

Load Indices on Heterogeneous Systems– Past, Present and Future

Kalinka Regina Lucas Jaquie Castelo Branco¹, Edward David Moreno Ordonez²

¹ UNIVEM – Centro Universitário “Eurípides Soares da Rocha” de Marília

² UEA Universidade do Estado do Amazonas Manaus – Amazonas - Brasil
kalinka@univem.edu.br, edmoreno@uea.edu.br

Abstract

To achieve superior performance levels when working with heterogeneous distributed platforms, process-scheduling decisions must be based essentially on the platform’s features and on the application’s demands. A primary objective of computing systems is the efficient use of available resources. The effective use of these resources in this type of system therefore requires strategies to distribute process scheduling equally among the multiple processors available. An essential element of the scheduling policy is the measurement of loads, and an accurate load index is required to manage the load distribution efficiently. Therefore, this paper sought to compile and present a survey of the load indices contained in the literature, present a new one used to obtain better performance in heterogeneous systems and propose new investigation to the future of load indices and load metrics.

1. Introduction

In recent decades, computer sciences have made great advances in the development of micro processing technology and interconnection hardware. This development, allied to the acquisition and maintenance costs of massive parallel processing machines (MPPs), has led to the use of systems composed of general purpose computers equipped with distributed software for the execution of parallel applications.

Parallel computation performed in distributed computer systems, or distributed parallel computation, is a challenging field. The feasibility of techniques, models and performance measurements, which are common for the construction, partition and mapping of parallel programs used in the past, is often much reduced in distributed systems.

Characteristics inherent to these systems, such as heterogeneity [1] [2] [3], weak coupling of resources, and the presence of applications having different computational needs and user satisfaction, which used the system interactively, create additional requirements for parallel software.

Parallel machines have homogeneous processing elements (PEs) and a dedicated and high performance interconnection network. Distributed systems, on the other hand, have potentially heterogeneous PEs and communication means, shared by applications and users with different requirements.

In homogeneous systems or even in configurationally heterogeneous systems, the mechanisms and concepts employed to obtain information about applications and workloads are relatively simple and are presented clearly in the literature. However, the acquisition and utilization of this information in distributed systems is not so simple and the mechanisms and concepts used for homogeneous systems are little suited to heterogeneous systems.

The constantly evolving field of process scheduling offers a wide variety of excellent reports on research work. Information about scheduling, and particularly about load indices, is widely available in the literature.

These load indices are one of the key issues in the design of any process-scheduling algorithm, particularly when it comes to load balancing. Their purpose is to predict the performance of a task executed by a given machine.

This paper aiming to presents the several stochastic models that exist in the literature for load indices in architecturally and configurationally homogeneous environments [4] [5] [6] [7] [8] [9] [10] [11], that can be see in section 2. However, such a wide range of models is not available for heterogeneous environments, especially in terms of architectural heterogeneity. So, is also present specifically discussing the requirements and challenges involved in obtaining indices of loads carried out in distributed platforms. In this context, a discussion is given of a relevant set of knowledge, i.e., the knowledge of workloads, looking to the past to make the present (section 3) and imagine the future that can be seeing in section 4. And, finally section 5 presents the conclusion.

2. State of the art

In a distributed computational system, the potential for sharing resources and the possible gains deriving there from are substantial. The main advantages of sharing resources are the large number of available resources (in terms of both type and quantity) and the high potential resulting from the great variety of these resources.

Several measurements and metrics that provide information about these resources have been presented in order to obtain the best possible use of the available resources. One of the items of information pertinent to the system's state, whose basic characteristic is its rapid mutability, is the load to which these resources are subjected.

Because these loads change very rapidly (depending mostly on the application's domain), they soon tend to become obsolete. While the need to update this load index is obvious, the frequency at which this updating is performed inevitably depends on the application's characteristics.

Load information is a resource generally defined in terms of a load index. This load index is a metric that quantifies the load to which an element of the system is being subjected [4] [5]. The purpose of this metric is to indicate whether the element in question is idle (with no load or with a very small load), overloaded (in which case the scheduler should avoid allocating processes in this element, or even consider the possibility of migration to relieve the element from its load), or in a normal processing situation and available to accept a few additional processes.

The load index can therefore be defined formally as a numerical, non-negative variable that assumes the value of zero when the resource is idle and whose value increases positively as the load to which it is subjected increases [4] [5].

The identification of appropriate loads with the purpose of considerably increasing the utilization of idle resources is proven an important point to be considered in the design of any attempt to evaluate performance, and particularly in the design of a load-balancing algorithm.

For a load index to be effective, it must reflect accurately, or at least as closely as possible, the state of the evaluated resource [4], its purpose therefore being to predict the performance of a task executed by a given machine.

However, although the techniques used to measure loads are efficient, they must impose a minimum load on the system [6] so that the cost of obtaining this index does not interfere with the system's performance [4] [5] [17] [18].

The load index should therefore focus on the objective proposed by the process scheduling algorithm, indicating not only the system's current load but also predicting a future behavior, based on a current situation or on a situation of the recent past [5] [7].

The characteristics listed below are important points to consider when choosing a load index:

The need to quantify not only the use of the processor required for the processes, but also the memory and I/O requirements;

Predict future loads on a resource, since these loads affect a system's performance more strongly than the current load;

Allow for a simple relation with the load index, so that these values are easily translatable;

Adopt stable indices, rejecting variable values.

There is no consensus, however, about the efficiency and effectiveness of a single load index, which is why the literature contains a wide range of load indices.

Some of the indices that have been used, studied, analyzed and evaluated are the CPU queue length, the average CPU queue length within a given time frame, CPU utilization, the system's call rate, the amount of memory available, and the response time and hybrid functions that use junctions and combinations of the abovementioned indices [4] [5] [17] [18] [19].

Generically speaking, load indices can be divided into groups based on, for instance, length of the queue waiting to use the resource, percentage of usage of the resource, and execution/response time (in performance measures). These groups of indices can also be called specific and generic indices. Specific indices involve obtaining a single value and, according to some authors, they are useful in specific cases, imposing lower overloads [5]. Generic indices involve the joining of two or more values obtained and are recommended by some authors when not enough is known about the application or about the objectives of the scheduler [4] [20] [21].

Generic load indices, in addition to displaying a greater tendency to overload, are not expected to show the same quality of workload representative as correctly utilized specific indices [4] [6].

The works involving load indices also commonly use abstract queue models representing computational systems to analytically obtain load index functions. Among these models, the ones proposed by Ferrari and Zhou stand out [4] [20] [21].

Most of the models to calculate load indices given in the literature are destined for configurationally and architecturally homogeneous environments. Only a small minority of these models is based on heterogeneity and, even then, only configurationally heterogeneity. The following exemplify these models (both homogeneous and heterogeneous):

Ferrari & Zhou [4] propose the use of a load index obtained through a linear combination of the service time, s_j , required for the execution of a task in a given resource, r_j , in which the queue length of the r_j resource is given by q_j , so that the load index is obtained through:

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

where N is the total number of resources having queues. This model takes into account environments composed of configurationally and architecturally homogeneous machines.

In their research, Lin & Keller [22] used a load index that considers information obtained from the number of tasks and the amount of memory that is being utilized. Based on a simulator (Rediflow [22]), they specified the various parameters, including, among others, the number of machines, the amount of memory, the machines' configurationally composition, the communication bandwidth. Based on the data supplied by the simulator, the load index is then calculated through:

$$IC = \text{number-of-tasks} + \text{parameters} / (1 - \text{memory-in-use})$$

where parameters stands for the parameters supplied by the simulator.

In his study, Kunz [5] stated that the scheduling effected with any load index is much better than a system that does not perform any balancing (as previously demonstrated by [4]). His study evaluated the performance achieved by the scheduler when the load is balanced in relation to six linear load indices, and found that the best index is the number of processes in the queue of completed ones. In another evaluation, Kunz attempted to use aggregated load indices; however, his experiments demonstrated that these indices did not improve the system's performance in comparison to the linear indices and that, besides, they caused overloading when obtaining the various linear indices that would make up the aggregated indices. Once again, the studies made here were carried out in typically homogeneous environments.

Zhou et al. [21] propose that load indices should vary according to the nature of the resource under evaluation. Thus, these indices might be the queue length, the use or the amount of free resource. This proposal implies the use of specific resource load indices for each machine. The indices used are obtained through the average CPU queue length, the amount of free memory, the mean rate of transfer from a disc to all the other discs when an I/O is effected, the amount of available disc space for the exchange of pages, and the number of users in the system.

In their research, Mehra & Wah [6] used neural networks to obtain load indices. The model considers only configurationally heterogeneity. However, to calculate the load index of the system's machines takes five days of processing. During these five days, the index is calculated based on the knowledge learned by the neural network. The load index considers that S is the number of machines, F is the foreground workload, T is the time to capture the load, and B is the background workload. Hence, $lb,f(t),s$ is the vector that contains the level of utilization of CPU (in an machine of S), memory, disc and network in each T time for b and f begun in t . In fact, this is a four line matrix (representing the resources: disc memory, network and disc) and a column (representing the time T – the moment of time in which the measurement is being made). Therefore, $lb,f(t),s$ is the vector of values derived from the recent behavior of the loads present in the various machines, both of foreground (f) loads and of background (b) loads begun in t , in which case it is assumed that, in all the machines, the time taken to obtain the index is the same, although this is not what occurs in reality. The vector obtained in time t is then divided by a vector obtained from a reference machine, i.e., an idle machine. Given these values, the objective of learning of the load index is:

$$F_s^u[\hat{f}_{b,f,s,t}, f] \quad F_s[\hat{f}_{b,f,s,t}, f] = \frac{C[\hat{f}_{b,f,s,t}, f]}{C[\hat{f}_{0,f,ref}, f]} \quad (2)$$

Franklin & Govindan [23] propose an iterative matrix model to obtain the load index, considering that the tasks are identical and that the processors' computational power may differ, although the architectural homogeneity must be maintained. According to these definitions, the number of processors is given by p and the computational power is given by C_i , and this, in turn, is proportional to the mean number of operations executed by the processors in one unit of time. Hence, the time spent to execute a task in P_i is inversely proportional to C_i . The load index is therefore obtained through the allocation of the number of tasks proportionally to the respective computational power of the machine.

Wolffe, Hosseini & Vairavan [8] propose the use of Load Capacity as a load index for heterogeneous environments. Load capacity is understood as the effective use of the processor. A load index is considered for heterogeneous environments by normalizing the CPU's velocity.

$$(1 - \text{CPU Utilization}) * \text{Relative CPU Velocity}$$

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 the processing as a whole, such as memory, disc, and network, which

renders the index highly specific and somewhat inflexible.

In addition to the aforementioned considerations regarding this index, it should also be noted that the experiments were carried out in architecturally homogeneous machines, which leaves signs of the non-applicability of this index in architecturally heterogeneous environments.

Fontlupt et al. [9] propose identifying the load as the number of items of data existing in the processor's queue. The function of the load is denoted by w . The system's total load is given by W , which is obtained by:

$$W = \sum_{i=1}^{P-1} w[i] \quad (3)$$

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

The literature generally recommends the adoption of a load index consisting of the average of a few essential characteristics of some resources over a given time frame, reducing the possibility of choosing a value that does not correspond to the system's real state [4].

Thus, there are significant differences in the efficiency of each load index, proving, through the various research projects that have been carried out [4] [5] [24], that the simplest load indices are the most efficient ones, better reflecting the system's state.

The load index most commonly assumed in the literature as the best index is the CPU queue length. This index is widely used because it allows for analytical modeling and is often utilized in theoretical analyses. The greatest advantage it offers is the fact that it permits precise mathematical characterization, in addition to being highly attractive for simulation purposes.

However, it must be kept in mind that the work of [4] [5] and others is typical of initial investigative and research studies of load indices and that the characteristics of the physical resources of those days differ very substantially from those available today, almost twenty years later. Their work consisted of investigations of load index factors such as the frequency of load updating, I/O and computational characteristics.

In short, they found that scheduling using the results of any load index is always better than systems that do not perform any balancing; that load indices based on averages are usually better than instantaneous indices; and that the mean CPU queue length is considerably better than CPU utilization. Nonetheless, it must be kept in mind that their studies were mostly conducted in homogeneous environments.

The load index is even more important when using heterogeneous and multi-user platforms. In addition to the alterations made by the application itself, the presence of other users' applications and differences between processors exert a decisive influence on the application's expected performance.

The different computing powers of distributed and heterogeneous computer platforms add to the load index the task of normalizing the computing loads in relation to the power of each processing element. This normalization is not simple, for it implies a greater or lesser impact of heterogeneity, depending on the application utilized.

3. The Present

In the present, to improve the quality of the load characterization in heterogeneous machines, two new performance indices (PIV - Performance Index Vector and WPIV - Weighted Performance Index Vector) [12] [13] [14], to evaluate heterogeneous computing systems, based on a Euclidian metric are proposed. 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.

As mentioned earlier, the various articles published over the last few decades have not proposed a metric aimed at a more comprehensive situation that takes into account the current situation of systems.

With the purpose not only of obtaining information about both the workload and the operating condition of each of the system's elements involved in the process, our future work will focus on the conception of performance indices that, potentially, will be able to supply information taking into account not only configurationally and architectural heterogeneities, but also the temporal heterogeneity of the machines in the system.

Performance index is understood as the metric capable of supplying an image of the work capacity, or better yet, the metric that constitutes the magnitude that clearly illustrates what can be expected from the element in question in terms of performance. 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 [4, 13].

In this sense, therefore, the environment's heterogeneity must be considered in order to obtain a real view of the available computing power. This index must therefore also take into account the applications' heterogeneity in order to obtain a real measurement of the computing power.

A good performance index, similarly to load indices, must offer the means to estimate the future based on current values and past factors. Hence, so that a good performance index can be obtained, its theoretical bases must be founded on load indices.

As has been observed, load indices are highly variable, revealing the apparent instability of the metrics considered here. This instability derives from the oscillations of nondeterministic workloads. Since determinism is absent, stochastic models must be elaborated in order to obtain results.

The challenge of no determinism lies in the learning strategy to discover heuristic rules that allow the "best" alternative to be chosen, without the need to explore them all.

Stochastic processes and models allow for the establishment of solutions to obtain these performance indices, since they take into account the no determinism allow for a reliable characterization of the heterogeneity.

The purpose of performance indices, in this context, is to generalize procedures, besides allowing existing approaches to be tested and considered, thus constituting a new approach that includes the environment's heterogeneity. This can also be seen as a new strategy to measure the behavior of a given system.

Considering the four basic resources to be analyzed in a machine, the function presented in Equation 4 can be obtained:

$$ID = f(I_{CPU}, I_{Memory}, I_{Disk}, I_{Network}) \quad (4)$$

The function presented in Equation 4 can, still, use weights for each one of the specific indices of the resources. ID is the performance index that considers the four basic resources and 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 [14]. 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.

An interesting characteristic of the function presented in the Equation 4 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 n resources 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>.

The Equation 5 shows the metric:

$$ID = \sqrt{I_{Cpu}^2 + I_{Disk}^2 + I_{Memory}^2 + I_{Network}^2} \quad (5)$$

For the example presented in Figure 1, 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. 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).

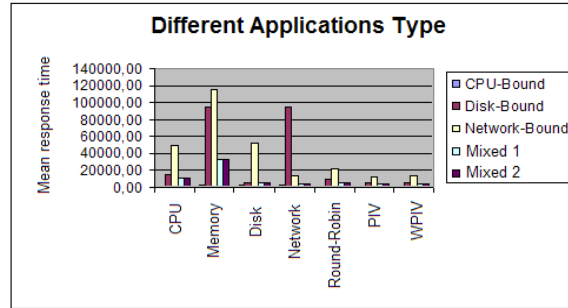


Figure 2 Summary of the behavior of different types of applications subjected to the various traditional load indices and to the indices PIV and WPIV

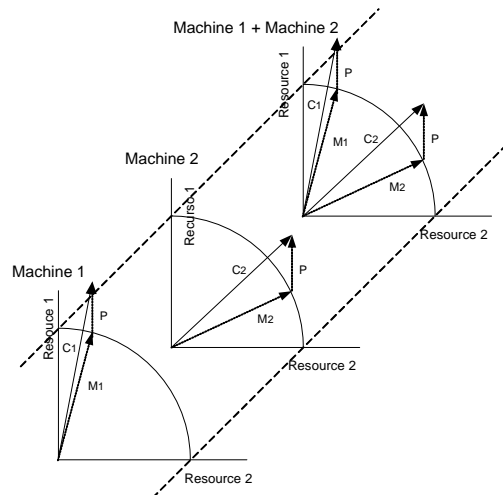


Figure 1 - Two-dimensional space formed by resources 1 and 2, and two machines with the same loadings (process limited by a resource) [13]

PIV considers that weights defined in the Equation 4 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. Some Tests to validate the new indices were made using simulation [15] [16] and relevant results were obtained.

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.

The code of the simulation program developed for carrying out these tests uses SMPL [16] and SMPLx [25] libraries. 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-robin scheduling, PIV and WPIV performance indices. Table I presents the result obtained through the model simulation of the scheduler of processes (the presented results represent the average of 15 executions).

Table 1. 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

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 (Figure 2), to facilitate the visualization of the several kinds of applications behavior when submitted to the scheduling using several load indices.

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 [4]. 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.

The study of load indices is not a simple matter, particularly since the literature does not contain references that offer a broad vision of the subject that encompasses all the necessary characteristics and parameters so as to obtain a load index that faithfully represents the similarities, differences and peculiarities of heterogeneous machines.

4. The Future

Given the past and the present, we can look to the future and the load index more specific the performance indices can be seen in some knowledge of: application, architectural parameters and collect information.

With the purpose not only of obtaining information about both the workload and the operating condition of each of the system's elements involved in the process, our future work will focus on the conception of performance indices that, potentially, will be able to supply information taking into account not only configurationally and architectural heterogeneities, but also the temporal heterogeneity [12] of the machines in the system.

In the knowledge of architectural parameters can be observed that the concepts of CPU, Disk, Memory and Network could be specify in other level, i.e., CPU index can be specify in branch or arithmetic co-processor information; Network index can be specify bandwidth and latency information; Memory index can be specify in different kinds of level caches.

In the knowledge of application we can seek to use the performance indices in a Virtual Reality (VR) system [26] to promote the scheduling and the load balancing in the environment. Another interesting use is in the image processing [27], to make the process in parallel to reduce the overhead and improve the performance of the image processing in large medical images.

We can use and make a performance evaluation in the use of the performance indices in irregular application [28]. Parallelism with irregular patterns of data, communication and computation is hard to manage efficiently. If a proper index was used maybe some problem can be solved.

In the knowledge of collect information, where this component is responsible for collecting load information,

calculating the load, and propagating it to the other hosts, the performance indices can be improved using mobile agents, genetic algorithms. These implementations can reduce the overhead and maximize the performance of the information policy in a scheduler. Fuzzy also can be used to find the perfect weight for the WPIV.

5. Conclusion

The constantly evolving field of process scheduling offers a wide variety of excellent reports on research work. Information about scheduling, and particularly about load indices, is widely available in the literature.

These load indices are one of the key issues in the design of any process-scheduling algorithm, particularly when it comes to load balancing. Their purpose is to predict the performance of a task executed by a given machine.

The study of load indices is not a simple matter, particularly since the literature does not contain references that offer a broad vision of the subject that encompasses all the necessary characteristics and parameters so as to obtain a load index that faithfully represents the similarities, differences and peculiarities of heterogeneous machines. Therefore, this paper sought to compile and present a survey of the load indices contained in the literature and give a glimpse of the future.

Empirical studies involving load indices have shown that increases in performance through load balancing are strongly dependent on the quality of the index utilized. The aim of obtaining good load balancing in distributed computing systems led to the need for indices that consider the heterogeneity of these systems.

The performance index is a novel proposal whose purpose is to support process scheduling and it is designed specifically to address the drawbacks highlighted above. The performance index, besides providing flexibility and transparency, can be used in other research projects involving, for instance, hardware and software simulations, as well as for the study of different scheduling policies.

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.

6. Acknowledgments

The authors gratefully acknowledge the financial support of the Brazilian research funding agencies CAPES, CNPq and FAPESP under process #99/09333-3, and the support given to the projects of the ICMC-USP Laboratory of Distributed Systems and Concurrent Programming.

References

- [1] 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.
- [2] Braun, T. D.; Siegel, H. J.; Beck, N.; Boloni, L.; Maheswaran, M. ; Reuther, A. I.; Robertson, J. P.; Theys, M. D.; Yao, B (1998). A Taxonomy for Describing Matching and Scheduling Heuristics for Mixed-Machine Heterogeneous Computing Systems. *IEEE Workshop on Advances in Parallel and Distributed Systems*. October, pp. 330-335.
- [3] 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.
- [4] 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.
- [5] 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.
- [6] Mehra, P.; Wah B. W., (1993) Automated learning of workload measures for load balancing on a distributed system, in *Proceedings of the 1993 International Conference on Parallel Processing*, vol. 3, (Boca Raton, FL), pp. III--263--III--270, CRC Press.
- [7] Schnor, B.; Petri, S.; Langendörfer, H. (1996). Load Management for Load Balancing on Heterogeneous Platforms: A Comparison of Traditional and Neural Network Based Approaches. In *Second International Euro-Par Conference - Euro-Par'96, Lecture Notes in Computer Science v.1124*, Lyon, France, p.611-614, August.
- [8] 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.
- [9] Fontlupt, C.; Marquet, P.; Dekeyser, J. (1998). Data Parallel Load Balancing Strategies. *Parallel Computing*, 24 p.

- 1665-1684.
- [10] Voorsluys, W.; Souza, P.S.L. (2002). Uma Política de Escalonamento de Processos como uma Ferramenta de Testes para Índices de Carga. In XI EAIC - Encontro Anual De Iniciação Científica, Maringá. Anais do XI EAIC.
 - [11] Mello, R. F. (2003). Proposta e Avaliação de Desempenho de um Algoritmo de Balanceamento de Carga para Ambientes Distribuídos Heterogêneos Escaláveis. Tese (Doutorado) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos.
 - [12] 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.
 - [13] BRANCO, K.R.L.J.C. ; SANTANA, M.J. ; SANTANA, R.H. C. ; BRUSCHI, S. M. (2006). PIV and WPIV: Performance Index for Heterogeneous Systems Evaluation. In: IEEE International Symposium on Industrial Electronics, 2006, Montreal. Proceedings of the IEEE International Symposium on Industrial Electronics, 2006. p. 323-328.
 - [14] BRANCO, K.R.L.J.C.; SANTANA, M.J.; SANTANA, R.H.C.; BRUSCHI, S. M. ; KAWABATA, C.L.O. (2006). Índices de Carga para Ambientes Paralelos/Distribuídos Heterogêneos. In: XXVI Congresso da SBC - IV Workshop em Desempenho de Sistemas Computacionais e de Comunicação, 2006, Campo Grande. Anais do XXVI Congresso da SBC, 2006. p. 144-162.
 - [15] BRANCO, K.R.L.J.C. ; SANTANA, M.J. ; SANTANA, R.H.C.; BRUSCHI, S. M. (2006) . A Novel Simulator for Evaluating Performance Indices. In: IEEE International Symposium on Industrial Electronics, 2006, Montreal. Proceedings of the IEEE International Symposium on Industrial Electronics - ISIE2006, 2006. p. 3316-3321.
 - [16] MacDougall, M.H. (1987). Simulating Computer Systems Techniques and Tools. The MIT Press.
 - [17] Zhou, S. (1987). An Experimental Assessment of Resource Queue Lengths as Load Indices. Proc. Winter USENIX Conf., p.73-82.
 - [18] Dantas, M. A. R.; Zaluska, E. J. (1998). Efficient Scheduling of MPI Applications on Networks of Workstations. Future Generation Computer Systems – FGCS, v. 13, p. 489-499.
 - [19] Xu, C.; Lau, F.C.M. (1997). Load Balancing in Parallel Computers: Theory and Practice. Kluwer Academic Publishers, Boston, USA.
 - [20] Zhou, S.; Zheng, X.; Jingwen, Z.; Delisle, P. (1992). Utopia: A Load Sharing Facility for Large Heterogeneous Distributed Computer Systems. Tech-Report CSRI-257, Computer Systems Research Institute. University of Toronto, April.
 - [21] Zhou, S.; Zheng, X.; Wang, J.; Delisle, P. (1993). Utopia: a Load Sharing Facility for Large, Heterogeneous Distributed Computer Systems. Software: Practice and Experience, v.23(12), p.1305-1336, December.
 - [22] Lin, F. C. H.; Keller, R. M. (1987). The Gradient Model Load Balancing Method. IEEE Transactions on Software Engineering, v.SE-13, n.1, p. 32-38, January.
 - [23] Franklin, M. A.; Govindan, V. (1996). A General Matrix Iterative Model for Dinamic Load Balancing. Parallel Computing, v. 22, p. 969-989.
 - [24] Shirazi, B.A.; Hurson, A.R.; Kavi, K. M. (1995). Mechanisms for Process Migration. Introdução do Sexto Capítulo do Livro Scheduling and Load Balancing in Parallel and Distributed Systems, IEEE Computer Society Press, Los Alamitos, CA, USA, p.411-413.
 - [25] 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) - Institute of Physics, USP – University of São Paulo.
 - [26] Kirner, C., Tori, R. (ed.) (2004). Realidade Virtual, Conceitos e Tendências. Pré Simpósio SVR. VII Symposim on Virtual Reality.
 - [27] McPherson, K., Banerjee, P. (1996). Integrating task and data parallelism in an irregular application: a case study. Parallel and Distributed Processing. Eighth IEEE Symposium on vol. issue, 23-26 Oct p. 208 – 213.

Kalinka Regina Lucas Jaquie Castelo Branco, Received the MSc. and PhD degrees in Computer Science from University of So Paulo - Brazil, in 1999 and 2004, respectively. She is a lecturer in the Fundao de Ensino "Eurpides Soares da Rocha" (UNIVEM). Her research interests include distributed systems, performance evaluation, load balance, computer architecture and computer network. She is a member of the Brazilian Computer Society (SBC).

Edward David Moreno. Received the MSc. and PhD degrees in Electrical Engineering from University of Sao Paulo - Brazil, in 1994 and 1998. During 1996 and 1997 he stayed as invited researcher at University of Toronto, Canada, and Chalmers University of Technology, Sweden. Recently, he is prof. at the UEA (Amazonas State University), and researcher in the BENq Mobile Corporation, Manaus, Amazonas, Brazil. The research areas are: computer architecture, reconfigurable computing, embedded systems, hardware security and power-aware computing.