. As can be observed in the table, the introduction of the new version of MPI implies a reduction in the software overhead. As a consequence, start-up times are lower and the latency is noticeably reduced when messages are short. However, for messages over 4 KB the new version performs worse (the explanation of this phenomenon can be found in

[Fra95]; we will deal with this later on in this section). The maximum achievable throughput remains the same with the old and the new MPI implementation.

The introduction of the new switch clearly brings about a reduction of latency for all length ranges. This latency reduction is specially significant for medium and long messages, thus allowing the achievement of a higher throughput.

L (bytes)	Latency (µs)			Throughput (MB/s)		
	o-v3	o-v4	n-v4	o-v3	o-v4	n-v4
0	59.37	43.59	39.39	0	0	0
8	59.30	45.33	44.38	0.13	0.18	0.18
16	59.42	45.31	42.78	0.27	0.35	0.37
32	60.10	46.90	46.31	0.53	0.68	0.69
64	61.66	47.11	44.84	1.04	1.36	1.43
128	64.54	50.54	49.58	1.98	2.53	2.58
256	80.38	63.16	51.79	3.19	4.05	4.94
512	95.59	78.16	61.95	5.36	6.55	8.27
1024	118.95	97.40	87.68	8.61	10.51	11.68
2048	165.12	134.44	114.83	12.40	15.23	17.83
4096	229.33	197.40	164.69	17.86	20.75	24.87
8192	359.22	393.11	319.74	22.81	20.84	25.62
10^4	415.45	448.76	351.56	24.07	22.28	28.45
10^5	2964.4	2997.4	1693.9	33.73	33.36	59.04
10^6	28318	28364	11587	35.31	35.26	86.30

Table 2. Average values of latency and throughput for point to point communication. L is the message length. Configurations (o-v3, o-v4, n-v4) are described in Table 1.

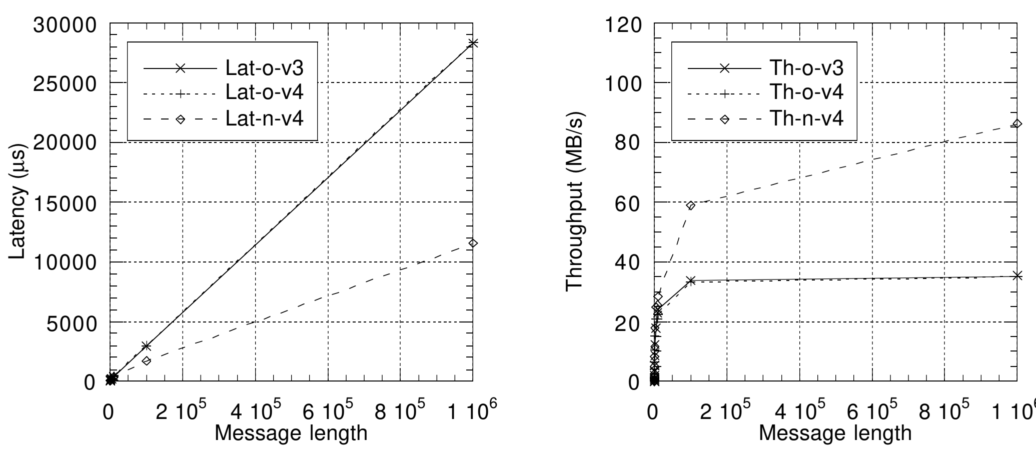

Figure 2. Latency and throughput for point to point communication. Data from Table 2.

A preliminary analysis of the results can be easily established through a characterization of latency. In general, the latency T of a message of length L can be decomposed into two components: the start-up time (T_h), which is the time used by the message header to reach the destination, and the spooling time (T_s), which is the time required to transmit the remainder of the message when the path from origin to destination has already been established.

$$T(L) = T_h + T_s(L) \tag{1}$$

The spooling time T_s is usually modelled as a linear function of L. Thus, $T(L)$ can be expressed as:

$$T(L) = T_h + T_b L \tag{2}$$

where T_b represents the time required to transmit a byte, once the path has been established. The other parameter of interest, throughput, is defined as:

$$TH = \frac{L}{T(L)} = \frac{L}{T_h + T_b L} \tag{3}$$

In general, the asymptotic behavior of T as a function of L is as follows:

$$T(L) = \begin{cases} T_h & \text{when } L \to 0 \\ T_b L = L / TH_{max} & \text{when } L \to \infty \end{cases} \tag{4}$$

TH_{max} being the maximum (asymptotic) throughput achievable from the communication subsystem. From equations (3) and (4), we see that TH_{max} can be calculated as $1/T_b$.

The latencies of Table 1 can be fitted * to equation (2), and the obtained maximum throughputs (for uni-directional, point to point communications) are:

	T_b	$TH_{max} = 1/T_b$ *(MB/s)*
o-v3	0.0282	35.5
o-v4	0.0282	35.5
n-v4	0.0120	83.3

The system behavior is not the same for short messages as for long messages. To see this more clearly, a detailed set of experiments have been carried out with the new switch with the results shown in Figure 3. In this case, minimum latency values are represented, because average values are too noisy to produce a clear graph.

* All the curve fittings have been done using least squares.

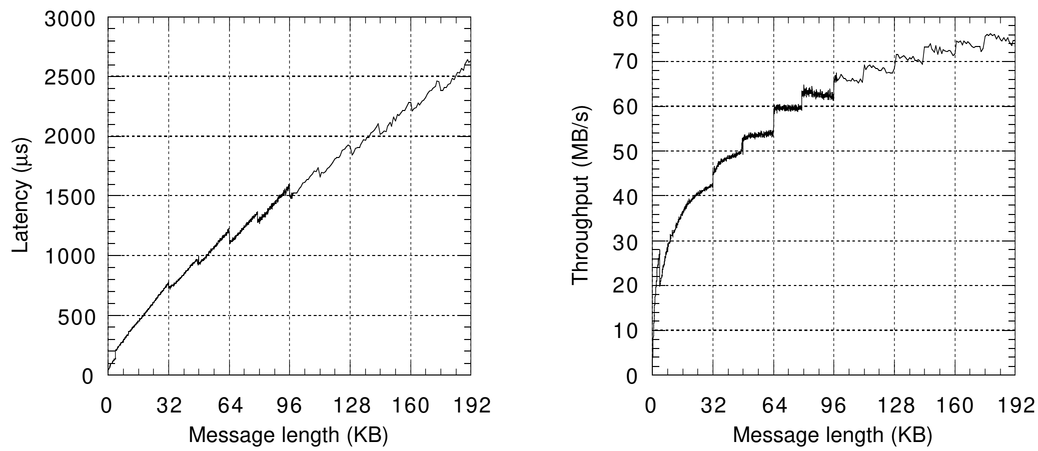

Figure 3. Minimum latency and maximum throughput with the new switch, for messages smaller than 192 KB.

A detailed analysis of the data indicates the existence of three different regions, defined by the message length (see Figure 4). In each one of these 3 regions, T can be fitted to equation (2), but with different parameters:

$$T_1(L) = \mathbf{47} + 0.0254\,L \qquad \text{if } 0 < L < 4\text{ KB} \qquad (5)$$

$$T_2(L) = 131 + 0.0195\,L \qquad \text{if } 4\text{ KB} < L < 32\text{ KB} \qquad (6)$$

$$T_3(L) = 390 + \mathbf{0.0115}\,L \qquad \text{if } L > 32\text{ KB} \qquad (7)$$

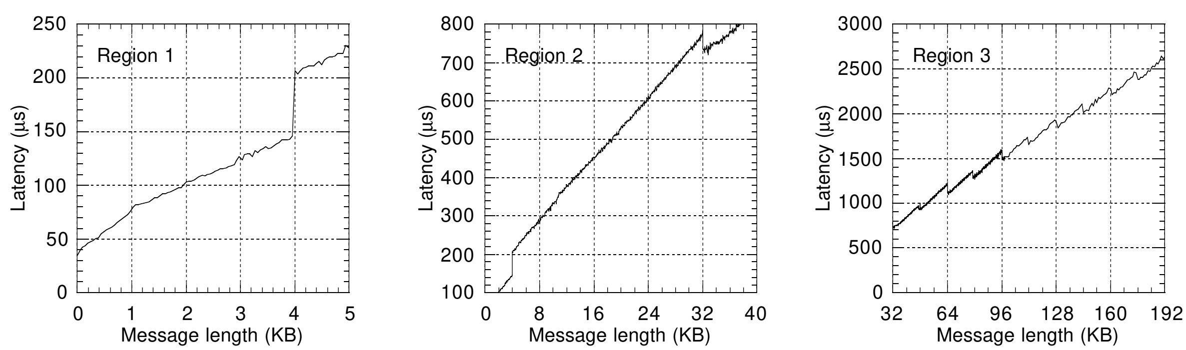

Figure 4. Message latency in the three different message length regions.

It is interesting to observe how T_h is smaller in the regions that correspond to smaller messages, while T_b is smaller in the regions that correspond to longer messages. This means that the system is optimized to minimize latency of short messages while maximizing throughput for long messages.

The above mentioned reference [Fra95] explains the change from region 1 to region 2 perfectly: for short messages, an "eager" protocol is used (i.e., a message is sent to its destination immediately), while for longer messages, a "rendezvous" protocol is used (i.e., a message is sent only when the receiver node agrees to receive it). Switching to the rendezvous protocol incurs in higher start-up costs, but reduces the number of times information is copied, thus reducing the cost per byte. This change of protocol comes with IBM's native version of MPI, so the discussion is also valid for experiments "o-v4". In the third region, the behavior of the latency is not as linear as in regions 1 and 2. The figures show a clear saw-edge effect, with jumps every 16 KB. At the moment, without any information from the manufacturer, we are unable to offer a reasonable hypothesis to explain this behavior.

From the set of equations, it is easy to see that the maximum throughput is achieved in region 3 (long messages), and can be computed from the parameter T_b in equation (7), while the minimum latency is obtained in region 1 (short messages) and is dominated by the parameter T_h in equation (5).

Going back to the comparison among several SP2 configurations, we can consider, for example, the data of Table 1 for $L = 32$ bytes, a quite short message. It can be seen that the latency is basically the same with the old and with the new switch. In contrast, for long messages (e.g., $L = 10^6$ bytes), the latency is noticeably reduced, and so the throughput is increased. In short, the effect of the communication subsystem upgrade has been: $T_{h\text{-}n} \approx T_{h\text{-}o}$ and $TH_{max\text{-}n} \approx 2.4\ TH_{max\text{-}o}$. Therefore, the upgrade has basically had an effect on the spooling time, which has been reduced to less than one half of the original. However, the start-up time used by the message header to reach its destination is almost the same. Summarizing, the channel throughput has been doubled, but the start-up time remains the same.

These results, in a first approach, tell us that coarse grain parallel applications requiring the interchange of very large data structures will experiment a significant performance improvement. This improvement will also be noticeable at the operating system level, and when using the system to increase the throughput of batch tasks—when the nodes run in a de-coupled fashion.

In contrast, those applications that require a frequent interchange of short messages will not experiment a significant reduction of execution time after the upgrade. In fact, it happens that the message size required to obtain a reasonable performance has been increased after the change. Hockney defines $L_{1/2}$ as the length of the messages that allows a utilization of one half of the maximum channel bandwidth [Hoc94]. With the old subsystem, $L_{1/2}$ was aprox. 3000 bytes (see Table 2). Now it has been increased to reach around 32 KB (see Figure 3). Therefore, the minimum message size required to efficiently run parallel applications has been increased in an order of magnitude. If the goal is to increase the performance of the SP2 when running applications requiring frequent interchange of short messages, it is imperative to reduce the start-up time in the same proportion as, or even more than, the maximum throughput has been increased.

Table 3 summarizes the parameters T_h, TH_{max} and $L_{1/2}$ for several well-known MPPs, including the three considered SP2 configurations.

Machine	T_h (µs)	TH_{max} (MB/s)	$L_{1/2}$
Convex SPP1000 (PVM)	76	11	1000
Convex SPP1200 (PVM)	63	15	1000
Cray T3D (PVM)	21	27	1502
Intel Paragon	29	154	7236
KSR-1	76	8	635
Meiko CS2	83	43	3559
SP2-o-v3	59	35	4000
SP2-o-v4	44	35	3000
SP2-n-v4	39	86	32000

Table 3. Latency and asymptotic throughput of some parallel computers. Data for the Convex, Cray, Paragon, KSR-1 and Meiko taken from [DD95].

4.2 Collective Communication

We have measured the characteristics of some representative communication patterns involving more than two processors: several MPI collective operations, and random traffic between two sets.

Regarding the **broadcast** collective operation, the experiments offered us the information summarized in Figure 5. It can be seen that the behavior is very much like that of the point to point case. The change in software means a change in the behavior of the latency curves: the use of the native MPI does not always reduce latency. Regarding the hardware upgrade, the graphs show a significant improvement—although it is only significant for messages longer than about 4 KB. Even with large enough messages, the latency reduction of the complete broadcast operation is not as spectacular as it is in the point to point case and, therefore, the throughput increase is not that impressive.

The performance of the remaining collective operations (including the **reduction**, described in the previous section), have been measured applying the same strategy used for broadcast, and show very similar behavior. Therefore, the same comments are applicable—so they are not repeated.

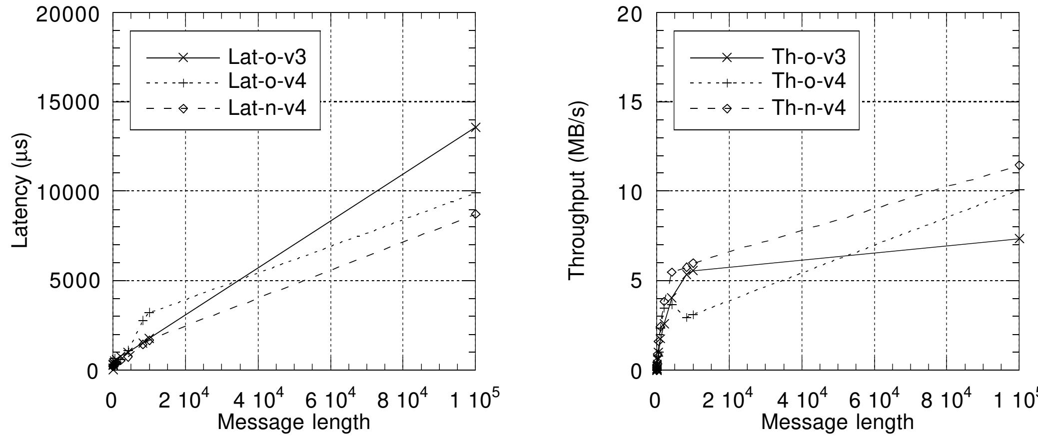

Figure 5. Latency and throughput for the broadcast operation.

The results obtained running the experiment with **random** communication between **two sets** of 8 processors each have been plotted in Figure 6. As in the above-mentioned broadcast case, significant latency reductions are achieved only for message sizes over 4 KB.

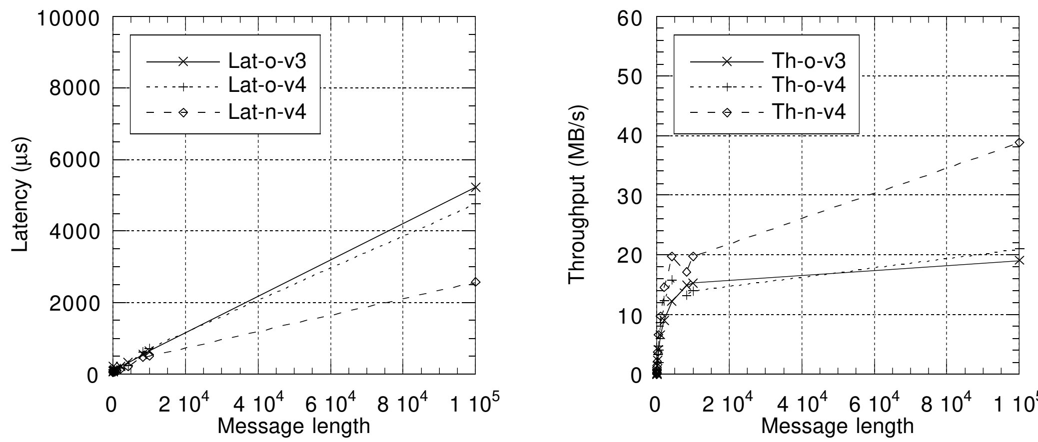

Figure 6. Latency and throughput for the random-2 sets test.

4.3 Parallel Applications

As we have commented in Section 3, the execution times of the 6 benchmark applications have been measured, in order to get an insight into how the changes in the communication subsystem may affect real parallel applications which combine computation with communication. From the data analyzed in the previous sections, the reader can infer that reductions in execution times will be marked only in those cases where the parallel applications require processes to interchange very long messages.

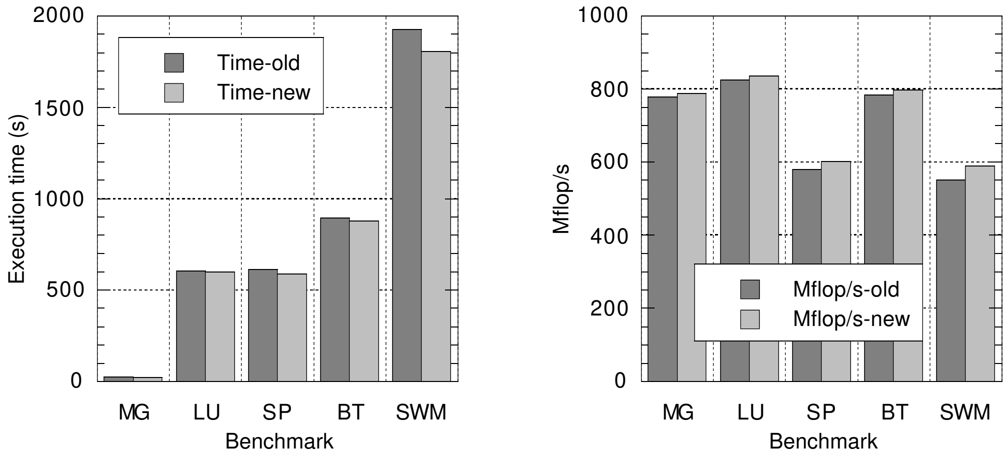

Figure 7. Results of the NPB and SWM benchmarks.

The graphs of Figure 7 show the execution time (in seconds) and performance (in Mflop/s) of BT, LU, MG, SP and SWM. We should mention that only the native version of MPI has been used for these tests; therefore, it is basically the change in the hardware that affects the performance. Results are as expected: improvements are quite modest in most cases. Things are not better with the parallel simulator: no performance increase has been measured. Figure 8 shows the execution times when varying the load of the simulated model (which is an indicator of the number of messages that the application manages). As a reference point for comparison, execution times of this experiment on 16 nodes of an Intel Paragon are 2.5 times those achieved on the SP2.

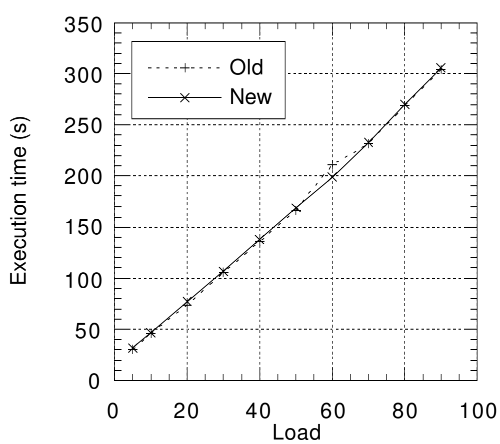

Figure 8. Execution times of the parallel simulator.

To summarize this section, we can state that a change in the communication subsystem will be effective for running parallel applications only if the designers are able to achieve important reductions on the start-up time, i.e., the latency for short and medium messages.

5 Conclusions

Our aim when writing this paper has been to offer a preliminary evaluation of the effect that the upgrade introduced by IBM in the communication subsystem of the SP2 will have on the performance of parallel applications. It should be clear that we don't pretend this analysis of the upgraded system to be comprehensive—it is mainly to offer a snapshot of its behavior. To do so, a collection of experiments have been performed using an SP2 frame with 16 THIN2 nodes, running some benchmarks that measure the performance of specific communication operations, plus others that represent typical parallel applications. The purpose of these experiments was to analyze progressively the effects of the upgrade on the overall system performance: from point to point communication, to collective operations, kernels of numeric applications (NAS Parallel benchmarks, SWM), and a non-numeric application with fine grain characteristics.

After analyzing the obtained measurements, we can conclude that an important improvement has been achieved in the bandwidth of the communication channels, which allows applications to reach throughput values much higher than those achievable with the old network—although this is only possible when applications interchange long enough messages (10 KB — 1 MB). In absolute terms, for point to point communications, the asymptotic throughput of the new communication subsystem more than doubles the previous one. In contrast, if messages are short, the improvement in bandwidth does not translate in better performance, because the start-up time has not been reduced. In other words, the latency for short and medium messages has not changed significantly from the old to the new system. Consequently, only those applications requiring a frequent interchange of massive amounts of information will experience clear reductions in execution time.

The results shown in this report confirm the conclusions of other researcher's studies: the real bottleneck in the MPP communication lies in the message-passing software and in the message interfaces that attach the nodes to the interconnection network. Therefore, these are some points where the MPP designers should focus their interest. In our opinion, (1) the overhead of message passing software has to be minimized, even if this means introducing changes in the architecture of node processors and (2) the message interface should be located as close to the processor as possible, connecting it to the system bus, or even to the bus between the off-chip cache and the processor.

Acknowledgements

We want to express our grateful acknowledgement to the following institutions:

- C4 (Centre de Computació i Comunicacions de Catalunya) for providing access to the machine under test, and for the technical support.
- CICYT. This research has been done with the support of the Comisión Interministerial de Ciencia y Tecnología, Spain, under contract TIC95-0378.
- DGICYT, the Dirección General de Investigación Científica y Técnica, which, with grant PR-95-190, allowed J.A. Gregorio to stay at UCI as Visiting Associate Researcher.
- The Department of Electrical and Computer Engineering, University of California at Irvine, for providing support and equipment access.

References

[Age95] T. Agerwala et al. *SP2 System Architecture*. IBM Systems Journal, Vol. 34, No. 2, 1995, pp. 152-184.

[Bai95] David Bailey et al. *The NAS Parallel Benchmarks 2.0*. Report NAS-95-020, Dec. 1995. The NAS Parallel Benchmarks version 2 are available at http://www.nas.nasa.gov/NAS/NPB.

[Bri95] P. Bridges et al. *User's Guide to MPICH, a Portable Implementation of MPI*. Argonne National Laboratory, 1995.

[Cul96] D.E. Culler, L.T. Liu, R.P. Martin and C.O. Yoshikawa, *Assessing Fast Network Interfaces*. IEEE Micro, February 1996, pp. 35-43.

[DD95] J.J. Dongarra and T. Dunigan. *Message-Passing Performance of Various Computers*. Technical report ut-cs-95-229. University of Tennessee, Computer Science Department. Available at http://www.cs.utk.edu/~library/1995.html. The benchmark for point to point communication, used in this report is available at http://www.netlib.org/benchmark/comm.shar

[Fra95] H. Franke et al. *MPI Programming Environment for IBM SP1/SP2*. IBM Technical Report RC19991, March 1995.

[Gar96] R. Garmire, *Personal communication.* IBM Power Parallel Division, Switch Development.

[Gei94] A. Geist et al. *PVM: A User's Guide and Tutorial for Networked Parallel Computing*. MIT Press, 1994. Available at http://www.netlib.org/pvm3/book/pvm-book.html.

[Hoc94] R.W. Hockney. *The Communication Challenge for MPP: Intel Paragon and Meiko CS-2*. Parallel Computing 20 (1994), pp. 389-398.

[HP96] J.L. Hennessy and D.A. Patterson, *Computer Architecture. A Quantitative Approach*. Ed. Morgan Kaufmann, 1996.

[Mig95] J. Miguel. *An Empirical Evaluation of Techniques for Parallel Simulation of Message Passing Networks*. PhD dissertation. Departamento de Arquitectura y Tecnología de Computadores. UPV/EHU, Oct. 1995.

[MPI94] Message Passing Interface Forum. *MPI: A Message Passing Interface Standard*. International Journal of Supercomputer Applications and High Performance Computing, 8(3/4), 1994. Special issue on MPI. Available at ftp://www.netlib.org/mpi/mpi-report.ps.

[Myr96] Myricom's Myrinet information. Available at http://www.myri.com.

[Sni95] M. Snir et al. *The Communication Software and Parallel Environment of the IBM SP2*. IBM Systems Journal, Vol. 34, No. 2, 1995, pp. 205-221.

[Stu95] C.B. Stunkel et al. *The SP2 High-Performance Switch*. IBM Systems Journal, Vol. 34, No. 2, 1995, pp. 185-204.

[WF95] P.H. Worley and I.T. Foster. PSTSWM v4.0 (ParkBench MPI version). Dec. 1995. Available at http://www.netlib.org/parkbench/compapps.

[XH96] Z. Xu and K. Hwang. *Modeling communication overhead: MPI and MPL performance on the IBM SP2*. IEEE Parallel & Distributed Technology, Spring 1996, pp. 9-23.