Metrics for Parallel Job Scheduling and their Convergence

Dror G. Feitelson
School of Computer Science and Engineering
The Hebrew University, 91904 Jerusalem, Israel
http://www.huji.ac.il/~feit

Abstract

The arrival process of jobs submitted to a parallel system is bursty, leading to fluctuations in the load at many time scales. In particular, rare events of extreme load may occur. Such events lead to an increase in the standard deviation of performance metrics, and thus delay the convergence of simulations used to evaluate the scheduling. Different performance metrics have been proposed in an effort to reduce this variability, and indeed display different rates of convergence. However, there is no single metric that outperforms the others under all conditions. Rather, the convergence of different metrics depends on the system being studied.

1 Introduction

It has long been recognized that the performance of computer systems depends not only on their design and implementation, but also on the workload to which they are subjected. But the results may also depend on the *metric* being used for the evaluation. In some cases interactions may occur between the metric and certain characteristics of the system, leading to results that actually depend on the metric being used [21]. In this paper we concentrate on another effect, whereby some metrics converge more rapidly than others.

The conventional methodology of simulating computer systems calls for continuing the simulation until the desired relative precision is achieved with the desired level of confidence [22]. The relative precision reflects the size of the confidence interval relative to the estimated value. The confidence level is a statistical statement regarding the probability that the actual value we are trying to estimate actually resides within the confidence interval. Put together, the calculation is based on the ratio of the standard deviation of the performance metric to its mean, multiplied by some factor that takes the statistical properties of the simulation into account.

The standard deviation measures the divergence of individual measurements from the mean. Due to the averaging over multiple measurements, it tends to shrink as the simulation is continued. This leads to the conventional wisdom that any desired relative precision and level of confidence can be achieved by running the simulation for long enough. This conventional wisdom has been challenged lately with the realization that workloads that are characterized by heavy tailed distributions may prevent the simulation from reaching a steady state [4]. But even if the simulation does not diverge, it may take a long time to reach the desired relative precision. Moreover, the relative precision may not improve monotonically as the simulation is extended.

The conventional way to deal with these problems is to employ advanced statistical techniques for variance reduction. An alternative is to use performance metrics that are more robust in the face of a fluctuating workload. Indeed, several different performance metrics have been proposed for the evaluation of parallel job schedulers, with the goal of reducing the susceptibility to being affected by extreme workload conditions. We compare the convergence properties of these metrics, and evaluate the degree to which they achieve this goal.

2 Variability in Workloads

The root cause for convergence problems is variability in the workloads. We therefore start by characterizing the variability in the runtimes and arrivals of workloads observed on different systems, and in models based on them.

2.1 The Distribution of Job Runtimes

We begin by collecting some data about the runtime distributions of jobs executed on large scale parallel supercomputers. This is based on the following logs, which are available on-line from the Parallel Workloads Archive (www.cs.huji.ac.il/labs/parallel/workload/):

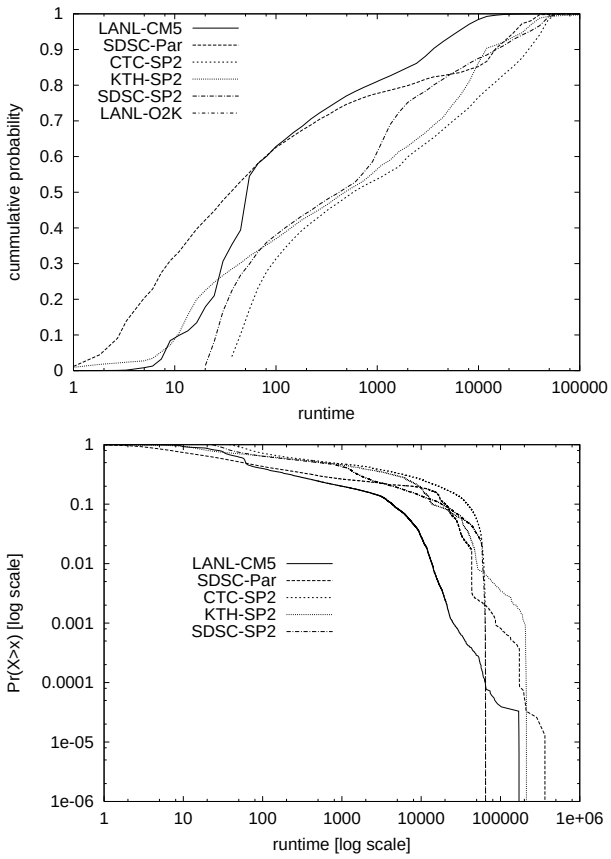

Figure 1: *Cumulative distribution of job runtimes for different systems, and log-log complementary distribution plots.*

LANL-CM5: The Los Alamos National Lab 1024-node CM-5 (201387 jobs, 10/1994 to 9/1996)

SDSC-Par: The San-Diego Supercomputer Center 416-node Intel Paragon (115595 jobs, 1/1995 to 12/1996)

CTC-SP2: The Cornell theory Center 512-node IBM SP2 (79296 jobs, 7/1996 to 5/1997)

KTH-SP2: The Swedish Royal Institute of Technology 100-node IBM SP2 (28490 jobs, 10/1996 to 8/1997)

SDSC-SP2: The San-Diego Supercomputer Center 128-node IBM SP2 (67665 jobs, 4/1998 to 4/2000)

LANL-O2K: The Los Alamos National Lab 2048-node Origin 2000 cluster (122233 jobs, 12/1999 to 4/2000)

Whenever the logs contain data about jobs that did not complete execution successfully, this data was discarded.

Regrettably, it is not clear that this data accurately represents real distributions of job runtimes. One problem is that most sites have limits on the allowed length of jobs, and these limits can be pretty low during the working hours of weekdays (e.g. 4 hours or 12 hours). Users that need to run very long jobs therefore resort to making a checkpoint whenever they run out of time, and then restarting the job. Such behavior appears as a sequence of short jobs in the data rather than a single long job.

system	mean	median	std dev	CV
LANL-CM5	1232.21	62.00	3268.29	2.65
SDSC-Par	4223.77	43.00	10545.10	2.50
CTC-SP2	9580.50	705.00	16388.91	1.71
KTH-SP2	6146.68	583.00	14483.63	2.36
SDSC-SP2	5481.20	521.00	12776.01	2.33
LANL-O2K	1965.26	23.20	7203.51	3.67

Table 1: *Statistics of runtimes in different workloads.*

The conjecture that this is the case is strengthened by observations of the cumulative distribution function of the job runtimes on the different systems: they all seem to have about the same upper bound, with little if any tail (Fig. 1). This bound, at about 50000 seconds (14 hours) looks like an administrative constraint based on daily work cycles. To verify the absence of a heavy tail we also create log-log complementary distribution plots (Fig. 1). Again, the evidence is against such tails as there are no linear regions that spans several orders of magnitude. Note that the limited and seemingly non-dispersive distributions we see are in stark contrast to data from interactive workstations, where the tail of the distribution is indeed heavy and follows a Pareto distribution [17, 12]. There is no reason to expect such long jobs to be absent on supercomputers — on the contrary, sans administrative restrictions, long jobs may be expected to dominate the workload.

Finally, we note that even if the distribution of job runtimes does not conform to the formal definition of having a heavy tail, it is nonetheless very skewed, and its mean is much higher than its median (Table 1). As we show below, this is enough to cause significant problems in the analysis and evaluation of job scheduling algorithms.

2.2 Workload Models

Several models have been proposed in the literature for the distribution of job runtimes.

Traditionally the observation that job runtimes have a coefficient of variation larger than 1 motivated the use of a hyperexponential distribution rather than an exponential distribution [23]. However, crafting a hyperexponential distribution so as to match the first two moments of the target distribution may create a distribution with the wrong shape, resulting in misleading evaluations [16].

Jann et al. have improved on this by using a hyper Er-

model	mean	median	std dev	CV
Jann	11547.19	794.12	18616.14	1.61
Feitelson	2700.89	81.11	8786.63	3.25
Downey	638.54	18.84	2107.36	3.30
Lublin	1668.34	18.00	6824.61	4.09

Table 2: *Statistics of runtimes in different models.*

lang distribution and matching the first three moments of the data in the CTC-SP2 log [15]. 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. The result was a model with a large number of parameters (about 40) that closely mimics the original data.

One problem with creating distributions based on moments is that with skewed distributions the estimation of high-order moments (and even the second moment) is very sensitive to the values of the few highest values sampled [6]. This has lead to the proposed use of distributions based on direct observations of the CDF and goodness of fit metrics, in lieu of trying to match the moments.

Feitelson used a two-stage or three-stage hyperexponential distribution, choosing the parameters so that the CDF "looked right" (that is, similar to that in various logs) [8]. To accommodate the slight correlation observed between runtime and the degree of parallelism, the probability of using each exponential depends on the degree of parallelism.

Downey has proposed the log uniform distribution based on observation of the SDSC-Par log [5]. 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.

Lublin proposed a hyper Gamma distribution, based on the CTC-SP2, KTH-SP2, and SDSC-Par logs [19]. This distribution requires 5 parameters: two for each Gamma distribution, and the probability of selecting one or the other. This probability is modified based on the degree of parallelism as was done by Feitelson. The parameters of the Gamma distributions were selected using an Expectation-Maximization algorithm and goodness of fit metrics.

Programs for generating workloads according to these models are available on-line at the Parallel Workloads Archive. Generating such workloads and calculating their statistics and running average leads to the results shown in Fig. 2 and Table 2. The models are much more stable than any of the real workloads, and quickly converge to a stable average runtime. However, they are just as skewed as the original workloads.

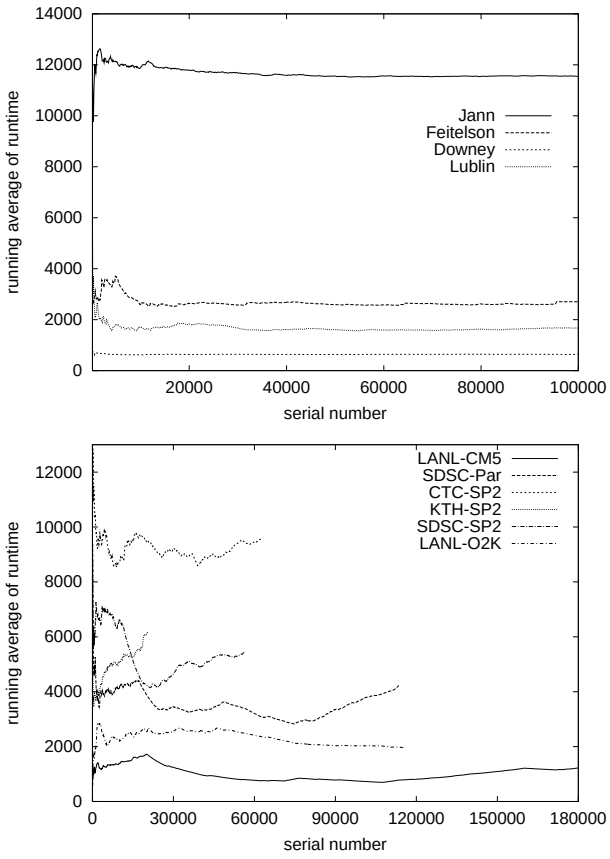

Figure 2: *Running average of mean job runtime from different models. Data for logs is shown for comparison.*

2.3 Burstiness of Arrivals

The arrival process of parallel jobs has received much less analysis than the distribution of job runtimes. Most studies simply make the assumption of a Poisson process, with exponentially distributed interarrival times. A notable exception is the Jann model, which creates a model of interarrival times that parallels the model of run times [15].

Burstiness, or self-similarity, in the arrival process can lead to large fluctuations in load, just like fat-tailed runtime distributions. To check the degree of burstiness we first plot the arrival process of 5 logs, using aggregation in 10-minute buckets. The results are shown in Fig. 3, and are indeed bursty. Similar results are observed for the arrival of processes.

In order to test for self-similarity, we use the eyeball method of plotting the same data using different levels of aggregation. The righthand side of Fig. 3 shows results at 5 decimal orders of magnitude for the SDSC-Par log, and seems to indicate that self-similarity is present. Somewhat surprisingly, job arrivals even show some burstiness at the very low scale of 36 seconds. Daily cycles are barely dis-

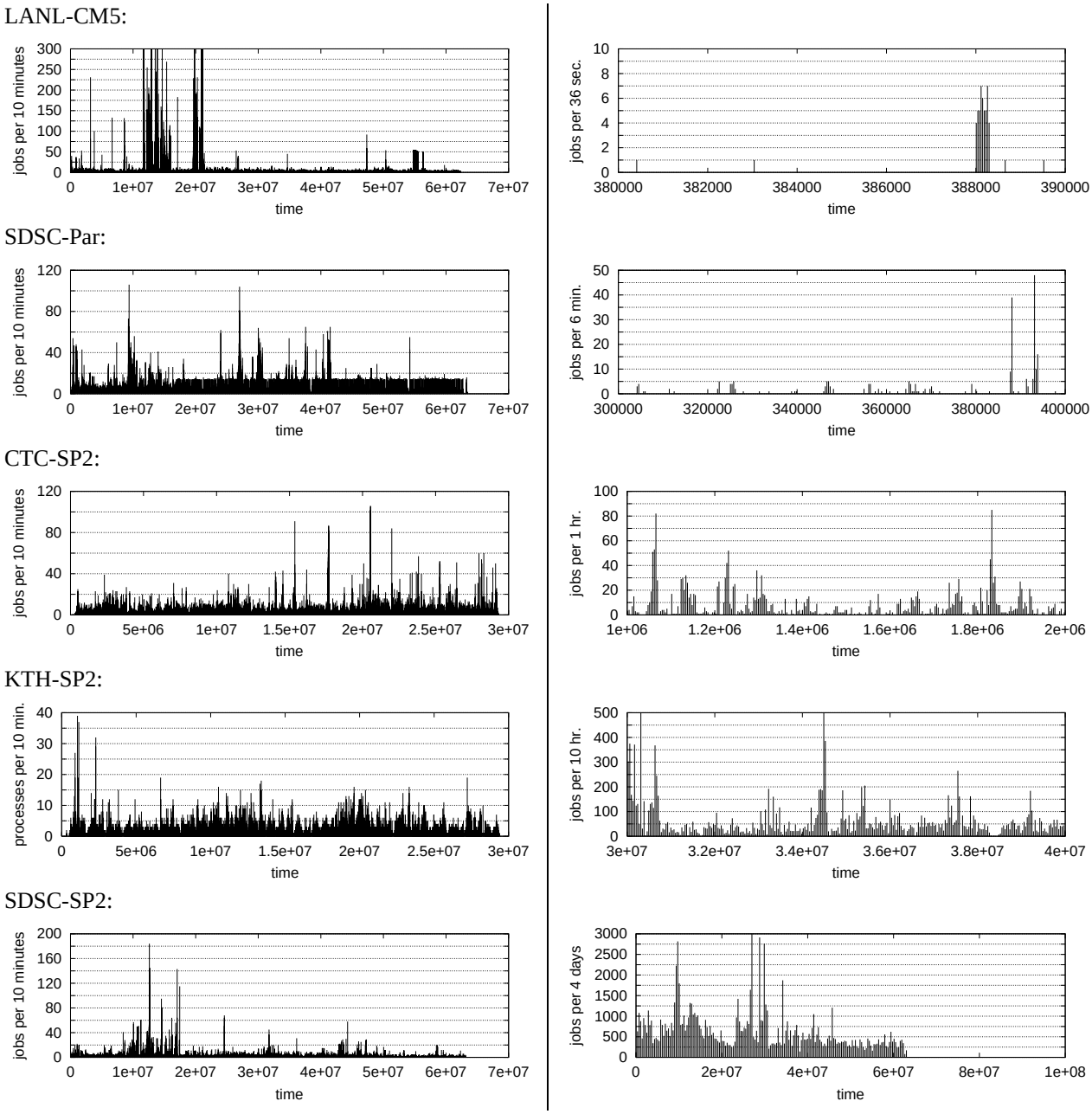

Figure 3: *Left: arrival pattern of jobs for five logs. Right: Burstiness of arrivals to SDSC-Par at different time scales.*

cernible in the middle plot.

Rather than focusing on the phenomenon of self similarity, we are interested in the distributions describing the arrival process. Self similarity and burstiness seem to imply that there is a non-negligible probability that many jobs will arrive practically at once. In other words, we expect the distribution of the number of jobs (or processes) arriving per unit time to have a fat tail. To check this, we plot log-log complementary distribution plots of this distribution, at different levels of aggregation. This means, in essence, that different time units are used (e.g. jobs per 6 minutes, jobs per hour, and jobs per 10 hours). Crovella et al. have shown that with heavy-tailed distributions these plots should be linear with the same slope, whereas if the tail is not heavy (and the variance is finite) the slope should become steeper with higher levels of aggregation, and the plots will seem to converge [3]. Our results are mixed (Fig. 4). For some workloads the plots do indeed seem to be parallel (albeit over a smaller scale than for the web traffic data of [3]). For others they seem to converge.

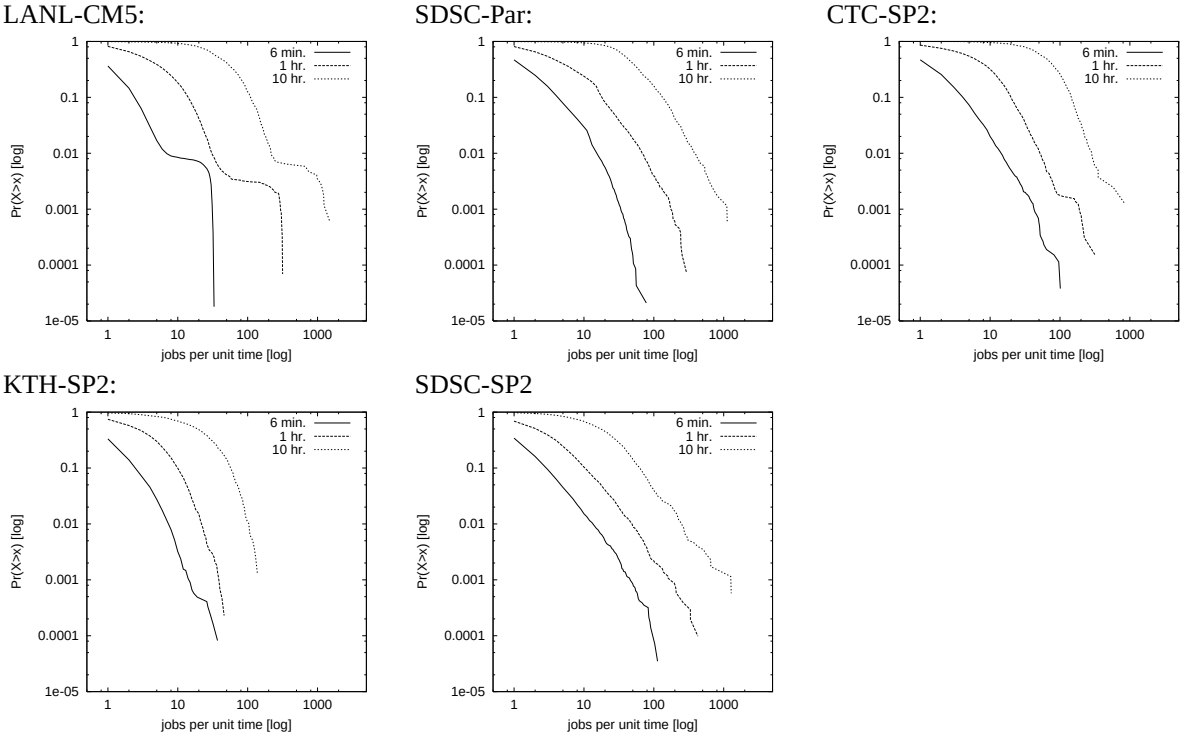

Figure 4: *Log-log complementary distribution plots of job arrivals with different aggregation levels.*

In conclusion, a heavy-tailed distribution cannot be postulated, despite the fact that the arrival process is bursty.

3 The Effect of Workload on Convergence

The skewed nature of dispersive distributions and the non-negligible probability of sampling very high values have significant implications on systems. For example, Harchol-Balter and Downey have shown that when the distribution of job runtimes has a heavy tail, migration for load balancing can be very beneficial [12]. This is based on the fact that a small number of jobs use more CPU time than all the others; the benefits come from identifying these jobs and migrating only them. This contradicts evaluations based on a more moderate distribution, in which migration did not lead to significant benefits [7]. In another work, Harchol-Balter et al. have shown that when job sizes are heavy tailed it is beneficial to distribute jobs among servers according to size, thus effectively serving the short jobs on a dedicated set of servers that are unaffected by the long jobs from the tail of the distribution [11]. But what happens with the bounded runtime distribution observed on parallel systems?

As an initial check of how simulations of parallel systems behave, we simulated two versions of backfilling when operating on the Jann workload[1]. The load was adjusted to 0.75 by modifying all interarrival times by a constant factor. The algorithms are EASY backfilling, in which short jobs are allowed to move ahead provided they do not delay the first job in the queue [18], and a more conservative version of backfilling, in which jobs move forward provided they do not delay any previously scheduled job [21]. The metric used was the mean response time. Confidence intervals are calculated using the batch means method [14], with a batch size of 5000 job completions (matching the recommendation of MacDougall for a CV larger than 1 and high load [20]).

	1st 5% xing	2nd 5% xing	million
EASY	16427.23	16774.40	16865.80
cons	17727.68	18154.75	18313.90

Table 3: *Mean response time selected by confidence interval criteria.*

The results are shown in Fig. 5. While the mean response time does seem to converge, there are relatively

[1] This model has a problem in that the program occasionally does not manage to solve the equations used for the distributions, and prints an error message. This happened 20 times when generating a workload of 1000000 jobs, and was ignored. in an additional 43 jobs the program created an infinite runtime. These jobs were simply replaced by a clone of the previous job.

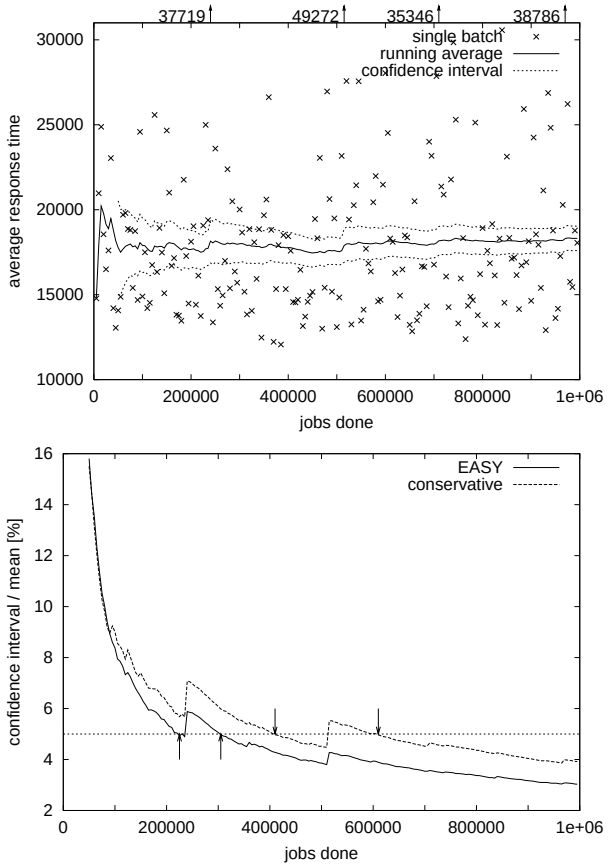

Figure 5: *Backfilling using the Jann model as the workload. Top: convergence of mean response time for conservative backfilling. Bottom: behavior of the confidence interval.*

big jerks even after half a million job terminations. And it is not clear that additional jerks will not occur even after a million jobs were simulated. Note that this represents more or less the whole lifetime of a large-scale parallel machine. Thus it is not clear what such simulation results mean with relation to the use of new machines.

The 95% confidence interval suffers from similar jerks. As a result the size of the confidence interval is not monotonically decreasing. Using the common methodology of terminating the simulation when the confidence interval becomes smaller than say 5% of the mean [22] would lead to different results for the first and second times such a crossing is made, though the "final" result — after simulating 1000000 jobs — is within the confidence interval in both cases (Tab. 3). However, there are points in the simulation (e.g. after the termination of 475000 jobs) where the confidence interval is smaller than 5% of the mean, and does *not* contain the final value obtained after simulating a million jobs. A detailed coverage analysis is needed to determine whether this occurs more or less than 95% of the time, but it is troubling that it was so easy to find an example.

We note in passing that warmup is not a problem in our setting. The problem leading to the variable and jerky simulation results shown above is not one of initialization, but a problem of real variability and burstiness in the workload, due to its dispersive distributions. Indeed, Crovella and Lipsky have suggested that such situations be handled by explicitly noting the time horizon for which the results are valid [4]. Another approach is to use the techniques of rare event simulation [13]. We leave the detailed study of such optimizations in the context of scheduling with dispersive distributions for future research.

4 Performance Metrics for Job Scheduling

As we saw above, it may take the average response time a very long time to converge when the simulated job stream being scheduled is based on dispersive distributions. Do other metrics converge more quickly? And indeed, what is the most meaningful metric?

The first metric we deal with is the response time. We define "response time" to mean the total wallclock time from the instant at which the job is submitted to the system, until it finishes its run. This can be divided into two components: the running time T_r, during which the job is actually running in parallel on multiple processing nodes, and the waiting time T_w, in which it is waiting to be scheduled or for some event such as I/O. The waiting time itself can also be used as a metric, based on the assumption that T_r does not depend on the scheduling.

Obviously, a lower bound on the response time of a given job is its running time. As the runtimes of jobs have a very large variance, so must the response time. It has therefore been suggested that a better metric may be the slowdown (also called "expansion factor"), which is the response time normalized by the running time:

$$\text{slowdown} = \frac{T_w + T_r}{T_r}$$

Thus if a job takes twice as long to run due to system load, it suffers from a slowdown factor of 2, etc. This is expected to reduce the extreme values associated with very long jobs, because even if a week-long job is delayed for a whole year the slowdown is only a factor of 50. Moreover, slowdown is widely perceived as better matching user expectations that a job's response time will be proportional to its running time. Indeed, 30 years ago Brinch Hansen already suggested that slowdowns be used to prioritize jobs for scheduling [1].

The problem with the slowdown metric is that it overemphasizes the importance of very short jobs. For example, a job taking 100 ms that is delayed for 10 minutes suffers from a slowdown of 6000, whereas a 10-second job delayed by the same 10 minutes has a slowdown of only 60. From a user's perspective, both are probably annoying to similar degrees, but the slowdown metric gives the shorter job an extremely high score, because the running time appears in the denominator.

To avoid such effects, Feitelson et al. have suggested the "bounded-slowdown" metric [9]. The difference is that for short jobs, this measures the slowdown relative to some "interactive threshold", rather than relative to the actual runtime. Denoting this threshold by τ, the definition is

$$\text{bounded-slowdown} = \max\left\{\frac{T_w + T_r}{\max\{T_r, \tau\}}, 1\right\}$$

The behavior of this metric obviously depends on the choice of τ. In the simulations below, we check three values: 10 seconds, one minute, and 10 minutes.

The fact that the definition of slowdown (and bounded slowdown) is based on a job's running time leads to new problems. On one hand, it makes practically equivalent jobs look different. On the other hand, it encourages the system to make the jobs run longer!

Zotkin and Keleher have noted that jobs that do the same amount of work with the same response time may lead to different slowdown results due to their shape (that is, ratio of processors to time). For example, a job that runs immediately on one processor for 100 seconds has a slowdown of 1, whereas a job that is delayed for 90 seconds and then runs for an additional 10 seconds on 10 processors (thus utilizing the same 100 processor-seconds as the first job, and finishing with the same 100 seconds response time) has a slowdown of 10. This lead them to suggest a new metric, which we shall call "per-processor slowdown" [24]:

$$\text{pp-slowdown} = \max\left\{\frac{T_w + T_r}{P \cdot \max\{T_r, \tau\}}, 1\right\}$$

where P is the number of processors used by the job. The name derives from the fact that this has the units of $1/P$, and divides the original bounded slowdown by the number of processors used; it can be understood as a further normalization of the slowdown metric, for the putative case where the job runs on a single processor. In terms of the above example, this normalizes the delayed 10-processor job to the undelayed single-processor job, so both now have a pp-slowdown of 1.

A possible counter argument is that if a user makes the effort to parallelize a program, and runs it on more processors, he actually expects it to finish faster. Therefore a parallel program that is delayed is not equivalent to a serial one that runs immediately. But what about cases in which the number of processors used is chosen automatically by the scheduler? Cirne and Berman observe that in this scenario the system can improve its slowdown metric by choosing to use fewer processors: the job will then probably start running sooner, and even if not, it will run for longer. As a result, the ratio of the response time to the running time will be smaller, even if the response time itself is larger [2].

Their solution to this problem is to do away with the use of slowdowns altogether, and stick with response times. They then go on to suggest the use of a geometric mean rather than an arithmetic mean to calculate the average response time, with the goal of reducing the effect of excessively long jobs. Notably, a similar argument is used to justify the use of a geometric mean in calculating the score of SPEC benchmarks. However, it has also been noted that the geometric mean may order results differently from the sum of the represented times [10]. In other words, given two sets of measurement A and B, it is possible that the sum of the measurements in A is smaller, but their geometric mean is larger. Obviously, the arithmetic mean does not suffer from such inversions.

5 Convergence Results for Different Metrics

To compare the behavior of the different metrics, we ran long simulations as in Section 3, and observe the way in which the different metrics converge. This simulation again uses EASY and conservative backfilling on one million jobs generated according to the Jann model.

The results are shown in Fig. 6. The most important observation from these graphs does not concern the convergence, but rather the ranking produced by the different metrics: the response time and wait time metrics give lower (better) scores to EASY, whereas slowdown-based metrics give lower (better) scores to conservative. Using a geometric mean of response times is in the middle: it asserts that they are both the same. The interactions between the scheduling algorithms and the workloads that lead to these divergent results are interesting in their own right [21], but lie beyond the scope of the current paper; here we are interested in the convergence properties.

To the naked eye, all the graphs seem to be jerky in similar degrees. The slowdown graph is distinguished by the fact that the jerks are in different places than in other graphs. For slowdown they result from short jobs that get delayed, whereas for bounded slowdown they result from long jobs (these jerks actually also appear in the slowdown curve, but are less prominent there). A some-

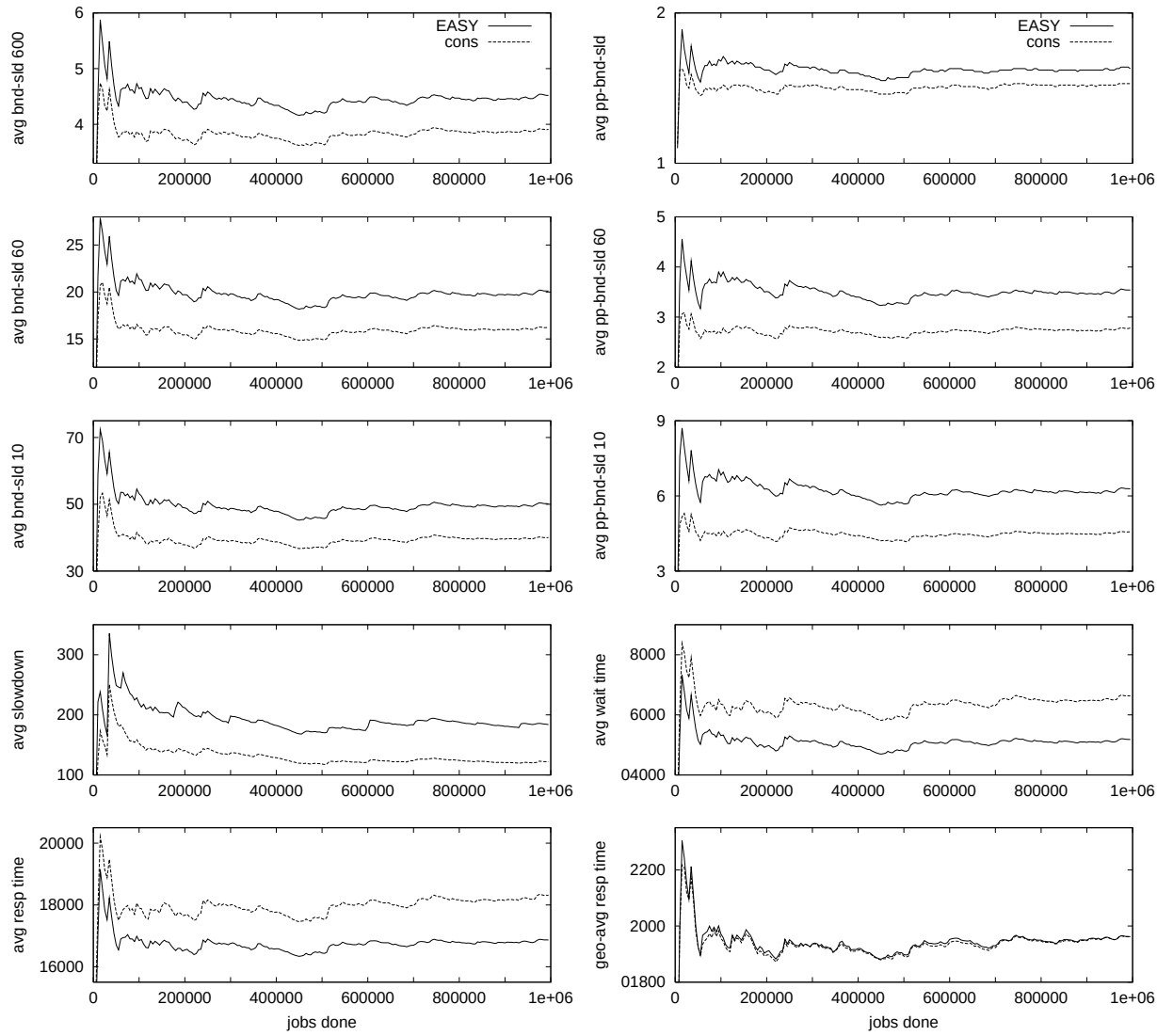

Figure 6: *Convergence of different metrics during long simulation.*

what surprising result is that the shape of the graphs for bounded slowdown are practically identical to that of the response time! The explanation is that as the value of τ grows larger, more and more jobs are covered — in the Jann model, specifically, about 48% of the jobs are shorter than 10 minutes. For these jobs, the definition of bounded slowdown is just the response time divided by a constant. As these are the higher values, they dominate the shape of the curve. Moreover, comparison with the graph for wait time shows that the wait time is indeed the dominant factor in these cases.

To quantify the rate of convergence, we plot the size of the confidence intervals calculated using the batch means approach. The results for the SDSC-SP2 and CTC-SP2 logs are shown in Fig. 7. It seems that the slowest metrics to converge are either the slowdown or the geometric mean of response times. The arithmetic mean converges rather quickly. Bounded slowdown is in the middle, and is not very sensitive to the threshold value. Per-processor slowdown is much more sensitive, and provides better convergence as the threshold value is increased. It should be stressed that the differences are very significant: for some metrics, the confidence interval size is more than 10% of the mean even after the simulation of a million jobs. For many it would require unrealistically long simulations to get within 5% of the mean.

Finally, we note that there is another variable that may influence the convergence: the scheduling algorithm itself. Specifically, part of the problem with queue-based scheduling algorithms such as backfilling is that jobs get

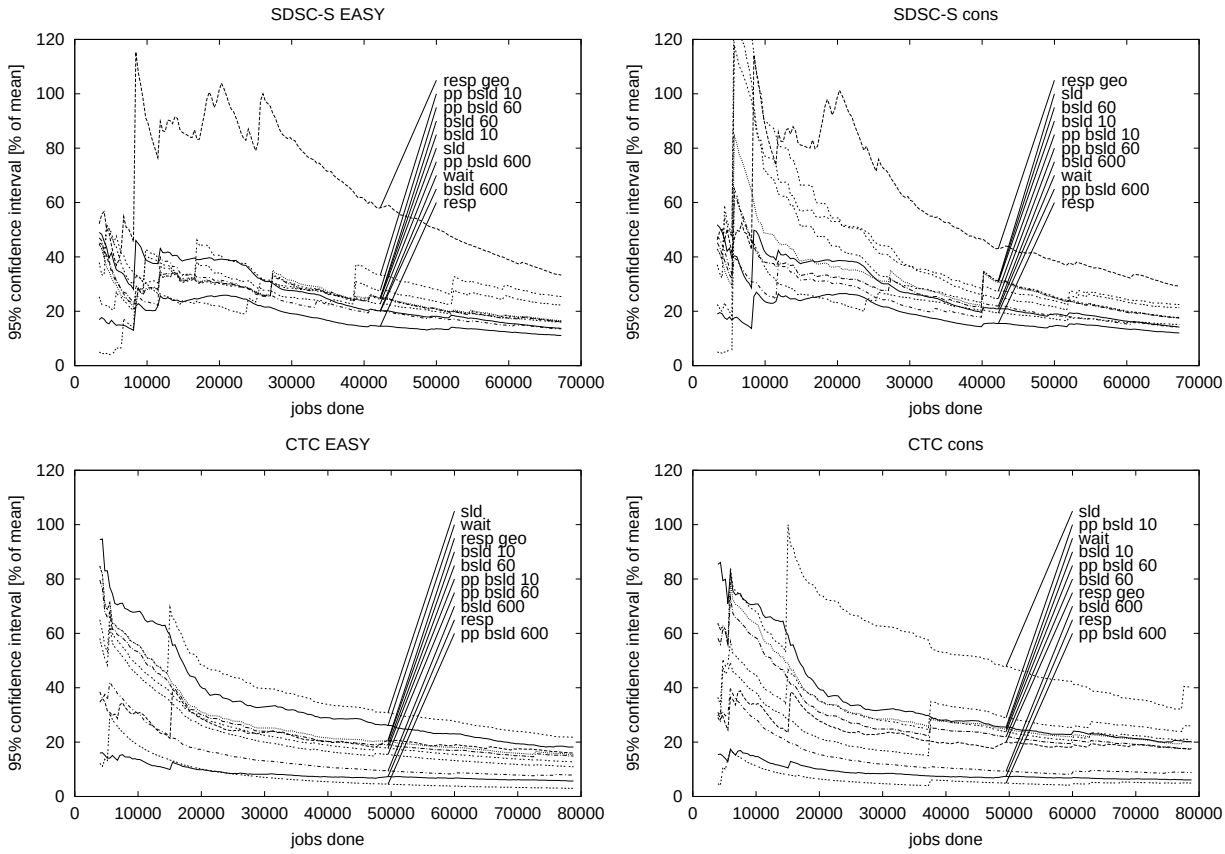

Figure 7: *Convergence of different metrics for real logs.*

held up in the queue. They then tend to pile up, leading to jerks in the various metrics. But with time slicing, jobs don't affect each other as much, and as a result smoother convergence can be expected. To check this conjecture, we used the same SDSC-SP2 log as an input to a simulation of gang scheduling, with two different time quanta: one minute and 10 minutes. The results are shown in Fig. 8. With short time quanta, this is indeed similar to processor sharing, and leads to smoother convergence. In addition, the normalization inherent in the slowdown-based schemes causes them to converge significantly faster than the un-normalized response-time based metrics. However, the absolute size of the confidence interval is not any smaller than for the non-preemptive backfilling schemes. Things improve somewhat for the longer time quantum.

6 Conclusions

Contrary to common belief, the distribution of job runtimes on parallel supercomputers is not fat tailed, possibly due to the widespread use of administrative limitations. However, the distribution of load on these machines is indeed fat-tailed (and possibly even heavy tailed) due to the burstiness of arrivals. This means that occasionally the load becomes excessively high, enough to counterweigh the lower load between high-load events.

The existence of high-load events means that it is hard or impossible to converge to a stable result. The rate of convergence depends on the metrics being used, and on the nature of the system. For example, with non-preemptive scheduling it seems that using the well-known response-time metric leads to faster convergence, whereas with preemptive scheduling it seems that slowdown-based metrics converge faster. Plain slowdown displays very slow convergence in some cases, indicating that some version of bounded slowdown is preferable; within this family of metrics, high thresholds lead to faster convergence. As for using the geometric mean instead of the arithmetic mean of response times, this too suffers from slow convergence in some cases.

This paper has served to showcase the difficulties resulting from the complexities of real workloads. Much work remains to be done, both in terms of further characterization, analysis, and modeling of workloads, and in terms of understanding their effects on system perfor-

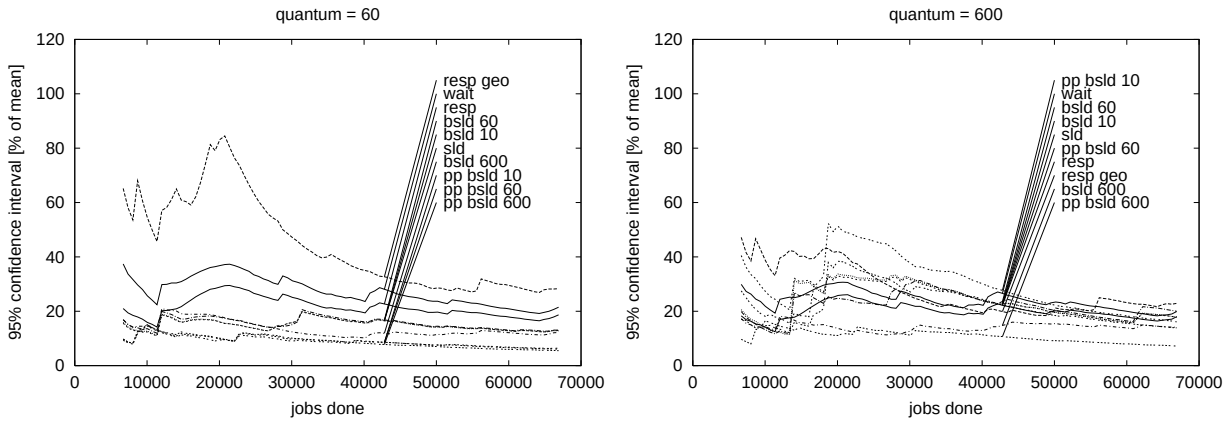

Figure 8: *Convergence of different metrics for gang scheduling of the SDSC-SP2 log.*

mance. Of particular interest is the identification and characterization of cases in which the relative performance of different systems depends on the workloads and the metrics being used.

Acknowledgements

This research was supported by the Israel Science Foundation founded by the Israel Academy of Sciences and Humanities. This research could not be conducted without the wealth of data available on-line at the Parallel Workloads Archive. The following acknowledgements are from there. The workload log from the LANL CM-5 was graciously provided by Curt Canada, who also helped with background information and interpretation. The workload log from the SDSC Paragon was graciously provided by Reagan Moore and Allen Downey, who also helped with background information and interpretation. The workload log from the CTC SP2 was graciously provided by the Cornell Theory Center, a high-performance computing center at Cornell University, Ithaca, New York, USA. The workload log from the KTH SP2 was graciously provided by Lars Malinowsky, who also helped with background information and interpretation. The workload log from the SDSC SP2 was graciously provided by Victor Hazlewood of the HPC Systems group of the San Diego Supercomputer Center (SDSC), which is the leading-edge site of the National Partnership for Advanced Computational Infrastructure (NPACI), and is available from the NPACI JOBLOG repository at http://joblog.npaci.edu. The workload log from the LANL Origin 2000 was graciously provided by Fabrizio Petrini, who also helped with background information and interpretation. Thanks are also due to Joefon Jann, Allen Downey, and Uri Lublin for providing C programs that implement their models.

References

[1] P. Brinch Hansen, "*An analysis of response ratio scheduling*". In *IFIP Congress, Ljubljana*, pp. TA–3 150–154, Aug 1971.

[2] W. Cirne and F. Berman, "*Adaptive selection of partition size for supercomputer requests*". In *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph (eds.), pp. 187–207, Springer Verlag, 2000. Lect. Notes Comput. Sci. vol. 1911.

[3] M. E. Crovella and A. Bestavros, "*Self-similarity in world wide web traffic: evidence and possible causes*". In *SIGMETRICS Conf. Measurement & Modeling of Comput. Syst.*, pp. 160–169, May 1996.

[4] M. E. Crovella and L. Lipsky, "*Long-lasting transient conditions in simulations with heavy-tailed workloads*". In *Winter Simulation conf.*, Dec 1997.

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

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

[7] D. L. Eager, E. D. Lazowska, and J. Zahorjan, "*The limited performance benefits of migrating active processes for load sharing*". In *SIGMETRICS Conf. Measurement & Modeling of Comput. Syst.*, pp. 63–72, May 1988.

[8] 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.

[9] D. G. Feitelson, L. Rudolph, U. Schwiegelshohn, K. C. Sevcik, and P. Wong, "*Theory and practice in parallel job scheduling*". In *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph (eds.), pp. 1–34, Springer Verlag, 1997. Lect. Notes Comput. Sci. vol. 1291.

[10] R. Giladi and N. Ahituv, "*SPEC as a performance evaluation measure*". *Computer* **28(8)**, pp. 33–42, Aug 1995.

[11] M. Harchol-Balter, M. E. Crovella, and C. D. Murta, "*On choosing a task assignment policy for a distributed server system*". In *Computer Performance Evaluation*, R. Puigjaner, N. Savino, and B. Serra (eds.), pp. 231–242, Springer-Verlag, 1998.

[12] M. Harchol-Balter and A. B. Downey, "*Exploiting process lifetime distributions for dynamic load balancing*". *ACM Trans. Comput. Syst.* **15(3)**, pp. 253–285, Aug 1997.

[13] P. Heidelberger, "*Fast simulation of rare events in queueing and reliability models*". *ACM Trans. Modeling & Comput. Simulation* **5(1)**, pp. 43–85, Jan 1995.

[14] R. Jain, *The Art of Computer Systems Performance Analysis*. John Wiley & Sons, 1991.

[15] 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.

[16] 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.

[17] W. E. Leland and T. J. Ott, "*Load-balancing heuristics and process behavior*". In *SIGMETRICS Conf. Measurement & Modeling of Comput. Syst.*, pp. 54–69, 1986.

[18] 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.

[19] U. Lublin, *A Workload Model for Parallel Computer Systems*. Master's thesis, Hebrew University, 1999. (In Hebrew).

[20] M. H. MacDougall, *Simulating Computer Systems: Techniques and Tools*. MIT Press, 1987.

[21] A. W. Mu'alem and D. G. Feitelson, "*Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfilling*". *IEEE Trans. Parallel & Distributed Syst.* **12(6)**, Jun 2001.

[22] K. Pawlikowski, "*Steady-state simulation of queueing processes: a survey of problems and solutions*". *ACM Comput. Surv.* **22(2)**, pp. 123–170, Jun 1990.

[23] R. F. Rosin, "*Determining a computing center environment*". *Comm. ACM* **8(7)**, pp. 465–468, Jul 1965.

[24] D. Zotkin and P. J. Keleher, "*Job-length estimation and performance in backfilling schedulers*". In 8th *Intl. Symp. High Performance Distributed Comput.*, Aug 1999.