A Comprehensive Model of the Supercomputer Workload

Walfredo Cirne
Universidade Federal da Paraíba
Departamento de Sistemas de Computação
http://walfredo.dsc.ufpb.br

Francine Berman
University of California San Diego
Computer Science and Engineering
http://www-cse.ucsd.edu/~berman

As with any computer system, the performance of supercomputers depends upon the workloads that serve as their input. Unfortunately, however, there are many important aspects of the supercomputer workloads that have not been modeled, or that have being modeled only incipiently. This paper attacks this problem by considering requested time (and its relation with execution time) and the possibility of job cancellation, two aspects of the supercomputer workload that have not been modeled yet. Moreover, we also improve upon existing models for the arrival instant and partition size.

1. Introduction

Since the performance of a computer system depends on the workload submitted to such a system [2] [3] [17] [21], workload modeling plays a vital role in performance evaluation. A workload model enables the researcher to explore the performance of the system in a multitude of scenarios. As with any model, however, a key issue with workload models is how well they represent reality. When workloads do not represent the real usage of the system, there may be a discrepancy between the success of the system on paper and the success of the system in practice.

The need for realistic workloads model is particularly important in evaluating large massively parallel supercomputers (such as those at PACIs, DOE and NASA labs, etc). This is because such supercomputers are very expensive machines and thus conducting extensive experiments on them is rarely an option. A good workload model allows for the use of simulation to investigate innovative design solutions and novel scheduling policies for parallel supercomputers.

Although supercomputer workload modeling has been target of considerable interest in recent years [3] [5] [9] [11] [12] [14] [17] [20], there are many important aspects of the supercomputer workloads that have not been modeled, or that have being modeled only incipiently [12]. In fact, most models characterize a supercomputer job through three parameters only, namely: the arrival instant, the partition size (i.e., the number of processors used by the job), and the execution time. There are other aspects of a supercomputer job *that affect performance* that remain poorly understood. For example, most supercomputers sites require the user to supply the job's *requested* *time*, an upper bound to the job's execution time. Since the relation between execution time and request time has impact on the performance attained by the supercomputer scheduler [16] [22], models that overlook requested time can produce misleading results.

This paper attacks this very problem, providing a model for the supercomputer workload that encompasses two aspects that have not been modeled yet. More precisely, our model addresses requested time (and its relation with execution time) and the possibility of job cancellation. Moreover, our modeling for arrival instant is also novel because it takes into account the seasonal workday cycle. Finally, a user survey we conducted allowed us to shed some light on an ongoing discussion, namely: whether the prevalence of jobs with power-of-2 partition sizes is intrinsic to the workload [15], or is an artifact of behavioral inertia and poor submission interface design [10].

This paper is organized as follows. Section 2 describes typical supercomputer usage, introducing the elements a workload model for supercomputers. Section 3 presents the data we used to base our model upon. Sections 4 to 7 present the model per se: Section 4 describes how we model the arrival instant; Section 5 introduces our modeling for jobs that are canceled by the user; Section 6 discusses our findings regarding partition size; and Section 7 explains the modeling of execution time and requested time, which display strong correlation. Section 8 includes a note on validation. Section 9 closes the paper with a summary.

2. Supercomputer Usage

Parallel supercomputers (or simply *supercomputers* in this paper) are high-end machines designed to support the execution of parallel jobs. A parallel supercomputer is composed of many processors, each with its own memory. The processors are interconnected by very fast internal networking. In order to promote the performance of parallel jobs whose tasks frequently communicate and synchronize, parallel supercomputers are typically *space-shared*. That is, jobs receive a dedicated *partition* to run for a pre-established amount of time.

Since jobs have dedicated access to processors in a spaced-shared supercomputer, an arriving job may not find enough resources to execute immediately. When this happens, the arriving job waits until enough processors become available. More precisely, jobs that cannot start immediately are placed in the *waiting queue*, which is controlled by the supercomputer scheduler. The *supercomputer scheduler* is the entity that receives requests to run jobs. It decides when jobs start and what processors they use. In particular, the supercomputer scheduler decides which job in the wait queue is the next to run. In order to make this decision, it typically requires each job to specify n, the number of processors it needs, and tr, the time requested for execution of the job. In the current practice, the supercomputer scheduler enforces the request time tr. That is, a job is killed if it exceeds its request time tr. Of course, the user can also kill a job at any time. If the cancelled job is yet in the waiting queue, it is removed from it and never starts.

In short, a supercomputer workload is composed of a stream of jobs. Each job j is characterized by its instant of arrival ia, partition size n, requested time tr, and execution time te. For the jobs that are cancelled by the user, we also want to know their instant of cancellation $ic > ia$.

3. Collecting Information

Since a key goal here is to produce a realistic model, we need information on the workloads experienced by production supercomputers. We considered workload logs from different sources (from fellow researchers to supercomputer centers to the web [18] [19]). Our criteria for using a log as *reference* for our model were that (i) the log should come from a supercomputer that could receive arbitrary requests (not only requests for power-of-2 partitions), and (ii) the log should contain a minimum of information to be useful in building our model (i.e., submission time, partition size, requested time, and execution time). We were able to find four workloads that meet these criteria. Such workloads are summarized in Table 1.

Name	**Machine**	**Processors**	**Jobs**	**Period**
ANL	Argonne National Laboratory SP2	120	7995	Oct 1996 Dec 1996
CTC	Cornell Theory Center SP2	430	79279	Jul 1996 May 1997
KTH	Swedish Royal Institute of Technology SP2	100	28479	Sep 1996 Aug 1997
SDSC	San Diego Supercomputer Center SP2	128	16376	Jan 1999 May 1999

Table 1 – Workloads used in this research

All four reference logs come from IBM SP2 machines. This is because they were the only ones that meet our criteria. In particular, many of the available logs miss the jobs' requested times. We believe that using SP2 logs does not bias our results in any way, as SP2s are typical representatives of the machines we target, namely distributed-memory spaced-shared parallel supercomputers.

4. Instant of Arrival

The pattern of job submission is affected by the work cycles of the supercomputer's users [10] [13] [17]. For example, typically more jobs are submitted during the day than during the night, as seen in Figure 1 (which indicates how many jobs arrived by hour of the day for the reference workloads). For a workload model to better capture the dynamics of the system, such behavior must be represented.

Methodology

As Figure 1 suggests, it is very hard to model the job arrival time through "common" distributions. In this work, we apply the methodology proposed by Calzarossa and Serazzi [1] to numerically fit a polynomial to the job arrival rate λ_a. Although the methodology was conceived to model the process arrival for a uniprocessor time-shared system, we found it to be applicable in our scenario, namely job arrivals for parallel supercomputers.

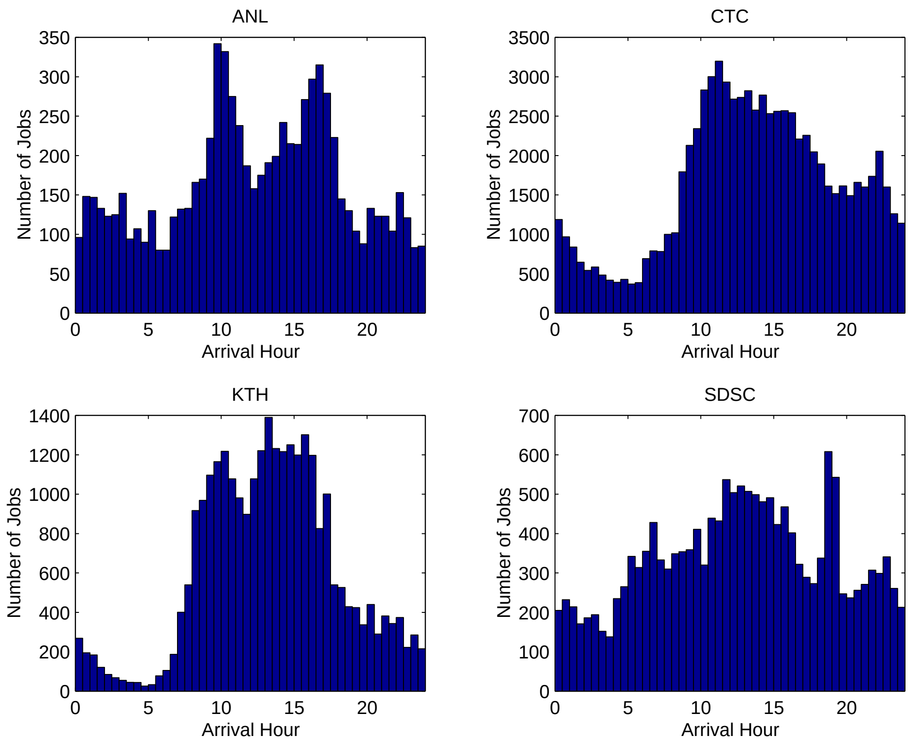

Figure 1 – Histograms of arrival hour for our reference workloads

In order to fit a polynomial to the arrival rate found in our reference workload logs, we must derive the arrival rate from the arrival instants (which are in the logs). Following the methodology of Calzarossa and Serazzi [1], we smoothed out the arrival rate by using a moving average estimator. More precisely, for a given time *m* (in minutes), the arrival rate $\lambda_a(m)$ is estimated using all arrivals in the 10 minute interval centered on *m*. Also, in order to avoid numerical instabilities while fitting the data, we scale a given minute of the arrival *m* to the range [-0.5, +0.5] as follows:

$$m_s = \frac{m - \frac{m_{max} - m_{min}}{2}}{m_{max} - m_{min}}$$

The polynomial fitting itself is done using the least square error estimator [8]. The degree *d* of the polynomial is chosen by incrementing *d* until the square error doesn't decrease significantly over two successive increments [1].

Eliminating Outliers

The analysis of the data from the reference logs reveals that a few days strongly deviate from the normal submission pattern. For example, Figure 1 shows the arrival of jobs at SDSC to have a spike between 18:30 and 19:30, a phenomenon that is not present in any of the other workloads. It turns out that, on 7 February 1999, 592 jobs (of which 579 were from the same user) were submitted to the supercomputer between 18:30 and 19:30. By not considering that single day, the spike in the graph disappears.

Such "uncommon" days appear in all workloads. In effect, we fitted a polynomial *per day* and ran a cluster analysis technique to classify the days in two groups. For all supercomputers, the clustering technique segregated *a single* day in one of the two groups. Since we want to model the common usage of a supercomputer, we eliminated the uncommon days that were clustered alone.

More precisely, we ran a Z score [8] over the coefficients of the polynomials to make the magnitude of such coefficients unimportant, and regarded the results as a point in $\mathbf{R}^d$. We then applied standard hierarchical cluster analysis (using the euclidian distance in $\mathbf{R}^d$) to separate the days in two groups. If a group consists of a single day (i.e., all other days are closer to each other than to this day), it is considered an outlier and excluded from the data. The procedure is repeated until no day is clustered alone. Table 2 shows how many days were excluded from each supercomputer log.

Workload	Days in the log	Uncommon Days Excluded
ANL	78	4
CTC	339	3
KTH	745	2
SDSC	150	2

Table 2 – Amount of days eliminated from the experimental data

We also used cluster analysis to look for other patterns of day-to-day variations. In particular, we tried to establish a statistically valid differentiation between weekdays and weekends. However, we were not able to find a way to cluster the days into disjoint groups that could be explored to enhance the model.

Fitting Arrival Rate

The polynomial fitting per se was a straightforward task. ANL required a degree 12 polynomial, CTC a degree 8 polynomial, KTH a degree 13 polynomial, and SDSC a degree 10 polynomial. Table 3 shows the coefficients of such polynomials. We suspect that ANL and KTH required higher-degreed polynomials to better model the small decrease in the job arrival rate verified around 12:00 (see Figure 1). The CTC and SDSC reference workloads do not present such a decrease.

The fitted polynomials are very different from one another. This suggests that there is no single model for the arrival time that works well across different sites. This is in agreement with the original work of Calzarossa and Serazzi, who warn against using the polynomial they found for alternative contexts [1]. Rather they highlight the importance of their work as a methodology.

Figure 2 shows the observed arrival rate for each workload under consideration, as well as the polynomials that fit them. It is important to point out that filtering out the outliers made it possible to model the workloads with simpler, lower-degree polynomials that avoided the uncommon behavior of a few days. More notably, the SDSC model avoids the one-day spike around 19:00 discussed above.

Term	ANL	CTC	KTH	SDSC
1	7166.5	254.04	-32683	-5499.6
m_s	64174	-25.820	-61444	1527.0
m_s^2	866.10	-258.51	32553	3015.9
m_s^3	-45317	8.4442	49406	-951.95
m_s^4	-2922.9	81.612	-12611	-573.08
m_s^5	11761	-3.6628	-15113	199.32
m_s^6	895.15	-9.6309	2380.9	48.583
m_s^7	-1349.0	0.76455	2154.0	-15.611
m_s^8	-93.266	0.56501	-221.65	-2.5879
m_s^9	62.485		-134.82	0.36675
m_s^{10}	2.2601		8.3039	0.21262
m_s^{11}	-0.66152		1.9175	
m_s^{12}	0.17191		0.022406	
m_s^{13}			0.097106	

Table 3 – Coefficients of the polynomials that model job arrival rate

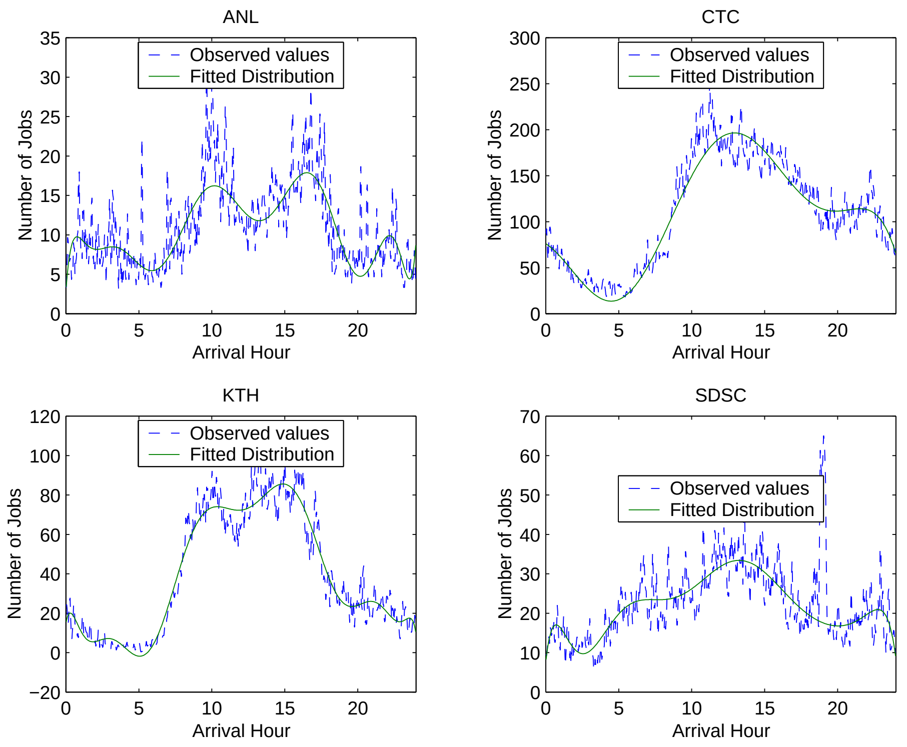

Figure 2 – Observed data and fitted model for arrival rate

We should also mention that we tried to correlate arrival time with other parameters. For example, we expected large jobs to be submitted at night. However, this was not supported by the reference workload logs. The correlation coefficient between these two parameters was low (in the range [-0.1, 0.1]) for all reference workloads. More generally, we didn't detect strong correlation between arrival instants and any other parameter we investigated.

5. Cancelled Jobs

Not all supercomputer jobs complete their execution. Some of them are *cancelled* by the user. Cancelled jobs may affect the course of action of the scheduler, even when they are cancelled before they start. Cancellation may also make the load offered by a given workload to change depending on the scheduler being used. In fact, consider a scheduler *s* that finishes a job *j* before its cancellation arrives, and a scheduler *r* for which the cancellation of *j* arrives before the job starts. Everything else being the same, scheduler *s* has to deal with a greater load than scheduler *r*. Consequently, due to its impacts on scheduling, cancellation should be taken into account when modeling supercomputer workloads.

We model cancelled jobs by providing a probability *pc* that a given job will be cancelled. Moreover, for each cancelled job, we also need to know the instant of cancellation *ic*, or alternatively the *cancellation lag lc = ic – ia* (where *ia* is the instant of the arrival of the job in the system).

We have information on cancellations only for CTC and SDSC. The ANL and KTH logs do not discriminate between completed and cancelled jobs. Table 4 displays the percentage of completed and cancelled for the CTC and SDSC reference workloads. Such values can be used to provide realistic estimates of the probability of cancellation *pc*.

Workload	**Cancelled Jobs**
CTC	12.22%
SDSC	23.31%

Table 4 – Percentage of completed and cancelled jobs

A visual inspection of the cancellation lag histograms for both CTC and SDSC show a fat-tailed distribution (see Figure 3). Although the cancellation lag can be as large as 10 days for the SDSC workload and over 2 months for the CTC workload, most cancellations happen shortly after the job's submission. In fact, 54.65% of SDSC's and 24.73% of CTC's cancellations happen less than 10 minutes after the job submission.

To build a realistic workload model, it is important to represent both the agglomeration of short cancellation lags *lc* and the fat tail of the distribution. This behavior can be effectively modeled using a uniform log distribution. 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 $cdf(x) = \chi \cdot \log_2(x) + \rho$ [10]. 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 cancellation lag *lc* are written as χ_{lc} and ρ_{lc}.

Using the least-square linear regression technique, we fitted SDSC's cancellation lags obtaining χ_{lc} = 0.06442 and ρ_{lc} = -0.1498. For CTC, we obtained χ_{lc} = 0.06421 and ρ_{lc} = -0.3180. The SDSC and CTC values for χ_{lc} and ρ_{lc} are similar, suggesting that the cancellation lag may not change significantly across workloads (unlike arrival rate). Figure 4 shows the data as well as the fitted uniform-log distributions for both CTC and SDSC.

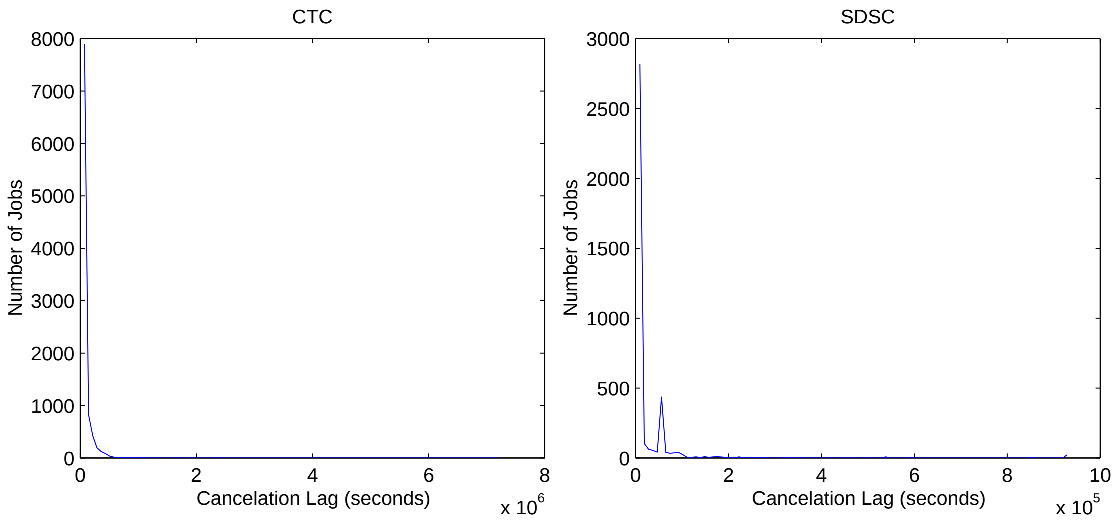

Figure 3 – Histograms of cancellation lag

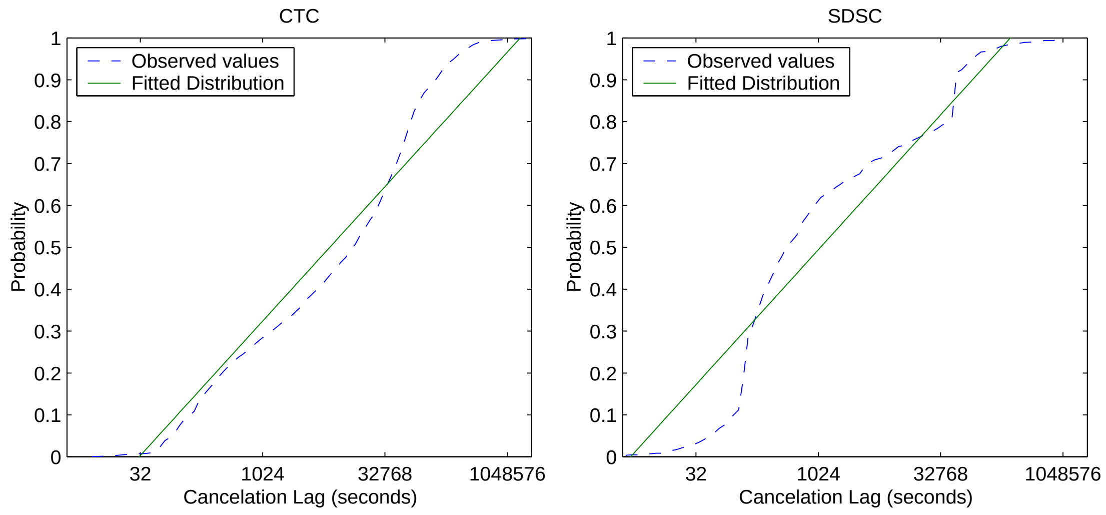

Figure 4 – Cancellation Lag CDF

6. Partition Size

As first noticed by Downey [10] [11], the uniform-log distribution also provides a good fit for the partition sizes in a supercomputer workload log. This was the case for our four reference workloads. Table 5 shows the parameters obtained by fitting the observed partition sizes to a uniform-log distribution via the method of least squared error, and Figure 5 plots the observed partition sizes and the uniform-log distributions fitted to them.

Workload	χ_n	ρ_n
ANL	0.1381	0.2163
CTC	0.0709	0.5283
KTH	0.0893	0.4764
SDSC	0.1150	0.2537

Table 5 – χ_n and ρ_n obtained by fitting partition size to a uniform-log distribution

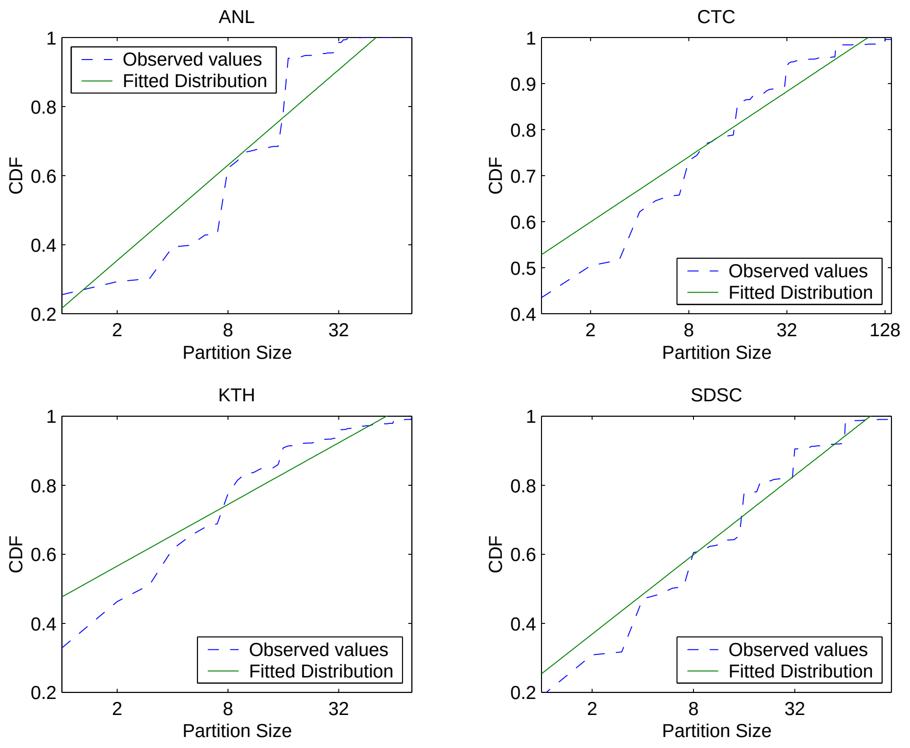

Figure 5 – Uniform-log fit for partition size

However, the uniform-log distribution alone does not capture a salient characteristic of how partition sizes are distributed. As Table 6 and Figure 6 show, the distribution of partition sizes seems to be dominated by power-of-2 values. This is a relevant characteristic because the fraction of power-of-2 jobs in the workload strongly influences the performance of the system [21].

Workload	**Power-of-2 Jobs**
ANL	69.87%
CTC	83.16%
KTH	73.45%
SDSC	83.95%

Table 6 – Percentage of jobs with a power-of-2 partition size

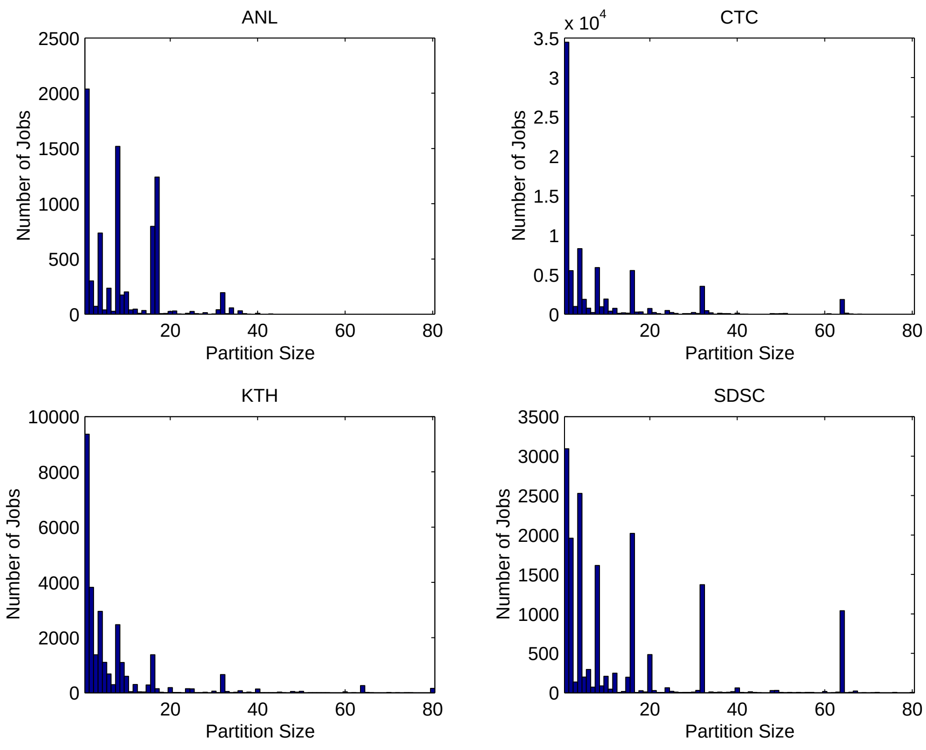

Figure 6 – Histogram of partition sizes

There has been some controversy on 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 [15]. Others, however, believe this is mainly due to old habits (the first parallel supercomputers *required* power-of-2 partition sizes) and the design of some submission interfaces (which use power-of-2 by default) [10]. Based solely on the workload logs, it is impossible to decide 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.

Between 17 April and 31 May 2000, we conducted a survey among supercomputer users that, among other questions, inquired about the constraints jobs have regarding partition size [4] [5]. Figure 7 summarizes the responses. To our surprise, the majority of the answers (69.4% of them, excluding "do not know") described jobs that have no partition size restriction. This result suggests that the high incidence of power-of-2 requests in the workload logs is *not* an intrinsic characteristic of the jobs. It thus seems that the popularity of power-of-two partitions is due to behavioral inertia and interface design. We believe that the fact that the selection of partition size is made by humans contributes for 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.

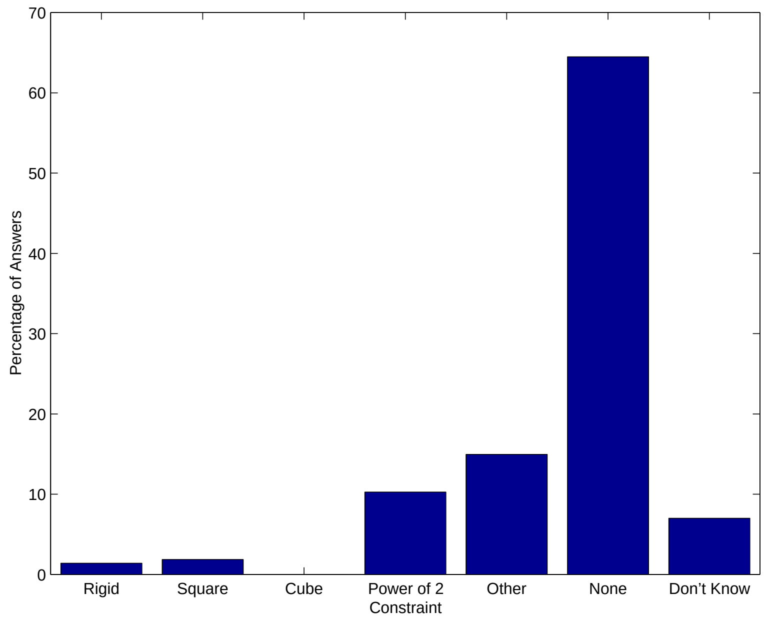

Figure 7 – Survey results for the constraints jobs face regarding partition size

In short, it seems that supercomputers workloads *do not have* to be dominated by power-of-2 jobs, but in practice they *are*. We believe that the high percentage of power-of-2 jobs is related to fact that the user is the one who chooses the partition sizes. Since we expect the user to keep playing this role, our model for partition size has a bias in favor of power-of-two partitions.

More precisely, we define the probability *pb* of a job been a power-of-2 partition size. Table 6 shows values *pb* assumed for our reference workloads. We start by using a uniform-log distribution to generate the partition size. The resulting partition size then has a probability *pb* of being changed to its closest power-of-2 value.

We also checked for high incidence of square, even, multiple-of-4, and multiple-of-10 jobs among the jobs that are not power-of-2. As can be seen in Table 7, our four reference workloads do not exhibit a concentration of such jobs that is large enough to be explored in the model.

Workload	**Square**	**Even**	**Multiple-of-4**	**Multiple-of-10**
ANL	9.55%	28.27%	5.56%	9.92%
CTC	10.35%	49.53%	19.14%	24.05%
KTH	17.88%	36.93%	15.84%	16.91%
SDSC	6.09%	61.49%	37.67%	30.75%

Table 7 – Percentages of different kinds of non-power-of-2 jobs

7. Execution Time and Requested Time

Execution time *te* and requested time *tr* are obviously related ($te \le tr$). In order to capture the relationship between *te* and *tr* in the model, we define the *request accuracy a* as the fraction of the requested time that was indeed used by a job. That is, $a = te / tr$. Since jobs cannot run longer than the amount of time they request, *a* is always a number between 0 and 1.

Note that we only need to model two parameters out of *te*, *tr*, and *a*. The third parameter can be derived from the others (by using the relation $a = te / tr$). We would prefer to use two parameters that are not strongly related. This simplifies the model because it eliminates the need to account for any correlation: Two parameters that are not related can be modeled *independently*. As mentioned above, *te* and *tr* are related. This leaves us with *a* and either *te* or *tr* as the parameters to model.

The analysis of the reference workloads reveals accuracy to have a much greater correlation with execution time than with requested time. For all workloads, the correlation coefficient between accuracy and execution time is considerably greater than the correlation coefficient between accuracy and requested time, as shown in Table 8.

Workload	***r*(*a*,*te*)**	***r*(*a*,*tr*)**
ANL	0.5273	0.0772
CTC	0.7024	0.2651
KTH	0.5010	0.2098
SDSC	0.7398	0.3985

Table 8 – Correlation coefficient between accuracy and execution time *r*(*a*,*te*) and correlation coefficient between accuracy and requested time *r*(*a*,*tr*)

Figure 8 shows the accuracy *a* as a function of the execution time *te*. The figure groups the completed jobs in 100 equally-sized "buckets" according to their execution time. The jobs in a bucket have their accuracies averaged to produce the values plotted in Figure 8. Of course there is high variance in the execution time of the jobs grouped in a given bucket, but the average shows a trend for accuracy to grow with execution time. On the other hand, plotting accuracy *a* as a function of requested time *tr* (see Figure 9) doesn't reveal much correlation between these two parameters.

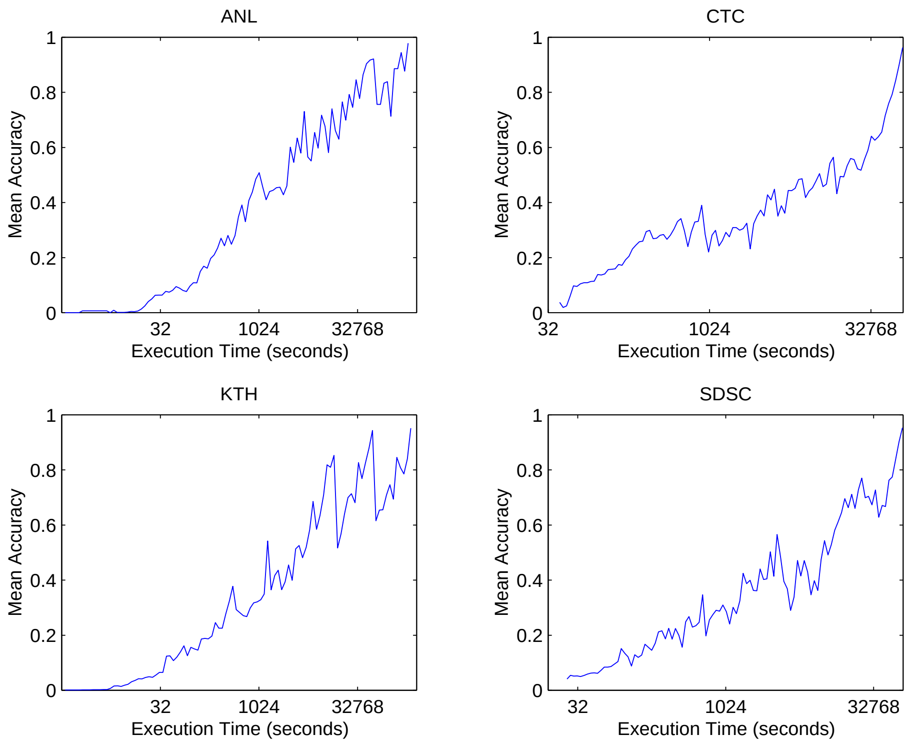

Figure 8 – Accuracy × Execution Time

We believe that the correlation between execution time and accuracy is related to the fact that *jobs fail, and thus execute for much less than the user expected*. In other words, we believe failed jobs to have in general poorer accuracy *a* than successful jobs. Unfortunately the logs do not contain information on whether a completed job failed or succeeded. But it is reasonable to imagine the longer the execution time; the more likely it is that the job succeeds. In fact, many failures happen at the beginning of the execution, while the job is setting up its environment. For example, many jobs open files in the beginning of their execution, and a misspelled filename could cause a failure. This argument would also help to explain the large number of jobs with very low accuracy (as we soon shall see, there are many jobs with poor accuracy in all reference workloads).

Since requested time shows little correlation to accuracy, we model these parameters independently. Execution time is then derived by using $te = tr \cdot a$. As a side note, we should also mention that requested time was easier to model (i.e., it yielded a better fit) than execution time. Although this was not our primary reason, it provided additional support to our choice of deriving execution time from the explicitly modeled accuracy and requested time.

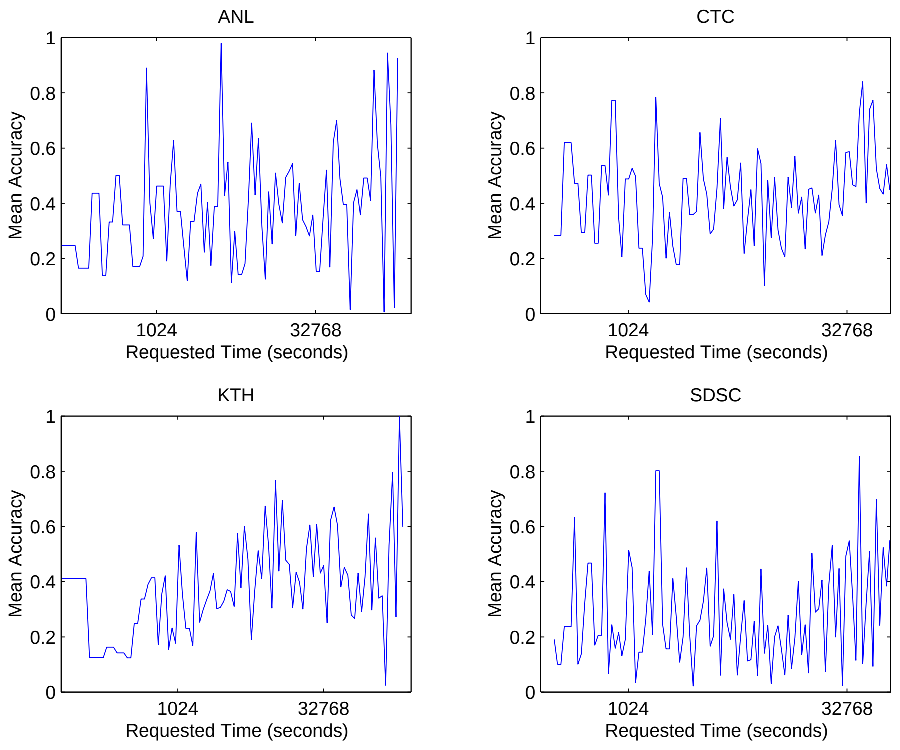

Figure 9 – Accuracy × Requested Time

7.1. Accuracy

Figure 10 shows how accuracy is distributed in the workload logs. Only completed jobs are considered. The figure groups the completed jobs in 100 equally-sized "buckets" according to their accuracy, and plots the fraction of jobs in each bucket. It is somewhat surprising to see how bad accuracy can be. The ANL workload, for example, has 27.82% of the requests with accuracy below 0.01 (i.e., these jobs used less than 1% of their requested time).

The number of jobs with low accuracy decreases as accuracy a grows (leveling off at around $a = 0.3$). This suggests the use of a distribution that favors small values and quickly decreases. We use the gamma distribution [8] to model such behavior. Table 9 shows the parameters obtained by fitting the observed accuracy to a gamma distribution through the method of maximum likelihood [8]. Figure 11 presents the observed distribution of accuracy and the corresponding model for all four reference workloads.

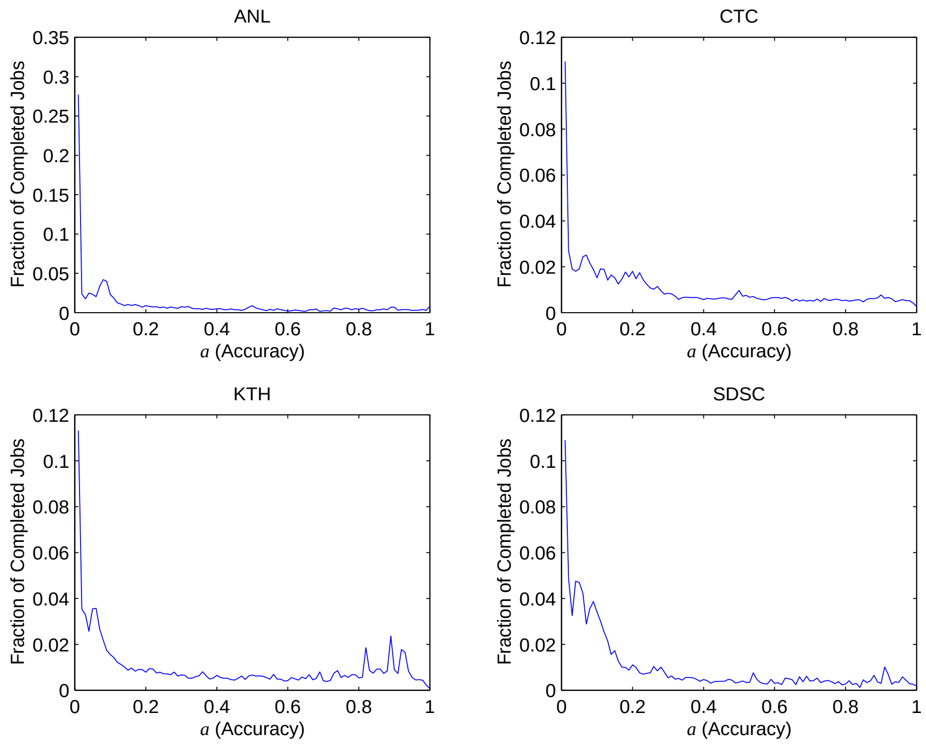

Figure 10 – Distribution of the accuracy *a* for the completed jobs

Workload	α_a	β_a
ANL	0.3779	1.0599
CTC	0.6743	0.7808
KTH	0.6153	1.0635
SDSC	0.5898	0.5793

Table 9 – α_a and β_a obtained by fitting accuracy to a gamma distribution

The gamma distribution parameters obtained by fitting the accuracy to the four reference workloads are somewhat similar to one another. ANL exhibits the smallest α value. This is because ANL presents a much higher number of jobs with very poor accuracy. Such a fact indicates that the distribution of accuracy can vary somewhat from site to site in practice.

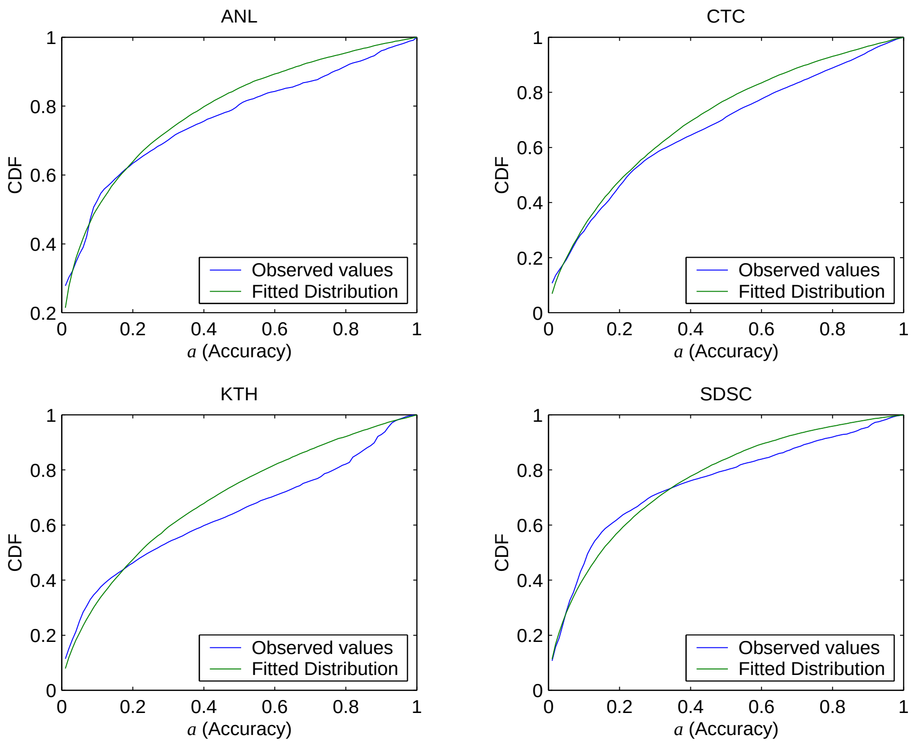

Figure 11 – Observed data and statistical model for accuracy

7.2. Requested Time

For all reference workloads, the uniform-log distribution provides a very good fit for request times, and thus it is the distribution used in our model. We use least-square linear regression to find χ_{tr} and ρ_{tr} for the different reference workloads. Table 10 displays such values and Figure 12 plots the observed requested times against the model. The similarity of the values obtained for χ_{tr} and ρ_{tr} across all four reference workloads suggests that the requested time doesn't vary widely across workloads.

Workload	χ_{tr}	ρ_{tr}
ANL	0.1146	-0.8579
CTC	0.0941	-0.7256
KTH	0.1035	-0.7313
SDSC	0.1032	-0.7027

Table 10 – χ_{tr} and ρ_{tr} obtained by fitting requested time to a uniform-log distribution

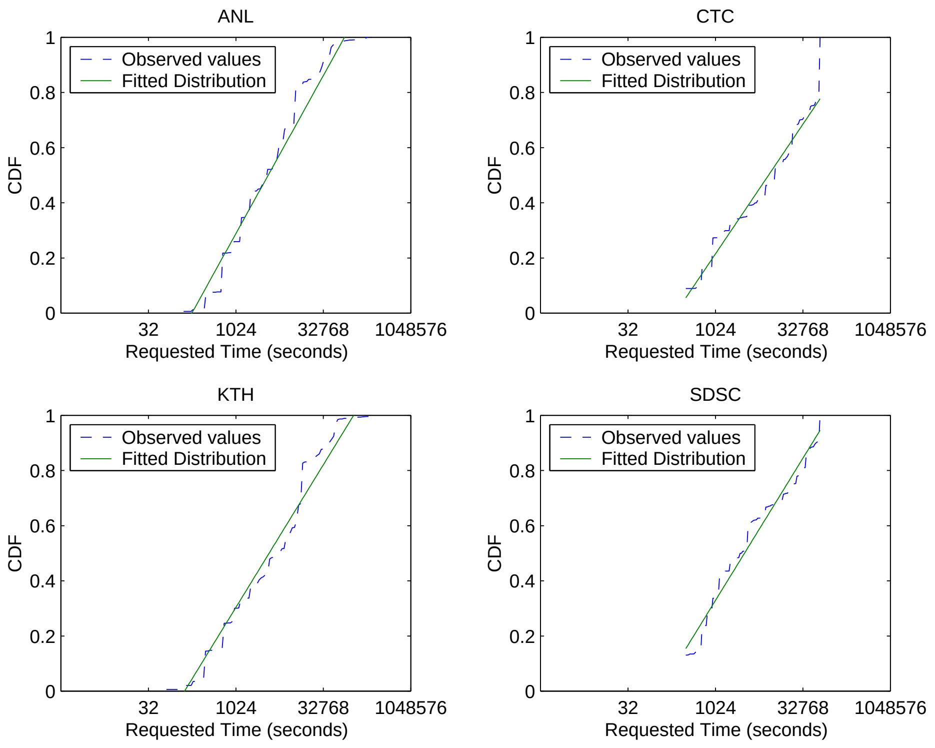

Figure 12 – Observed data and statistical model for requested time

8. Validation

The model herein described has been used to support research on scheduling parallel supercomputers [4] [6] [7]. A key advantage of using a model is that we can tune it to explore various scenarios of the workload space, therefore identifying when a given solution is expected to perform well (or poorly, for that matter). We can also tune the model to produce a workload whose features resemble an available workload log, therefore allowing us to validate the workload regarding that workload. We have performed such validation for the logs described in Table 1 with very good results [4] [6] [7]. We are in the process on validating our model against other logs.

9. Summary

This paper presents a supercomputer workload model that improves upon the state of the art in several aspects. First and foremost, our model covers two aspects of the supercomputer job that have not been modeled yet. More precisely, our model addresses requested time (and its relation with execution time) and the possibility of job cancellation. Second, our model for arrival instant is also novel by taking into account the seasonal workday cycle. Third, a user survey we conducted allowed us to shed some light on an old discussion happening within the community, namely: whether the prevalence of jobs with power-of-2 partition sizes is intrinsic of the workload [15], or an artifact of behavioral inertial and poor submission interface design [10].

We should also point out three interesting insights that occurred during this modeling effort. First, the pattern of job arrival within the day show significant difference from site to site, being the hardest characteristic to generalized (among those covered in this work). Second, there seems to be no intrinsic algorithmic reason for the prevalence of jobs with power-of-2 partition size, but there is no sign of change in that either. As long as the process of selecting the job's partition size (today in charge of the user) remains unchanged, we do not expect to see a change in the dominance of power-of-2 jobs. Third, there is a strong correlation between short execution time and poor request accuracy (the accuracy is the fraction of the requested time that was indeed used by the job). We believe that such correlation is related to the fact that jobs sometimes fail, and thus terminate much sooner than the user expected. In other words, we believe failed jobs to have in general poorer accuracy than successful jobs.

Of course, this is not to suggest that the modeling of supercomputer workloads is a solved problem. There is still much work to be done. Topics that need further research include priorities, user behavior and its feedback effect, and memory requirements [12].

Acknowledgments

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. Finally, thanks to the members of the AppLeS group, Keith Marzullo, Reagan Moore, Rich Wolski, Allen Downey, and Warren Smith for the insightful discussions and valuable suggestions.

References

[1] M. Calzarossa and G. Serazzi. *A characterization of the variation in time of workload arrival patterns*. IEEE Transactions on Computers, vol. C-34, (no.2), Feb. 1985. p.156-62.

[2] M. Calzarossa and G. Serazzi. *Workload Characterization: A Survey*. Proceedings of the IEEE, vol. 81, no. 8, pp. 1136-1150, August 1993.

[3] 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

[4] Walfredo Cirne. *Using Moldability to Improve the Performance of Supercomputer Jobs.* Ph.D. Thesis, University of California San Diego, Feb 2001. http://walfredo.dsc.ufpb.br/publications/thesis.pdf

[5] Walfredo Cirne and Francine Berman. *A Model for Moldable Supercomputer Jobs*. In Proceedings of the IPDPS 2001 - International Parallel and Distributed Processing Symposium, April 2001. http://walfredo.dsc.ufpb.br/resume.html#publications

[6] Walfredo Cirne and Francine Berman. *Using Moldability to Improve the Performance of Supercomputer Jobs*. Submitted to Publication, 2001. http://walfredo.dsc.ufpb.br/resume.html#publications

[7] Walfredo Cirne and Francine Berman. *When the Herd is Smart: The Emergent Behavior of SA*. In Preparation.

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

[9] 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/

[10] 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/

[11] 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/

[12] 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

[13] Dror Feitelson and Bill Nitzberg. Job characteristics of a production parallel scientific workload on the NASA A*mes 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

[14] 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

[15] 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

[16] Dror Feitelson and A. Mu'alem Weil. *Utilization and predictability in scheduling the IBM SP2 with backfilling*. In 12th Intl. Parallel Processing Symp., pp. 542-546, Apr 1998. http://www.cs.huji.ac.il/~feit/pub.html

[17] 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.

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

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

[20] 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.

[21] 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.

[22] D. Zotkin and P. Keleher. *Job-Length Estimation and Performance in Backfilling Schedulers*. 8th International Symposium on High Performance Distributed Computing (HPDC'99), Redondo Beach, California, USA, 3-6 Aug 1999.