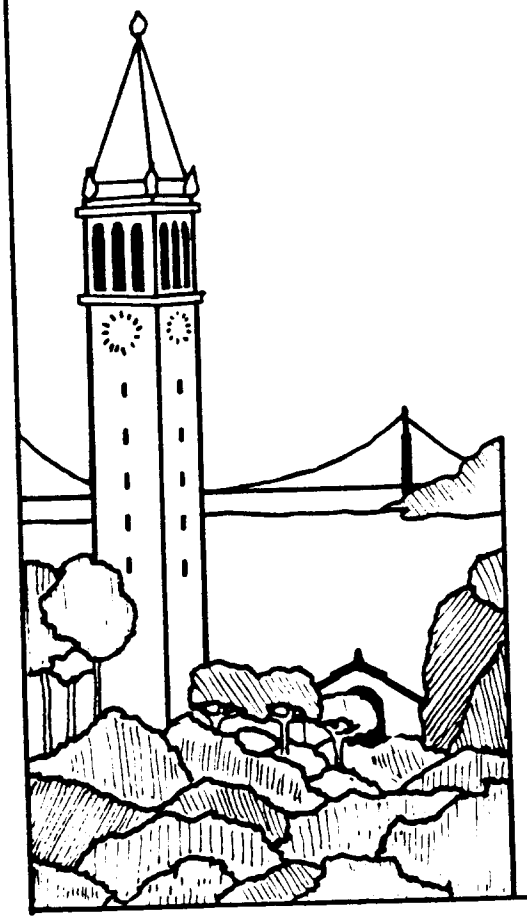
A Study of Load Indices for Load Balancing Schemes

Domenico Ferrari



Report No. UCB/CSD 86/262

October 1985
PROGRES Report No. 85.14

Computer Science Division (EECS)
University of California
Berkeley, California 94720

A Study of Load Indices for Load Balancing Schemes (*)

Domenico Ferrari

Computer Systems Research Group
Computer Science Division
Department of Electrical Engineering and Computer Sciences
and the Electronics Research Laboratory
University of California, Berkeley

Abstract

The problem of selecting the load index or indices to be used in dynamic load balancing policies is discussed. One such index, based on a mean-value equation, is proposed, and its main characteristics investigated. The index is obtained assuming that the goal of the load balancing scheme is the minimization of the response time of the user command being considered for possible remote execution. A few major obstacles to the practical use of the index are also discussed.

(*) The research reported herein was partially supported by the Defense Advanced Research Projects Agency (DoD), ARPA Order No. 4031, monitored by the Naval Electronics Systems Command under Contract No. N00039-82-C-0235. The views and conclusions contained in this document are those of the author, and should not be interpreted as representing official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or of the U.S. Government.

1. Introduction

One of the most important potential benefits of loosely-coupled distributed systems is in the area of resource sharing. By interconnecting a number of machines via a data communications network with an adequate bandwidth, a larger variety and a larger number of hardware and software resources can be made available to the users of the resulting distributed system than is usually possible in a centralized system. The processing powers of the hosts in a distributed system are among the sharable resources, and are indeed made available to remote users in most such systems, sometimes via a *remote login* mechanism, some other times by allowing users to subdivide the work to be accomplished between a foreground machine and a background "number cruncher," and some other times by having on the network a pool of "public" machines (compute servers) accessible on demand to the users.

In spite of these mechanisms and provisions, we cannot state that the processing resources of a distributed system are always shared as much as they could and should be. This unsatisfactory state of affairs is particularly noticeable, and particularly unfortunate, in many local-area network (LAN) based distributed systems, where the small intercomputer distances, the relatively broad bandwidths, and the greater homogeneity of host ownership (which is frequently restricted to a single organization) would make it easier and more rewarding to share the processing resources of the hosts among all or most of the system's users. The dramatic imbalances among the loads of the various hosts we often observe in these systems cause poor performance and a waste of system resources. In certain LAN-based installations, both the workload imposed on the system by each user and the set of active users have largely predictable and not too rapidly changing characteristics; under these conditions, the *manual* (or *static*) approach to load balancing, which consists of distributing the users over the available hosts (one of them being defined as the "usual host" of each user), may be quite successful. Since no workload is absolutely constant in its volume and characteristics, it will be necessary to rebalance the loads periodically; that is, we will have to retune the system when its operating point has gone far enough from the point of balance. However, a large fraction of the LAN-based installations are characterized by a workload so dynamic that the maximum frequency at which the load can be rebalanced manually is too low for manual rebalancing to be effective. As in the case of the bottlenecks that are found in centralized systems, the initial balancing and periodic retuning with the manual approach may be useful; however, shorter-lived bottlenecks, whose impact on performance grows with the width of the workload's frequency spectrum, can only be eliminated by *automatic* (or *dynamic*) approaches [1]. In principle, providing the users with such mechanisms as a *remote login* and a network-wide load reporting command allows them to give their individual contributions to the balancing of the loads. This may, however, be ineffective and even counterproductive, due to the frequency of the possible interventions, which may still be too low, and to the necessarily limited and incomplete information that each individual user has about the system's state. Thus, for a large number of installations, dynamic load balancing is required to improve resource usage and performance.

This paper discusses the problem of selecting the load index or indices to be used in dynamic load balancing policies, and proposes one such index. Section 2 defines the load balancing schemes the rest of the paper refers to. The criteria for load index selection and the assumptions to be made are presented in Section 3. The specific index we propose in this paper is introduced in Section 4, together with the arguments for its choice. Section 5 discusses the advantages and the drawbacks of the new index when applied in a load balancing scheme, and outlines the work that remains to be performed before the index can be accepted as a practically viable one.

2. Dynamic load balancing policies

Since the terminology used in the load balancing literature is still fluid, we must provide our definitions of the terms we will use, and clarify, by introducing a classification of schemes as well as the various assumptions we are making, the scope of our investigation. This will be done both

in this and in the next section.

First, we notice that the two terms "load balancing" and "load sharing" very often appear in the literature with the same meaning. One could easily introduce a distinction between them based on the different meanings the terms "balancing" and "sharing" suggest: for instance, use of the term "balancing" could be restricted to those schemes whose objective is to keep the loads on the machines within a relatively narrow band around the instantaneous average, whereas "sharing" could refer to those schemes in which a machine sends some of its load away (or accepts some of the load of other machines) only when its load goes beyond an upper threshold (or falls below a lower threshold). In both types of schemes the decision-maker must know the load existing on the machine being considered; in the load balancing ones, also the current average system load [2] must be known. Note that, however, though drawing a distinction between "balancing" and "sharing" may be useful in certain contexts, both types of schemes are dealt with in the same way in this paper. We therefore use "load balancing" as a generic term encompassing both, even though our primary objective in selecting a load index is not that of equalizing the loads (we shall indeed see that, with our approach, this objective would be meaningless).

It is also useful to distinguish *preemptive* from *non-preemptive* load balancing schemes. In the former, a running process may be suspended and migrated to a remote machine, where its execution will resume from the point of suspension. A non-preemptive scheme is one in which a process is assigned to a machine before beginning its execution, and cannot be moved to another after its execution has begun. We shall usually refer to non-preemptive schemes in the sequel, though most of our considerations apply to the preemptive ones as well. In a non-preemptive scheme, the *local machine* is the machine at which a given process entered the system.

Another classification of load balancing schemes that is sometimes useful is the one based on the identity of the machine which takes the initiative. When the scheme involves a centralized controller that makes the placement decisions, this controller can be thought of as the initiative-taker as well as the decision-maker. In other schemes (the so-called "sender-initiated" ones [3]), the local machine takes the initiative when a new process is to be executed or when its load has gone beyond the upper threshold. In the "receiver-initiated" schemes [3], instead, underloaded prospective receivers take the initiative of broadcasting information about their enviable state so as to attract currently running or soon-to-arrive new processes.

Thus, the initiator will have to select senders and receivers in the centralized controller case, one of the eligible receivers in sender-initiated schemes, and one of the eligible senders in receiver-initiated schemes. This selection can be either load-independent or load-dependent.

Among the *load-independent* policies are Random [4] or Random Splitting [5], which chooses at random the destination of a process to be executed remotely; Fixed Destination or Fixed Source [5], which, in the sender-initiated version, statically binds a given receiver to a given sender or group of senders, or, in the receiver-initiated version, a given sender to a given receiver or group of receivers; Cyclic Splitting or Cyclic Service [5], in which the destination (respectively, the source) is selected according to a cyclic ordering of the available machines; and Proportional Branching [6], a probabilistic policy which selects destinations or sources according to probabilities proportional to their processing speeds.

Examples of *load-dependent* policies include Lowest Load [6,7,8], which selects the machine with the smallest load at the time the decision is to be made; Shortest [4], which does the same but restricts its search to a randomly chosen subset of the eligible machines; Threshold [4], which probes randomly selected machines to determine whether adding a process would raise their loads above a given threshold; Broadcast When Idle or Poll When Idle [7], a receiver-initiated policy in which a lightly loaded machine invites or polls the others, so that only those prospective senders whose loads are heavier than a threshold will ship processes to the initiating machine (note that, if priority is given to the most heavily loaded machine, we obtain the Highest Load policy, the receiver-initiated analog of Lowest Load); Neighbor Pairing [9], which shifts load from the more heavily to the less heavily loaded machine in dynamically defined pairs of machines; and the policy used in the MOS operating system [10], which cyclically selects machines among the lightly loaded ones in a subset of the eligible machines.

While in load-dependent policies the load of each machine in the system is to be measured, and its value known by at least some of the possible decision-makers, load-independent policies do not have such a requirement. However, as was observed at the beginning of this section, at the very least a policy must rely on the knowledge of the load on the local machine, as moving processes is not a zero-cost operation and should be done only when necessary. Thus, the load of each machine must be measured, i.e., quantitatively expressed, in all cases. What index (or indices) should be used to measure a machine's load?

3. A criterion for load index selection

Many indices have been explicitly or implicitly used in the load balancing literature to express the load existing on a machine at a given time. Examples of such load indices include the utilization of the CPU, the length of the ready queue (in UNIX† terminology, the "load average"), the stretch factor (defined as the ratio between the execution time of a process on a loaded machine and its execution time on the same machine when it is empty), and more complicated functions of these simple variables. However, to the author's knowledge, a scientific justification for the choice of a load index has never been given. In fact, some papers even ignore the question altogether, and simply refer to "the load" as if a universally accepted definition of this term had been known for a long time. Is any of the indices used in the literature a correct one? Which? And what does it mean for a load index to be "correct"?

To simplify our discussion, we shall assume that the object to which load balancing applies is the *interactive user command*, as represented by the typing in of a command line or equivalent action. In other words, the execution of a command will be considered atomic from the load balancing viewpoint, even when a command causes the creation and execution of several processes that, in principle, could be executed on different machines. This assumption is made to facilitate the description, but is not essential for the application, of our approach.

An assumption that is essential is the choice of the command's response time as the performance index to be minimized by the load balancing scheme. This is another area in which some of the papers on load balancing leave something to be desired: the objectives of the schemes proposed therein are sometimes not clearly specified. Our choice of the response time objective is certainly questionable, since the exact relationship between it and such system-wide objectives as throughput maximization are unknown, but (we believe) not unreasonable.

Under these assumptions, a correct load index li must be such that the relationship between the response time rt of a command and the index is represented by a single-valued curve. In other words, rt must be a function of li. The reason for this condition is obvious: if the function $rt(li)$ for a given machine, a given configuration, and a given command is known, the value of li at the time a decision is to be made can be used to predict the response time that the command will have if it will be sent to that machine. The predicted response times for all the eligible machines (including the local one), adjusted for the expected communications delays due to the shipment of the command, of the output, and possibly of the files it needs to execute, can be compared by the decision-maker in order to determine the machine which is likely to process the command in the shortest possible time. Note that this condition, even though it seems to be not very restrictive, is not generally satisfied by any of the indices proposed in the literature. Simple experiments performed by the author have shown that the curves relating rt to CPU utilization, ready queue length, stretch factor, and other indices are multi-valued ones for at least some types of commands.

If more restrictive conditions are imposed, the selection of the least loaded machine can be made more efficient. For instance, in a system consisting of identical machines, a monotonically non-decreasing $rt(li)$ function would allow the decision-maker to restrict the choice to the local

† UNIX is a trademark of AT & T Bell Laboratories

machine and the remote machine with the smallest value of li, and to compute the function only for these two machines. Furthermore, if the monotonic function has a known minimum slope or, even better, is linear, then all comparisons will involve only the values of li, and no computation of rt will have to be performed. These observations can be extended to heterogenous systems, but will apply only to each group of identical machines within them.

Note that our choice of command response time minimization as the objective of load balancing makes it impossible to define the load of a machine as a command-independent quantity. This means that our answer to the question: "How much load is there on this machine?" will be another question: "For what command?". Thus, our approach does not help when the policy is load-independent, unless a standard "basket" of commands is defined to be used in the computation of a command-independent load index for each machine. Also, our approach does not provide help in making command migration decisions when the policy is preemptive, again unless the idea of a basket of commands is adopted.

4. A load index based on a mean-value equation

In this section, we shall propose a load index which, under certain assumptions, satisfies the criterion introduced in the previous section. Section 5 outlines a possible implementation of a scheme based on the index, and discusses some of the index's properties and potential drawbacks.

Consider a machine M, a command A, and a mix of commands B. Among all the possible loads that M may be processing, consider the following:

(A) command A runs alone on M;

(B) mix B (the background load) runs on M;

(C) the combination of A and B runs on M.

Our problem can now be expressed in these terms: predict the response time of A when load C is running on M from the knowledge of A and of the background load B (the load that was there just before the arrival of A).

We make the assumption that machine M can be accurately modeled by a closed queueing network model having:

(i) R chains;

(ii) L service centers 1, 2, . . . L of the FCFS, PS (processor sharing), LCFSPR (last-come-first-served-preemptive-resume), and IS (infinite servers) types [11]; center 1 is an IS-type service center representing user terminals;

(iii) a fixed number of customers (i.e., commands) in each chain;

(iv) service rates independent of the number of customers at the respective centers;

(v) FCFS centers that are all single-server centers.

The three loads A, B, and C can be modeled as follows:

(a) command A is the only customer in chain 1;

(b) the commands in load B are clustered, and each cluster is represented by one of the chains 2 through R (R is set equal to the number of clusters of B plus 1);

(c) load C is represented by chains 1 through R.

Under these assumptions, the mean-value equation of Corollary 1 of [12] holds for each non-IS center l in such a model:

$$w_{r,l}(\mathbf{K}) = \tau_{r,l}[1 + n_l(\mathbf{K} - \mathbf{e}_r)], \tag{1}$$

where

$\mathbf{K}$ = population vector (k_r = population size of chain r),

$\mathbf{e}_r$ = R-dimensional unit vector in direction r,

$w_{r,l}$ = mean time spent by a chain r customer at center l at each visit,

$\tau_{r,l}$ = mean service time per visit of a chain r customer at center l,

$n_l(\mathbf{K} - \mathbf{e}_r)$ = mean number of customers (mean queue length) at center l in the same queueing network with one less customer in chain r.

Note that, for an IS center, we have

$$w_{r,l} = \tau_{r,l}. \tag{2}$$

If the model includes IS centers other than center 1, the corresponding n_l will be defined to be 0.

Denoting by $rt_r(X)$ the mean response time (i.e., the mean time spent outside service center 1) of a chain r command under load X, and by $v_{r,l}$ the mean number of visits a chain r customer makes to service center l, we can write for command A

$$rt_1(A) = \sum_{l=2}^{L} v_{1,l}\tau_{1,l}, \tag{3}$$

and for command C

$$rt_1(C) = \sum_{l=2}^{L} v_{1,l}w_{1,l}(\mathbf{K}). \tag{4}$$

Substituting (1) into (4), and using (3), we obtain

$$rt_1(C) = \sum_{l=2}^{L} v_{1,l}\tau_{1,l} + \sum_{l=2}^{L} v_{1,l}\tau_{1,l}n_l(\mathbf{K} - \mathbf{e}_1) =$$

$$= rt_1(A) + \sum_{l=2}^{L} v_{1,l}\tau_{1,l}n_l(B), \tag{5}$$

since $\mathbf{K} - \mathbf{e}_1$ represents load C with one less customer in chain 1, i.e., with no customers in that chain; in other words, it represents load B.

By (2), the increase in command response time can be written as:

$$\Delta rt = rt_1(C) - rt_1(A) = \sum_{l=2}^{L} v_{1,l}w_{1,l}(A)n_l(B). \tag{6}$$

Thus, under the assumptions made, the response time of a command A is a linear combination of the mean queue lengths at the non-IS centers under load B, the coefficients being the total times being spent by A in the respective centers when running alone on the same machine. Note that, in the terminology adopted here, the queue length at a center also includes the customer in service at that center. Furthermore, note that IS centers do not contribute to the sum on the right-hand side of (6).

Equation (6) can be used to predict $rt_1(C)$. Its right-hand side is a load index satisfying the condition discussed in Section 3. Furthermore, rt is a linear function of the index, which, in turn, is a linear function of the mean queue lengths.

5. Characteristics of the new index

The load index introduced in the previous section is a linear combination of queue lengths: thus, the two ingredients that are needed to compute it for each eligible machine are the queue lengths and their coefficients, which are the total times spent by the command in the respective servers when there is no background load on the machine. The question of what queues or servers ought to be considered does not have a direct answer. Ideally, the servers should be those appearing in an accurate product-form closed queueing network model of the machine; in practice, they might be chosen to correspond to the points in the machine at which the execution of the command involved could be suspended because of the presence of a background load; the service times to be used in the measurement or computation of the coefficients will then equal the durations of the execution intervals between two consecutive potential suspension points when there is no extra load.

Instantaneous queue length measurements are not difficult to perform. They could be gathered periodically (or on demand) and broadcast (or sent to the requesting machine). Whether instantaneous or smoothed queue lengths should be used is not clear, and can only be decided after careful experimentation; however, we note that smoothing can be easily performed when instantaneous values are available. The contribution of each machine to the value of the load index would thus be a vector [13]. That of each command would be another vector, with components equal to the times spent by the command in each server when running alone. Clearly, each command must be represented by different vectors of coefficients for different machines or even different configurations. This characteristic should not, however, be regarded only as a drawback: in fact, the ability elegantly to adapt to configurationally and architecturally heterogeneous networks is a major advantage of the load index introduced in this paper.

The dependence of the value of the index on the particular command being considered is a very simple one, and the coefficients that characterize each command are easy to measure. However, command dependence causes two problems:

(i) the *absolute* load of a machine cannot be defined; this problem can be alleviated, as noted in Section 3, by referring to a standard workload (a "basket" of commands); in any case, the index only measures the load *relative* to a given command or mix of commands;

(ii) the coefficients characterizing a command generally depend on the command's arguments (input files, input data, options, and so on); thus, an accurate characterization of a command type may turn out to be very complex and very expensive to build; also, any modifications to the command's code may cause appreciable changes in the values of the coefficients.

Problem (ii) is a very serious one, and needs to be investigated, as its satisfactory solution is an essential requirement for the practical applicability of the index. Our hope is that, for most important commands, the dependence of the coefficients on the arguments can be approximated by functions with simple mathematical forms, and that their sensitivity to changes is low. For instance, we conjecture that the coefficients (or at least some of them) of text processing and compilation commands are roughly linear functions of the size of the input file. These and other similar conjectures will, however, have to be validated by an extensive study which is being planned

now.

Another question that needs to be addressed is the one concerned with the realism of the class of models on the basis of which our load index has been defined: how accurately can a real machine be represented by a queueing network to which the mean-value equation (1) in Section 4 applies? One of the most restrictive assumptions we made in Section 4 was that service rates were independent of the number of customers present in the server. This assumption is restrictive because population-dependent service rates often result from the application of flow-equivalence approximation methods, which can reduce a wide variety of non-product-form queueing networks to product-form networks. Relaxing this assumption, however, would make the load index much more complicated, as the mean-value equation in this case (see Theorem 1 of [12]) involves the marginal distributions of the queue lengths, and not only their means. An observation being taken into serious consideration in designing the experiments with the new load index is that the absolute accuracy of the mean-value equation (1) is much less important than the robustness of the relative results. That is, when using the load index in a load balancing scheme, the crucial questions are whether the least loaded machines can be identified by looking at the values of the load index, and whether the differences among their load index values are in the same ballpark as those among the response times the command being considered would exhibit if it ran on those machines at that time. Again, we conjecture that this condition is more likely to be satisfied than the one stating that for each machine an accurate model to which the mean-value equation (1) applies can be built; experiments are being designed and will be performed to verify this conjecture.

Even more fundamentally, one must wonder whether an equation that is valid for a model in steady state can be taken as the basis for the definition of a load index to be used in a highly dynamic context. We have already encountered this problem in our discussion about whether the values of the queue lengths ought to be instantaneous or smoothed. Again, experimentation (both empirical and simulation-based) is to be resorted to in order to obtain a reliable answer.

6. Conclusion

A load index to be used in load balancing schemes has been presented. The index is based on a mean-value equation that applies to closed multichain queueing network models with population-independent service rates and PS, LCFSPR, IS, or single-server FCFS service centers. The objective that has guided the choice of the index is the minimization of a command's execution time. The proposed index is a linear combination of the mean queue lengths in the machine being considered, the coefficients being the total times the command would spend in each service center if it ran alone on the machine. Use of the index in a heterogeneous distributed system seems to be easy, but the dependence of a command's coefficients on the command's arguments, and the possible incorrectness of the results the index will produce, that may be due to the inaccuracy of the type of model on which the index is based or to the steady-state assumptions under which the mean-value equation holds, are the main obstacles to practical use, and are being investigated now.

References

[1] D. Ferrari, G. Serazzi, and A. Zeigner, *Measurement and Tuning of Computer Systems.* Prentec-Hall, 1983.

[2] A. Barak and Z. Drezner, "Distributed Algorithms for the Average Load of a Multicomputer," Tech. Rept. CRL-TR-17-84, Computing Research Laboratory, University of Michigan, March 1984.

[3] D. L. Eager, E. D. Lazowska, and J. Zahorjan, "A Comparison of Receiver-Initiated and Sender-Initiated Dynamic Load Sharing," Proc. 1985 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems (August 1985), 1-3.

[4] D. L. Eager, E. D. Lazowska, and J. Zahorjan, "Dynamic Load Sharing in Homogeneous Distributed Systems," Technical Report 84-10-01, Department of Computer Science, University of Washington, Seattle (October 1984).

[5] Y.-T. Wang and R. J. T. Morris, "Load Sharing in Distributed Systems," *IEEE Transactions on Computers C-34*, 3 (March 1985), 204-217.

[6] Y.-C. Chow and W. H. Kohler, "Models for Dynamic Load Balancing in a Heterogeneous Multiple Processor System," *IEEE Transactions on Computers C-28*, 5 (May 1979), 356-361.

[7] M. Livny and M. Melman, "Load Balancing in Homogeneous Broadcast Distributed Systems," *Proc. ACM Computer Network Performance Symposium* (April 1982), 47-55.

[8] J. A. Stankovic, "Decentralized Control of Job Scheduling," *IEEE Transactions on Computers C-34*, 2 (Feb. 1985), 117-130.

[9] R. Bryant and R. A. Finkel, "A Stable Distributed Scheduling Algorithm," *Proc. 2nd International Conference on Distributed Computing Systems* (April 1981), 314-323.

[10] A. Barak and A. Shiloh, "A Distributed Load Balancing Policy for a Multicomputer," Internal Report, Department of Computer Science, The Hebrew University of Jerusalem, 1984.

[11] F. Baskett, K. M. Chandy, R. R. Muntz, and F. G. Palacios, "Open, Closed, and Mixed Networks of Queues with Different Classes of Customers," *Journal of the ACM 22*, 3 (April 1975), 248-260.

[12] M. Reiser and S. S. Lavenberg, "Mean-Value Analysis of Closed Multichain Queueing Networks," *Journal of the ACM 27*, 2 (April 1980), 313-322.

[13] S. Zatti, "A Multivariable Information Scheme to Balance the Load in a Distributed System," Rept. No. UCB/CSD 85/234 (PROGRES Rept. No. 85.6), University of California, Berkeley, May 1985.