Comparative analysis of the prototype and the simulator of a new load balancing algorithm for heterogeneous computing environments

R.F. de Mello*
Institute of Mathematical Sciences and Computing,
University of São Paulo, São Carlos, Brazil
E-mail: mello@icmc.usp.br
*Corresponding author

L.C. Trevelin
Department of Computer Science,
Federal University of São Carlos, Brazil
E-mail: trevelin@dc.ufscar.br

M.S.V. de Paiva
Department of Electrical Engineering,
São Carlos Engineering School,
University of São Paulo, Brazil
E-mail: mstela@sel.eesc.usp.br,

Laurence Tianruo Yang
Department of Computer Science,
St Francis Xavier University, Canada
E-mail: lyang@stfx.ca

Abstract: The availability of low cost hardware has increased the development of distributed systems. The allocation of resources in these systems may be optimised through the use of a load balancing algorithm. The load balancing algorithms are responsible for the homogeneous distribution of the occupation in the environment, with the aim of obtaining gains on the final performance. This paper presents and analyses a new load balancing algorithm that is based on a logical computer tree hierarchy. The analysis is made using results obtained by a simulator and a prototype of this algorithm. The algorithm simulations show a significant performance gain, by lowering the response times and the number of messages that pass through the communication system to perform the load balancing operations. After good results were obtained by the simulations, a prototype was built and validated such results.

Keywords: load balancing algorithm; heterogeneous environments; high performance; simulation; prototype.

Reference to this paper should be made as follows: de Mello, R.F., Trevelin, L.C., de Paiva, M.S.V. and Yang, L.T. (2004) 'Comparative analysis of the prototype and the simulator of a new load balancing algorithm for heterogeneous computing environments', Int. J. High Performance Computing and Networking, Vol. 1, Nos. 1/2/3, pp.64–74.

Biographical notes: Rodrigo Fernandes de Mello is a Professor at The Institute of Mathematical Sciences and Computing at the University of São Paulo, São Carlos, Brazil. He completed his PhD degree from University of São Paulo, São Carlos in 2003. Since 1998 he has been researching in distributed systems, real time kernels and internet.

1 INTRODUCTION

The availability of low cost microprocessors and the evolution of computing networks have increased and enabled the construction of distributed systems. On such systems, the processes are executed on network computers and communicate to each other to perform a collaborative computing task. In order to distribute the processes among the computers of these systems, a load balancing algorithm may be adopted.

The load balancing algorithms are responsible for the equal distribution of the processes load among the computers of an environment. Krueger and Livny (1987) demonstrate that load balancing methods can reduce the average and standard deviation of processes response times. Shorter response times are desirable, as they are related to high performance in the execution of the processes.

The load balancing algorithms involve four policies: transference, selection, location and information (Shivaratri et al., 1992; Sinha, 1997). The transference policy determines whether a computer is in a suitable state to participate in a task transfer, either as server or receiver of the processes. The selection policy defines the process that should be transferred from the busiest computer to the idlest one. The location policy is responsible to find a suitable transfer partner (sender or receiver) for a computer once the transfer policy has decided about its state. A serving computer offers the processes when it is overloaded, a receiving computer requests processes when it is idle. The information policy defines when and how the information on the computers' availability is updated on the system.

Several load balancing methods have been proposed (Shivaratri et al., 1992; Zhou and Ferrari, 1987; Theimer and Lantz, 1988). According to Shivaratri et al. (1992), these algorithms can be subdivided into:

- server-initiated
- receiver-initiated
- symmetric initiated
- stable server-initiated
- stable symmetrically initiated.

Out of these algorithm classes, the one that shows the highest performance is the stable symmetrically initiated.

Despite recent works on load balancing (Hui and Chanson, 1999; Mitzenmacher, 2000; Anderson and Abraham, 2000; Amir, 2000; Kostin et al., 2000; Figueira and Berman, 2001; Zomaya and Teh, 2001; Zhang, 2001; Mitzenmacher, 2001; Woodside and Monforton, 1993; Page et al., 1993; Hsu et al, 2000), the studies and algorithms from Krueger and Livny (1987), Zhou and Ferrari (1987), Theimer and Lantz (1988) and Shivaratri et al. (1992) have been considered the main works for this area. The recent work does bring significant contributions to the processes' performance and reduction of the number of messages on the computer network. The recent work in this area, presented in this paper, is based on the main studies of the load balancing area.

Figure 1: A 2-D, 9-velocity lattice (Q9D2) model A 2-D, 9-velocity lattice (Q9D2) model A 2-D, 9-velocity lattice (Q9D2) model A 2-D, 9-velocity lattice (Q9D2) model A 2-D, 9-velocity lattice (Q9D2) model A 2-D, 9-velocity lattice (Q9D2) model

Later on, Mello et al. (2002a,b,c,d), during others studies on clusters and distributed systems, observed and proposed a new method for load balancing. Through these studies they concluded that by choosing a load index based on the process behaviour and on the computer capacity at the environment, it would be possible to obtain better results (Mello et al, 2003). Based on these studies, the authors show a comparative study between the Lowest load balancing algorithm using the new load index and the original Stable Symmetrically Initiated algorithm. According to the obtained results in Mello et al (2003), the improved Lowest algorithm shows the highest performance. These analyses confirm that the adequate load index choice modifies the behaviour of the load balancing algorithm.

Vital aspects for the success of a load balancing algorithm are: stability on the number of messages generated in order to not overload the communication system; support for environments composed of heterogeneous computers and processes; low cost update on the environment load information; low cost choice for the ideal computers to execute a process received by the system; stability on high load; system scalability, and short response times. The study and analysis of each of the above mentioned aspects influenced the definition of a new load balancing algorithm named TLBA (Tree Load Balancing Algorithm).

This paper presents the TLBA, a new load balancing algorithm focusing on environments where: high scalability; support for heterogeneous processes and computers; small number of messages that pass through the communication system; stability on high loads, and short response times are desired. A comparative study between the TLBA and the improved Lowest algorithm has been realised by Mello et al (2003) using simulators. The TLBA is compared

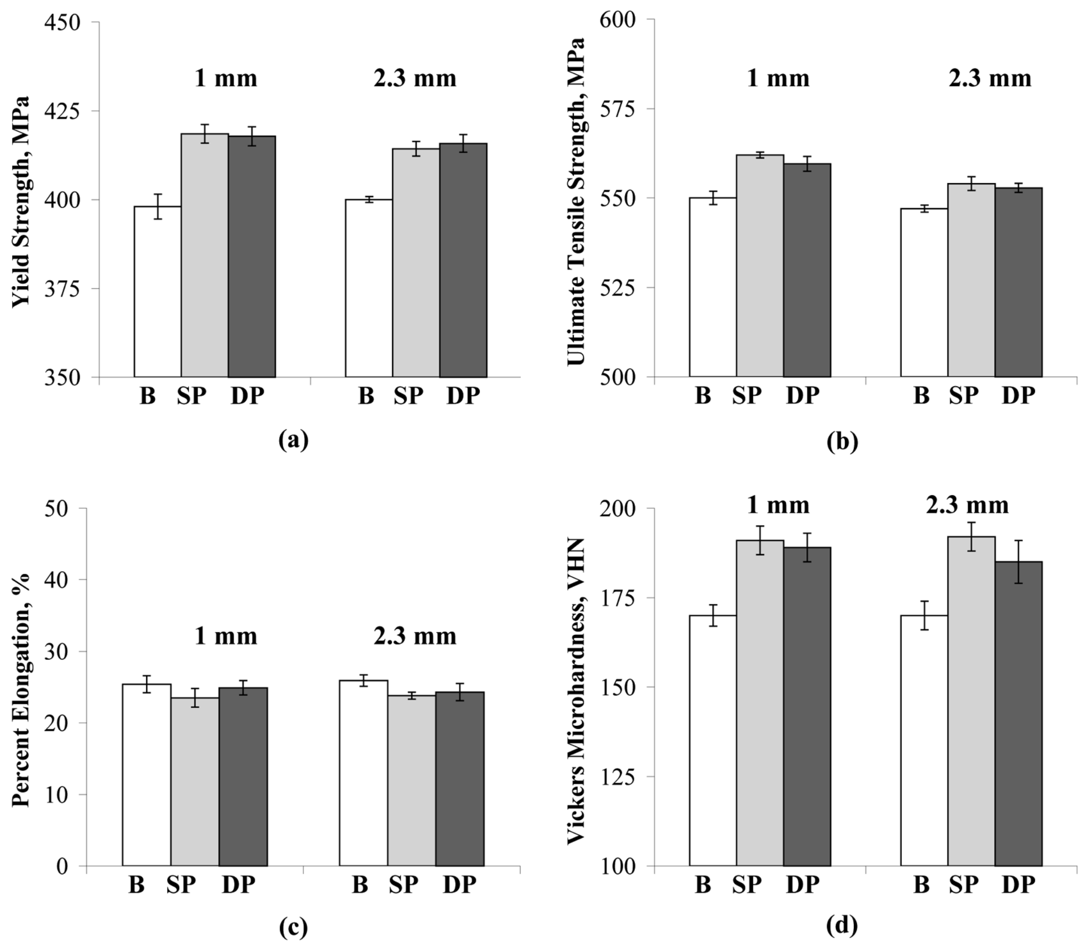

Figure 2: Effect of multiple welds on mechanical properties of single-pass (SP) and double-pass (DP) Cp-Ti weldments compared with the as-received material (B): (a) yield strength, (b) ultimate strength, (c) elongation, and (d) hardness.

to improved Lowest algorithm, because the last yielded better results than Stable symmetrically initiated. After simulating and proving the performance gains of TLBA, a prototype has been implemented. This prototype was submitted to tests and the results confirmed the values obtained through the simulations.

The paper has been divided into the following Sections: Section 2 presents the State of Art; Section 3 presents the Tree Load Balancing algorithm; Section 4 presents comparative studies based on simulations; Section 5 presents the TLBA prototype and compares it to the results obtained on the simulations; Section 6 presents the conclusions.

2 STATE OF THE ART

Among the main studies on load balancing, we may highlight the ones by Zhou and Ferrari (1987), Theimer and Lantz (1988), Shivaratri et al. (1992), and Mello et al (2003). Other recent works were proposed (Hui and Chanson, 1999; Mitzenmacher, 2000; Anderson and Abraham, 2000; Amir, 2000; Kostin et al., 2000; Figueira and Berman, 2001; Zomaya and Teh, 2001; Zhang, 2001; Mitzenmacher, 2001; Woodside and Monforton, 1993; Page et al., 1993; Hsu et al, 2000); however they do not offer significant performance contributions. These recent works are also based on the studies of Zhou and Ferrari (1987), Theimer and Lantz (1988) and **?**.

Zhou and Ferrari (1987) evaluate four server-initiated load balancing algorithms, i.e., initiated by the most overloaded computer. These algorithms are: Disted, Global, Central, Random and Lowest. On Disted, when a computer realises any alteration on its load, it emits messages to the other computers to inform them about its current load. On Global, there is a computer that centralises all the computers' load information on that environment and this centralising computer sends broadcasts to the environment in order to keep the other ones updated about the situation. On Central, as on Global, a central computer receives all the load information of the system; however, it does not update the other computers about it. This centralising computer decides about the resources allocation at the environment. On Random, no information about the environment load is handled. On this algorithm, a computer is selected at random in order to receive a process to be initiated. On Lowest, the load information is sent when demanded. When a computer starts a process, it requests information and analyses the loads of a small

Table 1: Response time variation of the lowest algorithm to the different occupations of the process

Occupation	*Average response time of the improved lowest algorithm*	*Average response time of the TLBA*
1.00 E + 01	6.1275533000 E+ 05	6.0450143000 E + 05
1.00 E + 02	6.3940207300 E + 06	6.3176135300 E + 06
1.00 E + 03	6.4152160200 E + 07	6.3446532460 E + 07
1.00 E + 04	6.4098737963 E + 08	6.3473560175 E + 08
1.00 E + 05	6.4189585691 E + 09	6.3476263532 E + 09
1.00 E + 06	6.5020352364 E + 10	6.3476532508 E + 10
1.00 E + 07	6.4361410067 E + 11	6.3476559492 E + 11
1.00 E + 08	6.4261208947 E + 12	6.3476562200 E + 12
1.00 E + 09	6.4770720080 E + 13	6.3476562470 E + 13
1.00 E + 10	6.4230638584 E + 14	6.3476562497 E + 14
1.00 E + 11	6.5764266304 E + 15	6.3476562500 E + 15
1.00 E + 12	6.4104959239 E + 16	6.3476562500 E + 16

set of computers and submits the processes to the idlest one. The idlest computer has the shortest processes queue.

When Zhou and Ferrari (1987) analysed the Disted, Global, Central, Random and Lowest algorithms, they reached the conclusion that all of them show a performance higher than the one presented by a system with no load balancing at all. However, the Lowest algorithm showed the highest performance of all. This algorithm generates demand messages to transfer the process, causes a lower overload on the communication system and, thus, allows the incremental growth of the environment. The number of generated messages in an environment does not depend on the number of computers.

Theimer and Lantz (1988) have implemented algorithms similar to the Central, Disted and Lowest ones. Nevertheless, they have analysed such algorithms in systems composed of a larger number of computers (about 70). For the Disted and Lowest ones, some process receiver and sender groups were created. The communication within these groups was made by using a multicast protocol, in order to minimise the messages exchange among the computers. Computers with load lower than an inferior limit participate in the process receiver group whilst the computers with load higher than a superior limit participate in the process sender group.

Theimer and Lantz recommend decentralised algorithms such as Lowest and Disted, as they do not generate single points of fault as Central does. Central presents the highest performance for small and medium size networks but it degrades on large environments.

Through the analyses, Theimer and Lantz (1988) concluded that algorithms such as Lowest work with the probability of a computer being idle. They assume system homogeneity as they use the size of the CPU's waiting queue as load index. The process behaviour is not analysed; therefore, the actual load of each computer is not measured.

Shivaratri et al. (1992) analyse the algorithms, server-initiated, receiver-initiated, symmetrically initiated, adaptative symmetrically initiated and stable symmetrically initiated. In these studies, the length of the process waiting queue at the CPU was considered as the load index. This measure has been chosen because it is simple and therefore, consumes fewer resources.

Table 2: Lowest algorithm response times for the different process occupation at a heterogeneous system

Average response time of the improved lowest algorithm	*Average response time of the TLBA*
6.1275533000 E+ 05	6.0450143000 E + 05
6.3940207300 E + 06	6.3176135300 E + 06
6.4152160200 E + 07	6.3446532460 E + 07
6.4098737963 E + 08	6.3473560175 E + 08
6.4189585691 E + 09	6.3476263532 E + 09
6.5020352364 E + 10	6.3476532508 E + 10
6.4361410067 E + 11	6.3476559492 E + 11
6.4261208947 E + 12	6.3476562200 E + 12
6.4770720080 E + 13	6.3476562470 E + 13
6.4230638584 E + 14	6.3476562497 E + 14
6.5764266304 E + 15	6.3476562500 E + 15
6.4104959239 E + 16	6.3476562500 E + 16

Shivaratri et al. have concluded that the receiver-initiated algorithms present a higher performance than the server-initiated ones, such as Lowest. In their conclusions, the algorithm that showed the highest final performance was the stable symmetrically initiated one. This algorithm preserves the history of the load information exchanged in the system and takes actions to transfer the processes by using this information.

Mello et al (2003) analyse a load index based on the process behaviour as well as on the system computers capacity. The authors propose a comparison between the Stable Symmetrically Initiated algorithm by Shivaratri et al. (1992) and the improved Lowest algorithm. The improved Lowest uses an analysis of the process behaviour, as well as the system computers' capacity as load index. Through these studies, it was concluded that, by using a new load index, the Lowest algorithm provides better response times. These results were reached through the simulation of homogeneous and heterogeneous environments.

3 THE TREE LOAD BALANCING ALGORITHM

The TLBA algorithm, named Tree Load Balancing Algorithm, creates a virtual interconnecting tree among the computers of the system. On this tree, each computer of an N level sends its updated load information to the $N-1$ level computers. The lower the level is, the closer it is to the root. The root is located on level zero.

The selection of the best computer to execute a process received by the system, works as a deep search on the interconnecting tree. Thus, a level N computer may define which of its branches located in level $N+1$ presents the lowest occupation, by re-passing a search message that is propagated through all the branches of the tree until it finds the idlest computer.

The tree may be defined as a non-cyclic connected graph. $G = (X, U)$ being a graph, where X is the set of vertices and U is the set of edges which interconnects an ordered pair. This graph presents $n > 2$, where n is the maximum number of elements of the vertices set and satisfies the following properties:

- G is connected and non-cyclic
- G is non-cyclic and has $n-1$ edges
- G is connected and has $n-1$ edges
- G is non-cyclic and, by adding one edge, only one cycle is created
- G is connected, but stops being so if one edge is suppressed
- Every G's vertices pair is connected by only one simple chain.

After defining the tree, it is necessary to define the levels that make it up. Being the same oriented graph $G = (X, U)$ shown before, a partition of N of the set X may be defined as:

$$N = \{N0, N1, \ldots, Nr\}$$

where the elements $N0, N1, \ldots, Nr$, called levels, are arranged by the inclusion of their $R^{-1}(xi)$ as follows:

- $N0 = \{xi/R^{-1}(xi) = \{\}\}$
- $N1 = \{xi/R^{-1}(xi)CN0\}$
- $N2 = \{xi/R^{-1}(xi)C(N0UN1)\}$
- $Nr = \{xi/R^{-1}(xi)CU(dek = 0ar - 1)Nk\}$

$R^{-1}(xi)$ is the set of predecessor vertices of an element xi of the set of vertices X. Every vertice xi only has predecessors in the previous levels. The last level of the tree is depicted by $R^{+}(Nr) = \{\}$, where Nr is the largest level and $R + (Nr)$ the set of vertices which succeed the level Nr.

3.1 Information policy

On TLBA, each computer of the system has an updated table with the occupation information. This table includes information about the computer as well about the other computers in its subtree. The subtree of a computer α, located on level N, is given by all the computers from the level $N+1$, which succeed to a computer through an oriented arch. The computer α participates in this subtree, being defined as its father.

The computers at level $N+1$ send information about their load to the predecessor computer located on the level N. The father computer generates a single number that represents its occupation and the one of its subtree. To make this calculation, the father computer sums up the occupation related to the capacities of each computer of its subtree, generating a single load value for the subtree controlled by it. The load value is propagated until the root node.

A computer located in level $N+1$ only sends its load information when the change exceeds a certain threshold percentage value. Assuming a threshold of 5%, the current load is compared to the one taken at the latest reading; if the current load is 5% above or below the previous load, then the load information is sent to the father computer, which is located in the level N. The definition of this threshold allows the decrease in the number of messages related to the information update on the system. Decreasing the communication system load increases the stability (Shivaratri et al., 1992).

3.2 Load index calculation

The load index used by TLBA is based on the analysis of the CPU and memory resources consumed by the processes. This load index proposed by **?** is detailed as following.

Consider as the occupation of the computing resources $O_r(t, i)$, where: i identifies the analysed computer; r is the computing resource analysed (CPU and main memory) and t is the occupation reading instant.

ACKNOWLEDGMENT

We would like to thank Dr. Gonzalo Travieso for his collaboration in this work.

REFERENCES

Amir, Y. *et al.* (2000) 'An opportunity cost approach for job assignment in a scalable computing cluster', *IEEE Transactions on Parallel and Distributed Systems*, July, Vol. 11, No. 7, pp.760–768.

Anderson, J.R. and Abraham, S. (2000) 'Performance-based constraints for multidimensional networks', *IEEE Transactions on Parallel and Distributed Systems* January, Vol. 11, No. 1, pp.21–35.

Figueira, S.M. and Berman, F. (2001) 'A slowdown model for applications executing on time-shared clusters of workstations', *IEEE Transactions on Parallel and Distributed Systems*, June, Vol. 12, No. 6, pp.653–670.

GNU/GCC Supported Platforms (2003) URL: http://gcc.gnu.org/install/specific.html, Consulted in: 25th June, http://www.icmc.usp.br/∼mello.

Hsu, T. *et al.* (2000) 'Task allocation on a network of processors', *IEEE Transactions on Parallel and Distributed Systems*, December, Vol. 49, No. 12, pp.1339–1353.

Hui, C. and Chanson, S.T. (1999) 'Hydrodynamic load balancing', *IEEE Transactions on Parallel and Distributed Systems*, November, Vol. 10, No. 11, pp.1118–1137.

Kostin, A.E., Aybay, I. and Oz, G. (2000) 'A randomised contention-based load-balancing protocol for a distributed multiserver queuing system', *IEEE Transactions on Parallel and Distributed Systems*, December, Vol. 11, No. 12, pp.1252–1273.

Krueger, P. and Livny, M. (1987) 'The diverse objectives of distributed scheduling policies', *Proc. Seventh Int'l Conf. Distributed Computing Systems*, IEEE CS Press, Los Alamitos, Calif., No. 801, (microfiche only), pp.242–249.

Mello, R.F. *et al.* (2003) 'Comparative study of the server-initiated lowest algorithm using a load balancing index based on the process behaviour for heterogeneous environment, accepted for publication in the special issue of cluster computing of the Kluwer', *The Journal of Networks, Software Tools and Applications.*

Mello, R.F., Paiva, M.S. and Trevelin, L.C. (2002b) 'OpenTella: a peer-to-peer protocol for the load balancing in a system formed by a cluster from clusters', *4th International Symposium on High Performance Computing ISHPC-IV*, Japan, May.

Mello, R.F., Paiva, M.S. and Trevelin, L.C. (2002c) 'Analysis on the significant information to update the tables on occupation of resources by using a peer-to-peer protocol', *16th Annual International Symposium on High Performance Computing Systems and Applications*, Moncton, New-Bruncwick, Canada, June.

Mello, R.F., Paiva, M.S. and Trevelin, L.C. (2002d) 'A java cluster management service', *IEEE International Symposium on Network Computing and Applications*, Cambridge, Massachusetts, USA, February.

Mello, R.F., Paiva, M.S., Trevelin, L.C. and Gonzaga, A. (2002a) 'Model for resources load evaluation in clusters by using a peer-to-peer protocol', *The 2002 International Conference on Parallel and Distributed Processing Techniques and Applications*, Las Vegas, USA, June.

Mitzenmacher, M. (2000) 'How useful is old information?', *IEEE Transactions on Parallel and Distributed Systems*, January, Vol. 11, No. 1, pp.6–20.

Mitzenmacher, M. (2001) 'The power of two choices in randomised load balancing', *IEEE Transactions on Parallel and Distributed Systems*, October, Vol. 12, No. 10, pp.1094–1104.

Page, I, Jacob, T. and Chern, E. (1993) 'Fast algorithms for distributed resource allocation', *IEEE Transactions on Parallel and Distributed Systems*, February, Vol. 4, No. 2, pp.188–197.

Shivaratri, N.G., Krueger, P. and Singhal, M. (1992) 'Load distributing for locally distributed systems', *IEEE Computer,* December, pp.33–44.

Sinha, P.K. (1997) *Distributed Operating Systems: Concepts and Design*, IEEE Institute of Electrical and Electronics Engineers, Inc, 345 East 47th Street, NY 10017–2394.

Theimer, M.M. and Lantz, K.A. (1988) 'Finding idle machines in a workstation-based distributed systems', *Proc. IEEE 8th Int. Conf. on Distributed Computing Systems*, pp.112–122.

Woodside, C.M. and Monforton, G.G. (1993) 'Fast allocation of processes in distributed and parallel systems', *IEEE Transactions on Parallel and Distributed Systems*, February, Vol. 4, No. 2, pp.164–174.

Zhang, Y. (2001) 'Impact of workload and system parameters on next generation cluster scheduling mechanisms', *IEEE Transactions on Parallel and Distributed Systems*, September, Vol. 12, No. 9, pp.967–985.

Zhou, S. and Ferrari, D. (1987) *An Experimental Study of Load Balancing Performance*, Report No. UCB/CSD 87/336, Progress Report No. 86.8, Computer Science Division (EECS), University of California, Berkeley, California 94720, January, pp.1–27.

Zomaya, A.Y. and Teh, Y. (2001) 'Observations on using genetic algorithms for dynamic load-balancing', *IEEE Transactions on Parallel and Distributed Systems*, September, Vol. 12, No. 9, pp.899–911.