Proceedings of the 1990 Winter Simulation Conference
Osman Balci, Randall P. Sadowski, Richard E. Nance (eds.)

ON COMPARING LOAD INDICES USING ORACLE SIMULATION

Sayed A. Banawan

Department of Computer Science
University of Houston
Houston, Texas 77204

John Zahorjan

Department of Computer Science
and Engineering
University of Washington
Seattle, Washington 98105

ABSTRACT

Load sharing has been the focus of a great deal of research as a means of enhancing the performance of distributed systems. To insure its success, a load sharing policy must use some metric to compare the load at different nodes in the system. The result of the comparison is used to make decisions regarding the assignment of jobs to nodes. The goal is to optimize some performance measure such as the time to completion for individual jobs or the mean response time of all jobs.

In this paper, we use oracle simulation to compare different load measures in their impact on the performance of load balancing policies. During oracle simulation, the system is simulated and the performance measure of interest is evaluated in the future for *each* possible decision in the current state. By comparing the value of the performance measure in the future, the optimal decision for the current state can be reached. our results show that the obvious load measures can be classified into two sets. The first set includes the instantaneous queue length, average queue length, utilization, response ratio and total unfinished work. Any of these measures can greatly improve system performance. The other set includes arrival rate and throughput. These measures have only a slight effect on performance. Finally, the peformance of practical load balancing policies, that do not use future information, support the same conclusion.

1. INTRODUCTION

Load sharing has been the focus of a great deal of research as a means of enhancing the performance of distributed systems. It can be seen as an instance of the problem of resource allocation and scheduling. The goal is to make efficient and judicious use of the replicated resources available to improve system performance.

Traditionally, load sharing has been approached from two different views: *task allocation* and *job scheduling*. They differ in the nature of workload to be shared. Task allocation is concerned with the assignment of modules that belong to a single computational task to available nodes so that some performance measure, typically the task completion time, is optimized [Efe 1982; Stankovic and Sidhu 1984; Lo 1988].

Job scheduling, on the other hand, attempts to improve the overall performance by transferring jobs between nodes to make use of concurrently idle or lightly loaded resources in the system. In this case, the overall mean response time is the performance measure of interest. It differs from task allocation in that it deals with streams of jobs, each of which usually consists of a single module. Jobs arrive at arbitrary times, perhaps according to some stochastic process. Furthermore, their computation demands may not be known *a priori*. Livny and Melman [Livny and Melman 1982] demonstrated the potential benefits of transferring jobs for the purpose of load sharing. They analytically derived an expression for the probability that a distributed system reaches a state in which a job is waiting for service while at the same time one or more idle nodes are available. Under fairly reasonable assumptions, this probability approaches 1 over the entire range of system utilization. In other words, almost all the time one node is idle while another experiences congestion.

Job scheduling can further be classified as static or dynamic. A scheduling policy whose decisions are independent of system state is considered static [Tantawi and Towsley 1985]. A static policy determines the branching probabilities according to which jobs are assigned to nodes. Although static policies are simple, easy to implement, and have a minimal runtime overhead, they lack the flexibility of making scheduling decisions based on the current state of the system. A dynamic policy, on the other hand, can use such information to resolve any *transient* congestion in the system [Wang and Morris 1985; Zhou 1987; Eager et al. 1986, 1988; Pulidas et al. 1988; Banawan 1989].

A key issue in the design of a dynamic load sharing policy is to select a metric to compare the load at different nodes. Although a vast number of load sharing policies have been proposed, only a few results are available regarding the most appropriate load measure for load sharing purposes [Ferrari and Zhou 1986]. In this paper, we compare several intuitive load measures in their impact on the performance of load sharing.

The paper has the following organization. In §2 we present our model of a distributed system. §3 defines different load measures that can potentially benefit dynamic load sharing. In §4 we describe how oracle simulation is used to compare these measures. The actual performance of (practical) load balancing is reported in §5. Finally, §6 summarizes our conclusions.

2. THE MODEL

Our model of a distributed system consists of nodes that communicate over a local area network. Furthermore, the following assumptions regarding the system are made:

- It is a medium size system. The number of nodes may range from few to a few hundreds.
- It is a homogeneous system. All nodes are functionally equivalent. Thus, a job can be executed by any node.
- It has a broadcast capability. All nodes can listen to the information transmitted over the communication network.
- Job transfer can occur only at the job's arrival time. Once a job has been assigned to a node, it continues to run there until completion, i.e., process migration is not allowed.

There are several implications of these assumptions. First, the broadcast capability makes it possible for each node to be aware of the "state" of other nodes. Second, there is a uniform distance between any pair of nodes in the system. Thus, it takes the same amount of time to send a message from any node to any other node. This time may be negligible provided that the communication network has a high bandwidth. Third, the execution time on a remote node is equal to the execution time on the local node. Finally, the overhead of collecting state information is relatively small. (The experimental results of Korry [Korry 1986] and Zhou and Ferrari [Zhou and Ferrari 1987] suggest that the overhead of load sharing in locally distributed systems is negligible.)

Although the model is simple, even for this model the optimal dynamic load sharing policy is not yet known. An intuitively appealing approach is to attempt to balance the load among all nodes as follows. Let $l_i(t)$ denote the load of node i at time t, thus $l_i(t)$ reflects the current level of congestion at that node. The system-wide load information at time t is then given by the vector $L(t) = [l_1(t), l_2(t), \ldots, l_n(t)]$ where n is the number of nodes in the system. To quantify the system balance (or more accurately the system imbalance), we will use the variance of the load vector L, defined by:

$$\sigma^2_{L(t)} = \frac{1}{n}\sum_{i=1}^{n}[l_i(t) - \bar{l}(t)]^2$$

where $\bar{l}(t) = \sum_{i=1}^{n} l_i(t)/n$, the average node load. A system is perfectly balanced if $\sigma^2_{L(t)} = 0$, which implies all nodes have exactly the same load. From a practical point of view, however, $\sigma^2_{L(t)}$ changes over time. A load balancing policy should react by assigning jobs to nodes so as to minimize it.

3. LOAD MEASURES

In the previous section, a measure of system balance was defined assuming that the load at each node can be determined. There are several apparent choices for the load measure $l_i(t)$. For example, it may reflect the instantaneous state of node i at time t. Alternatively, it can reflect the changes in the node state over a time interval T that ends at time t, i.e., the interval $(t - T, t)$. The beginning of the interval may be coincident with a change in the system state, e.g., the time of the last arrival.

The following load measures are considered:

1. *Number of arrivals*: the load at node i at time t is equal to the number of jobs that arrived at that node during the interval T. That is:
$$l_i(t) = A_i(t - T, t)$$
2. *Number of departures*: similar to 1, but the number of departures instead of arrivals is used. Thus, we have:
$$l_i(t) = D_i(t - T, t)$$
3. *Instantaneous queue length*:
$$l_i(t) = q_i(t)$$
where q_i is the length of the ready queue at node i.
4. *Average queue length*: the load at node i at time t is the average queue length during the last interval T. That is:
$$l_i(t) \equiv \bar{q}_i(t - T, t) = \frac{1}{T}\int_{t-T}^{t} q_i(u)du$$
5. *Mean response time*: the load at node i depends on the average response times of jobs that departed from i during the interval T. To account for the case of no departures we use the following expression for $l_i(t)$:
$$l_i(t) = \begin{cases} \dfrac{\bar{q}_i(t-T,t)T}{D_i(t-T,t)} & D_i(t-T,T) > 0 \\ \bar{q}_i(t-T,t)T & D_i(t-T,T) = 0 \end{cases}$$
6. *Mean stretch factor*: Let t_1 and t_2 be the endpoints of a time interval. Then, the stretch factor of j during that interval is defined as follows:
$$S_j(t_1, t_2) = \frac{t_2 - t_1}{\text{service time received by } j \text{ in } (t_1, t_2)}$$
If t_1 is the job arrival time and t_2 is the job departure time then S_j becomes identical to the process response ratio defined by Krueger and Finkel [Krueger and Finkel 1984], i.e., the ratio of the job response time to its service time. The stretch factor has a minimum value of 1.0 when the job does not experience any congestion. A load measure $l_i(t)$ may be defined by averaging the stretch factors of all jobs that were *fully or partially* executed during the time interval $(t - T, t)$. In other words,
$$l_i(t) = \frac{1}{|J|}\sum_{j \in J} S_j(t - T, t)$$
where J is the set of jobs that received service during the interval $(t - T, t)$.
7. *Utilization*:
$$l_i(t) = \frac{B(t - T, t)}{T}$$
where $B(t - T)$ is the total time the node was busy during the interval $(t - T, t)$.
8. *Unfinished work*: the load at node i at time t depends on the total amount of work yet to be done. That is:
$$l_i(t) = \sum_{1 \le j \le q_i(t)} W_j(t)$$
where $W_j(t)$ is the remaining service time of job j running on node i and $q_i(t)$ is the node's queue length.

Since the above measures are different, they may not be equally useful for load balancing. For example, the number of arrivals or departures captures less information about the node state than other measures since they do not take into account the initial state of the node or jobs' service demands. Utilization tells us how much work has been done, while the unfinished work indicates the amount of work that is yet to be done. Both the mean response time and the mean stretch factor convey some information about the congestion and the service demands of jobs at the node. On the other hand, the queue length and the average queue length contain information about the congestion experienced by the node but nothing about how much work has been or is yet to be performed.

4. IDEAL BALANCING USING ORACLE SIMULATION

To determine which load measure is most useful, we compare the performance of load balancing policies. Except for the load measure used, all policies attempt to maintain system balance as much as possible so that each new job finds the system in the *most balanced* state. The most balanced state is such that the variance of the load vector, σ^2_L, is minimum.

The comparison is based on simulation. Figure 1 shows a flowchart of our simulator. We call it *oracle* simulation because the simulator looks ahead in the *future* to determine the consequences of an action taken at the *present time*. It runs as follows. Whenever a new job arrives, say at time t, it is assigned to some node and the system is simulated until the time of the next arrival, $t+T'$. At time $t+T'$ the system balance is evaluated using σ_L^2. The simulator then backtracks and tries assigning the *same* job to *all* other nodes, one at a time. For each assignment it evaluates σ_L^2 at time $t+T'$. The job is then *permanently* assigned to the node that yields a minimal value for σ_L^2 at $t+T'$. If there are more than one such assignment, one is selected at random. The simulation continues with the next arrival at time $t+T'$. Note that the scheduling decision at time t is made to minimize the system balance measure σ_L^2 at time $t+T'$. Later, we will consider the practical case in which the goal is to minimize the current value of the system balance measure σ_L^2 instead of its future value at time $t+T'$.

Because of the time complexity of oracle simulation, $O(NJ)$ where N is the number of nodes and J is the number of jobs, our examples use a system with a moderate number of nodes. This allows us to simulate a large number of jobs J, which is necessary to obtain valid results from the simulation.

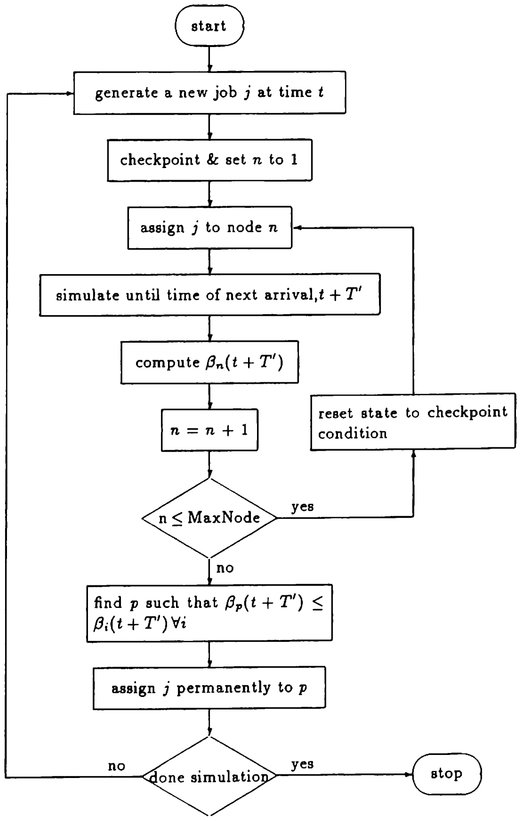
In the simulation, each node is modeled as a single service center. Each job is assumed to be monolithic, i.e., it has no subtasks. The local scheduling discipline at each node is assumed to be processor Sharing (PS), which is an approximation of Round-Robin (RR) scheduling when the quantum is small [Coffman and Denning 1973]. Both the service time and the interarrival time are exponentially distributed. The time unit is set equal to the average job service time, thus the service rate at each node, μ, is 1 job/unit time. The overall arrival rate λ can be calculated from the desired utilization ρ using the formula $\lambda = n\rho\mu$, where n is the number of nodes in the system. The cost of job transfer is assumed negligible.

Two values of system utilization are considered: 50% and 80%. Outside this range, the distinction between different load measures would be harder to draw accurately since most nodes are either lightly loaded or heavily loaded. In both cases, load balancing has only a minor effect on system performance. Table 1 shows the results of the oracle simulation for a system with 10 nodes. Each row corresponds to a different load measure that was used to evaluate the system balance. Each entry contains the overall mean response time and its 95% confidence interval. For comparing the results, the last row in the table gives the mean response time if no load balancing is performed during simulation. Some comments about the results presented in Table 1 are in order. First, the variance of the load vector was used as the measure of the system balance except when the load measure is based on node utilization. With utilization, keeping only a single node busy yields a minimum variance as the vector $[1,0,0,0,0,0,0,0,0,0]$ has a lower variance than $[1,1,0,0,0,0,0,0,0,0]$. However, this is exactly the opposite of what we would like to achieve. (In fact, it may cause the system to saturate since the queue at the busy node grows unbounded.) To overcome this problem, we chose to maximize the overall system utilization instead of balancing the utilization of individual nodes. The overall system utilization is the sum of the utilizations of all nodes in the system. Second, it was found that balancing the system using the number of departures as load measure does not exhibit an improvement on system performance. Another policy that was investigated is to maximize the total number of departures. Unfortunately, the new policy that attempts to maximize the system throughput does not improve performance either as shown in Table 1. Finally, balancing the arrival rates does not require backtracking during simulation. It suffices to assign jobs to nodes in a round-robin fashion. This scheduling policy is called *cyclic scheduling* [Wang and Morris 1985].

The results in Table 1 suggest that load measures can be grouped in two sets. The first set includes instantaneous queue length, mean queue length, mean response time, mean stretch factor, utilization and unfinished work. Any of these measures exhibits a very good performance compared to the case of no load balancing. The other set includes balancing arrival rates, balancing the number of departures from different nodes and maximizing the total number of departures from the whole system. They exhibit far inferior performance compared to the measures in the first set. The measures in the second set, as we pointed out earlier, neither take into account the initial state of a node, nor consider job service demands. We conclude that any of the load measures in the first set has a potential to yield a substan-

Table 1. Ideal Load Balancing

Policy	$\rho = 0.5$	$\rho = 0.8$
Bal. Queue Length	1.019 ± 0.018	1.273 ± 0.058
Bal. Ave. Que. Len.	1.002 ± 0.017	1.259 ± 0.057
Bal. Resp. Time	1.003 ± 0.017	1.258 ± 0.060
Bal. Str. Factor	1.007 ± 0.017	1.267 ± 0.062
Max. System Util.	1.007 ± 0.017	1.318 ± 0.080
Bal. Unfinished Work	1.074 ± 0.019	1.283 ± 0.070
Cyclic	1.296 ± 0.039	2.880 ± 0.313
Max. Departures	1.838 ± 0.075	4.888 ± 0.699
Bal. Departures	1.987 ± 0.073	3.552 ± 0.251
No Load Sharing	2.001 ± 0.101	4.796 ± 0.486



start
generate a new job j at time t
checkpoint & set n to 1
assign j to node n
simulate until time of next arrival, $t+T'$
compute $\beta_n(t+T')$
$n = n + 1$
reset state to checkpoint condition
$n \leq$ MaxNode
yes
no
find p such that $\beta_p(t+T') \leq \beta_i(t+T') \forall i$
assign j permanently to p
done simulation
no
yes
stop

Figure 1. Oracle Simulation

tial improvement in the performance of a distributed system and they are more or less equivalent in their impact on system performance. The other set should be ruled out.

Heterogeneous Jobs

The results in Table 1 assume that the job service time distribution is exponential. This distribution genereates jobs that are considered "homogeneous". Leland and Ott [Leland and Ott 1986], having examined the behavior of an extremely large number of processes, found that the actual distribution of service time may not fit the exponential distribution very well. Furthermore, they observed that most processes are short. Only a minute fraction of all processes are very long.

To check whether the results in Table 1 are sensitive to the workload characteristics, the service time distribution was changed from exponential to 2-stage hyper-exponential with the probability density function:

$$P(x) = \alpha\,\mu_1 e^{-\mu_1 x} + (1-\alpha)\,\mu_2 e^{-\mu_2 x}$$

The above distribution is the merge of two exponential distributions whose average values are $1/\mu_1$ and $1/\mu_2$. By choosing $\mu_1 = 100, \mu_2 = 1/10$ and $\alpha = 900/999$ the distribution tries to capture the effect of very long jobs since $\frac{1}{\mu_2} = 1000\frac{1}{\mu_1}$. This distribution has the same mean as that of the exponential distribution $P(x) = e^{-x}$ used to generate previous results, so we may compare the results directly. The variance of the hyper-exponential distribution, however, is much higher. In particular, the CV^2 (the variance divided by the square of the mean) is 18.82 while it is 1 for the exponential distribution. According to the hyper-exponential distribution, most jobs are short. In fact, 98% of

Table 2. Ideal Load Balancing–Heterogeneous Jobs

Policy	$\rho = 0.5$	$\rho = 0.8$
Bal. Queue Length	1.043 ± 0.072	1.390 ± 0.215
Bal. Ave. Que. Len.	1.036 ± 0.072	1.392 ± 0.203
Bal. Resp. Time	1.047 ± 0.076	1.388 ± 0.202
Bal. Str. Factor	1.039 ± 0.073	1.384 ± 0.203
Max. System Util.	1.040 ± 0.074	1.534 ± 0.259
Bal. Unfinished Work	1.050 ± 0.074	1.355 ± 0.217
Cyclic	2.016 ± 0.310	5.481 ± 1.358
Max. Departures	2.092 ± 0.268	5.881 ± 1.150
Bal. Departures	2.125 ± 0.293	6.299 ± 1.264
No Load Sharing	2.112 ± 0.274	4.559 ± 0.826

all jobs consume less than half the total processing time, which agrees with the measurements of Leland and Ott. The remaining 2% of jobs consume more than half the total processing time. The two types of jobs, which differ greatly in their demands, make the job stream "heterogeneous" as opposed to the homogeneous job stream generated from the exponential distribution. Table 2 shows the results of oracle simulation when the service time has the above hyper-exponential distribution. The table shows that, even when the service time has a hyper-exponential distribution, the classification of load measures according to their usefulness for load balancing purposes into two sets is still valid. Any of the measures in the first set yields a substantial improvement in the overall mean response time and they remain equivalent in their potential. On the other hand, the table confirms our conclusion that the load measures in the other set should be ruled out as a basis for load balancing. Even balancing arrival rates, which exhibited some improvement previously, loses its advantage when the service time distribution becomes hyper-exponential. As the arrival rates to different nodes remain the same long jobs, while there are only a few of them, would have an adverse effect on short jobs that form the majority of the workload.

5. PRACTICAL LOAD BALANCING

In the previous section oracle simulation allowed us to compare several load measures in their potential for improving system performance under load balancing. The oracle technique was used so that the decision regarding the assignment of each job guarantees that the system will reach the most balanced state at the arrival time of the following job. This time must be known by the scheduler.

In practice, however, no future information is usually available to the scheduler and the consequences of a decision taken now is not yet known. Thus, a practical load balancing policy should have a slightly different goal, namely to reduce the *present* system imbalance. This can be achieved by assigning each new job to the least loaded node.

In this section we compare the performance of practical load balancing based on those load measures that proved useful under ideal load balancing. For some of them, the formula used to evaluate the node load may have to be modified since no future information is available. In particular, the mean response time and the mean stretch factor are computed slightly differently. Let $\bar{R}_i$ be the mean response time of all jobs that departed from node i during the last interarrival time period. $\bar{R}_i$ is used to estimate the load at node i if the new job is assigned to it as follows:

$$\hat{l}_i(t) = \begin{cases} \bar{R}_i(t)\dfrac{q_i(t)+1}{q_i(t)} & q_i(t) > 0 \\ 0 & q_i(t) = 0 \end{cases}$$

where $q_i(t)$ is the instantaneous queue length at node i. The new job which arrives at time t is then assigned to node k such that:

$$\hat{l}_k(t) = \min(\hat{l}_1(t), \hat{l}_2(t), \ldots, \hat{l}_n(t))$$

Note that if $q_i(t) = 0$, node i is already idle and the new job can immediately be assigned to it. A similar formula is also used when the load measure is based on the mean stretch factor.

As for the total unfinished work as a load measure, there are two possible versions. In the first version it is assumed that the total service time of each job is known in advance. At any instant we can determine exactly the remaining service time of a job by subtracting the service time already used from the job total service time. In the second version the total service time of a job is not available. To determine the remaining service time we use an estimate Bryant and Finkel [Bryant and Finkel 1981] suggested that a good estimate of the remaining service time is the amount of time already used. In other words, each job is assumed to be half processed. The sum of the estimates of remaining service times for all jobs running on a particular node is used as a load measure. The first version is called the "Min. Unfinished Work" policy while the second version is called the "Min. Finished Work" policy.

The comparison of practical load balancing policies is also based on simulation. However, in this case, the simulation time is monotonically increasing since no backtracking takes place. In other words, a job is assinged only once. The decision cannot be revoked. Table 3 shows the performance of practical load balancing. Again, the system is assumed to have 10 nodes and the service time distribution is exponential. The table gives the mean response time and the corresponding 95% confidence interval at $\rho = 50\%$ and 80%. The results indicate that those measures that proved to be useful under *ideal* load balancing do perform well under *practical* load balancing. In fact, Table 1, which assumes ideal balancing and Table 3 are very comparable save for a very small increase in mean response times and their confidence intervals under practical load balancing. This increase is due to the fact that scheduling decisions under practical load balancing aim at "fixing" the present system imbalance, while ideal load balancing guarantees that each new job finds the system at the most balanced state. Should the workload consist of heterogeneous jobs the same conclusion holds true. The simulation results in this case are shown in Table 4. The service time has the same hyper-exponential distribution used before. Indeed, the entries in Table 4 are also very close to their counterparts in Table 2 under ideal balancing.

Notice that the simulation results of practical load balancing show that the "Min. Finished Work" policy, which uses estimates of remaining service times, does not perform as well as other measures, particularly when the workload consists of heterogeneous jobs. Recall that the estimate is based on the assumption that at any time instant each job is half processed. Clearly, this approximation is not valid at the early and last stages of job execution. Since the hyper-exponential distribution generates some jobs that have very long service times, these jobs can cause significant errors in estimating the amount of unfinished work at different nodes. The erroneous estimates are used more often to

Table 3. Practical Load Balancing

Policy	$\rho = 0.5$	$\rho = 0.8$
Min. Queue Length	1.010 ± 0.018	1.331 ± 0.069
Min. Ave. Que. Len.	1.011 ± 0.019	1.331 ± 0.068
Min. Est. Resp. Time	1.010 ± 0.018	1.330 ± 0.071
Min. Est. Str. Factor	1.010 ± 0.018	1.332 ± 0.069
Min. Utilization	1.011 ± 0.019	1.445 ± 0.101
Min. Unfinished Work	1.000 ± 0.016	1.286 ± 0.073
Min. Finished Work	1.022 ± 0.021	1.597 ± 0.125
No Load Sharing	2.001 ± 0.101	4.796 ± 0.486

balance the system at higher utilizations since the arrival rate is higher. This explains the high mean response time of the "Min. Finished Work" policy at ρ =80%. Although long jobs represent a small fraction of all jobs their impact on the performance of "Min. Finished Work" policy is evident since they stay in the system for long periods of time. Among all load measures that were compared in Tables 3 and 4 the queue length is perhaps the simplest. Nevertheless, both tables suggest that it is as effective as other measures in its potential to improve the system performance through load balancing. According to this policy each new job is assigned to run on the node that has the least number of jobs at the job arrival time. While other measures sometimes perform better, the difference is not significant and may not justify using complex load measures, e.g., the mean stretch factor. This conclusion was verified by comparing the load measures over a wide range of system utilization. Figure 2 shows the results of simulating a system with 10 nodes. The service time has an exponential distribution. The figure shows the mean response time versus system utilization using different load measures to balance the system. The two limiting cases: M/M/1 (no load sharing) and M/M/10 (perfect load sharing) are also included. Figure 3 shows the simulation results when the service time distribution is changed from exponential to hyper-exponential. In both cases it is hard to distinguish between the performance of policies that use the queue length, the mean queue length, the estimated mean response time, or the mean stretch factor. Though the "Min. Unfinished Work" offers a slightly better performance, its approximate version performs worse than other measures, particularly at high utilizations. This is due to the approximation errors in estimating the total unfinished work as discussed earlier. Note also that the "Min. Utilization" policy performs worse than other measures except the approximate version of "Min. Unfinished Work", particularly at high utilizations. At these utilizations most nodes are fully utilized and jobs are assigned to nodes arbitrarily since ties are broken randomly. In other words, this policy does not distinguish between nodes with different number of jobs or different amount of unfinished work as long as they have been equally busy during the last interarrival time period.

Table 4. Practical Load Balancing–Heterogeneous Jobs

Policy	$\rho = 0.5$	$\rho = 0.8$
Min. Queue Length	1.037 ± 0.073	1.420 ± 0.225
Min. Ave. Que. Len.	1.041 ± 0.072	1.398 ± 0.202
Min. Est. Resp. Time	1.039 ± 0.075	1.411 ± 0.195
Min. Est. Str. Factor	1.039 ± 0.075	1.413 ± 0.208
Min. Utilization	1.043 ± 0.078	1.297 ± 0.246
Min. Unfinished Work	1.030 ± 0.072	1.339 ± 0.217
Min. Finished Work	1.051 ± 0.078	1.649 ± 0.229
No Load Sharing	2.112 ± 0.274	4.559 ± 0.826

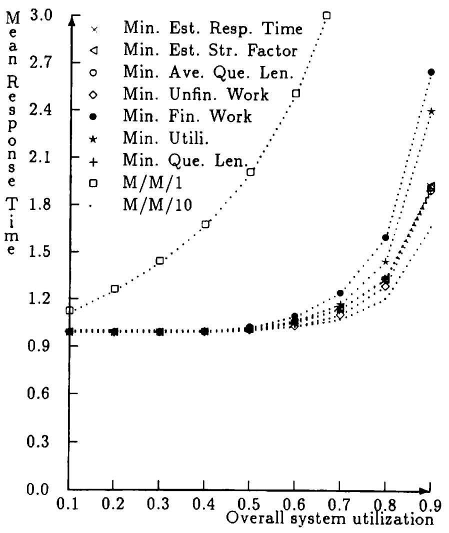


Figure 2. Practical Load Balancing–Homogeneous Jobs

6. CONCLUSIONS

Load balancing aims at improving system performance via the judicious distribution of the workload among the constituents

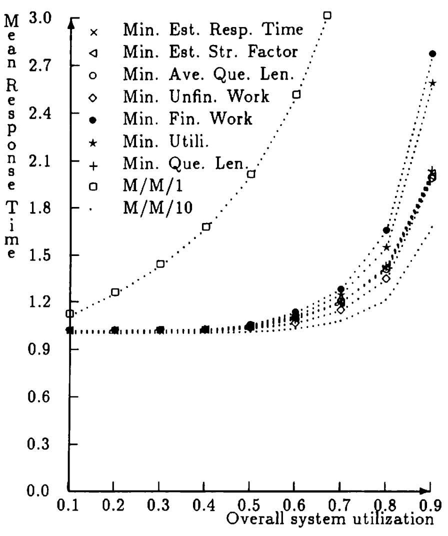


Figure 3. Practical Load Balancing–Heterogeneous Jobs

nodes. Incorporating a load balancing strategy requires a load measure to compare different nodes and a measure of the system balance to be used in making allocation decisions. We chose the variance of node loads as an appropriate measure of the system balance. Several measures of load were proposed and compared using oracle simulation. The oracle simulation guarantees that each new job finds the system in the most balanced state. The simulation results indicate that load measures can be grouped in two sets. The first set includes the instantaneous queue length, mean queue length, utilization, mean response time and mean stretch factor. Balancing the load using any of these measures yields a substantial improvement in system performance. The other set of load measures failf to exhibit performance significantly better than the no load sharing case. This result was found to be true in the practical versions of load balancing policies that aim at reducing the current system imbalance and do not use future information. The performance of practical policies compares favorably with ideal policies. Among the load measures that proved to be successful, the queue length is the simplest, yet it is as effective as more complex measures in its potential to improve the system performance. This result does not come as a surprise. It agrees with the intuition that *Join the Shortest Queue* is an excellent policy for load sharing in homogeneous systems.

Finally, we note that while we used oracle simulation for a specific application, namely, to compare load sharing policies, the technique can be used to optimize other decision processes as well.

ACKNOWLEDGEMENT

This material is based on work supported by the National Science Foundation (Grants DCR-8352098, CCR-8619663, and CCR-8703049), the Naval Ocean Systems Center, US WEST Advanced Technologies, and the Washington Technology Center. Additional partial support was provided by Bell Communications Research, Boeing Computer Services, Digital Equipment Corporation, Tektronix, Inc., the Xerox Corporation, and the Weyerhauser Company.

REFERENCES

Banawan, S.A., and J. Zahorjan (1989),"Load Sharing in Heterogeneous Systems," In *Proceedings of the 1989 IEEE INFOCOM Conference*, IEEE, Washington, D.C., 731–739.

Bryant, R., and R.A. Finkel (1981), "A Stable Distributed Scheduling Algorithm," In *Proceedings of the 2nd International Conference on Distributed Computing Systems*, 314–323.

Coffman, E.G., and P.J. Denning (1973), *Operating Systems Theory*, Prentice-Hall, Englewood Cliffs, NJ.

Eager, D.L., E.D. Lazowska, and J. Zahorjan (1986), "A Comparison of Receiver-initiated and Sender-initiated Adaptive Load Sharing," *Performance Evaluation 6*, 1, 53–68.

Eager, D.L., E.D. Lazowska, and J. Zahorjan (1988), "The Limited Performance Benefits of Migrating Active Processes for Load Sharing," In *Proceedings of the 1988 ACM SIGMETRICS Conference*, 63–72.

Efe, K. (1982), "Heuristics Models of Task Assignment Scheduling in Distributed Systems," *IEEE Computer 15*, 6, 50-56.

Ferrari, D., and S. Zhou (1986), "A Load Index for Dynamic Load Balancing," In *Proceedings of the 1986 Fall Joint Computer Conference*, 684–690.

Korry, R. (1986), "A Load Sharing Algorithm for a Workstation Environment," M.S. Thesis, Department of Computer Science, University of Washington, Seattle, WA.

Krueger, P., and R. Finkel (1984), "An Adaptive Load Balancing Algorithm for a Multicomputer," Technical Report #539, Computer Science Department, University of Wisconsin, Madison, WI.

Leland, W., and T. Ott (1986), "Load Balancing Heuristics and Process Behavior," In *Proceedings of PERFORMANCE'86 and ACM SIGMETRICS 1986,* 54–69.

Livny, M., and M. Melman (1982), "Load Balancing in Homogeneous Broadcast Distributed Systems," In *Proceedings of the 1982 Computer Network Performance Symposium*, 47-55.

Lo, V.M. (1988), "Heuristics Algorithms for Task Assignment in Distributed Systems," *IEEE Transactions on Computers C-37*, 11, 1384-1397.

Pulidas, S, D. Towssley, and J. Stankovic (1988), "Imbedding Gradient Estimators in Load Balancing Algorithms," In *Proceedings of the IEEE 8th International Conference on Distributed Computing Systems*, 482-490.

Stankovic, J.A., and I.S. Sidhu (1984), "An Adaptive Bidding Algorithm for Processes, Clusters, and Distributed Groups," In *Proceedings of the IEEE 4th International Conference on Distributed Computing Systems*, 49–59.

Tantawi, A.N., and D. Towsley (1985), "Optimal Static Load Balancing," *Journal of the Association for Computing Machinery 27*, 2, 323-337.

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

Zhou, S. (1987), "Performance Studies of Dynamic Load Balancing in Distributed Systems," Ph.D. Thesis, Department of Computer Science, University of California, Berkeley, CA.

Zhou, S., and D. Ferrari (1987), "An Experimental Study of Load Balancing Performance," Technical Report UCB/CSD 87/336, Computer Science Department, University of California, Berkeley, CA.