The Workload on Parallel Supercomputers: Modeling the Characteristics of Rigid Jobs

Uri Lublin* Dror G. Feitelson
School of Computer Science and Engineering
The Hebrew University, 91904 Jerusalem, Israel

Abstract

The analysis of workloads is important for understanding how systems are used. In addition, workload models are needed as input for the evaluation of new system designs, and for the comparison of system designs. This is especially important in costly large-scale parallel systems. Luckily, workload data is available in the form of accounting logs.

Using such logs from three different sites, we analyze and model the job-level workloads with an emphasis on those aspects that are universal to all sites. As many distributions turn out to span a large range, we typically first apply a logarithmic transformation to the data, and then fit it to a novel hyper-Gamma distribution or one of its special cases. This is a generalization of distributions proposed previously, and leads to good goodness-of-fit scores. The parameters for the distribution are found using the iterative EM algorithm.

The results of the analysis have been codified in a modeling program that creates a synthetic workload based on the results of the analysis.

1 Introduction

The study and design of computer systems requires good models of the workload to which these systems are subjected. Until recently, the data necessary to build these models — observations from production installations — were not available, especially for parallel computers. Instead, most models were based on assumptions and mathematical attributes that facilitate analysis. Recently a number of supercomputer sites have made accounting data available that make it possible to build realistic workload models. It is not clear, however, how to generalize from specific observations to an abstract model of the workload. This paper addresses the analysis of real workloads and the creation of realistic workload models based on this analysis; special emphasis is placed on the tradeoff between the complexity of the model and the resulting predictive power. The use of such models will lead to substantial improvements in experimental procedures, and to higher confidence in their results.

*Current affiliation: SANgate Systems

The scope of this paper involves rigid parallel jobs. The reason for focusing on parallel jobs is that they lead to intricate workloads, where each job's parallelism and runtime may interact; thus a more extensive modeling effort is needed than in traffic models in which packets have a predefined size, and only the arrival process is not known. "Rigid" jobs are jobs that do not change their parallelism at runtime [8]. We chose not to address malleable or dynamic jobs at this stage due to the complexities involved, and the lack of data, as described below.

1.1 Why Model?

There are two common ways to use a recorded workload to analyze or evaluate a system design: (1) use the logged workload directly to drive a simulation, or (2) create a model from the log and use the model for either analysis or simulation. For example, trace-driven simulations based on large address traces are often used to evaluate cache designs [13, 12]. But models of how applications traverse their address space have also been proposed, and provide interesting insights into program behavior [24, 25].

The advantage of using a trace directly is that it is the most "real" test of the system; the workload reflects a real workload precisely, with all its complexities, even if they are not known to the person performing the analysis.

The drawback is that the trace reflects a specific workload, and there is always the question of whether the results generalize to other systems or load conditions. In particular, there are cases where the workload depends on the system configuration, and therefore a given workload is not necessarily representative of workloads on systems with other configurations. Obviously, this makes the comparison of different configurations problematic. In addition, traces are often misleading if we have incomplete information about the circumstances when they were collected. For example, workload traces often contain intervals when the machine was down or part of it was dedicated to a specific project, but this information may not be available.

Workload models have a number of advantages over traces.

- The modeler has full knowledge of workload characteristics. For example, it is easy to know which workload parameters are correlated with each other because this information is part of the model.
- It is possible to change model parameters one at a time, in order to investigate the influence of each one, while keeping other parameters constant. It is also possible to select model parameters that are expected to match the specific workload at a given site. Such manipulations are problematic when using logs. For example, it is common practice to increase the modeled load on a system by reducing the average interarrival time. But this practice has the undesirable consequence of shrinking the daily load cycle as well. With a workload model, we can control the load independent of the daily cycle. Another example is the ability to create long workloads with more jobs than are available in any given log.
- A model is not affected by policies and constraints that are particular to the site where a trace was recorded. For example, if a site configures its NQS queues with a

maximum allowed duration of 4 hours, it forces users to break long jobs into multiple short jobs. Thus, the observed distribution of durations will be different from the "natural" distribution users would have generated under a different policy.

- Logs may be polluted by bogus data. For example, a trace may include records of jobs that were killed because they exceeded their resource bounds. Such jobs impose a transient load on the system, and influence the arrival process. However, they may be replicated a number of times before completing successfully, and only the successful run represents "real" work. In a model, such jobs can be avoided (but they can also be modeled explicitly if so desired).
- Finally, modeling increases our understanding, and can lead to new designs based on this understanding. For example, identifying the repetitive nature of job submittal can be used for learning about job requirements from history. One can design a resource management policy that is parameterized by a workload model, and use measured values for the local workload to tune the policy.

1.2 How to Model

The main problem with models, as with logs, is that of representativeness. That is, to what degree does the model represent the workload that the system will encounter in practice? The answer depends in part on the degree of detail that is included. Each job is actually composed of procedures that are built of instructions, and these interact with the computer at different levels. One option is to model these levels explicitly, creating a hierarchy of interlocked models for the different levels [2, 1]. This has the obvious advantage of conveying a full and detailed picture of the structure of the workload. For example, the sizes[1] of a sequence of jobs need not be modeled independently. Rather, they can be derived from a lower-level model of the jobs' structures [7]. Hence the combined model will be useful both for evaluating systems in which jobs are executed on predefined partitions, and for evaluating systems in which the partition size is defined at runtime to reflect the current load and the specific requirements of jobs.

The drawback of this approach is that as more detailed levels are added, the complexity of the model increases. This is detrimental for two reasons. First, more detailed traces are needed in order to create the lower levels of the model. Second, it is commonly the case that there is wider diversity at lower levels. For example, there may be many jobs that use 32 nodes, but at a finer detail, some of them are coded as data parallel with serial and parallel phases, whereas others are written with MPI in an SPMD style. Creating a representative model that captures this diversity is hard, and possibly arbitrary decisions regarding the relative weight of the various options have to be made. We will therefore concentrate on *single-level* models, which may be viewed as the top level in a hierarchical set of models [7].

The most common approach used in workload modeling is to create a statistical summary of an observed workload. It is typically assumed that the longer the observation period, the

[1] We will use "size", and also "large" and "small", to denote the degree of parallelism of a job. "Duration", "short", and "long" denote runtimes.

better. Thus we can summarize a whole year's workload by analyzing a record of all the jobs that ran on a given system during this year.

The simplest and most widely used statistics rely on moments, especially the mean and the variance of the sample data. For example, these statistics indicate that the distribution of job runtimes has a wide dispersion, leading to a preference for a hyperexponential model over an exponential one. However, such summaries may be misleading, because they may not represent the shape of the distribution correctly. Indeed, more than 20 years ago Lazowska showed that models based on a hyperexponential distribution with matching moments can lead to incorrect results [15]. In addition, the calculation of moments can be unduly influenced by extreme values, that are not necessarily representative [4]. Thus, we prefer the calculation of percentiles and an attempt to match the CDF of the target distribution.

In Section 3, we present our techniques for modeling the CDF of a distribution. Our chosen model is the hyper-Gamma distribution, which is a combination of two Gamma distributions. The parameters of these distributions are discovered using the EM algorithm. We then apply this methodology to the characterization of the distributions of parallelism, runtimes, and interarrival times in different workload logs. By comparing the logs, we identify features and parameter values that are more representative than others, and include them in the final workload model.

2 Workload Logs

The analysis presented here is based on workload logs from 3 locations. Each log contains tens of thousands of jobs, and spans many months of activity. The logs are typically available as ASCII files in which each job that ran on the system is represented by a single line, with several space-separated fields providing data about different job attributes. These logs and others are available on-line from the Parallel Workloads Archive [19].

The first log is that from the San-Diego Supercomputer Center Intel Paragon machine. This machine has 416 nodes, of which 352 constitute the batch partition and 48 are in the interactive partition. Scheduling is based on NQS, with special handling of partitions and node configurations [26]. The log spans all of 1995 and 1996, with about two thirds of the jobs in the first year; the total is 113515 successful jobs. This log was originally available are two separate logs for the two years, and therefore sometimes appears so in our analysis (as SDSC95 and SDSC96).

The second log is from the 1024-node Connection Machine CM-5 installed at Los-Alamos National Lab (LANL). This machine is scheduled using DJM, with a minimal partition size of 32 nodes. The log spans the period from January through September 1996, and contains 36306 jobs that completed successfully (the full log is longer, but we only used the most recent part).

The third and final log is from the 100-node IBM SP2 machine installed at the Swedish Royal Institute of Technology in Stockholm (KTH). This machine is scheduled using the EASY backfilling scheduler [16]. The log contains 16221 jobs that completed successfully, in the period of October 1996 to August 1997.

3 Statistical Techniques

This section covers statistical distributions and techniques that we used to find the workload model. The idea in statistical modeling is to characterize the workload using distributions, such that the workload is consistent with sampling from these distributions. This is done in three phases:

1. Decide which distribution to use in the model.
2. Find the values of that distribution's parameters, such that it would provide a good fit to the samples.
3. Check the goodness of fit of that specific distribution to the sample data.

3.1 Candidate Distributions

In the first phase we typically considered the distributions described below.

Two-phase-uniform distribution: This is a generalization of the uniform distribution. Four parameters are required in order to define it: l (low), m (medium), h (high), and p (proportion). The cumulative distribution function is then constructed as two straight lines, passing through three points: $(l, 0)$, (m, p), and $(h, 1)$.

Exponential distribution: This well-known distribution is often used in modeling, due to the availability of good estimators, and its nice mathematical properties (e.g. being memoryless). Its pdf is $f(x;\theta) = \frac{1}{\theta} e^{-\frac{x}{\theta}}$.

Hyper-exponential distribution: A mixture of two (or more) exponential distributions. The two-stage version, which is a mixture of two exponentials, has 3 parameters: θ_1, θ_2 and p. θ_1 and θ_2 are the parameters of the two exponential distributions, and p is the proportion of the first one. Thus $0 \leq p \leq 1$ of the population comes from an exponential distribution with parameter θ_1, and $1 - p$ from a distribution with parameter θ_2.

Gamma distribution: This distribution is defined by the pdf

$$f_X(x;\alpha,\beta) = \frac{1}{\Gamma(\alpha)\beta^\alpha} x^{\alpha-1} e^{-\frac{x}{\beta}}$$

where $x, \alpha, \beta > 0$ and

$$\Gamma(\alpha) = \int_0^\infty t^{\alpha-1} e^{-t}\, dt$$

This is also a member of the exponential family of distributions, and therefore has good estimators.

The gamma distribution is much more general and "flexible" than the exponential distribution (which is actually a special case). Several examples of the effect of its parameters are shown in Fig. 1. The α is the shape parameter. As can be seen, when $\alpha > 1$ the distribution is bell-shaped; when $\alpha \leq 1$ it has a right tail. The smaller α,

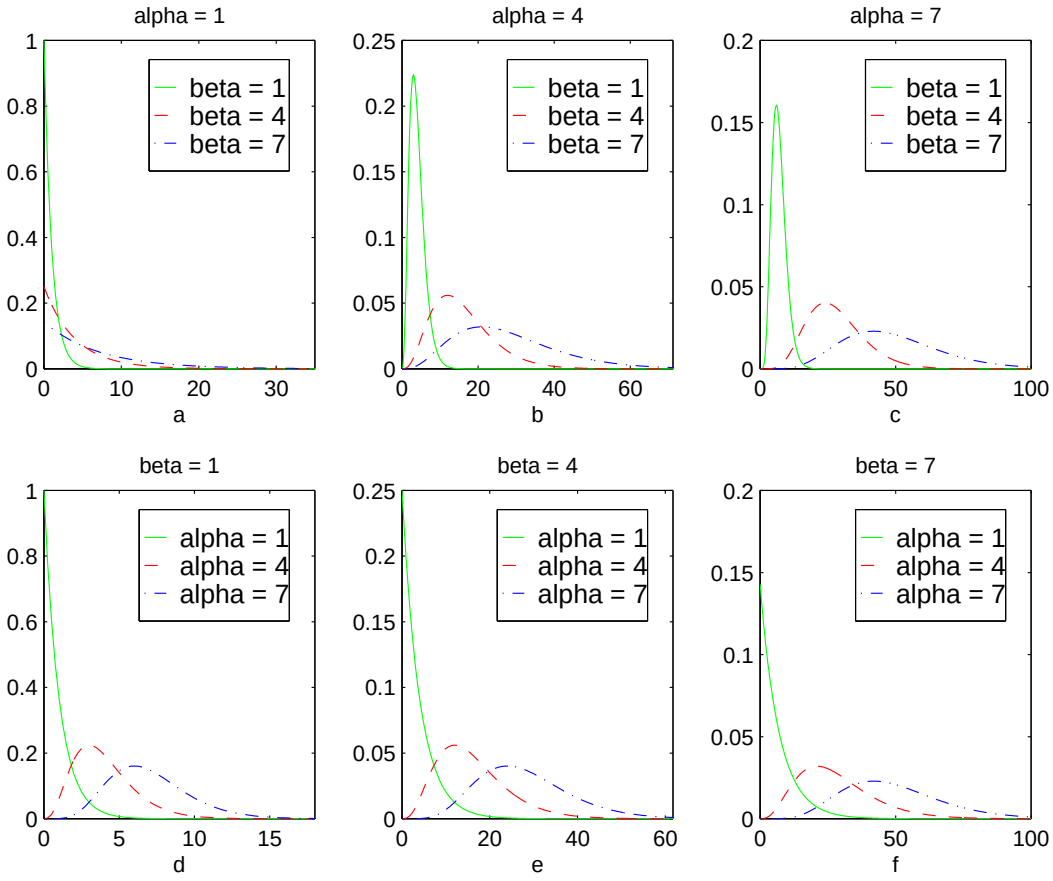

Figure 1: *Examples of Gamma distributions.*

the heavier the tail. The β is called the scale parameter, as it determines the spread of the distribution (see the x scale in the bottom 3 graphs). The mean of the distribution is equal to the product of its parameters: $m = \alpha\beta$. The variance is $\alpha\beta^2$.

As noted, several important distributions are special cases of the Gamma distribution. The exponential distribution is obtained by setting $\alpha = 1$. The Erlang distribution is obtained by setting α to be an integer. In both cases, β is the parameter of the resulting distribution ($\beta = \theta$).

Note: Sometimes the gamma distribution is represented by α, β' parameters where $\beta' = 1/\beta$. The mean is then $\frac{\alpha}{\beta'}$, and the variance $\frac{\alpha}{\beta'^2}$. We will use this notation as it is the one employed by Matlab, which we used for our analysis.

Hyper-Gamma distribution: A mixture of two (or more) gamma distributions, and therefore a generalization of the hyper-exponential and hyper-erlang distributions. The version that we will use, which is a mixture of two Gamma distributions, has 5 parameters: α_1, β_1, α_2, $\beta 2$, and p.

3.2 Fitting a Distribution

Given a data sample and the functional form of a certain distribution, we need to find "good" values of the distribution's parameters. These values will define exactly one distribution; for example if the given function is of the form $f(x; a, b) = ax + b$, we need to find the values of a and b. Parameter values are judged by the degree to which the resulting distribution matches the sample data. There are several methods to find parameter values.

3.2.1 Moments Estimation Method

One way to characterize a distribution is by its moments (mean, variance, etc.), which can typically be expressed as a function of its parameters (e.g. the mean of a Gamma distribution is the product of its parameters α and β). By inverting these functions, one obtains a formula that expresses the parameters as a function of the moments. The moments of the sample can then be calculated and plugged into the formula, giving the desired parameters.

The problem with this method is that it does not deal well with distributions (or samples) that have fat tails. The number of equations needed is the number of unknown parameters of the distribution. Thus if the number of parameters is large, high-order moments are required. These are very sensitive to sample outliers, leading to situations in which most of the data is effectively ignored. For example, the exponential distribution has only one parameter, and this can be estimated based on only the first moment (the mean). But the hyper-Gamma distribution has five, so the fifth moment is also required. Consider a sample with one outlier, e.g. 1,1,1,1,2,2,2,3,3,4,5,15. The fifth moment, defined as $M_5 = \frac{1}{5}\sum_{i=1}^{n} x_i^5$, is 760995; 99.8% of this is due to the outlier, $15^5 = 759375$. The resulting parameters are therefore dominated by rare samples, that are not necessarily very representative.

3.2.2 Maximum Likelihood Estimation Method

The idea behind maximum likelihood is to derive those parameter values that would lead to the highest probability of sampling the given data values. Since log is a monotonic function, the maximum of the likelihood function is the maximum of the log likelihood function. Often it's easier to calculate the maximum of the log likelihood function, since log makes a product into a sum and an exponent into a product.

For example, consider the exponential distribution. The likelihood of sampling values $X_1, \ldots, X_n$, given a parameter θ, is the product of the likelihoods of the individual values:

$$L(X_1, \ldots, X_n; \theta) = \prod_{i=1}^{n} f_X(X_i; \theta)$$

Taking a log and developing this equation leads to

$$\begin{aligned} \ln(L(X_1, \ldots, X_n; \theta)) &= \ln\left(\prod_{i=1}^{n} \frac{1}{\theta} e^{-X_i/\theta}\right) \\ &= \sum_{i=1}^{n} \left(\ln(1/\theta) - \frac{X_i}{\theta}\right) \\ &= n \ln(1/\theta) - \frac{1}{\theta}\sum_{i=1}^{n} X_i \end{aligned}$$

To find the θ that maximizes this expression we differentiate and equate to zero, leading to

$$\frac{n}{1/\theta} = \sum_{i=1}^{n} X_i$$

and the solution of θ equal to the mean of the sample. This procedure is based on the assumption that the random variables of the sample are independent.

3.2.3 Finding Parameters for a Mixture

In a mixture, each sample comes from one (and only one) of the distributions forming the mixture. But we don't know which one, making it harder to assess the parameters of the individual distributions. The missing data is sidestepped by using the iterative EM (Expectation Maximization) algorithm. This algorithm is based on the assumption that the number of distributions in the mixture and the functional form of each distribution are given, and that for each distribution estimating the parameters is not hard; it usually produces near-optimal results. The Algorithm proceeds as follows [18]:

1. Initialize the parameters of the distributions somehow.
2. Expectation-Step: For each observation and for each distribution decide what part of this observation "belongs to" this distribution. Since currently the parameters of the distribution are set, we can find for each distribution the probability of getting the observation from that specific distribution. This probability is the "relative part" of the observed value that is assigned to this distribution.
3. Maximization-Step: For each distribution estimate the parameters using the maximum likelihood estimation method. This estimation is done according the "part of observations" that "belong to" this distribution.
4. Repeat Expectation-Maximization steps until the sample likelihood converges.

In our case the number of distributions is two, and the functional form of each is given (whether it is Gamma or exponential). Thus both steps (2) and (3) are easy to calculate.

Given that EM is a heuristic iterative algorithm, its results and convergence depend on how it is initialized. We developed the Medium Minimum Estimation Method for this purpose. This method is based on the fact that often it is easy to distinguish between the two distributions in the mixture just by looking at the histogram of the samples, because it is bimodal (or becomes bimodal after applying a logarithmic transformation). We can thus simply divide the sample at the minimum point between the two modes, and estimate the parameters for each subsample. Likewise, the estimate for the probability of belonging to each distribution is initialized according to the fraction of the samples that fall in each subsample.

3.3 Assessing Goodness of Fit

After finding the parameters and thus defining a specific distribution, we want to know how well it actually models the original sample data. In other words, we want to ensure that

the best model we managed to find is indeed a good model in absolute terms. We used the Kolmogorov-Smirnov goodness of fit test for this purpose [14]. For the distributions we are considering, this is known to be a conservative test.

3.3.1 The Kolmogorov-Smirnov Method

The Kolmogorov-Smirnov calculates the *maximal* distance between the cumulative distribution function of the theoretical distribution and the sample's empirical distribution, over all the points.

1. Sort the sample's observations: $X_1 \dots X_n$ such that $X_1 \leq X_2 \dots \leq X_n$.
2. Define the sample's empirical cumulative distribution function:

$$F_n(x) = \begin{cases} 0 & \text{if } x < X_1 \\ i/n & \text{if } X_i \leq x < X_{i+1} \\ 1 & \text{if } x \geq X_n \end{cases}$$

   The empirical CDF $F_n(x)$ is an estimator for the theoretical distribution $F(x)$. Furthermore, if the samples are indeed from the theoretical distribution $F(x)$, then $\Pr(\lim_{n\to\infty} |F(x) - F_n(x)| = 0) = 1$.
3. Define $D_n = \sup_x \{|F(x) - F_n(x)|\}$. Since $F_n(x)$ is a step function with all steps of height $1/n$, this is equivalent to

$$D_n = \max_{i=1\dots n} \left\{ \left| \frac{i}{n} - F(X_i) \right|, \left| F(X_i) - \frac{i-1}{n} \right| \right\}$$

4. If the value of D_n is large we will reject the hypothesis that the theoretical and empirical distributions are the same. If D_n is small enough we will not reject that hypothesis. The actual threshold to use depends on the sample size n and the confidence level α wanted, and can be found in statistical tables.

3.3.2 The χ^2 Method

When F(x) is not available in the form of a simple equation, the alternative is to create random samples from this distribution, and then check whether it is reasonable to say that the original data and these samples come from the same distribution. This is done using the χ^2 test. The number of samples generated should equal the number of original data observations.

In this test, the range of values is partitioned into a certain number k of subranges. Then the number of observations that fall into each range (O_i) and the expected number of observations that *should* fall into each range (E_i) are tabulated (in our case, this is not based on the formula but rather on the created random samples). The metric is then

$$\chi^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i}$$

This is then compared with statistics tables to ascertain if it is small enough to warrant the conclusion that the distributions are probably the same.

4 The Degree of Parallelism

Rigid jobs can be thought of as rectangles in processors-time space. The most important aspects of modeling them are therefore modeling their dimensions in terms of processors, as we do here, and in terms of runtime, as we do in the next section. Direct modeling of the degree of parallelism is possible because this is an input parameter provided by the user: a request to submit a job for execution is always accompanied by a specification of how many processors to use. When considering moldable jobs, it is necessary to model the total work and the speedup function, in order to derive the runtime for any allocated number of processors.

4.1 Handling Power-of-Two Jobs

Experience shows that the distribution of job sizes is typically dominated by sizes that are powers of two [4]. This is obvious for early machines that only allowed powers of two, such as hypercubes and connection machines. However, it persists also in systems that have no architectural preference for powers of two. This raises the question of what is the source of this phenomenon, and whether it should be included in the workload model.

Jann et al. [11] and Downey [3] chose not to include any special treatment for powers of two in their models. This is based on the belief that the emphasis on powers of two is not intrinsic to the workload. Instead, it is thought to stem from habit and from influence of queue configuration choices, where system administrators define queues for different job sizes that are typically delimited by power-of-two nodes.

We disagree on this point, and believe that jobs with sizes that are powers of two should be emphasized, as is observed in the logs. One argument for this approach is that if using powers of two is habitual, then it is indeed an intrinsic part of the workload — this is what users do today, and can be expected to do in the future as well. Another argument is that we continue to see a strong use of powers of two even in logs that come from machines that do not have queues configured according to powers of two, e.g. machines running the EASY or Maui schedulers. Finally, we note that power-of-two jobs have been shown to have a strong impact on performance evaluation results [17], so departing from empirical observations should be justified carefully; retaining the observed characteristics is safer.

4.2 The Modeling Procedure

Apart from power-of-two jobs, we also note the high fraction of serial jobs that is typically observed. This motivates us to partition the jobs into three classes: serial, power-of-two, and the rest. Our modeling procedure will be based on a combination of this distinction and a general distribution over all parallel jobs.

Our procedure is shown graphically in Fig. 2. With probability p_1, the job is selected to be serial. If not, the log of its size is selected from an appropriate distribution. Then, with probability p_2, this value is rounded to the nearest integer so as to represent a power-of-two job (note that p_2 is not the fraction of power-of-two jobs out of the total, but only out of the

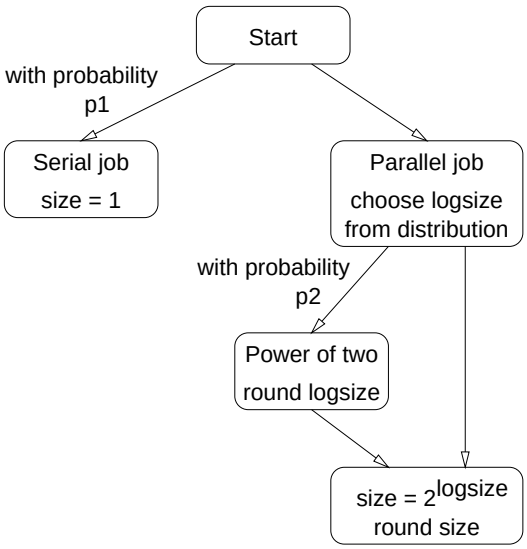

Figure 2: *Algorithm for modeling the size of a job.*

			Two-stage uniform					Gamma		
	p_1	p_2	l	m	h	p	D_n	α	β	D_n
SDSC95	0.21	0.76	0.75	5.00	8.64	0.92	0.040	4.04	0.77	0.064
SDSC96	0.21	0.89	1.00	5.00	7.00	0.86	0.038	4.92	0.69	0.052
KTH	0.30	0.60	0.70	4.00	6.64	0.84	0.033	3.87	0.73	0.058
LANL	–	–	4.50	7.00	10.0	0.82	0.083	24.95	0.25	0.114
Model	0.24	0.75	min−0.2	max−2.5	max	0.86				

Table 1: *Parameter values for logs and model of job sizes.*

parallel jobs). The job size is then 2 raised to the selected value and rounded if necessary (in case its not a power of 2).

4.3 The Distribution and Parameters

To complete the model, we must specify the parameters p_1 and p_2, and the distribution.

For the evaluation of the two parameters we exclude the LANL-CM5 log, as this machine only allows partitions that are powers of two with at least 32 nodes. The values for the other logs are shown in Table 1. They lead to the choice of $p_1 = 0.24$ and $p_2 = 0.75$ for the model.

To find a fitting distribution, we first apply a logarithmic transformation, and regard the result as a continuous distribution. We checked fits to a Gamma distribution, a uniform distribution, a two-stage uniform distribution, and a hyper-exponential distribution. For each one parameters were estimated and the fit evaluated using the Kolmogorov-Smirnov method. The best results were obtained for the Gamma and two-stage uniform distributions, and they are shown in Table 1. We chose the two-stage uniform distribution for the model for two reasons: first, its fit is slightly better. Second, the parameter values obtained were more consistent across the different machines, and it was relatively straightforward to relate

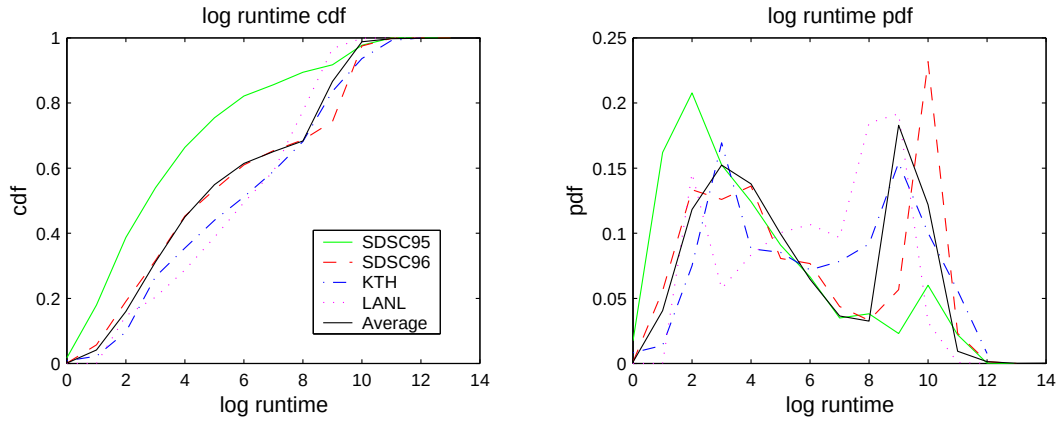

Figure 3: *Distributions of runtimes after a logarithmic transformation, and the derived model for these distributions.*

them to the machine size. Thus in the model we define l to be 0.2 less than the minimal possible value (to correct for the fact that when using a uniform distribution and rounding, the probability to get the end value is reduced), m to be 2.5 less than the maximal value, and h to be the maximal possible value, which is $\log_2 P$. p is set to the arithmetic mean of 0.86.

5 Job Runtimes

As noted above, runtime is the second major characteristic of a rigid job. The distribution of runtimes, however, is very different from the distribution of sizes: It is continuous rather than being discrete, and spans a much larger range of values. The modeling technique is therefore completely different. We start by describing the modeling of a single system, and then show how the models for the different systems were combined into a single representative model.

5.1 Modeling Individual Systems

As in the case of job sizes, we start by applying a logarithmic transformation to the data. This reduces its range, reduces the influence of extreme values that may not be representative in general, and does not affect the quality of the results.

The candidate distributions checked were the Gamma distribution, the hyper-exponential distribution, and the hyper-Gamma distribution (which is actually a generalization of the previous two, and also of the hyper-Erlang distribution). The hyper-Gamma distribution is especially appealing because the distributions seem to be bimodal in log space, as shown in Fig. 3.

The parameters of the hyper-Gamma distribution were estimated using the middle-minimum method as a starting point for the application of the EM algorithm, as explained in Section 3. The goodness of fit was evaluated using the Kolmogorov-Smirnov method. The results are shown in Table 2. In most cases, the hyper-Gamma model is very close to the original data and provides a better match than other distributions. However, it does not

	α_1	β_1	α_2	β_2	p	D_n
SDSC95	2.35	1.50	1533	0.007	0.94	0.037
SDSC96	4.15	0.96	569.7	0.018	0.70	0.035
KTH	5.76	0.82	159.5	0.05	0.61	0.081
LANL	6.00	0.66	68.5	0.13	0.55	0.034
Average model	4.20	0.94	312.0	0.03	0.685	
Search model	4.00	1.00	500.0	0.02	0.70	

Table 2: *Parameter values for model of job runtimes.*

pass the statistical goodness-of-fit test. The reasons for this are that our samples are not necessarily independent, and that we are dealing with extremely large numbers of samples by statistical standards: 15000 to 70000 jobs in each log [14]. In order to check this conjecture, we chose a random subsample of between 500 and 1000 jobs from each data set, and checked the fit of the models to these subsamples. In all cases, the models passed the test at the 95% confidence level.

5.2 Creating a Representative Model

In order to create a representative model, we need to reduce the four models shown in Table 2 into one. We considered two ways of doing so.

The first method calculates an average value for each of the five parameters of the hyper-Gamma distribution, based on the values this parameter has in the four models. Given that the salient features of the Gamma distribution depend on products of these parameters, the geometrical mean was chosen (but the arithmetic mean was used for p). The resulting parameter values are shown in Table 2, and the resulting distribution is compared with the originals in Fig. 3. As can be seen, its shape is a plausible representative for the data.

The second method is to tabulate the range of values of each parameter, and systematically check different combinations. Specifically, for each parameter we selected three values which divide the range into equal parts, and checked the distributions resulting from all 243 combinations of these values. Most of the distributions turned out not to have the desired characteristics (bimodal with maxima and minima in the right places) and were eliminated. Of the remaining ones, the one whose pdf lay closest to the middle of the pdf's of the data sets was chosen for the model. The resulting parameter values are shown in the bottom row of Table 2. They are very close to the values calculated using the previous method, leading to greater belief that these values are indeed representative.

Given that the model is a compromise representing different systems, it is also possible to consider minor modifications in the interest of efficient computability. For example, it is possible to round the αs to the nearest integer, modify the β to compensate for the change in the mean, and derive a hyper-Erlang model in place of the hyper-Gamma model (note, however, that the modeling phase was still done using the more accurate hyper-Gamma distribution). From our experience, this cuts about 12% off the time required to compute samples from the model.

	α_1	β_1	α_2	β_2	p	D_n
SDSC95	3.66	1.32	13.4	0.87	0.88	0.033
SDSC96	5.12	1.09	120.0	0.10	0.71	0.026
KTH	118.4	0.06	42.2	0.28	0.30	0.050
LANL	5.16	1.10	30.6	0.34	0.59	0.019
Average model	10.74	0.55	37.96	0.37	0.577	
Search model	5.00	1.10	45.00	0.30	0.700	

Table 3: *Parameter values for model of total job work.*

5.3 Modeling Total Work

The techniques described above can also be applied to the total work (the area of the job's rectangle, that is the number of processors multiplied by the runtime). The results of doing so are shown in Table 3. Obviously, this should be combined with a model of the speedup function of parallel jobs, like those proposed by Sevcik [20] or Downey [3]. However, the validity of this model is debatable, as it is based on jobs that ran on different numbers of processors. This is akin to an assumption of ideal speedup. Nevertheless, no direct data is available, so models such as this one must be used.

6 Correlation Between Parallelism and Runtime

The idea that parallelism can be used to reduce the completion time of a task dates back at least to Biblical times. In terms of the workload on parallel supercomputers, this may be expected to lead to an inverse correlation between parallelism and runtime: large jobs use more processors, so they should complete faster. But this is based on the implicit assumption that the total amount of work remains the same. Alternative scalability models assert that the added parallelism is not used to solve the same problem faster, but rather to solve larger problems [9, 22]. This may even lead to an increase in the total processing time.

6.1 Establishing the Existence of a Correlation

The simplest way to check for a correlation between the parallelism and runtime is to calculate the correlation coefficient between these variables. Doing so for our logs yields a small positive correlation (ranging from 0.02 for KTH to 0.40 for SDSC96). An alternative is to divide the jobs in each log into groups according to their degree of parallelism, and plot the CDF of the runtimes distribution for each such group (Fig. 4). The fact that larger jobs generally have a longer runtime is then evident from the fact that their CDF is below that of smaller jobs. This is clearly seen in the SDSC logs. In the KTH and LANL logs there is some mixture among the small jobs (in the first two groups), but not with the large jobs in the third group.

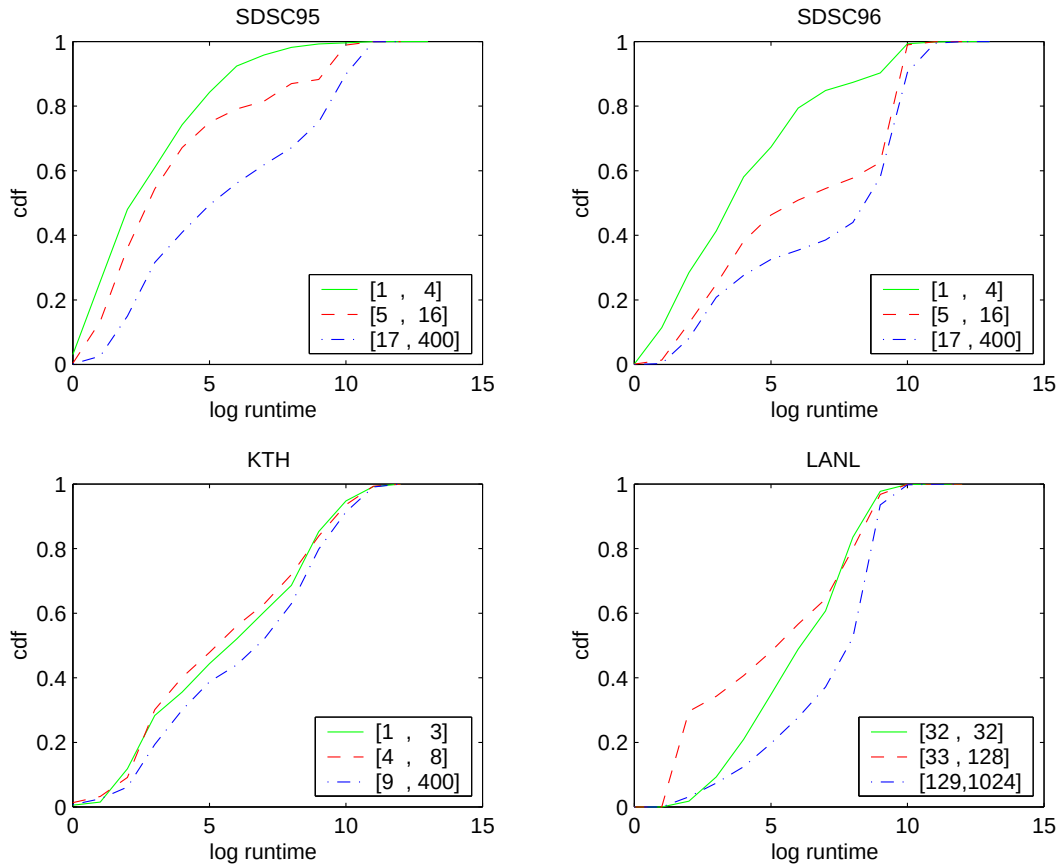

Figure 4: *Distributions of runtimes for different job sizes in the four systems.*

6.2 Modeling Approaches

Given that there is a connection between the parallelism and the runtime, the question is how to incorporate it into the workload model. A straightforward approach, adopted by Jann at al., is to divide the jobs into groups according to the degree of parallelism (as we did above), and create a distinct runtime model for each group. In the Jann model, runtimes are modeled using a hyper-Erlang distribution with 4 parameters. Given that the model deals with jobs using between 1 and 322 processors, dividing the jobs into groups according to powers of two yields 10 groups, for a total of 40 parameter values. The statistics of the resulting model are indeed very similar to those of the original data (a log from the SP2 machine at CTC). However, there is a risk of overfitting to this data, at the price of not resembling other logs.

An alternative approach is to focus on how the parameters of the runtime distribution change with job size. In other words, instead of having parameter a assigned the values $a_1, a_2, \ldots a_k$ for jobs in groups $1 \ldots k$ respectively, we can assign a the value $f(s)$ where s is the individual job's size. The function f can be completely arbitrary, although we might endeavor to find a simple function that still gives us a good fit to the data.

Our suggested approach is in this vein. We start by modeling the runtimes in each size-based group, using the hyper-Gamma distribution used in the previous section (Fig. 5 and

	sizes	α_1	β_1	α_2	β_2	p	D_n
SDSC95	1–4	3.73	1.05	0.19	4.77	0.97	0.038
	5–16	9.68	0.56	161.0	0.074	0.82	0.034
	17–400	12.63	0.62	197.1	0.07	0.66	0.063
SDSC96	1–8	5.46	0.92	230.5	0.05	0.87	0.033
	9–16	9.87	0.66	1577	0.008	0.40	0.062
	17–320	11.65	0.71	378.4	0.037	0.46	0.065
KTH	1–4	5.36	0.89	39.6	0.232	0.61	0.045
	5–100	8.68	0.84	89.97	0.133	0.58	0.016
LANL	32	29.96	0.29	330.6	0.036	0.63	0.021
	64	38.33	0.25	560.6	0.023	0.45	0.039
	128	6190	0.001	35.11	0.331	0.47	0.140
	256–1024	31.82	0.37	773.3	0.019	0.47	0.052

Table 4: *Parameters of runtime model for different size groups.*

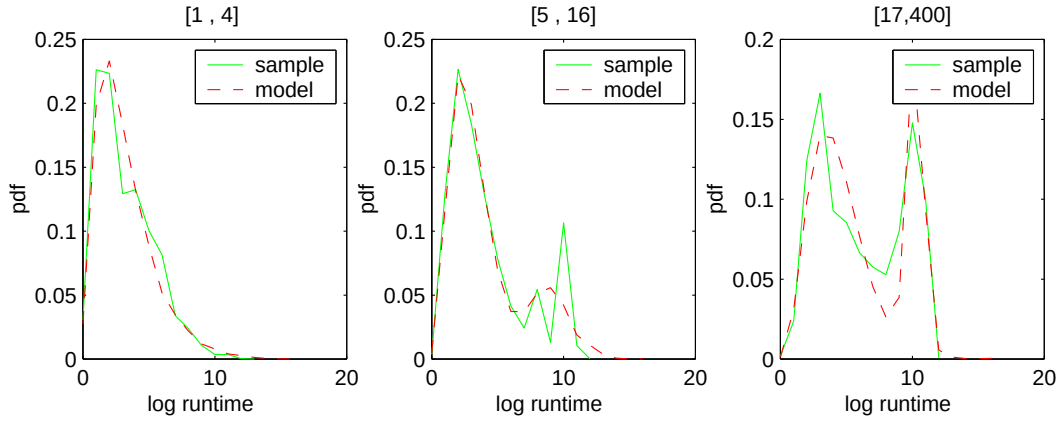

Figure 5: *Comparison of runtime model with original distributions for different size groups in the SDSC95 log.*

Table 4). This yields a good fit as measured by the Kolmogorov-Smirnov method. It also leads to the observation that the shapes of these distributions change in a similar manner: They are all composed of a pair of Gamma distributions with similar means, but the relative weights of these two distributions changes according to the range of sizes. For small jobs, the distribution of runtimes is well modeled by the lower Gamma alone. For middle-size jobs, the low Gamma is dominant, but the higher Gamma is unmistakable. For large jobs, the runtime distribution is composed of the two Gammas with roughly equal weights.

Based on this observation, we devised a model in which runtimes are represented by a hyper-Gamma distribution. Four of the five parameters, those describing the two constituent Gamma distributions, are fixed. Only the p parameter, which determines the probability of sampling either of the Gamma distributions, is correlated with the job size. In effect, it is replaced by a pair of parameters that specify how it depends on the job size. Thus the total number of parameters used by the model is 6. The procedure for sampling a runtime from

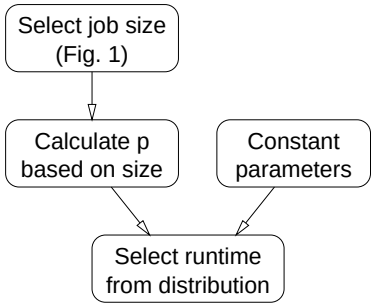

Figure 6: *Algorithm for modeling the runtime of a job.*

	a	b	D_n
SDSC95	−0.0043	0.995	0.038
SDSC96	−0.0107	0.900	0.056
KTH	−0.0065	0.649	0.058
LANL	−0.00027	0.575	0.089
Model	−0.0054	0.780	

Table 5: *Parameter values for linear dependence of p on job size.*

this model is pictured in Fig. 6.

Results obtained by this method are shown in Fig. 7. While the model cannot track the exact differences in the shapes of the distributions for different sizes, it is much more accurate than using the same model for all sizes.

6.3 Calculating p

Our model for runtimes involves two fixed Gamma distributions, and a parameter p which represents the proportion of the first one in the population. The point is that p is a function of the job size. The simplest approach is to use a linear relationship: $p = a \cdot s + b$, where s is the job size. Such a relationship was previously employed by Feitelson, but in the context of a hyper-exponential distribution [5].

Given that p should decrease as s increases, a will be negative. There is therefore a danger that p might become negative for large s. Also, b might be larger than 1 again forcing p out of its useful range. One way to solve this problem is to use a log odds transformation, based on the equation $\log\left(\frac{p}{1-p}\right) = a \cdot s + b$. This ensures that p remains between 0 and 1, but changes the shape of its dependence on s. We checked it and found that it does not fit our needs for expressing the dependence of p on s, as the resulting distributions did not fit the data. We therefore use an alternative in which p is set to 0 if it turns out to be negative, and to 1 if it is higher than 1. In effect, this divides the spectrum of job sizes into three segments. For $s \geq -\frac{b}{a}$ we get $p = 0$ and use only the first Gamma. For $s \leq \frac{1-b}{a}$ we get $p = 1$ and use only the second Gamma. In between we have a linear relationship.

To actually find suitable values for a and b, we divided each data set into 8 roughly equal groups of jobs (according to job size, not splitting any single size among two groups; for

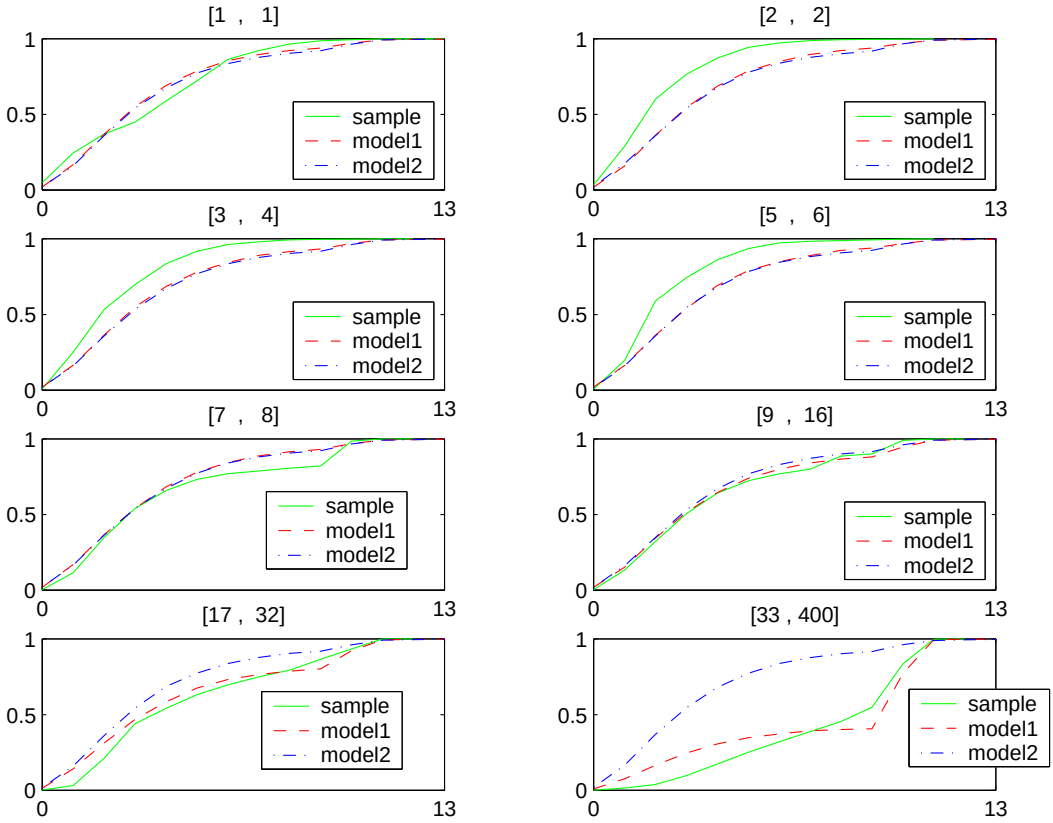

Figure 7: *Resulting runtime distribution for different job sizes. Model1 (linear dependence of p) is compared with Model2 (p is constant) for the SDSC95 data (sample).*

LANL there are only 6 sizes). For each group we found the best value of p for the two given Gamma distributions. This value of p was associated with the median number of processors used by jobs in the group. Thus we obtained 8 $\langle s, p \rangle$ pairs, which were used as an input to a linear regression that yielded a and b. The results for the 4 logs are shown in Table 5. The final model was created as the arithmetic average of the four logs.

7 The Arrival Process

The arrival process has received relatively little attention in the past, with many modelers content with using a Poisson process. In particular, such modeling eliminates the daily cycle of many arrivals during the day and less during the night. This is unfortunate, as the distribution of runtimes includes very long jobs, which may interact with this daily cycle. In addition, it has been shown that the details of the arrival process have an impact on the results of evaluations [21]. An exception is Jann et al., who created a detailed model of interarrival times for different job size ranges, on par with their model of runtimes [11].

Modeling the arrival process is made difficult by the need to fit both the overall distribution of interarrival times, and the distribution across hours of the day (the daily cycle).

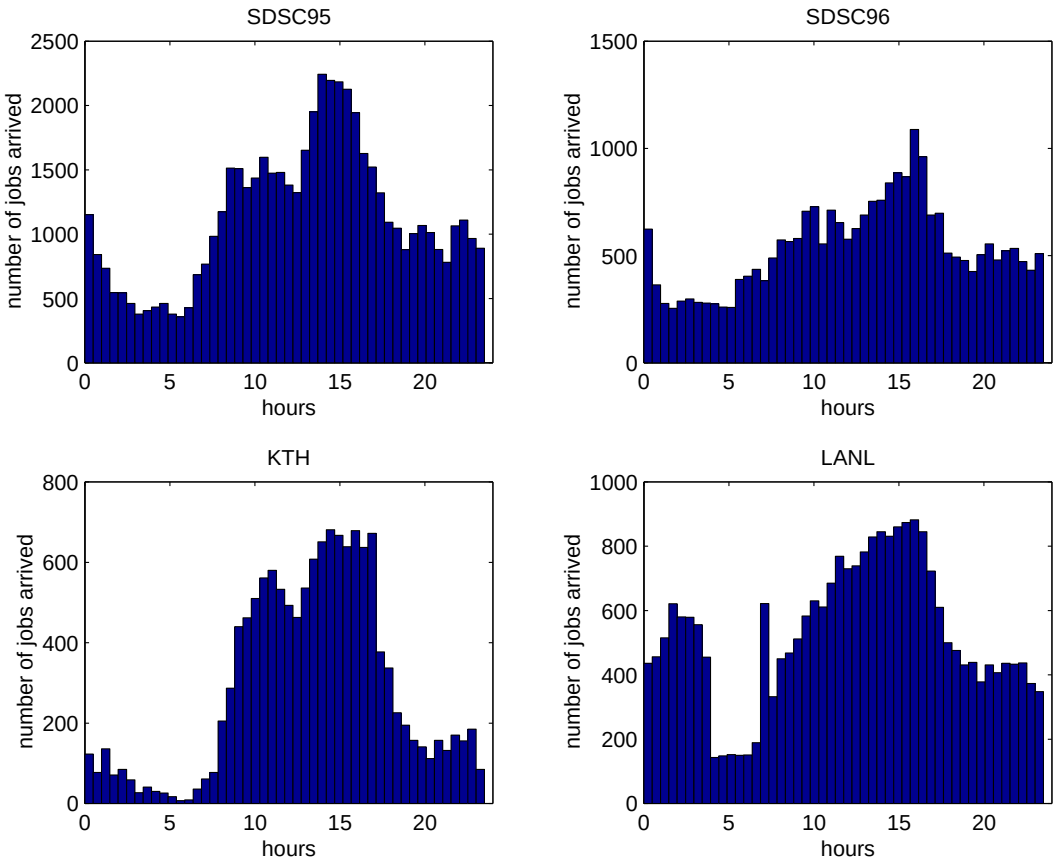

Figure 8: *Histograms of job arrivals per time of day.*

Our approach is to first model the relatively stationary arrival process at peak hours, then characterize the daily cycle, and finally combine the two.

7.1 Modeling Arrivals at Peak Hours

We start by modeling the arrival process at peak hours. This can be used in isolation, or combined with a daily model as described below. Based on the histograms of job arrivals during the day (Fig. 8) we define the peak hours to be from 8:00 AM to 7:00 PM, Mondays through Fridays. The interarrival times of jobs submitted during these times were tabulated, and compared to the following distributions: exponential, hyperexponential, Gamma, and hyper-Gamma. The goodness of fit was evaluated using the Kolmogorov-Smirnov method.

The results were that all but the exponential distribution gave good fits. The hyper-Gamma distribution was marginally better than the Gamma distributions, at the expense of using more parameters. We therefore chose the Gamma distribution for the model. The parameter values found are shown in Table 6. The model is the geometrical mean of the four systems.

	α	β	D_n
SDSC95	6.89	0.62	0.053
SDSC96	7.13	0.70	0.072
KTH	14.10	0.37	0.064
LANL	15.81	0.35	0.084
Model	10.23	0.49	

Table 6: *Parameter values for model of interarrival times at peak hours.*

	α	β	D_n
SDSC95	7.77	4.16	0.047
SDSC96	6.84	4.77	0.050
KTH	12.00	2.55	0.054
LANL	7.00	4.86	0.056
Model	8.17	3.96	

Table 7: *Parameter values for model of daily cycle.*

7.2 Modeling the Daily Cycle

Workloads can be expected to have cycles at three levels: daily (people work less at night), weekly (people work less on the weekend), and yearly (people work less on holidays). Of these, the most important effect is the daily cycle, although the weekly cycle may also have an effect on the scheduling of long jobs. We shall only deal with the daily cycle.

The daily cycle on the different systems is shown in Fig. 8. This is after removal of a large peak that appears at 3:30AM in the SDSC logs, probably due to the automatic invocation of a routine administrative script.

To model the daily cycle, we first note that the minimum is around 5 AM. We therefore perform a cyclic shift of the range 00:00–04:59 to 24:00–28:59, leading to a distribution that is essentially unimodal (ignoring the slight dip at lunchtime). This is then modeled using a Gamma distribution, whose parameters are shown in Table 7.

Note that these parameters depend on the cyclic shift of five hours, and on the use of 48 30-minute slots for the histogram. Thus to calculate the fraction of jobs that arrive in a half-hour interval around time $t \in [0, 24]$ we first multiply t by 2 (change hours to half-hour slots), add 48 if the result is less than 10 (to account for the 5 hour shift), and finally evaluate $w(t') = F(t' + \frac{1}{2}) - F(t' - \frac{1}{2})$ (where F is the CDF of the model Gamma distribution).

7.3 A Combined Model

Our goal is to derive a workload model with interarrival times that satisfy two distinct constraints: all interarrivals taken together should match the correct distribution of interarrival times, and at the same time the fraction of jobs arriving at each hour of the day should match the daily cycle. The way we go about achieving this is to use the distribution of interarrivals at peak hours as the basis, and modify it in order to achieve the correct number

of jobs for each hour of the day. In order to get the correct overall arrival rate, we modify the peak distribution by multiplying its mean by a constant we call ARAR (Arrive Rush-to-All Ratio). This is simply defined as the ratio of the mean of the complete distribution to the mean of the peak distribution, and was rather close to 1 for all four logs (which is partly explained by the fact that the distributions are after a logarithmic transformation).

One approach to achieving our goal is to modify the parameters of the interarrival time distribution according to the time of day, in order to match the desired arrival rate for that time. In order to reduce the complexity of the model, we strive to find a functional relationship of the parameters on the desired arrival rate (similar to what we did in the context of the correlation between the parallelism and runtime). Another approach is to keep the underlying distribution as is, but modify the interpretation of sampled interarrival times according to the time of day. We will see examples of both types below.

To check the quality of each approach, we need to compare its results with the original data. Assessing the fit of the interarrival distribution is done using the Kolmogorov-Smirnov method, as for all other distributions. Assessing the fit to the daily cycle is done using the χ^2 metric. In our case this is calculated as

$$P = \sum_{i=1}^{48} \frac{(m_i - d_i)^2}{m_i + d_i}$$

where m_i and d_i are the number of observations in the model and data, respectively, in each of the 48 half-hour slots of the day.

The expected interarrival time method

The first method tried is to modify the interarrival time distribution according to the desired arrival rate for a certain time slot. To do so we pre-compute different values of the α parameter of the Gamma distribution for each slot (denoted by $\alpha(t)$). Given these values, computing the next interarrival time after a job that arrived at time t will be done by e^x where x is sampled from a Gamma distribution with parameters $\alpha(t)$ and β (recall that the model is in log space).

The $\alpha(t)$ are computed as follows. Let $\bar{w}$ be the average over all slots of the values $w(t)$ representing the fraction of the jobs that arrive at time t. The ratio $\bar{w}/w(t)$ gives the ratio of the average number of jobs that arrive at any time divided by the expected number of jobs that will arrive at time t. This is the inverse of the ratio of the interarrival times. Thus

$$\frac{\bar{w}}{w(t)} = \frac{e^{\alpha(t)\beta}}{e^{\alpha\beta}}$$

Which leads to

$$\alpha(t) = \alpha + \frac{\log\left(\bar{w}/w(t)\right)}{\beta}$$

While this method is very simple, it led to a poor match with the daily cycle. The reason is that the number of jobs that end up arriving in each slot does not depend only on the distribution of interarrival times in that slot, but also on the *previous* slots. Thus using this method resulted in a distribution that is flatter than desired, and is also shifted towards later times.

The expected time slots method

The second method tries to amend the first one, by calculating $\alpha(t)$ values that take the $w(t)$ values of subsequent slots into account. This is done iteratively, starting from $\alpha(t) = \alpha$, until the values converge.

In each iteration, the calculation proceeds as follows. For each slot t, we calculate the expected jump $E(t)$ to the slot in which the next arrival will occur, but taking into account the fraction of jobs that should arrive in that slot. Thus if the arrival rate in the target slot is high we increase the probability of this jump, whereas if this probability is low we decrease it:

$$E(t) = \frac{\sum_{j=t}^{t+m-1} w(j \bmod m)(j-t)\Pr(j-t)}{\sum_{j=t}^{t+m-1} w(j \bmod m)\Pr(j-t)}$$

where $m = 48$ is the number of slots, and $\Pr(k)$ is the probability that sampling the interarrival distribution would lead to a value k slots away. Our goal is to adjust the distribution of interarrival times so that this holds. To do so we compute the weighted average of such jumps according to the desired $E(t)$ and according to the current iteration's values of the interarrival distribution parameters, and adjust the parameters to make then closer. Specifically, we calculate the new parameter value by

$$\alpha_{i+1}(t) = \frac{E(t)}{\sum_{j=1}^{m} w(j)E(j)} \sum_{j=1}^{m} w(j)\alpha_i(j)$$

where the subscript on α denotes the iteration number. This can be understood as follows: for each time t, the new value of α is related to the weighted average of αs in the same way that this time's E is related to the weighted average of all Es. As the mean interarrival time is proportional to α, this will make the distribution of αs more similar to the desired distribution of Es.

This method produced better results, but in the end we chose to use the simpler third method.

The slot weight method

In this method interarrival times are chosen from the original distribution which is not modified. However, the slots are given different weights so that "time passes faster" in slots where more jobs are supposed to arrive.

The idea is simple: each slot t is considered as having $1800\frac{w(t)}{\bar{w}}$ virtual seconds, instead of the original 1800 real seconds (half an hour). When calculating job arrivals, we sample the Gamma distribution of peak interarrival times, but regard the result as virtual rather than real seconds. Thus the next arrival has a larger chance to be trapped in a slot with a high $w(t)$. Naturally, care must be taken to account for progress within each slot. For example, assume a certain slot's duration is 1000 virtual seconds, and a value of 600 seconds was sampled. In this case, the next arrival will be 0.6 of the way into this slot. If the next sample is 500 seconds, 400 of them are used to complete the current slot, and 100 to progress into the next one.

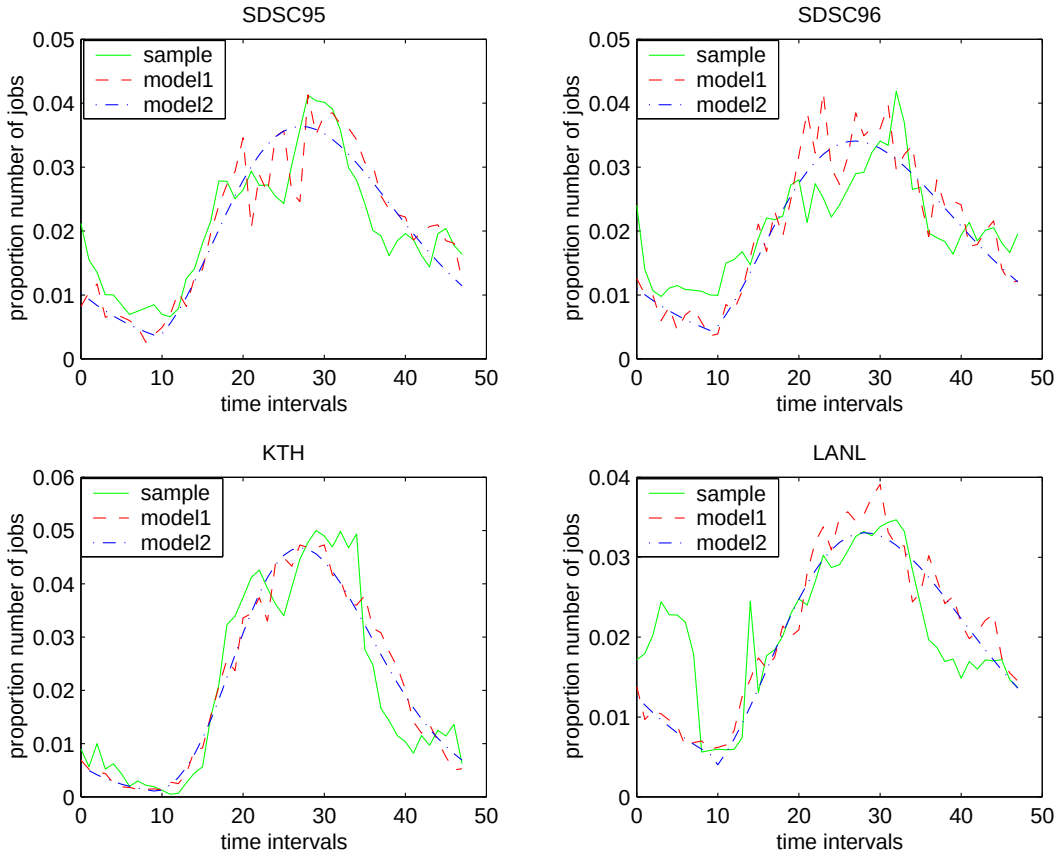

Figure 9: *Comparison of the daily arrival cycle using the final model (model1) with a smooth Gamma model (model2) and with the original data (sample).*

As noted above, this method was selected for use in the model. Its performance is shown in Fig. 9. Remarkably, just two parameters for the daily cycle Gamma suffice to create the 48 values of $w(t)$ being used, and provide a decent fit.

8 Workload Consistency

An important question when creating workload models based on logs is whether the data in the logs is consistent to begin with. This has two facets. One is the distinction between different job classes, e.g. between interactive and batch jobs. The other is the possibility that the workload evolves over time.

8.1 Distinction between Interactive and Batch Jobs

The SDSC95 and SDSC96 logs include a distinction between interactive jobs, which are submitted directly to the machine's scheduler, and batch jobs that are handled by the NQS batch queueing system [26]. This enables us to repeat all the analysis of the previous sections on each subset of jobs separately.

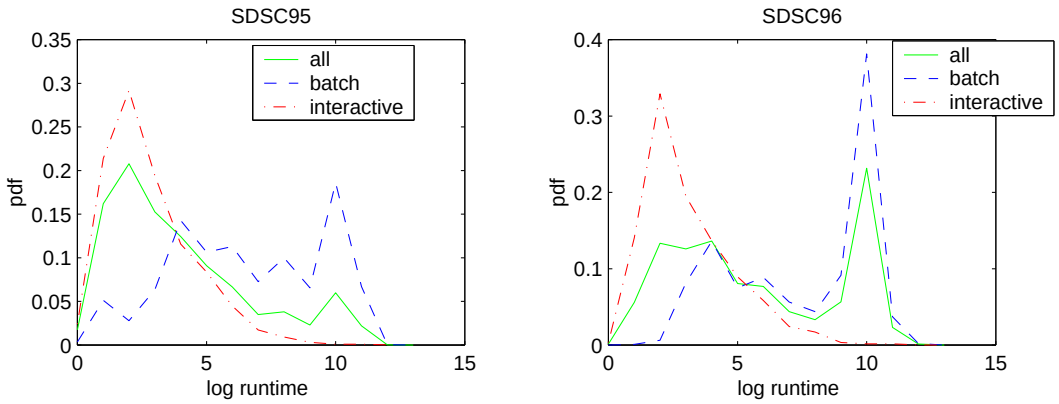

Figure 10: *Comparison of the runtime distribution of interactive and batch jobs in the SDSC logs.*

The results are that significant differences can be seen between the two job classes. Starting with the distribution of job sizes, we find that batch jobs used up to 256 nodes, whereas interactive jobs rarely used more than 32 (although this could also be a result of system administration policies). In general batch jobs were larger, but, somewhat surprisingly, there were about twice as many batch serial jobs as interactive serial jobs. Batch jobs also use slightly more power-of-two sizes, at least in the 1996 log.

Comparing the runtimes, we find that batch jobs in general ran longer. In fact, the runtimes of interactive jobs are dominated by the lower Gamma of the hyper-Gamma distribution, whereas the runtimes of the batch jobs are decidedly bimodal (in log space; see Fig. 10). It may be conjectured that the bimodality of runtimes in other logs is also a result of different job classes.

Considering the correlation between size and runtime, the correlation coefficient calculated for batch jobs is significantly higher than for interactive jobs (0.24 and 0.35 for SDSC95 and SDSC96, respectively, vs. 0.02 and 0.06). The parameters for the dependence of p on s in the model are also different. For batch jobs the absolute value of a is smaller, but this is because the range of values of s is actually much larger.

The arrival process for the two job classes is also somewhat different. The distributions of interarrival times at peak hours are similar, but the curve for batch jobs is shifted slightly towards higher values relative to the curve for interactive jobs. The daily cycle is also similar, but the differences between the daytime highs and the nighttime lows are smaller for batch jobs.

These results imply that it may be important to model interactive and batch jobs separately, especially if the model will be used to evaluate a scheduler that may provide different levels of service to different job types.

8.2 Workload Evolution

The analysis of long logs assumes that the characteristics of the workload are stable. This is in fact known not to be so: the workload changes when users learn to use a new machine

[10], and again when they migrate to the next machine once the current one becomes dated. However, it is hard to assess these processes from the logs.

To check the degree to which workloads change we compared the SDSC95 and SDSC96 logs — two consecutive years from the same machine. The results are that the two logs are similar, but certainly do have significant differences. The main change is the reduction in the number of interactive jobs in the 1996 log: from 55208 jobs to 17807 jobs (the number of batch jobs remained about the same, a bit more than 20000). As the characteristics of interactive and batch jobs are quite different, this led to noticeable changes in the overall workload: on average, in 1996, jobs used more processors and ran longer, but arrived at larger intervals.

9 Summary and Conclusions

9.1 Summary of the Model

The final model is somewhat involved, and includes the following components:

- A model of job sizes based on a two-stage uniform distribution with 4 parameters, including the two specifying the minimal and maximal sizes desired. Two additional parameters account for the fractions of serial and power-of-two jobs.
- A model of job runtimes based on the hyper-Gamma distribution. This has 6 rather than 5 parameters, because the p parameter is replaced by a linear model of how p depends on the job size.
- A model of arrivals based on two Gamma distributions: one for the peak interarrival times, and the other for the daily arrivals cycle. Two additional parameters are the shift used to model the daily cycle, and the ARAR value to match the peak arrivals to the overall arrivals.

All the above is replicated three times: once for the full workload, and again for interactive and batch jobs separately.

A program that implements this model is available on-line at Parallel Workloads Archive [19].

9.2 Comparison with Other Models

Several other models of parallel workloads have been proposed in the literature.

Probably the first detailed model, which is also a precursor of the current model, was proposed by Feitelson in 1996 [5]. This model used a two-stage hyperexponential distribution for the runtimes, choosing the parameters so that the CDF "looked right" (that is, similar to that in various logs). A subsequent version used a three-stage hyperexponential [6]. To accommodate the slight correlation observed between runtime and the degree of parallelism, the probability of using each exponential depended on the degree of parallelism. The arrival process was not modeled, and was assumed to be a Poisson process.

Jann et al. used a hyper-Erlang distribution for both the runtimes and for the interarrival times, with parameters based on matching the first three moments of the data in the CTC-SP2 log [11]. In addition, they divided the jobs submitted to a parallel machine according to their degree of parallelism, and created a separate model for each range of degrees of parallelism. However, this ignores the extra weight of powers of two that appears in the original log. The result was a model with a large number of parameters (about 90) that closely mimics the original data.

Downey has proposed the log-uniform distribution for total work based on observation of the SDSC-Paragon log [3]. This uses the smallest number of parameters, unless multiple segments are used. Unlike the other distributions, it has an upper bound on the values it might produce. Given the total work, a model of the speedup function is used to derive the runtime based on the number of processors allocated. Thus this model is for moldable jobs rather than for rigid jobs.

The current model improves upon these models in several respects. It employs more rigorous statistical procedures that have not been used previously in this field. It uses matching to the CDF of distributions, without relying on moments that may be unduly influenced by outliers. It is based on logs from three different locations, rather than only one or two. It is more thorough, by modeling parallelism, runtimes, the correlation between them, and the arrival process (including daily cycles).

An analysis by Talby et al., which compared these models and the logs upon which they are based, has found that the current model is closest to the average of all the others in all the metrics checked [23]. These metrics include the mean and variance of runtimes, total work, interarrival times, and degree of parallelism. Thus it can be said that this model is the most representative available in a general sense, albeit it does not completely match any of the different logs used in its creation.

Acknowledgments

This research was supported in part by the Israel Science Foundation (grant no. 219/99). The workload logs on which it is based are available on-line from the Parallel Workloads Archive [19]. The workload log from the SDSC Paragon was graciously provided by Reagan Moore and Allen Downey. The workload log from the KTH SP2 was graciously provided by Lars Malinowsky. The workload log from the LANL CM-5 was graciously provided by Curt Canada. Many thanks to them for making the data available and for their help with background information and interpretation. Thanks also to Prof. Ya'acov Ritov for statistics advice.

References

[1] M. Calzarossa, G. Haring, G. Kotsis, A. Merlo, and D. Tessera, "*A hierarchical approach to workload characterization for parallel systems*". In *High-Performance Computing and Networking*, pp. 102–109, Springer-Verlag, May 1995. Lect. Notes Comput. Sci. vol. 919.

[2] M. Calzarossa and G. Serazzi, "*Workload characterization: a survey*". *Proc. IEEE* **81(8)**, pp. 1136–1150, Aug 1993.

[3] A. B. Downey, "*A parallel workload model and its implications for processor allocation*". In 6th *Intl. Symp. High Performance Distributed Comput.*, Aug 1997.

[4] A. B. Downey and D. G. Feitelson, "*The elusive goal of workload characterization*". *Performance Evaluation Rev.* **26(4)**, pp. 14–29, Mar 1999.

[5] D. G. Feitelson, "*Packing schemes for gang scheduling*". In *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph (eds.), pp. 89–110, Springer-Verlag, 1996. Lect. Notes Comput. Sci. vol. 1162.

[6] D. G. Feitelson and M. A. Jette, "*Improved utilization and responsiveness with gang scheduling*". In *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph (eds.), pp. 238–261, Springer Verlag, 1997. Lect. Notes Comput. Sci. vol. 1291.

[7] D. G. Feitelson and L. Rudolph, "*Metrics and benchmarking for parallel job scheduling*". In *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph (eds.), pp. 1–24, Springer-Verlag, 1998. Lect. Notes Comput. Sci. vol. 1459.

[8] D. G. Feitelson and L. Rudolph, "*Toward convergence in job schedulers for parallel supercomputers*". In *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph (eds.), pp. 1–26, Springer-Verlag, 1996. Lect. Notes Comput. Sci. vol. 1162.

[9] J. L. Gustafson, "*Reevaluating Amdahl's law*". *Comm. ACM* **31(5)**, pp. 532–533, May 1988. See also *Comm. ACM* **32(2)**, pp. 262–264, Feb 1989, and *Comm. ACM* **32(8)**, pp. 1014–1016, Aug 1989.

[10] S. Hotovy, "*Workload evolution on the Cornell Theory Center IBM SP2*". In *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph (eds.), pp. 27–40, Springer-Verlag, 1996. Lect. Notes Comput. Sci. vol. 1162.

[11] J. Jann, P. Pattnaik, H. Franke, F. Wang, J. Skovira, and J. Riodan, "*Modeling of workload in MPPs*". In *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph (eds.), pp. 95–116, Springer Verlag, 1997. Lect. Notes Comput. Sci. vol. 1291.

[12] R. E. Kessler, M. D. Hill, and D. A. Wood, "*A comparison of trace-sampling techniques for multi-megabyte caches*". *IEEE Trans. Comput.* **43(6)**, pp. 664–675, Jun 1994.

[13] E. J. Koldinger, S. J. Eggers, and H. M. Levy, "*On the validity of trace-driven simulation for multiprocessors*". In 18th *Ann. Intl. Symp. Computer Architecture Conf. Proc.*, pp. 244–253, May 1991.

[14] A. M. Law and W. D. Kelton, *Simulation Modeling and Analysis.* McGraw Hill, 3rd ed., 2000.

[15] E. D. Lazowska, "*The use of percentiles in modeling CPU service time distributions*". In *Computer Performance*, K. M. Chandy and M. Reiser (eds.), pp. 53–66, North-Holland, 1977.

[16] D. Lifka, "*The ANL/IBM SP scheduling system*". In *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph (eds.), pp. 295–303, Springer-Verlag, 1995. Lect. Notes Comput. Sci. vol. 949.

[17] V. Lo, J. Mache, and K. Windisch, "*A comparative study of real workload traces and synthetic workload models for parallel job scheduling*". In *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph (eds.), pp. 25–46, Springer Verlag, 1998. Lect. Notes Comput. Sci. vol. 1459.

[18] G. J. McLachlan and K. Thriyambakan, *The EM Algorithm and Extensions*. Wiley, 1997.

[19] *Parallel Workloads Archive*. URL http://www.cs.huji.ac.il/labs/parallel/workload/.

[20] K. C. Sevcik, "*Application scheduling and processor allocation in multiprogrammed parallel processing systems*". *Performance Evaluation* **19(2-3)**, pp. 107–140, Mar 1994.

[21] M. S. Squillante, D. D. Yao, and L. Zhang, "*The impact of job arrival patterns on parallel scheduling*". *Performance Evaluation Rev.* **26(4)**, pp. 52–59, Mar 1999.

[22] X-H. Sun and L. M. Ni, "*Scalable problems and memory-bounded speedup*". *J. Parallel & Distributed Comput.* **19(1)**, pp. 27–37, Sep 1993.

[23] D. Talby, D. G. Feitelson, and A. Raveh, "*Comparing logs and models of parallel workloads using the co-plot method*". In *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph (eds.), pp. 43–66, Springer Verlag, 1999. Lect. Notes Comput. Sci. vol. 1659.

[24] D. Thiébaut, "*On the fractal dimension of computer programs and its application to the prediction of the cache miss ratio*". *IEEE Trans. Comput.* **38(7)**, pp. 1012–1026, Jul 1989.

[25] D. Thiébaut, J. L. Wolf, and H. S. Stone, "*Synthetic traces for trace-driven simulation of cache memories*". *IEEE Trans. Comput.* **41(4)**, pp. 388–410, Apr 1992. (Corrected in *IEEE Trans. Comput.* **42(5)** p. 635, May 1993).

[26] M. Wan, R. Moore, G. Kremenek, and K. Steube, "*A batch scheduler for the Intel Paragon with a non-contiguous node allocation algorithm*". In *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph (eds.), pp. 48–64, Springer-Verlag, 1996. Lect. Notes Comput. Sci. vol. 1162.