A Model for Moldable Supercomputer Jobs

Walfredo Cirne
Universidade Federal da Paraíba
Departamento de Sistemas e Computação
http://walfredo.cirne.net/comp

Francine Berman
University of California San Diego
Computer Science and Engineering
http://apples.ucsd.edu

The performance of supercomputer schedulers is influenced by the workloads that serve as their input. Realistic workloads are therefore critical to evaluate how supercomputer schedulers perform in practice. There has been much written in the literature about rigid parallel jobs, i.e. jobs that require partitions of a fixed size to run. However the majority of the parallel jobs in production today are moldable, i.e. jobs that can execute on a variety of partition sizes. In this paper, we describe a workload model for moldable jobs, which is based on a user survey and good analytical models. Our model can serve as the basis for the development of performance-efficient strategies for selection of the job partition size, as well as the basis for enhancing supercomputer schedulers to directly accept moldable request.

1. Introduction

Realistic workloads are critical for the design and analysis of parallel computer systems [2] [14] [19] [20]. In particular, validation of a system with respect to a realistic workload provides important information about the behavior of the system in practice. When workloads are not available or do not represent the real usage of the parallel system, there may be a discrepancy between the success of the system in theory and the success of the system in practice. The need for realistic workloads is particularly important in evaluating batch supercomputer schedulers. Such schedulers are critical to the effective use by many users of large parallel supercomputers (such as those at the PACIs, DOE and NASA labs, etc).

A good source of realistic workloads are *logs* that record the characteristics of jobs submitted to a production parallel computer. However, current workload logs only contain limited information about jobs. In particular, logs do not contain the set of partition sizes a given job can possibly use to run. Only the partition size actually used is available.

Parallel jobs that can only run on a given partition size are called *rigid jobs* in the literature, and there has been considerable discussion as to what fraction of jobs in a typical supercomputer workload is actually rigid. Respondents to a user survey indicated that a large fraction of supercomputer jobs is moldable. *Moldable jobs* are those parallel jobs that can execute on multiple partition sizes, but once a partition size is chosen, it cannot be changed during the execution of the job [12]. In current practice, a user who has a moldable job currently selects which partition size to use at submission time, turning the moldable job into a rigid one for the supercomputer scheduler.

Given that a substantial fraction of supercomputer jobs seem to be moldable (98% in a survey we conducted among supercomputer users), models of moldable jobs must be developed to guide the design and analysis of supercomputer schedulers that take advantage of workloads with moldable jobs. **In this paper, we describe a moldable job model that extends a rigid job into a moldable one.** A moldable workload can be generated by applying our model to all jobs of a rigid workload. We are currently using our model to evaluate application schedulers that improve the performance of individual jobs by adaptively selecting which partition size to use.

This paper is organized as follows: Section 2 provides an overview of our moldable job model and describes the survey we conducted among supercomputer users, which supplied important data for our model. Sections 3 and 4 describe how the model's parameters are distributed in practice. Section 3 addresses the partition sizes of a moldable job, while Section 4 focuses on the execution time of the moldable job on such partition sizes. Section 5 concludes the paper with a summary of this work and an indication of future work.

2. Model Overview

A *moldable job j* is a parallel job that can execute on any one of a collection of partition sizes $\boldsymbol{n} = (n^{[1]}, \ldots, n^{[v]})$. Note that the execution time of a moldable job *j* depends on the partition size $n^{[i]}$ it uses to run. For a given job *j*, the execution size typically decreases as the partition size grows (up to some point, at least). Therefore, in order to be useful, a model for moldable jobs has to provide not only a set of partition sizes $\boldsymbol{n} = (n^{[1]}, \ldots, n^{[v]})$ a moldable job *j* can use, but also the execution times $\boldsymbol{te} = (te^{[1]}, \ldots, te^{[v]})$ of j on each of these partition sizes.

We often find the need to refer to a partition size n, and the corresponding execution time te a job has when running with n processors. We define a job's *shape* to be this pair (partition size n, execution time te). A moldable job can thus have multiple shapes. A rigid job, in contrast, has only a single shape.

In our model, moldable jobs are an extension of rigid jobs. The model generates alternative shapes for a given job j, for which only one shape in known. More precisely, the moldable job model uses the known partition size n and execution time te to produce the v-tuples $\boldsymbol{n} = (n^{[1]}, \ldots, n^{[v]})$ and $\boldsymbol{te} = (te^{[1]}, \ldots, te^{[v]})$, which describe v shapes for job j.

A moldable workload is obtained from a rigid workload by applying our moldable job model to the jobs in the rigid workload. The source of the rigid workload is not important for our model. It can be a workload log (such as the those described in Table 1 and elsewhere [11] [15] [16] [18]) or originate from a model for rigid workloads [2] [5] [8] [9] [12] [14] [17].

In order to be able to build a realistic model, we designed a user survey to investigate the characteristics moldable jobs exhibit in practice. The survey consisted of 12 multiple-choice questions, and was conducted online via email and the Web between 17 April and 31 May 2000. Electronic questionnaires were distributed among supercomputer users at NASA, NCSA, NERSC, NPACI, and elsewhere. Responses to the survey were of course voluntary, which renders it a self-selected sampling. Multiple-choice questions were used because they raise the number of responses as well as ease the analysis of the results [1]. We received 214 responses to the survey. The survey questionnaire and results are available at http://apples.ucsd.edu/~survey/.

The survey results indicate that several parameters are important in realistically determining v (the number of shapes), $\boldsymbol{n}$ (the partition sizes), and $\boldsymbol{te}$ (the execution times). To determine v and $\boldsymbol{n}$, we must determine the minimum partition size c_{min} for j, the maximum partition size c_{max} for j, the probability pb that a partition size for j is a power-of-2, and the number of requests c_u that the user is willing to provide. The values that these parameters assume is the focus of Section 3. For now, it is relevant to know that most of these parameters are stochastic values (whose distributions are discussed in Section 3). Therefore, every time the model is used, a new set of parameters is randomly generated according to their distributions. Once the parameters have been fixed, we can determine v and $\boldsymbol{n}$. The number of shapes v available for job j is bounded by the range of partition sizes job j can use $[c_{min}, c_{max}]$, and by how many requests the user is willing to provide c_u. Symbolically, $v = \min(c_{max} - c_{min} + 1, c_u)$. After v is determined, we generate $\boldsymbol{n}$ by randomly choosing v partition sizes in the $[c_{min}, c_{max}]$ range. A randomly chosen partition size is replaced by its closest power-of-2 value with probability pb.

To determine $\boldsymbol{te}$, we use Downey's analytical model for job speed-up. Downey's model requires two parameters: the average parallelism A and an approximation to the coefficient of variance of parallelism σ. These parameters also take stochastic values (whose real-life distributions are described in Section 4). Therefore, every use of the model employs a statistic realization of A and σ, according to their distributions. Recall that speed-up measures how much faster a job j that uses n processors executes in comparison to j's execution using only one processor. That is: $S(n) = te(1) / te(n)$. Note that A, σ, n, and te uniquely determine the sequential execution time L of the job: $L = te(1) = te \cdot S(n)$. L represents how "large" a job is. The greater the L, the more processing is required to complete the job. With A, σ and L, we can determine the execution time of the job running over an arbitrary partition size n' by $te(n') = \dfrac{L}{S(n', A, s)}$. In particular, we can generate $\boldsymbol{te} = (te^{[1]}, \ldots, te^{[v]})$ by evaluating $te(n')$ at the partition sizes $\boldsymbol{n} = (n^{[1]}, \ldots, n^{[v]})$.

3. Moldable Partition Sizes

Most moldable jobs cannot run over a partition of arbitrary size. Factors such as memory requirements, amount of parallelism, and algorithmic constraints restrict the partition size that can be used by a given moldable job j. For example, memory requirements can establish a minimum partition size on which j can run, a factor we model as c_{min}. Similarly, the amount of parallelism determines the maximum partition size j can use, a factor we model as c_{max}. Moreover, some parallel algorithms have constraints on which partition sizes they can use. However, the partition sizes actually selected by users seem to be even more restricted than these algorithmic constraints, as we shall see in Section 3.1. More precisely, users' preference for power-of-2 partitions appear to be stronger than the algorithmic constraints. We will thus model the probability pb of a partition size to be a power-of-2.

Note that we cannot expect that the user will necessarily provide requests for *all* partition size a moldable job can possibly use. We define c_u to be the number of requests that the user is willing to provide. That is, c_u establishes an upper bound on v, the number of shapes for job j. The next subsections describe how pb, c_{min} and c_u are distributed in practice. c_{max} can be uniquely determined by the Downey's parameters A and σ (as we shall see in Section 4) and hence does not need to be modeled directly.

3.1. Partition Size Constraints

As Table 1 shows, the distribution of partition sizes are dominated by power-of-2 values in many workload logs. This phenomenon was also observed by others [7] [9] [13] [20]. There has been some controversy as to whether to incorporate the dominance of power-of-2 partitions into a workload model. Some researchers have accounted for the high incidence of power-of-2 jobs and modeled the partition size accordingly [13]. Others, however, believe that this phenomenon is mainly an artifact of behavioral inertia (the first parallel supercomputers *required* power-of-2 partition sizes) and the design of some submission interfaces (which "suggest" the submission of power-of-2 jobs) [7]. Based solely on workload logs, it is impossible to determine whether the prevalence of power-of-2 jobs is due to the nature of the parallel jobs, or is an artifact of behavioral inertia and interface design.

In the survey, we inquired about the constraints jobs have regarding partition size (question 7). Figure 1 summarizes the responses. To our surprise, the majority of the respondents (69.4% of them, excluding "do not know") described jobs that have no partition size restriction. This result indicates that most jobs are already moldable. It also suggests that the high incidence of power-of-2 requests in the workload logs is *not* an intrinsic characteristic of supercomputer jobs. We believe that the fact that the selection of partition size is made by humans also contributes to the prevalence of power-of-two partitions. When no constraint exists, humans tend to pick "round" numbers, and powers of two are many people's idea of "round number" when they deal with computers.

In short, it seems that supercomputer workloads do not *have* to be dominated by power-of-2 jobs, but in practice they *are*. We thus define the probability *pb* that a job been biased towards a power-of-2 partition size. Table 1 shows values *pb* assumed for our reference workloads and thus provides basic guidelines on which value to use for *pb*. Note that modeling the bias towards power-of-2 is particularly relevant because it strongly impacts the performance of many scheduling solutions [19].

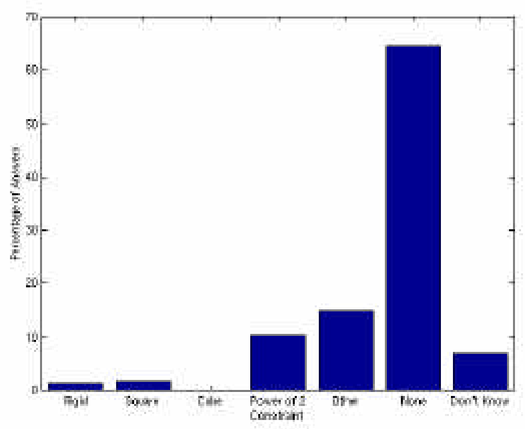

Figure 1 – Survey results concerning job partition size constraints

Machine	Processors	Jobs	Period	Power-of-2 Jobs
Argonne National Laboratory IBM SP2	120	7995	Oct 1996 Dec 1996	69.9%
Cornell Theory Center IBM SP2	430	79279	Jul 1996 May 1997	83.2%
Swedish Royal Institute of Technology IBM SP2	100	28479	Sep 1996 Aug 1997	73.4%
San Diego Supercomputer Center Intel Paragon	400	55526	Jan 1995 Dec 1996	83.7%
San Diego Supercomputer Center IBM SP2	128	16376	Jan 1999 May 1999	83.9%

Table 1 – Percentage of jobs with a power-of-2 partition size in five workload logs

3.2. Minimum Partition Size

Figure 2 displays the results for the survey question that asked for the minimum partition size that can be used to run the respondent's job (question 4). Note that most jobs can run sequentially, but some really need larger partitions. These characteristics suggest the use of the uniform-log distribution to model c_{min}. In such a distribution, the *logarithms* of the values are uniformly distributed. More precisely, a *uniform-log distribution* is characterized by parameters χ and ρ, and has cumulative distribution function $cdf(x) = c \cdot \log_2(x) + r$ [6].

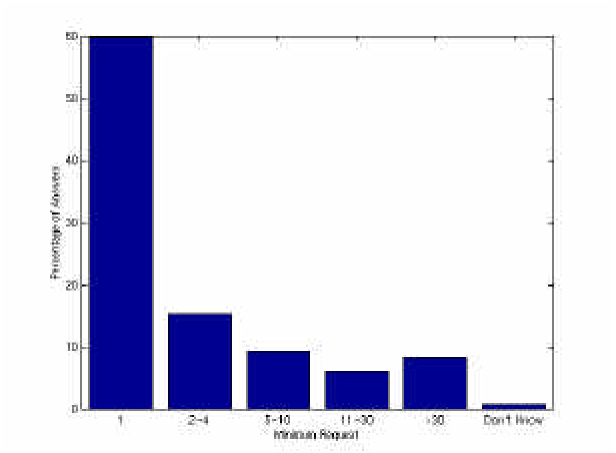

Figure 2 – Survey results for minimum partition size

Since we deal with many distributions in this paper, we index the distribution parameters with the variable they are modeling in order to avoid ambiguity. For example, the uniform-log parameters that model the minimum partition size c_{min} are written as χ_{cmin} and ρ_{cmin}. Using the method of the least squared error [4], we obtained χ_{cmin} = 0.06920 and ρ_{cmin} = 0.6279, which resulted in a very good fit to the survey responses, as shown in Figure 3.

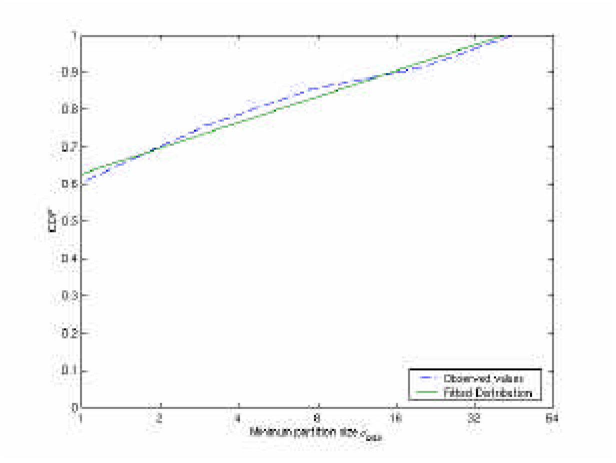

Figure 3 – Survey results and model for the minimum partition size c_{min}

3.3. Number of Requests Provided by the User

Figure 4 shows how many different partition sizes users have requested for their jobs. These results provide further support to our conjecture that most jobs are already moldable. Moreover, since they indicate how many requests users can currently use to submit their jobs, they are the natural estimate for c_u. It may be that such values increase when schedulers make it more advantageous for the user to determine more alternative requests, but we take a conservative approach and model c_u after the current practice.

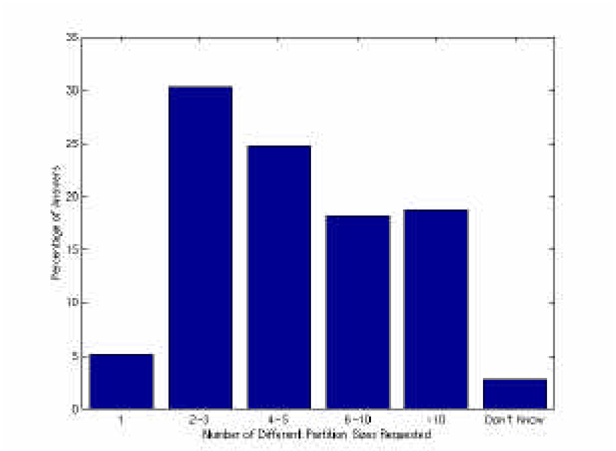

Figure 4 – Survey results for number of requests used to submit a job

In order to better model the survey results, we use a two-stage model, segregating the probability that $c_u = 1$ from the probability that $c_u > 1$. The survey indicates that around 5% of the jobs have $c_u = 1$. We thus make $\Pr[c_u = 1] = 0.05$ and $\Pr[c_u > 1] = 0.95$. Note that by making $c_u = 1$ for 5% of the jobs, we also address the fact that part of the workload is formed by rigid jobs (about 2% of the jobs seem to be rigid, see Figure 1). Here again, the user's choices of partition sizes seem to exhibit stronger restrictions than the intrinsic algorithmic constraints.

For $c_u > 1$, we use a uniform-log distribution to determine which value c_u assumes. By fitting the survey results through the least squared error method, we determined the parameters χ_{cu} = 0.1918 and ρ_{cu} = 0.1876. Figure 5 shows the model of the distribution when $c_u > 1$, as well as the plot of the corresponding survey results.

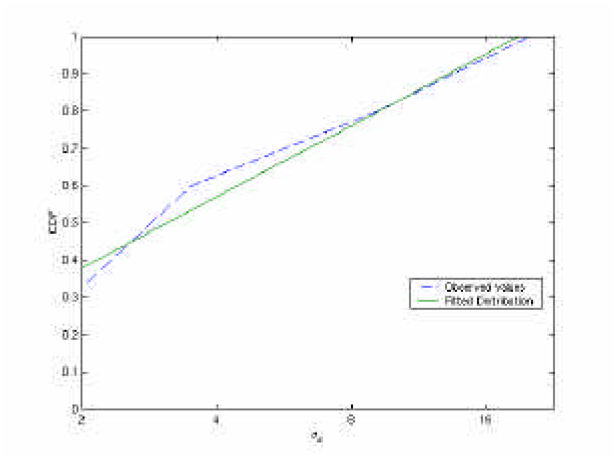

Figure 5 – Survey results and statistic model for $c_u > 1$

4. Moldable Execution Times

We use Downey's model of the speedup of parallel jobs [5] to derive the execution times $\boldsymbol{te} = (te^{[1]}, \ldots, te^{[v]})$ for the partition sizes $\boldsymbol{n} = (n^{[1]} \ldots, n^{[v]})$. Speed-up measures how much faster a job j that uses n processors executes in comparison to j's execution using only one processor. Symbolically: $S(n) = te(1) / te(n)$. Downey's speedup model uses two parameters: A (the *average parallelism*) and σ (an approximation of the *coefficient of variance in parallelism*). The speed-up of a job j is then given by:

$$S(n) = \begin{cases} \dfrac{An}{A + s(n-1)/2} & (s \le 1) \wedge (1 \le n \le A) \\ \dfrac{An}{s(A - 1/2) + n(1 - s/2)} & (s \le 1) \wedge (A \le n \le 2A - 1) \\ A & (s \le 1) \wedge (n \ge 2A - 1) \\ \dfrac{nA(s+1)}{s(n + A - 1) + A} & (s \ge 1) \wedge (1 \le n \le A + As - s) \\ A & (s \ge 1) \wedge (n \ge A + As - s) \end{cases}$$

Intuitively speaking, A establishes the maximum speedup a job can achieve. The larger the value of A, the greater the speedup a job can achieve. σ, on the other hand, determines how fast a job achieves its maximum speed-up (A). That is, σ determines how close to linear the speed-up is. The smaller the σ, the faster the job reaches its maximum speedup, and hence the closer to linear the speed-up curve is.

Notice that the determination of A and σ uniquely establishes c_{max}, the partition size after which adding more processors doesn't reduce the job's execution time. In effect, from the definition of $S(n)$, it follows that $c_{max} = \begin{cases} 2A - 1 & (s \le 1) \\ A + As - s & (s \ge 1) \end{cases}$.

In order to complete the description of our moldable job model, we need to establish how A and σ are distributed in practice. Unfortunately A and σ cannot be directly modeled after the survey. We felt that asking a direct question about the average in parallelism (A) or its coefficient of variance (a close approximation to σ) would be too technical for most users. Instead we indirectly inferred A and σ based on the survey's questions about the minimum, efficient and maximum partition sizes, as described below.

4.1. Modeling *A*

We use the survey's *efficient partition size* s_{effic} as an estimate for the average parallelism A. The efficient partition size s_{effic} was defined in the survey as "the partition size beyond which additional processors do not reduce the application's execution time enough to make it worth requesting them" (question 6). The intention is for s_{effic} to represent the "knee" in the speed-up curve, i.e. the point that maximizes the benefit/cost ratio (benefit meaning "lower execution time" and cost meaning "use of more processors"). This concept is in consonance with a more formal analysis of speed-up behavior by Eager et al [10]. Eager et al found that (i) the knee k of the speed-up curve must satisfy $\frac{A}{2} \le k \le 2A - 1$, and (ii) adding more processors when the partition size is smaller than A has much greater impact on the execution time than when the partition size is greater than A [10]. These results provide the rationale for modeling A after the efficient partition size s_{effic}.

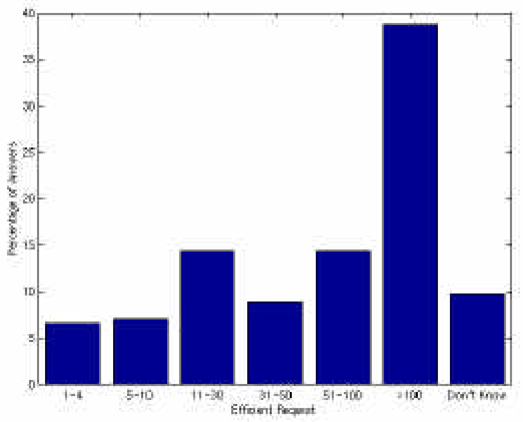

Figure 6 – Survey results for efficient partition size s_{effic}

Figure 6 displays how the efficient partition size s_{effic} was distributed among the survey respondents. Unfortunately, modeling A solely after s_{effic} can introduce a bias in the model. This is because s_{effic} and the minimum partition size s_{min} are correlated. Since $s_{min} \le s_{effic}$, the distribution of s_{effic} skews towards larger values as s_{min} grows. For example, Figure 7 presents the distribution of s_{effic} for the survey responses that had s_{min} in the [11, 30] range. Since we are already using s_{min} to model c_{min}, we must take this correlation into consideration when modeling A.

To understand how s_{min} and s_{effic} interact, consider their joint distribution of probability [4], whose CDF is shown in Figure 8. Note that the Figure's axes are in log scale, which suggests that a generalization of the uniform-log distribution might provide an adequate fit for this joint distribution. Note also that the CDF slope is more accentuated for large values of s_{min} (compared to small values of s_{min}). That is because when s_{min} is large, s_{effic} is also large, as exemplified in Figure 7.

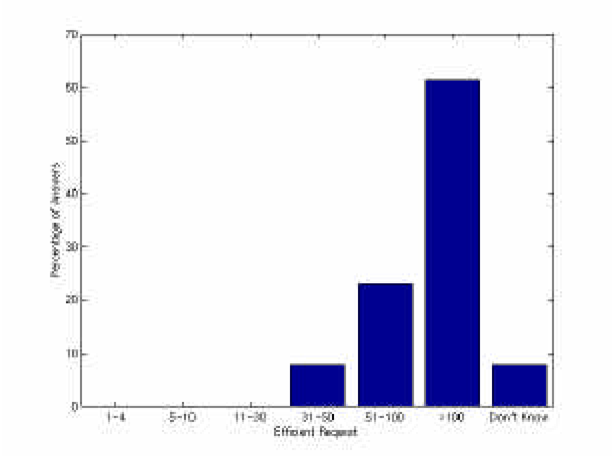

Figure 7 – Distribution of s$_{\text{effic}}$ *for the responses with* s$_{\text{min}}$ *in the [11,30] range*

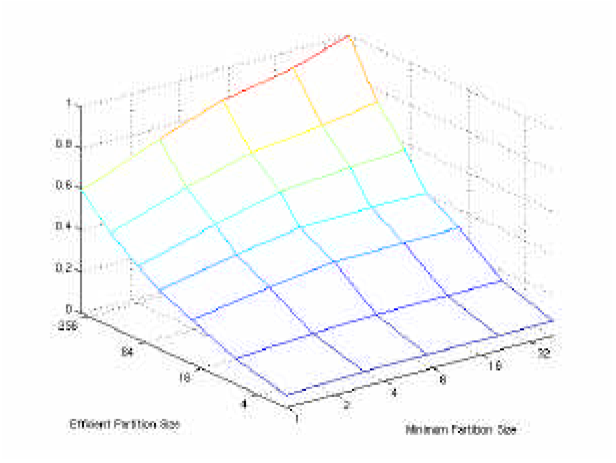

Figure 8 – Joint CDF for s$_{\text{effic}}$ *and* s$_{\text{min}}$

We are able to capture this behavior by using a *joint uniform-log distribution*, which is a generalization of the uniform-log distribution. The joint uniform-log distribution is determined by parameters φ, γ, η and ρ, and has $cdf(x,y) = j \cdot \log_2(x) \cdot \log_2(y) + g \cdot \log_2(x) + h \cdot \log_2(y) + r$. Making $x = s_{min}$ and $y = s_{effic}$, we found $\varphi_A = 0.009548$, $\gamma_A = -0.01877$, $\eta_A = 0.07468$, and $\rho_A = -0.009198$ via least squares fit. The fit was very good, with correlation coefficient of 0.974. The resulting joint uniform-log distribution for A and c_{min} can be seen in Figure 9.

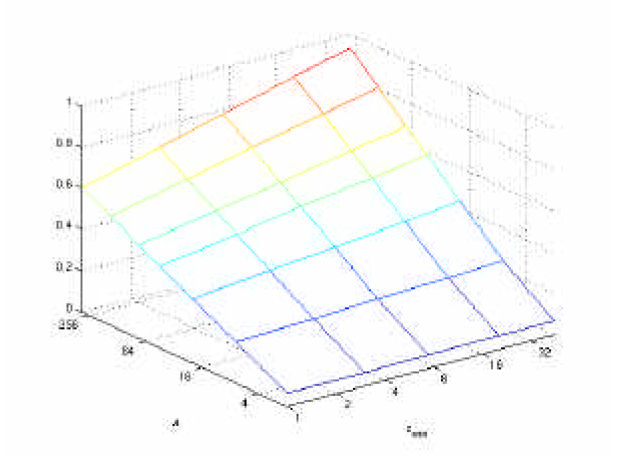

Figure 9 – Model CDF for the joint distribution of c$_{\text{min}}$ *and* A

4.2. Modeling s

As discussed above, σ cannot be directly modeled after a question asked in the survey. Instead we indirectly infer σ based on the relation between the efficient partition size s_{effic} (question 6) and the maximum partition size s_{max} (question 5). The idea is that when the s_{effic} and s_{max} are close, the speedup is close to linear, and thus the job has small σ. Conversely, when s_{effic} and s_{max} are far apart, the speedup should be strongly sublinear, and hence the job has large σ.

More specifically: We can again use s_{effic} as an estimate for A. Assuming $s_{effic} = A$ and using the equations that define the Downey model, we have that (i) $s_{effic} = s_{max} \Leftrightarrow s = 0$, and (ii) $s > 0 \Rightarrow s = \dfrac{s_{max} - s_{effic}}{s_{effic} - 1}$.

Since $s = \dfrac{s_{max} - s_{effic}}{s_{effic} - 1}$ is a generalization of both equations (i) and (ii), we use it as the estimate for σ. Figure 10 displays how these estimates for σ are distributed in the survey's results. Note that the discontinuities in the graph are due to the fact that the survey used multiple-choice questions, therefore creating an artificial discretization for s_{effic} and s_{max}.

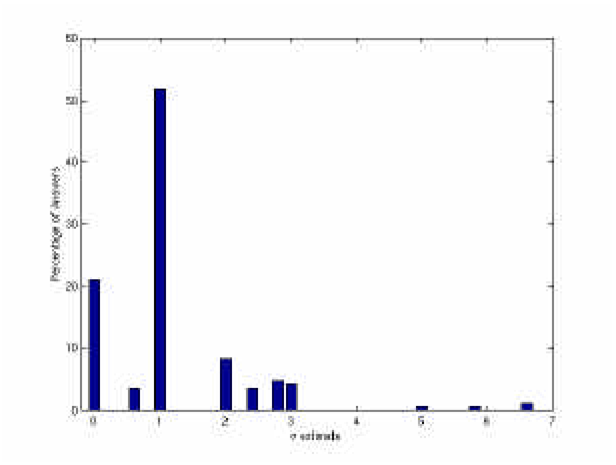

Figure 10 – Distribution of the s *estimates*

We use a normal distribution to model σ. It provides a good fit for the σ estimates derived from the survey, especially when we consider that the discontinuities in the distribution of such estimates are an artifact of the survey. Using maximum likelihood fitting [4], we obtained parameters $\mu_\sigma = 1.209$ and $\sigma_\sigma = 1.132$. Figure 11 shows the CDF of the model and the observed estimates.

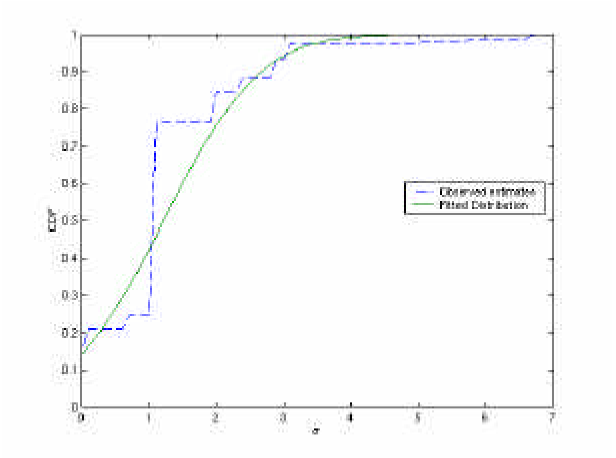

Figure 11 – CDF for the survey based estimate of s *and the corresponding model*

5. Summary

This work describes a model for moldable supercomputer jobs. The moldable job model takes as input a rigid job with only one known shape (partition size and execution time) and provides as output a realistic set of shapes for the job. A moldable workload model can be generated by applying the moldable job model to all jobs of an input rigid workload.

The moldable job model captures important characteristics that moldable jobs display in practice. These include memory constraints, maximum parallelism, algorithmic constraints on partition size, user behavior in generating requests, and speedup. A user survey was used as basis for the realistic estimation of the parameters that model these characteristics.

One key result of the user survey is that most parallel jobs are already moldable. This suggests that good scheduling strategies that assume jobs to be moldable would have an immediate impact when deployed. Although some research has already been done in scheduling moldable jobs [3] [7], scheduling strategies for moldable job are not currently in production. We expect our model to encourage and facilitate research for such scheduling strategies. We are currently using our model to evaluate application schedulers that improve the performance of individual jobs by adaptively selecting which partition size to use.

There is still much work to be in modeling supercomputer workloads. Topics that need further research include accuracy of requests, cancellation of requests, priorities, user behavior and its feedback effect, memory requirements, and more [9].

Acknowledgments

We thank the supercomputer users who answered our survey. We thank our supporters: CAPES grant DBE2428/95-4, DoD Modernization contract 9720733-00, NPACI/NSF award ASC-9619020, and NSF grant ASC-9701333. We also want to express our gratitude to Victor Hazlewood, Dror Feitelson and the people who made their workloads available at the Parallel Workloads Archive [15]. Finally, thanks to the members of the AppLeS group, Rich Wolski, Allen Downey, Warren Smith, and Reagan Moore for the insightful discussions and valuable suggestions.

References

[1] Earl Babbie. *Survey Research Methods*. Wadsworth Publishing Company, 2nd edition, 1990.

[2] Steve Chapin, Walfredo Cirne, Dror Feitelson, James Jones, Scott Leutenegger, Uwe Schwiegelshohn, Warren Smith, and David Talby. *Benchmarks and Standards for the Evaluation of Parallel Job Schedulers*. In Job Scheduling Strategies for Parallel Processing, *D*. Feitelson and Larry Rudolph (Eds.) Springer-Verlag, Lecture Notes in Computer Science, vol. 1659, pp. 66-89, 1999.
http://www-cse.ucsd.edu/users/walfredo/resume.html#publications

[3] Walfredo Cirne and Francine Berman. *Adaptive Selection of Partition Size for Supercomputer Requests*. Proceedings of 6th Workshop on Job Scheduling Strategies for Parallel Processing , May 2000

http://www-cse.ucsd.edu/users/walfredo/resume.html#publications

[4] Jay Devore. *Probability and Statistics for Engineering and the Sciences*. Fourth Edition, Wadsworth Publishing Company, 1995.

[5] Allen Downey. *A model for speedup of parallel programs*. U.*C*. Berkeley Technical Report CSD-97-933, January 1997. http://www.sdsc.edu/~downey/model/

[6] Allen Downey. *Predicting queue times on space-sharing parallel computers*. 11th International Parallel Processing Symposium (IPPS'97), Geneva, Switzerland, April 1997. http://www.sdsc.edu/~downey/predicting/

[7] Allen Downey. *Using Queue Time Predictions for Processor Allocation*. In Job Scheduling Strategies for Parallel Processing, Springer-Verlag, Lecture Notes in Computer Science Vol. 1291, Dror Feitelson and Larry Rudolph (eds.), 1997. http://www.sdsc.edu/~downey/predalloc/

[8] Allen Downey. *A parallel workload model and its implications for processor allocation*. 6th IEEE International Symposium on High Performance Distributed Computing (HPDC'97), August 1997. http://www.sdsc.edu/~downey/allocation/

[9] Allen Downey and Dror Feitelson. *The elusive goal of workload characterization*. Perf. Eval. Rev. 26(4), pp. 14-29, March 1999. http://www.cs.huji.ac.il/~feit/pub.html

[10] Derek Eager, John Zahorjan, and Edward Lazowska. *Speedup Versus Efficiency in Parallel Systems*. IEEE Transactions on Computers, vol. 38, no. 3, March 1989.

[11] Dror Feitelson and Bill Nitzberg. *Job characteristics of a production parallel scientific workload on the NASA Ames iPSC/860*. In Job Scheduling Strategies for Parallel Processing, Dror Feitelson and Larry Rudolph (Eds.), Lecture Notes in Computer Science Vol. 949, pp. 337-360, Springer-Verlag, 1995. http://www.cs.huji.ac.il/~feit/pub.html

[12] Dror Feitelson, Larry Rudolph, Uwe Schweigelshohn, Kenneth Sevcik, and Parkson Wong. *Theory and Practice in Parallel Job Scheduling*. 3rd Workshop on Job Scheduling Strategies for Parallel Processing, Springer-Verlag Lecture Notes in Computer Science, vol. 1291, pp. 1-34, April 1997. http://www.cs.huji.ac.il/~feit/parsched/parsched97.html

[13] Dror Feitelson and Morris Jette. *Improved Utilization and Responsiveness with Gang Scheduling*. In Job Scheduling Strategies for Parallel Processing, Dror Feitelson and Larry Rudolph (Eds.), pp. 238-261, Lecture Notes in Computer Science Vol. 1291, Springer-Verlag, 1997. http://www.cs.huji.ac.il/~feit/pub.html

[14] Dror Feitelson and Larry Rudolph. *Metrics and Benchmarking for Parallel Job Scheduling*. In Job Scheduling Strategies for Parallel Processing, Dror Feitelson and Larry Rudolph (Eds.), pp. 1-24, Springer-Verlag, Lecture Notes in Computer Science vol. 1459, 1998.

[15] Dror Feitelson. *The Parallel Workloads Archive*. http://www.cs.huji.ac.il/labs/parallel/workload/

[16] Victor Hazlewood. *NPACI JobLog Repository*. http://joblog.npaci.edu/

[17] Joefon Jann, Pratap Pattnaik, Hubertus Franke, Fang Wang, Joseph Skovira, and Joseph Riordan. *Modeling of Workload in MPPs*. In Job Scheduling Strategies for Parallel Processing, Springer-Verlag, Lecture Notes in Computer Science Vol. 1291, Dror Feitelson and Larry Rudolph (eds.), 1997.

[18] James Jones and Bill Nitzberg. *Scheduling for Parallel Supercomputing: A Historical Perspective of Achievable Utilization*. In Job Scheduling Strategies for Parallel Processing, Springer-Verlag, Lectures Notes in Compututer Science vol. 1659, 1999.

[19] 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, Dror Feitelson and Larry Rudolph (eds.), pp. 25-46, Springer Verlag, Lect. Notes Comput. Sci. vol. 1459, 1998.

[20] Jaspal Subhlok, Thomas Gross, and Takashi Suzuoka. *Impact of Job Mix on Optimizations for Space Sharing Schedulers*. Supercomputing'96, 1996. http://www.supercomp.org/sc96/proceedings/SC96PROC/TTITLES.HTM