Proceedings of the 2014 Winter Simulation Conference
A. Tolk, S. Y. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller, eds.

A STUDY OF THE IMPACT OF SCHEDULING PARAMETERS IN HETEROGENEOUS COMPUTING ENVIRONMENTS

Sarah Powers

Computer Science and Mathematics Division
Oak Ridge National Laboratory
1 Bethel Valley Road
Oak Ridge, TN 37831, USA

ABSTRACT

This paper describes a tool for exploring system scheduler parameter settings in a heterogeneous computing environment. Through the coupling of simulation and optimization techniques, this work investigates optimal scheduling intervals, the impact of job arrival prediction on scheduling, as well as how to best apply fair use policies. The developed simulation framework is quick and modular, enabling decision makers to further explore decisions in real-time regarding scheduling policies or parameter changes.

1 INTRODUCTION

A wide variety of computer clusters exist today, varying from supercomputers such as Titan at Oak Ridge National Laboratory (ORNL) to grid computing clusters spread across geographical locations. As High Performance Computing (HPC) continues to expand into the Exascale realm, the computational capability, heterogeneity of these systems and workload diversity increase. Past and on-going research continues to investigate optimal scheduling algorithms for these systems.

On most large systems, a built-in scheduler is responsible for deciding when and where to run compute jobs (i.e., jobs) submitted by users. These individuals may be scientists or engineers with or without a deep understanding of the inner-machine workings or scheduler. In most cases, their primary concern involves running research calculations and obtaining results in a timely manner. The system managers on the other hand, have either a mandate or a vested interest in having well-working, heavily utilized and efficient systems. With some oversight of the system schedulers and the ability to modify certain policies (such as preemption, priority, etc.), these individuals need to satisfy both their mandate and the users, which may require making policy changes without a full understanding of the effect on the system or the users down-wind. Simulation models enable one to run scenarios, performing "what-if" type analysis of real-world systems. In this work, we leverage this capability and demonstrate techniques embedded into a simple tool that can be used in real-time by system level managers to enable more informed decisions about scheduling policies.

Scheduling is a heavily researched topic with a variety of application areas such as production, supply chain, personnel management or traffic patterns. Most research related to job scheduling focuses on the scheduling algorithm itself, primarily attempting to reduce the makespan of jobs (Schwiegelshohn 2011). In this work, a slightly broader view is taken in an attempt to incorporate system knowledge about job

arrivals and show that through the use of prediction and learned arrival patterns, more intelligent scheduling and policies can be established.

As aptly pointed out by Pidd (2012), many techniques are often required to solve a single problem. Through the use of simulation, optimization and time series analysis tools, this work investigates the use of job arrival prediction to provide better scheduling patterns as well as optimal "look-ahead" values for both the prediction and the scheduling frequency windows. A simulator is presented that can be used by a decision maker in order to enable real-time decisions regarding 1) if/when prediction is useful, 2) additional policies and their levels that may be necessary to ensure continued fair use policy and 3) test ideas that may or may not lead to increased system performance and user satisfaction.

The paper is organized as follows: section 2 presents the background to the problem and introduces the underlying Mixed Integer Programming (MIP) approach. Section 3 describes the basic methodology and algorithms. Section 4 presents the simulation findings, followed by the general conclusions and next steps in section 5.

2 BACKGROUND

As evidenced in the survey provided by Potts and Strusevich (2009), the research on scheduling has a wide span of contributing disciplines and solution approaches, with recent work focusing on heuristic techniques. The majority of the investigations with respect to scheduling computing jobs focuses on one of a few key objectives: minimizing the makespan of jobs (Schwiegelshohn 2011), maximizing the utilization of the system (Samuel et al. 2011) or more recently minimizing the energy use of the system (Nesmachnow et al. 2013, Li 2012). Each represents a primarily management driven objective.

User satisfaction, an often overlooked metric by system managers, is critical in the sense that unsatisfied users are like customers who do not return and spread a bad reputation for the system. Without users, the system becomes obsolete. This requires maintaining a delicate balancing act between system administrator objectives and user desires. In more recent research, the issue of scheduling fairness has been studied (Schwiegelshohn and Yahyapour 2000), both from the user perspective (Kluscek and Rudov 2013) and the resource usage angle (Ren, He, and Xu 2012).

Initial scheduling research implements off-line algorithms or those requiring one-time scheduling of a batch of jobs. As noted by Blanco et al. (2012), many approaches (on-line primarily) consider each job to be scheduled sequentially without considering the impact on subsequent jobs, likely resulting in sub-optimal schedules. To counter this issue, the approach described here implements an on-line algorithm for real systems where jobs arrive continuously and complete at various intervals, thus requiring constant problem re-optimization. This is done using a Mixed-Integer Programming (MIP) approach to provide an optimal scheduling of the current set of jobs in the queue coupled with discrete event simulation to simulate longer term system behavior. This pairing provides the necessary tools to develop a framework within which to investigate the impact of scheduling parameters.

3 PROBLEM DESCRIPTION

Consider a multi-cluster heterogeneous system comprised of various types of systems (machines). Users submit compute jobs at different times to a queue for processing by the HPC system. These tasks may require different types of computing capabilities and take differing amounts of time to process based on the job type and the machine on which it is run. Execution times are assumed to include some measure of stochasticity. The scheduler runs at pre-specified time intervals processing as many jobs as possible from the queue.

This problem has been shown to be NP-hard, but for small job/machine sets, it can still be solved optimally in a reasonable time frame. When computation time becomes unsuitable, the MIP can easily be replaced by a scheduling heuristic.

3.1 Problem parameters

Determining user satisfaction is a difficult if not somewhat arbitrary task. To incorporate this metric into the model, the concept of job utility is employed (Jensen, Locke, and Tokuda 1985, Lee and Snavely 2007). Defined as the amount of benefit gained from running a job (from the user's point of view) and assumed to be a decaying function over time, this construct can easily incorporate any functional form.

Utility: each job type "earns" or has varying amounts of utility u_{jk} based on the job completion time (makespan). This serves to incorporate both the notions of makespan (traditional) as well as user satisfaction. As an example, a scientist may submit a job at the end of a work day with several hours of expected completion time. As long as the computations complete by the time they return to work the next morning, the precise end time does not matter. The corresponding utility function is fairly constant for the first few hours after submission, then decays more rapidly as the need for completion increases. Utility functions are based on Subject Matter Expert (SME) input.

Utility functions: Each job has an associated utility function assigned at arrival time with the assumption that $U(j) \geq 0 \ \forall j$. For this research, a set of piecewise linearly decaying utility functions of the form $U(j) = max(aT + b, 0)$ are used where T is the estimated completion time of job j. For additional proposed functional forms see Briceno et al. (2011).

Job classes: Jobs submitted to a system are (mostly) unique. However, jobs can be grouped into categories based on problem type or system requirements to process the job, see Joubert and Su (2012) for some examples of groupings. Let $C = \{C_1, C_2, \ldots, C_{N_c}\}$ denote the job classes where $N_c = |C| \geq 1$ is the total number of classes and job $j \in C_i$ where $C_i \in C$.

Machine types: HPC systems tend to be heterogeneous (e.g., CPU and GPU nodes). Similar to the jobs, machines are grouped by type. Let $MT = \{MT_1, MT_2, \ldots, MT_{N_{mt}}\}$ denote the machine types where $N_{mt} = |MT| \geq 1$. Let $M = \{M_1, \ldots, M_{N_m}\}$ be the set of machines where $N_m \geq 1$ and $M_i \in MT$. The term "machine" is used loosely here to define a single homogeneous unit.

Expected Time to Compute (ETC): Machine processing times may vary by type and/or job class. The ETC for a job j_{ik} is defined for each class $C_i \in C$ and $MT_k \in MT$. To capture the stochastic nature of runtimes, an error component e_j is introduced for each job.

Arrival times: The modeled system is considered "in use," therefore job arrivals are not static, nor a one time occurrence. Additionally, this scheduling in done on-line, thus jobs that will arrive in the future are unknown. Modeling job arrivals is a whole area of research with a large body of literature (Li and Muskulus 2007, Song, Ernemann, and Yahyapour 2005). Several researchers, such as Minh and Wolters (2011) and Squillante, Yao, and Zhang (1999) note that the arrivals can affect the scheduling decisions and outcome, thus impacting the general research conclusions. In this work two general arrival situations are considered: low arrivals rates such that the machines are underutilized and high arrival rates resulting in an over-prescribed system. The arrival patterns themselves are chosen to be representative of scientific workloads seen in HPC systems such as Titan.

3.2 Mixed Integer Programming model

The core optimization problem can be formulated as a resource allocation (assignment) problem, specifically, a Mixed Integer Program (MIP) to find the optimal schedule for the problem of scheduling jobs on machines providing the highest utility achievable:

$$\max \quad \sum_{j \in Q} \sum_{k \in M} u_{jk} x_{jk} \tag{1}$$

$$\text{s.t.} \quad \sum_{j \in Q} \sum_{k \in M} x_{jk} = \min(N_{jobs}, N_{mfree}) \tag{2}$$

$$\sum_{k \in M} x_{jk} \leq 1 \qquad \forall j \in Q \tag{3}$$

$$\sum_{j \in Q} x_{jk} \leq 1 \qquad \forall k \in M \tag{4}$$

$$x_{jk} \text{ binary} \tag{5}$$

where

Sets
Q set of jobs
M set of machines

Parameters
u_{jk} $j \in Q$, $k \in M$ utility earned by assigning j to machine k

Variables
x_{jk} binary variable assigning job j to machine k

The first constraint forces the system to schedule no more jobs than available machines. In other words, the implicit assumption here is that machines do not contain individual queues and unscheduled jobs remain in the general queue, consistent with real-world systems. The second constraint implies that each job may only be paired with one machine. The last constraint implies that each machine can only run one job at a time. Obviously, this would be relaxed for a case where each machine is composed of multiple nodes/cores. As stated, here "machine" is equivalent to "core," which allows for a flexible model formulation for future extensions to parallel jobs (though additional problem constraints would be needed). The MIP represents a single instance of optimization and is used to find the best possible matching between the current jobs in the queue and available machines.

4 METHODOLOGY

Despite the plethora of scheduling heuristics proposed in the literature, it is common to find a basic First In First Out (FIFO) algorithm implementation in system schedulers, scheduling one job at a time. In addition, most research focuses primarily on the algorithms themselves. The basic MIP described in section 3.2 is coupled with discrete event simulation (DES) to test the system behavior under alternative decision scenarios as well as using different system and scheduler control parameters. Specifically, we develop a tool to be used on-the-fly by decision makers which can be tweaked and modified in real-time to see scheduling policy implications.

4.1 Scheduling window

In an on-line environment, the scheduler runs at fixed time windows w which can be set by the system administrator. Jobs accumulate in the queue between intervals. In most systems, this value is typically around $w = 60$ seconds. To study the impact of w on earned utility, wait times, and machine idle times, Algorithm 4.1 with various values of w is used.

Algorithm 4.1: MIP
At scheduling time S:
Retrieve all pending jobs requiring scheduling from the queue
Gather available resources
Run MIP optimization
For each job j
if j scheduled on m
then run j on m
else
leave j in queue until next scheduling time

4.2 Prediction of job arrivals

In previous work, Shmueli and Feitelson (2005) consider "look-ahead" methods, (i.e., future job arrival prediction) to improve scheduling. Leveraging this idea, a second approach uses the MIP coupled with prediction and DES to improve the scheduling outcome. These predictions can be made up to varying distances into the future, namely $predW$. Assuming known or pre-computed models of arrival, at a given scheduling time S, jobs predicted to arrive between S and $S + predW$ are temporarily added to the queue and the optimal schedule is obtained. To simplify notation, let $predW$ be a multiple of w such that $predW = \alpha w$ where $\alpha \geq 1$ can be optimized.

Algorithm 4.2: MIP with prediction (MIP-P)
At scheduling time S:
Retrieve all pending jobs requiring scheduling from the queue
Predict any jobs arriving in the next $S + \alpha w$ time steps
add to the queue with future arrival times
Gather available resources
Run MIP optimization
For each job j
If arrival time $t > S$
If optimal schedule places j on m
then "hold" m idle until next scheduling slot
Else
If j scheduled on m
then run j on m
Else
leave j in queue until next scheduling time
Discard all predicted jobs from the queue

4.3 Scheduling fairness

Various features have been used to optimize fairness, Ren, He, and Xu (2012) use queue length as a guide, Schwiegelshohn and Yahyapour (2000) perform weighting based on resource consumption, while Kluscek and Rudov (2013) use the mean queue wait time.

To address the issue of fairness in scheduling from the user perspective, an artificial job class with increasingly high utility is added to the model. Similar to Kluscek and Rudov (2013), this relies on the job's time in the queue. If a job has exceeded a *maxWait* threshold time in the system and its utility is low, it joins this artificial class. This creates priority of this job over others. The final utility earned by this job will be its original value, but the total wait time should improve. To avoid assigning absolute priority for

this job on all machines, high utility values are only given on a subset of the machines. The size of this subset can be adjusted and optimized.

Algorithm 4.3: MIP with fairness

```
At scheduling time S:
    Retrieve all pending jobs requiring scheduling from the queue
    For each job j
        If waitTime_j > maxWait
            Set class_j = C_max
    Gather available resources
    Run MIP optimization
    For each job j
        if j scheduled on m
            then run j on m
        else
            leave j in queue until next scheduling time
```

5 NUMERICAL RESULTS

A simulation testbed was developed to investigate the parameters described in section 4. To simulate a moderate size cluster, the setup included 50 heterogeneous machines, 10 different job classes, and linear decaying utility functions. Storage and networking considerations are assumed to be adequate and are not modeled explicitly.

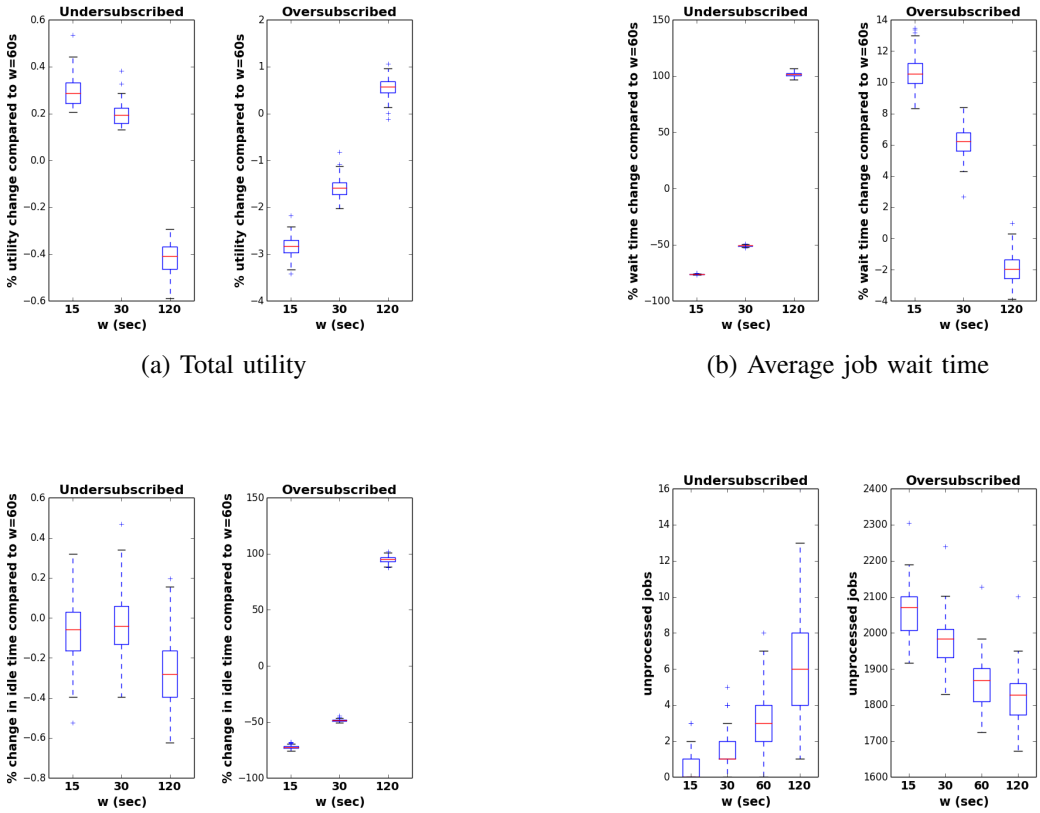
The Estimated Time to Compute (ETC) values were drawn with some stochastic variation from a set of timing tests run on simple codes on different machines types at ORNL. The jobs were simulated with ETC times ranging from 8 to 21 minutes with a mean of 14 minutes and a standard deviation of 3.5 minutes. Two main arrival patterns were used resulting in one system with underutilized machines and one with more job arrivals than the system can process. Through model calibration, it was discovered that a warm up period was necessary in order to achieve steady-state results representative of an "in-use" system. As a result, simulations were run for 18 hours of simulated time. At each scheduling step, the MIP solver from the GLPK toolkit (Makhorin 2012) was used to solve the MIP. The simulation was implemented using a python wrapper script and a minimum of 100 Monte Carlo runs were performed for each of the described cases.

5.1 Scheduling window

The scheduling window was tested with values $w = \{15, 30, 60, 120\}$ seconds respectively. The results are shown for each value in Figure 1.

In the under-provisioned system, the smaller the value of w, the higher the utility earned. Intuitively this makes sense, since the system has free machines, the sooner a scheduling event takes place, the sooner a job starts running and the higher the potential utility earned. For the under-provisioned scenario, it would appear that if a machine is available and a job starts as soon as it arrives, a maximum amount of utility could be earned. Note that this neglects to account for the time and system resources needed to constantly check for machine availability and run the scheduling algorithm. Further analysis indicates that it is not always optimal to schedule a job as soon as it arrives due to trade-offs with more optimal jobs that enter the queue slightly thereafter. In addition, there may be cases where, despite immediate machine availability, the job would be significantly better suited to run on an occupied machine and thus waiting would actually enhance the earned utility.

In the oversubscribed case, the exact opposite appears true. Longer polling intervals lead to higher utility given that the system has more time to accumulate jobs in the queue and thus can schedule more



(a) Total utility

(b) Average job wait time

(c) Average machine idle time

(d) Total unprocessed jobs

Figure 1: Variation of scheduler metrics under different scheduling windows

optimally. This is similar to a long range view. For equivalent reasons, there are fewer unprocessed jobs at higher values of w. One could argue that the "bin-packing" is better. With more jobs being processed, the average wait time decreases. Despite these improvements, the machine idle time increases with larger w given the growing gap between job completion and the next job assignment opportunity. The decision makers will need to weigh the importance of each metric and find the balancing point.

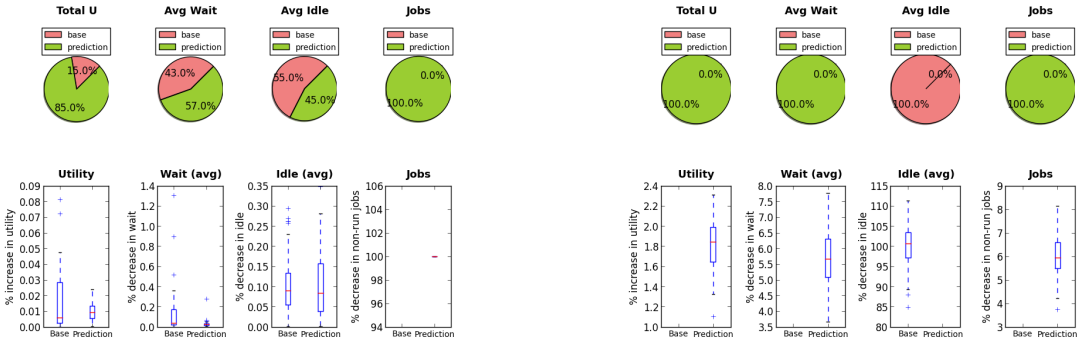
5.2 Prediction impact

Based on the results obtained in studying the scheduling window, a value of $w = 30$ seconds is selected as a balance between high utility, realistic processing time of the scheduler itself and minimizing the machine idle time. The prediction window is set to 30 seconds.

There is a clear distinction between the under-provisioned system and the oversubscribed system (Figure 2). In the former, the improvement obtained using prediction occurs about 85% of the time, but the actual values are very small. Intuitively this makes sense since there are enough machines to satisfy demand even if it results in assignments that are slightly less than optimal. Since the system has plenty of available machines in this case, it is unclear if prediction is truly necessary.

With high arrival rates, the MIP-P algorithm outperforms the simple MIP for all of the runs and obtained close to a 2% improvement in utility. While the resulting schedules are better, this actually produces higher machine idle time on average in every case.

Conceptually, prediction is similar to using larger values of w, yet it allows some jobs to be scheduled sooner, thus reducing the job wait times. As noted in Section 5.1, the jobs can be packed more effectively



(a) Undersubscribed system

(b) Oversubscribed system

Figure 2: Change in scheduler performance using prediction

when having some foreknowledge of future arrivals. This is seen in the reduction of unprocessed jobs, wait times and increased utility.

5.3 Job Fairness

For the Monte Carlo simulations, the *maxWait* threshold is set to a value at which at least half the job classes have reached zero utility. These jobs are passed to an additional class which has high utility on only 10% of the machines in order to not monopolize the system.

MIP-fair outperforms the base case (MIP) in 80% of the iterations. The utility gains (Figure 3) are on par with those using MIP-P. The key item of note is the large improvement obtained in the number of jobs left unscheduled as well as in the job wait time itself. From the user perspective, this results in a more fair assignment over the course of the simulation time with frequent gains and little lost in terms of utility (or user satisfaction). Unfortunately, from an administrative standpoint, the machines spend less time being utilized.

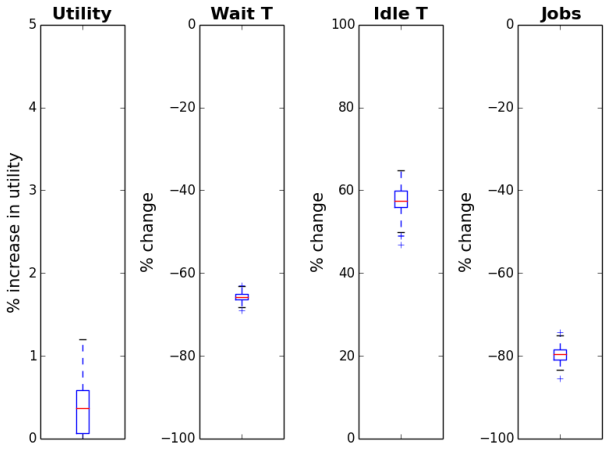


Figure 3: Scheduler performance using an additional priority class

6 CONCLUSIONS

This paper outlined a framework for investigating heterogeneous computing system scheduling parameters. The structure is flexible allowing for the incorporation of new variables, optimization approaches or extensions to different system designs.

The analysis using the developed model showed that current typical scheduling interval settings of 60 seconds should be reconsidered, especially in systems with abundant job arrivals and shorter completion times. Care and further study may be required for environments with longer run times (e.g., jobs with more than a day of compute time) and wide variations in job run times. The value of w is impacted by the average job completion time in the system.

The simulation results outline how the use of future job arrival prediction increases the utility, improves the wait time of jobs and achieves a higher throughput rate for oversubscribed systems. In cases where resources are abundant, this technique may not be necessary.

This work does not claim to have resolved the issue of scheduling fairness for users. The approach presented here, which allows for a small subset of the system to take care of jobs with lengthy wait times, did however show significant improvements if trying to reduce wait time, maximize throughput or increase utility.

The numerical simulation results have only scratched the surface of the host of parameter combinations, machine and job configurations and scheduling assumptions. Nonetheless, this study shows the impact that these factors have on the scheduling results.

For administrators or those unfamiliar with the mathematical underpinnings of scheduling or queuing theory, the tool presented here provides an easy to use modular framework for studying and setting system parameters as well as a process for evaluation of real time policies or on-the-fly decisions and their downstream impact.

ACKNOWLEDGMENTS

This research used resources of the National Center for Computational Sciences at Oak Ridge National Laboratory, supported by the Extreme Scale Systems Center at ORNL, which is supported by the Department of Defense. The author thanks Chris Gröer and Greg Koenig for the initial LP formulation and AMPL code. The author wishes to thank DoD and Greg Koenig for their insight and helpful discussions.

REFERENCES

Blanco, H., F. Guirado, J. L. Lérida, and V. M. Albornoz. 2012. "OAS: A MIP Model for Ordering and Allocating Parallel Jobs on Multi-Cluster systems". In *XIV Congreso Ibero-Latino-Americano de Investigacion de Operaciones*, 3040–3051. Latin-American Association of Operations Research Societies.

Briceno, L. D., B. Khemka, H. Siegel, A. A. Maciejewski, C. Gro er, G. Koenig, G. Okonski, and S. Poole. 2011. "Time Utility Functions for Modeling and Evaluating Resource Allocations in a Heterogeneous Computing System". In *2011 IEEE International Symposium on Parallel and Distributed Processing Workshops*, 7–19.

Jensen, E. D., C. D. Locke, and H. Tokuda. 1985. "A Time-Driven Scheduling Model for Real-Time Operating Systems". In *Proceedings of the 1985 IEEE Real Time Systems Symposium*, 112–122: IEEE Computer Society.

Joubert, W., and S.-Q. Su. 2012. "An Analysis of Computational Workloads for the ORNL Jaguar System". In *Proceedings of the 26th ACM International Conference on Supercomputing*, 247–256. New York, NY, USA: ACM.

Kluscek, D., and H. Rudov. 2013. "Performance and Fairness for Users in Parallel Job Scheduling". In *Job Scheduling Strategies for Parallel Processing*, edited by W. Cirne, N. Desai, E. Frachtenberg, and

U. Schwiegelshohn, Volume 7698 of *Lecture Notes in Computer Science*, 235–252. Springer Berlin Heidelberg.

Lee, C. B., and A. E. Snavely. 2007. "Precise and realistic utility functions for user-centric performance analysis of schedulers". In *Proceedings of the 16th International Symposium on High Performance Distributed Computing*, 107–116. ACM.

Li, H., and M. Muskulus. 2007, March. "Analysis and Modeling of Job Arrivals in a Production Grid". *ACM SIGMETRICS Performance Evaluation Review* 34 (4): 59–70.

Li, K. 2012. "Energy efficient scheduling of parallel tasks onmultiprocessor computers". *The Journal of Supercomputing* 60 (2): 223–247.

Andrew Makhorin 2000-2012. "GNU Linear Programming Kit (GLPK)". Version 4.47, http://www.gnu.org/software/glpk/glpk.html.

Minh, T. N., and L. Wolters. 2011. "Performance impact of job arrivals on clusters and grids through realistic model-based simulation". In *Proceedings of the 2011 International Symposium on Performance Evaluation of Computer Telecommunication Systems*, 22–29.

Nesmachnow, S., B. Dorronsoro, J. Pecero, and P. Bouvry. 2013. "Energy-Aware Scheduling on Multicore Heterogeneous Grid Computing Systems". *Journal of Grid Computing* 11 (4): 653–680.

Pidd, M. 2012. "Mixing other methods with simulation is no big deal". In *Proceedings of the 2012 Winter Simulation Conference*, edited by C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A. M. Uhrmacher, 751–757. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Potts, C. N., and V. A. Strusevich. 2009. "Fifty years of scheduling: a survey of milestones". *Journal of the Operational Research Society* 60:S41–S68.

Ren, S., Y. He, and F. Xu. 2012. "Provably-Efficient Job Scheduling for Energy and Fairness in Geographically Distributed Data Centers". In *Proceedings of the 32nd IEEE International Conference on Distributed Computing Systems*, 22–31.

Samuel, T., T. Baer, R. Brook, M. Ezell, and P. Kovatch. 2011. "Scheduling diverse high performance computing systems with the goal of maximizing utilization". In *Proceedings of the 18th International Conference on High Performance Computing*, 1–6.

Schwiegelshohn, U. 2011. "A system-centric metric for the evaluation of online job schedules". *Journal of Scheduling* 14 (6): 571–581.

Schwiegelshohn, U., and R. Yahyapour. 2000. "Fairness in parallel job scheduling". *Journal of Scheduling* 3 (5): 297–320.

Shmueli, E., and D. G. Feitelson. 2005, September. "Backfilling with Lookahead to Optimize the Packing of Parallel Jobs". *Journal of Parallel and Distributed Computing* 65 (9): 1090–1107.

Song, B., C. Ernemann, and R. Yahyapour. 2005. "Parallel Computer Workload Modeling with Markov Chains". In *Job Scheduling Strategies for Parallel Processing*, edited by D. Feitelson, L. Rudolph, and U. Schwiegelshohn, Volume 3277 of *Lecture Notes in Computer Science*, 47–62. Springer Berlin Heidelberg.

Squillante, M. S., D. D. Yao, and L. Zhang. 1999, March. "The Impact of Job Arrival Patterns on Parallel Scheduling". *ACM SIGMETRICS Performance Evaluation Review* 26 (4): 52–59.

AUTHOR BIOGRAPHIES

SARAH POWERS is a research staff member at Oak Ridge National Laboratory in the Computer Science and Mathematics Division. Her current work focuses on algorithms, predictive modeling and graph analytics in the area of High Performance Computing (HPC) applications. She received her M.S. and Ph.D. degrees in Operations Research & Statistics from Rensselaer Polytechnic Institute, and B.S. in Mathematics from Gordon College. Her research interests span a variety of areas related to Operations Research (OR) and applied mathematics including optimization, algorithm development and scalability, visualization techniques for big data problems, and modeling & simulation.