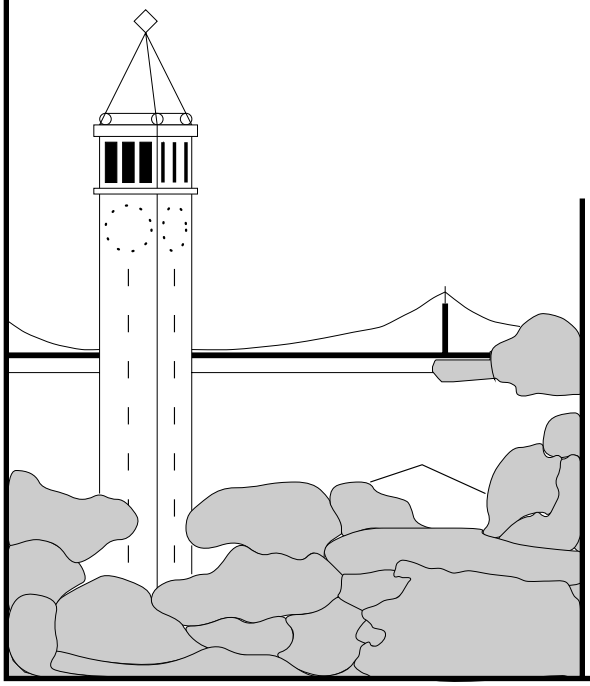
Predicting queue times on space-sharing parallel computers

Allen B. Downey



Report No. UCB/CSD-96-906

July 1996

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

Predicting queue times on space-sharing parallel computers

Allen B. Downey *

July 1996

Abstract

We present statistical techniques for predicting the queue times experienced by jobs submitted to a space-sharing parallel machine with first-come-first-served (FCFS) scheduling. We apply these techniques to trace data from the Intel Paragon at the San Diego Supercomputer Center and the IBM SP2 at the Cornell Theory Center. We show that it is possible to predict queue times with accuracy that is acceptable for several intended applications. The coefficient of correlation between our predicted queue times and the actual values from the simulated schedules is between 0.65 and 0.7.

1 Introduction

On space-sharing parallel machines, it is useful to be able to predict how long a submitted job will be queued before processors are allocated to it. Some of the applications of these predictions are:

Load metrics: They provide a measure of load that is more concrete than abstractions such as load average, allowing users to make decisions about what jobs to run, where to run them and what size problems they can solve in an allotted time.

Internal resource selection: They allow malleable parallel jobs (jobs that are not limited to a specific cluster size, but can run on any number of processors) to choose a cluster size that will minimize their turnaround time (the sum of queue time and run time).

External resource selection: They allow distributed jobs to choose among various computing resources on a network, based on the quality of service they expect to receive at each host.

*EECS — Computer Science Division, University of California, Berkeley, CA 94720. Supported by NSF grant ASC-89-02825. email: downey@cs.berkeley.edu

This paper presents a model of the workload on a parallel machine and shows how to use this model to predict queue times. We present observations of the workload on the Intel Paragon at the San Diego Supercomputer Center (SDSC) and the IBM SP2 at the Cornell Theory Center (CTC), and show that they fit the proposed model well. We use these workloads and a trace-driven simulator to evaluate the proposed techniques for predicting queue times. We conclude that our techniques can predict queue times on real machines with accuracy sufficient for the proposed application.

1.1 Motivating problem

Of the three applications listed above, the focus of this paper is internal resource selection — choosing a cluster size on a space-sharing parallel computer. We assume that users have parallel programs that can run on a variety of cluster sizes, and that they know the run time of these programs as a function of the cluster size (at least approximately). We would like to choose a cluster size that will minimize the expected turnaround time for the job; in other words, choose a cluster size n that minimizes $E[Q(n)] + R(n)$, where $R(n)$ is the run time of the job on n processors and Q(n) is a random variable representing the queue time the job will have to wait until n processors are available.

Consider, for example, a small parallel computer with only two nodes. The user might have an application that takes 1 hour to run on one node or 35 minutes to run on two nodes. If the user arrives at the system and discovers that one of the nodes is busy, he might choose to run the job immediately on the free node or wait for the other job to complete and then run on both nodes. If the expected time until the first job completes is less than 25 minutes, the user might choose to wait.

A crucial piece of information for making that estimation is the age of the running job. According to the *past repeats* heuristic, the expected remaining lifetime of a job is equal to it's current age. Thus, in this example, the user might choose to wait if the age of the running job is less than 25 minutes.

But the past repeats heuristic is based on a particular distribution of process lifetimes (see Section 2:Related Work, below) that may not describe the workload of the system in question. For example, if all jobs in the system have the same duration, then the expected remaining lifetime of a job *decreases* as the job ages and the user in this example would make the wrong decision.

Thus the goal of this paper is to show how to use observations of the distribution of lifetimes to calculate predictions of queue times.

1.2 Internal resource selection

It is not clear exactly *when* a malleable job should choose a cluster size. There is a tradeoff between the interests of the user, who would like to make a decision as late as possible in order optimize the expected performance of a job, and the

system, which would like to encourage jobs to allocate fewer processors when the system load is high.

For example, if jobs are forced to choose a cluster size when they enter the queue, they might not be able to predict the future state of the system with enough accuracy to guarantee good performance. On the other hand, if jobs are allowed to choose their cluster sizes after the jobs in the system have terminated, they are likely to allocate more processors (often *all* of them) than would be appropriate for the current load.

We feel that an appropriate compromise is to allow the user to postpone his decision until the job reaches the head of the queue. It is our goal in future work to validate this decision by evaluating the performance of the system as a whole when users do this sort of local optimization.

In the meantime, we will focus on the problem of making predictions based on the state of jobs running in the system, ignoring jobs waiting in queue. We will discuss an extension to these techniques that deals with queued jobs in Section 8.1: Future Work.

1.3 Overview

Section 2 describes related work. Section 3 presents the workload and system models we will be using. Section 4 presents our observations of the distribution of lifetimes of jobs on the Intel Paragon at SDSC, and proposes a new model that describes this workload.

In Section 5 we present new techniques for using the distribution of job lifetimes to predict queue times for arriving jobs. Section 6 describes a trace-driven simulator we used to evaluate these techniques and presents our results.

Section 7 evaluates the robustness of the proposed techniques in other environments (the IBM SP2 at the Cornell Theory Center). Finally, Section 8 summarizes our findings and suggests areas of related future work.

2 Related work

2.1 Distribution of lifetimes

Previous work [8] [4] has shown that sequential jobs on UNIX machines have a distribution of process lifetimes with the following functional form:

$$cdf_L(t) = Pr\{L < t\} = 1 - \left(\frac{t}{t_{min}}\right)^k \qquad t > t_{min} \quad (1)$$

where L is a random variable indicating the CPU lifetime of the process. The constant k varies from workload to workload, but is generally near -1.0.

From this distribution we can calculate the distribution of lifetimes conditioned on the current age of a job:

$$cdf_{L|a}(t) = Pr\{L < t | L \geq a\} = 1 - \left(\frac{t}{a}\right)^k \qquad t > t_{min} \quad (2)$$

where a is the cpu age of the process. This distribution has the property that the median remaining age of a process is equal to its current age. This is consistent with the past repeats heuristic that is commonly hypothesized.

In [4], we develop a technique for using this distribution to predict the slowdown experienced by sequential processes on a network of workstations. We used these predictions to derive a policy for using preemptive migration (migration of active processes) for load-balancing.

For parallel applications, several papers have reported process lifetime distributions from a variety of architectures:

Hotovy *et al.* [6] [7] report the workload characteristics of the IBM SP2 at the Cornell Theory Center. Steube and Moore [10] and Wan *et al.* [11] describe the workload and scheduling policy on the Intel Paragon at the San Diego Supercomputer Center; [10] shows that the distribution of lifetimes of batch jobs fits the uniform-log distribution model proposed here (for jobs less than six hours in duration). Feitelson and Nitzberg [3] describe the workload on the Intel iPSC/860 at NASA Ames. Finally, Windisch *et al.* [12] compare the workloads from SDSC and NASA Ames. None of these papers discusses the shape of the distribution of lifetimes or uses it to develop a workload model.

2.2 Predicting resource use

Devarakonda and Iyer [2] use information about past executions of a program to predict the lifetime of processes, as well as their file I/O and memory use. For each past execution of a program, they plot a point in the 3-space (cpu lifetime, file I/O, memory use) and use k-cluster analysis to identify classes of programs with similar resource usage. For a given program execution, then, they can calculate the probability that the execution will fall into each cluster. Their predictions are the weighted averages of the median values from each cluster, with the weights given by the calculated probabilities.

This work differs from ours in that Devarakonda and Iyer predict the resource requirements of a process before it begins execution. Thus, they use information about previous executions rather than observations of the current behavior of a process. Also, their predictions are specific values rather than distributions of possible values.

2.3 Using predicted queue times for resource selection

Atallah *et al.* [1] propose the idea of choosing a cluster size that minimizes the turnaround time (queue time plus run time) of the job, but they do not address the question of how to predict queue time as a function of the number of

processors requested. Like us, they raise the issue of how this local optimization — minimizing the turnaround time of individual jobs — affects overall system performance, but they do not answer it. Our future work will attempt to do so, based on the mechanism presented here for predicting queue times.

3 Model of computation

We assume the following model of computation:

- Jobs are submitted to a space-sharing parallel computer and allocate some number of processors (a cluster) before they begin execution. The size of the cluster does not change during the execution of the program.
- A job may or may not wait in queue before processors are allocated to it. Jobs are allocated to processors in a first-come-first-serve (FCFS) discipline.
- Once a job is assigned to a cluster, it is not preempted until it completes.
- Once a job completes, the processors in its cluster are freed and may be allocated to another job.

This model does not consider timesharing or interactions between jobs in different clusters, such as communication interference. Thus, the turnaround time for a job is the time from submission to completion, which is the sum of the run time and the time the job spends in queue.

The total number of processors is likely to be greater than the typical cluster size, so most of the time several jobs will run simultaneously on disjoint clusters.

4 Lifetime distributions

During the calendar year 1995, 24946 jobs were submitted to the batch partition of the 400-node Intel Paragon at SDSC. Figure 1 shows the distribution of lifetimes for these jobs. Although there is a time limit on this machine of 12 hours, only 5% of the jobs ran to the limit (a few jobs ran longer by special permission).

Except for the longest and shortest jobs, this distribution is nearly straight (on a logarithmic x-axis); in other words, the logs of the lifetimes are distributed uniformly. Although we do not know any theoretical reason why lifetimes should have this *uniform-log distribution*, we will take advantage of this observation by fitting a straight-line approximation to the observed distribution. Using this approximation (instead of the observed values) simplifies the calculation of the conditional distribution of lifetimes (see Equation 4).

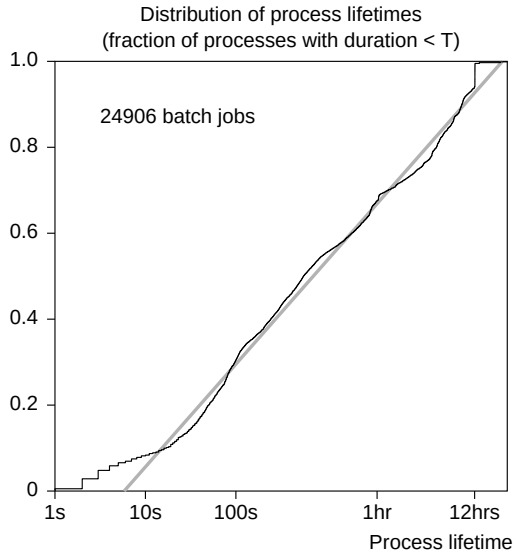


Figure 1: The distribution of lifetimes for 24906 batch jobs submitted to the Intel Paragon at SDSC.

We calculate a straight-line approximation by least-squares regression (omitting the longest 10% and the shortest 10% of jobs). The gray line in Figure 1 shows the fitted line. The R^2 value of this fit is over 0.99, indicating that the linear approximation is very good.

Table 1

	value	t-ratio
$intercept = \beta_0$	-.183	-430
$slope = \beta_1$	.104	1700

Thus our model of the distribution of lifetimes is

$$cdf_L = Pr\{L < t\} = \beta_0 + \beta_1 lnt \qquad t_{min} \leq t \leq t_{max} \quad (3)$$

where L is a random variable representing the lifetime of a job, $t_{min} = e^{-\beta_0/\beta_1} = 5.8$ seconds and $t_{max} = e^{(1.0-\beta_0)/\beta_1} = 24.2$ hours.

For this model we can calculate the distribution of lifetimes conditioned on the current age of a process:

$$
\begin{aligned}
1 - cdf_{L|a} &= Pr\{L > t | L > a\} \\
&= \frac{1 - cdf_L(t)}{1 - cdf_L(a)} \\
&= \frac{1 - \beta_0 - \beta_1 lnt}{1 - \beta_0 - \beta_1 lna} \qquad\qquad t_{min} \leq t \leq t_{max} \qquad (4)
\end{aligned}
$$

The median remaining lifetime of a job as a function of its current age is

$$median(L|a) = \sqrt{t_{max} \cdot a} - a$$

This property is unusual, since it implies that the remaining lifetime of a job does not increase monotonically with age. For values of a greater than $\frac{1}{4} t_{max}$, the median remaining lifetime begins to decrease. This property is also true of the average remaining lifetime, which is

$$E[L|a] = \frac{t_{max} - a}{ln(t_{max}) - ln(a)}$$

Figure 2 shows the median and average remaining lifetime as a function of age. The proposed model fits the observed data well for ages less than 6 – 12 hours. For longer lifetimes, the distribution of lifetimes diverges from the model. The remaining lifetimes of old jobs do not decline, as predicted, but appear to increase. It is difficult to generalize this behavior, however, since there are few jobs with lifetimes greater than 12 hours (less than 6%) and their distribution is influenced by the machine's time limit.

In any case, since our model is not accurate for jobs with lifetimes greater than 12 hours, and since the actual remaining lifetimes of these jobs are difficult to characterize, we will need to be careful to use our model only in the range where it is accurate.

4.1 Using additional information about jobs

On the Intel Paragon at SDSC, jobs are submitted to the batch partition through NQS, which requires the user to specify a queue according to the resources the job requires. Most jobs are submitted to queues that specify the cluster size of the job (rounded up to a power of two) and the length of time it will run (short, medium, or long). We will take advantage of this information by modeling a different distribution of lifetimes for each class of jobs.

In this environment, we have defined job classes according to queue name, but in another environment it might be useful to distinguish jobs by the name of the executable or the user, by requested cluster size by other declared resource requirements (*e.g.* memory size).

Figure 3 shows the distribution of lifetimes for parallel jobs submitted to the short, medium, and long queues. The differences among these distributions indicate that the queue name provides significant information about the expected

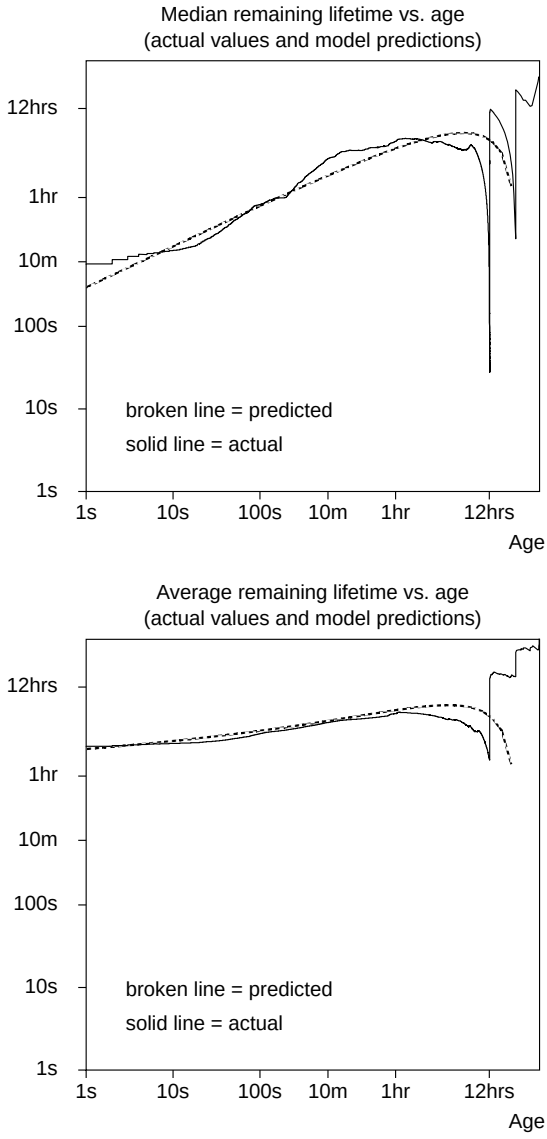


Figure 2: The median and average remaining lifetimes of a job, as a function of its current age. The broken line shows the values predicted by our model, with the estimated parameters from Table 1.

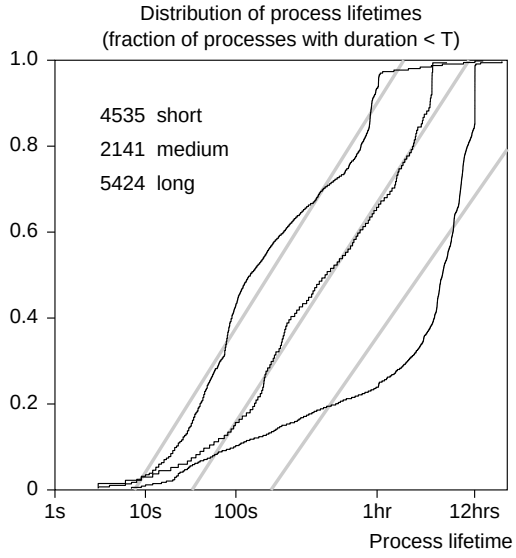


Figure 3: The distribution of lifetimes for parallel batch jobs submitted to the Intel Paragon at SDSC, broken down according to the queue to which they were submitted. Choosing a queue is one way the user communicates the resource requirements of the job to the system.

run times of jobs. This is not surprising, since users usually know the run times of their jobs, at least approximately. On the other hand, this information is far from perfect. There is considerable overlap between the different queues; in fact, many jobs are submitted to what turns out to be the wrong queue. For example, 30% of the jobs from the short queue run longer than the median lifetime in the medium queue, and 17% of the jobs from the medium queue are shorter than the median of the short queue.

Thus, it is not trivial to use queue names to predict run times. The technique we propose here, using a different conditional lifetime distribution for each queue, seems like an effective way to use information provided by the user without suffering greatly if that information turns out to be wrong.

Using the same technique as in the previous section (discarding the longest and shortest 10% of each group), we fit a line to each lifetime distribution curve. Table 2 shows the resulting coefficients. We are treating sequential jobs separately because their distribution of lifetimes is significantly different from that of the parallel jobs.

Table 2

	β_0	β_1	R^2
sequential	-.65	.21	.99
short	-.30	.15	.97
medium	-.50	.14	.99
long	-.73	.13	.74

For all but the long queue, the goodness of fit metric is quite good; for the long queue, it is only 0.74, which is not surprising, since the distribution is obviously not a straight line. Lifetimes of jobs in the long queue are more skewed than in other queues, suggesting that many users know the run times of their jobs with some accuracy and only submit them to the long queue if necessary. More than 60% of the jobs in the long queue exceed the time limit for the medium queue and therefore were required to run in the long queue.

In Section 6.5 we will present a technique to address this poor fit, but in the meantime we will ignore it and use the estimated parameters as is.

5 Calculating the distribution of queue times

Given the state of a system — the number of jobs currently running, their ages and their cluster sizes — we can use the distribution of lifetimes to estimate the queue time expected for a new arrival.

Suppose that the state of the machine at the time of an arrival is as follows: there are p processes running, with ages a_i and cluster sizes n_i (in other words, the ith process has been running on n_i processors for a_i seconds). We would like to predict the median value of $Q(n')$, the time until n' additional processors become available ($n' = n - free\ processors$).

The distribution of $Q(n')$ can be calculated exactly by considering all possible outcomes (which processes complete and which are still running) at time t. These outcomes are denoted by the bit vector b, where $b_i = 0$ indicates that the ith process is still running, and $b_i = 1$ indicates that the ith process has completed before time t.

Since we assume independence between jobs in the system, the probability of a given outcome is the product of the probabilities of each event (the completion or non-completion of a job), and the probability of these events is given by the conditional cumulative distribution of lifetimes.

$$Pr\{b\} = \prod_{i|b_i=0} cdf_{L|a_i}(t) \cdot \prod_{i|b_i=1} 1.0 - cdf_{L|a_i}(t) \tag{5}$$

For a given outcome, the number of free processors is the sum of the processors freed by each job that completes:

$$F(b) = \sum_{i|b_i=1} n_i \tag{6}$$

Thus at time t, the probability that the number of free processors is equal to the requested cluster size is the sum of the probabilities of all the outcomes that satisfy the request:

$$Pr\{F \geq n'\} = \sum_{b|F(b)\geq n'} Pr\{b\} \tag{7}$$

Finally, we find the median value of $Q(n')$ by setting $Pr\{F > n'\} = 0.5$ and solving for t. Of course, the number of possible outcomes (and thus the running time to evaluate $Q(n')$) increases exponentially with p. Thus it is not feasible to calculate this value exactly in a real system.

There are, however, two approximations to this value that are fast to compute and that turn out to be sufficiently accurate for our intended purposes. The next two sections describe these approximations.

5.1 Predictor A

When the number of additional processors required (n') is small, it is often the case that there are several jobs running in the system that will single-handedly satisfy the request when they complete. In this case, the probability that the request will be satisfied by time t is dominated by the probability that any one of these benefactors will complete before time t.

In other words, the chance that the queue time for n' processors will exceed time t is approximately equal to the probability that none of the benefactors will complete before t:

$$Pr\{F < n'\} \approx \prod_{i|n_i \geq n'} 1 - cdf_{L|a_i}(t) \tag{8}$$

This calculation is only approximately correct, since it ignores the possibility that several small jobs might complete and satisfy the request. Thus, we expect this predictor to be inaccurate when there are many small jobs running in the system, few of which can single-handedly handle the request. The next section suggests an approximation that we expect to be more accurate in this case.

5.2 Predictor B

Before addressing the most general case, we will consider a simplified model in which each job is allocated a single processor, and at the time of an arrival all running jobs have the same age. From the distribution of process lifetimes, we can find $p(t)$, the probability that a job of age a will complete before time t:

$$p(t) = cdf_{L|a}(t) \tag{9}$$

Then the probability that n processors will be free by time t is given by the binomial distribution:

$$prob\{Q(n) < t\} = \sum_{i=n}^{N} \binom{N}{i} p(t)^i (1 - p(t))^{N-1} \tag{10}$$

where N is the total number of processes in the system and n is the number of processors requested. In other words, the probability that $Q(n)$ will be less than t is the sum of the probabilities that the number of free processors at time t is greater than n. The median value of $Q(n)$ can then be found by solving $Pr\{Q(n) < t\} = 0.5$ for t.

Numerically, this is best done by replacing the summation with the following integral:

$$1 - Pr\{Q(n) < t\} =$$

$$(N - n + 1)\binom{N}{n-1} \int_0^{1-p(t)} t^{N-n}(1-t)^{n-1} dt \tag{11}$$

Then the integral can be solved by simple quadrature (*e.g.* a trapezoid method) and the root can be found quickly by Newton's method. Even so, this approach is not a feasible addition to an online scheduling algorithm. Fortunately, it can be approximated quickly by making the following observation: as the number of processors becomes large, the number of processors that are free at time t approaches $N \cdot cdf_{L|a}(t)$. This comes from the definition of the cumulative distribution.

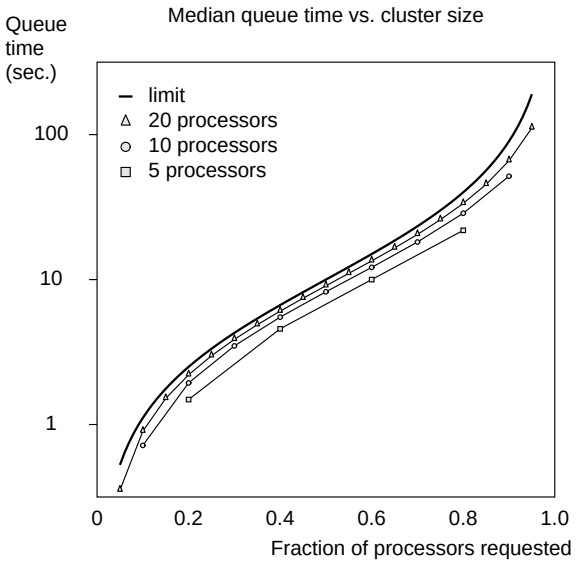


Figure 4: The three bottom lines show the median value of $Q(r)$ for $n = 5, 10$ and 20, calculated exactly using Equation 11. The top line is the approximation from Equation 12.

We can approximate the median value of $Q(n)$ by solving for t:

$$N \cdot cdf_{T|a}(t) = n \tag{12}$$

Figure 4 shows that (1) as the number of processors increases, this approximation becomes exact, and (2) even for small numbers of processors, the approximation is within a factor of $2 - 3$.

So far we have been dealing with the simplified case where all jobs are allocated a single processor and all jobs have the same age. But the approximation proposed above can be extended in a natural way to the more general case of varied cluster sizes and ages.

Given the age of a process, the conditional cumulative distribution tells us the probability that the job will complete before time t. By an abuse of the notion of probability, we can imagine that this probability indicates what fraction of a process' processors will be available at time t.

For example, a job that has been running on 10 processors for 6 minutes might have a 30% chance of completing in the next 2 minutes, releasing all ten of its processors. As a peculiar approximation of this behavior, though, we might imagine instead that the process will release 30% of its processors some time in the next two minutes.

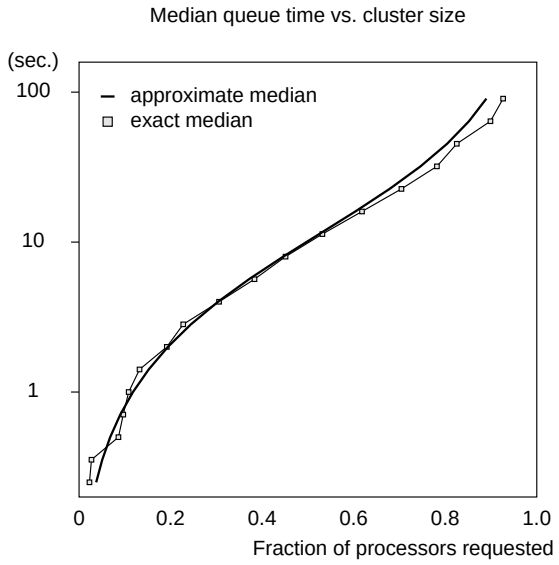


Figure 5: Given a randomly generated machine state, we calculated the exact queue times (using the combinatorial algorithm) and an approximation based on the distribution of process lifetimes.

This allows us to predict the number of free processors at time t:

$$F = \sum_i n_i \cdot cdf_{L|a_i}(t) \tag{13}$$

In order to evaluate the accuracy of this approximation, we simulated the execution of a set of processes (with random cluster sizes and durations) on a parallel computer with 64 processors. At the time of a new arrival to the system, we examined the state of the processes already running and compared the approximate median of $Q(n)$ (Equation 13) with the exact median (Equation 7) for a range of cluster sizes. Figure 5 shows a typical result; the approximation is quite accurate over a wide range of values.

In the next section we will apply the same technique to a more realistic workload, and show that the predictions based on the estimated $Q(n)$ accurately predict the behavior of the system.

6 Trace-driven simulator

We obtained traces of 24906 jobs submitted to the batch partition of the Intel Paragon at SDSC between January 1, 1995 and December 31, 1995. Using the

arrival times, cluster sizes and durations of these jobs, we simulated their execution on a parallel computer with 368 nodes (this is the size of the batch partition at SDSC). This data and our simulator are available from http://whatever.

Table 3 shows the average and median lifetimes and cluster sizes for these jobs (CV is the coefficient of variation — the ratio of the standard deviation to the mean). The total utilization of the system, averaged over the year, is 72%.

Table 3

	average	CV	25% below	median	75% below
lifetime	9020 s.	1.74	77s	720s	14500s
cluster size	25	1.47	1	16	32

The jobs in these traces are not malleable (at least not from the system's point of view). The user chooses a job's cluster size and the system cannot allocate fewer than the requested number (and does not allocate more).

Although the arrival times of the jobs are taken from the traces, the schedule used by the simulator differs from the actual schedule that was executed on the Paragon:

- The simulator uses strict first-come-first-serve scheduling. In reality, the Paragon at SDSC assigns different priorities to different queues.
- The Paragon allocates processors using a modified 2-D buddy system based on power-of-two partition sizes. In order to reduce fragmentation, small jobs are sometimes allowed to run in the interstices. In the simulator, we make no effort to allocate contiguous clusters.
- The Paragon at SDSC has some nodes with more memory than others, and some jobs can only run on these "fat" nodes. The simulator ignores this distinction.
- During the year, the size of the batch partition changed several times; in the simulator we held the number of processors constant.

Thus, in evaluating our predictors, the "actual" queue times are the queue times based on the schedule generated by the simulator, not the queue times from the trace data.

6.1 Making predictions

The approximations we presented in Section 5 consider only the jobs that are running in the system at the time of an arrival; they do not consider any jobs that might be in queue. Thus, the queue time we are dealing with is the time from when a job reaches the head of the queue until it begins execution.

In our simulations, most jobs spend no time at the head of the queue because there are enough free processors for them to begin execution immediately. Thus,

only 8545 (34%) of the simulated jobs received predicted queue times. In a system with malleable jobs, though, all jobs will make predictions when they reach the head of the queue, usually for a range cluster sizes.

6.2 Calculating Predictor A

When a job reaches the head of the queue and there are not enough free processors to satisfy its request, we calculate n', the number of additional processors required. We then consider the number of jobs in the system with cluster sizes greater than n'; that is, the jobs that will single-handedly satisfy the request if they complete.

In our simulations, the number of processors required tended to be small (50% below 16), and thus the number of potential benefactors was often large (75% of the time there are 3 or more). Table 4 shows the average and median values of n' and the number of benefactors. Under these circumstances, we expect Predictor A to perform well.

Table 4

	average	25% below	median	75% below
n'	30.6	7	16	45
benefactors	8.1	3	6	11

Of the 8545 times that a job arrived at the head of the queue, 393 times it found no processes running that could satisfy the request. Under these circumstances, Predictor A can not make a prediction.

The remaining 8152 predictions are shown in Figure 6, which is a scatter plot of predicted values on the x-axis and actual queue times on the y-axis. There is a clear correlation between the predicted and actual queue times; the coefficient of correlation between them is 0.62.

The solid line shows the median of the actual values for each column. For predicted values between 5 seconds and 3000 seconds, the median value closely follows the 45-degree line, indicating good agreement between the predicted and actual values. For predicted values greater than 3000 seconds, the predicted values tend to be higher than the actual values. This bias reflects the inherent conservative nature of the approximation: the predictor ignores the possibility of multiple processes completing in order to fulfill a request. When there are few processes that can satisfy a request single-handedly, the predicted queue times will tend to be too high.

6.3 Predictor B

When there are many small processes in the system we expect Predictor B to provide more accurate approximations of queue times. Figure 7 shows a scatter plot of these predictions versus the actual values.

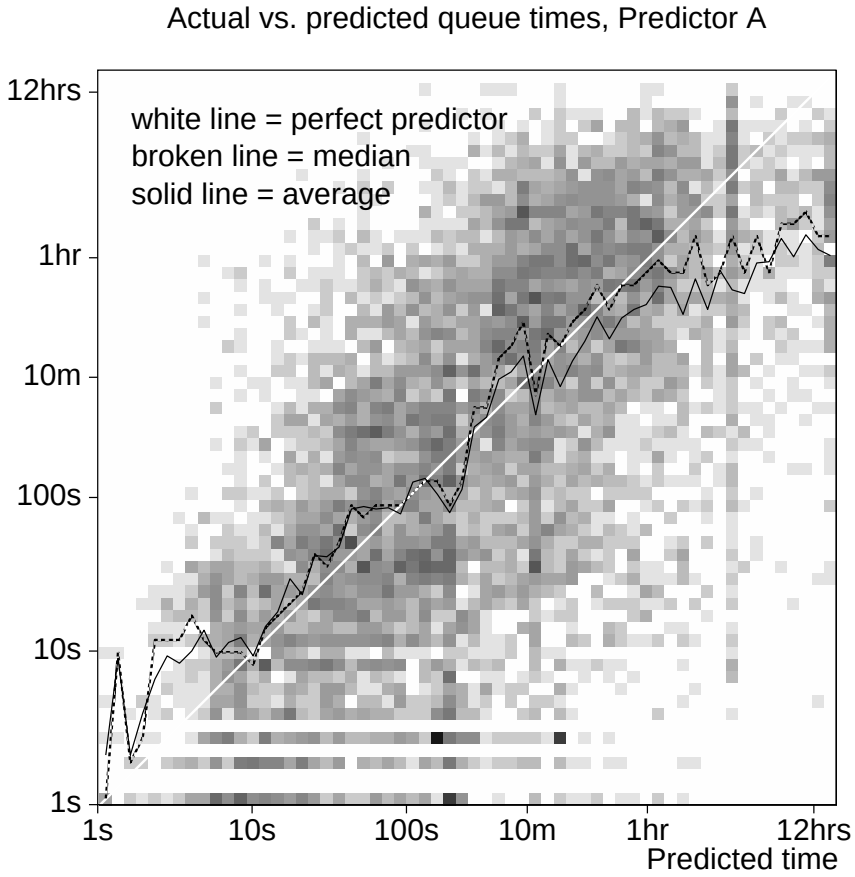


Figure 6: Scatterplot of actual queue times versus prediction calculated by Predictor A. The white line shows the identity function; i.e. a perfect predictor. The solid line shows the average value of the predictions in each column. The broken line shows the median value of the predictions in each column.

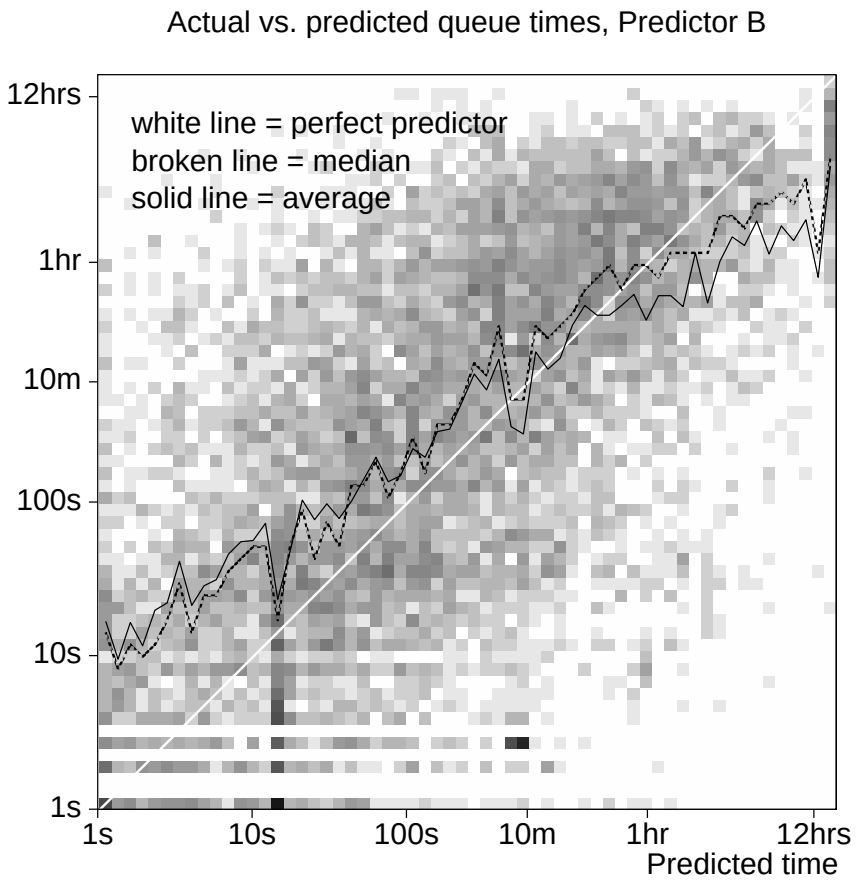


Figure 7: Scatterplot of actual queue times versus prediction calculated by Predictor B.

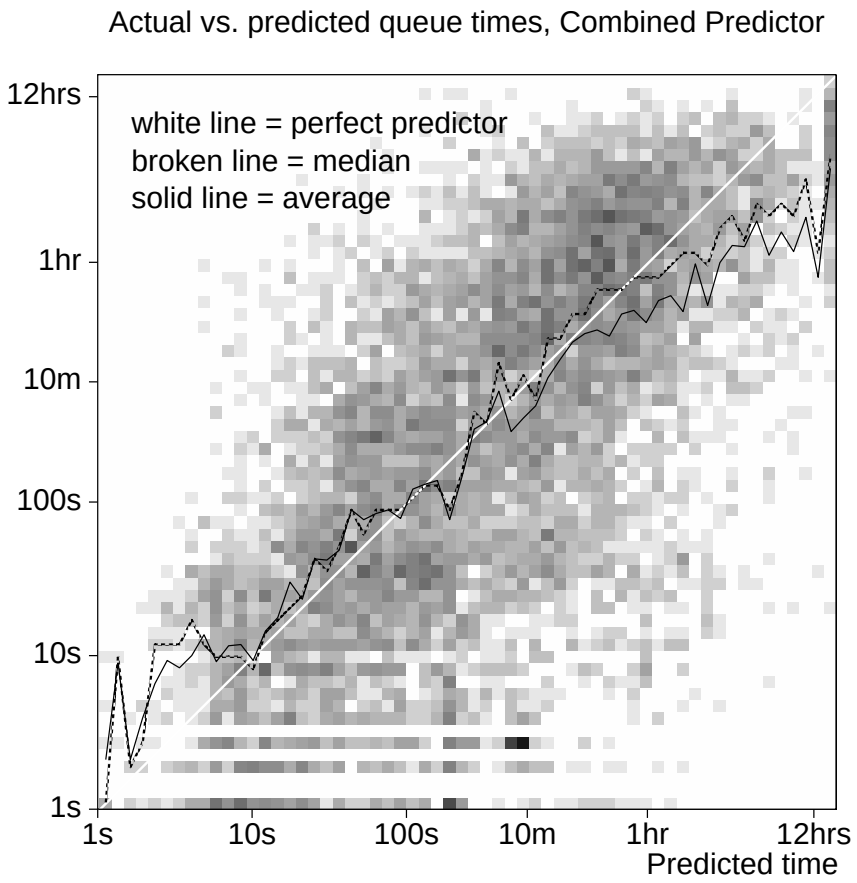


Figure 8: Scatterplot of actual queue times versus prediction calculated by the Combined Predictor.

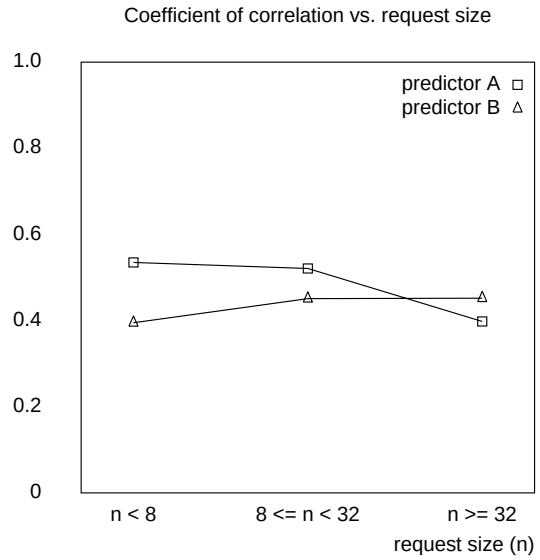


Figure 9: The accuracy of the two predictors (measured by the coefficient of correlation) varies with the size of the requested cluster. Predictor A does best for small requests; Predictor B for large requests.

In general, the correlation here is about as good as in Figure 6; the coefficient of correlation is 0.61. For short queue times, though, the predicted values tend to be too low. This reflects the optimism of the approximation: instead of having to wait for a process to complete and release all its processors, the approximation is based on the assumption that jobs are constantly giving up some of their processors. Thus, for small values of n' this predictor tends to guess too low.

6.4 Combining the predictors

Since the two predictors perform well under different circumstances, it seems natural to combine them by using each one when it is likely to be most accurate. Figure 9 shows the coefficient of correlation of each predictor for three ranges of n'. As expected, Predictor A does best for small values of n'; Predictor B does best for large values of n'.

Using Predictor A when n' is less than 32 and Predictor B when n' is greater than 32, we get a combined predictor with a coefficient of correlation of 0.65. The scatterplot for the combined predictor is shown in Figure 8.

6.5 An improved model for long jobs

In Section 4.1 we observed that the uniform-log distribution model does not fit the distribution of jobs from the long queue well. Using an alternative functional form we were able to fit the observed data with an R^2 value of 0.92 (compared to 0.74 for the uniform-log model).

Using the parameters estimated by this model and the corresponding conditional distribution of lifetimes, we found that the accuracy of the predictors increased slightly, but the difference is not significant. This result suggests that the relatively poor fit of the uniform-log model for long jobs does not significantly impair our predictors.

7 Another place, another time

Since there is no theoretical reason why the distribution of lifetimes should fit the uniform-log model we propose, it is natural to ask whether the techniques presented here are applicable to other environments.

To answer this question, we obtained data from the IBM SP2 at the Cornell Theory Center [5], including the submission times, execution times and cluster sizes for 50862 jobs submitted during the six-month interval from June 18 to December 2, 1995.

We divided these jobs into three groups — short, medium and long — according to the name of the queue to which they were submitted. Figure 11 shows the distribution of lifetimes for these groups. As with the jobs on the Paragon at SDSC, the uniform-log model fits well for the short and medium groups, and not as well for the long group. Nevertheless, the R^2 values for this curve fits are somewhat better across the board than with the Paragon data:

Table 5

	β_0	β_1	R^2
short	-.6	.23	.999
medium	-.58	.17	.995
long	-.65	.13	.84

The relationships among the three curves are not the same as they were among the jobs from the Paragon. On the Paragon the estimated lines were nearly parallel; only the intercept varied from group to group. On the SP2, the lines have nearly the same intercept; it is their slopes that vary. This observation suggests that these distributions vary from environment to environment, but that the proposed model is able to span this range of behavior.

Next, we submitted the trace data from the SP2 to the same simulator we used for the Paragon data. The only change we made to the simulator was to plug in the six estimated parameters from Table 5.

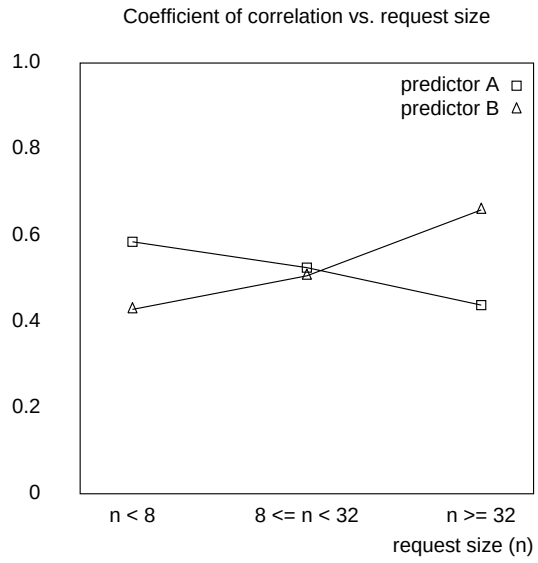


Figure 10: Accuracy of the predictors for the data from the IBM SP2 at CTC. The performance of the two predictors is somewhat better than on the Paragon. The crossover pattern is similar.

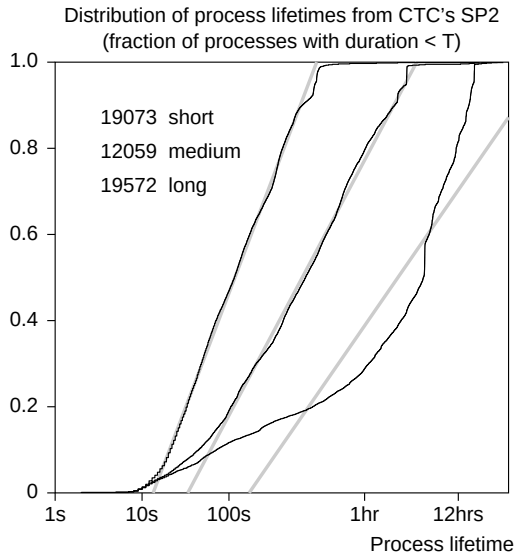


Figure 11: The distribution of lifetimes for batch jobs submitted to the IBM SP2 at CTC, broken down according to the queue to which they were submitted.

Of the 50862 jobs in the trace, 12967 received predictions from Predictor A and 13414 received predictions from Predictor B. Figure 10 shows the performance of the two predictors as a function of the number of processors requested. As before, Predictor A does well for small values of n' and Predictor B does better for large values of n'. Thus, we combined the two predictors using the same threshold as before. The resulting combined predictor had a coefficient of correlation of 0.7.

Figure 12 shows a scatterplot of the actual queue times versus the predicted values. For most values between 3 seconds and 2 hours the average value in each column lies near the predicted value; this indicates that the predictions are unbiased. Also, the coincidence of the average and mean values in each column indicates that the actual outcomes were distributed symmetrically around the predicted values. Compared with Figure 8 it is clear that the predictor is somewhat more accurate for this data than it was for the Paragon data.

One reason for this improvement is that the SP2 is a bigger machine (430 nodes vs. 368 on the Paragon), and the average cluster size of jobs tended to be smaller (9.2 vs. 25.2 on the Paragon). Thus, there tend to be more jobs in the system at any given time. Since our predictions are based on the aggregate behavior of many jobs, we expect them to improve as the number of jobs in the system increases.

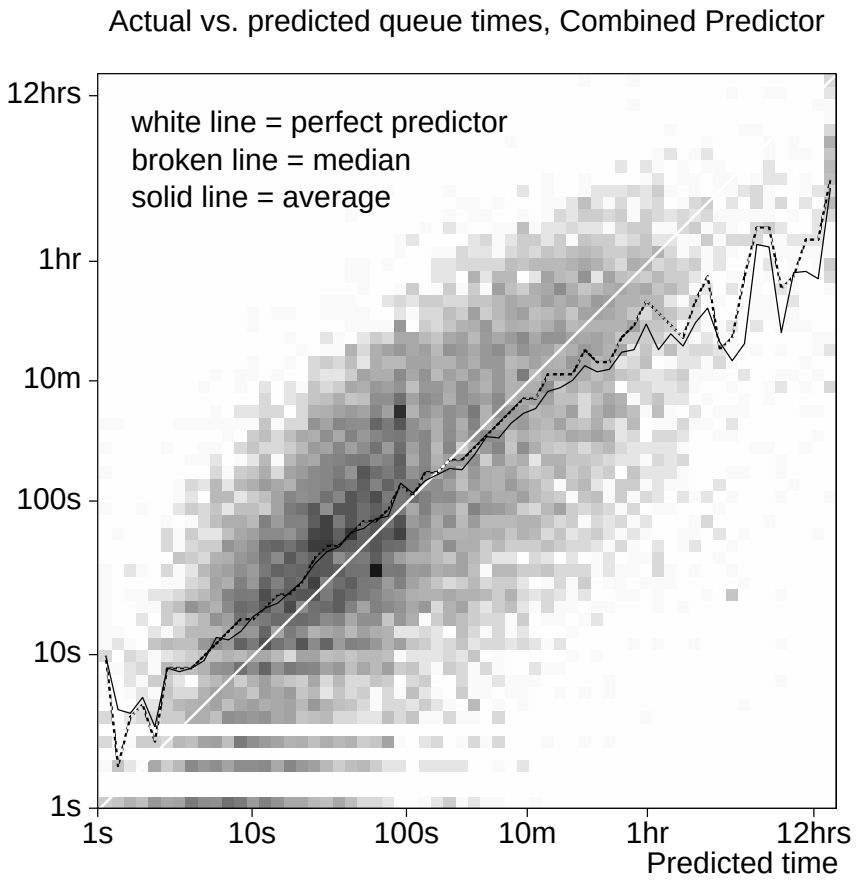


Figure 12: Scatterplot of actual queue times versus prediction calculated by the combined predictor for trace data from the IBM SP2 at the Cornell Theory Center.

7.1 Another time

In previous sections, we have cheated slightly by using the trace data to estimate the distribution parameters, and then used the parameters in the trace-driven simulation to calculate predictions. Of course, a real system would not have the luxury of knowing ahead of time the exact distribution of lifetimes of the jobs that would be submitted.

Assuming that the distributions do not change drastically over time, though, it should be possible to make accurate predictions using parameters estimated from recent accounting data. To test this hypothesis, we obtained a second trace from the Paragon at SDSC from January 1, 1996 to May 15, 1996. Without ever examining this data or calculating the distribution of lifetimes, we blindly applied the same parameters derived from the previous year of data. The results were:

Table 6

	coefficient of correlation	
	new data	old data
Predictor A	.56	.62
Predictor B	.50	.61
Combined Predictor B	.58	.65

It is clear from the drop in accuracy that the parameters of the distribution of lifetimes does vary over time and that these variations affect the ability of the system to predict queue times. One of the issues that must be addressed in order to implement a queue-prediction tool in a real system is how often to update the parameters.

Figure 13 shows the distribution of lifetimes for eight two-month intervals including the original trace data and the new data presented in this section. It is clear that this distribution is changing over time in a consistent trend (except for the first two intervals) toward longer median lifetimes and a greater proportion of long jobs. This trend may reflect a more mature workload including more production runs. Hotovy *et al.* [7] report a similar maturing process in the workload at CTC.

8 Conclusions

- We have proposed a workload model for batch jobs on space-sharing parallel computers, based on a *uniform-log distribution* of lifetimes. The proposed model captures the behavior of real workloads in two different environments. This model can be used to generate synthetic workloads for simulating and evaluating scheduling policies for similar environments.

- We have used this workload model to develop statistical techniques for predicting queue times for incoming jobs. The proposed techniques worked

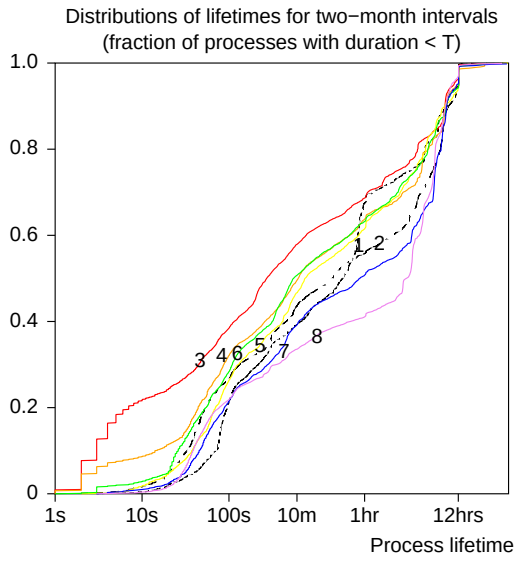


Figure 13: The distribution of lifetimes for eight two-month intervals. The first two intervals are shown with broken lines because they do not follow the pattern of the subsequent intervals.

well in simulations of both of the environments we tested. These techniques should be useful for several applications, especially selection of cluster sizes on space-sharing parallel computers.

- Many scheduling policies for supercomputing environments depend on information provided by users about the resource requirements of their jobs. We observe that this information is often unreliable, but show that the proposed techniques are able to distill this information in a useful way.

8.1 Future work

In this paper we have presented algorithms for predicting the queue time that a job expects after it reaches the head of the line. We have not addressed the question of how long it will wait between when it joins the queue and when it gets to the front. This time is likely to be less predictable, since we have less information about jobs while they are in queue than we do once they begin execution.

Nevertheless, Predictor B can be extended in a natural way to predict the behavior of jobs in queue, and thus to predict the entire queue time experienced by a new arrival. We plan to implement this extension and explore its application to *external resource selection*; that is, choosing among various computing resources available over a network.

We have suggested that our predicted queue times could be used to select a cluster size for a malleable job in order to minimize its expected turnaround time. We would like to evaluate the effect of this sort of local optimization on the performance of the system as a whole. Previous work ([9]) has shown that it is necessary, for the efficiency of space-sharing systems, for jobs to choose smaller cluster sizes when the system load is high. Allowing jobs to choose cluster sizes to minimize turnaround time will have this property, but it is not clear how system utilization under this discipline will compare with that of strategies that explicitly maximize utilization.

References

[1] M. J. Atallah, C. L. Black, D. C. Marinescu, H. J. Siegel, and T. L. Casavant. Models and algorithms for coscheduling compute-intensive tasks on a network of workstations. *Journal of Parallel and Distributed Computing*, 16(4):319–327, Dec 1992.

[2] Murthy V. Devarakonda and Ravishankar K. Iyer. Predictability of process resource usage: A measurement-based study on UNIX. *IEEE Transactions on Software Engineering*, 15(12):1579–86, December 1989.

[3] Dror G. Feitelson and Bill Nitzberg. Job characteristics of a production parallel scientific workload on the NASA Ames iPSC/860. In *Proceedings of the 9th International Parallel Processing Symposium*, pages 215–227, 1995.

[4] Mor Harchol-Balter and Allen B. Downey. Exploiting process lifetime distributions for dynamic load balancing. In *Proceedings of the ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, pages 13–24, May 1996.

[5] Steven Hotovy. Personal correspondence. 1996.

[6] Steven Hotovy. Workload evolution on the Cornell Theory Center IBM SP2. In *Proceedings of the 10th International Parallel Processing Symposium*, pages 15–22, 1996.

[7] Steven Hotovy, David Schneider, and Timothy O'Donnell. Analysis of the early workload on the Cornell Theory Center IBM SP2. Technical Report 96TR234, Cornell Theory Center, January 1996.

[8] W. E. Leland and T. J. Ott. Load-balancing heuristics and process behavior. In *Proceedings of Performance and ACM Sigmetrics*, volume 14, pages 54–69, 1986.

[9] K. C. Sevcik. Application scheduling and processor allocation in multiprogrammed parallel processing systems. *Performance Evaluation*, 19:107–140, 1994.

[10] Ken Steube and Reagan Moore. Paragon throughput analysis. In *Proceedings of the Intel Supercomputer Users Group*, pages 264–267, June 1994.

[11] Michael Wan, Reagan Moore, George Kremenek, and Ken Steube. A batch scheduler for the Intel Paragon with a non-contiguous node allocation algorithm. In *Proceedings of the 10th International Parallel Processing Symposium*, pages 29–40, 1996.

[12] Kurt Windisch, Virginia Lo, Dror Feitelson, Bill Nitzberg, and Reagan Moore. A comparison of workload traces from two production parallel machines. In *6th Symposium on the Frontiers of Massively Parallel Computation*, 1996.