Discovering the Petascale User Experience in Scheduling Diverse Scientific Applications: Initial Efforts towards Resource Simulation

Lonnie D. Crosby, Troy Baer, R. Glenn Brook, Matt Ezell, and Tabitha K. Samuel
National Institute for Computational Sciences, University of Tennessee at Knoxville

ABSTRACT: Newly emerging petascale computational resources are popular for both capacity and capability computing. However, these varied job classes have widely different resource requirements that make scheduling challenging. Beyond machine utilization, the scheduling of computational resources should provide reasonable throughput for all classes of jobs. This work examines initial efforts to identify the impact of various scheduling policies on different classes of user jobs running on a shared, petascale computing resource with a diverse workload.

KEYWORDS: petascale, scheduling, utilitization, policies, simulation, Moab

1 Introduction

Scheduling the use of a large, shared computational resource is a complex task involving the interaction of multiple factors such as job mix, resource availability, and the operational requirements of both the user community and the resource provider. This is especially true of petascale resources such as Kraken, the Cray XT5 located at Oak Ridge National Lab (ORNL) that is operated by the National Institute for Computational Sciences (NICS) of the University of Tennessee. As a resource provider for the National Science Foundation (NSF), NICS seeks to maintain high levels of resource utilization on Kraken while providing reasonable throughput to both capacity jobs that require 50% or less of the resource and capability jobs that require more than 50% of the resource. To effectively meet this goal, NICS employs a delicate balance of scheduling policies that have evolved over the lifetime of the resource to produce a very unique scheduling environment capable of supporting annual mean resource utilization in excess of 90% of the machine while ensuring mean effective queue durations of less than six hours for most classes of user jobs [1, 2, 3].

In a complex scheduling environment such as that employed on Kraken at NICS, it is extremely important that the scheduling staff understand the effects of scheduling policies on both resource utilization and job throughput. While the composite performance of deployed scheduling policies is easily monitored through the collection and analysis of scheduling statistics, the interaction between scheduling policies is often difficult to accurately predict. As such, the deployment of new or modified scheduling policies is often an extended process fraught with trepidation and risk that is undertaken with great care and constant monitoring. To ease this burden and to improve overall understanding of scheduling complexities, NICS seeks to develop and deploy a scheduling simulation environment that can provide accurate quantitative estimates of the impact of proposed changes to scheduling policies on Kraken. This work presents the initial efforts to establish such a simulation environment at NICS.

2 Objective and Implementation

The purpose of this work is to investigate the effect of scheduling policies on resource utilization and job throughput via appropriate models for job mix, resource availability, and job queues. With these models in place, simulations are performed to determine the impact of scheduling policy changes. Appropriate data collection and analysis methods are employed to obtain quantitative results.

2.1 Resource Scheduler

Like many large XT systems, Kraken uses the Moab scheduler on top of the open source TORQUE batch environment [4, 5]. Moab is an extremely powerful and flexible commercial scheduling software package that supports a wide variety of batch environments, including all PBS variants (such as TORQUE), LSF, LoadLeveler, and SLURM. Moab also supports a number of advanced scheduling capabilities such as advance reservations, quality of service (QoS) levels, consumable resource management, and a highly configurable priority and policy engine. On Cray XT/XE systems, Moab communicates with ALPS as well as TORQUE by interfacing to a native resource manager, a set of glue layer scripts that communicate with both ALPS and TORQUE services.

One of Moab's many features is a simulation mode that can be used to make projections or evaluate "what if?" scenarios [6]. This requires two inputs, a resource trace describing the simulated system and a workload trace describing the simulated workflow on the system. One of the workload trace formats usable by Moab simulation is Moab's own event log, so constructing a historical workload trace from a system running Moab simply requires concatenating and sorting the Moab event logs from the period of interest [7]. Similarly, a resource trace for a system running Moab can be created by dumping the output of "mnodectl -q wiki ALL" to a file [8].

2.2 Scheduling Policies

The scheduling policies on Kraken have been refined over time to optimize utilization while supporting both capability and capacity jobs. The scheduler is configured to use job size as the predominant factor in priority calculation. Under normal operating procedures, the eligible job with the highest priority automatically receives a reservation to hold nodes. Backfill is configured in a mode called firstfit that evaluates jobs in descending priority order to determine which jobs are eligible to be started, allowing large jobs to be scheduled first and smaller jobs to fit in the cracks that are left over. The scheduler only considers five jobs per user and ten jobs per project account as "eligible" to run at any given time; any jobs in excess of these limits are marked as blocked. A debug reservation exists for eight hours each weekday to support quick turnaround for jobs that request two hours or less of wallclock time. Data staging and archiving jobs are supported via "sizezero" jobs that request no compute cores and run on service nodes with a priority boost that allows them to start immediately. Finally, jobs charged against project accounts with negative allocation balances are allowed to run under the "negbal" quality of service, but the Moab scheduler heavily penalizes the priority of such jobs, making them the last jobs considered for backfill.

Capability jobs are subject to additional policies and restrictions due to the high utilization cost to drain the machine for them to run. To minimize the impact of drain time on utilization, NICS employs a bimodal scheduling scheme that accommodates both capability and capacity jobs while improving system utilization [3]. Under this scheme, capability jobs are prevented from running outside specific periods of time, referred to as capability periods, that typically occur immediately after system maintenance periods or during the latter part of weeks in which there are enough hours of queued capability jobs to justify the system drain. During a capability period, capability jobs run within a capability reservation that typically dedicates around 86% of the compute cores to capability jobs for a duration of around 72 hours. Capability jobs are typically scheduled in descending order based on the number of requested cores, and the size of the capability reservation is periodically reduced when possible to free idle nodes for use by non-capability jobs. If the scheduled capability jobs end earlier than expected, the capability reservation is released, and the machine reverts entirely to capacity computing.

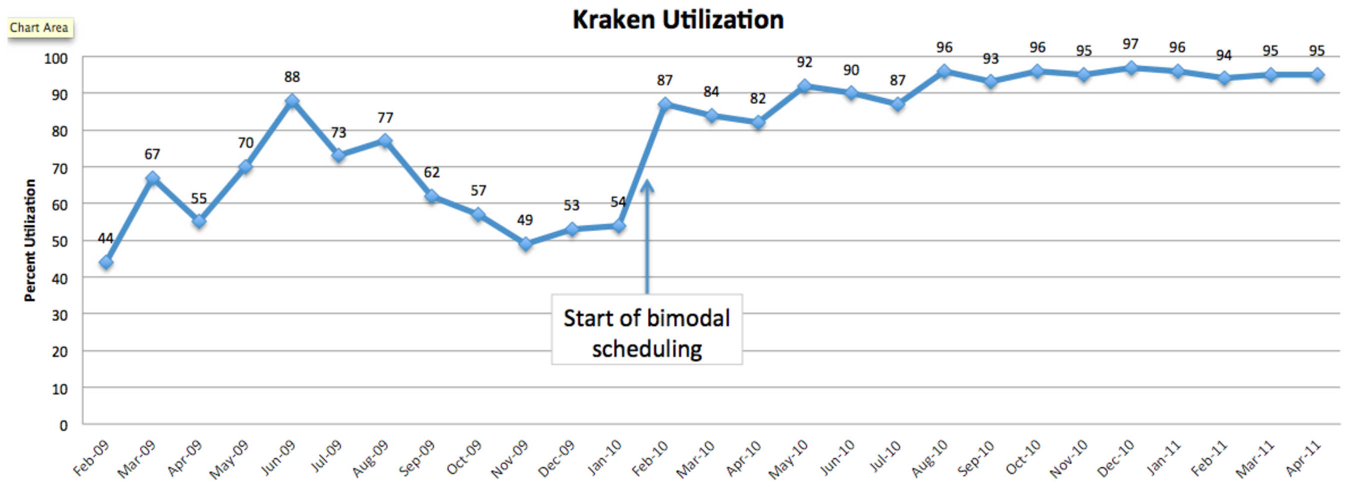


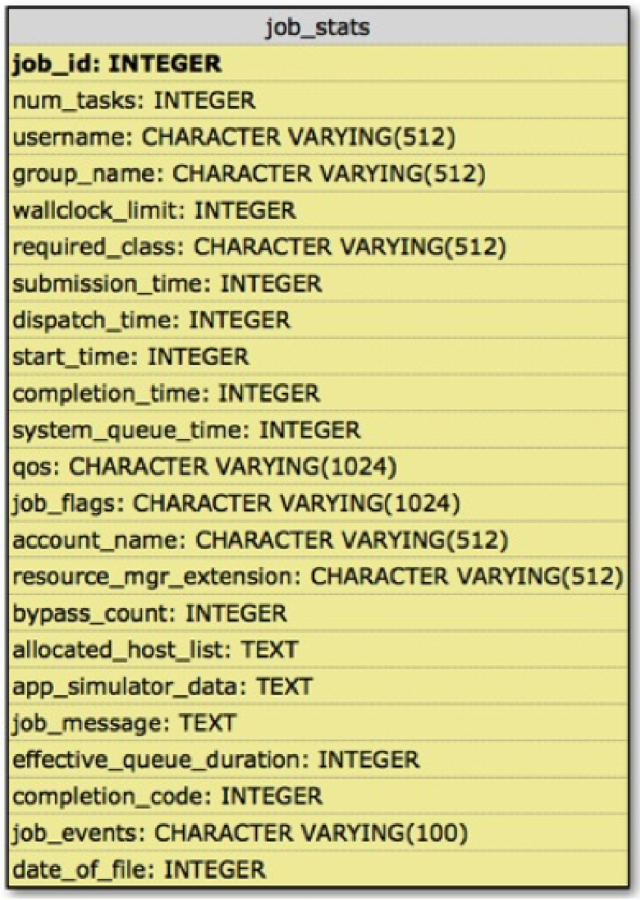
Figure 1: The utilization of Kraken over the lifetime of the Cray XT5 at NICS.

The scheduling polcies presented here allow NICS to achieve very high utilization while maintaining good service for all job sizes. Figure 1 shows the historic utilization over the life of the XT5.

2.3 Scheduling Statistics

2.3.1 Database

The database used to hold job data is based on the Moab Workload Event Record Format (v 5.0.0) [7]. The database table has the structure indicated in Figure 2, with each job having exactly one entry in the table. Fields completion_code, job_events and date_of_file are computed based on the data from the Moab log files. Each job in the log files can have a combination of JOBSTART, JOBEND and JOBCANCEL events. The completion_code field is an integer field based on the job events that occurred for a given job, and the job_events field is a comma-separated string of those job events. The date_of_file field contains the date of the log file where the first job event for a job occurred.



job_stats
job_id: INTEGER
num_tasks: INTEGER
username: CHARACTER VARYING(512)
group_name: CHARACTER VARYING(512)
wallclock_limit: INTEGER
required_class: CHARACTER VARYING(512)
submission_time: INTEGER
dispatch_time: INTEGER
start_time: INTEGER
completion_time: INTEGER
system_queue_time: INTEGER
qos: CHARACTER VARYING(1024)
job_flags: CHARACTER VARYING(1024)
account_name: CHARACTER VARYING(512)
resource_mgr_extension: CHARACTER VARYING(512)
bypass_count: INTEGER
allocated_host_list: TEXT
app_simulator_data: TEXT
job_message: TEXT
effective_queue_duration: INTEGER
completion_code: INTEGER
job_events: CHARACTER VARYING(100)
date_of_file: INTEGER

Figure 2: The structure of the database table containing the collection of job statistics.

2.3.2 Reports

Six reports are typically generated to give a clear picture of the performance of jobs in the simulation. Jobs that run from allocations with negative balance and jobs that run from the speciality queues are ignored except during utilization reporting. The first report plots the effective queue duration (the amount of time, in seconds, that the job was eligible for scheduling) against core count. The core count is done in logarithmic bins of 12. It is ensured that bin boundaries do not cross queue boundaries so that specific queue behavior can be noticed. The mean, median, minimum, maximum and standard deviation of the effective queue duration are calculated, as is the total number of jobs for each bin. Figure 3 presents a sample graph of the Mean of Effective Queue Duration by the number of Cores for a simulation run.

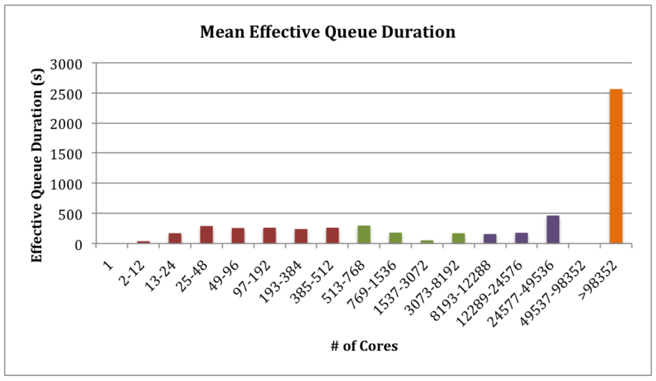


Figure 3: An example report comparing the mean effective queue duration against requested number of cores.

The second report plots the Effective Queue Duration against the Wallclock limit requested by jobs. Wallclock limits are binned in spans of 2 hours up to 24 hours, which is the maximum wallclock limit on Kraken for regular jobs. For capability and dedicated jobs, the limits are up to 60 hours. Binning is done in bins of 12 hours for jobs having wallclock limit of greater than 24 hours. Again, the mean, median, minimum, maximum and standard deviation of the effective queue duration are calculated for each bin. Figure 4 is a sample graph of the Median of Effective Queue Duration by the Wallclock limit requested for a simulation run.

The third report plots the effective queue duration against both Wallclock limit and Core count. Wallclock limits are binned in spans of 2 hours up to 24 hours and 12 hours beyond that. Core counts are binned in logarithmic bins of 12. Jobs are first binned by core count, and each bin is then subdivided by wallclock limit requested. The mean, median and total numbers of jobs in each bin are then calculated. A sample graph of the median of Effective Queue duration by wallclock limit requested and core count is shown in Figure 5.

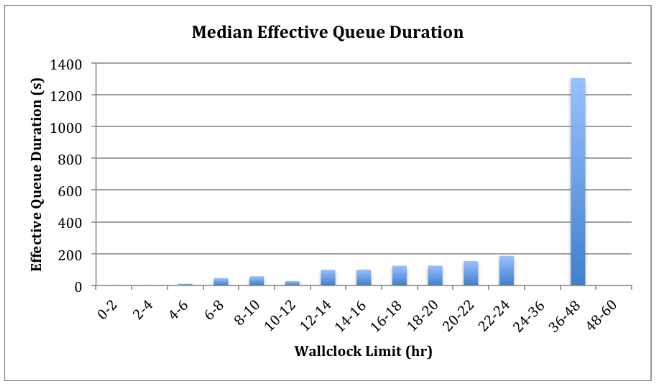


Figure 4: An example report comparing the median effective queue duration against requested wall clock limit.

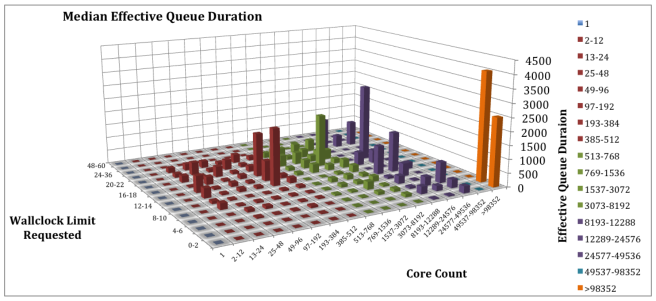


Figure 5: An example report comparing the median effective queue duration against both requested wall clock limit and requested number of cores.

The fourth report plots the bypass count against wallclock limit requested and core count. Bypass count is the number of times a job was bypassed by lower priority jobs via backfill. Jobs are binned by core count first and then by wallclock limit requested. The mean and median of the bypass count are calculated for each bin. A sample graph of the median of bypass count by wallclock limit requested and core count is shown in Figure 6.

The fifth report is the fraction of wallclock limit not used by jobs plotted against wallclock limit requested and core count. The mean of the values are used to calculate the percentage in each bin. A sample report is presented in Figure 7.

The sixth report is a utilization report for the machine. Utilization is calculated for the time period between specified dates in steps of a specfied time delta value. The report takes into account down time, such as time taken for preventative and emergency maintenance. Utilization is calculated based on the start and end times of jobs. A sample graph of utilization is shown Figure 8.

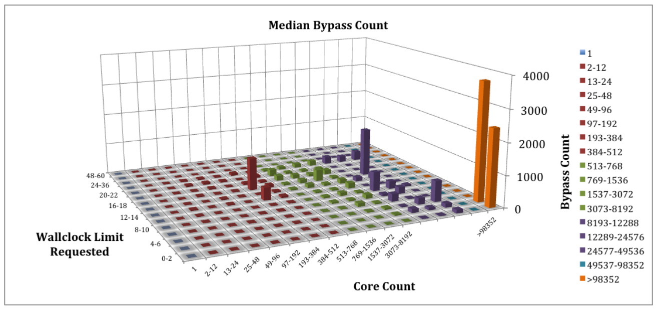


Figure 6: An example report comparing the median bypass count against requested wall clock limit and request number of cores.

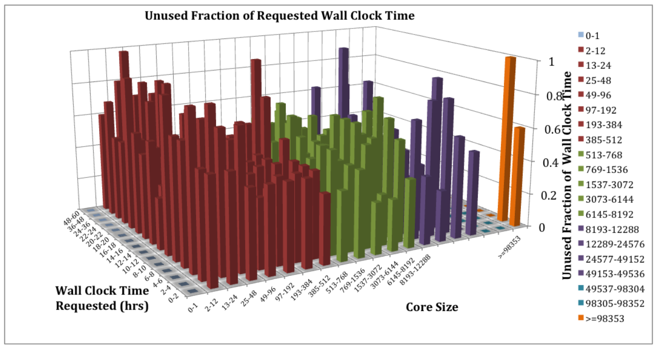


Figure 7: An example report showing the unused fraction of wall clock time by requested wall clock limit and requested number of cores.

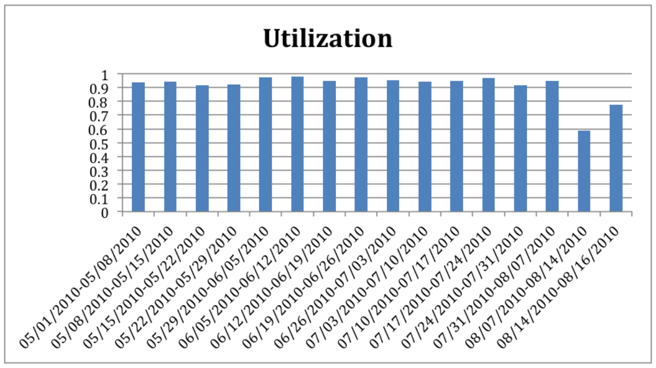


Figure 8: An example report showing the fractional utilization of the resource for a given period of time.

2.4 Simulation Experiment

The workload and resources traces used in the simulations presented here are taken from historical data for Kraken from May to December 2010. Both traces have been filtered to some degree to simplify the simulation. In the resource trace, service nodes were removed, as only jobs using compute nodes were of interest. Similarly, size=0 jobs were removed from the workload trace, as these do not use compute nodes. Also, since the simulations were intended to model the effects of policy changes on Kraken's capacity workload and since capability runs on the system involve a significant degree of manual intervention that would be difficult to simulate, all capability and dedicated jobs were removed from the workload trace. The configuration of the Moab simulation instance used to generate the baseline simulation was as close as possible to the production Moab configuration on Kraken, aside from changes needed specifically for simulation mode. The simulation was configured to have a constant queue depth of 1,000 jobs, as this is roughly what is seen in normal operation on Kraken.

However, configuring a simulation instance of Moab is more difficult that it appears at first blush. The documentation for Moab simulation mode is vague and poorly maintained, and there are a number of less-than-obvious behaviors that were discovered in the course of this project [6]. For instance, the Moab documentation does not mention that the POLLINTERVAL parameter, which controls how often Moab runs its scheduling iteration in normal operation, controls the simulation time step in simulation mode. While one would typically configure POLLINTERVAL to 30 to 60 seconds in normal operation, in simulation mode it must be set to values more on the order of 5 to 10 minutes in order to run a simulation at significantly faster than real time. A simulation POLLINTERVAL setting of 5 minutes will result in a simulation that runs at about 40 times real time.

The constant queue depth configuration of the simulation also causes problems due to bugs in Moab simulation mode. Jobs that are canceled through Moab (as opposed to TORQUE) will have JOBCANCEL records in the event logs; however, these canceled jobs are never actually purged in the simulation. Thus, large numbers of canceled jobs can fill up available slots in the fixed-length queue indefinitely. Also, in some cases jobs that become ineligible to run due to policy reasons (for instance, due to a limit on the number of idle jobs allowed per user) never return to eligibility. The combination of these resulted in simulations that had no eligible jobs past the middle of the fourth month of the simulation. Both of these bugs have been reported to Adaptive Computing and are being investigated at the time of this writing.

This experiment involves two simulations denoted "baseline" and "no negbal" that are differentiated only by whether jobs with a "negbal" quality-of-service utilize backfill. The goal of this experiment is to determine the impact that allowing "negbal" jobs to backfill has on resource utilization. A uniform range of dates is chosen for comparison between these simulations due to previously mentioned bugs which starve the simulation of available jobs at long times. As seen in Figure 9, the daily average resource utilization during the month of August 2010 exhibits a sharp decline in utilization due to available job starvation; thus, only the three month period between May 1, 2010 and July 31, 2010 is considered as viable for the purposes of this experiment.

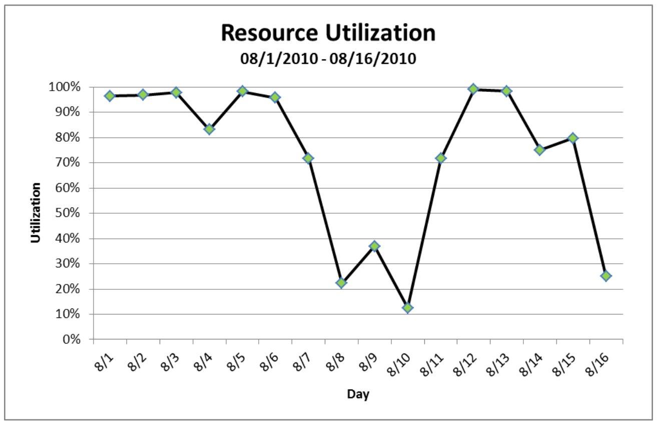


Figure 9: A plot showing the daily average resource utilization for the baseline simulation during the month of August 2010.

3 Results

Figure 10 shows the weekly average resource utilization between May 1 and July 30, 2010. The utilization for the baseline remains above 90% through the entire period. However, the utilization for the no negbal case is between 90% and 74%. Over the period between May 1 and July 31, 2010 the average utilization for the baseline and no negbal simulations are 95% and 84%, respectively. This 15% decrease in overall utilization suggests that allowing these jobs to run in backfill has a substantial positive effect on resource utilization.

To further investigate the cause of this decreased utilization, the resource utilization leading up to scheduled preventative maintenance (PM) periods is analyzed. In both the baseline and no negbal simulations, about 55% of non-negbal jobs require less than 2 hours and about 68% of non-negbal jobs require less than 512 compute cores, suggesting that sufficient jobs exist in both simulations to effectively utilize backfill. Despite the similarity between the simulations in the availability of non-negbal jobs capable of backfilling, the observed utilizations during the drains associated with the PM periods clearly indicate significantly different scheduling behavior for the two simulations.

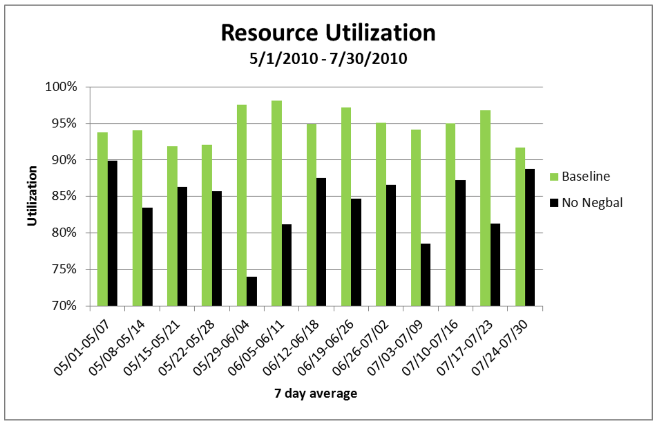


Figure 10: The weekly average resource utilization for the baseline and no negbal simulations.

Figure 11 shows the daily average resource utilization for two weeks between May 23 and June 5, 2010. A marked decrease in utilization is seen on Wednesdays during the period for the baseline simulation. This corresponds to the weekly PM scheduled between 8:00 AM and 8:10 AM each Wednesday. However, the utilization depression is more pronounced for the no negbal simulation extending to the previous day.

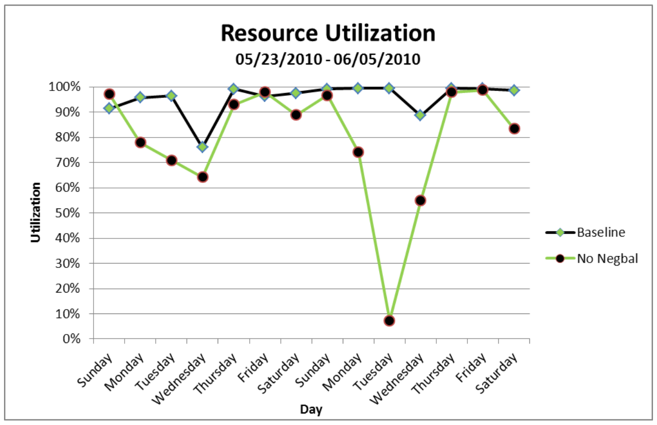


Figure 11: The daily average resource utilization for the baseline and no negbal simulations over two weeks.

Figure 12 shows the hourly average resource utilization for Tuesday May 25 at 6:00 AM to Wednesday May 26, 2010 at 11:00 AM. The baseline simulation starts draining, as defined by a sustained utilization less than 90%, 11 hours before the scheduled PM; but, the no negbal simulation begins draining 24 hours before the scheduled PM. This drastically different draining profile suggests that the execution of backfillable non-negbal jobs might be inhibited by disabling backfill for negbal jobs. Perhaps the most plausible explanation for the unexpected magnitude of this behavior is that ineligible negbal jobs are filling up the simulation queue and artificially inhibiting the injection of eligible jobs.

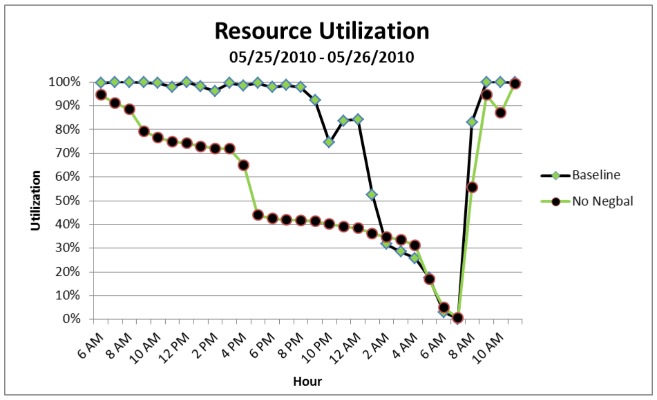


Figure 12: The hourly average resource utilization for the baseline and no negbal simulations over two days.

Figure 13 shows the mean effective queue duration, the cumulative time a job was eligible in the queue, for non-negbal jobs as a function of requested wallclock time for the period May 1 to July 31,2010. The no negbal simulation shows a significant decrease in the mean effective queue duration as compared to the baseline simulation. This result is consistent with the lower resource utilization of the no negbal simulation. However, this result is not consistent with a situation in which backfill efficiency is depressed as seen in previous examples.

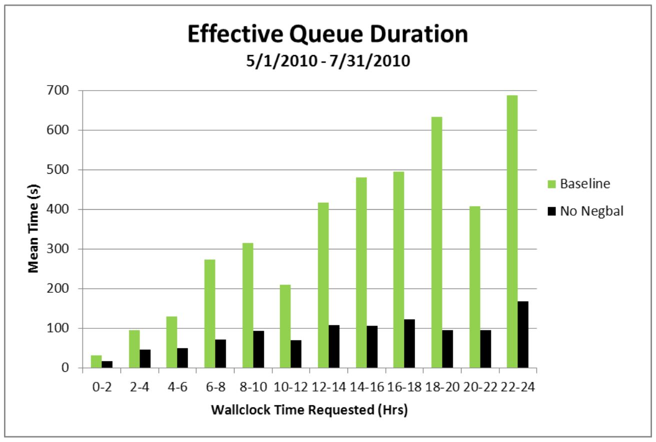


Figure 13: The mean effective queue duration as a function of wallclock time requested for the baseline and no negbal simulations.

4 Conclusion

Although this work shows progress towards modeling the drains associated with scheduled PMs, achieving appropriate performance toward high throughput simulation, and collecting and analyzing appropriate queue data, appropriate simulation of job interactions within the queue remains a challenge. The experiment presented in this work illustrates two problems with the handling of the simulation queue. First, the use of a fixed queue depth artificially limits the maximum number of jobs considered for scheduling due to bugs within the Moab simulator. Specifically, some jobs (such as canceled jobs) are allowed to fill a slot within the queue indefinately, causing such jobs to accumulate over long simulation periods. This accumulation effectively reduces the depth of the queue as simulations progress, resulting in eligible job starvation that adversely affects resource utilization. Secondly, the fixed queue depth adversely affects the results for the no negbal simulation by blocking higher priority jobs that would normally be injected into the job mix. This allows negbal jobs to run ahead of other higher priority jobs when the fixed-depth queue fills entirely with negbal jobs. Since the total number of negbal jobs in each studied simulation is roughly ten times the queue depth, it is likely that the queue can become dominated by these low priority jobs which could not be backfilled in the no negbal simulation. This situation explains the apparent lack of backfilling in system drains, the high number of negbal jobs run within the simulation, and the shorter effective queue duration.

5 Future Work

The path forward for this research focuses on examining alternate approaches for managing the injection of jobs into the simulator. Specifically, the use of job submission times included in the workload trace to control the injection of jobs merits investigation. This approach is expected to better model policy interactions, but eligible job starvation is expected to impact utilization due to the increase in efficiency of the simulated system as compared to the actual resource reflected in the workload trace. Future investigations are also expected to examnine the combined use of minimum queue depths and submission times as a possible means to more realistically model real-world queue behavior.

6 About the Authors

All of the authors are employed by the University of Tennessee at the National Institute for Computational Sciences (NICS) at Oak Ridge National Laboratory (ORNL). They can be contacted by mail at National Institute for Computational Sciences, Oak Ridge National Laboratory, P.O. Box 2008 MS6173, Oak Ridge, TN 37831-6173.

Lonnie D. Crosby is a computational scientist with a Ph.D. in Chemistry from The University of Memphis located in Memphis, TN. He can be contacted by email at lcrosby1@utk.edu.

Troy Baer is a HPC systems administrator with a M.S. in Aerospace Engineering and a B.S. in Aerospace Engineering. He can be reached by emailing tbaer@utk.edu.

R. Glenn Brook is a computational scientist with a Ph.D. in Computational Engineering from the University of Tennessee at Chattanooga. He can contacted by email at glenn-brook@tennessee.edu.

Matt Ezell is a HPC systems administrator with a B.S. in Electrical Engineering. He can be reached via email at ezell@nics.utk.edu.

Tabitha K. Samuel is a HPC systems programmer with a M.S. in Computer Science and a B.E. in Computer Science and Engineering. She can be reached via email at tsamuel@utk.edu.

References

[1] Troy Baer and Don Maxwell. Comparison of scheduling policies and workloads on the nccs and nics xt4 systems at oak ridge national laboratory. In *Proceedings of Cray User Group Meeting 2009*, Atlanta, GA, May 2009.

[2] Troy Baer. Using quality of service for scheduling on cray xt systems. In *Proceedings of Cray User Group Meeting 2010*, Edinburgh, Scotland, May 2010.

[3] P. Andrews, P. Kovatch, V. Hazlewood, and T. Baer. Scheduling a 100,000 core supercomputer for maximum utilization and capability. In *Parallel Processing Workshops (ICPPW), 2010 39th International Conference on*, page 421. IEEE, September 2010.

[4] Moab workload manager [online]. 2011. Available from: `http://www.clusterresources.com/pages/products/moab-cluster-suite/workload-manager.php` [cited 05/10/2011].

[5] Torque resource manager [online]. 2011. Available from: `http://www.clusterresources.com/pages/products/torque-resource-manager.php` [cited 05/10/2011].

[6] Simulations [online]. 2011. Available from: `http://www.adaptivecomputing.com/resources/docs/mwm/16.3.0simulations.php` [cited 05/10/2011].

[7] Workload accounting records [online]. 2011. Available from: `http://www.adaptivecomputing.com/resources/docs/mwm/16.3.3workloadtrace.php` [cited 05/10/2011].

[8] Resource traces [online]. 2011. Available from: `http://www.adaptivecomputing.com/resources/docs/mwm/16.3.2resourcetrace.php` [cited 05/10/2011].