Model for Simulation of Heterogeneous High-Performance Computing Environments

Rodrigo Fernandes de Mello[1] and Luciano José Senger[2]*

[1] Universidade de São Paulo – Departamento de Computação
Instituto de Ciências Matematicas e de Computação
Av. Trabalhador Saocarlense, 400 Caixa Postal 668
CEP 13560-970 São Carlos, SP, Brazil
mello@icmc.usp.br

[2] Universidade Estadual de Ponta Grossa – Departamento de Informatica
Av. Carlos Cavalcanti, 4748
CEP 84030-900 Ponta Grossa, PR, Brazil
ljsenger@icmc.usp.br

Abstract. This paper proposes a new model to predict the process execution behavior on heterogeneous multicomputing environments. This model considers the process execution costs such as processing, hard disk acessing, message transmitting and memory allocation. A simulator of this model was developed which help to predict the execution behavior of processes on distributed environments under different scheduling techniques. Besides the simulator, it was developed a suite of benchmark tools in order to parameterize the proposed model with data collected from real environments. Experiments were conduced to evaluate the proposed model which used a parallel application executing on a heterogeneous system. The obtained results show the model ability to predict the actual system performance, providing an useful model for developing and evaluating techniques for scheduling and resource allocation over heterogeneous and distributed systems.

1 Introduction

The evaluation of a computing system allows the analysis of its technical and economic feasibility, safety, performance and correct execution of processes. In order to evaluate a system, techniques that estimate its behavior on different situations are used. Such techniques provide numerical results which allow the comparison among different solutions for the same problem [1]. The evaluation of a computing system may use elementary or indirect techniques. The elementary ones are directly applied over the system, so it is necessary to have it previously implemented. The indirect ones allow the system evaluation before its implementation, what is relevant at the project phase [2–6].

The indirect techniques use mathematic models to represent the behavior of the main system components. Such models should be as similar as possible to the real problems, generating results for a good evaluation without being necessary to implement

* The authors thank to William Voorsluys for improving the source code of the benchmark *memo* and the fundings from Capes and Fapesp Brazilian Foundations (under the process number 04/02411-9).

them [6]. Several models have been proposed for the evaluation of the execution time and the process delay. They consider the CPU consumption, the performance slowdown due to the use of the virtual memory [7] and the time spent with messages transmitted through the communication network [8].

Amir *et al.* [7] have proposed a method for job assignment and reassignment on cluster computing. This method uses a queuing network model to represent the slowdown caused by virtual memory usage. In such model the static memory $m(j)$ used by the process is known. This model defines the load of each computer in accordance with the equation 1, where: $L(t, i)$ is the load of computer i at the instant t; $l_c(t, i)$ is the CPU occupation; $l_w(t, i)$ is the amount of main memory used; $r_w(i)$ is maximum capacity of the main memory; β is the slowdown factor due to the use of virtual memory. Such factor increases the process response time, what consequently reflects in a lower final performance. This work attempts to minimize the slowdown by means of scheduling operations.

$$L(t,i) = \begin{cases} l_c(t,i) & \text{if } l_w(t,i) \leq r_w(i) \\ l_c(t,i) * \beta & \text{otherwise} \end{cases} \tag{1}$$

Mello *et al.* [9] have proposed improvements to the slowdown model by Amir *et al.* [7]. This work includes new parameters which allow a better modelling of process slowdown. Such parameters are the capacity of CPU and memory, throughput for reading and writing on hard disk and delays generated by the use of the communication network. However, this model presents similar limitations to the work by Amir *et al.* [7], as it does not offer any resource to model, through equations, the delay caused by the use of virtual memory (represented in equation 1 by the parameter β), nor consider other delays of the process execution time generated by: message transmission, hard disk access and other input/output operations. The modeling of message transmission delays is covered by other works [8, 10].

Culler *et al.* [8] have proposed the LogP model to quantify the overhead and the network communication latency among processes. The overhead and latency cause delays among processes which communicate. This model is composed of the following parameters: L which represents the high latency limit or delay incurred in transmitting a message containing a word (or a small number of words) from the source computer to a destination; o represents the overhead which is the time spent by processor to prepare a message for sending or receiving; g is the minimum time interval between consecutive message transmittion (sending or receiving); P is the number of processors. The LogP model assumes a finite capacity network with the maximum transmission defined by L/g messages.

Sivasubramaniam [10] used the LogP model to propose a framework to quantify the overhead of parallel applications. In such framework are considered aspects such as the processing capacity and the communication system usage. This framework joins efforts of actual experiments and simulations to refine and define analytic models. The major limitation of this work is that it does not present a complete case study.

The LogP model can be aggregated to the model by Amir *et al.* [7] and Mello *et al.* [9], permitting to evaluate the process execution time and slowdowns considering the resources of CPU, memory and transmitted messages on the network. Although unifying the models, they are still incomplete because do not consider the spatial and message generation probability distributions. Motivated by such limitations, some studies have been proposed [11, 12].

Chodnekar *et al.* [11] have presented a study to characterize the probability distribution of messages on communication systems. In such work, the 1D-FFT and IS [13], Cholesky and Nbody [14], Maxflow [15], 3D-FFT and MG [16] parallel applications are evaluated executing on real environments. In the experiments, some informations have been captured such as the message sending and receiving moments, size of messages and destination. These informations were analyzed through statistic tools, and the spatial and message generation probability distributions obtained. The spatial distribution defines the frequency each process communicates with others. The message generation distribution defines the probability that each process sends messages to others.

They have concluded that the most usual message generation probability distribution for parallel applications are the exponential, hyperexponential and Weibull. It has also been concluded that the spatial distribution is not uniform and there are different traffic patterns during the applications' execution. In the most part of applications there is a process which receives and sends a large number of messages to the remainder processes (like a master for PVM – Parallel Virtual Machine – and MPI – Message Passing Interface – applications). The work also presents some features about message volume distribution, but there is not a precise analysis about the message size, overhead and latency.

Vetter and Mueller [12] have studied the communication behavior of scientific applications using MPI (Message Passing Interface). This study quantifies the average volume of transmitted messages and their size. It has been concluded that in peer-to-peer systems 99% of the transmitted messages vary from 4 to 16384 bytes. In collective calls this number varies from 2 to 256 bytes. This was combined with the studies on spatial and message generation distributions by Chodnekar *et al.* in [11] and to the LogP model [8] which allow the identification of overhead and communication latency in computing systems. By unifying these studies to the previously described slowdown models it is possible to evaluate the process behavior considering CPU, virtual memory and message transmittion. However, it is not possible to model voluntary delays in the execution of processes (generated by *sleep* calls) and accesses to hard disks.

Motivated by the unification of the previously presented models, the aggregation of the applications' voluntary delays and hard disk access, this paper presents the UniMPP (*Unified Modeling for Predicting Performance*) model. This model unifies the CPU consumption considered in the models by Amir *et al.* [7] and Mello *et al.* [9], the time spent to transmit messages modeled by Culler *et al.* [8] and Sivasubramaniam [13], the message volume and the spatial and message generation probability distributions by Chodnekar *et al.* [11], and Vetter and Mueller [12]. Experiments confirmed that this model can be used to predict the behavior of process execution on heterogeneous environments, once it generates the process response times very similar to the observed on real executions.

This model was implemented in a simulator which is parameterized with system configurations (CPUs, main and virtual memories, hard disk thoughput and network capacity) and receives processes for execution. Distribution functions are used to characterize the process CPU, memory, hard disk and network occupations. The simulator also generates new processes according to a probability distribution function, allowing to evaluate different scheduling and load balancing policies without needing the real execution.

As presented before, the simulator needs to be parameterized with the actual system configurations. For this purpose, a suite of benchmark tools was developed to collect informations such as the capacity of CPUs in MIPS (millions of instructions per second), the main and virtual memory behavior under a progressive occupation (this generates delay functions), the hard disk throughput in reading and writing operations (in MBytes per second) and the network delay (considering the overhead and latency in seconds).

The main contribution of this work is the UniMPP model which can be used with the simulator allied to the benchmark tools to predict the process execution time on heterogeneous environments. The simulator is prepared to receive new scheduling and load balancing policies and evaluate them using different workload models [17].

This paper is divided into the following sections: 2 The model; 3 Parameterization; 4 Model Validation; 5 Conclusions and References.

2 The Model

Motivated by the unification of the virtual memory slowdown models [7,9], by the models of delays in process execution caused by messages transmission [8, 10], by studies about spatial and message generation probability distributions [11], by the slowdown caused in main and virtual memory ccupation, by the definition of voluntary delay and access to hard disks, the UniMPP (*Unified Modeling for Predicting Performance*) model has been designed. These models are presented in the previous section. Unifying the ideas of each model and adding voluntary delays and hard disk access, we have defined a new model to predict the execution behavior of processes running on heterogeneous computers. By using this model, researchers can evalutate different techniques such as scheduling and load balancing without being necessary to run an application on an real environment.

In this model, a process p_j arrives at the system, following a probability distribution function, at the instant a_j. Such process is started by the computer c_i. Each computer maintains its queue $q_{i,t}$ of processes at the instant t. In this model, every computer c_i is composed of the sextuple $\{pc_i, mm_i, vm_i, dr_i, dw_i, lo_i\}$, where: pc_i is the total computing capacity of each computer measured in instructions per unit of time; mm_i is the total main memory; vm_i is the total virtual memory capacity; dr_i is the hard disk reading throughput; dw_i is the hard disk writing throughput; lo_i is the time spend in sending and receiving messages.

In the UniMPP, each process is represented by the sextuple $\{mp_j, sm_j, pdfdm_j, pdfdr_j, pdfdw_j, pdfnet_j\}$, where: mp_j represents the processing consumption; sm_j is the amount of static memory allocated by the process; $pdfdm_j$ is the probability distribution function used to represent the dynamic memory occupation; $pdfdr_j$ is the probability distribution function used to represent the hard disk reading; $pdfdw_j$ is the probability distribution function used to represent the hard disk writing; $pdfnet_j$ is the probability distribution function used to represent the sending and receiving operations on communication system.

Having formally defined computers and processes, equations were defined to obtain the process response time and delay. The first equation (equation 2) presents the response time (TE_{p_j,c_i}) of a process p_j being executed in a computer c_i, where the total computing capacity pc_i of c_i and the processing consumption of p_j should be represented by the same metric, such as MI (millions of instructions when the capacity of processors was obtained in Mips – Millions of instructions per second) or MF (millions of float-point instructions when the capacity of processors was obtained in Mflops – Millions of float-point instructions per second).

$$TE_{p_j,c_i} = \frac{mp_j}{pc_i} \tag{2}$$

The equation 2 presents a calculation method for the execution time of a process under ideal conditions, in which there is no competition nor delays caused by the memory and input/output usage. The work by Amir *et al.* [7] presents a more adequate equation in which, from the moment that the virtual memory starts to be used, there is a delay in the process execution. These authors use a constant delay in their equations. However, by using the benchmark tools described in section 3, it was observed that there are limitations in their model, since the performance slowdown is linear during the main memory usage and exponential from the moment the virtual memory starts to be used.

$$TEM_{p_j,c_i} = TE_{p_j,c_i} * (1 + \alpha) \tag{3}$$

The Amir's performance model does not consider this linear performance slowdown caused by the use of the main memory and considers a constant factor for the performance slowdown caused by the use of the virtual memory when, in fact, this slowdown is exponential. The UniMPP models the process performance slowdown generated by the use of main and virtual memories, by the equation 3, where α represents a percentage obtained from a delay function and TE is presented in equation 2. This delay function is generated by a benchmark tool (section 3) where in the $x-axis$ is the memory occupation up to use all the virtual memory and in the $y-axis$ is the α value (the slowdown imposed in the process execution by the memory occupation).

A model which considers the process execution slowdown caused by the use of main and virtual memories become more adequate, however, it does not allow the precise quantification of the total execution time of processes which perform input and output operations to the hard disk. For this reason, experiments have been conduced and equations developed to measure the delays generated by accesses to hard disk. The equation 4 models the process delay generated by reading operations from hard disk, where: nr represents the number of reading accesses; $bsize$ represents the data buffer size; dr_i represents the throughput capacity for reading accesses from hard disk; and $wtdr_k$ represents the waiting time for using the resource.

$$SLDR_{p_j,c_i} = \sum_{k=1}^{nr} \frac{bsize_k}{dr_i} + wtdr_k \tag{4}$$

The hard disk writing delay is defined by equation 5, where: nw represents the number of writing accesses; $bsize$ represents the data buffer size to be written; dw_i is the throughput capacity for writing accesses in hard disk; $wtdw$ is the waiting time for using the resource.

$$SLDW_{p_j,c_i} = \sum_{k=1}^{nw} \frac{bsize_k}{dw_i} + wtdw_k \tag{5}$$

In addition to the delays caused by memory usage and input/output to hard disks, there are delays generated by sending and receiving messages on communication systems. Such delays vary according to the network bandwidth, latency and overhead of communication protocols [18–20]. The protocol latency involves the transmission time on communication system, which vary in accordance with the message size and control messages generated by the protocol [18–20]. The protocol overhead is the time involved for packing and unpacking messages for transmission. This time also varies according to the messages size [18–20]. The delay for sending and receiving messages is defined by equation 6, where: nm represents the number of sent and received messages; $\theta_{s,k}$, described in equation 7 is the time used for sending and receiving messages on communication system, not considering the wait for resources; and wtn_k represents the wait time, the queue time, to send or receive a message, when the resource is busy. The components of equation 7 are: $o_{s,k}$ overhead, which when multiplied by two allows the quantification of packing time (by the sender) and the unpacking time (by the receiver) of a message; and $l_{s,k}$ is the latency to transmit a message.

$$SLN_{p_j,c_i} = \sum_{k=1}^{nm} \theta_{s,k} + wtn_k \tag{6}$$

$$\theta_{s,k} = 2 * o_{s,k} + l_{s,k} \tag{7}$$

Aiming the unification of all previously described delay models, it is proposed the equation 8, which allows the definition of the response time (the prediction of this time in a real enviroment) of a process p_j in a computer c_i, where: lz is the process voluntary delay generated by the system calls *sleep*. In the case of load transference (that is, process migration) the communication channels may modify their behaviors and perform a higher or lower number of input/output operations (a process migrating to a computer where there are others which it communicates, reduces the latency and overhead because does not use the communication system, although it can overload the CPU). The equation 9 is the response time of a process p_j transferred among n computers.

$$\begin{aligned} SL_{p_j} = SL_{pj,c_i} = {} & TEM_{p_j,c_i} + SLDR_{p_j,c_i} + \\ & SLDW_{p_j,c_i} + SLN_{p_j,c_i} + \\ & lz \end{aligned} \tag{8}$$

$$SL_{pj} = \sum_{k=1}^{n} SL_{pj,c_k} \tag{9}$$

The UniMPP model unifies the concepts from models by Amir *et al.* [7], Mello *et al.* [9] and Culler *et al.* [8] and extends them by adding voluntary delay equations and the time for reading and writing accesses to hard disks. In addition, based on experiments, this work proposes new equations to define the main and virtual memory slowdown. By these equations, it was observed that the slowdown is linear when using the main memory, and exponential using the virtual. Such experiments were carried though using the benchmark tools from section 3. This model allows studies of scheduling, load balancing algorithms and prediction of process response times on heterogeneous environments.

The proposed model has been implemented in a simulator, named SchedSim [3], which allows other researchers to conduct related studies. Such simulator is implemented in Java language and uses the object oriented concepts that simplify its extension and functionality additions. The simulator is parameterized with system configurations (CPUs, main and virtual memories, hard disk thoughput and network capacity) and receives processes for execution. It generates new processes according to a probability distribution function, allowing to evaluate different scheduling and load balancing policies without needing the real execution.

3 Parameterization

In order to parameterize the SchedSim simulator using real environment characteristics, a suite of benchmark tools[4] was developed. These tools measure the capacity of CPU, reading and writing hard disk throughput and the message transmission delays. Such tools evaluate these characteristics until they reach a minimum sample size based on the central limit theorem, allowing to apply statistical summary measures such as confidence interval, standard deviation and average [21]. This suite is composed by the following tools:

1. `mips`: it measures the capacity of a processor, in millions of instructions per second. This tool uses a `bench()` function implemented by Kerrigan [22];
2. `memo`: it creates child processes until all main and virtual memories are filled up, measuring the delays of the context switches among processes. The child processes only allocate the memory and then sleep for some seconds, thus it does not consider the processor usage;
3. `discio`: it measures the average writing throughput (buffered and unbuffered) and the average reading throughput in local storage devices (hard disks) or remote storage devices (via network file systems);
4. `net` - it is composed of two applications, a customer and a server, which allow the evaluation of the time spent to send and receive messages over communication networks (based on the equation 7).

[3] Source code available at http://www.icmc.usp.br/˜mello/outr.html
[4] Benchmark – source code available at http://www.icmc.usp.br/˜mello/outr.html

4 Validation

In order to validate the proposed model, executions of a parallel application developed in PVM (Parallel Virtual Machine) [23] in a scenario composed of two homogeneous computers have been considered. This adopted application is composed of a master and worker processes. The master process launches one worker on each computer and defines three parameters: the problem size, that is, the number of mathematic operations executed to solve an integral (eq. 10) defined between two points a and b using the trapezium rule [24, 25], the number of bytes that will be transferred over the network and recorded in the hard disk. These workers are composed of four stages: message receiving, processing, writing into the hard disk and message sending. The message exchange happens between master and worker at the beginning and at the end of the workers' execution. The workers are instrumented to account the time consumed in operations.

$$\int_a^b 2 * \sin x + e^x \tag{10}$$

Scenario details are presented on the table 1 and they have been obtained with the benchmark suite. A message size of 32 bytes has been considered for the benchmark *net*. The table 2 presents the slowdown equations generated by using main and virtual memories, respectively, on the computers c_1 and c_2. Such equations have been obtained through the experiments with the benchmark *memo*. The linear format of the equations is used when the main memory is not completely filled up, for instance, in the case of computers c_1 and c_2 not exceed 1 $Gbyte$ of its memory capacity. After exceeding such limit, the virtual memory is used and the delay is represented by the exponential funtion.

Table 1. System details

Resource	c_1	c_2
CPU (Mips)	1145.86	1148.65
Main memory (Mbytes)	$1Gbyte$	$1Gbyte$
Virtual memory (Mbytes)	$1Gbyte$	$1Gbyte$
Disk writing throughput (MBytes/seg)	65.55	66.56
Disk reading throughput (MBytes/seg)	76.28	75.21
Overhead + Latency (seconds)	0.000040	

The experiment results are presented in the table 3. It may be observed that the error among the curves is low, close to zero. Ten experiments have been conduced for different numbers of applications, each one composed of two workers executing on two computers. Such experiment was used to saturate the capacity of all computing resources of the environment. The figure 1 shows the experiment and simulation results.

Table 2. Memory slowdown functions for computers c_1 and c_2

Memory	**Regression**	**Equation**	R^2
Main memory	Linear	$y = 0.0012x - 0.0065$	0.991
Main and Virtual memory	Exponential	$y = 0.0938 * e^{0.0039x}$	0.8898

Table 3. Simulation results for computers c_1 and c_2

Processes	**Actual Average**	**Predicted**	**Error (%)**
10	151.40	149.51	0.012
20	301.05	293.47	0.025
30	447.70	437.46	0.022
40	578.29	573.58	0.008
50	730.84	714.92	0.021
60	856.76	862.52	0.006
70	1002.10	1012.17	0.009
80	1147.44	1165.24	0.015
90	1245.40	1318.37	0.055
100	1396.80	1471.88	0.051

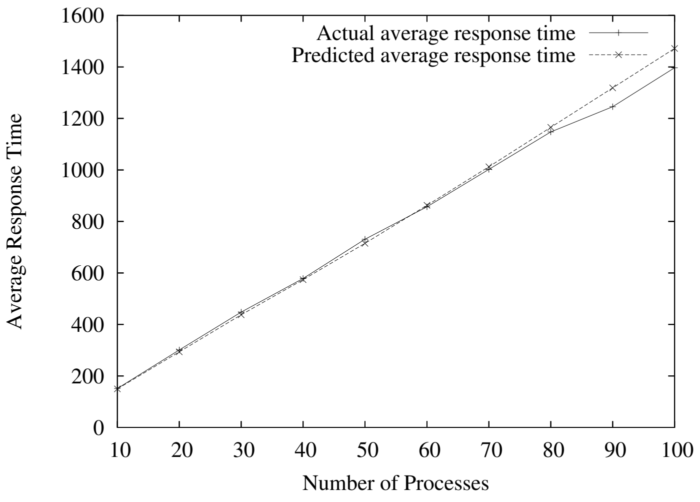

Fig. 1. Actual and predicted average response times for computers c_1 and c_2

The simulation obtained results show the model ability to reproduce the real system behavior. It is important to notice the increasing of the prediction errors when the system runs a number of processes between 90 and 100.

The real executions, using 90 and 100 processes, overloaded the computers and some processes were killed by the PVM system. The premature stopping of processes (at about 5 processes where killed) decreases the computer's load, justifiyng the model prediction error. The simulator was used aiming to predict the system behavior considering a number of processes greater than the number of processes executed by PVM.

After experiments in an homogeneous system, a new environment composed of heterogeneous computers were parameterized using the benchmark tools. In this environment, it was executed the same application, which computes an integral function between two points using the trapezium rule. The features of the heterogeneous computers are presented in the table 4.

Table 4. System details

Resource	c_3	c_4
CPU (Mips)	927.55	1600.40
Main memory (Mbytes)	256	512
Virtual memory (Mbytes)	400	512
Disk write throughput (MBytes/seg)	47.64	15.99
Disk read throughput (MBytes/seg)	41.34	32.55
Overhead + Latency (seconds)	0.000056924	

The tables 5 and 6 present the slowdown equations, obtained by the *memo* benchmarking, considering the main and virtual memory usage.

Table 5. Memory slowdown functions for computer c_3

Memory	**Regression**	**Equation**	R^2
Main memory	Linear	$y = 0.0018x - 0.0007$	0.9998
Main and Virtual memory	Exponential	$y = 0.7335 * e^{0.0097x}$	0.8856

Table 6. Memory slowdown functions for computer c_4

Memory	**Regression**	**Equation**	R^2
Main memory	Linear	$y = 0.0018x - 0.0035$	0.9821
Main and Virtual memory	Exponential	$y = 0.0924 * e^{0.0095x}$	0.8912

The experiment results are presented in the table 7. The error values obtained comparing the simulated and the actual execution time values are close to 0, allowing to confirm the model ability in predicting real executions. The figure 2 shows the experiment and simulation results.

Table 7. Simulation results for computers c_3 and c_4

Processes	Actual Average	Predicted	Error (%)
10	153.29	152.38	0.0059
20	306.63	304.66	0.0064
30	457.93	457.46	0.0010
40	593.66	610.78	0.0280
50	760.02	764.65	0.0060
60	892.29	918.97	0.0290
70	1040.21	1074.18	0.0316
80	1188.14	1230.75	0.0346
90	1333.70	1388.14	0.0392
100	1488.97	1572.22	0.0529

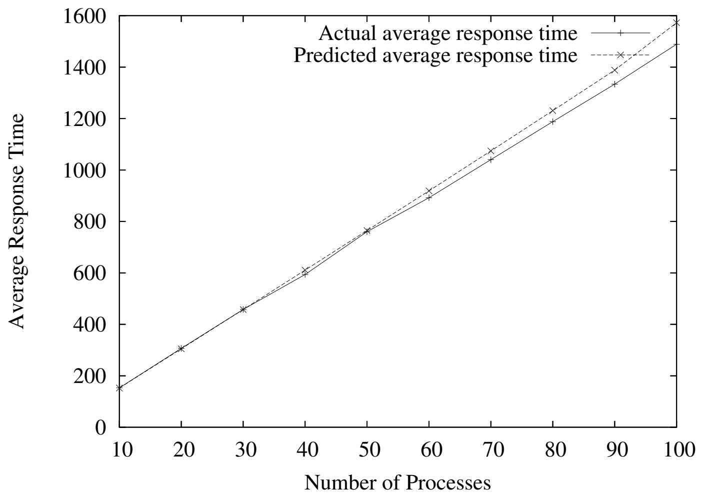

Fig. 2. Actual and predicted average response times for computers c_3 and c_4

When a number at about 60 processes are running, some problems were observed, due to PVM process management. It was observed that using some computers with less processing power, PVM started to kill processes earlier, when running more than 60 processes. These problems explain the difference between the actual and the simulated time values and the increasing in predicting errors.

The experiments presented in this section validate the model used by the simulator. The model and the simulator is able to predict the behavior of a real and dynamic system, modelling distinct parallel applications which solve problems from different areas, such as: aeronautics, fluid dynamics and geoprocessing. Thus, the system behavior can be predicted earlier, in project phase, minimizing the development costs.

5 Conclusions

Several models have been proposed to measure the response time of processes in computing systems [7,9]. Such models have presented some contributions, considering that the virtual memory occupation causes delays in process executions [7,9], as well as delays generated by the message transmissions on communication systems [8,10]. Nevertheless, such models do not unify all possible delays of a process execution.

Motivated by such limitations, this work has presented a new unified model to predict the applications' execution running on heterogeneous distributed envionments. This model considers the process execution time in accordance with the processing, accesses to hard disk, message transmissions on communication networks, main and virtual memory slowdowns.

This work has contributed by modeling the delays in reading and writing accesses to hard disks and presenting a new technique which uses equations to represent the delays generated by the main and virtual memory usage. This has complemented studies by Amir *et al.* [7] and Mello *et al.* [9], which consider a constant delay.

In addition it was developed a simulator of the proposed model which can be used to predict the execution of applications on heterogeneous multicomputing environments. Such simulator has been developed considering extensions such as the design of new scheduling and load balancing policies. This simulator is licensed under GNU/GPL which allows its broad use by the researchers interested in developing and evaluating resource allocation techniques. In order to complement this simulator and allow its parameterization using real environment information, a suite of benchmark tools was developed and is also available under the GNU/GPL license.

In order to validate the simulator, a parallel application was implemented, simulated and executed on a real environment. It was observed that the percentage error obtained between the actual and the predicted execution times was lower than 1%, what confirms the accuracy of the proposed model to predict the application execution on heterogeneous multicomputing environments.

References

1. de Mello, R.F.: Proposta e Avaliacão de Desempenho de um Algoritmo de Balanceamento de Carga para Ambientes Distribuídos Heterogêneos Escaláveis. PhD thesis, SEL-EESC-USP (2003)
2. et. al, E.L.: Quantitative System Performance: Computer System Analysis Using Queueing Networks Models. Prentice Hall (1984)
3. et. al, P.B.: A Guide to Simulation. Spring-Verlag (1987)
4. Kleinrock, L.: Queueing Systems - Volume II: Computer Applications. John Wiley & Sons (1976)
5. Lavenberg, S.S.: Computer Performance Modeling Handbook. Academic Press (1983)
6. Jain, R.: The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurements, Simulation and Modeling. John Wiley & Sons (1991)
7. Amir, Y.: An opportunity cost approach for job assignment in a scalable computing cluster. IEEE Transactions on Parallel and Distributed Systems **11**(7) (2000) 760–768

8. Culler, D.E., Karp, R.M., Patterson, D.A., Sahay, A., Schauser, K.E., Santos, E., Subramonian, R., von Eicken, T.: LogP: Towards a realistic model of parallel computation. In: Principles Practice of Parallel Programming. (1993) 1–12
9. et. al, R.F.M.: Analysis on the significant information to update the tables on occupation of resources by using a peer-to-peer protocol. In: 16th Annual International Symposium on High Performance Computing Systems and Applications, Moncton, New-Brunswick, Canada (2002)
10. Sivasubramaniam, A.: Execution-driven simulators for parallel systems design. In: Winter Simulation Conference. (1997) 1021–1028
11. et. al, S.C.: Towards a communication characterization methodology for parallel applications. In: Proceedings of the 3rd IEEE Symposium on High-Performance Computer Architecture (HPCA '97), IEEE Computer Society (1997) 310
12. Vetter, J.S., Mueller, F.: Communication characteristics of large-scale scientific applications for contemporary cluster architectures. J. Parallel Distrib. Comput. **63**(9) (2003) 853–865
13. Sivasubramaniam, A., Singla, A., Ramachandran, U., Venkateswaran, H.: An approach to scalability study of shared memory parallel systems. In: Measurement and Modeling of Computer Systems. (1994) 171–180
14. Singh, J.P., Weber, W., Gupta, A.: Splash: Stanford parallel applications for shared-memory. Technical report (1991)
15. Anderson, R.J., Setubal, J.C.: On the parallel implementation of goldberg's maximum flow algorithm. In: Proceedings of the fourth annual ACM symposium on Parallel algorithms and architectures, San Diego, California, United States, ACM Press (1992) 168–177
16. Bailey, D.H., Barszcz, E., Barton, J.T., Browning, D.S., Carter, R.L., Dagum, D., Fatoohi, R.A., Frederickson, P.O., Lasinski, T.A., Schreiber, R.S., Simon, H.D., Venkatakrishnan, V., Weeratunga, S.K.: The NAS Parallel Benchmarks. The International Journal of Supercomputer Applications **5**(3) (1991) 63–73
17. Feitelson, D.G., Rudolph, L., Schwiegelshohn, U., Sevcik, K.C., Wong, P.: Theory and Practice in Parallel Job Scheduling. In: Job Scheduling Strategies for Parallel Processing. Volume 1291. Springer (1997) 1–34 Lect. Notes Comput. Sci. vol. 1291.
18. Chiola, G., Ciaccio, G.: A performance-oriented operating system approach to fast communications in a cluster of personal computers. In: In Proc. 1998 International Conference on Parallel and Distributed Processing, Techniques and Applications (PDPTA'98). Volume 1., Las Vegas, Nevada (1998) 259–266
19. Chiola, G., Ciaccio, G.: (Gamma: Architecture, programming interface and preliminary benchmarking)
20. Chiola, G., Ciaccio, G.: Gamma: a low cost network of workstations based on active messages. In: Proc. Euromicro PDP'97, London, UK, January 1997, IEEE Computer Society (1997)
21. W.C.Shefler: Statistics: Concepts and Applications. The Benjamin/Cummings (1988)
22. Kerrigan, T.: Tscp benchmark (2004)
23. Beguelin, A., Gueist, A., Dongarra, J., Jiang, W., Manchek, R., Sunderam, V.: PVM: Parallel Virtual Machine: User's Guide and tutorial for Networked Parallel Computing. MIT Press (1994)
24. Pacheco, P.S.: Parallel Programming with MPI. Morgan Kaufmann Publichers (1997)
25. Burden, R.L., Faires, J.D.: Análise Numérica. Thomson (2001)