Load Balancing, Load Sharing and Performance in Distributed Systems

by

Phillip Krueger
and
Miron Livny

Computer Sciences Technical Report #700
August 1987

Load Balancing, Load Sharing and Performance in Distributed Systems

Phillip Krueger and Miron Livny

Computer Sciences Department
University of Wisconsin
Madison, WI 53706

Abstract

A distributed scheduling policy for a general-purpose distributed system can be divided into two components: a *local scheduling discipline* determines how the CPU resource at a single node is allocated among its resident processes, while a *load distributing strategy* distributes the system workload among the nodes through process migration. Since several choices exist for each of these components, we are confronted with a wide variety of distributed scheduling policies. In this paper, we address the question: Are some distributed scheduling policies better than others, or do they have different objectives? We find that there is significant diversity in objective and that the choice of an appropriate policy requires consideration of the performance expectations of the users together with the system workload characteristics.

1. Introduction

Scheduling for distributed systems is significantly more complex than for single-processor systems. A distributed scheduling policy for a general-purpose system can be divided into two components: a *local scheduling discipline* determines how the CPU resource at a single node is allocated among its resident processes, while a *load distributing strategy* distributes the system workload among the nodes through process migration. Eager, Lazowska and Zahorjan [Eager86a] have noted that within the myriad load distributing algorithms proposed in the literature lie two distinct strategies for improving performance. *Load balancing* algorithms strive to equalize the workload among nodes, while *load sharing* algorithms simply attempt to assure that no node is idle while processes at other nodes wait for service. The strategic goal of load balancing is a superset of that of load sharing, since balancing the workload may require migrations even when no nodes are idle. In this paper, we address the question: Do these additional process migrations improve performance? Similar to load distributing, many choices exist for local scheduling disciplines, ranging from simple non-preemptive disciplines to those that are complex and preemptive. As a result of the diverse choices for each of its components, we are confronted with a wide variety of distributed scheduling policies. Generalizing our previous question: Are some of these policies inherently better than others, or do they have different performance

objectives? In addition to identifying, comparing and evaluating the applicability of the objectives of distributed scheduling policies, this paper provides a unique perspective of the interaction between load distributing strategies and local scheduling disciplines in determining these objectives.

Computer performance is not easily defined. The users of a computer system have certain performance expectations. The goal of a scheduling policy is to allocate resources in such a way that these expectations are most nearly met. It is important, then, that the performance objective of the scheduling policy matchs the expectations of the users as nearly as possible. However, just as there is no universal set of user expectations, no single performance objective is applicable to every system.

User performance expectations generally center on the quality of service provided to the processes they initiate. In addition to the average quality of service received, fairness is an important concern. Two users simultaneously initiating equivalent processes expect to receive about the same quality of service. Similarly, a user submitting the same job several times, under equivalent workloads, expects each to receive about the same quality of service. To ensure fairness, the variance in quality of service under a given workload should be acceptably low.

A second aspect of fairness relates to the way in which quality of service is measured. Both wait time and wait ratio are accepted measures of the quality of service received by a process. **Wait time** is the total amount of time a process spends waiting for resources, while **wait ratio** is the wait time per unit of service. Which measure is used carries an implied assumption about the importance of fairness. The use of wait time implies that the important factor in assessing quality of service is the absolute amount of time one waits for a resource, *regardless* of one's service demand. A person wanting to check out a book from a library and a person requesting a complete library tour would be considered to have received equal service if each waited the same amount of time for the attention of the librarian. The use of wait ratio implies that the important factor is the amount of time one waits for a resource *relative* to one's service demand. In providing equal quality of service, the person requesting an exhaustive library tour would be expected to wait longer for that service than the person wanting to borrow a book. The use of wait ratio, in preference to wait time, may be considered to allow a more fair comparison of quality of service received.

Finally, fairness may imply that scheduling is *non-discriminatory*: variation in quality of service received by processes should be strictly random. The correlation between the quality-of-service metric and service demand can be used to measure an important aspect of discrimination in scheduling.

While previous performance studies have often been limited in scope to mean wait time, we examine a broad range of performance indices, including the means and standard deviations of process wait times and wait ratios, and the correlations between wait ratio and service demand and between wait time and service demand. To identify the performance objectives of distributed scheduling policies, we examine performance in the absence of scheduling overhead. This idealized framework allows a scheduling policy to achieve its best-case performance bound, which defines its performance objective. How nearly this objective can be met under more realistic assumptions is dependent on the efficiency with which primitive scheduling operations, such as context switching and message handling for migration, are implemented and on system workload characteristics. The effects of these factors are examined in [Krueg87a]. In the following pages, we show that:

- Load balancing has a significantly broader performance objective than load sharing.
- Load balancing is effective under a broader range of workloads than load sharing. Moreover, two workload characteristics that significantly degrade the performance of load sharing relative to load balancing may be common in distributed systems.
- The choice of local scheduling discipline may be more important than the choice of load distributing strategy in determining performance.
- Migrations between nodes with queue lengths differing by one can be productive.

We conclude that the choice of an appropriate distributed scheduling policy requires consideration of the performance expectations of the users together with the system workload characteristics. No single policy is best for all systems.

2. Distributed System Model

The models used in this study are an extension of the m*(M/M/1) family of distributed system models proposed by Livny [Livny82a, Livny83a], augmented to allow the processor queuing discipline, or *local scheduling discipline*, to be specified as a parameter and to allow hyperexponentially, as well as exponentially,

distributed task service demands. It is important that hyperexponential service demand distributions be considered, as those that have been observed [Rosin65a, Trive82a, Zhou86a] are poorly approximated by exponential distributions. We categorize service demand distributions according to C_X (see table 2.1 for notation), with exponential distributions having $C_X = 1$ and hyperexponential distributions having $C_X > 1$.

The resulting m*(M/H/1) system, illustrated in figure 2.1, consists of m nodes, connected by a communication device. A load distributing algorithm allows processes to migrate between nodes at any time during their execution. Each of the m processing elements provides identical functional capabilities. Tasks arrive independently at each node and join the queue. The distribution of interarrival times is exponential, so the task arrival process of the system consists of m independent Poisson processes.

In this study, we assume that nodes have equal processing bandwidths and that the service demands of processes arriving at different nodes are identically distributed. The rates at which processes initially arrive (as opposed to arriving as the result of migration), however, may be different at different nodes. We refer to such a workload as having *inhomogeneous initiation rates*. Studying load distributing under such a workload is important, since such workloads may be common for some types of distributed systems, particularly those composed of workstations. Since we are interested in scheduling for general-purpose computer systems, we assume that

m	number of nodes
n	total number of resident processes
$\overline{N}$	mean number of resident processes
ρ	system load
x	service demand of a process
$\overline{X}$	mean process service demand
σ_X	standard deviation of service demand
C_X	coefficient of variation = $\sigma_X / \overline{X}$
$\overline{WT}$	mean process wait time
$\overline{WR}$	mean process wait ratio
σ_{WT}	standard deviation of wait time
σ_{WR}	standard deviation of wait ratio
$r(wait\ time, X)$	correlation between wait time and service demand
$r(wait\ ratio, X)$	correlation between wait ratio and service demand
E(Y)	expected value of random variable Y

Table 2.1 Notation

the scheduler has no deterministic a priori information about process service demands. In addition, we assume that processes do not leave the system before completing service.

With one final assumption, we arrive at the M/H/m-*like* system, on which this study is based. We assume that no work is associated with scheduling; both context switches and process migrations are instantaneous and without cost. The M/H/m-like system is the distributed-queue analogue of the M/H/m queue, from which it derives its name. As such, it provides a vantage-point for identifying the performance objectives of distributed scheduling policies. Within this idealized framework, a scheduling policy is able to achieve its best-case performance bound for a given workload. This best-case bound defines the performance objective of the scheduling policy.

To allow an M/H/m-like system to be *work-conserving* [Klein76a], its local scheduling discipline must not allow its server to lie idle while processes wait in the queue. Correspondingly, its load distributing strategy must guarantee that no node is idle while processes at other nodes wait for service. Such a system assures that

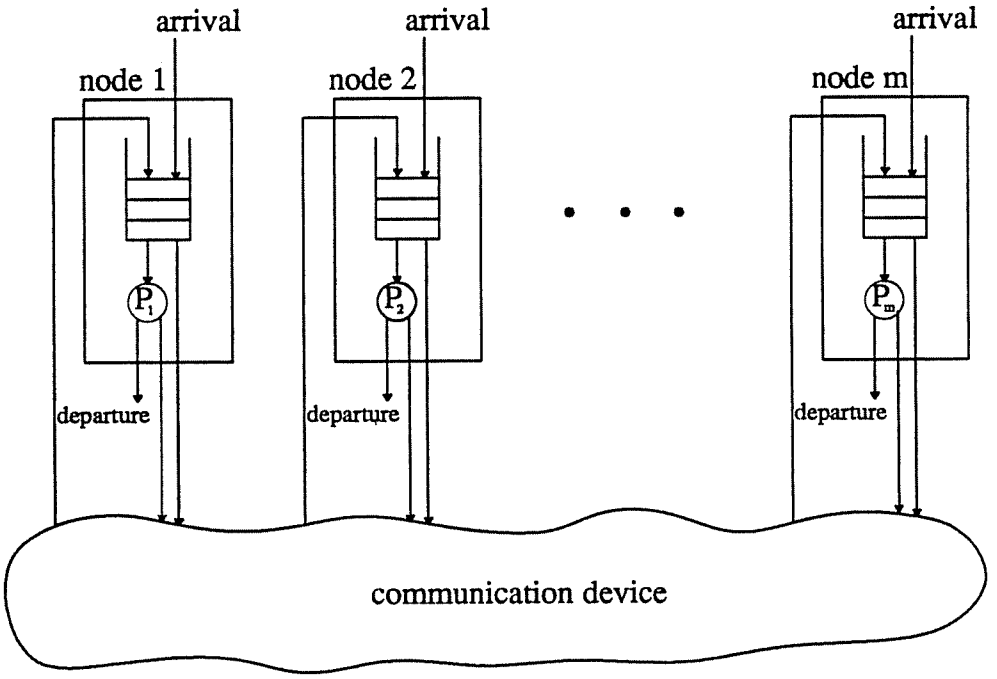

Figure 2.1 An m*(M/H/1) system

"no work (service requirement) is created or destroyed within the system" [Klein76a]. If scheduling is non-preemptive and $C_X = 1$, $\overline{WT}$ in a work-conserving system is [Laven83a]:

$$\overline{WT} = \bar{X} [(\bar{N} / m\rho) - 1] \qquad \text{for } \rho < 1 \tag{2.1}$$

Simulation results indicate that this equation also holds for all preemptive disciplines studied in this paper. Kleinrock sums up the lack of discriminatory power inherent in $\overline{WT}$ as a performance index: "One may therefore conclude that the average response time by itself is not a very good indicator of system performance" ([Klein76a] page 171).

3. Performance Objectives

Our approach to identifying the objectives of distributed scheduling policies is to separately identify the objectives of load distributing strategies and those of local scheduling disciplines. We then examine the performance of scheduling policies using combinations of these components. To simplify analysis, Round Robin scheduling is approximated by Processor Sharing (PS) [Klein67a]. For a description of the remaining local scheduling disciplines studied, the reader is referred to [Klein76a].

To begin, we note that in an M/H/m-like system, both the LB and LS strategies are work-conserving. None of the work potential of the system is wasted by leaving nodes idle while processes wait. However, unlike LS, LB goes beyond conservation of work. By balancing the workload among nodes, each process residing in the system perceives approximately the same level of contention. If the local scheduling discipline is PS, each resident process receives service at approximately the same rate. Thus, LB in an M/H/m-like system emulates PS in an M/H/m queue, with the goal of achieving similar performance. Essentially, LB is the distributed analogue of PS. LS, on the other hand, is the distributed analogue of any work-conserving M/H/m scheduling discipline.

This perspective of load distributing strategies allows us to predict the differences in performance between the LB and LS strategies in M/H/m-like systems through a less complicated study of performance in M/H/m queues, where many analytical results are available. In effect, the search for the performance objectives of these strategies is greatly simplified. These performance objectives are then validated through a study of performance in M/H/m-like systems. A second benefit of this perspective is that it allows us to identify a

new load distributing strategy, which we discuss in section 3.2.3.

Unfortunately, our second goal, identifying the objectives of local scheduling disciplines, is more complicated than identifying those of load distributing strategies. Even when the system arrival process is Poisson, load distributing causes the arrival processes at individual queues to be non-Poisson. Additionally, load distributing modifies the service demand distribution observed by individual servers. Local scheduling disciplines, then, operate on G/G/1 queues. Since some of our results for M/H/m queues also apply to G/G/1, we are able to identify some local scheduling objectives. The remaining predictions are validated through a study of performance in M/H/m-like systems. Performance is studied both analytically and through simulation, with simulation results having less than 3% error at the 95% confidence level.[1] Unless otherwise noted, results are from simulation.

3.1. Performance of Single-Queue Scheduling Disciplines

Since both LB and LS are concerned with conservation of work, with this being the sole goal of LS, we begin by considering work-conserving M/H/m scheduling disciplines. As analogues for LS, we consider non-preemptive M/H/m scheduling disciplines, since preemption gives no advantage in conserving work. When necessary, we specifically consider First-Come-First-Served (FCFS), since it is perhaps the simplest M/H/m scheduling discipline and has few goals beyond conservation of work. Simulation results displayed in figure 3.1 show that $\overline{WT}$ for FCFS increases with C_X,[2] as does σ_{WT}. In this plot, as well as all other plots of these indices, $\overline{WT}$ and σ_{WT} are normalized by $\bar{X}$. In their favor, work-conserving non-preemptive scheduling disciplines are non-discriminatory with respect to wait time; process wait times are independent of their service demands. However, non-preemptive scheduling has disastrous consequences for $\overline{WR}$ and σ_{WR}. In appendix A, it is proven that all such disciplines result in infinite $\overline{WR}$ and σ_{WR} in G/H/m or G/H/m-like systems. These performance indices become finite only when the processes having the highest wait ratios, those having the shortest service demands, are ignored.

1. This low error-level was achieved by repeating simulations, using independent streams of random numbers. Error bounds were calculated under an assumption of normally distributed errors. This assumption was validated at several data points and found to be a good approximation.

2. When $C_x > 1$, we assume a 2-phase hyperexponential distribution, with 70% of service demands drawn from the phase having the smaller mean.

PS is the single-queue analogue of LB. Similar to FCFS, PS is a non-discriminatory scheduling discipline, but with respect to wait *ratio*, rather than wait time. This property allows $\overline{WR}$ for PS to be easily derived from eq. 2.1:[1]

$$\overline{WT} = E[(wait\ time\ /\ x)(x)] = E[(wait\ ratio)(x)] = \overline{WR}\ \bar{X}$$

$$\overline{WR} = \overline{WT}\ /\ \bar{X} = (\bar{N}\ /\ m\rho) - 1 \qquad \text{for } \rho < 1 \tag{3.1}$$

An important property of PS is that $\overline{WR}$ and σ_{WR} are finite for stable ($\rho < 1$) M/H/m and G/G/1 queues (see appendix A). Another advantage of PS over non-preemptive disciplines is that $\overline{WT}$ is independent of C_X [Laven83a]. Regardless of C_X, $\overline{WT}$ for PS is given by eq. 2.1. Thus, as shown in figure 3.1, PS results in lower $\overline{WT}$ than FCFS when $C_X > 1$, as well as lower σ_{WT} for high C_X. Since $\overline{WT}$ is independent of C_X, eq. 3.1 shows that $\overline{WR}$ is also independent of C_X. Simulation results displayed in figure 3.2 validate this claim, as well as showing that σ_{WR} is also independent of C_X.

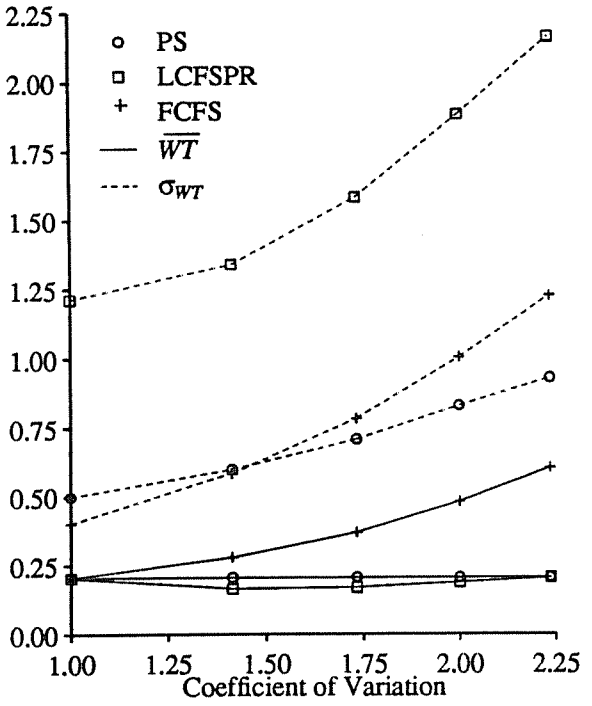

Figure 3.1 $\overline{WT}/\bar{X}$ and $\sigma_{WT}/\bar{X}$ vs. C_X (m= 10, ρ = 0.8)

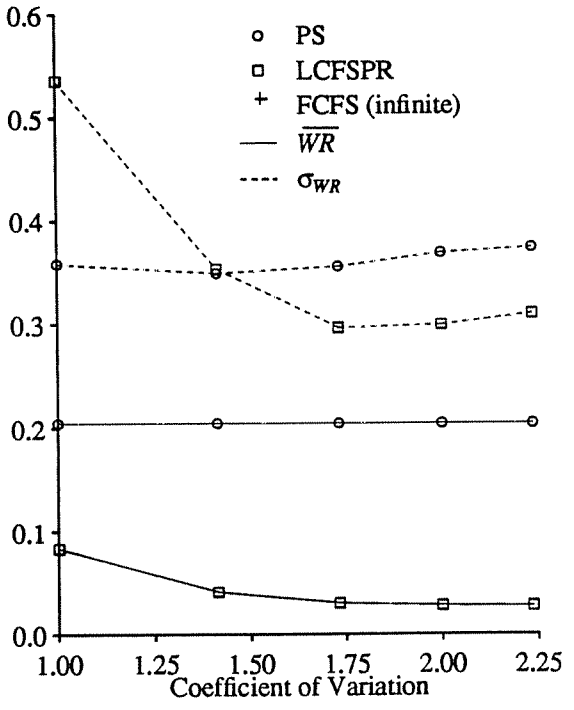

Figure 3.2 $\overline{WR}$ and σ_{WR} vs. C_X (m=10, ρ = 0.8)

1. When random variables Y and Z are independent, E[g(Y)h(Z)] = E[g(Y)] E[h(Z)].

Unlike FCFS and PS, simulations show Last-Come-First-Served-Preemptive-Resume (LCFSPR) to be a discriminatory scheduling policy, giving better quality of service, both in terms of wait time and wait ratio, to processes having short service demands. Figure 3.2 shows that $\overline{WR}$ and σ_{WR} *decrease* with increasing C_X, though σ_{WT} increases (figure 3.1). Based on these results, we can predict that load distributing algorithms in M/H/m-like systems that discriminate by providing better service to processes having short service demands will reduce $\overline{WR}$ with respect to those that are non-discriminatory. However, this reduction may come at the cost of increased σ_{WT}.

To summarize, conservation of work is not sufficient to provide finite $\overline{WR}$ or σ_{WR}. Even when the 1% of processes having the highest wait ratios are trimmed from the sample, simulations show that FCFS results in considerably higher $\overline{WR}$ and σ_{WR} than PS or LCFSPR. For example, when $C_X = 1$, the trimmed $\overline{WR}$ for FCFS is over 400% that for PS, while σ_{WR} is over 1300%. These values increase rapidly with C_X, to over 12000% for $\overline{WR}$ and 40000% for σ_{WR} at $C_X = 2.24$. Additionally, conservation of work is not sufficient to minimize $\overline{WT}$ when $C_X > 1$. Figure 3.1 shows that both PS and LCFSPR result in lower $\overline{WT}$ than FCFS under such a workload. In addition, PS results in lower σ_{WT} than FCFS for high C_X. Applying these results to distributed scheduling policies, the use of PS as a local scheduling discipline will result in finite $\overline{WR}$ and σ_{WR}, while the use of FCFS or any other non-preemptive discipline will result in infinite $\overline{WR}$ and σ_{WR}. We can predict that the use of PS will result in lower $\overline{WT}$ than any non-preemptive discipline when $C_X > 1$ and lower σ_{WT} for high C_X. Using the perspective of LB as the distributed analogue of PS and LS as the analogue of FCFS, we can predict that LB will result in lower $\overline{WT}$ than LS when $C_X > 1$, lower σ_{WT} for high C_X and, when PS is used as a local scheduling discipline, lower $\overline{WR}$ and σ_{WR} than LS under all conditions.

3.2. Performance of Distributed-Queue Scheduling Policies

3.2.1. Load Sharing (LS)

Under LS, simulations show that $\overline{WR}$, σ_{WR}, $r(\textit{wait ratio}, X)$ and $r(\textit{wait time}, X)$ are dependent on the level of inhomogeneity in the rates at which processes initiate at individual nodes. Performance in terms of these indices is generally best when initiation rates are homogeneous, and worst when all processes initiate at a single node. An additional complication of LS is that all the performance indices studied are dependent on the

criteria used to select a process to migrate. We will consider the two bounding criteria: First-Come-First-Migrate (FCFM) selects the process that has least recently arrived at the node, either through initiation or migration, while Last-Come-First-Migrate (LCFM) selects the most recent arrival. By giving an advantage to processes having short service demands, we will see that LCFM captures some of the properties of LCFSPR, reducing $\overline{WR}$ with respect to that of FCFM but increasing σ_{WT}, while FCFM retains the properties of FCFS. These differences are accentuated when process initiation rates are inhomogeneous.

Since LS is concerned solely with conserving work, it is naturally allied with a local scheduling discipline that shares this narrow perspective, such as FCFS. This partnership, which we refer to as LS_FCFS, results in a simple distributed scheduling policy that, in an M/H/m-like system, minimizes $\overline{WT}$ for workloads having $C_X = 1$. In addition, when $C_X = 1$, LS_FCFS generally results in slightly lower σ_{WT} than policies using PS for small numbers of nodes ($n \leq 6$) or high system loads ($\rho > 0.8$). However, as predicted in the previous section, $\overline{WT}$ increases with increasing C_X (figure 3.3) as does σ_{WT}. Also as predicted, the single-minded concern of

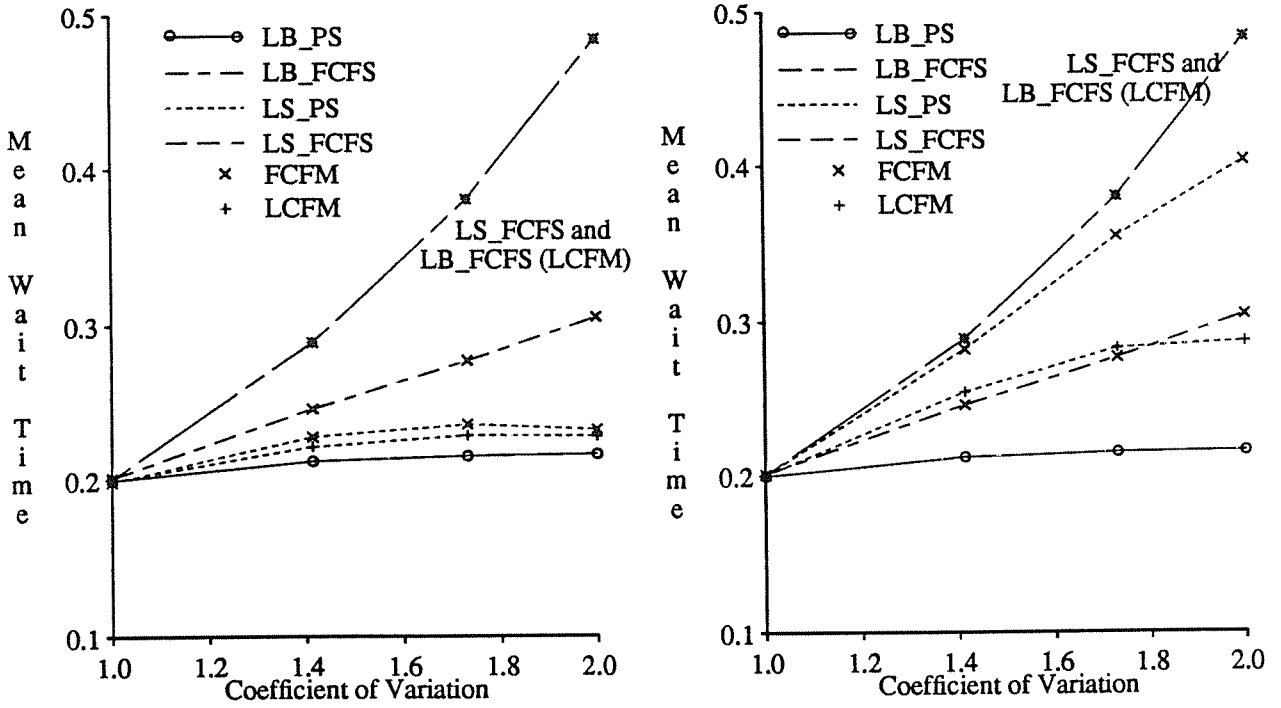

Figure 3.3 $\overline{WT} / \bar{X}$ vs. C_X assuming homogeneous initiation rates (left) or all process initiations at a single node (right) ($m = 10, \rho = 0.8$)

LS_FCFS with conserving work is an insufficient perspective from which to improve $\overline{WR}$ and σ_{WR}. As shown in appendix A, these statistics are infinite, becoming finite only if the processes having the highest wait ratios, which are those having the shortest service demands, are ignored. Simulations show that, even when the shortest processes are removed from the sample, $\overline{WR}$ and σ_{WR} are considerably higher for LS_FCFS than for distributed scheduling policies using PS as a local scheduling discipline.

Pairing LS with PS appears inconsistent, since LS is solely concerned with conservation of work, while PS has broader goals. However, studying this hybrid distributed scheduling policy provides insight into the relative effects on performance of the local scheduling discipline and the load distributing strategy. LS_PS corrects many of the weaknesses of LS_FCFS. One of the most important consequences of merging LS with PS is that $\overline{WR}$ and σ_{WR} are finite (see appendix A). In addition, LS_PS reduces the degrading effect of increasing C_X on $\overline{WT}$ (figure 3.3) and on σ_{WT}. However, inhomogeneity in initiation rates continues to have a degrading effect on performance, as can be seen in figures 3.3 through 3.9. Figure 3.10 shows that this degradation in performance is continuous with increasing inhomogeneity, rather than occurring suddenly at high levels of inhomogeneity. Like LS_FCFS, performance under LS_PS is also dependent on the migration selection criterion, with LCFM providing lower $\overline{WR}$ and σ_{WR}, while FCFM results in lower σ_{WT}.

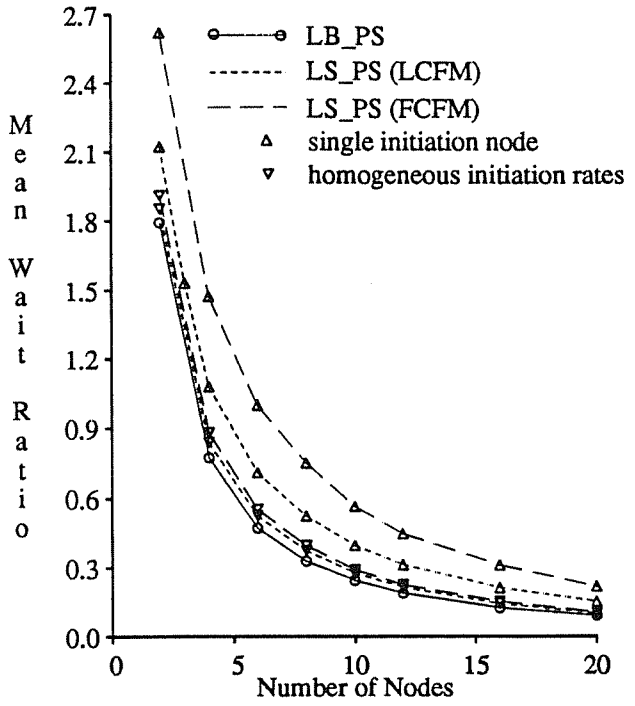

Figure 3.4 $\overline{WR}$ vs. m ($\rho = 0.8$, $C_x = 1$)

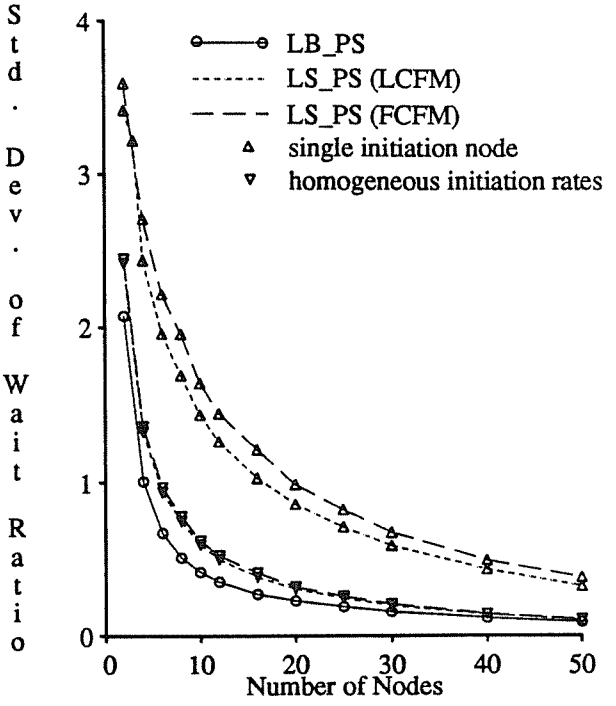

Figure 3.5 σ_{WR} vs. m ($\rho = 0.8$, $C_x = 1$)

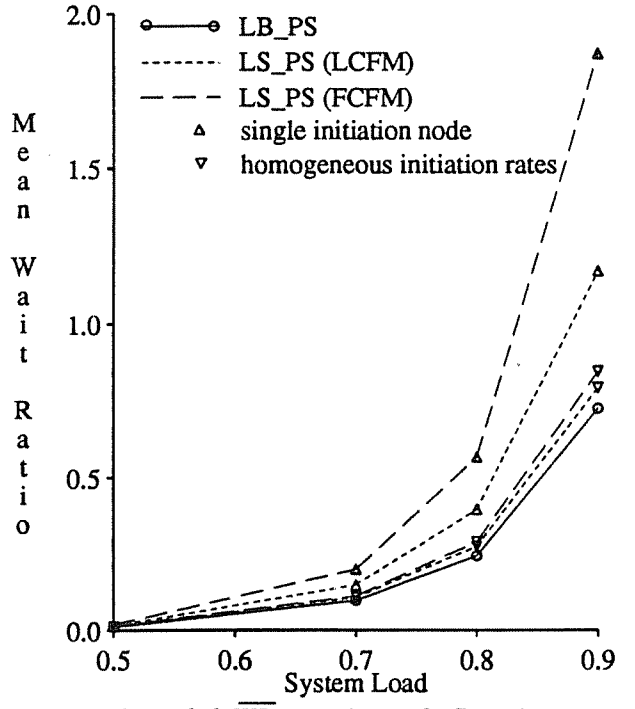

Figure 3.6 $\overline{WR}$ vs. ρ ($m = 10$, $C_x = 1$)

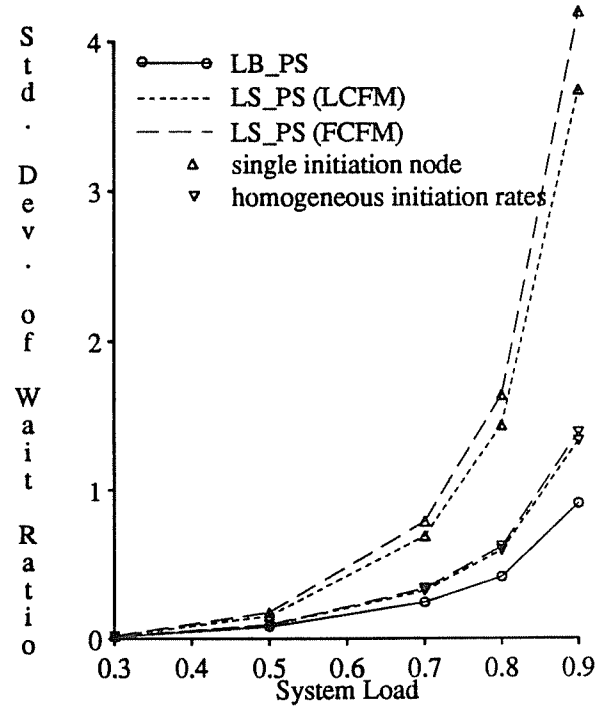

Figure 3.7 σ_{WR} vs. ρ ($m = 10$, $C_x = 1$)

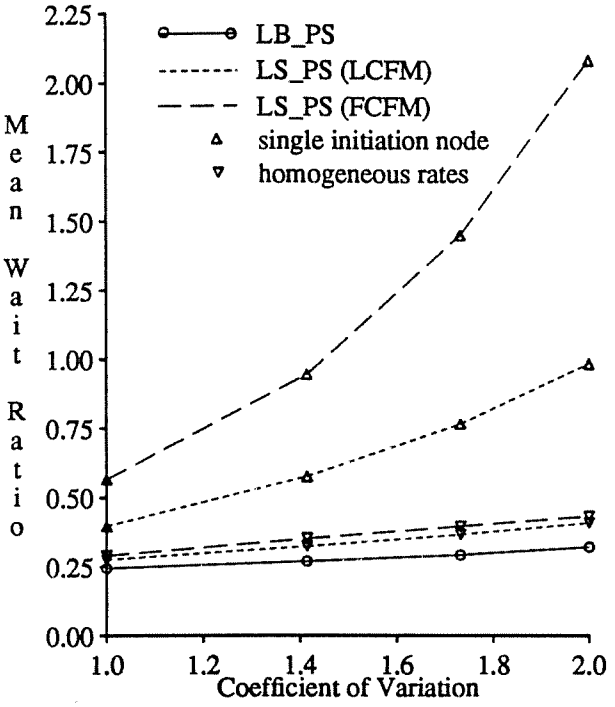

Figure 3.8 $\overline{WR}$ vs. C_X ($m = 10, \rho = 0.8$)

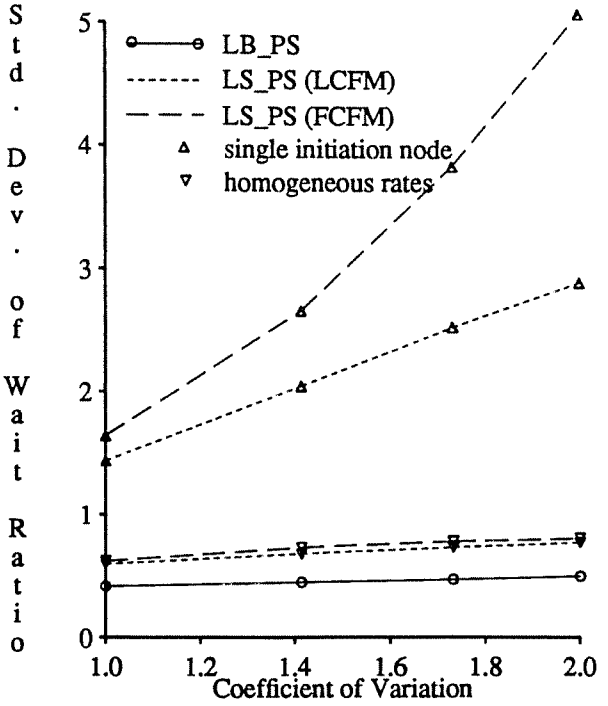

Figure 3.9 σ_{WR} vs. C_X ($m = 10, \rho = 0.8$)

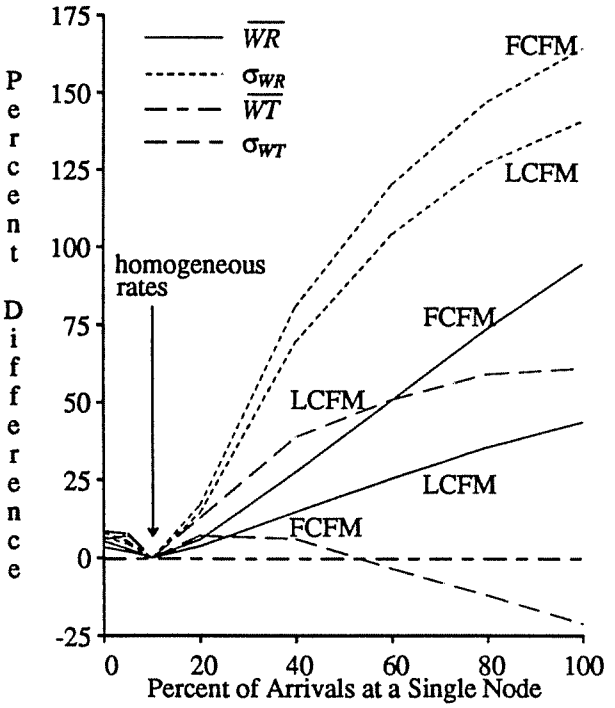

Figure 3.10 Degradation in performance of LS_PS (LCFM) caused by inhomogeneous initiation rates. Percent difference in performance index (100 (inhomogeneous - homogeneous) / homogeneous) vs. percent of processes that initiate at a single node, with remaining process initiations evenly divided among other nodes. ($m = 10, \rho = 0.8, C_X = 1$)

3.2.2. Load Balancing (LB)

True PS scheduling in a multiple-queue system, such as an M/H/m-like system, is not feasible, since it requires an infinite migration rate.[1] The goal of the LB strategy is to approximate PS scheduling, achieving similar performance, but with a finite migration rate.

LB is naturally allied with PS as a local scheduling discipline. In appendix B, $\overline{WR}$ and a close approximation of σ_{WR} are derived for LB_PS. In contrast to LS, neither of these performance indices is dependent on the homogeneity of process initiation rates or the criteria used to select a process to migrate. Through simulation, we have validated these results and have shown that the remaining performance indices are independent of these factors, as well. How closely LB_PS achieves its goal of emulating PS scheduling can be seen in figure 3.11. In terms of $\overline{WT}$ and σ_{WT}, LB_PS is identical to PS scheduling under all conditions. For other performance indices, LB_PS best approximates PS at high system loads, for small numbers of nodes or for low C_X.

As shown in figures 3.4 through 3.9, LB_PS achieves lower $\overline{WR}$ and σ_{WR}, under all conditions, than any other distributed scheduling policy examined. When all processes initiate at a single node, figure 3.12 shows that LB_PS results in considerable improvement in these indices over its nearest competitor, LS_PS(*LCFM*). These improvements become increasingly pronounced with increasing numbers of nodes, system load and coefficient of variation in service demands. However, when process initiation rates are homogeneous, improvement is large only for σ_{WR}. In addition, as predicted, LB_PS results in lower $\overline{WT}$ than any other policy when $C_X > 1$ (figure 3.3) and lower σ_{WT} when C_X is high. These advantages of LB_PS over LS_PS(*LCFM*) are become very large in environments having both inhomogeneous initiation rates and $C_X > 1$. For a system having 16 nodes, a system load of 0.8, C_X=2.24 and all processes initiating at a single node, the difference in performance between LB_PS and LS_PS(*LCFM*) is 280% for $\overline{WR}$, 575% for σ_{WR}, 30% for $\overline{WT}$ and 90% for σ_{WT}. Improvement under such conditions is important, since, as noted in chapter 2, inhomogeneous initiation rates and hyperexponentially distributed service demands may be common in many general-purpose distributed systems.

---

1. When the number of processes in service is greater than the number of nodes, but not a multiple of the number of nodes, some nodes serve one more process than other nodes. Since, under PS, the number of units of service each process must receive during *any* time period is not an integer in these cases, processes must migrate from high nodes to low nodes after a time period of infinitesimal duration.

Like LS_PS, pairing LB with FCFS as a local scheduling discipline results in a distributed scheduling policy with inconsistent goals. However, since FCFS is simple to implement and can be expected to result in less overhead than most local scheduling disciplines, it is interesting to examine how nearly the performance of LB_FCFS approaches that of LB_PS. In addition, studying such a hybrid policy allows us to gauge the relative effects on performance of its two components. Unfortunately, the influence of FCFS on $\overline{WR}$ and σ_{WR} is greater than that of LB. As shown in appendix A, both $\overline{WR}$ and σ_{WR} are infinite under LB_FCFS. Even when the shortest processes are ignored, simulations show that, while not as high as for LS_FCFS, $\overline{WR}$ and σ_{WR} are much higher than for LB_PS. Also showing similarity to LS_FCFS rather than LB_PS, performance is dependent on C_X, the level of inhomogeneity in initiation rates and the process selection criterion. Using FCFM as the selection criterion generally results in better performance than LCFM, since LCFM causes the same processes to be repeatedly migrated. Showing the influence of the LB strategy LB_FCFS(*FCFM*) results in lower $\overline{WT}$ than LS_FCFS for $C_X > 1$ (figure 3.3). When all processes initiate at a single node, $\overline{WT}$ for LB_FCFS(*FCFM*) is also generally lower than LS_PS for $C_X > 1$. LB_FCFS(*FCFM*) exhibits another characteristic trait of LB: it is discriminatory with respect to wait time. This trait arises because LB_FCFS(*FCFM*) is a preemptive discipline; processes that have begun service may not continue to receive service immediately after migrating.

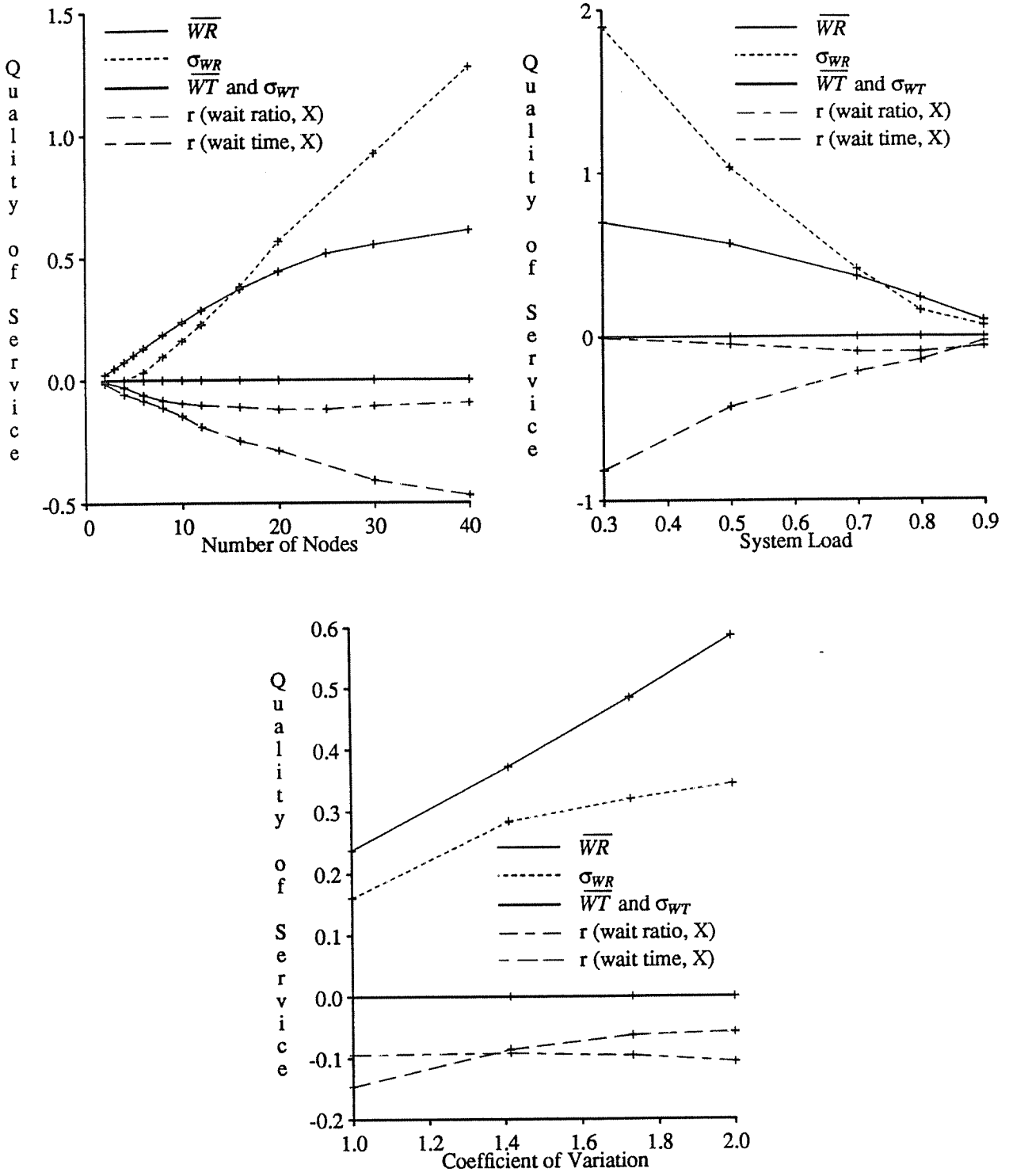

Figure 3.11 Comparison of LB_PS with PS. All indices except $r(wait\ ratio, X)$ are plotted as (LB_PS - PS) / PS. (defaults: $m = 10, \rho = 0.8, C_x = 1$)

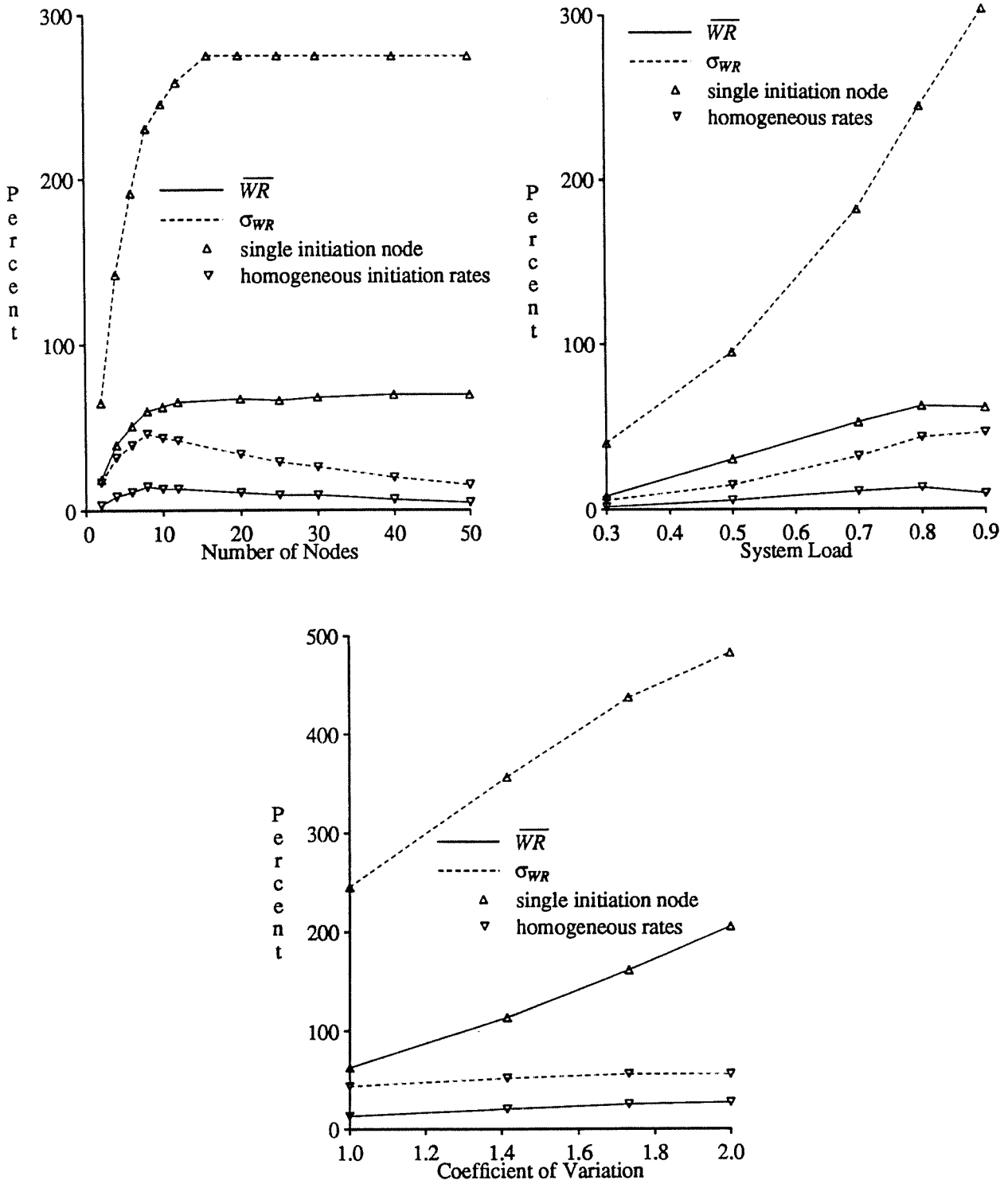

Figure 3.12 Comparison of LB_PS with LS_PS(*LCFM*): Percent difference in $\overline{WR}$ and σ_{WR} (*100(LS - LB) / LB*) (defaults: $m = 10, \rho = 0.8, C_x = 1$)

3.2.3. Load Shuffling (LSh)

LB_PS differs from PS scheduling in that not all processes receive service at the same rate when there are more processes than nodes and $n \ mod \ m \neq 0$. However, LB_PS can be extended to approximate PS scheduling arbitrarily closely by periodically varying the set of processes that receive service at a faster rate. We refer to these migrations between nodes differing by one in queue length as *shuffling* and to the resultant load distributing strategy as Load Shuffling (LSh). As the length of the *inter-shuffle time* approaches zero, the performance of LSh_PS more closely approximates that of PS. LCFM is not a suitable process selection criterion for LSh, since it results in the same process being repeatedly migrated. This property undermines the goal of LSh, which is to give equal service to all resident processes. The difference in performance between LB_PS in an M/H/m-like system and PS in in M/H/m queue is the potential improvement in performance that can result from LSh_PS. Figure 3.13 plots the percentage of this potential improvement that is achieved by LSh_PS against inter-shuffle time. Surprisingly, migrations between nodes that differ in load by one can significantly reduce $\overline{WR}$, σ_{WR} and $r(wait\ ratio, X)$. This improvement may be large under conditions in which LB_PS poorly approximates PS scheduling: low system load, high C_X or a large number of nodes.

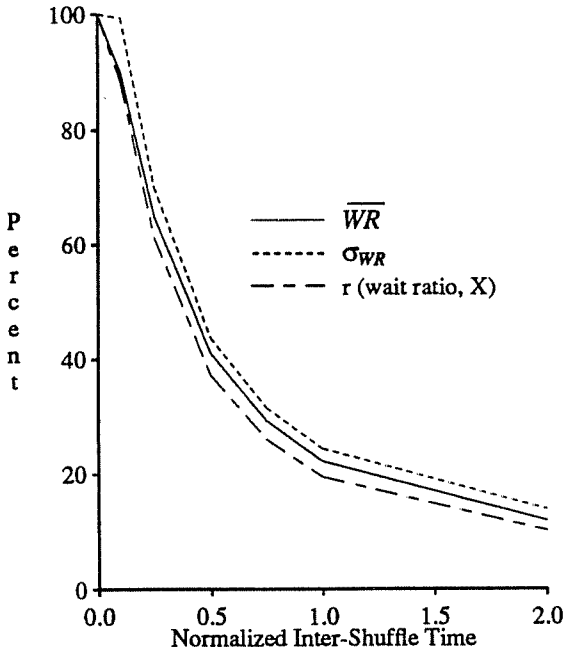

Figure 3.13 Percentage of potential improvement achieved through shuffling (100 (LB_PS - PS) / (LSh - PS)) vs. normalized intershuffle time (intershuffle time / $\bar{X}$) (m = 10, ρ = 0.8, C_x = 1)

3.3. Conclusions: Distributed Scheduling Objectives

From our study, we can identify the performance objectives of each of the components of a distributed scheduling policy in terms of a broad range of performance indices.

Among local scheduling disciplines, any work-conserving discipline, such as FCFS, minimizes $\overline{WT}$ when $C_X = 1$. However, conservation of work does not minimize $\overline{WT}$ when $C_X > 1$, nor does it address the wait ratio perspective of quality-of-service. When $C_X > 1$, PS provides lower $\overline{WT}$ than any non-preemptive discipline, and lower σ_{WT} than FCFS when C_X is high. Additionally, while many work-conserving disciplines, particularly non-preemptive disciplines, result in infinite $\overline{WR}$ and σ_{WR}, PS provides finite $\overline{WR}$ and σ_{WR}, as well as minimizing $r(\textit{wait ratio}, X)$.

Among load distributing strategies, the objective of LS is similar to that of non-preemptive local scheduling disciplines: reduce $\overline{WT}$ with respect to no load distributing, minimizing it when $C_X = 1$. However, like non-preemptive local scheduling disciplines, LS does not minimize $\overline{WT}$ when $C_X > 1$, nor does it address the wait ratio perspective of quality-of-service. The broader objective of LB is to reduce $\overline{WT}$ relative to LS when $C_X > 1$, and to reduce $\overline{WR}$ and σ_{WR} relative to LS under all conditions. Finally, the performance objective of LSh is to further reduce $\overline{WR}$ and σ_{WR} with respect to LB, and to minimize $r(\textit{wait ratio}, X)$.

Local Scheduling Disciplines	
FCFS[1]	minimize $\overline{WT}$ when $C_X = 1$.
PS	reduce $\overline{WT}$ relative to FCFS when $C_X > 1$ finite $\overline{WR}$ and σ_{WR} minimize $r(\textit{wait ratio}, X)$
Load Distributing Strategies	
LS	reduce $\overline{WT}$
LB	reduce $\overline{WT}$ relative to LS when $C_X > 1$ reduce $\overline{WR}$ and σ_{WR} relative to LS
LSh	reduce $\overline{WR}$ and σ_{WR} relative to LB minimize $r(\textit{wait ratio}, X)$

Table 3.1 Performance Objectives

1. This performance objective is shared by all work-conserving non-preemptive local scheduling disciplines.

When a load distributing strategy and local scheduling discipline are chosen to have matching performance characteristics, the resultant distributed scheduling policy mirrors those characteristics. However, when a distributed scheduling policy has components with inconsistent objectives, its performance is a hybrid of the objective of each component. For such distributed scheduling policies, while the load distributing strategy has a significant effect on $\overline{WR}$ and σ_{WR}, the effect of the local scheduling discipline on these indices is more fundamental, since it determines whether they are finite. In contrast, figure 3.3 shows that the relative effect of the load distributing strategy and local scheduling discipline on $\overline{WT}$ is dependent on the workload. The local scheduling discipline has greater effect when process initiation rates are homogeneous, while the load distributing strategy is more influential when all processes initiate at a single node.

In certain situations, these hybrid policies may be useful. If $\overline{WR}$ is considered important, but σ_{WR} is not, and if $C_X \approx 1$ and process initiation rates are generally homogeneous, LS_PS may be a suitable replacement for LB_PS. Alternatively, if only $\overline{WT}$ is considered important, but $C_X > 1$ or process initiation rates are often inhomogeneous, LB_FCFS may provide significantly lower $\overline{WT}$ than LS_FCFS, though not as low as results from LB_PS.

4. Summary

Much insight can be gained into the performance objectives of distributed scheduling policies by examining performance from a broader perspective than mean wait time. By considering a relatively large set of performance indices, we have shown that different scheduling policies have considerably different objectives and result in significantly different performance. However, since each index has significance only in terms of the performance expectations of the users, no single policy can be identified as being best for all systems. To choose the components of a policy suitable to a particular system, the performance expectations of the users together with the system workload characteristics must be considered.

We have shown that migrations beyond those necessary to conserve work (LS) can have a significant effect on performance. The additional migrations necessary for LB reduce $\overline{WR}$ and σ_{WR}, as well as reducing $\overline{WT}$ when $C_X > 1$. The still more extensive set of LSh migrations further reduces $\overline{WR}$ and σ_{WR} and minimizes $r(\textit{wait ratio}, X)$.

As a load distributing strategy, LS is suitable only for those systems having narrow performance objectives and tightly constrained workloads: $C_X \approx 1$ and generally homogeneous process initiation rates. More typical workloads or broader performance objectives necessitate the use of LB. Among local scheduling disciplines, a non-preemptive discipline, such as FCFS, may be used when $\overline{WR}$ and σ_{WR} can be ignored. When the performance objective of the system includes $\overline{WR}$ and σ_{WR}, a preemptive discipline that gives immediate service to newly arriving processes, such as PS, is necessary.

With consideration for the humanistic appeal of a performance metric that includes $\overline{WR}$ and σ_{WR}, together with the likelihood of workloads having inhomogeneous initiation rates and $C_X > 1$, we believe that LB_PS has broad applicability to general-purpose distributed systems. However, this observation is incomplete without considering the relative resource overheads of LB and LS. In [Krueg87a], we compare these overheads and their effects on the performance resulting from each of these strategies. We find that the distinction between the LB and LS strategies is obscured. The best strategy to meet a given objective is dependent on the overheads of individual load distributing operations, as well as on dynamic characteristics of the workload and resource availability. To be effective at meeting either the LB or the LS objective over the wide range of conditions occurring within a distributed system, a load distributing algorithm must be adaptive.

5. Appendix A: Measuring Wait Ratio Statistics

While the service demand of a process can not be less than the time required to execute an instruction, analysis and simulation of computer systems are simplified if service demands are modeled to follow a continuous distribution, such as exponential or hyperexponential, which allows infinitesimal values. Measurement of wait ratio statistics under such *synthetic* workloads is somewhat complicated if processes are scheduled in such a way that they may not receive service immediately upon initiating. Examples of such scheduling policies include all non-preemptive scheduling disciplines, such as FCFS, LB_FCFS and LS_FCFS, as well as preemptive priority scheduling disciplines, such as preemptive HOL.[1] For preemptive scheduling disciplines, it is useful to distinguish the wait time experienced by a process before it has received initial service (wt_b) from that

1. See [Klein76a] and chapter 3 for descriptions of these scheduling disciplines.

experienced after it has begun execution (wt_a). The expected wait ratio is then:

$$\overline{WR} = E(wait\ time\ /\ x) = E((wt_b+wt_a)\ /\ x) = E(wt_b\ /\ x) + E(wt_a\ /\ x)$$

Since we assume, as in the remainder of this paper, that scheduling makes no use of deterministic a priori information about process service demands, wt_b for a process is independent of its service demand. So:[1]

$$= E(wt_b)\,E(1/x) + E(wt_a\ /\ x) \tag{A.1}$$

For non-preemptive scheduling disciplines, $wt_a = 0$, so this simplifies to:

$$= E(wait\ time)\,E(1/x)$$

Similarly, for preemptive disciplines:

$$\overline{WR^2} = E(wait\ time^2/x^2) = E((wt_b+wt_a)^2/x^2) = E(wt_b{}^2/x^2) + 2E(wt_b\,wt_a/x^2) + E(wt_a{}^2/x^2)$$

$$= E(wt_b{}^2)\,E(1/x^2) + 2E(wt_b\,wt_a\ /\ x^2) + E(wt_a{}^2\ /\ x^2)$$

which, for non-preemptive disciplines simplifies to:

$$= E(wait\ time^2)\,E(1/x^2)$$

When service demands are exponentially distributed, with probability density function $f_x(x) = (1/\bar{X})e^{-x/\bar{X}}$, the distribution[2] of $1/x$ is $x^2 f_x(x) = (1/\bar{X})x^2 e^{-x/\bar{X}}$. The expected value of this random variable is:

$$E(1/x) = (1/\bar{X})\int_0^\infty xe^{-x/\bar{X}}\,d(1/x) = (1/\bar{X})\int_0^\infty (e^{-x/\bar{X}}\ /\ x)\,dx = (1/\bar{X})\int_0^\infty (e^{-u}\ /\ u)\,du = \infty$$

where $u = x\ /\ \bar{X}$. Substituting the above into eq. A.1, we see that whenever $E(wt_b) > 0$, $\overline{WR}$ is infinite. Similarly:

$$E(1/x^2) = (1/\bar{X})\int_0^\infty e^{-x/\bar{X}}\,d(1/x^2) = (1/\bar{X})\int_0^\infty (e^{-x/\bar{X}}\ /\ x^2)\,dx = (1/\bar{X})\int_0^\infty (e^{-u}\ /\ u^2)\,du = \infty$$

When $E(wt_b) > 0$, $\overline{WR^2}$ is also infinite. From this result, it can be shown that:

$$\sigma_{WR} = \left[\overline{WR^2} - \overline{WR}^2\right]^{1/2} = \infty$$

Similar arguments show that the service demand distribution can be generalized to a hyperexponential or a 2 or 3-phase Erlang distribution with the same results. In addition, these results are independent of the arrival

1. When random variables Y and Z are independent, E[g(Y)h(Z)] = E[g(Y)] E[h(Z)].

2. Setting y = 1/x: $f_y(y) = f_x(x)\,|dx\ /\ dy\,| = f_x(x)\ /\ y^2 = x^2 f_x(x)$.

process, and are therefore valid for G/H/m and G/H/m-like systems.

It is only due to the processes having infinitesimal service demands that $E(1/x)$ and $E(1/x^2)$ are infinite, resulting in infinite $\overline{WR}$ and σ_{WR}. For $C > 0$, the conditional expectations $E(1/x \mid x > C)$ and $E(1/x^2 \mid x > C)$ are finite, resulting in finite *conditional* $\overline{WR}$ and σ_{WR}. Thus, for synthetic workloads in which service demands strictly follow an exponential, hyperexponential or 2 or 3-phase Erlang distribution and processes are scheduled such that they do not necessarily receive service immediately upon initiating, $\overline{WR}$ and σ_{WR} are stable and can be meaningfully measured only over the portion of the population remaining after those processes having service demands less than $C>0$ are removed. The value chosen for C has an effect on the quantity of data that must be collected to achieve a given level of accuracy in measurements of conditional $\overline{WR}$ and σ_{WR}. Because the conditional moments of wait ratio increase with decreasing C, achieving the same level of accuracy requires more wait ratio samples (i.e. longer runs) for smaller values of C.

In contrast, PS scheduling results in finite $\overline{WR}$ and σ_{WR} for G/G/m queues. This result follows from the definition of PS scheduling. When n processes reside in the system, each process receives service at the rate $r = 1/n$, resulting in a *partial* wait ratio, the wait ratio over that period, for each process of $r - 1$. Since n remains finite under our assumptions [Laven83a], processes receive service at a finite rate, and each partial wait ratio is finite. Since the overall wait ratio of a process is the weighted sum of its partial wait ratios, the wait ratio of each process is finite, as are $\overline{WR}$ and σ_{WR}. This result generalizes to G/G/m-like systems using PS as a local scheduling discipline and having a work-conserving load distributing strategy. Since every process continuously receives service at a finite rate, all wait ratios are finite, as are $\overline{WR}$ and σ_{WR}.

6. Appendix B: The Mean and Standard Deviation of Wait Ratio for LB_PS

For an M/G/m-like system using the LB_PS distributed scheduling policy (without shuffling):

$$\overline{WR} = (1/\rho)\sum_{n=0}^{\infty} E(\textit{wait ratio} \mid n)\, p_n \qquad \text{(B.1)}$$

where p_n is the probability that the system contains n processes. This probability has been derived for work-conserving systems having workloads with exponentially distributed service demands [Klein76a].

For load balancing, when the number of processes in the system is n, the number of nodes having greater than the mean number of processes is $k = n \bmod m$, and the number of processes residing on each of these

nodes is $n_h = (n \; div \; m)+1$. The wait ratio of each of these processes during the period that there are n processes in the system is n_h-1. Similarly, $m-k$ nodes each have n_h-1 processes, each of which has a wait ratio of n_h-2 during this period. Thus the expected wait ratio when there are n processes in the system is:

$$E(wait\ ratio \mid n) = \frac{1}{n}\left[kn_h(n_h-1) + (m-k)(n_h-1)(n_h-2)\right]$$
$$= n_h\left[(k/n)+1\right] - 2 \qquad \text{(B.2)}$$

Substituting the results of eq. B.2 into eq. B.1 we see that $\overline{WR}$ for LB_PS is solely dependent on the system load and number of nodes. Similarly, an upper bound for σ_{WR} can be found:

$$Upper\ bound\ \text{for}\ \sigma_{WR} = \left[(1/\rho)\sum_{n=0}^{\infty} E(wait\ ratio^2 \mid n)\, p_n - \overline{WR}^2\right]^{1/2}$$

where:

$$E(wait\ ratio^2 \mid n) = \frac{1}{n}\left[kn_h(n_h-1)^2 + (m-k)(n_h-1)(n_h-2)^2\right]$$

This equation represents an upper bound because it depends on the assumption that each job receives the same wait ratio throughout its execution (this assumption does not affect the result of eq. B.1). If some process experiences more than one wait ratio over the course of its execution, σ_{WR} is reduced. However, simulation results show that, for exponentially distributed service demands, this upper bound closely approximates σ_{WR}.

References

[Eager86a] D. L. Eager, E. D. Lazowska, and J. Zahorjan, "Adaptive Load Sharing in Homogeneous Distributed Systems," *IEEE Transactions on Software Engineering SE-12*, 5, pp. 662-675 (May 1986).

[Klein67a] L. Kleinrock, "Time-Shared Systems: A Theoretical Treatment," *Journal of the ACM 14*, pp. 242-261 (1967).

[Klein76a] L. Kleinrock, *Queuing Systems: Volume 2, Computer Applications*, John Wiley & Sons (1976).

[Krueg87a] P. Krueger and M. Livny, "When is the Best Load Sharing Algorithm a Load Balancing Algorithm?," Technical Report 694, University of Wisconsin—Madison, Dept. of Computer Sciences (April 1987).

[Laven83a] S. S. Lavenberg, *Computer Performance Modeling Handbook*, Academic Press (1983).

[Livny82a] M. Livny and M. Melman, "Load balancing in homogeneous broadcast distributed systems," *Computer Network Performance Symposium*, pp. 47-55 (April 1982).

[Livny83a] M. Livny, The Study of Load Balancing Algorithms for Decentralized Distributed Processing Systems, PhD Thesis, Weizmann Institute of Science, Rehovot, Israel (available as Technical Report 570, University of Wisconsin—Madison Computer Sciences) (August 1983).

[Rosin65a] R. F. Rosin, "Determining a Computing Center Environment," *CACM 8*, 7, (July 1965).

[Trive82a] K. S. Trivedi, *Probability & Statistics with Reliability, Queuing and Computer Science Applications*, Prentice-Hall (1982).

[Zhou86a] S. Zhou, "A Trace-Driven Simulation Study of Dynamic Load Balancing," Technical Report, UCB/CSD 87/305, Computer Science Division, University of California, Berkeley (September 1986).