

# A New Migration Model based on the Evaluation of Processes Load and Lifetime on Heterogeneous Computing Environments

Rodrigo Fernandes de Mello  
Universidade de São Paulo  
Instituto de Ciências Matemáticas e de  
Computação Caixa Postal 668  
13560-970 São Carlos – SP  
mello@icmc.usp.br

Luciano José Senger  
Universidade Estadual de Ponta Grossa  
Departamento de Informática  
Av. Carlos Cavalcanti, 4748  
84030-900 Ponta Grossa – PR  
ljsenger@icmc.usp.br

## Abstract

*This paper presents a new model for evaluation of the positive and negative impacts related to the process migration on environments composed by heterogeneous capacity computers. On this model, a busy computer analyzes the occupation of each process and selects the more adequate for migration. The analysis and selection are done through a migration factor. This factor reflects how much the busy computer will be freed and how much the destination computer will be overloaded, in view of the migration of each process. The migrated processes are the ones that present migration factors to enhance the environment load balancing. The results from the carried out experiments have showed this model contributions when compared to related work. The main contribution is the decrease in processes average response time, which means higher performance.*

**Keywords:** process migration, load balancing, high performance computing.

## 1. Introduction

The availability of low cost microprocessors and the evolution of the computer networks have made economically feasible the development of distributed systems based on distributed memory multicomputers. On such systems, the processes execute on the computer network, communicating one each other to accomplish a same computing task. The distribution of such processes may be done manually or automatically, using a load balancing algorithm.

Load balancing algorithms try to distribute equally the process load among all the system computers. Krueger and Livny [11] show that load balancing methods reduce the average and standard deviations in the process response times.

Low response times show shorter process execution times and, consequently, higher system performance.

Load balancing algorithms involve the following policies: transference, selection, location and information [18, 21]. The transference policy classifies a computer as a process sender or receiver, according to its occupation. The selection policy defines which process should be transferred from the busiest computer to the idlest one. The location policy defines which computer should receive or send a process for transference. A server computer offers processes when it is overloaded; a receiver computer requests processes when it is idle. The information policy defines when and how the information on the computers' occupation is updated in the system. Such information is used by the location policy.

Work related to load balancing have been proposed for environments composed of homogeneous capacity computers [21, 20, 10] and heterogeneous capacity ones [14, 13, 16]. Such work do not evaluate the costs and impacts related to the process migration. The migration costs are related to the transference operation of the process context over the communication network and the re-establishment of communication channels existing among the processes (like open files, sockets etc). Such costs become more significant as the processes increase the communication and the network becomes a bottleneck, which is an usual situation on parallel applications. The migration impacts are related to the effect on the system performance caused by process migrations carried out without a previous analysis of the load situation of the processing elements. This impact increases when migration algorithms transfer loads to also busy processing elements, which, on their turn, tend to transfer the load to other processing elements of the network.

The limitations of the previous related work have motivated the creation of a model for evaluation of the benefits and impacts caused by process migration on environments

composed by heterogeneous capacity computers. Halchor-Balter and Downey [9] have carried out some studies on the subject and proposed a migration cost model based on the process aging. This study concludes that there are benefits on the migration of process with a long lifetime. However, this study is incomplete on its evaluation, as besides the evaluation of the process execution time, it is necessary to evaluate the load that such processes impose over the system. Processes with long execution time but with low occupation rate, when migrated do not aggregate benefits to the server computer. Moreover, these processes accumulate a migration cost increase on their execution time.

Observing these limitations, this paper presents an evaluation model for the positive and negative impacts related to load transference conducted through the process migration. This model evaluates three aspects: migration cost, imposed load and process lifetime. The experiments using a simulator of this model showed the contributions on environment load homogenization and on higher applications performance. The remain of this paper is organized as follows: 2) The new migration model based on knowledge about process lifetime and load; 3) Decision-making using the migration factor model; 4) Simulation results; 5) Conclusions and References.

## 2. A New Migration Model Based on Knowledge about Process Lifetime and Load

Zhou and Ferrari have studied four sender-initiated load balancing algorithms: Disted, Global, Central and Lowest. Out of the studied algorithms, it was concluded that the Lowest shows the best balancing results, as it minimizes the number of messages on the network and provides high choice probability of a good destination for processes of a busy computer. In order to make this choice, the Lowest sends out messages for a group of computers and selects the idlest one [21].

Shivaratri *et al.* have evaluated the following kinds of load balancing algorithms: receiver-initiated, sender-initiated, symmetrically-initiated, stable symmetrically-initiated [18]. Out of these studies, it was concluded that the stable symmetrically-initiated shows better results than the remainder ones. Such studies have added significant contributions to the studies made by Zhou and Ferrari [21].

Mello *et al.* have observed the limitations of the previous algorithms and proposed a new load balancing algorithm for scalable heterogeneous environments [14, 13]. The expressive results of this algorithm have been evidenced through simulations and, later on, confirmed by a prototype running on the Linux operating system.

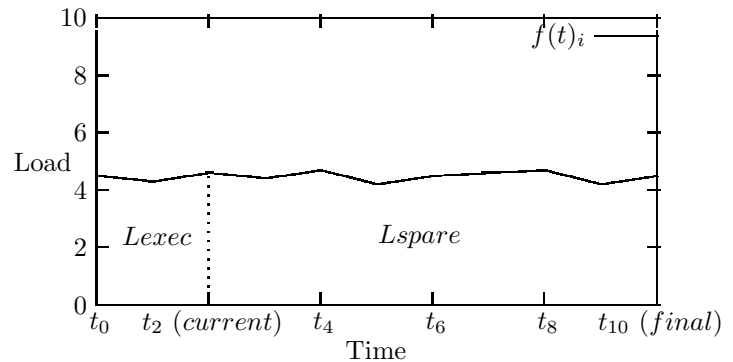
These studies are oriented to the optimization of the load balancing and do not evaluate the costs and impacts

of the process migration. Such limitations have motivated the creation of a new model for the evaluation of benefits and impacts caused by the process migration on environments composed by heterogeneous capacity computers. Some studies on the subject have been proposed and one of the most relevant of them was the one by Halchor-Balter and Downey [9].

Halchor-Balter and Downey [9] have presented a migration cost model based on the process aging. Such study concludes that there are benefits on migration of processes with a long lifetime, that is, processes that execute during long periods of time. However, these results are incomplete on their evaluation, as besides the evaluation of the process execution time, it is necessary to evaluate the load that such processes impose over the system. Processes with long execution time, but with low occupation rate, when migrated do not aggregate benefits to the sender computer. Moreover, these processes accumulate a migration cost increase on their execution time.

The limitations on the work by Halchor-Balter and Downey have motivated the creation of a process transference model that evaluates three issues: the migration cost, the imposed load and the process lifetime. On this model, each process  $P_i$  imposes, during its whole execution, a total load  $L_i$  over the computer that executes it. This total load is characterized by a function  $f(t)$  that varies during the process lifetime.

The Figure 1 depicts the load of a certain process during its execution.



**Figure 1. Example of a load imposed by an executing process**

The model proposed in this paper uses this function to define some of its parameters. The first parameter is the total load  $L_{total,i}$  of a process  $P_i$  (eq. 1), which is obtained through the function graph integration, from the instant  $t_0$

to  $t_{final}$ . The second parameter allows to quantify the process load  $P_i$  already executed, and for this purpose it should be integrated from the beginning of its execution, at the instant  $t_0$ , until its current age at the instant  $t_{cur}$  (eq. 2). The third parameter is the spare load, that is the load which will be executed to complete the process (eq. 3). This parameter is obtained through the integration of the current instant  $t_{cur}$  until the  $t_{final}$ , which represents the end of a process execution. In order to obtain the  $f(t)$  function, load measurements are made during the process execution and the model for predicting the process lifetime proposed by Senger *et al.* [17] is also used. This model allows the prediction of the process execution time using an instance-based learning algorithm.

$$L_{total,i} = \int_0^{final} f(t)_i dt_i \quad (1)$$

$$L_{executed,i} = \int_0^{cur} f(t)_i dt_i \quad (2)$$

$$L_{spare,i} = \int_0^{final} f(t)_i dt_i - \int_0^{cur} f(t)_i dt_i = \int_{cur}^{final} f(t)_i dt_i \quad (3)$$

Besides the parameters obtained by the  $f(t)$  function, the model proposed on this paper requires other parameters. One of them is related to the characterization of the environment computers' load, which is obtained through the eq. 4, where is made a sum of the spare loads of each process at the instant  $t_{cur}$ . Other parameter used is the process migration cost provided by the eq. 5, where:  $MC_{fixed}$  represents the process fixed cost to save its execution state (registers);  $MC_{mtr}$  represents the cost to transfer the process memory to the target computer, presented on the eq. 6;  $MC_{chr}$  represents the cost to restart the access communication channels to the hard disk and network. The eq. 6 used by the eq. 5 is composed by:  $P_{memory,i}$  that represents the quantity of memory used (in bytes) by the process  $P_i$ ;  $CAP_{nb}$  represents the bandwidth, or the data transference capacity, between the computers that send and receive the load.

$$CL_{cur} = \sum_{p=0}^{max} \int_{cur}^{final_p} f(t)_p dt_p \quad (4)$$

$$MC_i = MC_{fixed} + MC_{mtr,i} + MC_{chr,i} \quad (5)$$

$$MC_{mtr,i} = \frac{P_{memory,i}}{CAP_{nb}} \quad (6)$$

Using the previously presented equations, this model proposes a migration factor (eq. 8) to evaluate the benefits of process migration. When this factor is equal to 1, the system reaches a perfect balance, as each computer executes a

load equivalent to its execution capacity. If this is not possible, an approximation to this value may be done through the load sum of  $n$  processes whilst the factor  $\leq 1$ . If none process of the busy computer result in a migration factor  $M_{factor}$  equal or lower than 1, then a process that has a factor  $\leq (CAP_{recv} \div CAP_{send})$  may be selected. On this case, the factor is compared to the performance difference, in percentage, between the source (sender) and destination (receiver) computers.

$$N_{spare,i} = L_{spare,i} \times CAP_{send} \quad (7)$$

$$M_{factor,i} = \frac{N_{spare,i}}{CL_{send} - (CL_{recv} + \frac{N_{spare,i}}{CAP_{recv}} + MC_i)} \quad (8)$$

The following section presents examples of how the migration factor may help on the processes transference decision-making.

### 3. Decision-Making Using the Migration Factor

This section presents the decisions that may be taken from the process migration factor values (eq. 8). In order to characterize these decisions, consider two heterogeneous capacity computers  $C_1$  and  $C_2$ . The computer  $C_1$  has capacity of 1000 MIPS and the computer  $C_2$  has capacity of 500 MIPS. The computer  $C_1$  has 3 processes in execution with respective spare loads of  $P_1 = 5$ ,  $P_2 = 2$  and  $P_3 = 1$ , being the total computer load given by the equation eq. 4, the  $C_1$  load is equal to 8. The computer  $C_2$  has a process  $P_1$  with spare load equal to 4, being the load  $C_2 = 4$ . Consider that the computer  $C_1$  requires a load transference, as it observes its high load status compared to the other computers in the environment. On this moment, the  $C_1$  processes are sorted by the load and their migration factors are obtained according to table 1.

Process	Migration factor
$P_1$	$\frac{5}{8 - (4 + ((5 \times 1000) \div 500))} = -0.83$
$P_2$	$\frac{2}{8 - (4 + ((2 \times 1000) \div 500))} = \lim_{x \rightarrow 0} \frac{2}{x} = \infty$
$P_3$	$\frac{1}{8 - (4 + ((1 \times 1000) \div 500))} = 0.5$

**Table 1. Process migration factors of  $C_1$**

Observing the table 1, we may conclude that the migration factor of process  $P_1$  is equal to  $-0.83$ . On this case,

note that computer  $C_1$  will have a load equal to 3 after the migration and computer  $C_2$  is equal to 14. This unbalancing is not desired, since any negative value on the migration factor represents a performance degradation on the process receiver computer. For process  $P_2$ ,  $M_{factor} = \infty$ , which means that the receiving computer load will be the same current load of the process sender computer. On this case, the migration does not bring any benefit to the environment, only a load transference will be done. For process  $P_3$ ,  $M_{factor} = 0.5$ , which means there are benefits with its migration as there will be a better load balancing and each computer should execute loads based on their processing capacity.

There will be benefits whenever  $0 < M_{factor,i} \leq (CAP_{recv} \div CAP_{send})$ , that is, whenever the migration factor does not damage the balancing or overloads the receiver computer (when  $M_{factor} \leq 0$ ) or when there is only one load transference with no benefits for the environment (when  $M_{factor,i} > (CAP_{recv} \div CAP_{send})$ ).

The migration factor (eq. 8) may be extended to enhance the load balancing through the  $k$  process migration (eq. 9). For instance, consider a situation where a computer presents several  $P_i$  processes where  $i = 1, 2, 3, \dots, n$ . On this case, the migration factors of  $k$  processes may be summed whilst  $\leq (CAP_{recv} \div CAP_{send})$  and migrated to a set of idle computers.

$$\sum_{p=0}^k M_{factor,i} \leq \frac{CAP_{recv}}{CAP_{send}} \quad (9)$$

#### 4. Simulation Results

In order to evaluate the proposed model a simulator was implemented. This simulator<sup>1</sup> is parameterized to the CPU and memory occupations that each process causes on the system and to the capacity of each environment computer. The CPU occupation by the processes is defined as the total required number of MIPS (million of instructions per second) to process them. The memory occupation is defined in bytes. Each computer capacity is defined in MIPS.

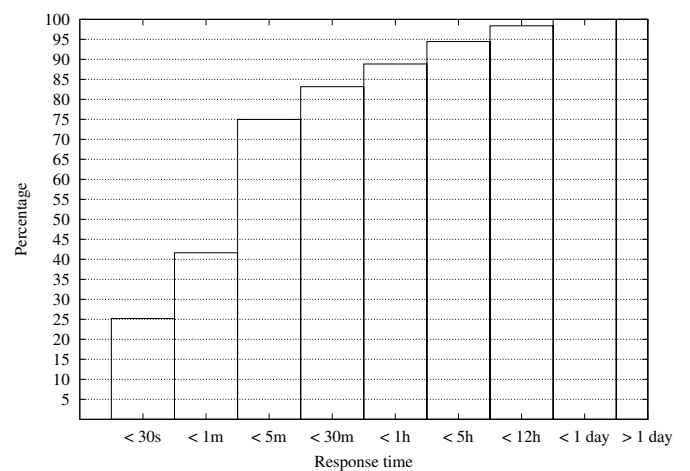
The system process arrival is defined by a probability distribution function. After the arrival of such processes to the system, they are submitted to the computers through a normal distribution function. The process arrival distribution function to the system and its parameters may be defined by the user. The available functions are the ones supported by the PSOL library (The Probability/Statistics Object Library)<sup>2</sup> adopted by the simulator. The simulator gen-

erates results on the process average response time, in seconds.

The Feitelson's workload model [5, 7, 6, 12, 8, 19, 3, 15, 4] is used. This model is based on the analysis of six execution traces of the following production environments:

- 128-node iPSC/860 at NASA Ames;
- 128-node IBM SP1 at Argonne;
- 400-node Paragon at SDSC;
- 126-node Butterfly at LLNL;
- 512-node IBM SP2 at CTC;
- 96-node Paragon at ETH, Zurich.

The Figure 2 illustrates the histogram of the accumulated execution time generated for this model.



**Figure 2. Feitelson's model accumulated execution time histogram**

Besides synthesizing the loads imposed by the applications, this load model defines the process arrival times to the environment, based on the Poisson probability distribution with average 1,500 seconds [5], which was also adopted in the experiments. The simulated environment is composed by 10 computers with CPU capacity (in MIPS) and memory capacity (in MBytes) presented on table 2.

The simulation results are presented on figure 3, which evidence the benefits of the migration factor model over the migration model by process aging and over the environment with no migration at all.

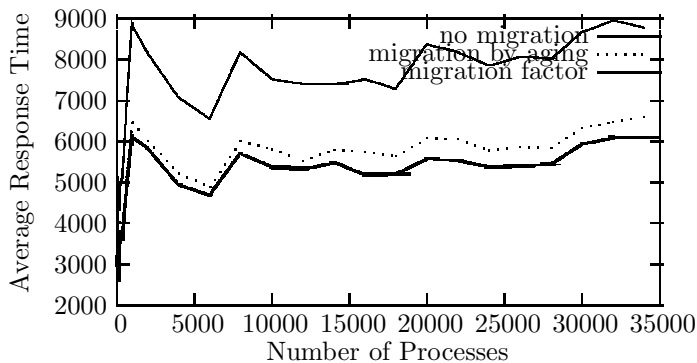
For a better understanding, linear regressions were made with a confidence interval of 0.99, which are presented on table 3, where  $x$  represents the quantity of processes that arrive to system according to the Feitelson's load model [5]

<sup>1</sup> Simulator available on <http://www.icmc.usp.br/~mello/outr.html>

<sup>2</sup> PSOL [2] is a statistic library licensed under GNU/GPL [1] and developed by the University of Alabama.

Computer	CPU (MIPS)	Memory (MBytes)
$C_1$	500	64
$C_2$	600	64
$C_3$	700	64
$C_4$	800	64
$C_5$	900	128
$C_6$	1000	128
$C_7$	1100	256
$C_8$	1200	256
$C_9$	1300	512
$C_{10}$	1400	512

**Table 2. Capacity of the Simulated Environment Computers**



**Figure 3. Results for Feitelson's workload model**

and  $y$  represents the average process response time. Such regressions allow the precise quantification of the contributions provided by the migration factor model. This model presents better results than the process aging model [9], as observed on figure 3. However, analyzing the regressions we may conclude that, in cases where more than 205,666.73 arrive to the system, the process aging migration model starts to behave better. This high process arrival rate to the system does not represent usual loads for this kind of system, as it may be noted by Feitelson's execution traces.

## 5. Conclusions

This paper has presented a new model for evaluation of the process migration on environments composed by heterogeneous capacity computers. This model analyzes pro-

Migration Model	Linear Regression	$R^2$
No migration	$y = 0.0950x + 5902.82$	0.4341
Mig. by Aging	$y = 0.0518x + 4807.20$	0.4606
Mig. Factor	$y = 0.0541x + 4334.17$	0.3948

**Table 3. Capacity of the simulated environment computers**

cesses occupation and selects the more adequate for migration. Such analysis and selection are done through a migration factor. This factor reflects how much the busy computer will be freed and how much the destination computer will be loaded in view of each process migration.

Experiments have been carried out by using a simulator. Such experiments evidenced that this model shows excellent results, as it significantly decreases the process average response times when compared to the process aging migration model [9] and to an environment with no migration at all. Linear regressions were carried out, which evidenced that migration factor model can only be surpassed by the process aging model on environments where the process arrival rate is higher than 205,666.73 for a Poisson distribution of process arrival with average 1,500 seconds. However, as it may be observed through the Feitelson's execution traces [5], this does not reflect the load behavior on such environments. Therefore, for the remainder cases, it is recommended the adoption of the process migration factor, which accomplish better results.

## 6. Acknowledgments

The authors thank to Capes and Fapesp Brazilian Foundations (under the process number 04/02411-9).

## References

- [1] Gnu public license. Available at <http://www.gnu.org>.
- [2] The probability/statistics object library. Available at <http://www.math.uah.edu/psol>.
- [3] K. aida. Effect of job size characteristics on job scheduling performance. In D. G. Feitelson and L. Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, pages 1–17. Springer, 2000. Lect. Notes Comput. Sci. vol. 1911.
- [4] D. G. Feitelson. Metrics for parallel job scheduling and their convergence. In D. G. Feitelson and L. Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, pages 188–205. Springer, 2001. Lect. Notes Comput. Sci. vol. 2221.
- [5] D. G. Feitelson and M. A. Jette. Improved utilization and responsiveness with gang scheduling. In D. G. Feitelson and L. Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, pages 238–261. Springer, 1997. Lect. Notes Comput. Sci. vol. 1291.

- [6] D. G. Feitelson and L. Rudolph. Metrics and benchmarking for parallel job scheduling. In D. G. Feitelson and L. Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, pages 1–24. Springer, 1998. Lect. Notes Comput. Sci. vol. 1459.
- [7] D. G. Feitelson and A. M. Weil. Utilization and predictability in scheduling the ibm sp2 with backfilling. In *12th Intl. Parallel Processing Symp.*, pages 542–546, Apr 1998.
- [8] G. Ghare and S. T. Leutenegger. The effect of correlating quantum allocation and job size for gang scheduling. In D. G. Feitelson and L. Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, pages 91–110. Springer, 1999. Lect. Notes Comput. Sci. vol. 1659.
- [9] M. Harchol-Balter and A. B. Downey. Exploiting Process Lifetimes Distributions for Dynamic Load Balancing. *ACM Transactions on Computer Systems*, 15(3):253–285, August 1997.
- [10] C. Hui and S. T. Chanson. Hydrodynamic load balancing. *IEEE Transactions on Parallel and Distributed Systems*, 10(11):1118–1137, nov. 1999.
- [11] P. Krueger and M. Livny. The diverse objectives of distributed scheduling policies. In *Seventh Int. Conf. Distributed Computing Systems*, pages 242–249, Los Alamitos, California, 1987. IEEE CS Press.
- [12] V. Lo, J. Mache, and K. Windisch. A comparative study of real workload traces and synthetic workload models for parallel job scheduling. In D. G. Feitelson and L. Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, pages 25–46. Springer, 1998. Lect. Notes Comput. Sci. vol. 1459.
- [13] R. F. Mello, L. C. Trevelin, M. S. Paiva, and L. T. Yang. Comparative analysis of the prototype and the simulator of a new load balancing algorithm for heterogeneous computing environments. In *International Journal Of High Performance Computing And Networking, ISSN 1740-0562*. Interscience, 2004.
- [14] R. F. Mello, L. C. Trevelin, M. S. Paiva, and L. T. Yang. Comparative study of the server-initiated lowest algorithm using a load balancing index based on the process behavior for heterogeneous environment. In *Networks, Software Tools and Application, ISSN 1386-7857*. Kluwer, 2004.
- [15] A. W. Mu’alem and D. G. Feitelson. Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfilling. *IEEE Trans. Parallel & Distributed Syst.*, 12(6):529–543, Jun 2001.
- [16] M. Neeracher. *Scheduling for Heterogeneous Opportunistic Workstation Clusters*. PhD thesis, Swiss Federal Institute of Technology, Zurich, 1999.
- [17] L. J. Senger, M. J. Santana, and R. H. C. Santana. Using runtime measurements and historical traces for acquiring knowledge in parallel applications. In *International Conference on Computational Science (ICCS) (to appear)*. Springer Lect. Notes Comput. Sci., 2004.
- [18] N. G. Shivaratri, P. Krueger, and M. Singhal. Load distributing for locally distributed systems. *IEEE Computer*, 25(12):33–44, 1992.
- [19] D. Talby, D. G. Feitelson, and A. Raveh. Comparing logs and models of parallel workloads using the co-plot method. In D. G. Feitelson and L. Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, pages 43–66. Springer, 1999. Lect. Notes Comput. Sci. vol. 1659.
- [20] M. M. Theimer and K. A. Lantz. Finding idle machines in a workstation-based distributed system. *IEEE Transactions on Software Engineering*, 15(11):1444–1458, nov. 1989.
- [21] S. Zhou and D. Ferrari. An experimental study of load balancing performance. Technical Report UCB/CSD 87/336, PROGRES Report N.o 86.8, Computer Science Division (EECS), Universidade da California, Berkeley, California 94720, jan. 1987.