High-Resolution Analysis of Parallel Job Workloads

David Krakov Dror G. Feitelson

School of Computer Science and Engineering
The Hebrew University of Jerusalem
91904 Jerusalem, Israel

Abstract. Conventional evaluations of parallel job schedulers are based on simulating the outcome of using a new scheduler on an existing workload, as recorded in a log file. In order to check the scheduler's performance under diverse conditions, crude manipulations of the whole log are used. We suggest instead to perform a high-resolution analysis of the natural variability in conditions that occurs within each log. Specifically, we use a heatmap of jobs in the log, where the X axis is the load experienced by each job, and the Y axis is the job's performance. Such heatmaps show that the conventional reporting of average performance vs. average load is highly oversimplified. Using the heatmaps, we can see the joint distribution of performance and load, and use this to characterize and understand the system performance as recorded in the different logs. The same methodology can be applied to simulation results, enabling a better appreciation of different schedulers, and better comparisons between them.

1 Introduction

The performance of a computer system obviously depends on the workload it handles. Reliable performance evaluations therefore require the use of representative workloads. This means that the evaluation workload should not only have the same marginal distributions as the workloads that the system will have to handle in production use, but also the same correlations and internal structure. As a result, logs of real workloads are often used to drive simulations of new system designs, because such logs obviously contain all the structure found in real workloads. But replaying a log in a simulation only provides a single data point of performance for one workload.

The most common approach to obtaining data for different conditions is to manipulate the log, and run multiple simulations using multiple manipulated versions. As an alternative, we suggest to exploit the natural variability that is inherent in real workloads. For example, if we are interested in performance as a function of load, we can distinguish between high-load periods in the log and low-load periods in the log.

In developing this idea, we first partition the jobs in each log into a small number of classes, according to the load that they each experience. Following the

pioneering work of Rudolph and Smith [8], we find the number of jobs in each class and their average performance, in order to create a "bubbles plot" describing the performance as a function of the load. We analyze the characteristics of such plots, and suggest that a higher resolution may be desirable; in particular, the distribution of performance often has a long tail, and thus the average is not a good representative value. This leads to the idea of creating heatmaps that show the full distribution of performance vs. load. Applying this idea to existing logs reveals several phenomena that have not been known before. The heatmaps can also be used to compare the performance of simulated schedulers, and reveal interesting differences between the behavior of EASY and FCFS — and between simulation results and the behavior of the production schedulers as recorded in the original logs.

2 Evaluating Parallel Job Schedulers with Log-Based Simulations

Log-based simulations have emerged as the leading methodology for evaluating parallel job schedulers. The logs used are actually accounting logs, which contain data about all the jobs that ran on some large-scale machine during a certain period of time. Such logs are available from the Parallel Workloads Archive [7], where they are converted into the Standard Workload Format (SWF) [1]. This makes them easier to use, as the simulators needs to know how to parse only this one format.

Given a log, the simulator simulates job arrivals according to the timestamps in the log. As each job arrives, the simulated scheduler is notified of the number of processors it requires, and possibly also of the user's expectation regarding the runtime. It then decides whether the job should run immediately (in the simulated system), or be queued and run later. Job terminations are simulated based on the scheduling decisions of the simulated scheduler, together with the runtime data provided in the log. Finally some overall average performance metric is computed over all the jobs in the simulation, such as the average response time or the average slowdown. This can be associated with the average load (utilization) of the log.

The main problem with the methodology described thus far is that it provides a single data point: the average performance for the average load. But an important aspect of systems performance evaluation is often to check the system's performance under *different* load conditions, and in particular, how performance degrades with increased load. Given a single log, crude manipulations are typically used in order to change the load. These are

- Multiplying all arrival times by a constant, thus causing jobs to arrive at a faster rate and increasing the load, or causing them to arrive at a slower rate and decreasing the load. However, this also changes the daily cycle, potentially causing jobs that were supposed to terminate during the night to extend into the next day, or causing the peak arrival rate to occur at night.

An alternative approach that has essentially the same effect is to multiply all runtimes by a constant. This doesn't change the arrival pattern, but may cause jobs that were previously independent to clash with each other. Worse, it creates an artificial correlation between load and response time, which essentially invalidates the use of response time as a performance metric.

- Multiplying all job sizes (here meaning the number of processors they use) by a constant, and rounding to the nearest integer. This has three deficiencies. First, many jobs and most machine sizes are powers of two. After multiplying by some constant in order to change the load, they will not be powers of two, which may have a strong effect on how they pack, and thus on the observed fragmentation. This effect can be much stronger than the performance effects we are trying to measure [6]. Second, small jobs cannot be changed with suitable fidelity as the sizes must always be integers. Third, when large jobs are multiplied by a constant larger than 1 in order to increase the load, they may become larger than the full machine.
  A variant on this method is to combine scaling with replication. This allows larger jobs to be generated without losing smaller jobs, and has been suggested as a method to adapt simulations to different machine sizes [2].
  An alternative approach that has essentially the same effect is to modify the simulated machine size. This avoids the problem presented by the small jobs. However it may suffer from changing the inherent fragmentation when packing jobs together. Also, when the machine size is reduced to increase load, the largest jobs in the log may no longer fit.
  A possible alternative is not to multiply job sizes by a constant but to change the distribution of job sizes. Consider the CDF of the distribution of job sizes. To increase the load we want more larger jobs. Multiplying job sizes by a constant will cause the CDF to shift to the right, with all the ill-effects noted above. The alternative is to shift the top-right part of the CDF downwards. As a result, the relative proportion of large jobs is increased and the total load increases too. However, the idea of changing the distribution in this way has only received limited empirical support [12], and more work is needed.

Another alternative is to use multiple logs that have different loads. The problem then is that the workloads in the different logs may have completely different characteristics, so comparing them to each other may not be meaningful. Also, the number of available logs and the available load values may not suffice.

The most common approach used is to artificially change the load of a log by multiplying arrival times by a constant as described above, despite this method's deficiencies. Our goal is to find an alternative to this approach.

3 Evaluations Based on the Variability in a Single Run

As an alternative to evaluating a parallel job scheduler using simulations with a job log that has been subjected to various manipulations, we suggest to exploit the natural load fluctuations that occur in any log. In other words, by observing

periods of low load separately from periods of high load, we may try to uncover the effects of load on performance. This has the following benefits:

- It is more realistic, because it is based on real load conditions that had occurred in practice when the log was recorded, with no artificial manipulations, and
- It is easier in the sense that a single simulation can be used instead of multiple simulations.

However, it also has its drawbacks. For example, in a given log the range of load conditions that have occurred may be limited. Nevertheless, we feel that this approach is worthy of investigation.

Note that the suggested approach is different from the conventional approach at a very basic level. In the conventional approach, the average performance is found as a function of the average load. This is similar to the outcome of queueing analyses, such as the well-known M/M/1 queue. The suggested approach does not concern itself with different average load conditions. It is actually about understanding the variability and dispersion of performance, and the possible correlation between this dispersion and load — not about performance under different average loads. We discuss this further in the conclusions.

The approach of analyzing a single log and dissecting it according to load conditions was pioneered by Rudolph and Smith in the context of evaluating large-scale systems in the ASCI project [8]. Their goal was to establish whether these machines were being used efficiently. By analyzing workload logs, they attempted to show that performance as a function of load exhibits a "knee" at some load level, and beyond that point performance deteriorates markedly. Then if most of the jobs execute under a load that is just below the knee, the machine is being used efficiently.

The procedure employed by Rudolph and Smith to analyze the logs was somewhat involved. The analysis was performed at the level of individual jobs. For each job, they first found the average system utilization experienced by that job during its tenure in the system (this is explained in more detail below). The jobs were then binned according to the load into deciles: those jobs that experienced around 0 load, those that experienced around 10% load, those that experienced around 20% load, and so on up to those that experienced 90% and 100% load. Then a "bubble plot" was drawn. The X axis in these plots is the load, and the Y axis is the performance metric, e.g. average slowdown. Each class of jobs is represented by a disk. The coordinates of the center of the disk are at the average load experienced by jobs in the class and the average slowdown of jobs in the class. The size of the disk represents the number of jobs in the class.

The load experienced by a job was calculated as follows. The load (or utilization) at each instant is simply the fraction of processors that are allocated to running jobs (due to fragmentation, there are often some unused processors even if additional jobs are waiting in the queue). This only changes when the scheduler decides to start running a job, or a running job terminates. Assume a certain job arrives at time t and terminates at time t'. Consider the set of time instants from t to t' at which any job either starts to run or terminates, and

number them from 0 to n (such that $t_0 = t$ and $t_n = t'$). Denote the utilization during the interval t_i to t_{i+1} by $U(t_i, t_{i+1})$. The load experienced by the job is then

$$\text{load} = \sum_{i=1}^{n} \frac{t_i - t_{i-1}}{t_n - t_0} U(t_{i-1}, t_i)$$

(Inexplicably, the instantaneous utilization calculation sometimes leads to values greater than 1, which implies data quality problems in the logs. We discuss such issues in a separate paper [4]; in the current context they are rare enough to be largely meaningless.)

In the following we use the Rudolph-Smith bubble plots as our starting point, and use them to further analyze parallel job logs. But the motivation is not only to understand the performance as it was on specific machines in the past. Rather, we contend that the same analysis can be applied to simulation results. In other words, when running a simulation of a parallel job scheduler on a given workload, a *new* log recording the performance of the simulated system can be recorded. This (single) log can then be analyzed in the same way as real logs are analyzed, to uncover the performance as a function of load.

4 Evaluating Parallel Job Logs with Bubble Plots

Example bubble plots are shown in Fig. 1. The plot for the CTC SP2 log looks approximately as expected: the average slowdown tends to grow from around 10 to about 30 with increased load, and most of the jobs observed what appears to be the maximum sustainable load, which is around 70% in this case. However, the few jobs that enjoyed near zero load suffered a slowdown of around 50, and those that suffered a load of 90-100% enjoyed a slowdown of less than 10. Also, the plots for other logs are messier. For example, the KTH log exhibits a zig-zag pattern, the SDSC Paragon shows marked decrease in slowdown with increased load, and in SDSC Blue the jobs are evenly distributed across all loads.

Rudolph and Smith also observed some strange patterns like this. Part of their solution was to filter some of the jobs in the log. In particular, they filtered jobs that were shorter than 1 minute, and jobs that had a very high slowdown. We also did so (the graphs in Fig. 1 are after such filtering, where the threshold for "high slowdown" was 1000). However, while the strange behavior is reduced, it is not eliminated.

The high slowdowns with low utilizations are somewhat of a mystery. The phenomenon seems to happen because jobs are not scheduled to run while processors are in fact available. Possible excuses are unrecorded down time, when the processors were actually not available but we don't know it, or special scheduler considerations, such as reserving processors for some other use. Another possible explanation is that processors are not the only important resource, and perhaps also not the most important one. Thus jobs may be delayed if sufficient memory is not available, or if they need to use a floating software license that is being used by another job. Such considerations do not affect the utilization metric.

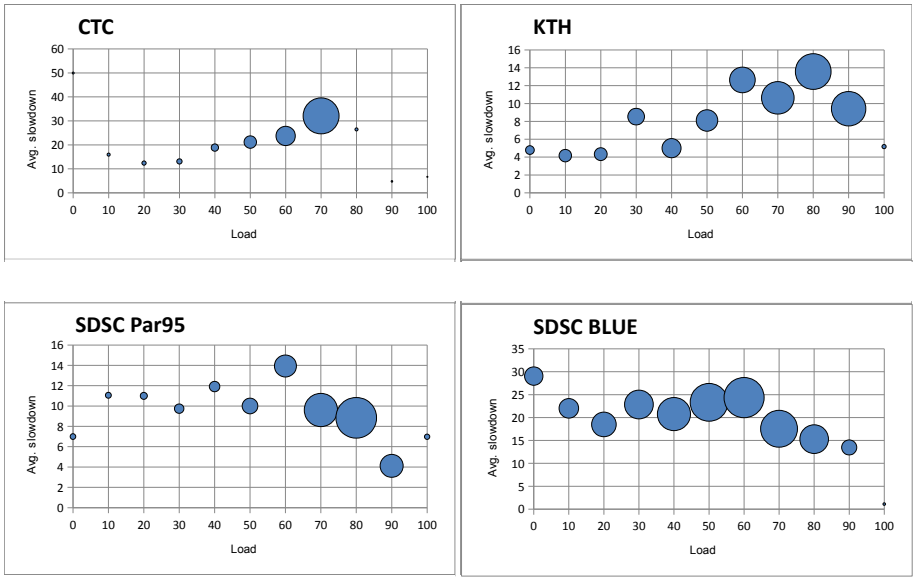


Fig. 1. *Examples of bubble plots derived from different logs. Short jobs (< 1 minute) and jobs with high slowdown (> 1000) were filtered out.*

The low slowdowns observed with high utilizations can be explained as follows. Consider a sequential job. When such a job arrives, it will be able to run immediately if the utilization in less than 100%. Moreover, when the utilization is high, *only* such jobs will be able to run immediately (because very few processors if any are available). So many jobs that see a high utilization throughout their lifetime can be expected to be serial jobs that run immediately, and therefore have slowdown 1 — the lowest (and best) slowdown possible.

Conversely, consider a large job that requires many processors. Such a job will most probably have to wait until the processors become available. Moreover, it will block other jobs and cause processors to become idle while it waits. Therefore it will see a lower average load during its lifetime, but suffer a high slowdown due to the waiting.

Slowdown is sensitive to short jobs that may have very high slowdown values [11, 3]. This may explain the variability in the bubble plots: if in a certain group of jobs one happened to have a very high slowdown, this could affect the average of all of them and cause the bubble to float upwards. Such effects could lead to the uneven behavior seen in Fig. 1. Additional support for this hypothesis comes from looking at the median slowdown instead of the average. The plot of medians and plot of averages turn out to be quite different from each other.

The conclusion is that bubbles may be too coarse, as they represent potentially large groups of unrelated jobs. As an alternative, we suggest looking at all the jobs at a much higher resolution.

5 Evaluating Parallel Job Logs with Heatmaps

The way to look at the performance vs. load data in more detail is to use heatmaps. The axes remain the same: the X axis represents load, and the Y axis represents performance (this can be slowdown, as used in the Rudolph-Smith bubble plots, but also response time or wait time). But instead of using a coarse classification of loads and lumping all the jobs that saw approximately the same load together, we now use a relatively fine classification according to both load and performance. The number of jobs that experienced approximately the same load and same performance is then represented by the shading: a darker shading corresponds to a higher numbers of jobs. The actual numbers differ according to the log's length, so contours are used to give an indication of the number of jobs in the high-density areas.

Applying this to the different logs leads to the heatmaps shown in Figs. 2 to 9. For each log, heatmaps of slowdown, response time, and wait time vs. load are shown. Note that the Y axis is logarithmic. Slowdowns of 1 are at the bottom of the scale, adjacent to the X axis. As wait times can be 0 and this cannot be shown on a logarithmic scale, we artificially change 0 values to 1, so they too are shown at the bottom adjacent to the X axis. Bubble plots calculated from the same data are superimposed on the heatmaps for comparison.

These graphs allow for several general insights and some specific ones. The main general insights are as follows:

- The average values are unrepresentative. In each heatmap, the point of average load and average performance is marked with an 'X'. This represents the output of conventional analysis. But as we can clearly see, this often falls in a relatively sparse area of the heatmap.
- Likewise, the bubbles, which are shown overlaid on the heatmap, are unrepresentative. (In these graphs the bubbles are based on all the jobs, with no filtering).
- The performance distribution tends to be highly skewed. Many jobs have very low wait times and a slowdown of 1. The higher averages are a result of combining this with relatively few jobs that suffer from much worse performance.
- In many cases the load distribution is also highly skewed. In particular, many jobs actually observe very high utilizations of near 100% as they run. The lower average load is a result of the low-load and idle periods, which actually affect only a small number of jobs.
- There appears to be *only a weak if any functional relationship between experienced load and performance*. In other words, it is not generally true that jobs that experience higher loads also suffer from worse performance. It is true that we often see a concentration of jobs at the right of the heatmap (high loads), and that this includes the top-right area (bad performance), but it also typically includes the bottom-right area (good performance) to a similar degree. To quantify the possible correlation of load and performance we calculated the Spearman rank correlation coefficient between them for

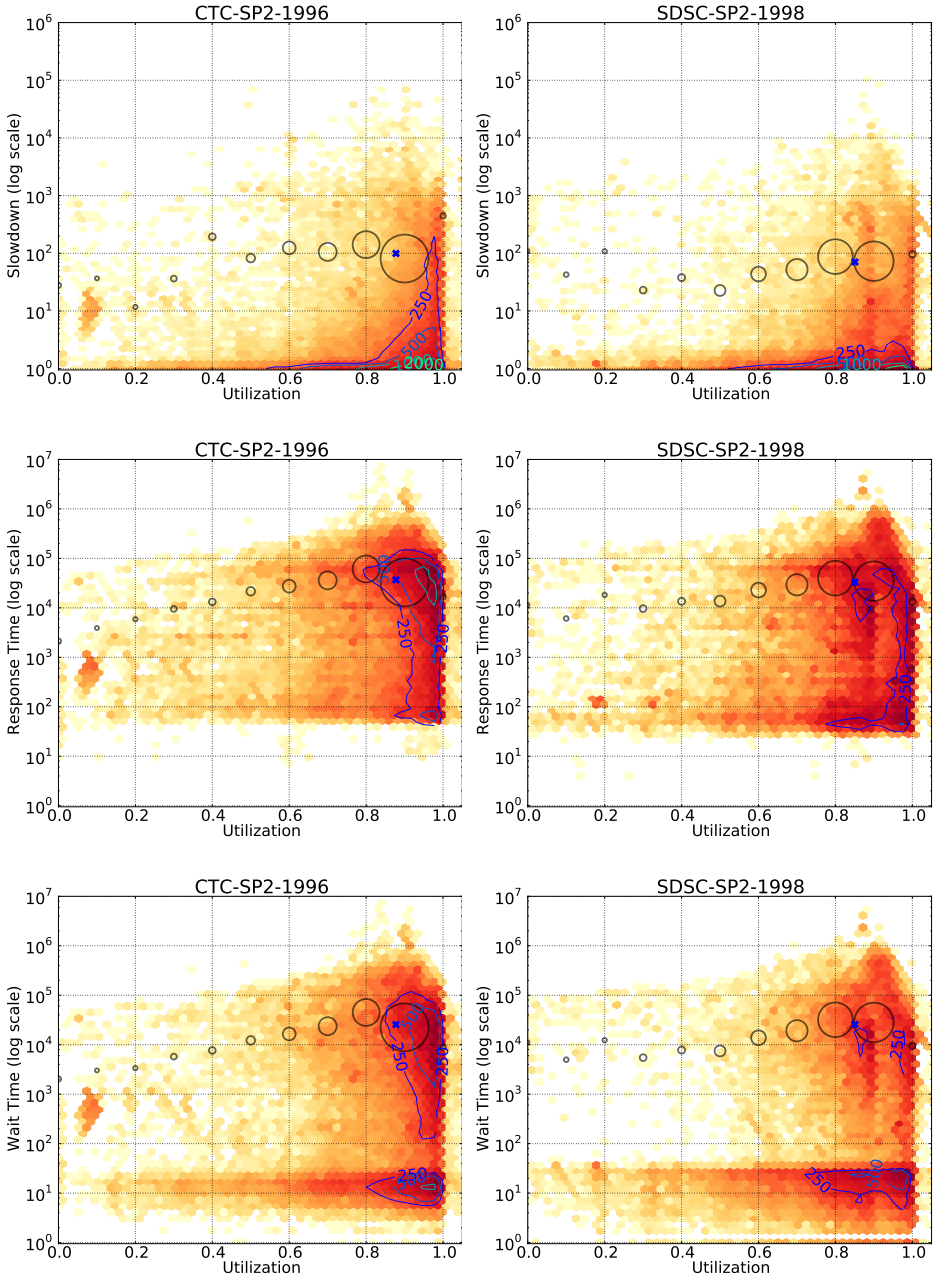


Fig. 2. *Heatmaps for the CTC SP2 log.*

Fig. 3. *Heatmaps for the SDSC SP2 log.*

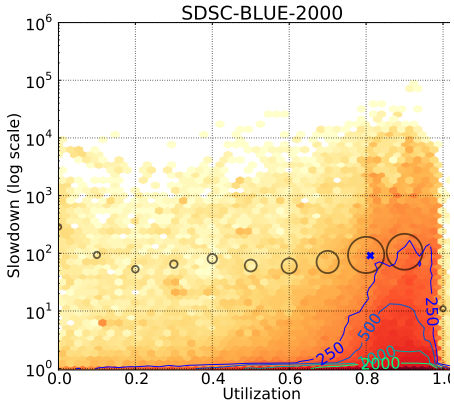


Fig. 4. *Heatmaps for the SDSC Blue log.*

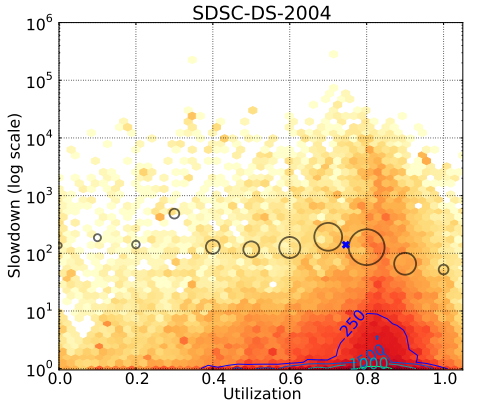
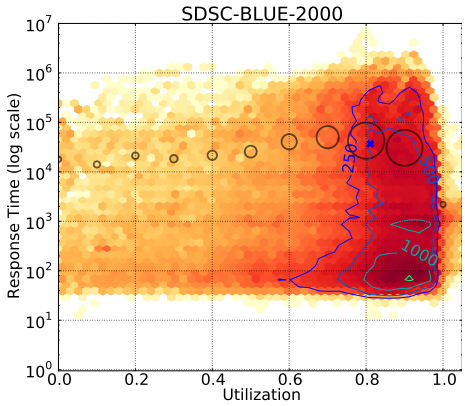
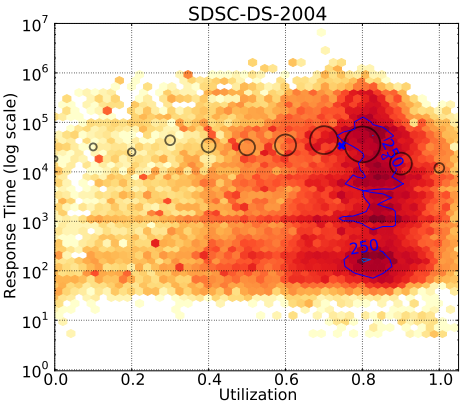
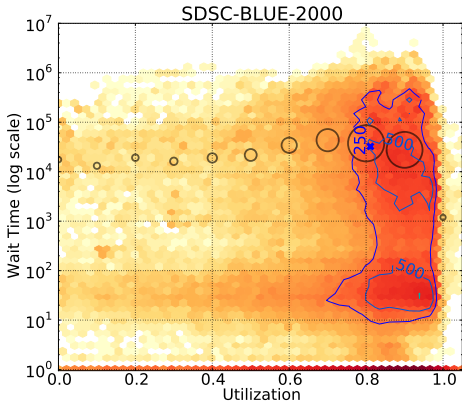
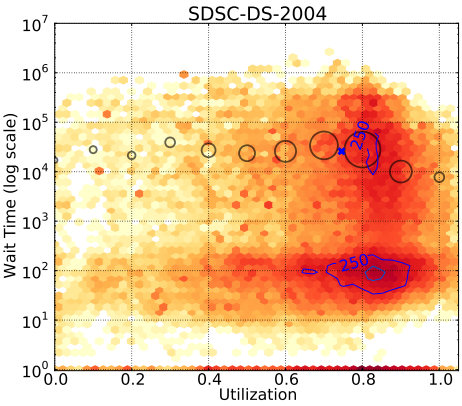


Fig. 5. *Heatmaps for the SDSC DS log.*









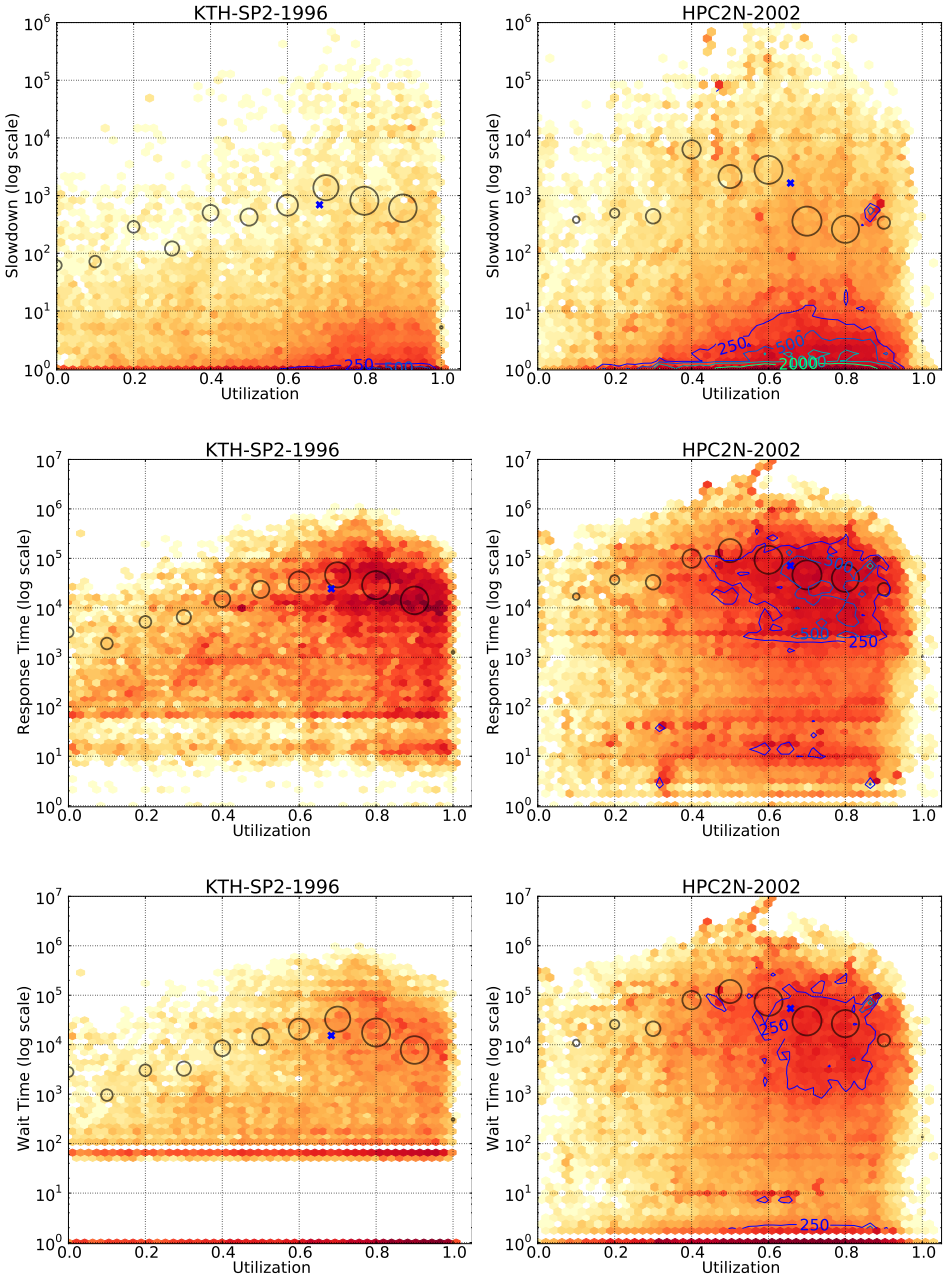


Fig. 6. *Heatmaps for the KTH SP2 log.*

Fig. 7. *Heatmaps for the HPC2N log.*

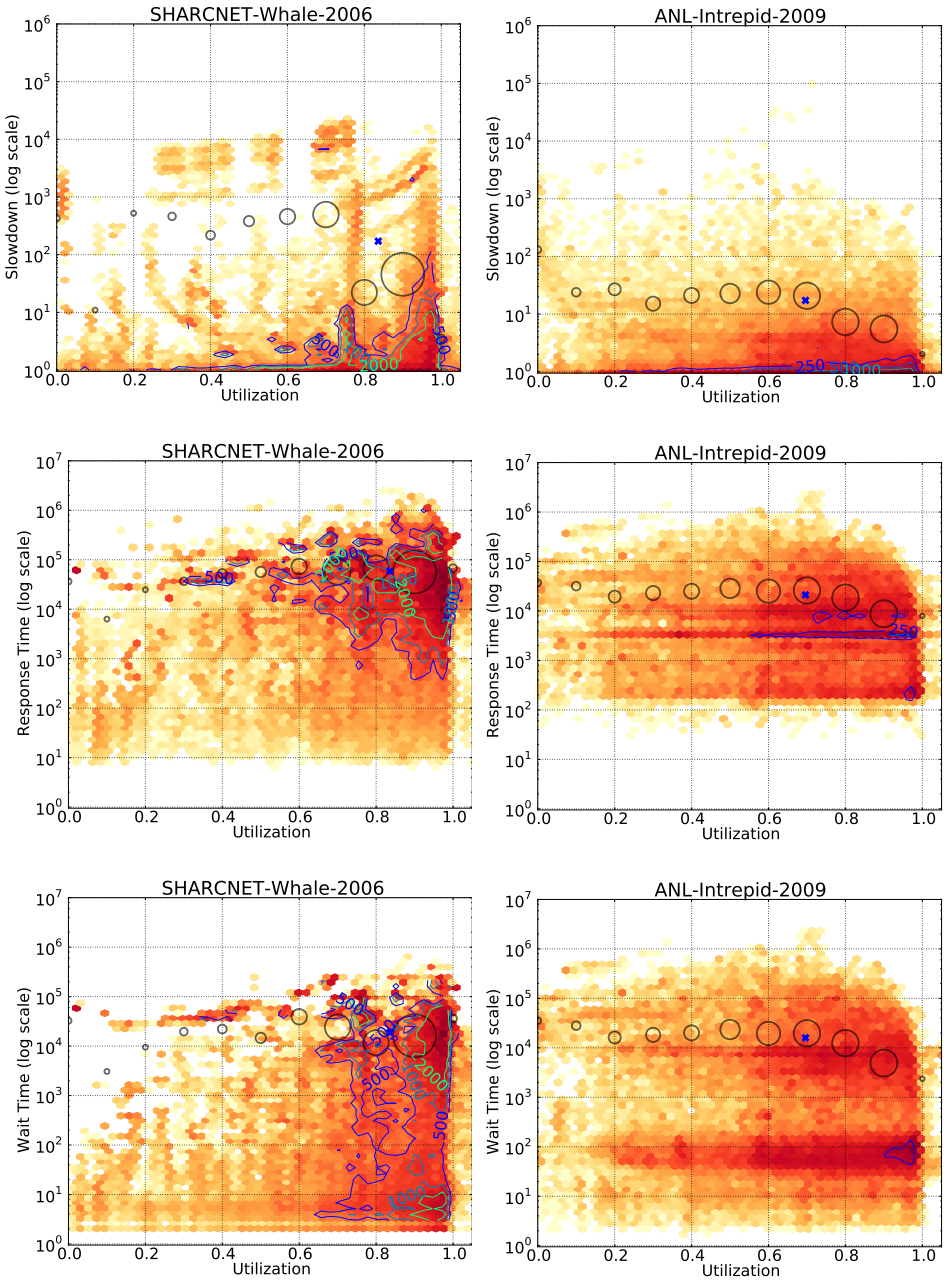


Fig. 8. *Heatmaps for the SHARCNET Whale log.*

Fig. 9. *Heatmaps for the ANL Intrepid log.*

different logs. The absolute values of the results are typically less than 0.2, and often also less than 0.05, indicating very low correlation (Table 1).

Rudolph and Smith in their paper that introduced the bubble plots were looking for the characteristic behavior of a queueing system: good performance at low loads, that deteriorates asymptotically as the system load approaches saturation. Their success varied; some of the bubble plots exhibited the expected characteristics, while others were rather messy and hard to understand. These results were replicated in our work above.

Using the heatmaps, we can take a more detailed look at performance as a function of load. Focusing on slowdown to begin with, we find that for many logs there is a strong concentration of jobs along the bottom and right boundaries of the plot. The concentration along the bottom represents the jobs that enjoyed the best possible performance, namely a slowdown of 1. This happened at all loads, and dominates low loads. The concentration at the far right represents jobs that suffered from congestion under a high load. in some logs, e.g. CTC, this only happens at near 100% utilization. In SDSC SP2 there seem to be two distinct concentrations, at 90% and at 100%. In SDSC Blue the concentration is near 90%. In SDSC DS, it happens at around 80%.

Interestingly, there were also logs that did not display the expected pattern at all. Examples are the KTH, HPC2N, and Intrepid logs. In these systems we see a wide smear, with no concentration of jobs that experience high loads. Rather, jobs seem to suffer approximately the same slowdowns regardless of load. An optimistic interpretation of this result is that the scheduler is doing something good, and manages to avoid bad performance under high load. As we show below, however, a more realistic interpretation seems to be that the scheduler is incapable (or unwilling) to exploit low load conditions in order to improve performance.

Similar observations may be made for response time. Here we do not see a concentration of low response times under low loads, because response times are more varied due to run times being varied. However, in many logs we do see a concentration of jobs at the right end of the plot, reflecting high loads. These include CTC, SDSC SP2, SDSC Blue, and SDSC DS.

The concentration at low values (indicating good performance) is clearly evident when we look at wait times. Several logs actually have a distinct bimodal distribution of wait times: very short wait times of up to about a minute, and long wait times of many minutes to several hours (note that in all the plots the Y axis is logarithmically scaled). Good examples are again CTC, SDSC SP2, SDSC Blue, and SDSC DS. The short wait times apparently reflect some minimal granularity of activating the scheduler, such that it only runs say once a minute and therefore does not schedule newly arrived jobs immediately [4].

Potentially interesting patterns that probably deserve further study appear in the heatmaps of specific logs. These include

- A distinct blob of jobs at the left side of the CTC heatmap. This is a concentration of jobs that saw very low load, but nevertheless suffered non-trivial slowdowns.

log	*metric*	*data*	*EASY*	*FCFS*
CTC SP2	slowdown	0.17	0.62	0.51
	response time	-0.03	0.13	0.20
	wait time	0.08	0.56	0.43
KTH SP2	slowdown	-0.01	0.55	-0.10
	response time	0.15	0.07	-0.28
	wait time	0.09	0.44	-0.26
SDSC SP2	slowdown	0.16	0.60	0.40
	response time	0.02	0.26	0.28
	wait time	0.11	0.55	0.37
HPC2N	slowdown	0.08	0.72	0.68
	response time	0.05	0.41	0.48
	wait time	0.08	0.70	0.69
SDSC Blue	slowdown	0.04	0.58	0.48
	response time	0.02	0.32	0.39
	wait time	0.00	0.57	0.47
ANL Intrepid	slowdown	-0.05	0.66	0.44
	response time	-0.07	0.32	0.32
	wait time	-0.05	0.63	0.41

Table 1. *Spearman's rank correlation coefficient of performance vs. load, showing much higher values for simulation results than for the original logs.*

- Strange patterns in the SHARCNET Whale log. These seem to reflect sets of jobs that suffered from some congestion conditions.
- A horizontal band in the Intrepid response time map, that probably reflects many jobs with the same runtime.

6 Comparing Heatmaps of Logs with Heatmaps from Simulations

We also ran straightforward simulations using the EASY [5] and FCFS schedulers on the logs, and compared the resulting heatmaps to the heatmaps produced based on the original log data. Examples are shown in Figs. 10 to 12. The leftmost column in these figures reproduces the data shown previously for the original log. The middle column is the result of an EASY simulation, and the rightmost one is FCFS. The comparison leads to two main observations.

- The simulations tend to produce "nicer" results. Specifically,
  - More jobs have a slowdown of 1,
  - High slowdowns and wait times occur only when load is near 100%, and
  - There are no strange patterns.
- The simulations do not reflect reality! It seems that the schedulers on the real systems are often restricted in some way, and cannot achieve efficient packing of the executed jobs. In addition, in the simulations there is a much stronger

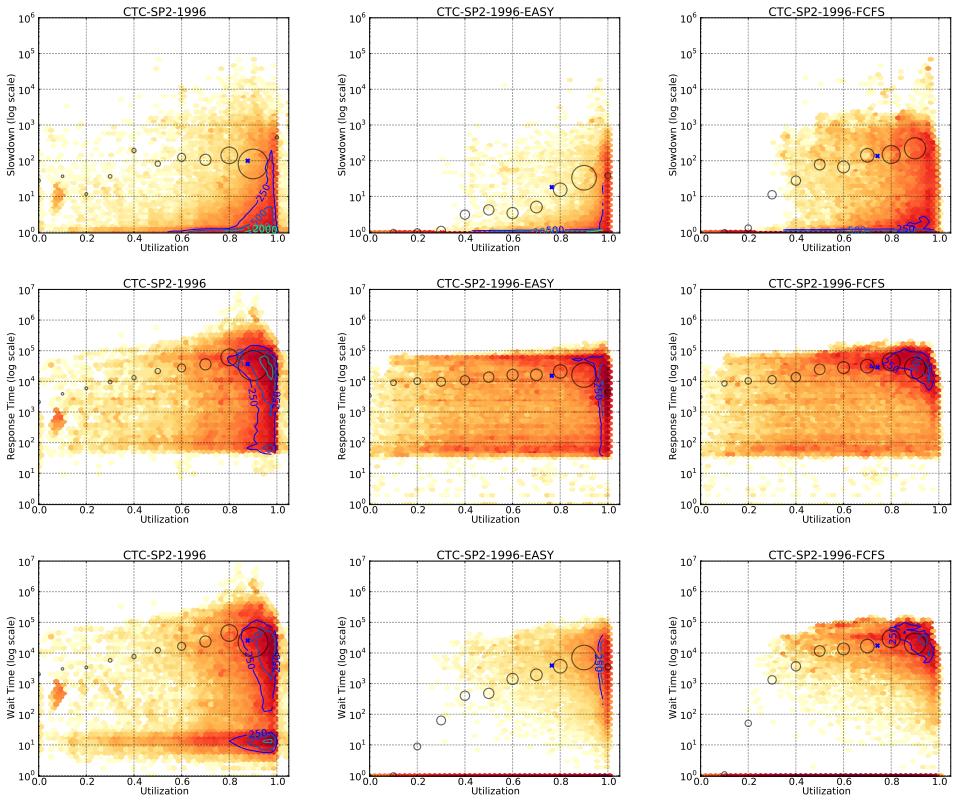


Fig. 10. *Heatmaps produced by running EASY and FCFS on the CTC log.*

correlation between experienced load and the resulting performance. With EASY in particular, high slowdowns and wait times are seen exclusively for jobs that suffered from high load conditions. With FCFS this happens to a somewhat lesser extent. Spearman rank correlation coefficients for several logs are shown in Table 1.

In more detail, consider the CTC workload shown in Fig. 10. From the heatmaps it appears that the original CTC scheduler is closer to FCFS than to EASY. But in fact it is even worse than FCFS. Looking at the scale, we find that for EASY simulations the response times are evenly smeared from around 30 seconds to around 80,000 seconds, with rather sharp boundaries. The top limit probably reflects a runtime limit imposed by the system administrators. But in the original log we don't see any such boundary, and response times may be as high as a million seconds.

Looking at the KTH log we see a different picture (Fig. 11). Here it seems that the heatmaps produced from the original log are somewhat more similar to the heatmaps produced by EASY, implying that the original scheduler behaves

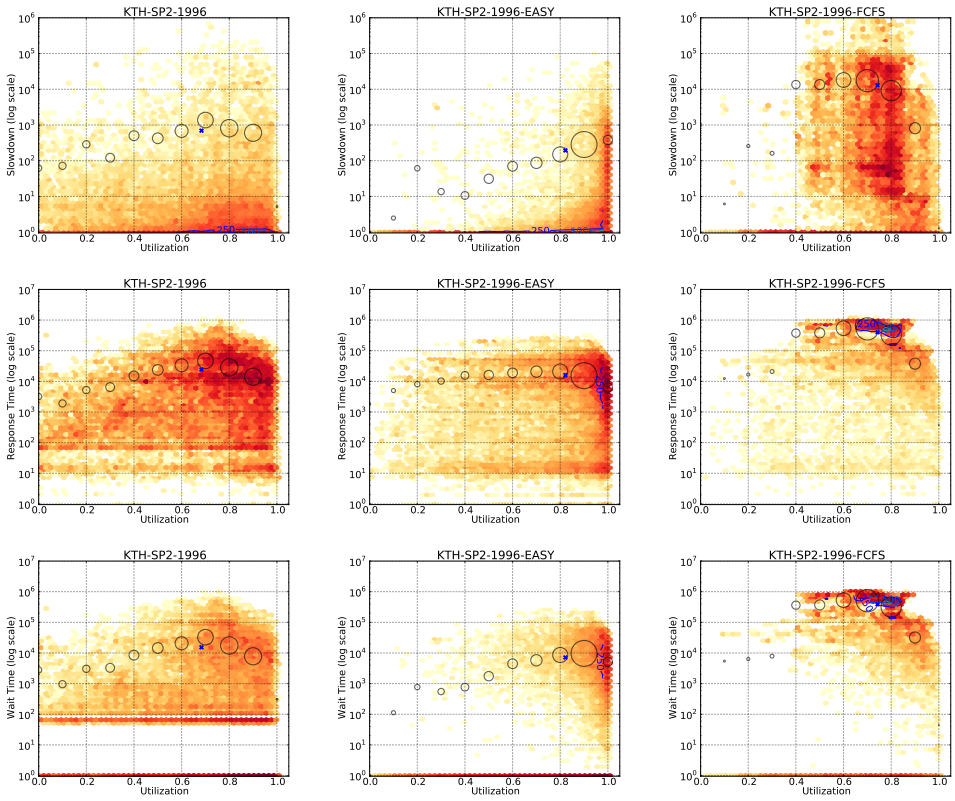


Fig. 11. *Heatmaps produced by running EASY and FCFS on the KTH log.*

more like EASY than like FCFS. FCFS produces much higher response times and wait times, and they are all concentrated at the same high values. Regarding slowdowns, EASY produces much lower slowdowns except at the very highest loads.

Another example comes from the HPC2N log, shown in Fig. 12. Here the distribution of response times is approximately the same for both the original scheduler and the simulated ones. However, with the simulated schedulers many fewer jobs see low loads, and most jobs are concentrated at the extreme right, indicating near 100% utilization. In the original log, in contradistinction, they were scattered from about 30% to about 90%.

7 Conclusions

We set out to devise a new way to use data from accounting logs for performance evaluations. The idea was that the long-term load on a system exhibits natural fluctuations, and these can be exploited in order to evaluate performance under

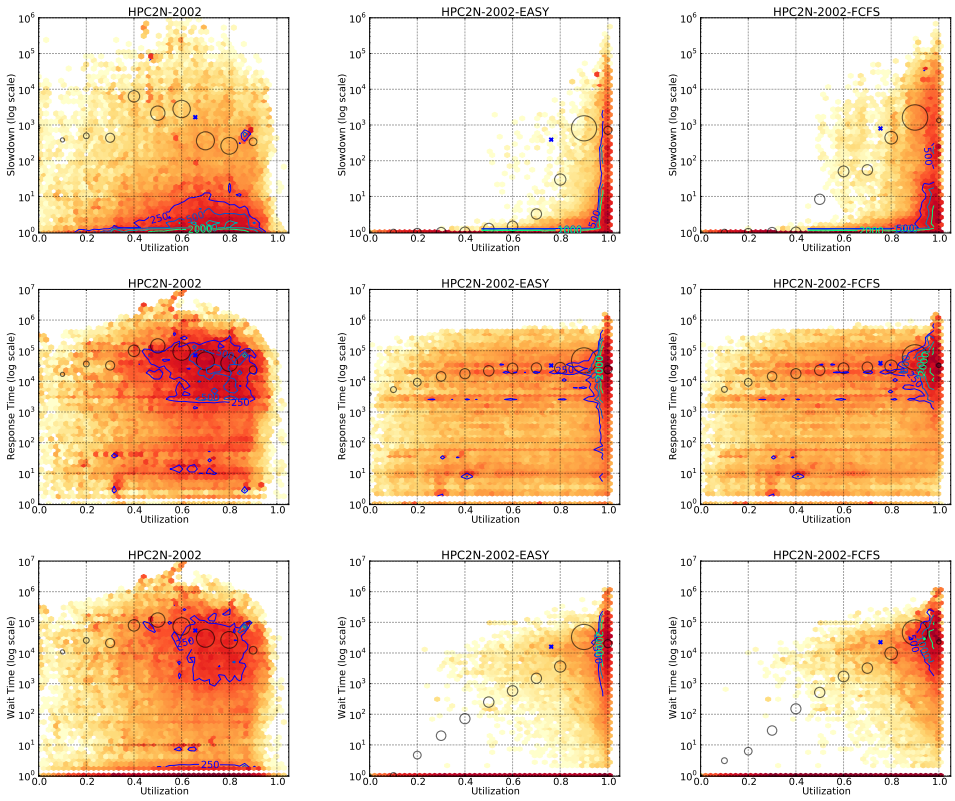


Fig. 12. *Heatmaps produced by running EASY and FCFS on the HPC2N log.*

different load conditions. This idea can be applied directly to the available logs in order to analyze the system in production use. It can also be applied to the output of simulations, whether driven by real logs or by synthetic workloads.

To implement this idea we use heatmaps, where the X axis represents load and the Y axis represents performance. The shading at each point reflects the number of jobs that experienced this load level and enjoyed this level of performance.

This analysis led to two main outcomes. The first was the observation that our heatmaps expose a wealth of information that has been glossed over till now. In particular, the common practice of reporting average performance as a function of average load seems ill-advised, as both load and performance have skewed distributions. Thus the average values do not reflect system behavior.

The second was the observation that conventional simulations do not reflect what is going on on real systems. Simulations using EASY, and sometimes also simulations using FCFS, produce behaviors that are markedly different and often much better than those observed in the original logs. This seems to indicate that

real schedulers employ various considerations that limit their options, and lead to sub-optimal packing of jobs. It is not clear at this point whether this reflects deficiencies in production schedulers, or maybe deficiencies in simulations. It is certainly possible that simulations like those we performed are over simplified, and do not take all the real world considerations into account. For example, real schedulers need to consider memory requirements, software licenses, and heterogeneous configurations, and do not just count processors.

Analyzing the variability in real systems or single simulation runs as we suggest represents a significant departure from current practice. This immediately leads to the question of whether this can indeed be used to gauge performance as a function of load, or maybe it is necessary to actually change the overall average load on the system. In defense of our approach, we note the recent interest in generative user-based workload models. In such models the simulation includes not only the system, but also the processes by which users generate the workload [9, 10]. An important element in such models is the feedback from the system to the users. In particular, when performance is bad users may elect to leave the system. Such feedback leads to a self-regulating effect, and may counteract attempts to increase the average load.

In any case, we suggest that evaluations of parallel job schedulers will do well to utilize heatmaps like the ones we produced in order to better understand the behavior of the systems under study. However, this is only the first step. Additional research is needed in order to make better use of the heatmaps. In particular, we suggest the following.

- In our work we interpret "load" as the average system utilization observed by each job (as was done by Rudolph and Smith). This ignores the backlog that may accumulate in the scheduler's queue (except for the fact that a large backlog may cause a job to be delayed in the queue, and therefore the load calculation will cover a longer interval). But it can be argued that the overload represented by this backlog is also an important component of the system load. The question is how to incorporate this information explicitly in the load calculation.
- Our heatmaps enable patterns to be observed for a specific log and scheduler. An important extension would be to find a good way to compare such heatmaps to each other. In particular, is there a good metric for evaluating whether one heatmap represents "better performance" than another?
- It may also be useful to consider subsets of jobs, and draw independent heatmaps for them. For example, this can be done for jobs with a certain range of degrees of parallelism, or jobs belonging to a certain user.
- Finally, the heatmaps may also be used to characterize and evaluate synthetic workload models. By comparing a heatmap representing the behavior of a given scheduler on a synthetic workload with a heatmap of that scheduler's behavior on a real log we can see whether the synthetic workload leads to reasonable behavior.

Acknowledgments

Many thanks to all those who have made their workload data available through the Parallel Workloads Archive. Thanks are also due to the anonymous reviewers who contributed many ideas and observations that improved upon the original paper.

References

1. S. J. Chapin, W. Cirne, D. G. Feitelson, J. P. Jones, S. T. Leutenegger, U. Schwiegelshohn, W. Smith, and D. Talby, "*Benchmarks and standards for the evaluation of parallel job schedulers*". In *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph (eds.), pp. 67–90, Springer-Verlag, 1999. Lect. Notes Comput. Sci. vol. 1659.
2. C. Ernemann, B. Song, and R. Yahyapour, "*Scaling of workload traces*". In *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson, L. Rudolph, and U. Schwiegelshohn (eds.), pp. 166–182, Springer Verlag, 2003. Lect. Notes Comput. Sci. vol. 2862.
3. D. G. Feitelson, "*Metric and workload effects on computer systems evaluation*". *Computer* **36(9)**, pp. 18–25, Sep 2003.
4. D. G. Feitelson, D. Tsafrir, and D. Krakov, "*Experience with the parallel workloads archive*", 2012. (In preparation).
5. 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.
6. V. Lo, J. Mache, and K. Windisch, "*A comparative study of real workload traces and synthetic workload models for parallel job scheduling*". In *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph (eds.), pp. 25–46, Springer Verlag, 1998. Lect. Notes Comput. Sci. vol. 1459.
7. "*Parallel workloads archive*". URL http://www.cs.huji.ac.il/labs/parallel/workload/.
8. L. Rudolph and P. Smith, "*Valuation of ultra-scale computing systems*". In *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph (eds.), pp. 39–55, Springer Verlag, 2000. Lect. Notes Comput. Sci. vol. 1911.
9. E. Shmueli and D. G. Feitelson, "*Using site-level modeling to evaluate the performance of parallel system schedulers*". In 14th *Modeling, Anal. & Simulation of Comput. & Telecomm. Syst.*, pp. 167–176, Sep 2006.
10. E. Shmueli and D. G. Feitelson, "*On simulation and design of parallel-systems schedulers: Are we doing the right thing?*" *IEEE Trans. Parallel & Distributed Syst.* **20(7)**, pp. 983–996, Jul 2009.
11. S. Srinivasan, R. Kettimuthu, V. Subramani, and P. Sadayappan, "*Selective reservation strategies for backfill job scheduling*". In *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson, L. Rudolph, and U. Schwiegelshohn (eds.), pp. 55–71, Springer-Verlag, 2002. Lect. Notes Comput. Sci. vol. 2537.
12. D. Talby, D. G. Feitelson, and A. Raveh, "*A co-plot analysis of logs and models of parallel workloads*". *ACM Trans. Modeling & Comput. Simulation* **12(3)**, Jul 2007.