



On the development of a communication-aware task mapping technique [☆]

Juan Manuel Orduña ^{a,*}, Federico Silla ^b, José Duato ^b

^a *Depto. de Informática, Universidad de Valencia, Av. Vicent Andrés Estellés, 46100 Burjassot (Valencia), Spain*

^b *DISCA, Universidad Politécnica de Valencia, Spain*

Received 28 May 2002; received in revised form 17 July 2003; accepted 18 September 2003

Abstract

Clusters have become a very cost-effective platform for high-performance computing. In these systems, although currently existing networks actually provide enough bandwidth for the existing applications and workstations, the trend is towards the interconnection network becoming the system bottleneck. Therefore, in the future, scheduling strategies will have to take into account the communication requirements of the applications and the communication bandwidth that the network can offer. One of the key issues in these strategies is the task mapping technique used when the network becomes the system bottleneck.

In this paper, we propose a communication-aware mapping technique that tries to match as well as possible the existing network resources to the communication requirements of the applications running on the system. Also, we evaluate the mapping technique using real MPI application traces with timestamps. Evaluation results show that the use of the proposed mapping technique better exploits the available network bandwidth, improving load balancing and increasing the throughput that can be delivered by the network. Therefore, the proposed technique can be used in the design of communication-aware scheduling strategies for those situations where the communication requirements lead the network bandwidth to become the system performance bottleneck.

© 2003 Elsevier B.V. All rights reserved.

Keywords: Interconnection networks; Cluster computing; Task scheduling

1. Introduction

Cluster computing has become nowadays a very cost-effective alternative for high-performance computing. Using commodity computers and interconnecting them through a high performance network (like Myrinet [5], Gigabit Ethernet [1], etc.) it results relatively cheap and easy to construct a supercomputer formed by hundreds or even thousands of workstations. Additionally,

[☆] Supported by the Spanish MCYT under Grant TIC2003-08154-C06-04.

* Corresponding author. Tel.: +34-96-3160424; fax: +34-96-3160418.

E-mail addresses: juan.orduna@uv.es (J.M. Orduña), jduato@gap.upv.es (J. Duato).

these systems have incremental expansion capabilities. When those clusters of workstations are expanded by adding more processors, they usually become heterogeneous.

In order to fully exploit the computing power of heterogeneous systems, a lot of research has focused on solving the *NP-complete* problem of efficiently scheduling diverse groups of tasks to the machines that form the system [7,10,14–17,22,23]. Nevertheless, these proposals only focus on computational aspects. They try to match as well as possible the existing computing resources to the computational power required by the applications running on the system, but they do not consider communication aspects at all, thus assuming that the communication subsystem provides enough bandwidth in any case. Although some proposals take into account communication requirements [13], these proposals are focused to multiprocessor architectures different from clusters.

Currently existing network technologies actually provide enough bandwidth for the existing applications and systems nowadays. However, the trend is towards the interconnection network becoming the system bottleneck. This trend is due to several factors:

- The increasing size of the clusters of workstations. For example, the Computational Plant, at Sandia National Laboratories (USA), is formed by 1369 commodity personal computers (see <http://www.cs.sandia.gov/cplant>). Another more recent example of this trend is the Tera-scale Computing System installed at Pittsburgh Supercomputing Center (USA). This system is formed by 3024 Compaq Alphaserwer processors running at 1 GHz (see <http://www.psc.edu>). If the network bandwidth does not linearly scale up with the network size, then this trend of increasing the cluster size can lead to network saturation.
- Uniprocessors are being replaced with multiprocessors (SMPs) as computing nodes. In this case, several CPU's are connected to the network subsystem through a single network interface, thus increasing the network bandwidth required for each computing node.

- The increasing use of grid computing. The use of the Internet resources as (part of) the interconnection network makes this subsystem to significantly reduce its average bandwidth, while the processor speed and memory bandwidth remain unchanged.
- Nowadays processor speed increases more rapidly than network speed, therefore increasing the network bandwidth required for each processor in the cluster.

These reasons suggest that the interconnection network will become the system bottleneck in the near future. Although the interconnection network has not reached this point yet, the current trend recommends the study of communication-aware scheduling techniques, in such a way that when this situation is reached, these techniques have already been developed and included in the schedulers.

Therefore, given a heterogeneous system (that may be formed by different groups of interconnected homogeneous systems) and given a certain set of different (parallel or sequential) applications from different users, an ideal scheduling strategy would map the processes to processors taking into account both the computing and the communication requirements of the applications running on the machine. The scheduler would choose either a computation-aware or a communication-aware task scheduling strategy depending on the kind of requirements that leads to the system performance bottleneck.

In order to develop a communication-aware task scheduling strategy for parallel applications on heterogeneous systems, several problems must be solved:

- The communication requirements of the applications running on the machine must be measured or estimated.
- The available network resources must also be characterized.
- Some criterion is needed to measure the suitability of each allocation of network resources to each parallel application, according to its communication requirements.

- Based on the previous criterion, some mapping technique based exclusively on the communication requirements should be developed.
- Finally, this communication-based mapping technique must be integrated with process scheduling, in order to be used when the communication requirements are the ones that lead to the system performance bottleneck.

In previous works, we addressed some of these problems. We proposed a clustering method to experimentally evaluate the communication requirements of message-passing parallel applications [19]. Also, we proposed a model of communication cost that provides a characterization of the network resources of any given irregular topology [2]. Additionally, we proposed a clustering method to provide a network partition adapted to the communication requirements of the applications running on the machine [3], a criterion to measure the suitability of each allocation of network resources to each of the parallel applications [3,18], and a mapping technique based exclusively on the communication requirements [18]. However, in that work we made some simplified assumptions, in order to quickly analyze the behavior of the network. The results showed that the network performance could actually be greatly improved if a communication-based mapping technique was used.

This paper presents, in an unified manner, an enhanced communication-based mapping technique that directly provides a mapping of processes to processors [20]. Unlike the previous version presented in [18], this new mapping technique takes into account not only the existing network resources, but also the traffic generated by the applications. In particular, the new mapping technique uses the information provided by the characterization of the applications proposed in [19]. We also present the evaluation of the proposed technique using real MPI application traces with timestamps. For the sake of simplicity of evaluation, we have considered the problem under simplified assumptions (all the network switches are attached to the same number of workstations, all the workstations are uniprocessors, and only

one process is mapped to each processor). However, these assumptions can be easily extended to more realistic conditions, as explained below. Evaluation results show that the proposed approach is able to exploit the available network bandwidth, significantly improving the throughput that can be delivered by the network. Therefore, this communication-aware mapping strategy can be used by the scheduler for those situations where the communication requirements are the system performance bottleneck.

In order to make this paper self-contained, we have summarized the development of all the previous steps required for the proposed mapping technique. Therefore, the rest of the paper is organized as follows: Section 2 shows the technique used to measure or estimate the communication requirements of the applications running on the machine. Section 3 shows the proposed technique for identifying the existing network resources in the system. The definition of the quality function to be used by the mapping technique, as well as the proposed mapping technique, are presented in Section 4. Section 5 describes the evaluation methodology used to measure network performance and the evaluation results obtained with the proposed method. Finally, Section 6 presents some concluding remarks and future work to be done.

2. Communication requirements

The first step in the development of a mapping technique that matches the communication requirements of the applications to the communication bandwidth available in different parts of the network is to measure or estimate these communication requirements. Obviously, measurement should not be done by completely executing the application, because no mapping is required once the applications have been executed completely. Nevertheless, there exists a wide variety of applications whose traffic pattern can be estimated simply by executing only a small part of those applications, since their traffic pattern remains unchanged during the rest of the execution. Additionally, some applications are regularly

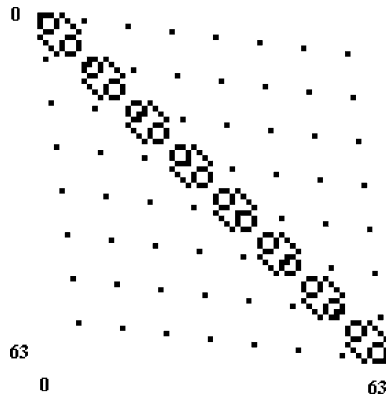


Fig. 1. Table of communication between processes obtained for CG benchmark.

executed and show similar communication requirements for different executions performed with different input data [9]. Some examples of such kind of application could be some weather forecast applications, satellite image processing applications, simulation tools for design processes, seismic data processing applications, etc.

We can apply the characterization method proposed in [19] for such kind of applications. This method simply consists of measuring the amount of information that each pair of processes exchange during (part of) the execution of the application, obtaining a *table of communication between processes*. As its name suggests, it provides the amount of packets¹ exchanged between each pair of processes. If there exist N processes, then this table will consist of $N \times N$ elements. Each element (i, j) in this table will be denoted as c_{ij} . This value represents the amount of packets that process i sends to process j . As an example, Figs. 1 and 2 present a graphical view of the tables of communication between processes obtained for two different NAS parallel benchmarks [4]. These benchmarks have been executed using the MPICH portable implementation of the MPI message-passing standard [11], and they have been configured to be executed with $N = 64$ processes. In

¹ We assume that long messages are split into a set of fixed size packets, as it is currently done by most communication libraries.

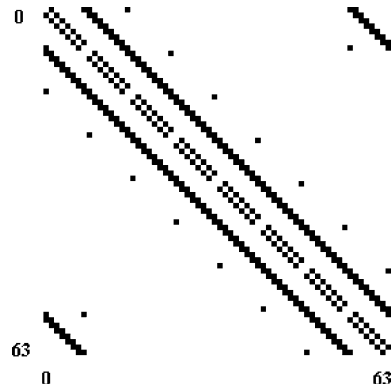


Fig. 2. Table of communication between processes obtained for SP benchmark.

these figures, we have coded each table of communications between processes as a pixel matrix: black pixels represent the elements of the table with nonzero values, while zero values are represented by white pixels.

As Figs. 1 and 2 show, there is some kind of communication pattern around the diagonal of these tables. Also, there are a few nonzero values far from the diagonal. However, from the visual analysis of the tables, it is very difficult to identify which are the groups of processes that have the higher communication requirements for these applications. The characterization method proposed in [19] uses a clustering technique for grouping the processes into clusters, based on the information shown in the table of communication between processes. However, in order to use the most accurate information available, the mapping technique proposed in this paper does not perform any clustering. Instead, the mapping technique proposed in this paper uses the table of communication between processes itself as the communication requirements of the application.

3. Network resources

The second step in matching the communication requirements of the applications to the communication bandwidth available in different parts of the network is to identify the existing network resources. In order to perform this step, we pro-

posed a model of communication cost [2] that can be applied to any switch-based network. This model is named the *table of equivalent distances*, and it computes the cost for communicating each pair of network switches without explicitly considering traffic pattern. The model assigns the unit cost to every single link in the network. The equivalent distance for a pair of nodes is computed taking into account all the shortest paths between them supplied by the routing algorithm. The name of the metric is derived from the analogy to the electrical equivalent resistance. Indeed, we use the same rules as for electrical circuits to compute the total communication cost between nodes, applying Kirchoff's laws.

In particular, the method used to compute the equivalent distance between each pair of nodes in the network, taking into account only the network topology and the routing function, is the following:

- (i) If there exists only one shortest path between a given pair of nodes then the communication cost between those nodes will be the sum of the costs of the links that form the path. That is, this case is similar to computing the equivalent resistance of an electrical circuit consisting of serially arranged resistors. Since we have assumed that all the links in the network have unit cost, the communication cost is equal to the number of links in the path.
- (ii) If there exists more than one shortest path supplied by the routing algorithm between a given pair of nodes then the communication cost between them is computed similarly to the electrical equivalent resistance between two points of an electrical circuit, replacing each link in a shortest path with a unit resistor and applying Kirchoff's laws. Note that we only consider the shortest paths provided by the routing algorithm. Those paths are not necessarily minimal (as is the case for up/down routing). Also, the paths not supplied by the routing algorithm are not considered.

Therefore, if the network is composed of S switches, then the table of distances between network switches will contain $S \times S$ elements. We will denote each element (i, j) in this table as s_{ij} . This

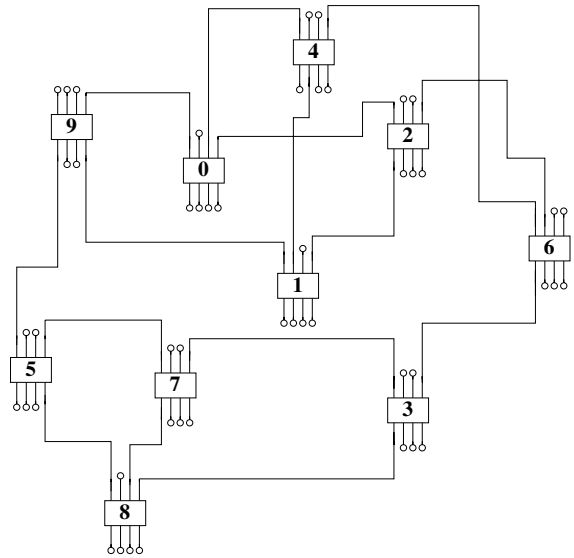


Fig. 3. Autonet network with 10 switches.

value represents the cost for sending a message from switch i to switch j , and it is inversely related to the existing network bandwidth in the path from switch i to switch j .

As an example of this method, consider the Autonet network shown in Fig. 3. The Autonet routing algorithm is distributed, and implemented using table-lookup [21]. In order to fill the routing tables, the interconnection network can be modeled as a multigraph $I = G(N, C)$, where N is the set of switches, and C is the set of bidirectional links between the switches. Fig. 4 shows the graph for the network in Fig. 3.

In order to fill the routing tables, a breadth-first spanning tree (BFS) is computed first, using a distributed algorithm. Based on this spanning tree, the link direction assignment is computed for graph I . The “up” end of a link is connected to the upper level node when the link connects nodes located at different tree levels. When communicating nodes at the same tree level, the “up” end of a link is connected to the node with the lower label. The result of this assignment is that each cycle in the network has at least one link in the “up” direction and one link in the “down” direction. Fig. 5 shows the link direction assignment for the graph in Fig. 4. In this figure, switches are arranged in such a way that all the switches at the

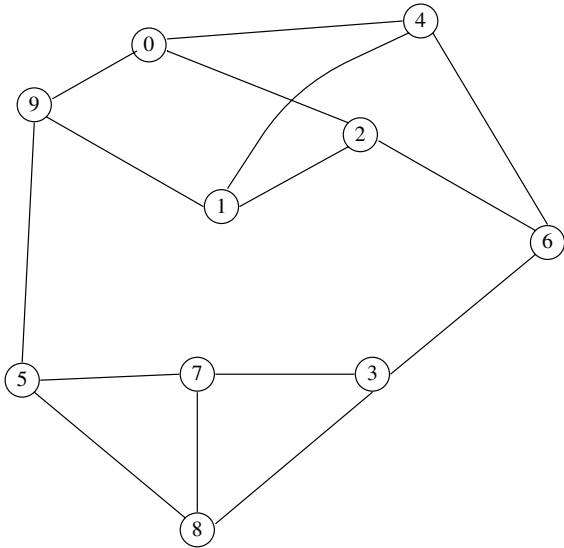


Fig. 4. Graph G for the 10-node Autonet network.

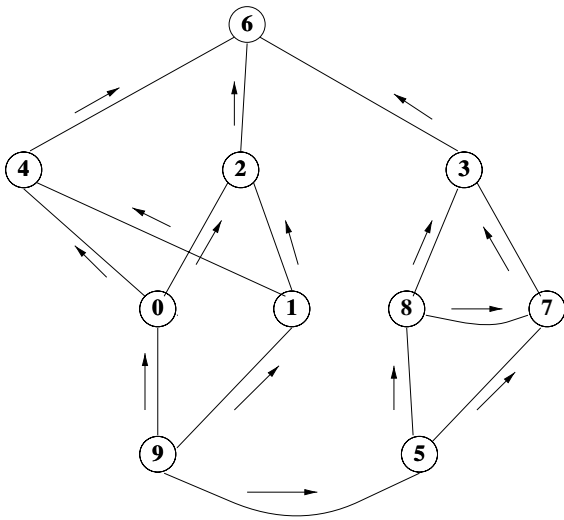


Fig. 5. Link direction assignment for the graph in Fig. 4.

same level in the spanning tree are at the same vertical position in the figure. The up/down routing scheme establishes that a legal route must traverse zero or more links in the “up” direction, followed by zero or more links in the “down” direction. A message cannot traverse a link along the “up” direction after having traversed a link in the

Table 1

Possible paths for going from node 0 to node 1

0-2-6-4-1
0-2-1
0-4-6-2-1
0-4-1

“down” direction. Although such a routing scheme is deadlock-free and allows some adaptivity, in some cases up/down routing is not able to supply any minimal path between two nodes.

Let us consider the equivalent distance between nodes 0 and 1. Taking into account the link direction assignment in Fig. 5 and considering up/down routing, all possible paths for going from node 0 to node 1 are shown in Table 1, existing two shortest paths. Therefore, in order to compute the equivalent distance between nodes 0 and 1, the source and destination nodes must be considered as the V_{cc} and GND points of an electric circuit. Nodes 2 and 4 must be considered as two different intermediate points, and each link must be considered as a resistor with unit resistance, resulting in the circuit shown in Fig. 6. Applying Kirchoff’s laws to this circuit, an equivalent resistance of 1 Ohm is obtained. Thus, the equivalent distance from node 0 to node 1 is set to 1. The rest of equivalent distances in the table of distances are computed in the same way.

Following this method, a table of equivalent distances can be obtained from a given topology and a given routing algorithm. As an example, Table 2 shows the table of equivalent distances for the network shown in Fig. 3 when this method is applied.

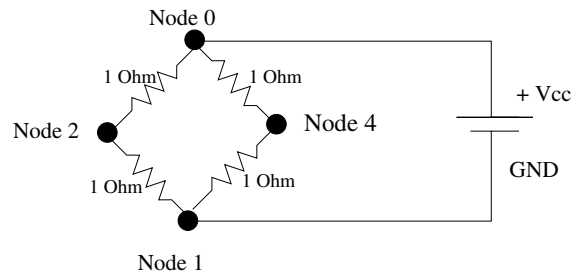


Fig. 6. Model of equivalent resistance between node 0 and node 1.

Table 2
Table of equivalent distances between nodes

Nodes	N0	N1	N2	N3	N4	N5	N6	N7	N8	N9
N0	0	1	1	2	1	3	1	3	3	1
N1	1	0	1	2	1	3	1	3	3	1
N2	1	1	0	2	2	3	1	3	3	1
N3	2	2	2	0	2	1	1	1	1	2
N4	1	1	2	2	0	3	1	3	3	1
N5	3	3	3	1	3	0	2	1	1	1
N6	1	1	1	1	1	2	0	2	2	1.25
N7	3	3	3	1	3	1	2	0	1	2
N8	3	3	3	1	3	1	2	1	0	2
N9	1	1	1	2	1	1	1.25	2	2	0

4. Mapping technique

The two models explained in the two sections above are the input data to be used by the mapping technique. However, prior to the application of any mapping technique, the table of distances between network switches must be expanded to form the *table of distances between processors*. Effectively, as processes are mapped on the processors existing in the system, the network distances should be computed between the processors existing in the system, instead of the network switches.

The table of distances between processors is computed taking into account how the workstations are attached to each of the network switches, and also taking into account if a given workstation is a uniprocessor or a multiprocessor workstation. Since the model of communication cost proposed in [2] assigns a unit cost to each network link, the distance between a given pair of processors is computed as the corresponding distance between the switches they are attached to (this value is given by the table of distances between network switches) plus two (for the cost of the links attaching each processor to its network switch). The distance between a given processor and itself is zero, since in this case the network is not reached. The table of distances between processors will consist of $P \times P$ elements. We will denote each element (i, j) in this table as d_{ij} . This value represents the cost for sending a message from processor i to processor j . In the case of SMPs, the

distance between a processor in a SMP workstation and a processor belonging to another workstation will be computed as the distance between the corresponding switches plus two. However, the distance between a processor in a SMP workstation and another processor in the same workstation will also be zero, since in this case the network is not reached, either.

The table of distances between processors and the table of communication between processes contain the required information to perform a communication-aware mapping of processes to processors. Therefore, the idea proposed in [20] is to perform a heuristic search based on these two tables, in order to find the best mapping of processes to processors.

We will represent a given mapping of processes to processors as a vector of N elements. We will denote each element i of this mapping vector as m_i . This value means that process i is assigned to processor m_i . That is, the order of a given element within the mapping vector will represent the process, and the value of that element will represent the processor that this process is mapped to. Thus, for example, if the third element of this mapping vector contains the value 5, this will mean that process 3 is assigned to processor 5.

4.1. Quality function

Any heuristic search method must find a solution within a space of solutions Ω . In our case, this space of solutions consists of all the possible

mappings of processes to processors for the existing sets of processors and processes. This space is associated with a target function F that assigns a cost to each particular solution (mapping of processes to processors) $P \in \Omega$. The heuristic search method must find a particular solution P_0 such that

$$F(P_0) \leq F(P) \quad \forall P \in \Omega$$

In this case, the target function F must take into account the information provided by both the table of distances between processors and the table of communication between processes. Therefore, we have used as the target function to be minimized the *mapping coefficient* M_c . This coefficient is defined as

$$M_c = \sum_{i=1}^N \sum_{j=1}^N c_{ij} \cdot d_{m_i m_j}$$

M_c represents the sum of all the messages exchanged between the existing processes, weighted by their corresponding cost of sending these messages between the corresponding processors according to the mapping vector. If the heuristic search minimizes this function, then the overall cost for transmitting the application messages will be minimized. The purpose of the proposed mapping technique is to search the best mapping vector for a given network topology and for a given communication pattern. Since the communication cost is defined as inversely proportional to network bandwidth [2], we are actually mapping the processes that communicate more frequently to processors with the higher network bandwidth between them. By doing so, we use network resources more efficiently and delay network saturation as much as possible.

4.2. Heuristic search

We have tried several of the heuristic search methods proposed in [6]. However, we have obtained the best results when using a random search method. This heuristic search method provided the same or better mapping coefficients with lower computational cost than other methods. This method starts with a random mapping vector.

Each iteration consists of exchanging two randomly selected values of the mapping vector. If the resulting mapping vector shows a better mapping coefficient, then this permutation is saved. If not, then it is discarded. The stop condition for the algorithm is to perform a given number of consecutive permutations without decreasing the resulting mapping coefficient. At this point, the current minimum mapping coefficient and its corresponding mapping vector are saved, and another seed (random mapping vector) is tried. The algorithm stops when a number of different seeds has been explored. In particular, we have computed this search method for 10 seeds (initial random mappings). For each of these seeds, the search has been performed until 40 consecutive iterations have been computed without decreasing M_c .

Fig. 7 shows the values of M_c reached in the search performed for matching a 16-switch network and the traffic generated by the NAS SP benchmark configured for 64 processes. In this figure, the total iteration count is shown in the X-axis. The values of M_c at the 10 different starting points of the search form the peak values in the figure. It can be seen that the value for M_c rapidly decreases in the first few iterations after a starting point. In this example, the global minimum value for the mapping coefficient is reached for three different seeds.

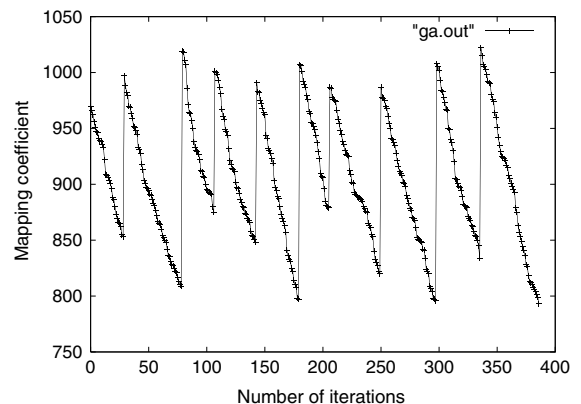


Fig. 7. Heuristic search for a 16-switch network and SP benchmark configured for 64 processes.

5. Performance evaluation

In this section we are going to study the improvement in network performance that the proposed mapping technique can provide. For this study, we have used several different randomly generated network topologies, executing several benchmarks on them. We have evaluated these networks with random mappings of processes to processors and also with the mapping provided by the proposed technique. We did not compare with other previously proposed mapping techniques since none of them considers the communication cost.

5.1. Evaluation setup

We have executed several parallel benchmarks on a simulated network of workstations using the MPI message-passing standard. Each parallel benchmark creates N processes that communicate between them by passing messages, where N is an input parameter set by the user. We have used the MPICH portable implementation of the MPI message-passing standard [11]. This implementation allows us to simulate several machines of a given architecture in a single machine [12], and it can be set to provide execution traces of each MPI call. Additionally, we have modified this implementation in order to add timestamps to each MPI call trace. Thus, using the modified libraries together with the Network Time Protocol daemon supplied in the Linux distribution, we have obtained execution traces with timestamps. We have executed the CG, EP, IS, LU, MG, and SP NAS Parallel Benchmarks 2.0 [4]. While SP benchmark requires N to be a square number, the rest of the benchmarks require N to be a power of 2. Each benchmark has been executed with 64 processes.

The execution traces of the benchmarks show that most of the communication between processes is performed through MPI point-to-point communication calls. Only some benchmark traces contain a few MPI collective communication calls, that in no case reach 0.5% of the total MPI communication calls. Therefore, we have only considered point-to-point communications.

On the other hand, the network is composed of a set of switches. In order to model real systems as closely as possible, we assume that the network topology is irregular. For the studies presented in this paper, the topology has been generated randomly. However, for the sake of simplicity we imposed three restrictions. First, we assumed that there are exactly four workstations connected to each switch. Second, two neighboring switches are connected by a single link. Finally, all the switches in the network have the same size. We assumed 8-port switches. Therefore, each switch has four ports available to connect to other switches. From these four ports, three of them are used in each switch when the topology is generated. The remaining port is left open. We have evaluated several different networks with a size of 16 switches (64 workstations), assuming that each workstation hosts only one of the processes generated by the application.

We have evaluated the performance of several irregular networks by simulation. The evaluation methodology used is based on the one proposed in [8]. The most important performance measures are latency and throughput. The message latency lasts from when the message is generated at the source node until it is completely received at the destination node. Throughput is the maximum amount of information delivered per time unit (maximum traffic accepted by the network). Latency is measured in clock cycles. Traffic is measured in flits per switch per cycle. A flit (or flow control unit) is the unit of information transmitted through a network link. Our simulator models the network at the flit level, and we have modified it in order to accept the execution traces with timestamps as the traffic source.

For the sake of simplicity, we have assumed in this evaluation that all the workstations are uniprocessors, and that only one process is mapped to each processor. However, note that these assumptions are not due to any intrinsic limitation of the proposed scheduling technique. Effectively, M_c is defined in such a way that it is not affected neither by the number of processors per node nor the number of processes mapped to each processor. These parameters will affect the size of the table of distances between processors and how the

mapping vector is constructed, respectively. However, they do not affect how the quality function is computed. For example, if more than one process is mapped to each processor, then the mapping vector will contain repeated values. However, the mapping coefficient will always be computed in the same way.

In order to analyze how effective the proposed strategy is, we assume that the network bandwidth is the system performance bottleneck, that is, the communication requirements of the applications (the execution traces with timestamps) lead the network to saturation. Although this will likely be the situation in the near future (as discussed in Section 1), it is not the case in current clusters. Therefore, in order for the execution traces to produce network saturation, the actual timestamps in the execution traces must be proportionally reduced. Thus, the timestamps of the execution traces have been decreased until network saturation is reached when the network simulator is fed with them.

5.2. Evaluation results

In order to show that the proposed mapping technique can increase network performance, we have evaluated each network with several distinct mappings of processes to processors. For each considered benchmark and each considered network topology, we have computed several random mappings of processes to processors. However, for the sake of clearness, we have included in the figures only the plot corresponding to the best network performance obtained with any of the random mappings. We have denoted this plot as random (r). Also, taking into account that any communication-aware mapping would map the processes with the highest communication requirements as closely as possible, we have computed several partially random mappings that assign the processes with the highest communication requirements to workstations attached to the same network switch. However, in these mappings the switch is selected randomly. We have denoted these mappings as random mappings with locality (rl). We have compared the network performance obtained with all of these mappings with the net-

work performance obtained with the mapping provided by the proposed approach, denoted as near optimal (no). We have not included for comparison purposes any mapping provided by the previous version of the mapping technique, since the previous version does not take into account the traffic generated by the applications, always assuming a regular traffic pattern. Although we have performed the performance evaluation using almost all the NAS parallel benchmarks, for the sake of shortness we only present in this section the evaluation results for two of the MPI benchmarks and for two different random network topologies. These evaluation results summarize the behavior of all the evaluated benchmarks and network topologies, as discussed at the end of this section.

Fig. 8 shows the evolution of traffic over time obtained with different mappings for the CG benchmark when executed on a 16-switch network, denoted as “A” network. In this figure, the time is shown in the X-axis, expressed as thousands of clock cycles. The traffic delivered by the network is shown in the Y-axis. Each point in a plot means that 10,000 messages have reached their destinations. The Y coordinate of that point represents the traffic rate corresponding to these messages. It can be seen that the network throughput achieved with the mapping provided by the proposed approach is at least a 60% higher than the one achieved with any of the random

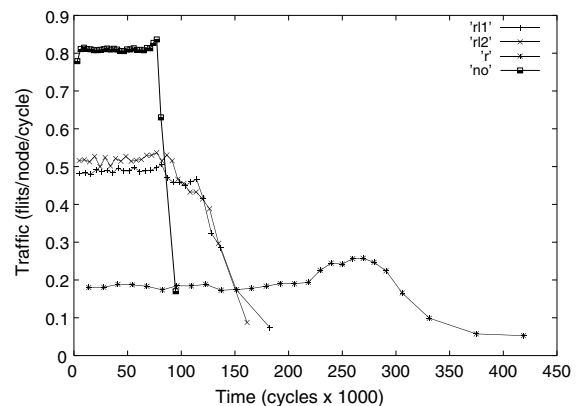


Fig. 8. Network traffic for a 16-switch “A” network and CG benchmark.

mappings with locality, and is up to 400% higher than the one achieved with any of the random mappings. As a consequence, messages are delivered faster, thus reducing overall execution time. The reason for this behavior is that a better mapping reduces contention in the network as well as the average distance traveled by messages. Therefore, network resources are better used, thus increasing throughput. The latencies obtained for this network topology show that the network is working beyond the saturation point all the time. These results show that when the network becomes the system bottleneck, then the proposed mapping technique is crucial to maximize performance.

Fig. 9 shows the latencies obtained for the SP benchmark when executed using the same network topology. In this case, the time scale used and the traffic generated by the benchmark lead to time intervals with deep network saturation interleaved with time intervals of low network load. It can be clearly seen that during the deep saturation intervals the latencies achieved by the mapping provided by the proposed approach are much lower than the latencies achieved by any other mapping. Fig. 10 shows the network traffic obtained with different mappings for the same benchmark and network topology. Since for this benchmark there are some time intervals where the network does not reach saturation, this figure does not show significant differences between the different plots. However, the mapping provided by the proposed

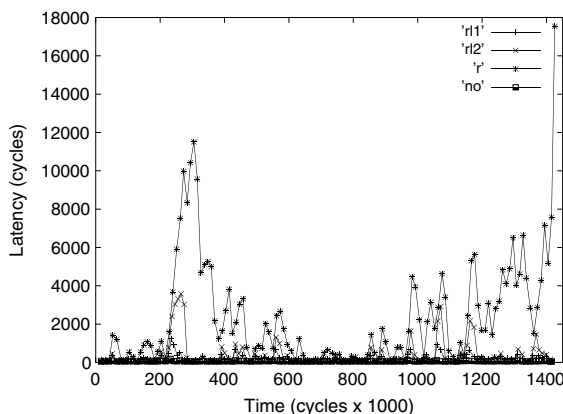


Fig. 9. Latencies for a 16-switch “A” network and SP benchmark.

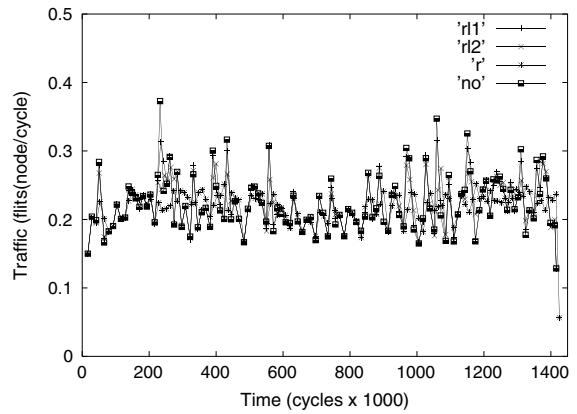


Fig. 10. Network traffic for a 16-switch “A” network and SP benchmark.

approach results in the best network throughput during the intervals with traffic peaks.

Fig. 11 shows the network throughput achieved by different mappings for the CG benchmark and for a different 16-switch network topology, denoted as “B” network. Again, the network throughput achieved with the mapping provided by the proposed approach is higher than the one achieved with any of the random mappings. Also, the latencies obtained with the different mappings show that the network is under saturation all the time.

Figs. 12 and 13 show the network latencies and network throughput, respectively, achieved by

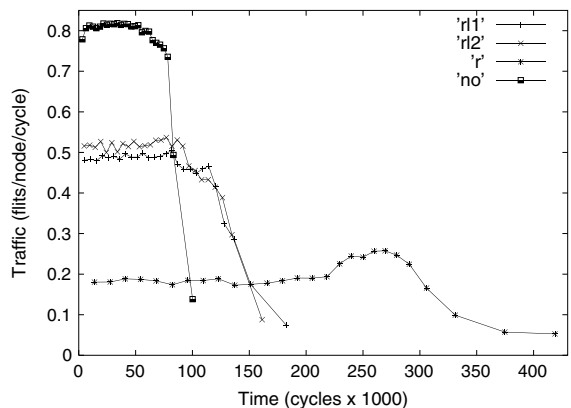


Fig. 11. Network traffic for a 16-switch “B” network and CG benchmark.

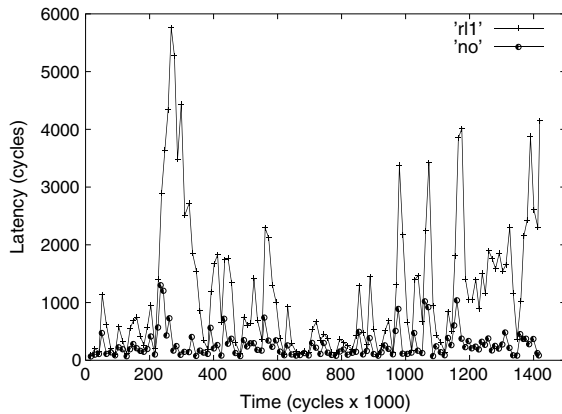


Fig. 12. Latencies for a 16-switch B network and SP benchmark.

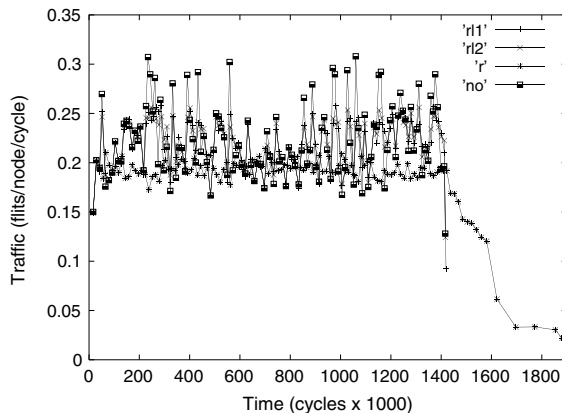


Fig. 13. Network traffic for a 16-switch B network and SP benchmark.

different mappings for the SP benchmark and the “B” network topology. The random mappings lead to network latencies that are several orders of magnitude higher than the ones for the mapping provided by the proposed approach. Therefore, for the sake of clearness, we have only included in Fig. 12 the plot for the mapping provided by the proposed approach and the plot for the random mapping with locality that produces the lowest latency. This figure shows that, during the time intervals of network saturation, the mapping provided by the proposed approach achieves much lower latencies. Fig. 13 shows that, again, the highest network throughput during traffic peaks is

achieved by the mapping provided by the proposed approach.

Additionally, it appears to be better taking into account the message size, rather than taking into account the number of fixed-size packets generated by each message. Therefore, for each benchmark we have computed again each element in the table of communication between processes, weighting each message by its own length. We have repeated the search process using the new tables, and we have evaluated again the same networks using each message length provided by the MPI traces. However, the simulation results obtained in this way do not vary from the simulation results shown above.

Although they are not shown here for the sake of shortness, the results obtained for other benchmarks are very similar. For those benchmarks where the network is under saturation all the time (as for the CG benchmark), the throughput achieved with the proposed mapping is significantly higher than for any other mapping. Also, for those benchmarks in which the network is saturated during some time intervals, the latency obtained with the mapping provided by the proposed technique is much lower than the one obtained with any of the random mappings, particularly for those time intervals where the network is under saturation. It should be noted that higher message latencies usually lead to longer process waiting time, and therefore, longer execution time. These results show that the proposed approach can provide a mapping of the application processes to the processors in the system such that it better exploits the available network bandwidth.

6. Conclusions and future work

In this paper, we have presented, in a unified manner, a communication-based mapping technique that directly provides a mapping of processes to processors. This mapping technique takes into account not only the existing network resources, but also the traffic generated by the applications. Also, we have evaluated the proposed mapping technique using real MPI application traces with timestamps.

For those cases where the application and the network topology result in network saturation during the entire execution of the application, the proposed mapping technique significantly improves network throughput. For those cases where the system results in some time intervals of network saturation, the proposed mapping technique provides the lowest peak latencies for those intervals. These results show that the proposed mapping technique is able to better exploit the available network bandwidth in different parts of the network, mapping the processes to processors in such a way that the processes with higher communication requirements are mapped on the processors belonging to the network areas with higher bandwidth.

As for future work, we plan to integrate the proposed strategy with mapping strategies that consider the computing requirements of the applications.

References

- [1] 10 Gigabit Ethernet Alliance, 10 Gigabit Ethernet Technology Overview White Paper, Available from: <<http://www.10gea.org/>> September, 2001.
- [2] V. Arnau, J.M. Orduña, A. Ruiz, J. Duato, On the characterization of interconnection networks with irregular topology: a new model of communication cost, in: XI IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS'99) (IASTED 1999) pp. 1–6.
- [3] V. Arnau, J.M. Orduña, S. Moreno, R. Valero, A. Ruiz, A clustering approach for improving network performance in heterogeneous systems, in: Lecture Notes in Computer Science, vol. 1900, Springer-Verlag, 2000, pp. 1206–1209.
- [4] D. Bailey, T. Harris, W. Saphir, R. vanderWijngaart, A. Woo, M. Yarrow, The NAS Parallel Benchmarks 2.0, Technical Report NAS-95-020. NAS Systems Division, NASA Ames Research Center, December 1995.
- [5] N.J. Boden, D. Cohen, R.E. Felderman, A.E. Kulawik, C.L. Seitz, J. Seizovic, W. Su, Myrinet—a gigabit per second local area network, IEEE Micro (February) (1995) 29–36.
- [6] T.D. Braun, H.J. Siegel, et al., A comparison study of static mapping heuristics for a class of meta-tasks on heterogeneous computing systems, in: Proceedings of 8th IEEE Heterogeneous Computing Workshop (HCW'99), pp. 15–29.
- [7] X. Du, X. Zhang, Coordinating parallel processes in networks of workstations, Journal of Parallel and Distributed Computing 46 (1997) 125–135.
- [8] J. Duato, A new theory of deadlock-free adaptive routing in wormhole networks, IEEE Transactions on Parallel and Distributed Systems 4 (1993) 1320–1331.
- [9] D.G. Feitelson, B. Nitzberg, Job characteristics of a production parallel scientific workload on the NASA Ames iPSC/860, in: Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science, vol. 949, Springer-Verlag, 1995.
- [10] R.F. Freund, M. Gherrity, S. Ambrosious, et al., Scheduling in multi-user, heterogeneous computing environments with SmartNet, in: Proceedings of 7th IEEE Heterogeneous Computing Workshop (HCW'98), pp. 184–199.
- [11] W. Gropp, E. Lusk, N. Doss, A. Skjellum, A high-performance, portable implementation of the MPI message passing interface standard, Parallel Computing 22 (1996) 789–828.
- [12] W. Gropp, E. Lusk, User's Guide for mpich, a Portable Implementation of MPI. Technical Report ANL-96/6. Mathematics and Computer Science Division, Argonne National Laboratory, 1996.
- [13] B. Gruber, G. Haring, D. Kranzlmüller, J. Volkert, Parallel programming with CAPSE—a case study, in: Proceedings of 4th EUROMICRO Workshop on Parallel and Distributed Processing (PDP), IEEE Computer Society Press, 1996, pp. 130–137.
- [14] P. Holenarsipur, V. Yarmolenko, J. Duato, D.K. Panda, P. Sadayappan, Characterization and enhancement of static mapping heuristics for heterogeneous systems, in: Proceedings of the 7th International Conference on High Performance Computing (HiPC 2000), Springer-Verlag, 2000, pp. 37–48.
- [15] M. Kafil, I. Ahmad, Optimal task assignment in heterogeneous distributed computing systems, IEEE Concurrency 6 (1998) 42–51.
- [16] B. Lowekamp, D. O'Hallaron, T. Gross, Direct network queries for discovering network resource properties in a distributed environment, in: Proceedings of the 8th IEEE Symposium on High-Performance Distributed Computing, 1999, pp. 38–46.
- [17] M. Maheswaran, S. Ali, H. Siegel, D. Hensgen, R. Freund, Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems, in: Proceedings of 8th IEEE Heterogeneous Computing Workshop (HCW'99), pp. 30–44.
- [18] J.M. Orduña, V. Arnau, A. Ruiz, R. Valero, J. Duato, On the design of communication-aware task scheduling strategies for heterogeneous systems, in: Proceedings of International Conference on Parallel Processing (ICPP-2000), pp. 391–398.
- [19] J.M. Orduña, V. Arnau, J. Duato, Characterization of communications between processes in message-passing applications, in: Proceedings of 1st IEEE International Conference on Cluster Computing (CLUSTER-2000), pp. 91–98.
- [20] J.M. Orduña, F. Silla, J. Duato, A new task mapping technique for communication-aware scheduling strategies,

in: Proceedings of International Conference on Parallel Processing Workshops (ICPP-2001), pp. 349–354.

- [21] M.D. Schroeder et al., Autonet: a high-speed, self-configuring local area network using point-to-point links, Technical Report SRC research report 59, DEC, April 1990.
- [22] H. Topcuoglu, S. Hariri, M.Y. Wu, Task scheduling algorithms for heterogenous processors, in: Proceedings of 8th IEEE Heterogeneous Computing Workshop (HCW'99), pp. 3–14.
- [23] V. Yarmolenko, J. Duato, D.K. Panda, P. Sadayappan, Characterization and enhancement of dynamic mapping heuristics for heterogeneous systems, in: Proceedings of the 2000 ICPP Workshop on Network-Based Computing, 2000, pp. 437–444.



Juan Manuel Orduña received the MS in computer engineering from the Technical University of Valencia, Spain, in 1990. He worked at Telefónica de España and Manpel Electrónica, S.A. as a computer engineer. He received the PhD in computer engineering from the University of Valencia in 1998. He is currently a lecturer professor at the Department of Informatics, University of Valencia, SPAIN. His research addresses cluster computing and distributed virtual environment systems.



Federico Silla received the MS and PhD degrees in computer engineering from the Technical University of Valencia, Spain, in 1995 and 1999, respectively. He is currently an associate professor at the Department of Computer Engineering at the Technical University of Valencia. His research addresses high performance interconnects.



José Duato received the MS and PhD degrees in electrical engineering from the Technical University of Valencia, Spain, in 1981 and 1985, respectively. Currently, Dr. Duato is Professor in the Department of Computer Engineering (DISCA) at the same university. He was also an adjunct professor in the Department of Computer and Information Science, The Ohio State University. His current research interests include interconnection networks, multiprocessor architectures, networks of workstations, and switch fabrics for

IP routers. Prof. Duato has published over 230 refereed papers. He proposed the first theory of deadlock-free adaptive routing for wormhole networks. Versions of this theory have been used in the design of the routing algorithms for the MIT Reliable Router, the Cray T3E supercomputer, the internal router of the Alpha 21364 microprocessor, and the BlueGene/L supercomputer. Duato is the first author of the book “Interconnection Networks: An Engineering Approach”. This book was co-authored by Prof. Sudhakar Yalamanchili, from Georgia Institute of Technology, and Prof. Lionel Ni, from Michigan State University. It is currently the reference book on interconnection networks. Duato served as a member of the editorial boards of IEEE Transactions on Parallel and Distributed Systems and IEEE Transactions on Computers. He has been the General Co-Chair for the 2001 International Conference on Parallel Processing and is the Program Committee Chair for the Tenth International Symposium on High Performance Computer Architecture (HPCA-10). Also, he served as Co-Chair, member of the Steering Committee, Vice-Chair, or member of the Program Committee in more than 30 conferences, including the most prestigious conferences in his area (HPCA, ISCA, IPPS/SPDP, ICPP, ICDCS, Europar, HiPC).