VICTORIA UNIVERSITY OF WELLINGTON

Te Whare Wananga o te Upoko o te Ika a Maui

School of Mathematical and Computing Sciences

Computer Science

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341, Fax: +64 4 463 5045
Email: Tech.Reports@mcs.vuw.ac.nz
http://www.mcs.vuw.ac.nz/research

A Compositional Classification For Load-Balancing Algorithms

K. Bubendorfer and J. H. Hine

Technical Report CS-TR-99-9
31 July 1998

Abstract

Load distribution seeks to improve the performance of a distributed system by allocating the workload amongst a set of cooperating hosts. Over the years a diverse collection of approaches for a variety of systems have been proposed. To assist the understanding of these approaches this paper presents a new classification method for load distribution systems that emphasises the separation of policy and mechanism components. The method clearly highlights a system's essential features in a way that supports analysis and comparison of different systems.

The classification identifies three key policy decisions: participation, location and candidate selection; and three key mechanisms: transfer, load metric and load communication. The paper analyses each of these presenting examples from existing systems, showing how the individual components relate to overall designs.

Key words: load distribution, classification.

1 Introduction

Load distribution seeks to improve the performance of a distributed system by allocating the workload amongst a set of cooperating hosts. Such systems may attempt to ensure the workload on each host is within a small tolerance of the workload on all other hosts (load balancing), or may attempt to avoid congestion on individual hosts (load sharing or levelling). Over the years a number of systems for load distribution have been proposed.

The variety of designs are diverse, ranging from algorithms for tightly coupled multiprocessor systems to wide area distributed systems. With this diversity, a method for classifying the various designs is extremely valuable. A good classification should be able to highlight a load distribution system's essential features in a way that supports analysis and comparison of different approaches. In addition, a classification should clearly separate policy decisions from the mechanisms chosen to support those policies. Classification also has the additional benefits of providing consistent terminology and improving the organisation of knowledge to assist in the identification of further research opportunities. In this paper we present a new classification that meets these goals.

The classic work in this field is [CK88], which presents a taxonomy for classifying individual load distribution systems. Our method improves on [CK88] by providing clear separation of policy and mechanism and providing a basis for the comparison of the major components of a load distribution system.

The following section briefly surveys previous load distribution classifications. This is followed in section 3 by an overview of our classification scheme, which is analysed in detail in section 4.

2 Previous Load Distribution Classifications

Much of the variety in load distribution systems is due to the pragmatics required to deal with various architectures. Consequently, many papers have introduced, at least to a limited extent, an attempt to classify past work and introduce terminology to foster constructive comparison and discussion of its own contribution. This led to the confused state that this field finds itself in today. The lack of rigorous, standard terminology makes comparison difficult and has resulted in incorrectly labelled algorithms. A detailed analysis is often required to determine the precise contribution of a paper and of the algorithms therein.

This section outlines the significant approaches to the classification and understanding of load distribution algorithms, beginning with the notable taxonomy presented in [CK88]. Classifications prior to this tended to be incomplete, and for that reason we limit our discussion to subsequent work. Referral to the reference sections of the papers discussed below provides a good survey of earlier work.

2.1 A Classification Taxonomy

Casavant and Kuhl [CK88] presented a pivotal work on load distribution classification based on strategic design choices, such as static or non-static and distributed or non distributed. Their taxonomy combined these decisions in a hierarchical classification augmented with a flat classification. The division between the hierarchical and flat classifications is due to characteristics that do not fit uniquely under any particular branch of their hierarchy, but nonetheless were important for describing the behaviour of the load distribution system.

However, classes in the flat classification such as bidding and the choice between initial-placement or process migration, are inherently different from those in the hierarchy. In the most part these characteristics defined by the flat classification are techniques, rather than choices representing load distribution design strategies.

The classification was designed to place a load distribution system into a well defined class. The authors demonstrated this by determining the true class of several published algorithms as opposed to the class implied by the title of the paper. The classification however, does not assist in comparing the various component algorithms of different load distribution systems.

Baumgartner and Wah [BW90] identified a degree of confusion with the taxonomy presented in [CK88]. They claimed the confusion arose because the strategy and problem classifications were interwoven rather than separate. Their solution was to take the original taxonomy and separate out the problem classification into a distinct classification of its own termed the ESR classification scheme (Event, Surroundings (environment) and Requirements). The remainder of the original classification formed a solution classification.

The ESR classification includes a number of mechanisms within the problem classification. For instance, the authors suggest that the choice between remote execution and process migration belongs within S as a capability of the environment. We argue that this is a strategy decision, as the preference between remote execution or process migration is in no way clear [Bub96], and the mechanism should reflect the requirements of the policies, not the limitations of the environment. There is also limited scope for describing the granularity of the load distribution algorithm under R (requirements).

2.2 A Functional Classification

A contrasting, but more specialised classification was proposed by Jacqmot and Milgrom [JM93]. They identified two significant and orthogonal features common to load distribution systems; process movement (PM) and information handling (IM). For each of these two *strategies* they isolated basic functional units, such as identify_source_node (under the PM strategy), that combine to form families (classes) that can describe a load distribution system's PM or IM. These families are constructed based on the order in which individual functional units are applied by the load distribution system. For example, under the PM strategy if the candidate process (and therefore implicitly the source node) is identified before the destination node, then it is in a different PM family to one which selects the target node before identifying a candidate process (and implicitly the source node).

A forerunner of this work is [WM85] which was concerned with the same two broad categories, the order of actions and the information required to enact load distribution policies, there is however no decomposition into functional units.

A problem with both of these functional classifications arises with symmetrically initiated systems such as PSI presented in [BMD94]. In this case load distribution is both sender and receiver initiated and cannot be accommodated by any family in [JM93] or by [WM85]. In addition the majority of current load distribution systems are sender initiated and invariably identification of the source proceeds the destination. As a consequence these classifications provide little distinction between the members of this large class.

2.3 Summary

In this section we have outlined the current state in the classification of load distribution systems. Space does not permit a complete survey of all work in this area, there are however other approaches that are worthy of recognition. Notably, [JV88] introduced a classification based purely on characteristics such as transparency, cost and preemption. Another more recent approach detailed in [LMR92] introduced a taxonomy based on the scope of decisions and migrations, i.e. whether a decision or migration involves only directly connected neighbours (local) or all nodes (global).

We see that previous attempts to classify load distribution systems have failed to meet the proposed criteria for providing a common framework which: clearly separates policy and mechanism components; facilitates the discussion and comparison of different component algorithms; and assists the understanding of the design of overall load distribution systems. The following sections present an alternative methodology that directly addresses these criteria.

3 A Compositional Classification

Load distribution systems are sometimes presented in terms of *policies* and *mechanisms*. Policies are the set of design choices made in deciding the goals of the load distribution system. Mechanisms carry out the physical distribution of load and provide any information required by the policies. The load distribution taxonomies discussed above all fail to provide a means for discussing and summarising load distribution systems that effectively separates policy decisions from mechanisms.

By focusing on the individual decisions made during load distribution, the following classification identifies the separate components of a load distribution system as policy or mechanism. The division of policy and mechanism, illustrated in figure 1, decomposes a load distribution system into six distinct components.

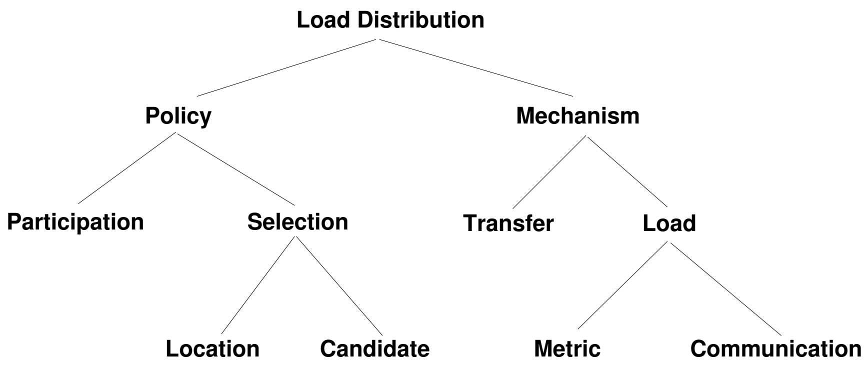

Figure 1: *Load Distribution Functional Taxonomy*

Clearly identified are three policy decisions that must be made: participation, location selection and candidate selection. Also identified are three key mechanisms that must be designed to support the policies: workload transfer, load metric and load communication. Each of these is briefly described below. Section 4 then discusses each in more detail providing a further refinement of the taxonomy.

Participation Policy: The participation policy determines whether or not a host is taking part in load distribution.

Location Selection Policy: The location selection policy is responsible for selecting the pair of participating hosts, between which a unit of load (job, process or object) will be transferred. This policy is also responsible for handling any heterogeneity issues among the participants.

Candidate Selection Policy: The candidate policy selects the jobs, process, or objects to be distributed.

Load Metric Mechanism: The load metric is the method used to measure the load on a server and the load imposed by a candidate on its host.

Load Communication Mechanism: Load communication defines the method by which information, such as the load on a host, is communicated between the host and the load distribution policies. The load communication mechanism may also include the communication between cooperating distributed policies.

Transfer Mechanism: The transfer mechanism is the protocol employed to transfer jobs or processes between hosts.

4 Analysis

The taxonomy outlined above is designed to aid the comparison of different load distribution systems rather than to produce a specific classification of a system. Any system will involve decisions (perhaps implicit) with respect to most of the six leaves shown in figure 1. Such decisions will not always be independent of each other. For example, the choice of location and or candidate selection policies may, as a consequence of the information required, determine the load metric. Similarly, some policies will not require all mechanisms. For instance a random selection policy does not require a load metric or a communication mechanism, however the associated participation policy may.

Each component of a load distribution system represented in the taxonomy could be dependent on the system for which the load distribution is intended, or may simply be a choice made for performance reasons or personal preference. All can however, exhibit a wide variety of interesting approaches and unique solutions due to the large design space. We now explore each component in greater depth.

4.1 Policies

The following looks at the three principal policy decisions that must be made by any load distribution scheme. These decisions determine which hosts are participating in the scheme and which hosts and candidate processes are involved in the distribution of load. The discussion includes examples of various ways in which these decisions may be made. We conclude with a look at how the different decisions combine to produce an overall design for a load distribution system.

4.1.1 Participation Policy

Participation policy decides if any particular host is to be involved in a load distribution activity. The reasons for participation can be varied. The following presents two common criteria for participation policy decisions, thresholds and ownership. Other criteria may also be used, e.g. security [PM83].

A threshold participation policy determines if a machine is over or under loaded, by considering whether the machine's current load is above or below a preset threshold. The most common form of this is in the guise of a two threshold policy - one threshold defines the point at which a host begins to offload excess load and the other threshold determines the point at which an overloaded host may refuse incoming load. Such a policy results in separate decisions as to whether a host will participate by offloading excess work or participate by accepting work from other hosts. One example of this approach is Holzer [Hol96].

Ownership of a personal workstation is recognised in some systems as the basis of the participation policy. Sprite [DO91] workstations will only accept work from other hosts if the (console) owner is inactive. On the other hand, workstations are always prepared to offload work to inactive workstations. Upon detection of activity from the owner - the participation is revoked and foreign processes are evicted.

4.1.2 Location Selection Policy

The location selection policy is responsible for choosing the source and or destination hosts between which load is to be transferred. This decision may be made centrally or independently at participating hosts.

When the decision is made centrally, a single agent is responsible for deciding both the source and destination hosts. This agent uses information about the loads on participating hosts to determine the source and destination and initiate a load transfer between them.

A good example is Condor [BLL92]. There is a single central agent and a local agent on each host. The central agent is responsible for sharing the resources of the system among participating hosts. The local agent is responsible for determining participation, based on the activity of the owner, and providing information about the load. When the central agent discovers an idle host B, it selects a host A which has excess load. The central agent then notifies the local agent on A, giving it permission to execute processes on B.

A variation on the centralised approach may be characterised as dictatorial. This approach is used by Harchol-Balter and Downey [HBD95]. A central agent periodically identifies the most and least loaded hosts and orders a migration between them. There are no participation constraints, and a migration is avoided only if there are no suitable candidate processes on the selected source.

As an alternative to centralised decision making, it is possible for hosts with either heavy or light loads to act independently and seek out a partner for load distribution.

Sender Initiated is a form of load distribution [Ber85, Rub87, DO91] which occurs when a host finds that its load is too high and initiates action to identify a candidate to accept its excess load. In this case, the source host is fixed and the location policy is only required to identify a destination host which will participate. The actual policy may be to locate the least loaded host, or a random idle host from a pool of participating hosts.

Receiver Initiated is the opposite approach and is much less common. In schemes such as the broadcast idle (BID) and poll when idle (PID) algorithms from Livny and Melman [LM82] and [LO86] respectively, a host finding itself lightly loaded initiates action to seek additional load from other hosts. In this case, the destination host is fixed as the initiating host, and the location policy is used to find a source host from which load may be transferred. Livny and Melman's BID algorithm selects the host with the greatest load while their PID algorithm selects a random busy host.

Eager et al. [ELZ86] compared sender and receiver initiated location policies and concluded that sender initiated policies are superior during light to medium system loads and receiver initiated policies are more effective at high system loads. They proposed a *symmetrically initiated* policy to take advantage of the different characteristics exhibited at high and low loads.

Such an approach was taken in [BMD94] in the Periodic Symmetrically-Initiated (PSI) Algorithm. If a host is overloaded, sender mode is initiated and a single random poll is employed to seek a destination for the excess load. If a host is underloaded, receiver mode is initiated and again a single random poll is used to seek a source of extra load. Krueger and Shivaratri [KS94] use a similar scheme that replaces the random poll with a list of potential partners.

4.1.3 Candidate Selection Policy

When a host participation policy triggers load distribution, a candidate needs to be selected. There are four identifiable strategies that each offer different levels of quality and cost tradeoff in selecting the candidate process.

Selecting Any Candidate either chooses one process at random for migration or selects the next job arrival for initial placement. It is clear that this form of selection may choose jobs or processes that are characteristically unsuitable for distribution [JXY95]. The most obvious subset of unsuitable processes are those which do not execute for a sufficient length of time to repay the overhead incurred in the remote execution.

Selecting a Reasonable Candidate eliminates those processes that are poor candidates for load distribution. The most common approach is based on the average runtime of each process - both static [Zho87, Hol96] and dynamic [Oss92] approaches have been used.

Selecting a Good Candidate is a tighter and more expensive criterion, as some form of matching between available resources on the destination host and the candidate's resource requirements is employed. Indeed, it is possible that the location selection policy is secondary to the candidate selection policy. This is true of Emerald [Leh93] which moves heavily communicating objects to reside on the same host. Systems relying on initial placement need the information about a job prior to it's assignment to a host machine. In this case good candidates can be selected using prediction or classification [Bub96]. By contrast, schemes which employ migration can obtain information about a current process's behaviour by monitoring each process as it executes and good candidates can be selected by referring to this behaviour [HBD95].

Self Selection is a compelling option for candidate selection, based on the idea that the process is better able to recognise it's needs than the system and therefore is in the ideal position to make the best choice. However candidate selection is just one of several policies, and the system must still consider the state of potential hosts. In the Sprite system application programs request remote execution, for example *pmake* uses idle workstations to implement a parallel version of *make*.

4.1.4 Discussion

We have looked at alternative policies that may be selected for each of the policy components of our classification scheme. We now look at how these combine to contribute to an overall design. First consider [Kar94] which sets out to achieve a high degree of balance between the hosts. The system is only considered balanced if the difference between the least and most loaded hosts is within a small limit Δ. This will require each host to carry, as nearly as possible, the same load. Such a system would: expect all hosts to participate; require a centralised location selection policy to ensure a complete picture of the load on all hosts; and would require good candidate selection to achieve the accuracy required.

By contrast we might consider a system such as [Hol96] which only attempts to avoid peak congestion. In this case a threshold policy is used to limit participation to heavily and lightly loaded hosts, a sender initiated policy to used to offload work and avoid congestion and any reasonable candidate is chosen.

4.2 Mechanisms

A range of mechanisms are available to support the various policies described in the previous section. While there has been considerable research into issues such as transparency, residual dependencies, performance and complexity, these mechanisms can be recognised as supporting one of three key roles: the transfer of load; the measurement of load; and the communication of measurements.

4.2.1 Transfer Mechanism

The transfer mechanism physically moves workload from source to destination host. Distribution of workload can occur in either of two phases: tasks can be allocated to a processor before they begin execution or they can be moved after they have begun execution. These two phases are known as *Initial Placement* and *Process Migration* respectively and are in principle independent.

Each offers different opportunities to the policy components of a load distribution system. Indeed several authors have used the choice between initial placement and process migration as a major defining component of a classification scheme. While the relative costs of these alternatives may influence a designer, we strongly believe that design choices should focus on policy, with mechanisms that support the policy subsequently chosen.

4.2.2 Load Metric

The term *load* describes the degree to which a systems collective resources are utilised - and is not a directly measurable quantity. The purpose of a load metric is to focus on a subset of resources that are good indicators of load for the intended policies, and allow different hosts to be compared using these terms. The major differences between load distribution metrics arise from the choices of resources measured and how the load is estimated.

Some resources can be measured directly - e.g., memory, while others need to be inferred, e.g., CPU utilisation is estimated by examining the length of the ready queue.

A delightfully simple metric was presented in [TLC85] and used the multicast facility of V. Participating hosts were polled and the first host to respond was assumed to be the least loaded. Thus, the round trip time was used to infer the load on all hosts in the multicast group. Obviously this metric provides limited information, enough to select at best a reasonable candidate.

Different load metrics can seek to isolate or combine measures of individual resources to estimate a load. One interesting example is Sprite [Dou90], where two independent load metrics are used to determine if a host is idle. An idle Sprite host must have had less than one runnable process, averaged over the last minute, and no keyboard or mouse input for the past 30 seconds. These are both required to satisfy Sprite's *ownership* participation policy.

Whether combined load metrics are better than individual resource measures will depend on the requirements of the policies. Ferrari and Zhou [FZ87] explored various load metrics ranging from the instantaneous CPU queue length, through to a linear combination of CPU, memory and IO queue lengths. They found that for their system and set of assumptions a linear combination of exponentially smoothed CPU, IO and memory queue lengths produced the best performance. In a contrary finding, Kunz [Kun91] found that single resource queue lengths were as good as a combined metric in his study with a stochastic learning automaton.

4.2.3 Load Communication Mechanism

Once the load on a host has been measured, it must be communicated to the agents making load distribution decisions. This poses a difficult problem in most distributed systems, as there is a cost involved in both the collection and distribution of the load data. There are also problems of reliability and consistency. A number of clever solutions have been proposed for this problem, most of which are variations on four basic approaches.

Polling can be employed to locate individual hosts on demand, resulting in the most up to date information - at a high cost. Examples of polling are found in [LM82] with the PID algorithm, [BMD94] and [KS94].

Broadcasting allows all hosts to exchange information without the costly overhead of individual polls. The other advantage is that all hosts on the network have access to the broadcast information which can be used to update recorded host loads and to synchronise a distributed load distribution policy. A good example of this is the BID algorithm implemented by Livny and Melman [LM82]. Unfortunately this approach may generate an unacceptable amount of network traffic and may be unacceptable if only a fraction of hosts are participating in the load distribution.

Multicasting is a more recent compromise that exhibits some of the selectivity of polling, while retaining many of the advantages of broadcasting. However this approach doesn't scale well as the number of participating hosts increases. The primary advantage of a group communication system is the immediate identification of participating hosts, making the location of suitable sources or destinations a much simpler problem. The group communication facility of the V system is also used as the basis for the load metric in [TLC85].

A collection agent uses a central point of collection rather than a distributed algorithm. Zhou [Zho87] identified three variations on this theme, which differ in how the load data is collected and when it it distributed. The simplest receives periodic updates from all participating hosts, and then broadcasts the accumulated load vector. The second variant piggybacks load information on requests to offload excess load, while the third variant combines periodic updates and piggy backing. This avoids the stale load data suffered in the second variant and reduces the required periodic update frequency. This last approach was used by Bond [Bon93] in his Distributed Resource Measurement Service (DRUMS).

5 Conclusion

Previous classification methods for load distribution systems have not clearly separated policy decisions from the choice of mechanisms used in the systems. We have presented a classification that shows how an overall system is composed of three policy decisions: participation; location; and candidate selection and three mechanism choices: transfer; load metric; and load communication. The classification clearly separates policy from mechanism.

We have examined alternative policies and mechanisms that have been used for the components in the classification scheme by previous designers of load distribution systems. We have shown that our classification scheme can be used to isolate and compare individual features of a wide range of load distribution systems using a common framework.

References

[Ber85] Brian Bershad. Load balancing with maitre d'. Technical Report CSD-85-276, University of California at Berkeley, December 1985.

[BLL92] Allan Bricker, Michael Litzkow, and Miron Livny. *Condor Technical Summary*. University of Wisconsin, January 1992.

[BMD94] K. Benmohammed-Mahieddine and P. M. Dew. A periodic symmetrically-initiated load balancing algorithm for distributed systems. *SIGOPS*, 28(1):66–77, January 1994.

[Bon93] Andrew Murray Bond. *Adaptive Task Allocation in a Distributed Workstation Environment*. PhD thesis, Victoria University of Wellington, 1993.

[Bub96] Kristian Paul Bubendorfer. Resource based policies for load distribution. Master's thesis, Victoria University of Wellington, August 1996.

[BW90] K. M. Baumgartner and B. W. Wah. Computer scheduling algorithms: Past present, and future. In *First Workshop on Parallel Processing*, pages 170–183, National Tsing Hua University, Hsinchu, Taiwan, December 1990.

[CK88] Thomas L. Casavant and Jon G. Kuhl. A taxonomy of scheduling in general-purpose distributed computing systems. *IEEE Transactions on Software Engineering*, 14(2):141–154, February 1988.

[DO91] Fred Douglis and John Ousterhout. Transparent process migration: Design alternatives and the sprite implementation. *Software-Practice and Experience*, 21(8):757–785, August 1991.

[Dou90] Frederick Douglis. *Transparent Process Migration in the Sprite Operating System.* PhD thesis, University of California at Berkeley, September 1990.

[ELZ86] D. Eager, E. D. Lazowska, and J. Zahorjan. A comparison of receiver-initiated and sender-initiated adaptive load sharing. *Performance Evaluation*, 6:53–68, March 1986.

[FZ87] Domenico Ferrari and Songnian Zhou. An empirical investigation of load indices for load balancing applications. Technical Report CSD-87-353, University of California at Berkeley, January 1987.

[HBD95] Mor Harchol-Balter and Allen B. Downey. Exploiting process lifetime distributions for dynamic load balancing. Technical Report UCB/CSD-95-021, Computer Science Division, University of California, Berkeley, May 1995.

[Hol96] Teijo Holzer. Performance measurement of task allocation in a distributed system. Master's thesis, Victoria Univeristy of Wellington, 1996.

[JM93] C. Jacqmot and E. Milgrom. A systematic approach to load distribution strategies for distributed systems. In *IFIP Transactions: International Conference on Decentralized and Distributed Systems.* ELSEVIER Science (North Holland), September 1993.

[JV88] W. Joosen and P. Verbaeten. On the use of process migration in distributed systems. Technical Report CW83, Department of Computer Science, Katholieke Universiteit, Leuven, November 1988.

[JXY95] Jiubin Ju, Gaochao Xu, and Kun Yang. An intelligent dynamic load balancer for workstation clusters. *SIGOPS*, 29(1):7–16, January 1995.

[Kar94] Mourad Kara. A global plan policy for coherent cooperation in distributed dynamic load balancing algorithms. Technical Report 94.21, University of Leeds, July 1994.

[KS94] P. Krueger and N. Shivaratri. Adaptive location polices for global scheduling. *IEEE Transactions on Software Engineering*, 20(6):432–444, June 1994.

[Kun91] Thomas Kunz. The influence of different workload descriptions on a heuristic load balancing scheme. Technical Report TI-6/91, Institut für Theoretische Informatik, Fachbereich Informatik, Technische Hochschule Darmstadt, December 1991.

[Leh93] Morten Lehrmann. Load distribution in emerald: an experiment. Master's thesis, University of Copenhagen, June 1993.

[LM82] Miron Livny and Myron Melman. Load balancing in homogeneous broadcast distributed systems. *Proceedings of the ACM Computer Network Performance Symposium*, pages 47–55, April 1982.

[LMR92] Reinhard Lüling, Burkhard Monien, and Friedhelm Ramme. Load balancing in large networks: A comparative study. In *Proceedings of the 3rd IEEE Symposium on Parallel and Distributed Processing (SPDP'92)*, pages 94–101. IEEE, 1992.

[LO86] Will E. Leland and Teunis J. Ott. Load balancing heuristics and process behaviour. *Performance Evaluation Review*, pages 54–69, May 1986.

[Oss92] William Osser. Automatic process selection for load balancing. Master's thesis, University of California at Santa Cruz, June 1992.

[PM83] Michael L. Powell and Barton P. Miller. Process migration in demos/mp. In *Proceedings of the Ninth ACM Symposium on Operating Systems Principles*, pages 110–119, October 1983.

[Rub87] Harry I. Rubin. The design of a load balancing mechanism for distributed computer systems. Technical Report CSD-87-362, University of California at Berkeley, July 1987.

[TLC85] Marvin M. Theimer, Keith A. Lantz, and David R. Cheriton. Preemptable remote execution facilities for the v system. In *Proceedings of the Tenth ACM Symposium on Operating Systems Principles*, pages 2–12, December 1985.

[WM85] Yung-Terng Wang and Robert Morris. Load sharing in distributed systems. *IEEE Transactions on Computers*, C-34(3):204–217, March 1985.

[Zho87] Songnian Zhou. A trace driven simulation study of dynamic load balancing. Technical Report CSD-87-305, University of California at Berkeley, September 1987.