Message Passing Evaluation and Analysis on Cray T3E and SGI Origin 2000 Systems

M. Prieto, D. Espadas, I.M. Llorente, and F. Tirado

Departamento de Arquitectura de Computadores y Automatica
Facultad de Ciencias Fisicas
Universidad Complutense
28040 Madrid, Spain
E-mail: {mpmatias,despadas,llorente,ptirado}@dacya.ucm.es

Abstract. We present the results of different communication tests on some current parallel computers, the Cray T3E and the SGI Origin 2000. The aim of this paper is to study the effect of the local memory use and the communication network exploitation on message sending. For this purpose, we have first designed experiments without network contention to establish the achievable bandwidths. We have then modified this base experiment by increasing the contention of the network and by decreasing the spatial locality properties of the messages. We analyse these results taking into account the underlying architectures and we conclude with some hints for regular applications.

1. Introduction

Since MPI has become the standard for message passing on distributed memory machines, some work has been done on measurement of latency and bandwidth for send/recv operations and other global communications [1][2][3]. It is usually assumed that the data to be communicated is contiguous in memory. This is not always the case in a 2-D or 3-D regular grid domain application. The boundary data are not, in general, contiguous in memory and due to the necessary memory access, the cost of communication depends also on the pattern of data in the user memory that has to be sent. In this case, In this case, variations on classical echo test such as the proposed in [4] and the OCCOMM benchmark [5] provide better information. Indeed, the latter was designed to deal specifically with the exchange of boundaries in a 3-D regular application (an ocean model) and its main goal is to find out the best way of exchanging non-contiguous data using different message passing facilities. It consists of a data exchange between two processes using three different data patterns that can be found in a boundary exchange for three-dimensional decompositions of finite difference grids: contiguous data, contiguous data separated by a constant stride (called single-stride data) and non-unit stride vectors separated by a constant stride (called double-stride data).

We have followed a similar approach to the OCCOMM benchmark. We have extended its results with a detailed analysis of the influence of the underlying architecture and a study of the effect of different measured network loads has been

P. Amestoy et al. (Eds.): Euro-Par'99, LNCS 1685, pp. 173-182, 1999.

also done. In addition, at the time of writing this paper, there are no official results for the SGI Origin 2000 on the OCCOMM Web page[5].

This paper is organised as follows. The general components of a simple communication cost model are described in section 2. The essential aspects, network load and the spatial locality analysed in detail on the systems under study, are presented in sections 3 and 4 respectively. Using an example code and based on the previous analysis, the influence in the partitioning of a regular application is presented in Section 5. The paper ends with some conclusions.

2. Communication Cost

Message sending between two tasks located on different processors can be divided into three phases: two of them are where the processors interface with the communication system (the send and receive overhead phases), and a network delay phase, where the data is transmitted between the physical processors. The network delay depends on the number of hops between adjacent network nodes or switches that the message traverses, the rate at which the message data arrives at the destination and the overhead induced by competition of resources with other activities. The interconnection systems of current parallel computers and high-performance networks such as Myricom's Myrinet have taken advantage from the increasing integration density and so the effective bandwidth and latency are now hundreds of times faster than years ago. The peak link bandwidth is growing around 100 percent per year [6]. As the interconnection networks increase their bandwidth, the send and receive overheads are becoming predominant.

Therefore, the degree to which the communication contributes to execution time in a real application depends not only on the amount of communication but also on how the data are structured in the local memories due to their impact on the send and receive overheads and how the network topology is exploited due to contention problems. The following sections analyse in more detail these factors on Cray T3E and Origin 2000 systems.

3. The Cray T3E Message Passing Performance

The T3E-900 system used in this study has 40 DEC Alpha 21164 (DEC Alpha EV5) processors running at 450 MHz. The EV5 contains no board-level cache, but the Alpha 21164 has two levels of caching on-chip: 8 KB first-level instructions and data caches, and a unified, 3-way associative, 96-KB second-level cache [7][8][9].

The local memory is distributed across eight banks, and its bandwidth is enhanced by a set of hardware stream buffers that improve the bandwidth of small-strided memory access patterns and instruction fetching. Each node augments the memory interface of the processor with 640 (512 user and 128 system) external registers (E-registers) [7].

Although it supports shared memory, the natural programming model for the machine is message passing. The library message passing mechanism uses the E-registers to implement transfers, directly from memory to memory. E-registers

enhance performance when no locality is available by allowing the on-chip caches to be bypassed. However, if the data to be loaded were in the data cache, then accessing that data via E-registers would be sub-optimal because the cache-backmap would first have to flush the data from data cache to memory [8].

The processors are connected via a 3D-torus network. Its links provide a raw bandwidth of 600 MB/s in each direction with an inter-processor payload bandwidth of 480 MB/s[9]. However, as we have studied previously [10], the effective one-way bandwidth using MPI is smaller due to overhead associated with buffering and with deadlock detection. The maximum achievable bandwidth for large messages is only around half of the peak (around 300 MB/s). The test program employed in that study was slightly different to the code provided by Dongarra to characterise point to point communication [1][2]. Our test use all the processors available in the system where one of them was always the sender processor and the other, which varied, was the receiver. The sender initiated an immediate send followed by an immediate receive and then it waited until both operations had been completed. The receiver began by starting a receive operation and it replied with another message using an immediate send/wait combination. Because this exchange was repeated many times to average results, in order to avoid temporal locality effects, the `suppress` directive [11] was used to invalidate the entire cache (it forced all entities in the cache to be read from memory) of every iteration. We obtained almost the same asymptotic bandwidth for all the processor pairs.

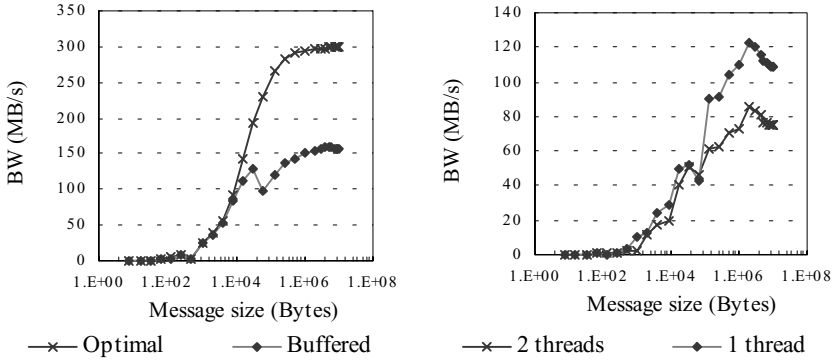

Fig. 1. Message Passing bandwidth in MB/s for contiguous data using buffered and synchronous communication in the CRAY T3E-900 (left-hand chart) and using 1 and 2 threads per node in the SGI Origin 2000 (right-hand chart) respectively.

The maximum bandwidth obtained in these measurements depends on the mode employed for sending and receiving data. Messages can either be buffered by the system or be sent directly in a synchronous way. The environment variable `MPI_BUFFER_MAX` can be used to control the maximum message size in bytes that will be buffered. The default value of `MPI_BUFFER_MAX` is no limit, i.e., it is only limited by the total amount of memory available in the system. A large limit is useful if an overlapping of computation with the buffered communication is possible. However it involves additional copies of data and increases the communication overhead. Indeed, the best performance attainable in buffered communications is only half of the maximum (around 160 MB/s). A size of 4099 bytes has been reported to be optimal [3]. This limit solves the trade-off between synchronous and buffered modes,

avoiding idle times for small messages and achieving the maximum bandwidth for point-to-point communications (around 300 MB/s) of messages above 2MB. The cache memory used in the buffered mode cause a reduction in the effective bandwidth when the internal buffers do no fit into the second level cache (from message larger than 64 KB in this experiment).

We have extended the previous experiment introducing different network loads. The behaviour of the network under load is complex, e.g. the T3E adaptive routing allows messages to be re-routed around a temporary hot spot, and so we only consider a qualitative description. In the second test, the sender and the receiver exchange their messages as in the previous one, but at the same time, the other processes, which belong to a different MPI communicator, perform a total exchange operation. We have employed the `MPI_Alltoall` function, a collective operation in which each process sends the same amount of data to every other process in the communicator. The degree of load introduced has been modified by changing the send/receive data count parameter of this function, which sets the number of elements that every process sends to the others. To get the total amount of data that is exchanged during an alltoall operation, one has to multiply this size by N(N-1), where N is the number of processes involved.

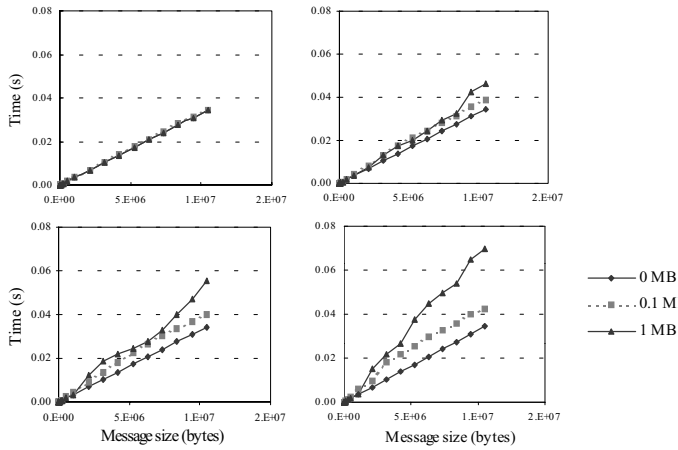

Fig. 2. Transfer times in seconds for varying message sizes and network loads for a point-to-point communication of contiguous data in the Cray T3E-900. The measurement has been taken under different network loads. The two processors involved in the exchange are 0 (top left), 1 (top right), 2 (bottom left) and 3 (bottom right) hops apart respectively.

As figure 2 shows, in a non-contention environment, adding hops does not have a significant effect on the communication bandwidth. However, as the network load increases, the effective bandwidth depends on the network distance between the two processors involved in the exchange for message sizes larger than 1 MB, although the measures are more erratic due to the network contention problems. Thus, 0 and 1 hop distances are advisable for neighbouring processors. An increase of the network load larger than that corresponding to a 1 MB alltoall operation does not produce a significant bandwidth worsening.

The behaviour under a loaded network is different using buffered communication. In this case, the effect of the network load is not so important, though the maximum bandwidth attainable in buffered communications is limited to 160 MB/s.

Figure 3 shows the effect of the spatial data locality. We also use the echo test with only two processors, but we modify the data locality by means of different strides between successive elements of the message. The stride is the number of double precision data between successive elements of the message, so stride-1 represents contiguous data. The measurements have been taking using the MPI datatypes (`MPI_Type_vector`) instead of the `MPI_Pack-Unpack` routines. Due to memory constraints the larger message is limited to 256 Kbytes, and although it is not big enough to obtain the asymptotic bandwidth for the stride-1 case, the measures are significant.

As in the first test, the performance depends on the communication mode employed. The bandwidth decreases with the stride using a buffered mode (`MPI_MAX_BUFFER` unlimited). For 256 KB messages, stride-1 bandwidth is around 5 times larger than stride-32. The extra memory copies needed in this mode explain this behaviour. The local memory is distributed across eight banks, and, when a cacheable load request is missed in the Scache, each bank supplies one word of the 8-word Scache line. For a stride larger than 8, only one word in an Scache line is useful, and thus, from stride 8, the effective bandwidth remains almost constant. We have measured no significant differences in bandwidth with the stream buffers disabled. Using the optimal value of `MPI_MAX_BUFFER` the performance is improved again.

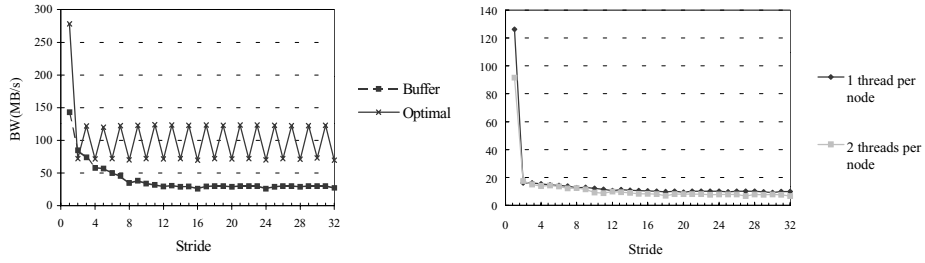

Fig. 3. Bandwidth in MB/s for non-contiguous data using buffered and synchronous communication in the CRAY T3E-900 (left-hand chart) and 1 and 2 thread per node in the SGI Origin 2000 (right-hand chart) respectively. The stride is the number of elements (double precision data) between successive elements of a 256 KB message in the T3E and a 2MB message in the SGI respectively.

Moreover, the bandwidth does not increase proportionally to the stride. The MPI use of E-register access explains this behaviour because the E-registers can be affected by bank conflicts in the local memory system. Strides that are multiples of 2, 4 or 8 are concentrated in 4, 2, or 1 of the eight banks respectively. Consequently, the effective bandwidth for odd strides (around 122 MB/s) is better than for strides that are multiples of 2 (around 72 MB/s), 4 (around 71 MB/s) or 8 (around 68 MB/s).

4. SGI Origin 2000 Message Passing Performance

We repeated these tests on an SGI Origin 2000, a cache-coherent distributed shared memory system. Each node of the Origin contains two processors connected by a system bus (SysAD bus), a portion of the shared main memory on the machine (512 MB in our system), a directory for cache coherence, the Hub (which is the combined communication/coherence controller and network interface) and an I/O interface called Xbow. The system used in this study has the MIPS R10000 running at 250 MHz. Each processor has a 32 Kbyte two-way set-associative primary data cache and a 4-Mbyte two-way set-associative secondary data cache [9][12][13][14].

The peak bandwidths of the bus that connects the two processors and the Hub's connection to memory are 780 MB/s. However the local Memory peak bandwidth is only about 670 MB/s. One important difference between this system and the T3E is that it caches remote data, while the T3E does not. All cache misses, whether to local or remote memory, go through the Hub, which implements the coherence protocol. The Hub's connections to the off-board network router chip and the I/O interface are 1.56 GB/s each. The SGI Origin network is based on a flexible switch called SPIDER, that supports six pairs of unidirectional links, each pair providing over 1.56 GB/s of total bandwidth in the two directions. Two nodes (four processors) are connected to each switch so there are four pairs of links to connect to other routers. Systems with 32 processors are created from a three-dimensional hypercube of switch where only five of the six links on each router are used. Xpress Links are optional cables that connect routers along the diagonal of the cube but the system used in this studied does not have these diagonal connections [9][12][13].

At the user level, due to the cache-coherency protocol and other overheads, the actual effective bandwidth between processors is much lower than the peak. [12][2].

The only different between the ping-pong test used in this system and the T3E one is the process employed to avoid temporal locality effects. Since there is not an SGI instruction similar to the T3E `suppress` directive, the cache is cleaned by reading a long auxiliary vector that is allocated and freed before each echo iteration.

For the purposes of buffering, MPI messages are grouped into two classes based on length: short (messages with lengths of 64 bytes or less) and long (messages with lengths greater than 64 bytes). The environment variables `MPI_BUFS_PER_PROC` and `MPI_BUFS_HOST` set the number of shared and private message buffers (16 KB each) that MPI have to allocate for each process and host respectively. These buffers are used to send long messages, for which one of them has to be different from 0. Unlike the T3E, we have not measured significant differences changing these variables.

Unlike the T3E, Silicon Graphics' MPI implementations use N+1 threads for an N processor job, although the first thread is mostly inactive. The environment variable `MPI_DSM_PPM` sets the number of MPI threads that can be run on each node. Only values of 1 or 2 are allowed. As figure 1 (right-hand chart) shows, using 1 thread per node, each thread has its own local memory and better performance is obtained since they do not contend for the same SysAD bus.

It is also interesting to note that the measured bandwidth decreases when the message sizes are larger than the L2 cache size. With the information that we have about the message passing implementation of SGI, we cannot be certain of the reason for this fact. Like in the T3E buffered communication mode, the internal message

buffers could cause this reduction since, for longer message sizes, they do no fit in the secondary cache and thus, extra main memory accesses are needed.

In this system, like in the T3E, the time required to send a message from one processor to another under an unloaded network varies little with the network distance between them, apart from the case where the two processors share their local main memory where the differences are larger, as we have explained before.

However, as figure 4 shows, the reduction (in percentage) in the effective bandwidth caused by extra network loads is larger than in the T3E. For example, in the worst case (a 1MB alltoall) and for a 2 MB message, the bandwidth is 4 times lower than under an unloaded network when the 2 processors are 3 routes apart. In this case for the 3TE, the reduction is only by a factor of 2. Besides, saturation is reached for a lower network load (corresponding to a 0.1 MB alltoall operation). We suppose that this behaviour could be improved by means of the Xpress Links.

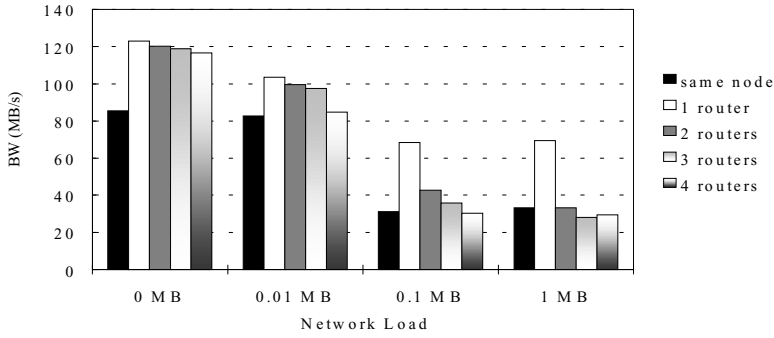

Fig. 4. Bandwidth in MB/s for a 2 MB contiguous message in the SGI Origin 2000. The measurement has been taken under different network loads and the two processors involved in the exchange are on the same node, 1, 2, 3 or 4 routes apart, respectively.

Unlike the T3E, the extra network load also has an effect when the two processes involved in the exchange are located on the same node, due to the cache coherence protocol.

Figure 3 (right-hand chart) shows the effect of the spatial data locality. The behaviour is similar to a buffered mode in the T3E. The second level cache line in this case is 128 Bytes (16 doubles) and thus, the actual bandwidth remains constant from stride 16, between 9 MB/s and 10.2 MB/s using 1 tread per node and between 6.7 MB/s and 8 MB/s using 2 threads.

5. Regular Partitioning

Summing up the results of the previous section, we can say that in the two systems under study, the bandwidth reduction due to non-unit-stride memory access is more important than the reduction due to contention in the network. The message passing performance is better in T3E where a cache coherence protocol is not employed and where it is possible to bypass the second level cache. Non-cache data access is

preferred for sending blocks of data that are too large to fit in the cache and for any irregular or strided pattern. How does these conclusions affect real applications?

As a sample problem, we have studied the numerical solution of a regular finite difference application [10]. The code was written in C, so a three dimensional domain is stored in a row-ordered (x,y,z)-array. It can be distributed across a 1D mesh of virtual processors following three possible partitionings: x-direction, y-direction and z-direction.

As is expected, the x and y-direction partitioning were found to be more efficient, because the message data exhibits a better spatial locality. For the 256x256x256 problem size, x- partitioning is found to be around 2 times better than z-partitioning using buffered communications. The difference between contiguous or non-contiguous boundaries is lower using a synchronous mode (around 1.6) as we have measured in the non-unit-stride ping-pong test. However, the performance results are just the opposite: buffered communications are more efficient than a synchronous mode. As in other parallel programs, all the processes of our application communicate at the same time. This can overburden the communication network and cause contention. Obviously, a buffered mode can relieve this problem.

In any case, we should also note that contention could be reduced using correct processor mapping. Although we have employed the process topologies created by the standard `MPI_Cart_create` function, using the option that allows process reordering to get the best performance, MPI on the Cray T3E does not do process reordering. An appropriate algorithm for building better Cartesian topologies has been recently proposed on [15].

In addition, the different E-register usage can cause some discrepancies. The application has some temporal locality and as we have explained in T3E description, if the data to be loaded were in the data cache, then accessing via E-registers involves the invalidation of the corresponding cache line, losing temporal locality advantages.

Although message-passing bandwidth is very important, we should also note that this difference is not only a message passing effect, e.g., partitioning also defines inner loop sizes, which can affect performance. In any case, by means of the MPP Apprentice performance tool we have found that the time spent in the initiation of message sending is 5 times larger in the Z-partitioning simulations using buffered communication.

Equivalent differences in the Origin 2000 can be observed, but are less important than in the T3E case. For the 256-element problem, X partitioning is only 1.3 times better. The large second-level cache of this system, which allows the best exploitation of the temporal locality, influences these results (it is less important the spatial locality).

Taking into account the effect of spatial locality, the choice of an optimal partitioning becomes a trade-off between the reduction of the send and receive overheads (mainly, spatial locality of the data to be exchange) and the efficient exploitation of the interconnection network. The correct processor mapping reduces contention, and thus 3D decompositions are the best choice for the T3E topology. In addition, for a 3D regular application the communication requirements for a process grow proportionally to the size of the boundaries, while computations grow proportionally to the size of its entire partition. However, higher dimensional decompositions require non-contiguous boundaries and they make messages greater in number and hence shorter. Since there is a fixed component of the send and receive

overheads, which is associated with initiating or processing a message, it is better to make messages fewer in number, just the opposite.

The left chart of figure 5 compares the different decompositions for our sample application in the Cray T3E. As the number of processors grows, the efficient exploitation of the underlying network becomes important. In the 32-processor simulation an appropriate 2D decomposition solves the trade-off between network exploitation and local memory access. In the larger problem the best 2D decomposition is 5% and 15% better than the 3D and linear decompositions, respectively. In any case, as we have explained before, a better algorithm for building Cartesian topologies could modify the results and its influence will be analysed in a future study. In the SGI Origin 2000 (right chart on figure 9) we have measured similar results, for the 32-processor simulation, the 2D decomposition is 22 % and 8% better than the 1D and 3D decompositions, respectively.

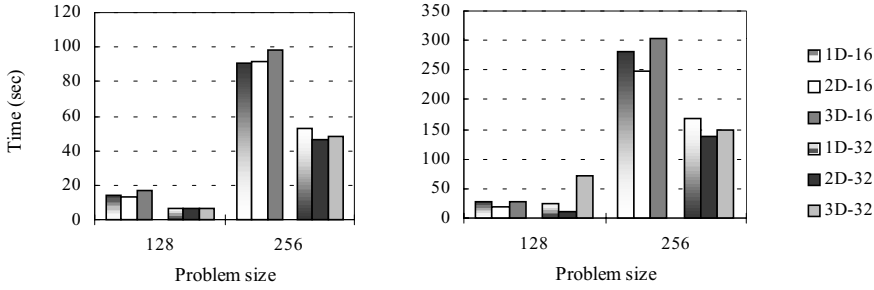

Fig. 5. Different decompositions for our sample program using 16 and 32 processors on the T3E (on the left) and on the SGI Origin 2000 (on the right). The problem size is the number of cells (2 double precision data each) in each dimension.

6. Conclusions

We have first shown how non-unit-stride memory access and network loads affect actual communication bandwidths on two different parallel computers, the Cray T3E and the Origin 2000. In both systems, the bandwidth reduction due to non-unit-stride local memory access is more important than the reduction due to contention in the network. The message passing performance is better in the T3E where a cache coherence protocol is not employed and where it is possible to bypass the second level cache for non-contiguous memory accesses. Secondly, we studied how these architecture properties affect he partitioning of regular domains. Obviously, data partitioning is a trade-off between the improvement of the message data locality and the efficient exploitation of the underlying communication system. Using up to 32 processors, in both systems, an appropriate 2D decomposition, where boundaries with poor spatial locality are not needed, solves that trade-off. The influence of spatial locality is lower in the SGI Origin where the large second level cache allows the best exploitation of the temporal locality. However, we should also note that there are other aspects to the evaluation of a parallel program. A lower-dimensional partitioning program is easier to code, so if we consider implementation cost, 1D

partitioning is the best choice. In Addition, it allows the implementation of fast sequential algorithms in the non-partitioned directions as we have explained in [16].

Acknowledgements
This work has been supported by the Spanish research grants TIC 96-1071 and TIC IN96-0510 and the Human Mobility Network CHRX-CT94-0459. We would like to thank Ciemat and CSC (Centro de Supercomputacion Complutese) for providing access to the parallel computers that have been used in this research.

References

[1] J. J. Dongarra, Tony Hey, and Erich Strohmaier. *"Selected Results from the PARKBENCH Benchmark"*, in Proceeding of EuroPar'96 Parallel Processing, Volume II, pages 251--254, August 1996.
[2] Aad J. van der Steen and Ruud van der Pas *"A performance analysis of the SGI Origin 2000",* in Proceedings of VECPAR 98, pp. 319-332. Porto, Portugal, 1998.
[3] Michael Resch, Holger Berger, Rolf Rabenseifner, Tomas Bönish "*Performance of MPI on the CRAY T3E-512"*, Third European CRAY-SGI MPP Workshop, PARIS (France), Sept. 11 and 12, 1997.
[4] Vladimir Getov, E. Hernández and T. Hey, *"Message-Passing Performance of Parallel Computers"*, in Proceeding of Europar'97, pp. 1009--1016. Passau, Germany, August 1997.
[5] OCCOMM home page: http://www.dl.ac.uk/TCSC/CompEng/OCCOMM.
[6] Shubhendu S. Mukherjee and Mark D. Hill. "*Making Network Interfaces less peripheral*", pp 70-76 Computer. October 1998
[7] S. L. Scott. *"Synchronization and Communication in the T3E Multiprocessor"*, in Proceeding of the ASPLOS VII, October 1996.
[8] E. Anderson, J. Brooks, C.Grass, S. Scott. *"Performance of the CRAY T3E Multiprocessor"*. in Proceeding of SC97, November 1997.
[9] David Culler, Jaswinder Pal Singh, Annop Gupta. *"Parallel Computer Architecture. A hardware /software approach"* Morgan-Kaufmann Publishers 1998.
[10] M. Prieto, I. M. Llorente, F. Tirado. *"Partitioning of Regular Domains on Modern Parallel Computers"*, in Proceedings of VECPAR 98, pp. 305-318. Porto, Portugal, 1998.
[11] Cray C/C++ Reference Manual, SR-2179 3.0.
[12] J. Laudon and D. Lenoski. *"The SGI Origin: A ccNUMA Highly Scalable Server"*, in Proceeding of ISCA'97.May 1997.
[13] H. J. Wassermann, O. M. Lubeck, F. Bassetti. "Performance Evaluation of the SGI Origin 2000: A Memory-Centric Characterization of LANL ASCI Applications". In Proceeding of the SC97, November 1997.
[14] Silicon Graphics Inc. "*Origin Servers"*, Technical Report, April 1997.
[15] Matthias Müller and Michael M. Resch , *"PE mapping and the congestion problem on the T3E"* in HErmann Lederer and Friedrich Hertweck (Ed.), Proceedings of the Fourth European Cray-SGI MPP Workshop, IPP R/46, Garching/Germany, 1998.
[16] D. Espadas, M. Prieto, I. Martín y F. Tirado "*Parallel Resolution of Alternating-Line Processes by means of Pipelining Techniques",* in Proceeding of the 7TH Euromicro Workshop on Parallel and Distributed Processing, pp. 289-296. Madeira, Portugal, 1999.