The Effect of Metrics and Workloads on the Evaluation of Computer Systems

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

Abstract

Metrics ought to be objective, as they are the judge of performance. Workloads ought to be representative, so that evaluations will lead to applicable results. But sometimes metrics and workloads collude to taint the performance evaluation process, leading to results of dubious merit. We use a case study dealing with parallel job scheduling to exemplify these issues. An analysis of interactions among the metrics, the workloads, and the systems being studied reveals that such interactions may dominate the evaluation results. Moreover, in some cases factors that were originally thought to be minor and ignorable are actually very important, and may overshadow the differences between the different systems. It is therefore recommended that multiple workloads and metrics be used in performance evaluation studies, and that the causes of inconsistent results be studied thoroughly.

Keywords: Workload model, performance metric, interactions among factors.

1 Introduction

The goal of performance evaluation is often to compare different system designs or implementations. The evaluation is expected to bring out performance differences that will allow for an educated decision regarding what design to employ or what system to buy. It is assumed that observed performance differences reflect important differences between the systems.

However, performance differences may also be an artifact of the evaluation methodology. The performance of a system is not only a function of the system design and implementation. It may also be effected by the workload to which the system is subjected. In addition, different metrics measure different things. In this paper we focus on the identification and analysis of situations in which workloads and metrics sway the results of performance evaluation.

There are two main approaches to performance evaluation: analysis and simulation. Analysis necessarily involves simplifications in the interest of mathematical tractability. Simulation is more realistic, and in particular, can directly use recordings of real workloads. The problems we discuss are more relevant to simulation, but we claim that this is not a deficiency of simulation. Rather, it

Revised and corrected version, June 2003.

echos the fact that simulation may directly reflect complex situations, even if they are not known or understood by the person performing the evaluation.

The domain used in this work is that of parallel job scheduling. Workloads in this field are interesting due to the combination of being relatively small (the size of typical workloads is tens of thousands of jobs) and at the same time relatively complex (jobs are characterized by attributes including size, runtime, runtime estimate, and arrival, and these attributes may be correlated). Naturally, the methodological concerns extend to other domains.

2 Experimental Design

We start by considering a simple question: what has more impact on performance results, the system being studied, or the methodology?

2.1 Factors and Levels

Experimental design is a useful technique to study the effect of different factors on a system's performance [9]. One first identifies the factors and their typical values (called "levels"), and then designs a set of experiments that will determine the relative importance of each factor. For example, when studying process scheduling, factors that affect the performance can be the use (or lack of use) of time slicing, the average process length, the arrival rate, the order in which queued processes are considered for scheduling, and so on.

We use this methodology with a twist: rather than studying factors that affect the system behavior, we study *factors that affect the evaluation procedure*. Specifically, we identify four main factors, each with several levels.

The first factor is the **metric** being used in the evaluation. The different metrics we consider are:

- Response time (the time from when a job is submitted until it terminates), using either an arithmetic average or a geometric average [1].
- Wait time, which is that part of the response time that is up to the system.
- Slowdown, which is the response time normalized by the job's actual running time.
- Bounded slowdown, in which the running time is used to normalize the response time only if it is higher than a certain threshold value [4]. This prevents very short jobs from creating very high values. Thresholds of 10, 60, and 600 seconds were used.
- Per-processor bounded slowdown, in which the bounded slowdown is further normalized by the number of processors used [12].

The second factor is the **load** on the system. When systems are underloaded, their performance is typically very similar. Higher load conditions expose differences in how systems react to load. Load conditions considered were 50%, 65%, and 80% of the system capacity, which are typical in production systems [5, 11]. The different load conditions were achieved by systematically changing the interarrival times of the jobs. However, due to burstiness in the workloads, this led to some variations in the loads experienced in the actual simulations.

The third factor is the **system**. The whole point of performance evaluation is to uncover performance differences between different systems, and our goal is to compare the magnitude of such differences with differences due to methodology. We compared systems with three different schedulers:

- Backfilling schedules jobs on dedicated partitions of processors, based on their order of arrival. However, if fragmentation occurs and processors are left idle, jobs from the back of the queue are allowed to bypass jobs that precede them (provided they fit). Two versions were used: in *conservative* backfilling jobs may backfill only if they will not cause delays for any bypassed jobs, whereas *EASY* is more aggressive and allows backfilling provided only the first queued job is not delayed [10]. Estimates of the runtimes are used to determine whether delays will occur; in real logs, the original estimates provided by users are used, whereas in models the actual runtime is used as an estimate.
- Gang scheduling is a preemptive scheme in which jobs are assigned to rows of a scheduling matrix, where columns represent the nodes of the system and rows represent time slots. The jobs in each row are scheduled in turn using coordinated context switching across all the nodes. The packing of the matrix is based on a variant of the Distributed Hierarchical Control scheme [3], which uses a buddy system to allocate processors in blocks that are powers of two.

The final factor is the **workload**. Seven different workloads were used, of which four were models and three were traces. Models are essentially representations of the workload using statistical distributions [6]. This has many benefits and can be used in analysis in addition to simulation. However, models are always a simplification of reality, and using the wrong statistical model can yield misleading results [7]. It is therefore sometimes argued that more reliable results are obtained by simulations that are driven directly by a trace that records the actual workload that was observed on a real production system. This has the benefit of including all the complexities of the real workload, even if they are unknown to the person performing the evaluation

The models used were those proposed by Feitelson, Jann, Downey, and Lublin. The real workloads are from three IBM SP2 systems, installed at KTH, CTC, and SDSC. Additional information, including software for the models and data for the logs, is available from the Parallel Workloads Archive at URL http://www.cs.huji.ac.il/labs/parallel/workload/.

We used a full factorial design, in which all combinations of the different levels were measured by simulation. Thus, for example, we ran a simulation of conservative backfilling using the Lublin workload model at a load level of 0.50, and measured all ten metrics. Overall there were 60 such simulations, for a total of 600 results (there should have been 63 simulations, but the combination of the Downey model and gang scheduling was problematic, as noted below).

2.2 Analysis of Variation

Analysis of variation (ANOVA) is a statistical technique used to assess the relative importance of different factors [9]. First, the average of *all* results is computed. Then the differences between specific groups of results and this global average are attributed to different factors and interactions. For example, if factor A has two levels a_1 and a_2, and the results of experiments using level a_1 lead to a higher average than the experiments using level a_2, then we say that the difference between

factors	*contribution*
A (metric)	80.43%
B (load)	1.66%
C (scheduler)	3.18%
D (workload)	3.06%
AB	0.12%
AC	3.32%
AD	6.38%
BC	0.05%
BD	0.16%
CD	0.98%
ABC	0.05%
ABD	0.08%
ACD	0.45%
BCD	0.03%
ABCD	0.05%

Table 1: *Results of analysis of variation (ANOVA).*

these two averages (which is part of the overall differences) is attributed to factor A. Likewise, interactions measure how combinations of multiple factors at specific levels effect the outcome. For example, given two factors A and B, if the experiments using the combination of A at level a_1 and B at level b_2 lead to very low results on average, we attribute this deviation to the AB interaction.

The results from the simulations described above were analyzed using Design-Expert 6 software from Stat-Ease, Inc. (http://www.statease.com/dx6descr.html). A logarithmic transform (base 10) was applied first, to reduce the range of values.

2.3 Results

The results of the analysis, showing the contribution of each factor and interaction to the variation, are given in Table 1.

The most striking result is obviously that over 80% of the variation[1] has been assigned to the metrics factor (designated A). However, this is not all that meaningful. Some metrics, like response time or wait time, are indeed very high on average. And they stay high regardless of the other factors. Others, like variants of bounded slowdown, are inherently lower. Thus this finding just means we should not mix metrics and compare them to each other without making sure that they are in the same units.

The contribution of load (designated B) is surprisingly low. Given that performance deteriorates as load increases, we would expect a larger effect. The explanation is probably that the highest load level, that of 80%, is still a moderate load, and does not push the systems to their

[1]The ANOVA methodology uses the square of deviations from the global average, so that deviations that are above and below it do not cancel out. This tends to inflate the largest values and diminish the smallest ones.

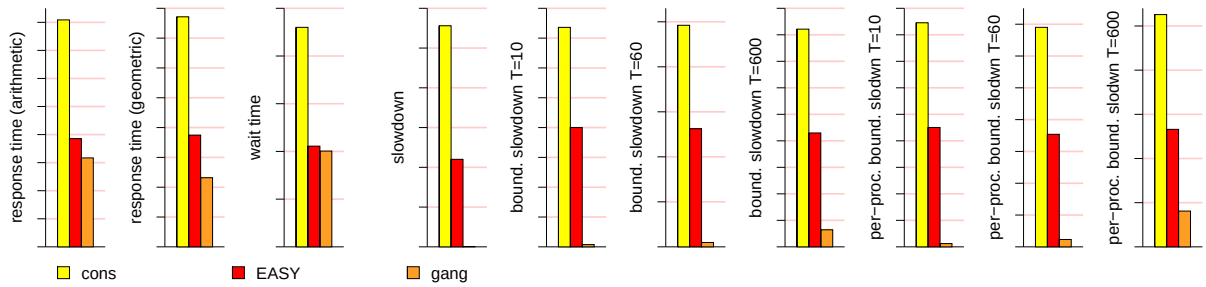

Figure 1: *Results for all 10 metrics when comparing the three schedulers, using an average of all workloads and loads (excluding the Downey workload for which gang scheduling results are not available).*

limit.

The scheduler (designated C) has some net effect, which is gratifying. It means that one of the compared schedulers is in general better than the others, for many different combinations of the other factors. This is the sort of results we are actually looking for in performance evaluation.

The workload (designated D) turns out to have a similar effect to the scheduler. As in the case of metrics, this is not necessarily bad. Some workloads may have longer jobs on average, leading to longer response times. Again, this just means that results obtained with one workload should not be compared directly with results obtained using a different workload: only results from the same workload on different systems are comparable.

The real problems exposed by the analysis are the interactions. For example, the AC interaction indicates that some metrics favor one scheduler, while other metrics favor another scheduler. Thus the selection of metric might determine the outcome of the evaluation in terms of which scheduler seems to be better! The AD interaction indicates that there is an even stronger interaction between the metric and the workload, and this interaction may swamp out the effect of the scheduler by itself.

3 The Double Interactions

We start with explaining the interactions between metrics, workloads, and schedulers. Load did not have any strong interactions, meaning that there were no system designs or metrics that consistently worked better under high or low loads.

3.1 Interaction of Metrics with Schedulers

In retrospect, the fact that metrics interact with schedulers should not be too surprising. Different schedulers are designed with different objectives in mind. If successful, they should therefore satisfy these diverse objective, among which are diverse performance goals. Oftentimes, satisfying one goal comes at the expense of another. Therefore metrics that are fashioned after specific goals will tend to give higher ranks to schedulers that include these goals among their stated objectives, and lower ranks to schedulers that do not.

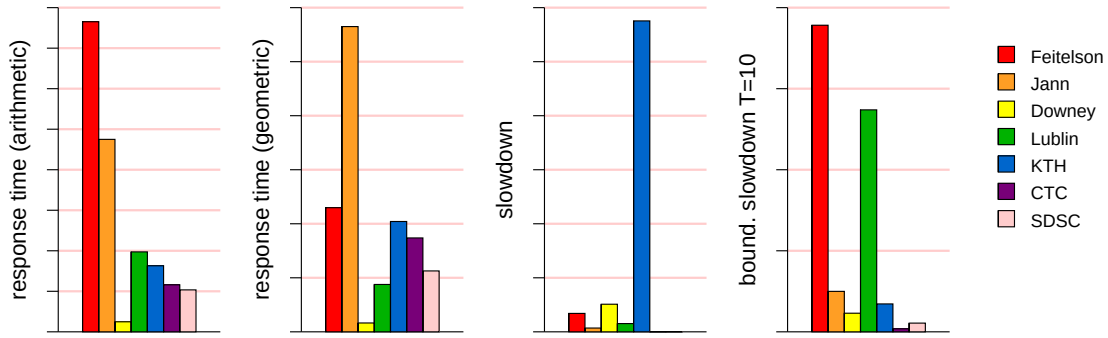

Figure 2: *Performance results based on select metrics, compared for the different workloads. Each bar is the average of the two backfill schedulers and all loads.*

In our testcase, a detailed study of the results reveals that the ten metrics may be divided into two groups. One group includes the response-related metrics: response time with either arithmetic or geometric averaging, and waiting time. These metrics, on average, show gang scheduling to be similar to EASY backfilling (first three charts in Figure 1). The other group, including the seven slowdown-based metrics, show gang scheduling to be significantly better than backfilling (following seven charts). This distinction can be explained by the fact that gang scheduling is preemptive, with the goal of preventing short jobs from waiting in queue until long ones terminate. Thus, while it should also reduce the average response time, it should reduce the response time of short jobs by much more. And this is precisely what is measured by the slowdown metrics.

3.2 Interaction of Metrics with Workloads

The interaction of metrics and workloads, like the interaction of metrics and schedulers, can be benign. For example, if one workload is characterized by jobs that are on average longer than those in another workload, their response times will also be longer on average. In other words, metrics that are naturally linked with a certain workload feature will cause interactions.

A more problematic case is when no such direct link is present, and especially if the metrics interact in conflicting ways with different workloads. An example is provided by the relative performance obtained by different workloads according to the different metrics (Figure 2). For example, the KTH workload has the highest slowdown by far, but the Feitelson and Lublin models have much higher bounded slowdowns; the CTC and SDSC workloads have higher response times than the Downey model (for both arithmetic and geometric averaging), but lower slowdowns (with and without a bound).

The problem with these results is not that we cannot rank workloads, as this is not our goal anyway and is rather meaningless. The problem is that there is a large effect that is due to the interaction of the metric and the workload. Thus when we use several workload/metric combinations, we might end up measuring these effects and not the system effects.

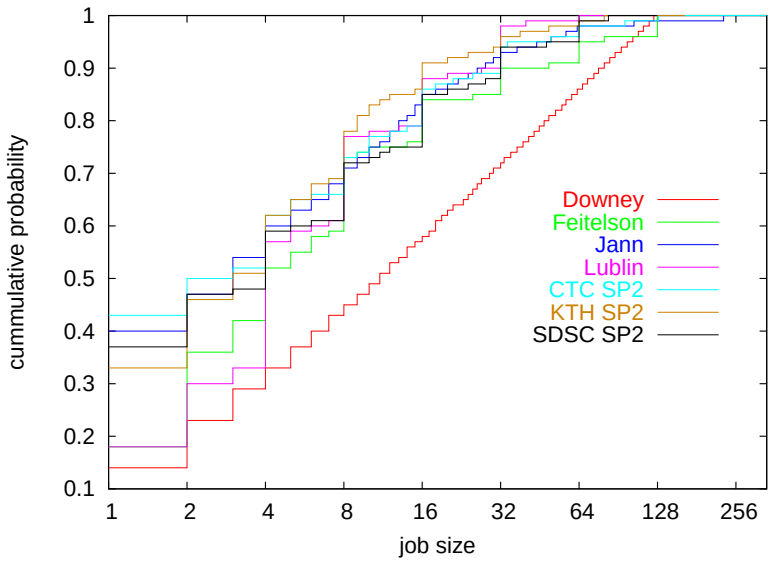

Figure 3: *Distributions of job sizes in the different workloads. Downey's model generates larger jobs than the others. The steps are due to the discreteness of the values, and preferences for powers of two.*

3.3 Interaction of Scheduler with Workload

Interactions of the scheduler with the workload are actually rather important, as they may uncover vulnerabilities of the system. For example, if a certain scheduler cannot handle a certain workload, this has two implications: first, this scheduler should not be used in systems whose workload resembles the specific problematic workload. Second, this information can be used to better understand the behavior of the scheduler and to improve it.

An example of this type occurred in the simulation of gang scheduling using the Downey workload. These simulations did not complete successfully. Specifically, the simulations for high loads became very backlogged and stretched the simulated time so much that the measured load was much lower than the input load.

The reason for this appears to be the distribution of job sizes. Downey's model employs a log-uniform distribution (actually, this is the distribution of the average parallelism of the jobs, as this model is designed for moldable jobs rather than for rigid ones). The other models and the logs have distributions that start with more small jobs and have a bit of a tail (Figure 3), leading to much smaller sizes throughout the distribution. Apparently, the packing algorithm used in the simulation of gang scheduling was unable to deal effectively with the log-uniform distribution, leading to high fragmentation: processors were left idle because allocations were in powers of 2, and for large jobs, this causes significant waste. This led to saturation at relatively low loads, starting from about 60%. The other distributions did not cause this problem, and the workloads were packed successfully even at loads of 80%.

job class	number of jobs*	response time		bound. slodwn. T=10	
		EASY	cons	EASY	cons
Jann model					
short	214715	8015.42	6403.52	142.99	109.16
long	118585	52654.77	65173.01	1.85	2.32
all	333300	23900.09	27313.15	92.77	71.15
CTC workload					
short	48020	3784.75	4632.44	22.85	45.39
long	30480	33866.62	37762.92	1.47	1.65
all	78500	15464.95	17496.77	14.55	28.41

* the numbers may differ slightly for the two schedulers as different jobs may remain in the queue at the end of the simulation.

Table 2: *Simulation results for different job classes. EASY backfilling is better in all cases for the CTC workload, and for long jobs in the Jann workload. Results shown are for load of 0.8.*

4 The Triple Interaction of Metrics, Schedulers, and Workloads

A more difficult situation occurs when three factors are involved in the interaction. An example is provided by the comparison of the conservative and EASY backfilling schemes. It turns out that, at least for some workloads, the response time metric favors EASY backfilling, whereas the (bounded) slowdown metric favored conservative backfilling. Particularly worrying is the fact that the Jann and CTC workloads, which are statistically very similar (the Jann model specifically tries to emulate the CTC workload), produce different results: the Jann model interacts with the metric, and produces opposite results for the two metrics, while the CTC workload favors EASY for both metrics (Table 2). This is therefore actually a triple interaction (denoted ACD in Table 1). The following analysis is based on reference [2].

4.1 Producing Conflicting Results

First, we try to understand the mechanism that causes the different metrics to produce conflicting results. Slowdown is known to be very sensitive to short jobs, as the job runtime appears in the denominator of the formula; thus short jobs that are delayed for even moderate times lead to high slowdown values. The effect of backfilling is also related to job duration, as short jobs have a better chance to fit into a hole in the schedule. Thus tabulating short jobs separately may lead to important insights.

Table 2 shows the results, defining short jobs as those shorter than one hour. For the CTC workload, both metrics favor EASY backfilling for each class individually, and also for both of them together. But in the Jann workload we indeed see a difference that depends on job class. For jobs that are longer than 1 hour, both metrics favor EASY. But for the shorter jobs both metrics favor conservative backfilling.

Given that for each job class both metrics agree, how does this turn into conflicting results

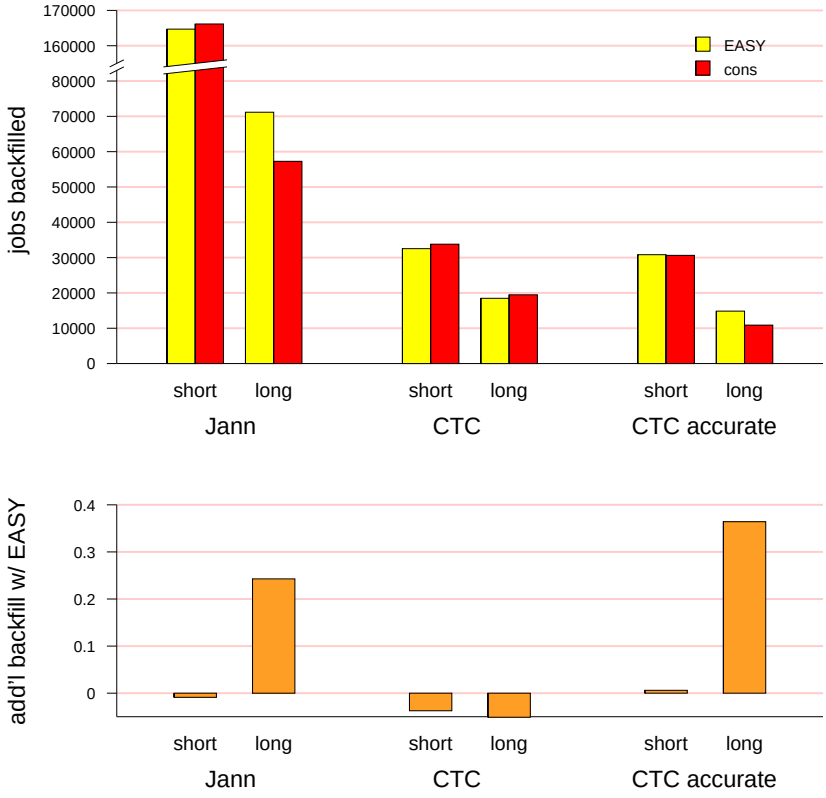

Figure 4: *Amount of backfilling (top) and increased backfilling with EASY relative to conservative (bottom). A value of 0.1 means that EASY did 10% more backfilling.*

when the whole workload is considered? The answer is that averages are dominated by the higher values. For response times the high values come from the long jobs, whereas when we calculate the average slowdown the high values come from the short jobs. Thus the average response time is similar to the response time for long jobs, which favors EASY, whereas the average slowdown is similar to the slowdown of short jobs, which favors conservative.

4.2 Underlying Performance Differences

For both workloads, the response times and slowdowns of long jobs under the EASY scheduler are substantially lower than under conservative scheduling. The difference in the results is due to the short jobs, that fare better under conservative in the Jann workload, but not in the CTC workload. To try and understand why this happens, we need to understand how the scheduler interacts with the workload.

Figure 4 shows the amount of backfilling achieved by the two Schedulers. Surprisingly, it turns out that the better performance for short jobs in the Jann/conservative combination is not the result of more backfilling. On the contrary, the main difference between the workloads is that under Jann/conservative there is less backfilling of *long* jobs. This result is most likely the consequence of a seemingly minor difference between the workloads: the fact that the CTC workload includes

user estimates of runtime that are used in the backfilling process, whereas the Jann model does not. Simulations based on the Jann model therefore use the actual runtime as an estimate. This leads to much less backfilling under the conservative scheme, because backfill jobs must fit into the smaller space that is left available by the tighter estimates.

To confirm this hypothesis, we re-ran the CTC simulations but using the actual runtimes rather than the original user estimates to control the backfilling. The results, also shown in Figure 4, confirm the conjecture. Moreover, in these runs the performance of short jobs was better under conservative than under EASY (as in the Jann workload), indicating that the disparity in backfilling long jobs is determinative for the performance of short jobs.

But how does the reduced backfilling of *long* jobs under conservative translate into better performance for *short* jobs? The answer is that in both workloads, many long jobs are serial. They are therefore prime candidates for backfilling. The question of whether backfilling will actually occur depends on the scheduler. EASY will backfill provided the job does not delay the first job in the queue. The conservative scheduler is stricter, and requires that no previously queued job be delayed. Given that the jobs in question are long, there is a significant danger that they delay some job even if they do not delay the first queued job. Short jobs that are thus delayed will suffer from high slowdown values. Thus by achieving less backfill for large jobs, conservative avoids delays for short jobs, resulting in better slowdown scores.

4.3 Discussion

To summarize, our analysis exposed the following triple interaction:

- The Jann and CTC workloads differ (among other things) in that the CTC workload is a real trace including user estimates of runtime, whereas the Jann model does not include this detail.
- Due to using accurate estimates for the Jann model, the conservative scheduler achieved less backfilling of long jobs that use few processors. This is obviously detrimental to the performance of these long jobs, but turned out to be beneficial to short jobs that don't get delayed by the long jobs.
- As response time is dominated by long jobs, the response time metric showed that EASY is better than conservative for the Jann workload. The slowdown metric, on the other hand, is dominated by short jobs, so it showed conservative to be better.

As real workloads have inaccurate runtime estimates [10], it seems that in this particular case the CTC results should be favored over the Jann results, leading to an unequivocal preference of EASY over conservative. However, this hinges on the very high number of long serial jobs, which is unique to the CTC machine and Jann workload (which is based on it) due to its history: it replaced a large mainframe, and inherited the mainframe's workload. Thus the results may actually not be representative.

The results of the analysis are interesting also because of what it didn't find. Specifically, seemingly important features of the workload turned out to be unimportant. An example is the details of the runtime distribution in the two models. The CTC workload is bounded at about 18 hours (an administrative issue), whereas the Jann workload has a tail that extends beyond 30 hours.

The CTC workload hardly has any jobs shorter than 30 seconds, probably due to the fact that the measurement includes the time to start up the required processes on all the nodes, and to report their termination. In the Jann model, by contradistinction, over 10% of the jobs are shorter than 30 seconds, and many jobs only run for a fraction of a second. However, rerunning the simulations on a truncated version of this workload, in which all jobs shorter than 30 seconds or longer than 18 hours were removed, did not change the results much.

Another major difference between the workloads is that in the original CTC workload most jobs use power-of-two nodes, whereas in the Jann model jobs are spread evenly between each two consecutive powers of two. Previous work has shown that the fraction of jobs that are powers of two is important for performance, as it is easier to pack power-of-two jobs [8]. However, in our case this seemed not to make a qualitative difference. It was checked by running the simulations on a modified version of the Jann workload in which the sizes of 80% of the jobs were rounded up to the next power of two.

5 Conclusions

An unstated assumption of performance evaluation is that the measured results are largely due to the systems begin studied. Another part of the variation is assumed to be related in the predictable way to the load conditions. Our simulations and analysis indicate that this is not necessarily the case: both the metrics and the workloads being used may have a large effect on the results, as well as interactions between these factors.

The reason for this is that real systems and real workloads are rather complex. When evaluating their performance, all sorts of unexpected interactions occur, and all sorts of problems that were thought to be marginal actually play a larger role than expected. Great care must be taken to understand such interactions, and to avoid them when they compromise the validity of the results.

In term of practical advice, our results indicate that it is desirable to use all relevant metrics and all available workloads for performance comparisons, and not settle for just one combination. If conflicting results are observed, this is an indication that a detailed study is needed in order to determine which results deserve to be given more weight. This can start by comparing the workloads, and trying to identify the differences between them. These differences can then be inspected for possible interactions with the various metrics, based on a thorough understanding of the domain. However, suspect interactions must be checked carefully, as complex systems tend to breed unexpected and counter-intuitive effects.

Acknowledgments

This research was supported in part by the Israel Science Foundation (grant no. 219/99). The workload models and logs on which it is based are available on-line from the Parallel Workloads Archive. Many thanks to all those who have made their data and software available.

References

[1] W. Cirne and F. Berman, "*Adaptive selection of partition size for supercomputer requests*". In *Job Scheduling Strategies for Parallel Processing*, LNCS vol. 1911, pp. 187–207, Springer Verlag, 2000.

[2] D. G. Feitelson, *Experimental Analysis of the Root Causes of Performance Evaluation Results: A Backfilling Case Study*. Technical Report 2002–4, School of Computer Science and Engineering, Hebrew University, Mar 2002.

[3] D. G. Feitelson and L. Rudolph, "*Evaluation of design choices for gang scheduling using distributed hierarchical control*". *J. Parallel & Distributed Comput.* **35(1)**, pp. 18–34, May 1996.

[4] 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*, LNCS vol. 1291, pp. 1–34, Springer Verlag, 1997.

[5] J. P. Jones and B. Nitzberg, "*Scheduling for parallel supercomputing: a historical perspective of achievable utilization*". In *Job Scheduling Strategies for Parallel Processing*, LNCS vol. 1659, pp. 1–16, Springer-Verlag, 1999.

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

[7] 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.

[8] 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*, LNCS vol. 1459, pp. 25–46, Springer Verlag, 1998.

[9] D. C. Montgomery, *Design and Analysis of Experiments*. John Wiley & Sons, 5 ed., 2001.

[10] 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)**, pp. 529–543, Jun 2001.

[11] L. Rudolph and P. Smith, "*Valuation of ultra-scale computing systems*". In *Job Scheduling Strategies for Parallel Processing*, LNCS vol. 1911, pp. 39–55, Springer Verlag, 2000.

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

Biography: Dror G. Feitelson received the BSc degree in mathematics, physics, and computer science in 1985, the MSc degree in Computer Science in 1987 (summa cum laude), and the PhD in 1991, all from the Hebrew University. He is on the faculty of the School of Computer Science and Engineering at the Hebrew University of Jerusalem, Israel, and heads the parallel systems laboratory. He is a senior member of the IEEE and IEEE computer society, and a member of the ACM.

His research focuses on parallel job scheduling and on workload modeling. He maintains the parallel workloads archive, with the goal of promoting the characterization and modeling of workloads on production parallel machines, and the use of solid experimental methods for the evaluation of parallel job scheduling policies. He is the founding co-organizer of a series of workshops on the topic of Job Scheduling Strategies for Parallel Processing, held annually since 1995.