Effective Load Balancing on a LAN-WLAN Cluster *

David Sánchez, Elsa Mª Macías and Álvaro Suárez
Grupo de Arquitectura y Concurrencia (GAC)
Departamento de Ingeniería Telemática (U.L.P.G.C.)
Campus Universitario de Tafira, 35017-Las Palmas de Gran Canaria, Spain
e-mail: {david, elsa, alvaro}@dit.ulpgc.es

Abstract

Fixed networks of computers constitute the lowest cost as well as the most available parallel computer, however the proliferation of wireless devices allows to expand the parallel virtual machine. These clusters are composite by resources with different performances and also are connected by communication links with different bandwidth, such as Fast-Ethernet, Standard Ethernet and Standard IEEE 802.11. Execution of a parallel application in these environments without keeping in mind the above considerations can imply that some nodes with least performance can be overhead, while other can be in an idle state. Therefore, in order to avoid this situation, it is necessary to apply some scheme of load balancing. In this paper we present a new dynamic load balancing strategy for LAN-WLAN clusters.

Keywords: *load balancing, LAN-WLAN cluster, wireless nodes, communication time.*

1. Introduction

In the last years a technological growth related to high speed networks and computers has happened. Networks of standard workstations provide an attractive scalability in terms of computation power and low cost. It is becoming strongly competitive compared to expensive parallel machines, and at present, many research groups use these environments to execute parallel scientific applications.

Cluster [1] and Grid [2] computing is a new computation idea that takes advantage of independent and different resources to build a unified resource for massive computation, distributed computation, data storing, and so on. When a parallel application is executed on a heterogeneous environment such as a network of workstations or HNOWS, several factors play an important role in the execution of applications, basically, due to the heterogeneity of the system. In the heterogeneous computing there are many open problems [3] that have not been solved yet, such as methods for dynamic task migration at execution time, effective mapping, to investigate the best way to scale these environments, and so on.

We consider HNOWS as resources with different processing power. The communication links among computers can have different speeds, as Fast-Ethernet, Ethernet and IEEE 802.11. Also, we consider this system as a multi-user environment where users can log in any time. Therefore, our research area is about of parallel computing over non-dedicated LAN-WLAN cluster [4] (figure 1).

If we execute a parallel application in a LAN-WLAN cluster and we do not keep in mind the processing power of the nodes and the different bandwidths of the communication links, then the application execution time depends on the node with the least performance. This implies that some slow nodes can be overloaded, while fast nodes can be in an idle state. Therefore, the execution time of each node must be balanced. For that reason, we need to apply some scheme of load balancing that considers both processing speed and communication time.

Load balancing implies to assign to each processor a load proportional to its performance, and so minimizing the execution time of the overall program, and preventing that some nodes can be an idle state while other are in an overload state. The minimal requirement in a load distribution scheme is to guarantee non-idleness during runtime [5].

Load balancing can be static (done at compile-time) or it may be dynamic (done at run-time) [6][7]. An optimum distribution of load to each computer can be a very arduous task if we consider an heterogeneous

* Research partially supported by Spanish CICYT under Contract: TIC2001-0956-C04-03, by ULPGC under Contract: UNI 2002/17 and by the Fundación Canaria Universitaria de Las Palmas, Lopesan, S.A. and Unelco, S.A.

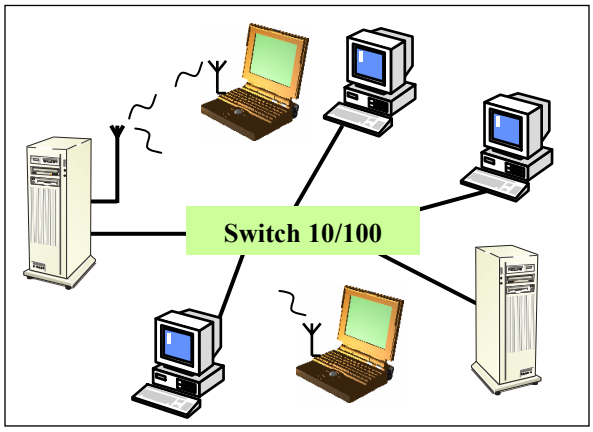

Figure 1. LAN-WLAN Cluster

environment that consists of processors with different speeds, different memory sizes, variable external load due to multiple users or network segments with heterogeneous physical layer network. Static balancing avoids overhead at run time. However, a dynamic load is more appropriate in a network of workstations because of the performance of nodes and traffic in the network changes randomly. This kind of mechanism is crucial in clusters like [4] in which nodes can be added to share the workload dynamically. We have implemented a library named LAMGAC [8] to ease the programming of MPI parallel applications in a LAN-WLAN cluster where the parallel virtual machine can change at run time of parallel applications (wireless nodes are dynamically added or removed).

In [11] we demonstrated that a strategy of load balancing in heterogeneous clusters is necessary to avoid idle state nodes. Besides, we presented a mechanism that balances the load in a heterogeneous cluster with homogeneous communication links. This mechanism obtains the optimal execution time, specially in applications with low size data communication and high calculation time. However, in parallel applications where there are high communication times, the results obtained applying this mechanism are not very good. These results could worsen if the applications are executed in a LAN-WLAN cluster.

Other recent works about heterogeneous computing [9][10][13][14] consider the load balancing keeping on mind only the different speeds of processors, however the communication time is always considered constant.

In this paper, we present a refinement of our initial ideas that constitutes a new and effective dynamic load balancing strategy that deals with the problem of load balancing in heterogeneous cluster, both computation and communication, which improves the execution time of the parallel applications executed with the mechanism presented in [11]. In practice, the execution time of the parallel applications using this technique is in a high degree decreased.

The rest of the paper is organized as follows. In section 2 we show some aspects about the LAN-WLAN cluster. In section 3 our new load balancing strategy is presented. Then, in section 4 two applications that use our strategy are explained. Next, in section 5 we present some experimental results for above applications. We briefly survey related work in section 6. Finally, we sum up the conclusions and we present the future work.

2. LAN-WLAN Cluster

The set of computers available in LAN-WLAN cluster can include a wide range of architecture types such as PC-based machines, shared memory multiprocessors, wireless laptops, etc. Despite the numerous difficulties caused by heterogeneity, parallel computing in these environments offers many advantages, such as low cost computing. On the other hand, resources take advantage of recent technologies.

In heterogeneous environments, nodes speed and data communications among processes are two dominant factors in the execution of parallel applications. Processing power of a node in a given time depends of parameters related to characteristics of resources such as CPU speed, memory size, and also, the external load at current time. Communication time depends of several factors such as network bandwidth, network topology, size of communicated data and transmission and reception buffer size. We consider both aspects in order to balance the load.

Figure 2 shows a comparison for three types of physical networks: Fast Ethernet, Standard Ethernet and IEEE 802.11 (2 Mbps). It shows the data block size versus the transfer time. The data measurement has been obtained with NetPIPE tool [12]. Clearly, we can observe that the sending of the same block size spends different time depending of network protocol. Besides, the latency of each physical network is different for all of them. Latency coincides with the time of the first data time on the corresponding curve. In an environment with this situation, and although all the resources have the same processor, we can not distribute the same load to each slave because the nodes connected with a faster network will finish before than others. Therefore, we have to consider the communication time in order to balance the load.

Also, we can observe that there is a linear ratio between the transfer time and block size, when this latter is approximately bigger than 1 Kbyte. This is, the graphs increase at constant rate, and therefore, the network throughput is not improved.

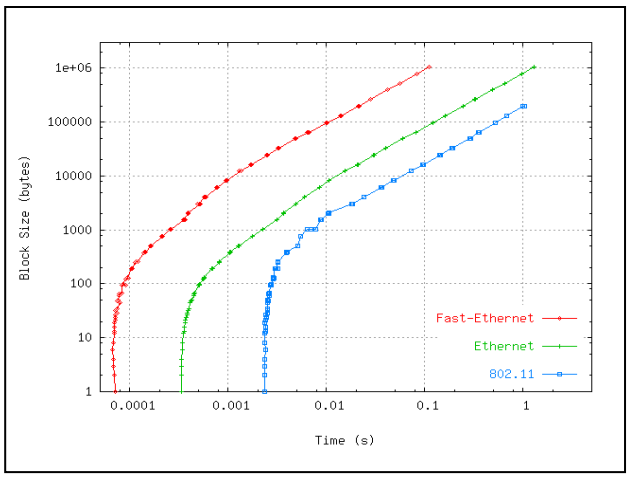

Figure 2. Time Communication vs Block Size

3. Load Balancing Strategy

In this section, we present our dynamic load balancing strategy for the execution of parallel applications in heterogeneous environments of computation and communication. Our strategy is used in applications that follow the master-slave paradigm [9]. That is, there is a master process that controls and distributes the load to the slave processes. When the slave processes finish the task, they send results back to the master process. The master process can compute some results after distributing data to the slaves and before receiving results from the slaves.

Due to we are interested in the execution of applications with dependencies among iterations, the master process can not send all data to the slaves. In other words, it is necessary to wait for results of each slave process before the master process can send new data again.

In order to avoid that some nodes are in an idle state during a long time in each iteration, an appropriate amount of data has to be sent to all slaves to finish at the same time. The execution time of each slave process must be similar, that is:

$$t\_exe_j(p_1) \approx t\_exe_j(p_2) \approx \cdots \approx t\_exe_j(p_n)$$

where

$$t\_exe_j(p_i) = t\_comm_j(p_i) + t\_calc_j(p_i) + t\_idle_j(p_i)$$

The calculation time of process i during the iteration j ($t\_calc_j(p_i)$) is the time spent to carry out the calculations. The communication time of process i during the iteration j ($t\_comm_j(p_i)$) is the time elapsed to send data from the master process to the process i and send results from slave process i to the master process. During the idle time of process i in the iteration j ($t\_idle_j(p_i)$) neither calculation nor communication is done. Our strategy minimizes this last parameter. In order to carry out the load balancing, we need to do three steps in each iteration:

- Collect information about process performance
- Estimate the next data distribution
- Send data and receive results for each process

These steps are explained next.

3.1. Collecting Information about Process Performance

In order to distribute an appropriate load to each slave it is necessary to know some characteristics about nodes performance that will make calculations. If a dynamic load balancing is used these characteristics have to be known at run time. To carry out an effective load balancing, the strategy presented in this paper needs to collect the following information:

- Execution time for each process, $t\_exe_j(p_i)$
- Calculation time spent for each process, $t\_calc_j(p_i)$

The execution time is measured in the master process, and is the time elapsed from the data are sent to the first slave process until the results calculated by the process i are received, therefore, implicitly we are considering the communication time. The calculation time is measured by own process i and it is communicated to the master process together with the results (figure 3).

This information is collected in each iteration, therefore, immediately we know the capabilities of a process in the node where it is running, and we can detect any imbalance at runtime.

3.2. Estimate the next Data Distribution

In order to estimate the next data distribution we use the following metric:

$$n\_unit_{j+1}(p_i) = \frac{\mu\_t\_exe_j}{t\_calc\_unit_j(p_i) + t\_comm\_unit_j(p_i)}$$

where:

- $n\_unit_{j+1}(p_i)$ is the number of units to send to the process i in the iteration $j+1$. A unit is the minimum amount of data to calculate one unit result, which represents the data granularity. For example, for matrix multiplication, a unit is equal to one row if the master process distributes rows among slave processes.

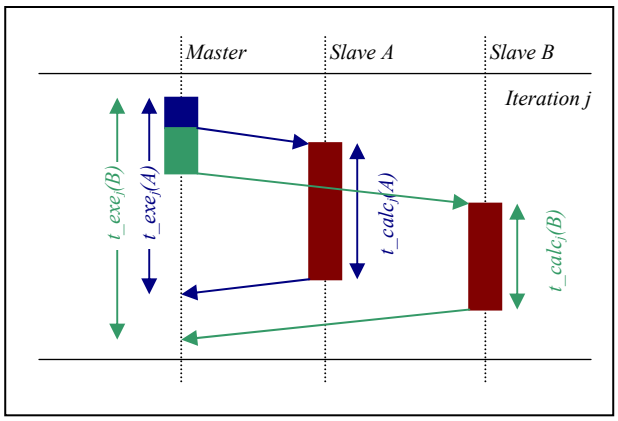

Figure 3. Execution and Calculation Time Measurements

- $\mu\_t\_exe_j$ represents the arithmetic average of the execution time of all slave processes in the iteration *j*.
- $t\_calc\_unit_j(p_i)$ is the ratio between $t\_calc_j(p_i)$ and $n\_unit_j(p_i)$. It gives us the calculation time by data unit for each process. If our environment of computing was a dedicated cluster this value should be constant in each computing node.
- $t\_comm\_unit_j(p_i)$ is the result of dividing $(t\_exe_j(p_i) - t\_calc_j(p_i))$ and $n\_unit_j(p_i)$. It gives us the communication time by data unit for each process. Therefore, here we are considering the communication link from master to slave process. Due to the master process measures the execution time from the data are sent to the first slave process until the results are received, the idle time of each slave process is considered within the execution time.

The communication time by data unit can be calculated using a linear ratio due to an increase in data block size results in a near-linear increment in transfer time. This is a valid value when the block size is approximately bigger than 1 Kbytes, as we shown in the figure 2.

In the first iteration, we do not know any information about process performance in each node, therefore the same amount of data is distributed to all processes.

3.3. Sending Data and Receiving results for each Process

Once the next data distribution is calculated, the following step is to send the data to the slave processes. The master and the slaves communicate through a shared medium, and in a given time-step can be accessed only in exclusive mode. Normally, the data are sent to the slave processes beginning with the faster worker and ending with the slowest one. It is done to avoid the faster workers are in idle state while are waiting for data. Like a LAN-WLAN cluster has communication links with different bandwidth, it is necessary to consider this situation to establish an ordered distribution of data. For example, if the faster worker has a bandwidth connection to 2 Mbps, it is possible that the rest of processes with high bandwidth have to wait more to receive the data.

In order to avoid this problem we have defined the following parameter named *perf_f*, which stands for performance factor:

$$perf\_f_j(p_i) = \frac{n\_unit_j(p_i)}{t\_exe_j(p_i)}$$

It is the inverse of the execution time by data units sent to process *i* in the iteration *j*, and it informs us about the process with the best performance. Therefore, the data units will be distribute beginning by the process with larger *perf_f* in the previous iteration, and so on.

Initial and manually, we make a mapping of slave processes to all wired nodes available in the cluster. The wireless nodes can be added or removed at run-time with the LAMGAC [8] library. If the load balancing strategy decides do not send data to a slave process, a signaling message is sent to that process to finish the execution. This situation can occur when there is a process with shortest performance than others, and therefore a balanced execution time can not be reached.

4. Application Examples

We have used two different applications for testing our strategy of load balancing: a Hw/Sw Codesign Tool and a Matrix Multiplication. The former needs a high power processing and the data packets size to send is small, while the latter uses a low power processing and the data packets size to distribute is big.

4.1. Hw/Sw Codesign Application

Our research group has developed a Hw/Sw Codesign Tool named GACSYS [16][17]. The objective of this tool is to optimize and satisfy a given set of design constraints. This tool is divided in two phases:

- *Parameter Estimation Phase*. Hw/Sw parameter estimations are obtained for each process of the input data specification written in VHDL language such as execution time, power consumption and area. Depending of functional units number, this phase can be heavy computationally.

- *Partitioning Phase*. An optimization algorithm based on Simulated Annealing finds a partition of functional units that satisfies the design constraints.

For testing our load balancing strategy we have used the parameter estimation phase.

In the solution of this problem, the data distribution consists of sending to each slave process the number of combinations of functional units and the combination from which it will begin to calculate. The data packet size of this information in bytes is the size of two integer values, usually eight bytes. Initially, the same number of combinations is sent to all slaves. For each combination, the slave process will carry out the estimation of parameters, and it will send the results to master process. The data packet size of results is the multiplication of the number of combinations by size of three float values (execution time, power consumption and area). When the master process receives the results of all slaves, it calculates a new data distribution (section 3.2.) with data time collected according to the section 3.1. These steps are repeated. As you can see, the data size communicated in each iteration is relatively small.

4.2. Matrix Multiplication Application

We want to calculate C = AxB, where the three matrixes A, B, and C are dense and square of order *n*. Initially, the first column of B is broadcasted to all slave processes, and the same amount of rows is sent to all slaves. The data packet size of this information is the size of an element of the matrix multiplied by the order *n* and by the number of rows. Each slave process computes the multiplication of each row by the column of B, and sends the results to the master process. Then, in the following iteration the load balancing strategy sends a given number of rows of A according with the collected time data. This task is repeated until all results have been obtained for all rows. Next, the second column of B is broadcasted to all slave processes, and the steps are repeated, and so on. The data packet size of results is the size of an element of the matrix by the number of rows. As it can see, the data size communicated in each iteration can be big.

5. Experimental Results

In [11] we demonstrate that a load balancing mechanism is necessary when a parallel application is executed in a heterogeneous environment. However, this mechanism was tested for cluster with homogeneous communication. In this section, we present some experimental results for our new load balancing strategy described in this paper, and are compared with the results obtained with the mechanism presented in [11], when the parallel application is executed in a LAN-WLAN cluster.

We have realized the experiments in a network of workstations with the specifications of table 1. The wired computers are connected to a switch 10/100 Mbps. The node where the master process is executed has two network cards: a Fast-Ethernet card to communicate with wired nodes, and a wireless card (IEEE 802.11 at 2 Mbps) to communicate with wireless nodes.

Table 1. Computing Resources Characteristics

Processor / Memory Size	Network Card (Mbps)
PIII 666Mhz / 128MB (master)	100 / 2
PIII 1Ghz / 256MB	100
PIII 666Mhz / 128MB	100
Celeron 633Mhz / 128MB	10
P 200Mhz MMX / 64 MB	10
PII 300 Mhz / 64 MB	2

We used C language and MPI library [18] to implement the applications described above.

5.1. Hw/Sw Codesign

Several experiments have been implemented to obtain performance results. Concretely, we made four experiments varying the limit of functional units available for the estimation phase. The execution time of this application depends directly of the bounds of functional unit established.

Figure 4 shows performance results for the simulations implemented with the strategy presented in this paper (labeled as New) and the mechanism presented in [11] (Old hereafter). Axis X represents the simulations, and axis Y represents execution time in seconds. We observe that the execution time using the new strategy is lightly better than the old mechanism. This little difference is due to the communication time in Hw/Sw Codesign application is smaller than the calculation time, and therefore, the heterogeneity only is affected by the processing power of each node, and therefore a bigger difference can not be reach. Old strategy is excellent for this type of applications.

Figure 5 shows the average execution time of all slave processes for the simulation S2 during the first seven iterations. Also, shows the typical deviation. Axis X represents the iterations, and axis Y represents time in seconds. As we can see, in the iteration one and two there is a high typical deviation. However, this value is going decreased with the iterations, and therefore, the execution time of each slave process is similar. It demonstrates that

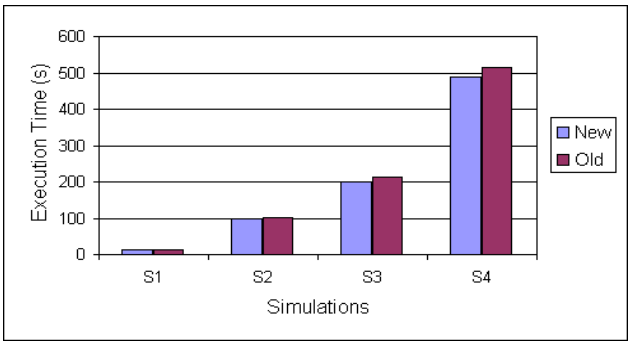

Figure 4. Hw/Sw Codesign: Execution Time

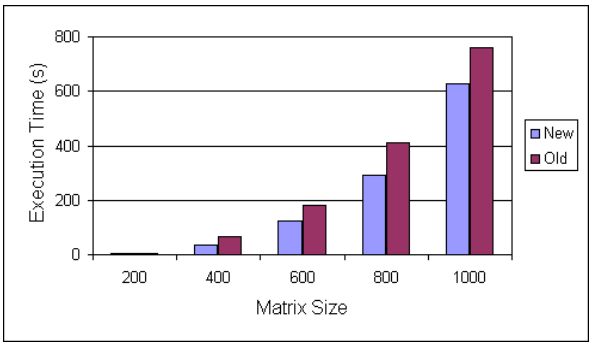

Figure 6. Matrix Multiplication: Execution Time

a balanced execution time is obtained during the execution of the parallel application.

5.2. Matrix Multiplication

In order to test our strategy with this application we have used different sizes of matrixes of integers. In particular, we made experiments with matrixes of order 200, 400, 600, 800 and 1000. Figure 6 shows performance results for different matrix sizes. We clearly observe that the execution time using the strategy presented in this paper is better than mechanism presented in [11]. Besides, when the order of matrix is increased the difference among strategies is higher. In this application there is a communication overhead due to the large data packets size distributed, and therefore, it is affected by heterogeneity both communication and calculation. With this experiment, we demonstrate that the mechanism presented in [11] is not very good when there is heterogeneity in the physical network and high communication, and the results are considerably improved.

Figure 7 shows the average execution time of all slave processes for the experiment with the matrix of order 1000, and also shows the typical deviation. The figure shows the first seven iterations of the application. The conclusions obtained for the figure 5 can be applied at this one. That is, a balanced execution time is obtained during the execution of the parallel application. Due to the data packet size sent for this application can be larger and, therefore the communication time influences considerably in the execution time, the nodes with least performance can be removed for the load balancing strategy because of they can not reach a balance. In the experiments for this application, usually removed P200 MMX processor is in the fourth or fifth iteration.

6. Related Work

In this section we discuss some recent works about load balancing techniques in heterogeneous environments.

O. Beaumont et al. [9] present theoretical ideas to give an optimal solution in the execution the parallel application that follows the master-slave paradigm with heterogeneous processors. They consider independent tasks to be processed by slaves, and assume that the cost of data communication is always equal. Due to we are interested in the execution of applications in a LAN-WLAN cluster with dependencies among iterations and

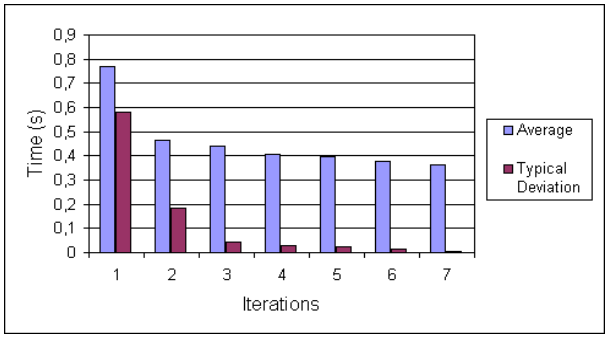

Figure 5. Hw/Sw Codesign: Average Time and Typical Deviation

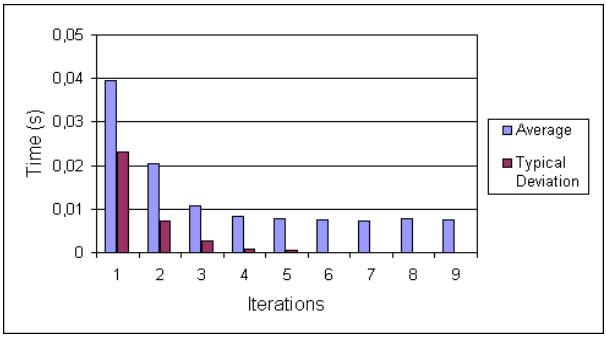

Figure 7. Matrix Multiplication: Average Time and Typical Deviation

with heterogeneous links among nodes, we can not assume their ideas.

Techniques describe in papers [10][13] use different parameters to characterize the performance of a node. In [10] is used a normalized relative processing power of each workstation. Reference [13] defines the degree of heterogeneity, which represents the difference in processing capabilities between machines for the application given. We can not adopt only these parameters because we consider the LAN-WLAN cluster as multi-user environment, and therefore, the processing power of each machine is not fully available for each task.

M. Eggen et al. [14] use the average number of jobs in the run queue to know the amount of work and to make an initial distribution. For the following iterations, it re-distributes in terms of the work completed during a time interval. It obtains a balance in a non-dedicated environment; however, it considers a homogeneous network to interconnect the resources. We keep in mind the communication time for each node to execute parallel applications in a cluster with heterogeneous networks.

D. Kulkarni et al. [15] consider the communication between nodes using an external application that collects information about the network. Then, the parallel application is supplied with the network information only if this information becomes critical, with which bounds are established. Our strategy collects information about nodes considering communication times, and it doesn't need an external application. Besides, our experience in the execution of parallel application says to us, that a minimal variation in the network can imbalance the application. Therefore, in [15] depending of the bounds established, the external application could inform with quite frequency generating enough network traffic, or it could not detect network variations if the bounds are higher. In both cases a load imbalance can happen.

7. Conclusions and Future Work

In this paper, a new strategy that improves the previous results obtained by us is presented, specially it improves when in the parallel application a high data bulk is communicated.

We will consider a static load balancing that keeps in mind the nodes performance at the beginning of the execution, and we will implement a library to ease the programming of parallel applications in these environments considering our load balancing strategy.

References

[1] M. Baker. *Cluster Computing White Paper.* Final Release, Version 2.0., 28th December 2000.

[2] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure.* Morgan Kaufmann, 1999.

[3] T.D. Braun, H.J. Siegel, A.A. Maciejewski. Heterogeneous Computing: Goals, Methods, and Open Problems. *Proceedings of Parallel and Distributed Processing, Techniques and Applications*, vol I, pp. 7-18. Las Vegas, Nevada, USA. June 2001.

[4] E. Macías, A. Suárez, C.N. Ojeda-Guerra, E. Robayna. Experimenting with the Implementation of Parallel Programs on a Communication Heterogeneous Cluster. *Proceedings of Parallel and Distributed Processing, Techniques and Applications*, vol II, pp. 829-835. Las Vegas, USA. June 2001.

[5] Markus Fisher. *Dynamic Load Balancing for Heterogeneous Parallel Enviroments*. Master Thesis. Paderborn, April 1997.

[6] M. Zaki et al. Customized Dynamic Load Balancing for a Network of Workstation. *Proceedings of 5th High Performance Distributed Computing*. Syracuse, NY, USA, August 1996.

[7] Shahzad Malik. *Dynamic Load Balancing in a Network of Workstations*. 95.515F Research Report. November 2000.

[8] E. Macías, A. Suárez, C.N. Ojeda-Guerra, E. Robayna. Programming Parallel Applications with LAMGAC in a LAN-WLAN Environment. *8th European PVM/MPI.* LNCS 2131. Springer Verlag, pp. 158-165. September 2001.

[9] O. Beaumont, A. Legrand, Y. Robert. The Master-Slave Paradigm with Heterogeneous Processors. *Proceedings of International Conference on Cluster Computing.* 2001.

[10] F. Tinetti et al. Heterogeneous Networks of Workstations and Parallel Matrix Multiplication. *8th European PVM/MPI.* LNCS 2131. Springer Verlag, pp. 296-303. September 2001.

[11] D. Sánchez, E. Macías, A. Suárez. An Application Level Load Balancing Mechanism for Heterogeneous Clusters Programming. *Proceedings of Parallel and Distributed Processing, Techniques and Applications*, vol II, pp. 872-878. Las Vegas, Nevada, USA. June 2002.

[12] Q. Snell, A. Mikler, J. Gustafson. NetPIPE: A network protocol independent performance evaluator. *International Conference on Intelligent Information Management and Systems.* June 1996.

[13] A. Furtado et al. Architectures for an Efficient Execution in a Collection of HNOWS. *9th European PVM/MPI.* LNCS 2474. Springer Verlag, pp. 450-460. September 2002.

[14] M. Eggen, R. Eggen. Load Balancing on a Non-dedicated Heterorgeneous Network of Workstations. *Parallel and Distributed Processing, Techniques and Applications*, vol II, pp. 856-862. Las Vegas, Nevada, USA. June 2002.

[15] D. Kulkarni, M. Sosonkina. Minimizing Communication Waiting Time in Sparse Linear Solution using Dynamic Network Information. *Proceedings of Communications in Computing*, Las Vegas, Nevada, USA. June 2002.

[16] J.P. Castellano, D. Sánchez, O. Cazorla, J. Bordón, A. Suárez. GACSYS: a VHDL based Hw/Sw Codesign Tool. *Design and Diagnostics of Electronic Circuits and Systems*, pp 293-299. Szcyrk, Poland. September 1998.

[17] D. Sánchez et al. Hardware/Software Partitioning based on Simulated Annealing. *Modeling and Simulation,* pp 115-122. Las Palmas de Gran Canaria, Spain. September 2000.

[18] Information available in http://www.mpi-forum.org.