Are user runtime estimates inherently inaccurate?

Cynthia Bailey Lee, Yael Schwartzman
Jennifer Hardy, Allan Snavely
San Diego Supercomputer Center
March 2004

Abstract

Computer system batch schedulers typically require information from the user upon job submission, including a runtime estimate. Inaccuracy of these runtime estimates, relative to the actual runtime of the job, has been well documented and is a perennial problem mentioned in the job scheduling literature. Typically users provide these estimates under circumstances where their job will be killed after the provided amount of time elapses. Also, users may be unaware of the potential benefits of providing accurate estimates, such as increased likelihood of backfilling. This study examines user behavior when the threat of job killing is removed, and when a tangible reward is provided for accuracy. We show that under these conditions, about half of users provide an improved estimate, but there is not a substantial improvement in the overall average accuracy.

1 Introduction

It is a well-documented fact that user-provided runtime estimates are inaccurate. Characterizations of this error in various real workload traces can be found in several classic and recent papers. Cirne and Berman [1] showed that in four different traces, 50 to 60% of jobs use less than 20% of their requested time. Ward, Mahood and West [7] report that jobs on a Cray T3E used on average only 29% of their requested time. Chiang, Arpaci-Dusseau and Vernon [4] studied the workload of a system where there is a 1-hour grace period before jobs are killed, but found that users still grossly overestimate their jobs' runtime, with 35% of jobs using less than 10% of their requested time (includes only jobs requesting more than one minute). Similar patterns are seen in other workload analyses [2,3,5].

Many factors contribute to the inaccuracy of user estimates. All workloads show a significant portion of jobs that crash immediately upon loading. This is likely more indicative of users' difficulties with configuring their job to run correctly, than difficulties with providing accurate runtime estimate [2]. However, a job's runtime may also vary from run to run due to load conditions on the system. In an extreme example, Nitzberg and Jones [9] found that on an Origin system where different jobs on the same node share memory resources, job runtime varied 30% on a lightly loaded system, to 300% on a heavily loaded system.

Mu'alem and Feitelson [2] note that because many systems kill jobs after the estimated time has elapsed, users may be influenced to "pad" their estimates, to avoid any possibility of having their job killed. Therefore, we believe that it is important to be precise about what users are typically asked to provide, which is a time after which they would be willing to have their jobs killed, and to distinguish this from the abstract notion of an estimate of their jobs' runtime. This leads us to prefer the term, *requested runtime* for the former, reserving the term *estimated runtime* for a best guess the user can make without any penalty (and possibly even with an incentive for accuracy).

This paper focuses on two specific causes of error in user provided runtime estimates:

(1) Requested runtimes are used as a "kill time" in other words, jobs are killed after the provided time has elapsed.

(2) Users may be insufficiently motivated to provide accurate runtime estimates. Many users are likely unaware of the potential benefits of providing an accurate request, such as higher probability of receiving quicker turnaround (because of an increased likelihood of backfilling), or this motivation may not be strong enough to elicit maximum accuracy.

A significant unanswered question is, can and would users be accurate if these two barriers to accuracy were removed? This study addresses this question by asking users of the Blue Horizon system at the San Diego Supercomputer Center (SDSC) [8] for a non-kill-time estimate of their jobs' runtime, and offering rewards for accuracy.

The rest of the paper is organized as follows. In Section 2, we describe the experiment design. In Sections 3 and 4 we present the results of the accuracy of users' non-kill estimates, and their confidence in their estimates, respectively. Section 5 reviews related work on the impact of user inaccuracy on scheduler performance. Finally, Sections 6 and 7 present the conclusions and future work.

2 Survey Experiment Design

Users of the Blue Horizon system submit jobs by using the command *llsubmit*, passing as an argument the name of a file called the job *script.* The script contains vital job information such as the location and name of the *executable*, the number of *nodes* and *processors* required, and a *requested runtime*. An analysis of the requested runtimes from the period prior to the experiment shows that the error has a similar distribution to that observed in other workloads. Specifically, a majority of jobs use less than 20% of their requested time.

During the survey period, users were prompted for a non-kill-time estimate of their jobs' runtime by the llsubmit program, randomly one of every five times they run. We asked, at the moment of job submission, hoping that this will be the most timely and realistic moment to measure the user's forecasting abilities. The traditional requested runtime is not modified in the job script, we merely reflect that value back to the user and ask them to reconsider it, with the assurance that their response in no way affects this job.

Users were notified of the study, by email and newsletter, a week prior to the start of the survey period. The notification included information about prizes to reward the most accurate users (with consideration given also to frequency of participation). One MP3 player (64MB Nomad, approximate value: 80 USD) and 18 USB pen drives (64MB, approximate value: 20 USD) were awarded. The prizes were intended to provide a tangible motivation for accuracy and thus to elicit the most accurate estimates users are capable of providing.

The text of the survey is as follows. First, the user is reminded of the requested runtime (kill time) provided in their script. The user is then queried for a better estimate. Finally, the user is asked to rate their confidence in the new estimate they provided, on a scale from 0 to 5 (5 being the highest). This question was designed to test if users could self-identify as good or poor estimators. The survey does not provide default values. A sample of the survey output is shown below in Figure 1.

```
% llsubmit job_script
##################################################################
#   You have been randomly selected to participate in a two-question survey  #
#    about job scheduling <as posted on www.npaci.edu/News>. Your             #
#   participation is greatly appreciated. If you do not wish to participate   #
#   again, type NEVER at the prompt and you will be added to a                #
#   do-not-disturb list.                                                      #
##################################################################
In the submission script for this job you requested a 01:00:00 wall-clock limit.

We understand this may be an overestimate of the wall clock time you expect the job to
take. To the best of your ability, please provide a guess as to how long you think your job
will actually run.
**NOTE: Your response to this survey will in no way affect your job's scheduling or
execution on Blue Horizon.
Your guess (HH:MM:SS)? 00:10:00
Please rate(0-5) your confidence in your guess: (0 = no confidence, 5= most confident): 3
Thank you for your participation.
Your Blue Horizon job will now be submitted as usual.
```

Figure 1. Sample user survey and response.

3 User Accuracy

Over the 9-week period of the survey there were 10,397 job submissions. However, only 2,870 of those ran until completion (many jobs are withdrawn while still waiting in the queue or cancelled while running). Since approximately one out of every five job submissions were requested to complete the survey 2,478 of the jobs that ran until completion were not surveyed. Furthermore, we did not survey automated submissions (81) or jobs that requested less than 20 minutes of runtime (172). We had 21 timeouts, where there was no response for more than 90 seconds; and 59 jobs that were submitted by the 11 people that decided not to take part in the survey.

Of the 143 jobs that ran until completion and were selected to complete the survey, 20 had equal or slightly higher runtimes than their requested runtime. This situation could either indicate that the user was very accurate or, more likely, that the job got killed once it reached its requested runtime due to scheduling policies. We decided to discard these survey entries since it was not possible to determine whether the job was completed or killed from the information we collected. In 16 of the responses, the estimate given in response to the survey was *higher* than the requested runtime in the script. Taken at face value, this means that upon further reflection, the user thought the job would need *more* time than they had requested for it, in which case the job is certain to be killed before completing. Some of these responses appeared to be garbage (e.g. "99:99:99") from users who perhaps did not really want to participate in the study or just hoped a random response had some chance of winning a prize. In our analysis, all of these higher responses were discarded, as well as a survey response indicating an expected runtime of 0 seconds.

Fifty-six of the survey response runtime estimates were the same as the requested runtime in the script. Of the 51 responses where users provided a tighter estimate, users cut substantially—an average of 35%—from the requested time. The average *inaccuracy* in this group decreased from 68% to 60%. By inaccuracy we mean the percent of requested (or estimated) time that was unused or exceeded (in the case of estimates it is possible, though unusual, in this survey, to underestimate the runtime), as given in the following formula:

$$Inaccuracy = abs(base - actual\_runtime) / base$$

Where *base* is either the requested runtime or the estimated time from the survey. So for example, a requested time inaccuracy of 68% means either that 32% of the requested runtime was used, or that 168% of the requested time was used.

Because not all users tightened their estimates, overall the inaccuracy decreased from an average of 61% to 57%. Those users who did not tighten their estimate were notably less inaccurate than those who did revise it; their initial inaccuracy was 55%. To fully understand our two metrics it is helpful to understand an example. A not atypical user requested their job to run for 120 minutes, revised (estimated) the runtime at 60 minutes in response to the survey, and the job actually ran for 50 seconds (!). In this example the user tightened their estimate by 50%. But the inaccuracy of the request is 99%, and the inaccuracy of the estimate is improved only 1% down to 98%. Intuitively, many users are substantially improving extreme overestimates, still without making the bounds very tight.

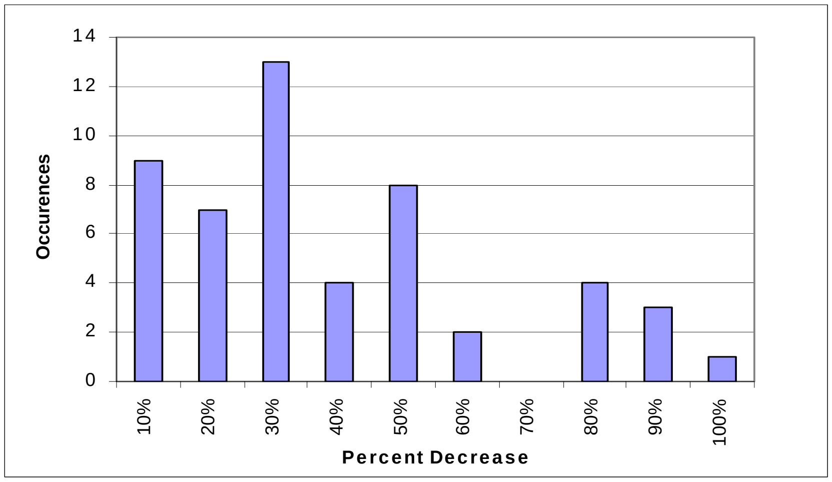

Figure 2. Histogram of percent decrease from the requested time to the estimate provided in response to the survey (includes only responses that were different from the requested time—72 responses had a 0% decrease). Categories represent a number of respondents up to the label, e.g. 20% represents 7 responses that were between 10% (exclusive) and 20% (inclusive) decreased from the requested time in the script.

In Figure 4 we show the comparison between the requested runtime in the script, and the actual runtime for the survey entries. The results are similar to those seen in Figure 3, where we see the same information but for the entire workload during the survey period, suggesting that the survey entries collected are a representative population sample. The results are also similar to those seen in the literature, in particular see [2]. Figure 5 shows the results if the estimate provided in the survey is used, instead of the requested runtime in the script. Note that no job's actual runtime can exceed the requested runtime, but because the survey responses were unconstrained in terms of being a kill time, the actual runtime can be either more or less than this estimate. The great majority of survey responses were still overestimates of the actual runtime. We cannot be sure why this is so, but it may be a lingering tendency due to users having been conditioned to overestimate by system kill-time policies.

Some degree of improvement can be seen in the pattern of error, for example a cluster of points on the right to right-bottom area of Figure 4 is largely dissipated in Figure 5. We can see that users still tend to round their times to 12, 24 and 36 hours in the survey, but not quite as heavily.

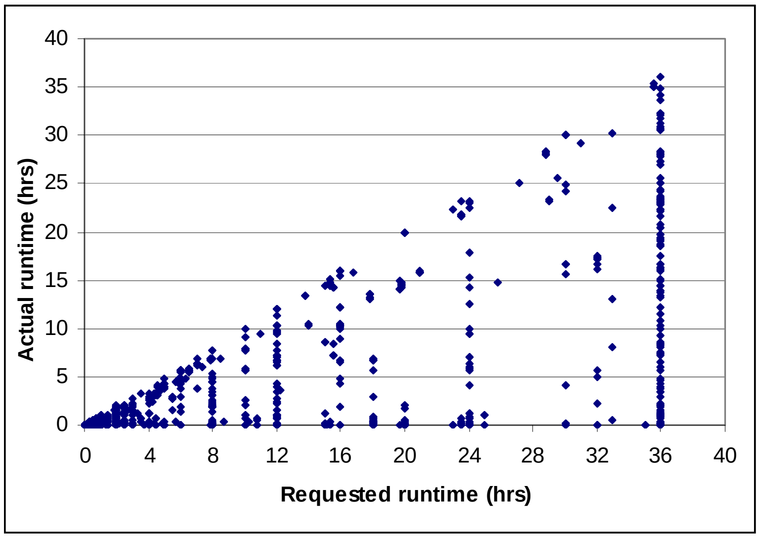

Figure 3. Comparison of actual runtime and requested runtime for all jobs on Blue Horizon during the survey period (Figure 4 shows the same data but only for jobs in the survey.) Note that some data points are overlapping.

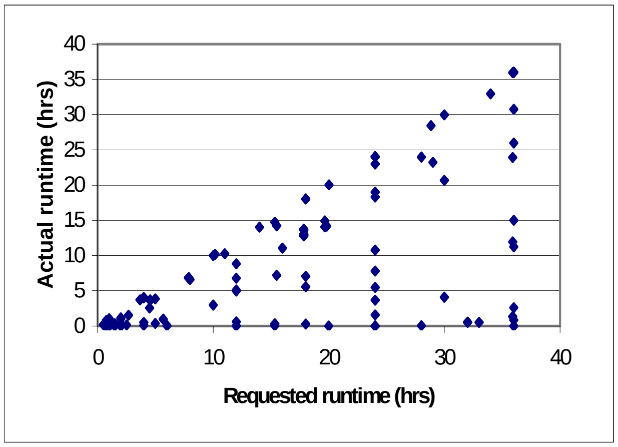

Figure 4. Correlation between requested runtime and actual runtime. Note that some data points are overlapping.

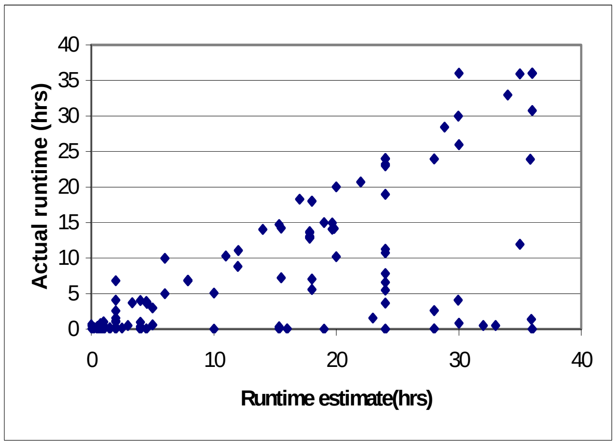

Figure 5. Correlation between users' survey runtime estimates and actual job duration. Note that some data points are overlapping.

4 User Confidence

It is likely that even the most motivated of users will not always be *able* to provide an accurate runtime request or estimate. But it may be useful if users can at least self-identify when they are unsure of their forecast. In our study, we asked users to rate their confidence in the runtime estimate they provided in response to the survey on a scale from 0 (least confident) to 5 (most confident). Figure 6, below, shows the distribution of responses. In a majority (70%) of the responses, users rated themselves as most confident or very confident (5 or 4 rating) in the estimate. This is in spite of the fact that, overall, the accuracy of the requested runtimes and runtime estimates was poor (though typical, as observed in other workloads). It may be that users did not significantly adjust their forecasts of their jobs' runtime to account for possible crashes and other problems [11,12].

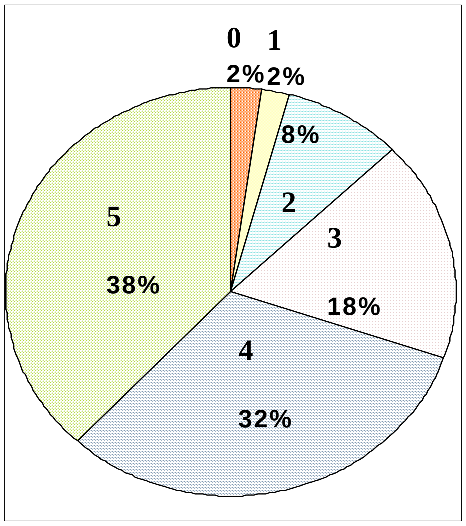

Figure 6. Distribution of user accuracy self-assessments (i.e. confidence).

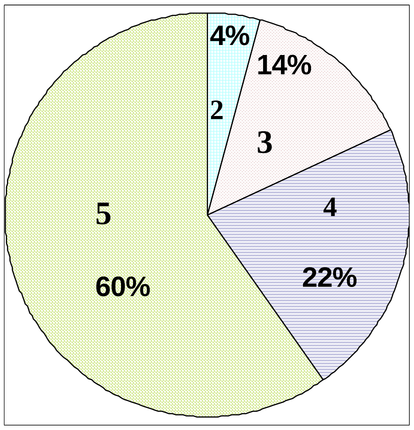

Figure 7a. Distribution of user accuracy self-assessments in users who *did not* change their requested runtime in response to the survey.

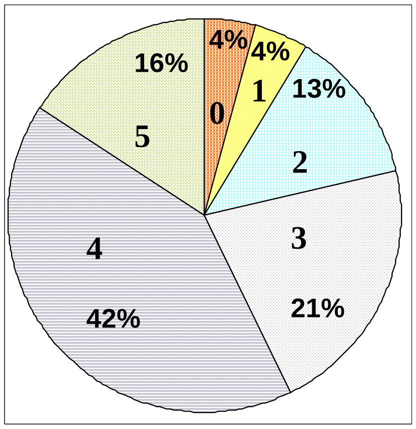

Figure 7b. Distribution of user accuracy self-assessments in users *did* change their requested runtime in response to the survey.

The responses can be divided into those users who provided a revised estimate in response to the survey, and those who reiterated the requested runtime in their script. In Figure 7a, we see that in 60% of responses that were the same as the requested runtime, users rated themselves as most confident (5), with another 22% rated very confident (4). No users rated themselves as low or very low confidence (1 or 0). In contrast, of those responses that were a different estimate (Figure 7b), most users rated themselves somewhere in the middle (4 or 3).

Psychologists Kruger and Dunning [11] have observed that people who are most ignorant of a subject area are *more* likely to overestimate their own abilities than those who are knowledgeable. We wondered if our results were an instance of the same phenomenon. In other words, perhaps users reiterated the same requested runtime out of ignorance, and were then very self-confident, as predicted by Kruger and Dunning. However, it appears that users who did not change in response to the survey, and had high confidence, did on average have more accurate estimates (as seen in Figure 8). For the unchanged responses, there is a clear pattern of decreasing average inaccuracy as the confidence increases. The same pattern is not seen in for those survey responses that were different from the requested runtime in the script. There does not seem to be a strong correlation between these users' confidence and the accuracy of the estimates they gave in the survey.

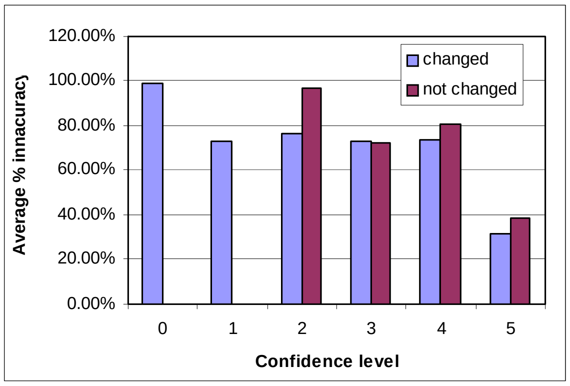

Figure 8. Average percent inaccuracy of user survey responses, separated into those responses that were changed and not changed with respect to the requested runtime in the script.

5 Impact of User Inaccuracy on Scheduler Performance

One might ask what impact user inaccuracy has on scheduler performance—why worry if user estimates are inaccurate? Indeed, some studies have shown that if workloads are modified by setting the requested times to *R * actual runtime*, average slowdown for the EASY and conservative backfilling algorithms actually *improves* when R = 2 or R = 4, compared to R = 1 (total accuracy) [3,14]. Similar results have been shown when R is a random number with uniform distribution between 1 and 2, or between 1 and 4, etc. [2,14].

But simply taking the accurate time and multiplying it by a factor does not mimic the "full badness of real user estimates" [2]. In other studies where real user-provided times were used [2,3], some scheduling algorithms did perform equivalently or slightly better, compared to the same workload with completely accurate times.

However, some other algorithms experience significant performance degradation as a result of user inaccuracy [4,5]. Also, even for an algorithm such as conservative backfilling, which shows some improvement with inaccurate estimates, it is at the cost of less useful wait time guarantees at the time of job submittal, and causing an increased tendency to favor small jobs over large jobs (which may or may not be desirable) [4,14].

Asking the user for a more accurate time, as we have done in this study, is not the only approach to mitigating inaccuracy. One suggestion is to weed out some inaccurate jobs through speculative runs, to detect jobs that immediately crash [5,13]. Or, the system could generate its own estimates for jobs with a regular loop structure, via extrapolation from timings of the first few iterations [4]. Another proposal [6] is to charge users for the entire time they requested, not only the time they actually used. This idea, meant to discourage users from “padding” their estimates, may seem unfair to users and thus be unattractive to implement.

6 Conclusion

Mu'alem and Feitelson [2] documented and modeled discrepancies between user-provided time limits and actual execution time on several HPC systems, including Blue Horizon. We analyze a more recent trace, with similar results. We then ask the question, are users are capable of providing more accurate runtime estimates?

To answer this question, we surveyed users upon job submittal, asking them to provide the best estimate they can of their job’s runtime, with the assurance that their job will not be killed after that amount of time has elapsed.

We have demonstrated that some users will provide a substantially revised estimate but that, on average, the accuracy of their new estimates was only slightly better than their original requested runtime. On the other hand, many users were able to correctly identify themselves as more or less accurate in their estimating than other users.

An inherent weakness in our survey experiment design is that we can never be sure if users are motivated “enough” to provide the best estimates they can. In other words, it is not clear if a bigger or better prize offering would have elicited better estimates from users. However, that most users were very confident in their estimates indicates that perhaps many were in fact exhibiting their “best” in our study.

7 Future Work

In future work, we will measure the impact that better user estimates have on supercomputer performance. We intend to carry out additional surveys to find a scheduling system that understands user behavior and uses this knowledge as a key scheduling factor. The survey will possibly include educating feedback in order to measure user’s improvement over the lifetime of the experiment. In addition, we wish to help users improve their estimates. One possible way to accomplish this is by educating them about the potential benefits of providing accurate estimates, other than the prizes offered specifically for this study. For example, our prototype web-based tool Blue View visually presents the Blue Horizon scheduler’s current plans for running and queued jobs. We hope this tool will give incentive to the users to give shorter time estimates with the promise that their jobs will fit the backfill slots shown in it. Furthermore, this tool will also give the user the opportunity to mold their job according to what is readily available.

8 Acknowledgements

This work was sponsored in part by NSF cooperative agreement STI-00230925 “Data Intensive Grid Benchmarks”, DOE SciDAC award titled “Performance Evaluation Research Center”, DoD HPCMP award titled “HPC Performance Evaluation”, and NSF cooperative agreement ACI-9619020 through computing resources provided by the National Partnership for Advanced Computational Infrastructure at the San Diego Supercomputer Center. We gratefully acknowledge all the users of Blue Horizon who participated in the survey, and the staff who supported the administration of the survey. We wish to thank the workshop referees for their thorough and very useful suggestions. Finally, Dan Tsafrir provided helpful correspondence.

References

[1] Cirne, Walfredo and Fran Berman. "A comprehensive model of the supercomputer workload." *Proceedings of IEEE 4th Annual Workshop on Job Scheduling Strategies for Parallel Processing*. Cambridge, MA. 2001.

[2] Mu'alem, Ahuva W. and Dror G. Feitelson. "Utilization, Predictability, Workloads, and User Runtime Estimates in Scheduling the IBM SP2 with Backfilling," *IEEE Trans. Parallel & Distributed Systems*, 12(6). June 2001.

[3] Srinivasan, Srividya, Rajkumar Kettimuthu, Vijay Subramani and P. Sadayappan. "Characterization of Backfilling Strategies for Parallel Job Scheduling," *Proceedings of 2002 International Workshops on Parallel Processing*, August 2002.

[4] Chiang, Su-Hui, Andrea Arpaci-Dusseau and Mary K. Vernon. "The Impact of More Accurate Requested Runtimes on Production Job Scheduling Performance," *Proceedings of the 4th Workshop on Workload Characterization*, Dror G. Feitelson, Larry Rudolph, Uwe Schwiegelshohn, eds. July 2002.

[5] Lawson, Barry G. and Evgenia Smirni. "Multiple-Queue Backfilling Scheduling with Priorities and Reservations for Parallel Systems," P*roceedings of 8th Workshop on Job Scheduling Strategies for Parallel Processing*, Dror G. Feitelson, Larry Rudolph, Uwe Schwiegelshohn, eds. July 2002.

[6] Stoica, Ian, Hussein Abdel-Wahab and Alex Pothen. "A Microeconomic Scheduler for Parallel Computers," *IPPS'95 Workshop: Job Scheduling Strategies for Parallel Processing*, Dror G. Feitelson and Larry Rudolph, eds. April 1995.

[7] Ward, William A. Jr., Carrie L. Mahood and John E. West. “Scheduling Jobs on Parallel Systems Using a Relaxed Backfill Strategy.” *Proceedings of the 8th Workshop on Job*

Scheduling Strategies for Parallel Processing, Dror G. Feitelson, Larry Rudolph, eds. July 2002.

[8] Blue Horizon, National Partner for Advanced Computing Infastructure (NPACI). www.npaci.edu/Horizon/

[9] Jones, James Patton and Bill Nitzberg. “Scheduling for Parallel Supercomputing: A Historical Perspective of Achievable Utilization.” *Proceedings of the 5th Workshop on Job Scheduling Strategies for Parallel Processing*, Dror G. Feitelson, Larry Rudolph, eds. April 1999.

[10] Kruger, Justin and David Dunning. “Unskilled and unaware of it: How difficulties in recognizing one's own incompetence lead to inflated self-assessments,” *Journal of Personality & Social Psychology,* 77(6). December 1999.

[11] Lovallo, Dan and Daniel Kahneman. “Delusions of Success.” *Harvard Business Review,* 81(7). July 2003.

[12] Buehler, Roger. “Planning, personality, and prediction: The role of future focus in optimistic time predictions.” *Organizational Behavior & Human Decision Processes,* 92(1/2). September/ November 2003.

[13] Perkovic, Dejan and Peter Keleher. "Randomization, Speculation, and Adaptation in Batch Schedulers." *Proceedings of Supercomputing 2000.* November 2000.

[14] Zotkin, Dmitry and Peter Keleher. "Sloppiness as a Virtue: Job-Length Estimation and Performance in Backfilling Schedulers", D.Zotkin, P.Keleher. Proc. 8th HPDC, Redondo Beach, CA. August 1999.