PIV and WPIV: Two New Performance Indices for Heterogeneous Systems Evaluation

KALINKA R. L. J. CASTELO BRANCO[1]
MARCOS JOSÉ SANTANA[2]
REGINA H. C. SANTANA[2]
SARITA MAZZINI BRUSCHI[2]
CÉLIA LEIKO OGAWA KAWABATA[3]
EDWARD DAVID MORENO ORDONEZ[1]

UNIVEM - Fundação de Ensino Eurípides Soares da Rocha de Marília
Av. Hygino Muzzi Filho, 529 - Campus Universitário, Marília, SP
[1](kalinka,edmoreno)@univem.edu.br

ICMC / USP - Univesidade de São Paulo
Av. Trabalhador Sancarlense, 400, São Carlos, SP, Brasil
[2](mjs,rcs,sarita)@icmc.usp.br

UNICEP – Centro Universitário Central Paulista
Rua Miguel Petroni, 5111 São Carlos, SP, Brasil
[3]celiak@gmail.com

Abstract. Two new performance indices (PIV - Performance Index Vector and WPIV – Weighted Performance Index Vector) are presented in this article. Those indices to evaluate heterogeneous computing systems are based on a Euclidian metric. Aiming to maximize the use of the machines, the proposed indices are a combination of several usual indices and the results of their evaluation through a simulator show an appropriate behavior for different kinds of applications.

Keywords: Performance Index, Load Index, Load Balancing, Distributed System, Heterogeneous Systems.

(Received January 27/2006 / Accepted August 03, 2006)

1 INTRODUCTION

The distributed computing systems have been showing through the years its advantages over the centralized systems, achieving a prominence place in the computing scenery in a very short time. That class of systems has been getting more improved in order to provide better performance, at a rather low cost.

Although the advantages obtained with the distributed parallel computing are evident, several new problems emerge from that new approach. Several researches have been developed considering the existing problems in the use of the distributed parallel computing. Problems with the means of interconnection, with the application portability, with the communication protocols, with the scheduling of processes and its applications in the final development of the system, among others, are approached on most of those researches. Scheduling of processes constitutes a theme of great importance, mainly for directly influencing the system performance.

The complexity in treating the heterogeneity of the systems concerning load indices leaves a quite attractive gap for the researches development opened. Filling in that gap may allow better performance levels to be reached and established when the simplified solutions,

already consolidated concerning homogeneous systems, may be successfully adopted for heterogeneous systems.

A novel way of treating the distributed computing systems heterogeneity, leading it to new performance indices establishment (replacing load indices) that may be used efficiently in that kind of system, basing itself on the famous rules in the literature [1-11] is presented in this article.

The performance indices here presented (PIV and WPIV) are based on a Euclidian metric and they use vectors to obtain the machine and the system load status characteristics. Once those indices match the different load indices presented in literature, results presented by them are far better than the ones presented by each individualized load index.

The purpose of this work is to contribute with the researches development both in scheduling of processes and in heterogeneous computing, using load indices already consolidated and individual characteristics of each machine or system to be evaluated through the use of those new performance indices.

2 RELATED WORK

A good performance index, as well as in the load indices, should have ways of estimating the future through current values and past factor, so that for a good performance index to be obtained, its bases should be founded in load indices. As it has been noticed, load indices are really volatile, showing the instability of the considered metric. That instability happens due to working loads flotation.

Although many researches have focused on process scheduling using load indices to improve performance, little has actually been achieved within the ambit of process scheduling in architecturally and configurationally heterogeneous systems. To date, the reference used in this area is [1]. The authors propose the use of a load index obtained from the linear combination of service time s_j, required by a task for its execution in a given resource r_j, where the length of the queue of the resource r_j is given by q_j, such that the load index l_i is obtained through:

$$li = \sum_{j=1}^{N} s_j \times q_j$$

Equation 1

where N is the total number of resources having queues. This model considers environments composed of configurationally and architecturally homogeneous machines. Wolffe at al. [12] proposes the use of Load Capacity as a load index for heterogeneous environments. Load capacity is understood here as the effective use of the processor. It is considered a load index for heterogeneous environments because it normalizes the speed of each CPU in relation to the others.

(1–Use_of_CPU)*Relative_Speed_of_the_CPU

This metric represents the processing capacity effectively remaining in the processing resource, but does not take into account the other resources that are involved in processing as a whole, such as memory, disk, and network, rendering it a very specific index with little flexibility. In addition to the considerations about this index, it should be pointed out that the experiments involving load indices (the execution of applications to obtain the final response time) were carried out on architecturally homogeneous machines.

On the other hand, Fontlupt at al. [13] proposes obtaining the load as the number of data items existing in the queue of the processor. The function of load is denoted by w. The system's total load is given by W, which is obtained through:

$$W = \sum_{i=1}^{P-1} w[i]$$

Equation 2

Considering a set of P processors that are completely homogeneous, and considering the also homogeneous tasks to be executed in these processors, one finds that the total average in these systems is given by W/P and is denoted by w.

In general, one finds that researches in the area are still ongoing, especially with regard to load indices and simulation models to verify the performance resulting from the adoption of the respective index.

The simulation models used in most cases involving studies of the implication of load indices on the final results of applications are simplified models, which have so far not taken into account the characteristics of heterogeneity of the studied platform [1, 12-14], and have focused mostly on parameters relating only to homogeneous machines and applications.

3 PIV - PERFORMANCE INDEX VECTOR

A good performance index, as well as in load indices, should have means of estimating the future through current values and past factors and, thus, for a good performance index to be obtained, its bases should be founded in load indices.

As it has been noticed, load indices are really volatile, showing the instability of the considered metrics. That instability happens due to the working loads flotation. If determinism does not exist, the need of elaborating models that reflect that characteristic and

that consider the load flotation over a period of time will exist.

The challenge of the non-determinism is in the learning strategy to discover heuristic rules that allow the choice of the "best" alternative, without needing to explore all of them.

Considering the four basic resources to be analyzed in a machine, the function presented in Equation 3 can be obtained:

$$ID = f(I_{CPU}, I_{Memory}, I_{Disk}, I_{Network}) \quad \text{Equation 3}$$

The function presented in Equation 3 can, still, use weights for each one of the specific indices of the resources:

Where ID is the performance index that considers the four basic resources. W_1, W_2, W_3 and W_4 are the weights that will be given to the indices according to the characteristic of the application to be scheduled [15]. I_{CPU} is a combination of the CPU indices that more adapt to applications strictly CPU-Bound. I_{Memory} is the combination of the Memory indices that are more adapted to applications strictly Memory-Bound, I_{disk} is the combination of the Disk indices that are more adapted to applications strictly Disk-Bound and $I_{network}$ is the combination of the Network indices that are more adapted to applications strictly Network-Bound.

Each load index is calculated independently and it considers a specific benchmark. Once the measures are presented with values that cannot be directly combined and compared, normalization should be used. Each measure operates in an open scale, what implicates that the minimum value is zero. However, the maximum value cannot be determined because it depends on the use and capacity of each machine.

Thus, each measure is normalized separately so that each specific index of CPU, Disk, Memory and Network resources has its value presented between 0 and 1 (I_{CPU}, I_{Disk}, I_{Memory} and $I_{Network}$ indices can be elaborated starting from a weighted average of several load indices, regarding several visions of the resource use).

Once each measure is normalized according to the relative benchmarks (allowing the comparison among machines in equality), and that the machines may be disposed according to a classification with values that range from 0 to 1, the values of each one of the different resources measures may be simply added and weighted.

Like load indices, a good performance index must be able to estimate the future based on current values and on past factors. Therefore, to obtain a good performance index, its bases must be founded on load indices. In this paper, a performance index is understood as the metric that can provide an image of the work capacity, or even better, that constitutes an order of magnitude capable of clearly illustrating what can be expected, in terms of performance, from the element under analysis [1]. On the other hand, the load index can be formally defined as a non-negative numerical variable, assuming the value zero when the resource is idle and having its value incremented positively when the load of this resource increases [1, 2]. Thus, the performance indices PIV and WPIV are presented below.

3.1 A Performance Index Variant

An interesting characteristic of the function presented in the Equation 3 is its possible use as a specific load index for each resource, CPU, Disk, Network, Memory, where each one of them may be seen as a vector base. This way, considering in the vector representation the order <CPU, Disk, Network, Memory>, the vector base <1,0,0,0> represents a 100% CPU application. In the same way, <0,1,0,0> can be had from 100% Disk application, <0,0,1,0> from 100% network application and <0,0,0,1> from 100% memory application.

The resources n that a machine can provide may be considered to form an n dimensional space. If a machine provides the CPU, Network, Disk and Memory resources then a four dimensional space is formed, in which a point locates the current state of this machine.

Once those resources strip of values are concentrated between 0 and 1, and that those resources are treated in a vector way, then an idle machine is located in the origin <0,0,0,0> and a completely overloaded machine is located in the opposite vertex <1,1,1,1>.

For instance, Figure 1 presents a three-dimensional space, once four-dimensional spaces are complex in terms of graphic and hypercube representations, it would turn the visualization very complex.

In Figure 2, three peculiar points of load can be observed in a machine. Situation A represents a machine with great use of CPU, that does not use memory and with an average use of network. Similarly, it can be observed that situation B makes an average use of the memory and of the network while situation C makes a great use of CPU, memory and network.

Through that representation two kinds of information may be obtained. The angle between the machine vector use and the x axis shows the relative percentage of each resource use. The length of the vector represents how much of each resource is used.

Considering two resources (1 and 2), the Load Balancing can be observed through θ angle, and when £ ≈ 45º, both resources are equally loaded; when £ >> 45º, it indicates that the resource 1 is predominant and when £ << 45º, it indicates that the resource 2 is the predominant one.

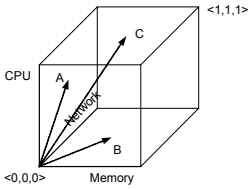

Figure 1: Three-dimensional space used to describe the current load of a machine and the three points indicating potential loads of the machine.

Once the length of the vectors is the same, in spite of the fact that machine b is balanced (|45º -£ | = 0) concerning 1 and 2 resources, it is equally classified to the machine A that is less overloaded than b regarding resource 2.

So that the overload conditions are noticed, £ angle and the length of the vector should be checked simultaneously. If £ ≈ 0 and length tends to 1, then resource 2 is close to saturation; in the same way, if £ ≈ 90 and length tends to 1, then resource 1 is close to saturation.

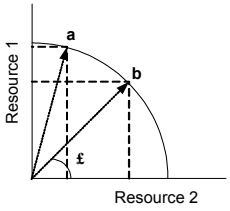

Figure 2: Different areas for two vectors with the same length (different angles)

Once the length of the vectors is the same, in spite of the fact that machine b is balanced (|45º -£ | = 0) concerning 1 and 2 resources, it is equally classified to the machine A that is less overloaded than b regarding resource 2.

So that the overload conditions are noticed, £ angle and the length of the vector should be checked simultaneously. If £ ≈ 0 and length tends to 1, then resource 2 is close to saturation; in the same way, if £ ≈ 90 and length tends to 1, then resource 1 is close to saturation.

Figure 3 illustrates the arrival of a Process P, that uses only resource 1 (R1 bound) in two different machines, but machines that are equally loaded.

Resource 1 is more loaded in machine M_1 while resource 2 is more loaded in machine M_2 concerning their use. Process P (that is resource 1 bound) can be allocated in M_1 and M_2. It should be determined in which situation a better result is obtained. A metric that can be adopted, in this case, is the Euclidian distance between the point and the origin, that is, the length of the vector from the origin to the point. Therefore Equation 3 can be written again as

$$ID = \sqrt{I_{Cpu}^2 + I_{Disk}^2 + I_{Memory}^2 + I_{Network}^2}$$

For the example presented in Figure 3, vectors C_1 and C_2 are obtained. The result of length C_2 is lower than length C_1 and this way, Process P is allocated in M_2.

It demonstrates that, in spite of the machines being equally loaded, the load distinction regarding the resources that are being used and the kind of task that will be allocated allows a better allocation of the task. This load identification by resource is provided by the metric here proposed.

The example presented in Figure 3 illustrates a simple example, because the process uses only one resource. A more complex example can be given by an application that uses more than a resource, as shown in Figure 4.

In this Figure, Process P uses more than one resource in two different machines but that are, again, equally loaded.

Resource 1 is more loaded in the machine M_1 while resource 2 is more loaded in the machine M_2 in terms of use. Process P can be allocated in M_1 and M_2. It should be determined; however, in which machine a better result is obtained, and the resulting vector is used for that. The resulting vector that presents shorter length indicates to which machine process P should be allocated. In this case, it is noticed that the process should be allocated in the machine M_2 so that a better performance can be obtained.

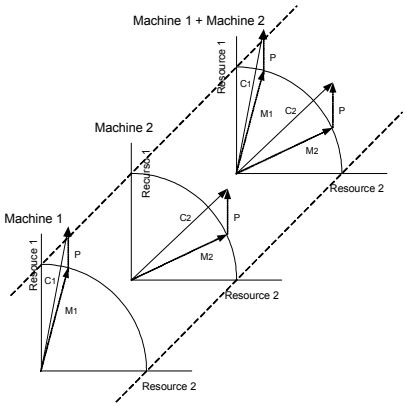

Figure 3: Two-dimensional space formed by resources 1 and 2, and two machines with the same loadings (process limited by a resource).

From the analyses made, it can be noticed that the proposal presented in this paper does not consider peculiar values of each resource, but the existing relationship among the different resources that compose a machine, allowing the allocation of the processes to be made in a more balanced way.

This way, the performance index presented in this article bases itself on the Euclidian distance between the origin point (where the machine is idle) and the resulting point among the load vectors of the machine before receiving particular application, plus the vector of the load imposed by that application. The most appropriate machine to receive the application is that one where the shortest Euclidian distance is obtained. That index, based on load vectors, will be referenced in this paper remaining as PIV (Performance Index Vector).

PIV considers that weights defined in the Equation 3 will be the same for all resources. Other PIV variants may be established with different weights where WPIV is obtained. PIV can present better results for the cases in which there are some knowledge from the kind of application to be considered or when the use of an adaptive index is possible.

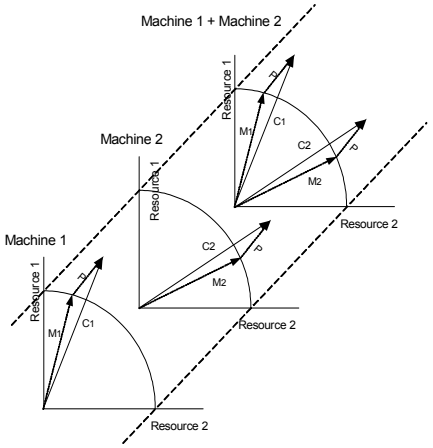

Figure 4: Two-dimensional space formed by resources 1 and 2, and two machines with the same loadings (process limited by two resources).

4 METHODOLOGY FOR THE PERFORMANCE EVALUATION DEVELOPMENT THROUGH MODELING TECHNIQUES

Several load indices execute tests were accomplished in the environment *AMIGO (*dyn**AM**ical flex**I**ble schedulin**G** envir**O**nment) [16]. Due to the amount of tests and the need to perform tests that would take a long time, the development of a queue network model and the use of simulation for the metric new proposal evaluation were chosen.

The representation of the system through a model that represents the load and performance indices and the solution of this model through simulation become attractive especially when one wants to insert modifications and obtain results. The analysis of load and performance indices is complex, mainly from the machines configuration and the scheduling environment that will be used point of view.

The process of a simulation development involves several stages. First, it is necessary to specify the model, abstracting the most important characteristics of the system. When the system is modeled, it is necessary to transform the model in a simulation program. In stochastic simulations, due to the randomness of the input data, the program should be executed several times in order to guarantee that the randomness influence in the final results is minimized.

The performance study methodology through the modeling techniques is composed of several steps that include model development, tests to guarantee that this one is correct and the results obtaining through the model experimentation. The first step in a performance evaluation study, no matter the technique used to solve the model, consists in identifying the problem that generated the need of a performance evaluation. Done that, the system that will be evaluated in the study should be analyzed and the goal to be reached should be established. Those steps, applied to load and performance indices study will be soon described.

Once having the goals of the performance analysis, the system is carefully studied so that the main characteristics for the construction of a representative model can be abstracted. A relatively difficult task in this phase consists in making a decision on which elements of the system should be included in the model and how to include them. The detailing level should be based on the purpose for which it is being built.

Model formulating step generates the requirements for the input data that will serve as its parameters. When the modeling is made on an existing system, model parameters can be measured, otherwise, they should be estimated.

Then, the technique for the model solution should be chosen, as presented in the previous section. The choice of the simulation involves the model computing representation development, which should be checked to guarantee that it is free from programming and logic mistakes. Verification compares the computing representation with the model representation, trying to guarantee that the model was faithfully represented. There are no specific rules in performing this task, but some approaches that can be followed, for instance, inspection, simulating comparison and modeling program [17]. In that case specifically, the tests executed in real platform allowed the parameters obtaining, and they are being of great value in the modeling validation.

The last step before the beginning of the model experiment to obtain the results involves the validation, which is used to show that the model represents the system in study, that is, reproduces its behavior. When the system in study exists and it can be used for measurements, the validation can base itself on the

comparison between the model results and those ones obtained by the measurements made in the real system. If there is no system, the use of some form of the conceptual modeling validation is necessary.

Being the model verified and validated, it can be experimented to obtain the expected results, selecting the measures that will be used to evaluate the performance. They should be careful with obtaining and analyzing the simulation results because it is a stochastic simulation. For that, output analysis techniques are used, which aid in obtaining a precise estimative of the performance measures.

The following sections describe the performance study through a representative model of the scheduler of processes behavior, according to the steps of a simulation. This scheduling uses only one politics; however, it evaluates different load and performance indices. The first step was taken in previous chapters that described the behavior of a scheduling of processes

4.1 Queue Network Model Development

There are several techniques to represent a model, for instance Queue Network [18], Petri-net, Statechart, Estelle. Queue Networks are more appropriate in situations where there are customers being assisted by a service company, like in the model to be implemented.

The simulating environment ASIA [19], developed at ICMC-USP was used as a tool for the model implementation. This environment uses SMPL language [18] which allows the model specification in a graphic and interactive way. The necessary base for the model construction was possible to be obtained taking as base the accomplished experiments with the load indices from the acquired experience concerning those indices and the scheduling environment used (*AMIGO*).

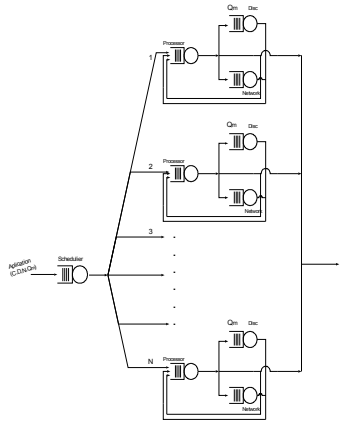

Figure 5: Queue Network model of Load and Performance Indices Simulator

The four main resources are modeled: CPU, Network, Disk and Memory. Load index model is projected to be used in heterogeneous environments.

The flow of each application (process or task) starts from the scheduling resource to the processor line. In this resource, the process stays the necessary time for processing until another resource is requested or the quantum expires. Processes move themselves through the system resources and come back to the end of the processing resource queue until it is concluded.

Figure 5 presents a macro vision of the implemented model in queue network. In this article, both for the verification stage and for the validation stage, the concepts proposed by Sargent [20] were used.

4.2 Parameters of the Model

To finalize the definition of the model, the parameters to be used in the service centers must be defined. The parameters and their meaning are listed in Table 1.

The processing elements have service times relative to each other that are measured according to each one's capacity. The disk element will be defined by t = seek time + file/bandwidth size [21]. For the network elements, the size of the message/80Mb/s will be used (considering a network interconnected by a switch that ensures this transmission capacity [22]).

The result supplied by the simulator is the time the application will take to be executed in the parallel architecture.

With these specifications, one can carry out experiments with different types of configurations of the cluster (altering parameters 1, 3, 7, 8, 9 and 10), of the applications (altering parameters 2, 12, 13, 14 and 15), and of the indices (altering parameters 4, 5, 6 and 16).

The code of the simulation program developed to carry out these tests uses the SMPL [18] and SMPLx [23] libraries.

5 EVALUATION OF THE OBTAINED RESULTS

The obtaining of the results for the performance index analysis proposed based itself on several executions of the model with the variation of several parameters, among them the parameters regarding the applications. To make the results become representative, executions with 15 different seeds were used in the execution.

To carry out tests to prove the efficiency of the performance index, several executions of the model were made, using different load indices and different kinds of applications submitted to it.

The load indices evaluated are: CPU, disk, network, memory indices, round-robing scheduling, PIV and WPIV performance indices.

Table 2 presents the result obtained through the model simulation of the scheduler of processes (the presented results represent the average of 30 executions).

Results presented in the tables demonstrate the viability of the performance index use proposed in this article, once the average times of response, when using it in the three kinds of platforms evaluated, is always better when compared to the traditional indices, being executed the specific indices for each application.

Next, the graph is presented (Figures 6), to facilitate the visualization of the several kinds of applications behavior when submitted to the scheduling using several load indices.

Table 2: Times of response in a heterogeneous machine setting

	CPU-Bound	Disk-Bound	Network-Bound	Mix 1	Mix 2
CPU	242,26	15332,33	49374,19	10800,79	10818,27
Mem.	2089,56	94195,76	115300,03	32792,78	32872,06
Disk	2089,56	6595,93	53405,00	6894,56	6894,65
Net.	2089,56	94195,76	13401,36	3628,12	3680,20
Round-Robin	289,72	9751,98	22956,65	5238,52	5135,99
PIV	212,70	6625,70	13227,89	3598,37	3624,56
WPIV	220,83	6577,75	13285,48	3667,18	3693,19

Through the observation of the results presented in the graphs, it can be noticed that in all the cases, the performance index provides better results than the ones presented by the other indices individually, executing the appropriate private index for the kind of application. However, when mixed applications are submitted, the ones that explore several resources, the behavior of the performance index is visibly better than the other indices individually.

The analyzed and presented results also encounter the results found in the literature, indicating that the generic load indices, besides presenting a tendency to a higher overload, should not present the same representation quality of load task, when compared to the specific indices used correctly [24] [1]. However, the performance index proposed, in spite of being generic, presents flexible characteristics, what makes it very close to the specific indices of each application, besides presenting very good results when submitted to mixed applications.

The presented graphs refer to the configuration where the machines are heterogeneous; however, the results presented for that configuration are expandable to other configurations, and they are not reproduced here only because it would overload the text.

Results previously presented reflect the averages of the simulation program executions with different seeds of random numbers. The simulation was developed to execute 5000 applications, no matter what kind of application is considered. This way, CPU-Bound applications will finalize in a real simulation time much shorter than Disk-Bound or Network-Bound applications.

That approach was adopted for providing more appropriate results, once the same number of applications will always be considered.

On the other hand, as the kinds of applications are different, the final timings are also different and they cannot be directly compared. This way, to make the comparison possible, normalize the obtained values was opted, basing itself on the index that generated the shortest time of response.

The evaluation of the obtained results is divided in two phases: I-Comparison among the traditional indices; II-Comparison among the indices proposed in this paper and the traditional indices.

Tables 3 to 5 present the results normalized for the traditional indices. It is observed in Table 5, for instance, that in homogeneous systems, for CPU-Bound application, the best index is the CPU one. If a memory, network or disk index is used, there will be a time of response 9,32 times higher than the one obtained for the CPU index. The applications involved in the process are evenly distributed among the resources.

Analyzing the results of the tables 3 to 5, it is observed that: The best index is always the one specific of the application. The problem found is to know the kind of the application before; For homogeneous and partially heterogeneous systems, when there are no information about the application, both the CPU and the round-robin indices can be used, so that the results are practically the same; For heterogeneous systems, the best option when the kind of application is not known is the use of CPU and Round-Robin indices. However, in those cases, there are variations that depend on the system heterogeneity level and its kind of application. In the round-robin case, it is noticed that more stable values are obtained.

In most of the papers presented in the literature, the CPU index is used for any kind of application. Considering the presented results, the use of the CPU index may be compared with the Round-robin approach.

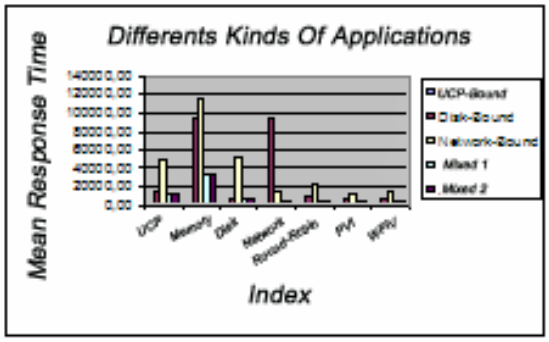

Figure 6: Summary of the different kinds of applications behavior when submitted to the several traditional load indices and the indices proposed in this article.

Table 3: Table normalized by the best load index by kind of application in homogeneous platform

	CPU-Bound	Disk-Bound	Net.-Bound	Mix 1	Mix 2	Avg
CPU	1,00	1,03	1,01	1,01	1,02	1,01
Mem.	9,32	9,42	9,38	9,48	9,31	9,38
Disk	9,32	1,00	1,04	1,02	1,00	2,68
Net.	9,32	9,42	1,00	1,00	1,01	4,35
Round-Robin	1,01	1,01	1,01	1,00	1,01	1,01

Table 4: Table normalized by the best load index by kind of application in partially heterogeneous platform

	CPU-Bound	Disk-Bound	Net.-Bound	Mix 1	Mix 2	Avg
CPU	1,00	1,00	1,02	1,03	1,02	1,02
Mem.	5,59	6,04	6,10	6,13	6,15	6,00
Disk	5,59	1,00	1,36	1,00	1,00	1,99
Net.	5,59	6,04	1,00	1,00	1,01	2,93
Round-Robin	1,18	1,28	1,30	1,29	1,30	1,27

Table 5: Table normalized by the best load index by kind of application in heterogeneous platform

	CPU-Bound	Disk-Bound	Net.-Bound	Mix 1	Mix 2	Avg
CPU	1,00	2,32	3,68	2,98	2,94	2,59
Mem.	8,63	14,28	8,60	9,04	8,93	9,90
Disk	8,63	1,00	3,99	1,90	1,87	3,48
Net.	8,63	14,28	1,00	1,00	1,00	5,18
Round -Robin	1,20	1,48	1,71	1,44	1,40	1,45

Thus, considering that in a group of applications there are:

- x CPU-Bound applications
- (1-x) non CPU-Bound applications

And that the times to execute the applications are:

- t1 = time to execute CPU-Bound applications with CPU load index
- t2 = time to execute non CPU-Bound applications with CPU load index
- t'RR = average time to execute any kind of application using round-robin

Being:

- the time to execute x applications CPU = x*t1; the time to execute (1-x) applications non CPU = (x-1) * t2 ; TRR = time to execute all the applications with round-robin = t'rr; TCPU = time to execute all the applications with cpu index = x*t1 + (1-x)*t2

This way, when TCPU <TRR the CPU index should be used and when TCPU> TRR the round-robin one should be used.

Therefore, $xt_1 + (1-x)t_2 < T_{RR}$ **Equation 4**

By solving the equation, it is had: $x < \frac{T_{RR} - T_2}{T_1 - T_2}$

So that the CPU index is more appropriate than the round-robin one. For the homogeneous and partially heterogeneous cases, the CPU index is clearly more appropriate; For the heterogeneous case, when applied in Equation 5, it is had: x> 0,77, that is, if more than 77% of the applications are CPU-Bound type, the use of the CPU index is more appropriate so that performance losses do not take place; otherwise the round-robin one can be used without any damage happens. On the other hand, different results can be observed when the proposed index – PIV is used

5.1 Comparison among the Indices Proposed in this Paper and the Traditional Indices

Table 6 present the results normalized for the traditional indices. The applications involved in the process are evenly distributed among the resources.

Similarly, analyzing the results of the table 6, it is noticed that: For heterogeneous systems, the best option, when the kind of application is not known, is the use of the proposed PIV index. Especially when the kind of application is mixed, the use of PIV becomes even more appealing in heterogeneous platforms; for the heterogeneous case, when the Equation 4 is applied, it is had: x> 1

Table 6: Table normalized by the best load index by kind of application in heterogeneous platform

	CPU-Bound	Disk-Bound	Net.-Bound	Mix 1	Mix 2	Avg.
CPU	1,14	2,32	3,73	3,00	2,98	2,64
Mem.	9,82	14,28	8,72	9,11	9,07	10,20
Disk	9,82	1,00	4,04	1,92	1,90	3,74
Net.	9,82	14,28	1,01	1,01	1,02	5,43
Round-Robin	1,36	1,48	1,74	1,46	1,42	1,49
PIV	1,00	1,00	1,00	1,00	1,00	1,00

That is, in 100% of the cases, PIV index should be used so that a performance loss does not take place.

From that analysis, it can be noticed that, executing the homogeneous platform, the performance index presents itself much better in terms of use, when the used platform is not known, as well as when the kind of application submitted to the scheduling is not known either.

The graph presented in the Figure 7 illustrates the existing interaction among the factors. It can be noticed in the graph that for the load indices 5 and 6 (PIV and WPIV respectively), the behavior of the applications is more stable than when the other load indices are used. This demonstrates that for those indices there is not practically any influence on their use for the several kinds of applications, so that they can be used indiscriminately. In the presented subtitles, the indices correspond to: 1 - CPU, 2 - Memory, 3 - Disk, 4 - Network, 5 PIV, 6 - WPIV and 7-Round-robin.

6 CONCLUSION

The new performance index proposed was used with several kinds of applications, and proper comparisons were made, presenting a performance increase, what demonstrated that the load index choice influences in the quality of scheduling of processes operations, more specifically in load balancing

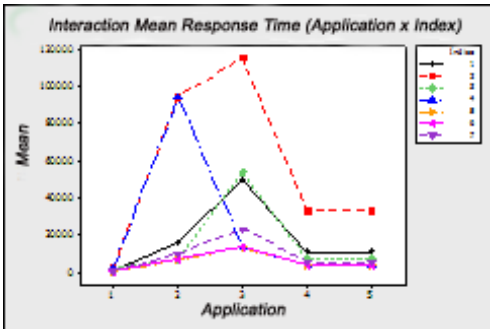

Figure 7: Relationship among the factors involved in the variance analysis.

The motivation for the construction of the model is in the fact that there is a lack of practical tools of those indices evaluation and the corresponding performance evaluation. The modeling techniques are suitable for that analysis exactly because the physical presence of the study object is not needed. The advantage of the simulation over the analytic techniques lies in the fact that the changes that are imposed to the model can be reflected more easily. This way, it was opted to carry out the study of the indices using modeling techniques and the model resolution for simulation.

The great advantages of the simulation related to the other evaluation techniques are the possibility of representing the model execution in different platforms and, through small changes in the behavior of the model, the capacity of representing different load indices.

From the manufacturing of the queue network model for scheduling of processes presented in section 4.1, not only the performance indices can be tested, but also the other existing load indices in the literature, demonstrating the usability and flexibility of that new index especially in heterogeneous platforms.

The results obtained here also demonstrate the need of considering all the resources involved in the process of scheduling of processes so that right decisions can be made, independently of the aim to be reached with the scheduling, especially when the kind of application uses several types of resources or when the kind of application that will be scheduled is not known. The accomplished evaluation, although partial, demonstrates the existing potential of the new proposed index. Used in a correct way, the performance index can improve the performance of a scheduler in a significant way

7 REFERENCES

[1] Ferrari, D.; Zhou, S. (1987). An Empirical Investigation of Load Indices for Load Balancing Applications. In Proceedings of Performance'87, the 12th Int'l Symposium on Computer Performance Modeling, Measurement, and Evaluation, p.515-528.

[2] Kunz, T. (1991). The Influence of Different Workload Descriptions on a Heuristic Load Balancing Scheme. IEEE Transactions on Software Engineering, v.17, n.7, p.725-730, July.

[3] Khokhar, A. A.; Prasanna, V.K.; Shaaban, M.E.; Wang, C.L. (1993). Heterogeneous Computing: Challenges and Opportunities. IEEE Computer, 26(6): 18-27, June.

[4] Ambrosius, S. L.; Freund, R. F.; Scott, S. L.; Siegel, H. J. (1996). Work-Based Performance Measurement and Analysis of Virtual

Heterogeneous Machines. In the 5th Heterogeneous Computing Workshop (HCW´96). April.

[5] Ekemecic, I; Tartalja, I.; Milutinovic, V. (1996). A Survey of Heterogeneous Computing: Concepts and Systems. Procceddings of IEEE, 84: 1127-1144.

[6] Beitz, A.; Kent, S.; Roe, P. (2000). Optimizing Heterogeneous Task Migration in the Gardens Virtual Cluster Computer. 9th Heterogeneous Computing Workshop, pp.140-146, Cancun - Mexico, May.

[7] Amir, Y. et al. (2000). An Opportunity Cost Approach for Job Assignment in a Scalable Computing Cluster. IEEE Transactions on Parallel and Distributed Systems, v. 11, n.7, p. 760-768, July.

[8] Abdelzaher, T. F., Shin, K. G.(2000). Period-Based Load Partitioning and Assignment for Large Real-Time Applications. IEEE Transactions on Computers, vol 49, Nº 1, p. 81-87, January.

[9] Beaumont, O.; Legrand, A.; Robert, Y. (2003). "Optimal algorithms for scheduling divisible workloads on heterogeneous systems", In HCW'2003, the 12th Heterogeneous Computing Workshop, IEEE Computer Society Press.

[10] Branco, K. R. L. J. C., Santana, M.J., Santana, R. H. C. (2003). A Novel Performance Metric for Evaluation of Computer System Heterogeneity. Proceedings of The International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS'2003). p. 292 – 301, V. 34, n. 04 SCS. Montreal – Canadá, july.

[11] Branco, K. R. L. J. C., Santana, M.J., Santana, R. H. C. (2003). A Novel Metric for Checking Levels of Heterogeneity in Distributed Computer Systems. Proceedings of The Fourth Congress of Logic Applied to Technology (LAPTEC'2003). p. 148 – 155, V. 101, IOS Press. Marília – São Paulo – Brazil, novembro.

[12] Wolffe, G.S.; Hosseini, S.H.; Vairavan, K. (1997). An Experimental Study of Workload Indices for Non-dedicated, Heterogeneous Systems. In the proceedings of PDPTA´97, v.1, p. 470-478.

[13] Fontlupt, C.; Marquet, P.; Dekeyser, J. (1998). Data Parallel Load Balancing Strategies. Parallel Computing, 24 p. 1665-1684.

[14] Karatza, H. D.; Hilzer, R. C. (2003). Parallel Job Scheduling in Homogeneous Distributed Systems. In Simulation, Vol 79, Issue 5-6, May-June, p. 287-298.

[15] Branco, K. R. L.J. C. (2004). Índices de Carga e Desempenho em Ambientes Paralelos/Distribuídos – Modelagem e Métricas. Phd Thesis (Doctorate in Computer Sciences and Computational Mathematics) – Institute of Mathematical and Computational Sciences –São Carlos campus. USP – University of São Paulo.

[16] Souza, P. S. L. (2000). AMIGO: Uma Contribuição para a Convergência na Área de Escalonamento de Processos. Phd Thesis (Doctorate in "Applied Physics – option: Computational Physics" Sciences) - São Carlos Institute of Physics, USP – University of São Paulo.

[17] Higginbottom, G. N. (1998). Performance Evaluation of Communication Networks. Artech House, Inc.

[18] Macdougall, M.H. (1987). Simulating Computer Systems Techniques and Tools. The MIT Press.

[19] Bruschi, S. M. (1997). Extensão do ASiA para Simulação de Arquiteturas de Computadores. São Carlos. Dissertação (Mestrado em Ciências da Computação e Matemática Computacional) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo.

[20] Sargent, R. G. (1999) Validation and Verification of Simulation Models. In Proceedings of the 1999 Winter Simulation Conference, p. 39 – 48. White Paper AMD (2002). QuantiSpeedTM Architecture. Advanced Micro Devices, INC - One AMD Place. Sunnyvale, CA 94088. January.

[21] White Paper AMD (2002). QuantiSpeedTM Architecture. Advanced Micro Devices, INC - One AMD Place. Sunnyvale, CA 94088. January.

[22] Kant, K., MOHAPATRA, P. (2000). Scalable Internet servers: Issues and challenges. In Proceedings of the Workshop on Performance and Architecture of Web Servers (PAWS). ACM SIGMETRICS, June.

[23] Ulson, R.S. (1999). Simulação Distribuída em Plataformas de Portabilidade: Viabilidade de Uso e Comportamento do Protocolo CMB. Phd Thesis (Doctorate in "Applied Physics – option: Computational Physics" Sciences) - São Carlos Institute of Physics, USP – University of São Paulo.

[24] Mehra, P.; Wah, B.W.(1993). Automated Learning of Load-Balancing Strategies for a Distributed Computer System. University of Illinois at Urbana-Champaign.