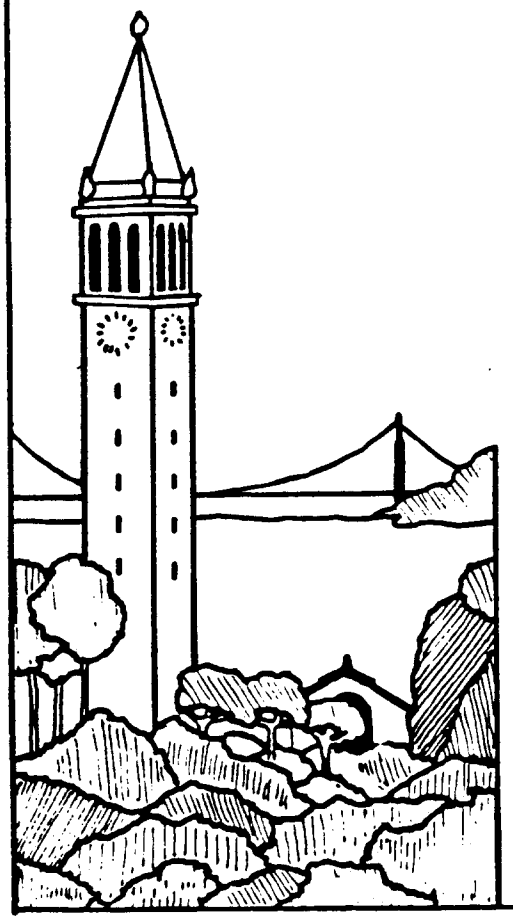
An Experimental Assessment of Resource Queue Lengths as Load Indices

Songnian Zhou



Report No. UCB/CSD 86/298

June 1986
PROGRES Report No. 86.3

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

An Experimental Assessment of Resource Queue Lengths as Load Indices †

Songnian Zhou
Computer Systems Research Group
Computer Science Division
University of California, Berkeley

ABSTRACT

Load indices that accurately reflect the current loads at computer system resources are crucial to the success of any dynamic load balancing scheme. However, few load indices have been experimentally validated as being suitable for load balancing. We conduct such a validation study for the resource queue lengths. We find that the CPU and disk queue lengths have very high correlation to the response times of CPU and I/O bound jobs, respectively. However, for the type of system that we studied, the system load changes very rapidly, making the response time high unpredictable. Simulation results suggest that load balancing will drastically reduce load fluctuation. The CPU is by far the most heavily used resource in the systems we studied. While this is also true in many other environments, measurement studies are called for before reaching such a conclusion in other systems.

1. Introduction

As distributed computer systems become increasingly popular, the significance of load balancing is also being more widely recognized. Load balancing attempts to improve system performance by redistributing the workload submitted to the system by the users. It has been demonstrated, using simple analytic and simulation models, that load balancing can yield significant performance gains without requiring upgrading the system by adding more hardware [Livny and Melman 1982; Eager et al. 1986; Zhou 1986]. A prerequisite for load balancing is the availability of system load indices that properly characterize the current system loading

† This work was sponsored by the Defense Advanced Research Projects Agency (DoD), ARPA Order No. 4031, monitored by the Naval Electronics Systems Command under contract No. N00039-C-0235, and by the National Science Foundation under Grant DMC-8503575. The views and conclusions contained in this document are those of the author and should not be interpreted as representing official policies, either expressed or implied, of the Defense Research Projects Agency or of the US Government.

condition. Such load indices are used by load balancing mechanisms to make job placement decisions. Although a large number of load indices have been proposed and used by the researchers in this field, few of them have been validated experimentally as being appropriate for load balancing purposes. In this report, we conduct such a validation study for a particular load index using measurements of production system loads.

A very important system performance metric is the average response time of the jobs submitted by the users. The minimization of the average response time is often the goal of load balancing, especially for interactive computing environments. Intuitively, the response time of a job is dependent on the *load* of the host on which it runs; the higher the load, the longer the response time. Therefore, a load index should ideally have a monotonic relationship with the job response time. The execution of a job requires certain amounts of various types of resources; among them are CPU cycles, disk bandwidth, memory space, and communication bandwidth. The response time of a job is determined by (1) the amounts of service time the job needs at the resources, (2) the amounts of waiting time the job spends at the resources due to contention, and (3) the synchronization and communication delays, such as locking, signals, and waiting for a message to arrive. Since (3) usually has a large variance, and is highly unpredictable, we decide not to study it here, but rather concentrate on the first two factors, which are related to the job's resource needs and the level of resource contention, respectively.

Ferrari pointed out, using mean value analysis of closed queueing networks, that a linear combination of the queue lengths of various resources should be a good predictor of the job response time, provided that the system is in steady state and that the queueing disciplines of the resources are FCFS, PS, or LCFSPR [Ferrari 1985]. However, these assumptions are not generally satisfied in real world computer systems. For example, in Berkeley UNIX†, a multi-class priority scheduling algorithm with round-robin preemption is used for the CPU, with the process priorities decreasing while running, and increasing while waiting to run. Jobs that return from disk I/O are given high priority in the hope that they will go back to the disk soon. Since scheduling discipline for the disks in Berkeley UNIX is one-way scan, an incoming I/O request may find itself right ahead of the disk arm, and thus be serviced first. With the complicated queueing disciplines adopted for various resources, it is not clear whether a strong linear relation exists between the resource queue lengths and the job response time. The closed system and steady state assumptions in Ferrari's model are also not generally satisfied in computer systems; users log on and off, and the number of jobs submitted can fluctuate widely. On the other hand, modeling experiences have repeatedly demonstrated that fairly simple models can provide very good results, even though some of the assumptions of the model do not strictly hold.

---

† UNIX is a trademark of AT&T Bell Laboratories.

We conducted a series of measurement experiments on production systems under both their natural workloads and artificial workloads, with the following three objectives:

(1) to evaluate the suitability of the resource queue lengths as load indices, and to study the sensitivity of different types of jobs to the queue lengths;

(2) to investigate the level of contention for various types of resources in a computer system, in order to assess their relative importance;

(3) to determine the rate at which the system load changes, and thus the effectiveness of load prediction.

In the next section, we describe the experimental set-up. In Section 3, we discuss the system load characterization in order to identify the contended resources, those causing queueing delays to jobs. The relationship between the queue lengths at the CPU and the disks, and the response times of CPU and I/O bound jobs are studied in Section 4. We study the load change rate in Section 5. Finally, we summarize our experimental findings in Section 6.

2. The Experimental Set-up

The first requirement of our measurement experiments was the availability of the resource queue lengths. Their values should be updated with a high frequency in order to reduce the random sampling error. Several resource queues were identified, and their instantaneous lengths (number of processes waiting or being served) were made available by modifying the UNIX kernel. The queues involved were:

(1) **CPU ready queue** (*cpu*): the number of processes that are loaded and waiting to be scheduled to run or running. The process in execution is included.

(2) **Disk process queue** (*dev*): the number of *processes* that are residing in the disk subsystem, waiting for their disk transfers to be processed. This is the aggregate queue for *all* the disks attached to the host. It was possible to get the queue lengths of the individual disks, but more system overhead would be incurred, and knowledge of which disk a particular I/O operation is directed to would be required, making analyses of the measurement data more complicated. We decided to concentrate on the disk subsystem as a whole.

(3) **Disk request queue** (*tdev*): the number of disk I/O *requests* that are waiting to be or being processed. The processes that initiated them may or may not be waiting (synchronous or asynchronous I/O, respectively), thus this queue is a superset of the disk process queue, and represents the actual load at the disks.

The instantaneous queue lengths were sampled every 10 milliseconds in the clock interrupt routine of the UNIX kernel, and used to compute the average queue length over a second (100 samples). We feel that this should provide enough accuracy, and that we do not need to get more than one value per second. To reduce the overhead of tracing the system, in-kernel buffering of the queue lengths was used, so that the tracing program needed only to extract the values from the

kernel at longer intervals (for example, once a minute). The overhead of system tracing, in terms of CPU time, was found to be below 0.5%.

There are other types of resources for which queue lengths do not exist explicitly, or are not appropriate as load indices. For example, a process occupies some memory space when running on the CPU, and, in addition, some memory buffer space while doing I/O. We decided to use host-wide variables such as the number of free memory page frames, and the paging and swapping rates as indices for memory contention. The network is another example of resource for which queue length is not appropriate. Processes may be waiting at a communication socket for messages to come, but not because of network bandwidth contention. Since the utilization of networks is usually low [Lazowska et al. 1985], and the protocol overhead for communication is already accounted for by the CPU queue length, we decided not to consider the network explicitly.

Most of our measurements were conducted on a production machine, *Ucbarpa*, which is a VAX-11/780 running Berkeley UNIX 4.3BSD, with 4 MB of main memory and 3 disk drives. The workload on it is usually quite heavy during the day, with load average† ranging from 3 to 10. We traced the system load indices discussed above for a number of days, with one eight-hour session per day. A number of other system load metrics, such as the one-minute load average, each individual disk's transfer rate and utilization, the multiprogramming level, and the percentages of CPU time spent executing system kernel code and user code, were also collected. At the same time, we ran benchmark jobs periodically and measured their response times (r). The data collected enabled us to conduct a series of regression studies to evaluate the potential of the resource queue lengths for predicting the job response times.

Three types of jobs were used at different times.

(1) A *troff* job (TROFF) that is mostly CPU bound, with some I/O. Two input text file sizes were used. We chose *troff* jobs as benchmarks because they are often big and therefore good candidates for load balancing.

(2) A purely CPU bound benchmark (CPU) that performs a lot of arithmetic computations.

(3) A predominantly I/O bound benchmark (IO) that copies a large file block by block in reverse order to defeat the read ahead mechanism in UNIX. Since the size of the file is chosen to be larger than the size of the system disk cache, which uses an LRU replacement algorithm, all I/O operations are forced to the disk.

To minimize the effect of the extra workload introduced by the benchmarks, we only ran one type of benchmark for each session. Each of the benchmarks consumed 4-10% of the total CPU time.

---

† *The load average* in UNIX is an exponentially smoothed average of the total number of processes in the CPU queue and the disk queues, and is widely used as a system load index.

3. System Load Characterization

From the measurements of the system queue lengths and other variables, we were able to study the system's load characteristics. Below is a summary of our findings.

CPU:

The CPU is usually heavily contended. Its queue length was observed to be above 3 during the day, and the utilization at or near 100% (no or little idle time).

Disks:

They are usually lightly loaded. The average queue length at the busiest disk is around 0.3 when the system load average is 5-10. The corresponding disk utilization (percentage of time the disk is busy) is about 25%. This can be explained by the large main memory size and the effectiveness of the large disk cache, which is reported to have an average block hit ratio of 85% [Leffler et al. 1984]. Figures 1 and 2 show the distributions of the utilization and transfer rate for the most heavily used disk observed during a day of heavy system load.

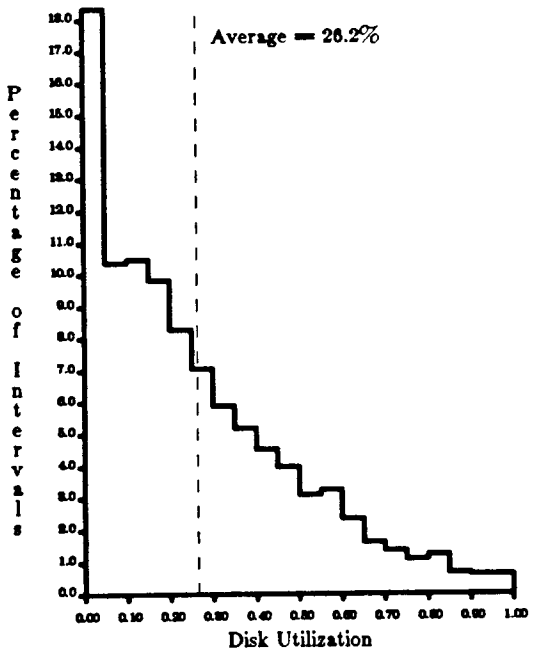


Figure 1. Disk Utilization Distribution.
(measured in 5 second intervals)

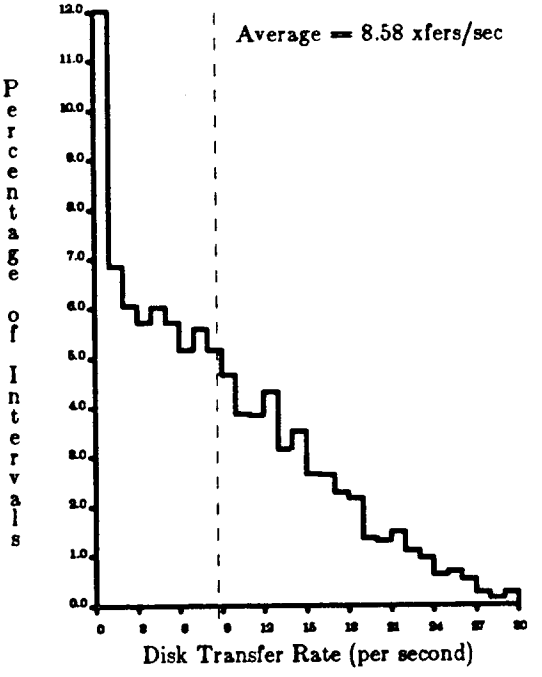


Figure 2. Disk Transfer Rate Distribution
(measured in 5 second intervals)

Memory:

There is usually plenty of memory, thus few pages that are currently being used are paged out, and few forced process swap-outs occur. The distributions of the number of free memory pages and the paging rates for a day of heavy load are shown in Figures 3 and 4.

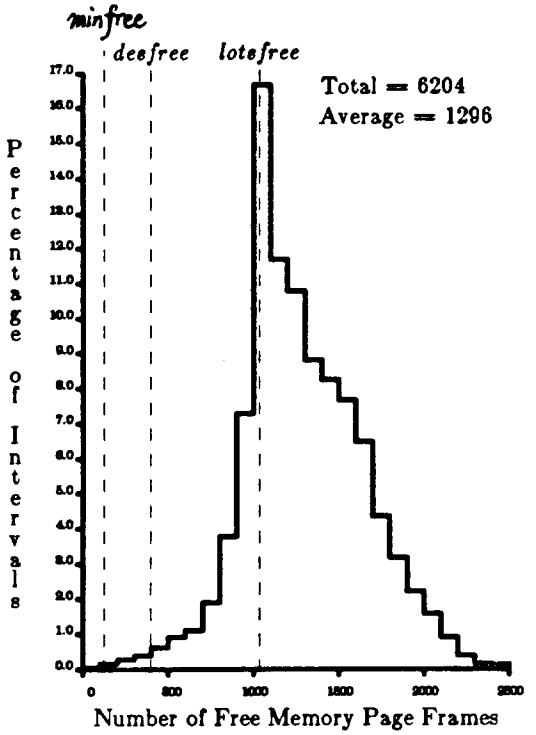


Figure 3. Distribution of Free Memery Page Frames (measured in 5 second intervals)

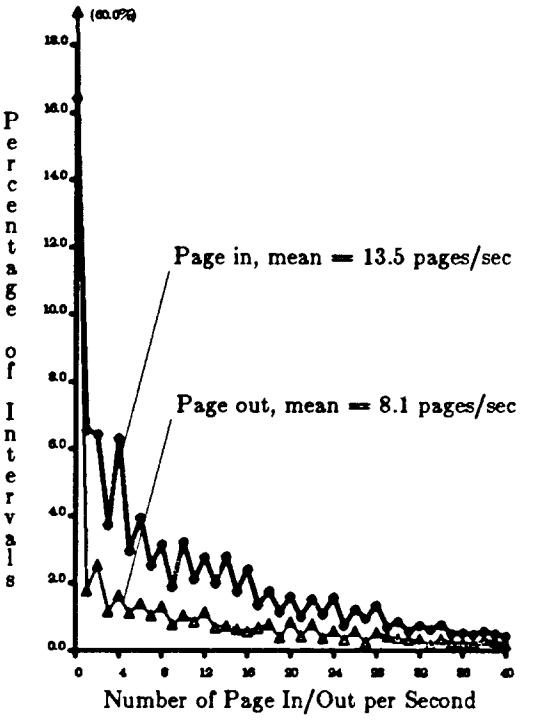


Figure 4. Paging Rates (measured in 5 second intervals)

There are three important parameters in the memory management system of Berkeley UNIX. If the number of free main memory page frames is above *lotsfree*, the page daemon is not activated, thus no page-outs occur. This was the case 80% of the time on the day the data in Figures 3 and 4 were collected. If the number of free memory pages falls below *lotsfree*, the page daemon will scan the page frames and free up those that have not been used for a while. Processes will be swapped out only if the number of free page frames falls below a very low level, *minfree*, the average number of free frames over a recent time interval is below a desirable level, *desfree*, and the CPU load is very high. In that case, only processes that have been dormant for quite a while are usually swapped out. We observed a swapping rate of about 4 processes per minute during the measurement session.

From the above observations, we conclude that the CPU is the bottleneck of the system. This is not peculiar to the machine we measured or to Berkeley UNIX running on a VAX; similar observations have been reported for other systems too.

For example, Lazowska and his coworkers found that, in distributed systems based on diskless workstations running SUN's version of Berkeley UNIX, the V system, and the Domain software of Apollo computers, the utilizations of the file server CPU's are two or more times higher than those of the disks and the networks [Lazowska et al. 1985]. Therefore, the CPU queue length can be expected to be the most important load index. In Ferrari's linear combination of queue lengths, the CPU term will tend to the dominant one, except for the extremely I/O bound jobs. This is confirmed by the analyses presented in the next sections. Considering that the memory size will increase very rapidly, hence larger and larger disk caches will be employed to cut down the slow disk traffic, and that greater network traffic will imply more CPU time spent in the communication protocols, it is probably safe to say that, despite the continuing increase in speed, the CPU will remain the most contended resource for a number of years to come.

4. Queue Lengths as Load Indicators

As pointed out earlier, a satisfactory load index should have a strong correlation with the job response time. We introduce the distinction between a load *indicator* and a load *predictor*. The value of an indicator should provide a clear measure of the *current* system load condition in terms of the response time a job would experience if it were executed at the time the load indicator is measured. On the other hand, the value of a predictor should provide a good estimate of the system's load condition in the near *future* in terms of the response time of a job about to begin. Hence, the requirements for a load predictor are much more stringent than those for a load indicator: a good predictor is usually a good indicator, but not necessarily vice versa. For load balancing, we need a good load predictor, but, as a start, we evaluate resource queue lengths as load indicators in this section.

We use the average queue lengths *during* the entire job execution period as our load indicators, and compute the correlation between them and the job response times. Linear regression tests are also performed to decide the amount of improvement we can obtain by using both CPU and disk queue lengths. The results are summarized in Table 1.

As can be seen in the table, the correlation between the average CPU queue length and the response time of a CPU bound job (TROFF or CPU) is very close to 1 under a very wide spectrum of system load conditions, both live and artificial (Sessions **I, II, III, IV,** and **V**). On the other hand, the correlation between the aggregate disk queue length and the response time is much lower, and the amount of improvement obtained by taking it into consideration is negligible (the correlations in the regression tests are only slightly better than that with CPU queue length alone). This is not only true for live workloads that usually do not cause much contention on the disks, but also with artificial workloads having high disk demands. It should be noted that with the live workloads, only the aggregate disk queue length is available, and the correlation between it and the job response time can be expected to be lower than that when the queue length of the disk to which

Table 1. The Correlation between Job Response Time and Queue Lengths

Session	I	II	III	IV	V	VI	VII
Job Type	TROFF	TROFF	TROFF	CPU	CPU	IO	IO
Load Type	*live*	*live*	*live*	*live*	*art,mix*	*art,io*	*art,mix*
sample size	109	104	188	52	22	53	101
cor(cpu,r)	0.9699	0.9767	0.9528	0.9505	0.9838	0.7459	0.0666
cor(dev,r)	0.4873	0.8179	0.7401	0.5739	0.4758	0.9429	0.8538
cor(tdev,r)	0.5097	0.8388	0.6582	0.6354	0.4686	0.9346	0.8412
reg(cpu,dev;r)	0.9702	0.9795	0.9604	0.9505	0.9918	0.9441	0.8611
reg(cpu,tdev;r)	0.9703	0.9794	0.9588	0.9507	0.9927	0.9346	0.8597
cor(cpu+dev,r)	0.9620	0.9793	0.9385	0.9446	0.9754	0.9394	0.3897
cor(cpu+tdev,r)	0.9537	0.9788	0.9564	0.9430	0.9618	0.9333	0.4547
mean(cpu)	3.3028	4.1186	3.0932	2.2770	4.2539	0.3606	5.4651
mean(dev)	0.4930	0.4501	0.4654	0.4574	3.3939	2.0377	3.8015
mean(tdev)	0.9852	0.8317	0.9029	0.7431	4.2289	2.5192	4.7390
mean(r)	101.11	136.63	44.36	201.5	897.8	97.69	119.04
stdv(r)	74.48	82.64	29.44	98.6	572.7	20.13	31.95

cor(a,b) - correlation coefficient between a and b;
reg(a,b; c) - correlation coefficient obtained by doing linear regression on c using a and b;
r - job response time;
mean - mean queue length during the whole measurement session, as a measure of the average load;
stdv - standard deviation of the queue length distribution during the whole measurement session, as a measure of the load fluctuation;
live - session under live, production workload;
art - session under artificial workload;
mix - artificial workload consisting of a mixture of CPU and I/O bound jobs;
io - artificial workload consisting of only I/O bound jobs.

the job's I/O is directed is used. With artificial workloads, however, a heavy traffic to/from a particular disk is generated, while all the other disks are virtually idle. Consequently, the aggregate disk queue coincided with the single disk queue, yet, we still observed a low correlation. We can safely conclude that the average CPU queue length alone is an excellent load indicator for CPU bound jobs. This finding is intuitive, yet very reassuring.

The situation with I/O bound jobs is slightly more complicated. Under a live workload, since the disks are not heavily loaded, the response time of an I/O bound

job is found to be insensitive to both the CPU and the disk queue lengths. To study the dependency of the I/O bound job's response time upon disk queue length, we were forced to use an artificial workload that generated a heavy traffic to/from a particular disk. Results similar to the CPU bound jobs were obtained: when the disk is contended for ($tdev > 2$), the correlation between the disk queue length and the response time is very high, whereas the CPU queue length has a much lower correlation (Sessions VI and VII). Considering both CPU and disk queues provides little improvement.

The above discussion can be summarized as follows: a good load indicator is job-type-dependent; for CPU bound jobs, the CPU queue length is predominant, whereas for I/O bound jobs, if there is high contention at the disk (which was rarely observed), the disk queue length is sufficient; otherwise, the job response time is insensitive to system load.

We have demonstrated experimentally that the resource queue lengths, if used properly, can be very good load indicators. Whether they are also good load predictors is determined by the rate at which workload fluctuates, in other words, by whether Ferrari's steady state assumption holds well in computer systems. We study this problem in the next section.

5. Queue Lengths as Load Predictors

To evaluate the ability of the resource queue lengths in predicting future jobs' response times, we used the average queue lengths for a period (15, 30 or 60 seconds) *before* the start of the job, instead of those *during* the execution of the job. Table 2 shows the various correlations for Sessions I, II and IV listed in Table 1.

Table 2. Queue Lengths as Predictors of Response Time.

	PAST15	PAST30	PAST60
Session I	*compare: cor(la, r) = 0.6964*		
cor(cpu,r)	0.6915	0.6928	0.6353
cor(dev,r)	0.1948	0.2932	0.2906
cor(tdev,r)	0.1639	0.3137	0.3210
Session IV	*compare: cor(la, r) = 0.6831*		
cor(cpu,r)	0.6866	0.7003	0.6867
cor(dev,r)	0.4130	0.4725	0.5076
cor(tdev,r)	0.3376	0.4522	0.4689

PAST15, PAST30, PAST60: average queue length for the 15, 30 or 60 seconds immediately before the job starts

The correlation coefficients between queue lengths and response times are much lower than in the "during" case, almost never exceeding 0.7. Prediction using average queue length is not much better than using the value of the load average (*la*) immediately before the job starts (see *compare*). We attempted to improve the prediction by using an exponential smoothing algorithm, but the improvement was very small. Furthermore, the higher the load, the greater the fluctuation observed, and the poorer the prediction (Session II). The consistently poor correlation in a number of sessions seems to suggest that there is a "sound barrier" that defies any attempt to predict the future load. The resource queue lengths are changing so rapidly that their values before the job starts poorly predict those during the job execution, and the job's response time cannot therefore be accurately predicted. In other words, the steady state assumption in Ferrari's model does not hold. This is confirmed by our measurements of the rate of change of the CPU queue length. Figure 5 shows the distribution of net CPU queue length changes during 30-second intervals over an entire session (about eight hours). † As can be observed, 33% of the time the net change is 3 or beyond. For an average queue length of 6, which is fairly high, this represents a 50% change in CPU load. For the machine we measured, the average net change in CPU queue length in a 30-second interval is 2.31.

The wide and rapid changes in system load make predicting future loads quite difficult, thus load balancing policies based on queue lengths are bound to make mistakes. The prediction of job response times gets worse as the job gets longer, which is bad for load balancing because load balancing policies are mostly concerned with long jobs, due to the non-zero costs of moving jobs over the network. However, it should be pointed out that, for load balancing, we are interested in *comparing* the loads of a number of hosts in order to choose one that is lightly loaded, rather than in determining the absolute response times of the job if it were executed on each of the eligible hosts. Another factor that is likely to ease the difficulty of prediction is that load balancing, if effective, should tend to reduce the fluctuations of the load on each machine, thereby improving the accuracy of response time prediction. In other words, load balancing should provide a responsive *negative feed-back* that will tend to stabilize machine loads. This conjecture was confirmed by the results obtained from a trace-driven simulator that we constructed for studying load balancing [Zhou 1986]. Records containing job arrival times and resource consumptions for all the commands submitted to a production system during a number of tracing sessions were used to drive a model that simulated the executions of the commands on a number of hosts, with and without load balancing. It was found that load balancing, besides effectively decreasing the average command response time, also tends to smooth out the temporal

† To make the measurements here compatible with those from the simulation to be discussed shortly, we use the instantaneous CPU queue length, rather than the average queue length over one second, which has been used up to this point.

fluctuations in the load of each machine. For a typical machine, the distributions of load changes during 30-second intervals as described above were computed for cases with and without load balancing, and are shown in Figure 6. The average net change in load decreased from 2.42 for the no-load-balancing case to 0.93 for that with load balancing.

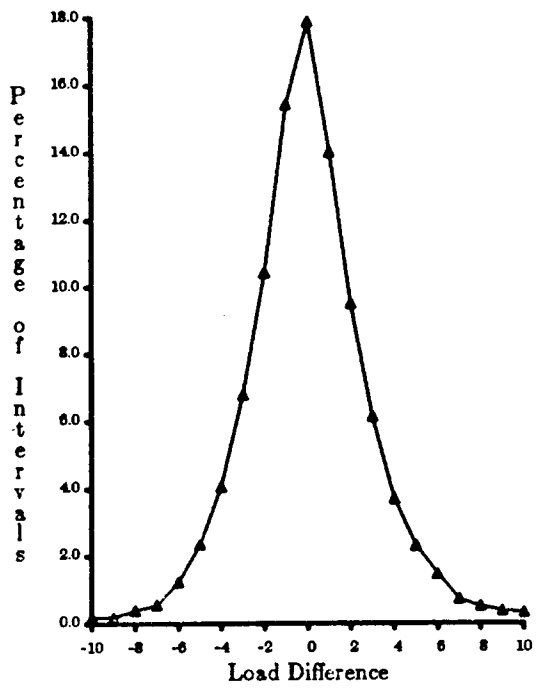


Figure 5. Load Change Frequency (in 30 Seconds) from Measurement

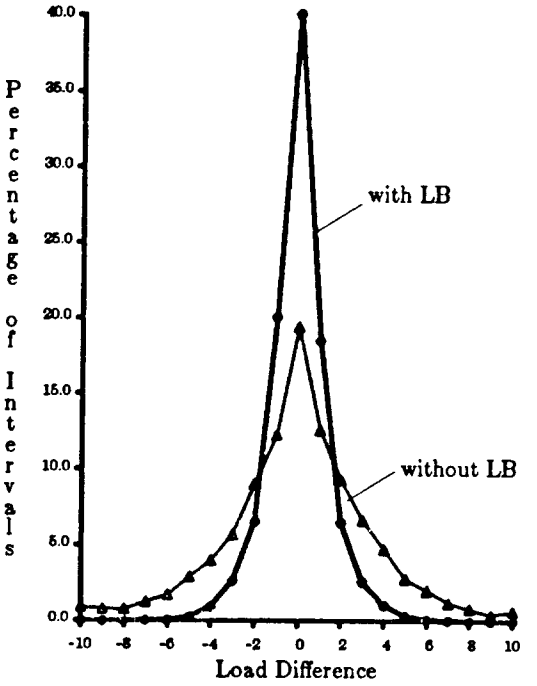


Figure 6. Load Change Frequencies (in 30 Seconds) from Simulation

6. Conclusions

The CPU is the most contended resource in systems similar to the one we measured, while there is usually plenty of main memory, so that little paging out and few forced process swap-outs occur. Traffic to and from the disks is light, and the queueing times processes spend at the disks are low. The addition of more disk drives is usually because of the need for more on-line storage space, rather than of constraints on disk bandwidths. This is also true for many other systems, and is going to be true for a number of years to come.

The resource queue lengths can be excellent load indicators if used properly. For CPU bound jobs, the CPU queue length is closely related to the response time, whereas the disk queue lengths have negligible correlations to it. For I/O bound jobs, the particular disk queue length reflects the response time very well when processes are queued up there. However, for live workloads characterized by short queues at the disks, the response time is insensitive to both CPU and disk queue

lengths.

The load of live systems changes very rapidly, making response time predictions difficult. On the other hand, load balancing is expected to smooth out the wide fluctuations, thereby improving the predictions. This conjecture is confirmed by simulation studies of load balancing.

7. Acknowledgements

The author is deeply indebted to Domenico Ferrari for his invaluable advice and continued support. Harry Rubin provided useful comments on earlier drafts and the benchmark program used in this work.

8. References

[Eager et al. 1986]

D. Eager, E. Lazowska, and J. Zahorjan, "Dynamic Load Sharing in Homogeneous Distributed Systems," IEEE Trans. Soft. Eng., SE-12,5, May 1986, pp. 662-675.

[Ferrari 1985]

D. Ferrari, "A Study of Load Indices for Load Balancing Schemes," Report No. UCB/CSD 85/262, Computer Science Division, University of California, Berkeley, October 1985.

[Lazowska et al. 1984]

E. Lazowska, J. Zahorjan, D. Cheriton, and W. Zwaenepoel, "File Access Performance of Diskless Workstations". Tech Report 84-06-01, Dept. of Comp. Sci, Univ. of Washington, June 1984.

[Leffler et al. 1984]

S. Leffler, M. Karels, and K. McKusick, "Measuring and Improving the Performance of 4.2 BSD," Proceedings of the Salt Lake City Usenix Conference, pp. 237-252, June 1984.

[Livny and Melman 1982]

M. Livny and M. Melman, "Load Balancing in Homogeneous Broadcast Distributed Systems," Proc. ACM Computer Network Performance Symposium, April 1982,

[Zhou 1986]

S. Zhou, "A Trace-Driven Simulation Study of Dynamic Load Balancing," Report, to appear, UCB/CSD, Computer Science Division, University of California, Berkeley, June 1986.