Empirical-based Probabilistic Upper Bounds for Urgent Computing Applications

Nick Trebon [#1] , Pete Beckman [*2]

[#] *Computer Science Department, University of Chicago*
1100 East 58th Street, Chicago, IL 60637 USA
[1] ntrebon@cs.uchicago.edu

[*] *Argonne National Laboratory*
9700 South Cass Avenue, Building 360, Argonne, IL 60439 USA
[2] beckman@mcs.anl.gov

***Abstract*—Scientific simulation and modeling often aid in making critical decisions in such diverse fields as city planning, severe weather prediction and influenza modeling. In some of these situations the computations operate under strict deadlines, after which point the results may have very little value. In these cases of *urgent computing*, it is imperative that these computations begin execution as quickly as possible. The Special PRiority and Urgent Compute Environment (SPRUCE) is a framework designed to enable these high priority computations to quickly access computational Grid resources through elevated batch queue priority. However, participating resources are allowed to decide locally how to respond to urgent requests. For instance, some may offer next-to-run status while others may preempt currently executing jobs to clear off the necessary nodes. However, the user is still faced with the problem of resource selection – namely, which resource (and corresponding urgent computing policy) provides the best probability of meeting a given deadline? This paper introduces a set of methodologies and heuristics aimed at generating an empirical-based probabilistic upper bound on the total turnaround time for an urgent computation. These upper bounds can then be used to guide the user in selecting a resource with greater confidence that their deadline will be met.**

I. Introduction

There exists a growing domain of computational problems that are constrained by a strict deadline. In this domain of *urgent computing*, results produced after the deadline may have very little use. For example, a tornado modeling application must provide enough warning for residents in the targeted area to seek shelter. Similar scenarios exist for applications modeling wildfires, hurricanes and tracking urban airflow contaminants. In order to meet their deadlines, it is imperative that these applications are able to access Grid resources easily and quickly.

The Special PRiority and Urgent Computing Environment (SPRUCE) [1], [2] is a token-based framework that enables users to submit high priority jobs to participating Grid resources. SPRUCE supports three urgency levels (i.e., medium, high and critical) that a user can specify during job submission. Participating resources map each urgency level to a locally-decided policy, such as elevating the job to next-to-run status or preemption. However, the challenging problem of resource selection is left to the user. Given the deadline constraint, the urgent computing user is interested in selecting the resource (and corresponding urgent computing policy) that provides them the greatest chance of meeting their deadline. Secondary concerns, such as intrusiveness to other users (e.g., preempting a job) may also be considered when multiple resources offer a high probability of success.

The remainder of this paper is organized as follows. The urgent computing resource selection problem is formally defined in Section II. The set of methodologies and heuristics used to generate the probabilistic upper bounds on total turnaround time is briefly outlined in Section III. A set of experiments designed to evaluate the methodologies for a case study application is presented in IV. Finally, conclusions are detailed in Section V.

II. Problem Statement

An urgent computing user is most interested in selecting a *configuration* of their urgent application that provides them with the greatest likelihood of meeting their deadline. The configuration consists of the computational resource and runtime parameters (e.g., urgency level, data repositories, requested number of CPUs). The traditional Grid resource selection problem is further complicated by the fact that resources may have multiple urgent computing policies in place and that these policies may differ across resources. This complexity further increases the number of choices the user is faced with when deciding upon a resource. For instance, consider the following example: a user has an urgent application that has a choice of four computational resources and may request either 64 or 128 nodes on each resource. The application requires an input file that is available from three repositories and the output data must be sent to one of two output destinations. Additionally, each resource has four policies in place that depend upon the urgency level (i.e., none, medium, high or critical). Thus, the user is faced with 192 different configurations from which to choose.

Furthermore, many of the current approaches to the Grid resource selection problem (cf. [3], [4]) seek to simply reduce the total turnaround time (or some aspect of the total turnaround time, such as execution delay) of a computation.

However, these approaches do not answer the question of whether the deadline is feasible. A more useful approach is to provide the user with the probability that a configuration will meet the given deadline. The user can then select a configuration that provides a high enough probability of success to warrant the initiation of the urgent computation.

The purpose of this poster is to present and evaluate a set of methodologies and heuristics that, taken together, calculate an empirical-based probabilistic upper bound on the total turnaround time for a given configuration of an urgent application. These upper bounds can then guide the user in selecting a configuration that provides them with the greatest probability of meeting their deadline.

III. Methodology

The total turnaround time of an urgent computation, as depicted in Figure 1, consists of the delay incurred from any file staging (both input and output), resource allocation (i.e., batch queue delay), and execution.

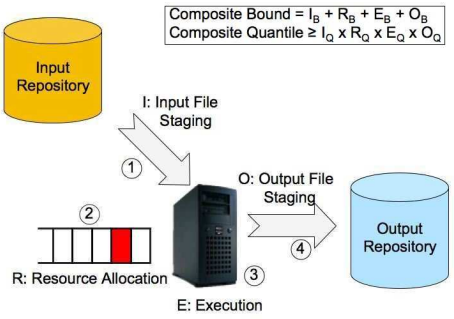

Fig. 1. The four phases of the total turnaround time for an urgent computation. In the simplest case, each phase occurs in serial.

A probabilistic upper bound on the total turnaround time can be generated by predicting an upper quantile. For instance, if the 0.95 quantile for the total turnaround time is 10,000 seconds, then one can say that the there is a 95% chance that the delay will be less than 10,000 seconds. Such a bound may be generated by combining the predicted upper quantiles for each of the individual phases of the total turnaround time. In the simplest case, where each of the individual phases is independent (i.e., no overlap), a bound on the total turnaround time may be computed as follows:

An upper I_Q x R_Q x E_Q x O_Q quantile corresponding to a bound of $I_B + R_B + E_B + O_B$.

It is important to note that in the above formula, the composite quantile is conservative in that the quantile is *at least* the product of the individual quantiles. This value actually represents the probability that all four individual phase bounds are satisfied individually. This formulation ignores the case where at least one of the individual bounds fail but the composite bound still succeeds. This occurs when the overprediction of the individual bounds that succeed compensate for the underprediction in the bound or bounds that fail. However, this conservativeness may be appropriate in the domain of urgent computing where it is better to be overly conservative. The remainder of this section briefly describes how the bounds for each phase are computed.

A. File Staging Bounds (I_B, I_Q and O_B, O_Q)

The methodology for bounding the input and output file staging phases is the same. Short, periodic probes monitor the bandwidth between a source and destination. These measurements are used to predict expected bandwidth in an actual file transfer. The Network Weather Service (NWS) [5] possesses the mechanisms both to monitor the network via short TCP-based probes and also to make forecasts (i.e., predictions). Furthermore, the authors have noted that for some measurement streams, an empirical confidence interval can be generated. However, during experiments, the short TCP-based probes were unable to match the bandwidth behavior achieved by large GridFTP transfers [6]. As a result, a GridFTP probe framework was implemented. NWS is still used to store the measurements and to make predictions. It is important to note that the predicted bounds for the output phase face an inherent challenge. In particular, a prediction is made based on current network conditions for a transfer that will take place at some later point in time. Part of the purpose of this research is to examine how these predictions fare for experiments where the average span of time between when the prediction is made to when the transfer occurs is on the order of two hours.

B. Resource Allocation Bound (R_B, R_Q)

The bounding methodology for the resource allocation phase depends upon the urgent computing policy. There are currently three policy choices that are employed. The first is the normal (or default) policy in which the urgent jobs are scheduled normally by the batch scheduler (i.e., equivalent to no SPRUCE support). In this case, the Queue Bounds Estimation from Time Series (QBETS) [7] tool may be used to generate the bounds.

The first elevated priority policy (and most popular) that SPRUCE-enabled resources employ is next-to-run. In this case, the urgent job is scheduled with an elevated priority that, in effect, places the job at the top of the batch priority queue. In order to predict an upper bound on the delay such a job would incur, the authors of QBETS have developed a special tool. Essentially, a Monte Carlo simulation of next-to-run batch queue delays is conducted on previously observed job history. The simulation produces a distribution of next-to-run delays for a number of different job sizes, and upper bounds can be predicted using the same internal software infrastructure as QBETS.

The second elevated priority policy that has been adopted is preemption. In this case, if necessary, currently running jobs are killed to free up the necessary resources for the urgent job. In order to predict upper bounds for preemption delay, the authors of QBETS have extracted the heart of their tool, which they call the Binomial Method Batch Predictor (BMBP [7]) and modified it to accept an unknown distribution rather

than a time series. The tool is seeded with historic preemption delays for similarly sized jobs. The BMBP tool was chosen because it is general in nature and non-parametric.

C. Execution Bound (E_B, E_Q)

In order to generate bounds on the execution delay, a similar approach to the preemption delay bounds is adopted. The modified BMBP tool is seeded with past observed execution history for similar inputs.

D. Limitations of the Composite Bounds

The composite bounds presented in this paper have a number of limitations. First, the bounds generated are for a single instance of the urgent computation. For example, the bounds are not valid if a user simultaneously submits 100 copies of the same configuration – the bounds do not account for the inherent correlation. Furthermore, as mentioned above, the composite bounds are conservative. The composite quantile can be increased through overlapping phases (e.g., staging input files while the batch job is queued) or through speculative execution of independent configurations [6]. These methods are outside the scope of this poster.

IV. Results

In order to evaluate the bounding methodologies detailed in Section III, a series of experiments were conducted using FLASH [8] as a sample scientific simulation code. FLASH is a modular, multiphysics, adaptive-mesh astrophysics code developed at the University of Chicago Astrophysics Department. It is used to study a variety of astrophysical phenomena and has scaled on some of the largest high-performance computing systems in the world. While FLASH does not typically operate under the deadline constraint that defines an urgent application, it is extraordinarily complex in terms of data and computation requirements. Hence, it is a suitable candidate as a representative scientific simulation code. For these experiments, the case study application executed 1,000 time steps using eight levels of mesh refinement.

The case study application was ported to the computational resources listed in Table I. Because the UC/ANL resource is much smaller, the case study application executed on 16 nodes with 2 processors per node. The UC/ANL resource supports next-to-run and preemption urgent computing policies, in addition to a normal priority. The Mercury and Tungsten resources, which are housed at the National Center for Supercomputing Applications (NCSA), executed the case study application on 32 nodes with 2 processors per node. The NCSA resources only support a normal batch priority.

TABLE I
THE THREE WARM STANDBY RESOURCES FOR THE CASE STUDY APPLICATION.

Name	CPU	Nodes	CPUs
UC/ANL IA-64	Itanium 2 (1.3 / 1.5 GHz)	62	124
Mercury (fastcpu)	Itanium 2 (1.5 GHz)	631	1262
Tungsten	Intel Xeon (3.5GHz)	1280	2560

Though the FLASH application does not have any file staging requirements, these were artificially added to create a complete urgent computing workflow. The requirements were loosely modeled after a Linked Environments for Atmospheric Discover (LEAD) [9] workflow. LEAD is a SPRUCE user, and in 2007 the team worked with SPRUCE to perform real-time, on-demand, dynamically adaptive forecasts [10]. In particular, the computation requires a 3.9 GB data file that is located at the University of Indiana and a similarly sized output file must be transferred to an analysis server at the University of Indiana.

Three of the six case study experiments [6] examined the performance of the individual and composite bounding methodologies for configurations utilizing three priorities: normal, next-to-run and preemption. The remainder of the section describes these experiments and presents the results.

A. Experiment 1: Normal Priority

The purpose of the first experiment was to evaluate the performance of the composite bounds for the normal priority. Additionally, the performance of the execution bounds is of interest in order to validate the selection of the modified BMBP tool as a bounds predictor. For this experiment, the 0.95 quantiles were predicted for each of the four individual phases (i.e., input file staging, resource allocation, execution, and output file staging). The resulting composite bound corresponded to the upper 0.815 quantile. Approximately 100 trials of each configuration were conducted and the actual delays were compared with the predicted bounds. The composite bounds and actual delays are depicted in Figure 2. It is clear that in terms of correctness, the composite bounds are performing well, with very few occurrences where the predicted upper bound is exceeded. However, the bounds are not very accurate, as there is a large degree of overprediction. In the case of the UC/ANL resource, there is a bias in the plot. This particular resource suffers from under-utilization. During the experiments, the large majority of jobs were immediately scheduled for execution (72 of the 99 jobs had a queue delay of one second or less). Because the stream of case study jobs were submitted during a phase of inactivity on the resource, the QBETS service detected a changepoint in the delays for those jobs and significantly reduced the predictions. However, even with the bias, the bounds are correct (i.e., very rarely exceeded).

The success rate of the individual and composite phase bounds are depicted in Table II and the rates of overprediction are presented in Table III. For the individual phases, the target success rate is 95%; for the composite bound, the target is 81.5%. For the input phase, only the Tungsten configuration is less than the target. However, there were a number of problems with the Tungsten GridFTP server. The dedicated GridFTP server often had long periods of time where it would not accept incoming connections. This problem caused probes to fail, which led to predicted bounds based off of stale probe data for the experimental trials that succeeded. After 15 trials, the experiments and probes began using the login

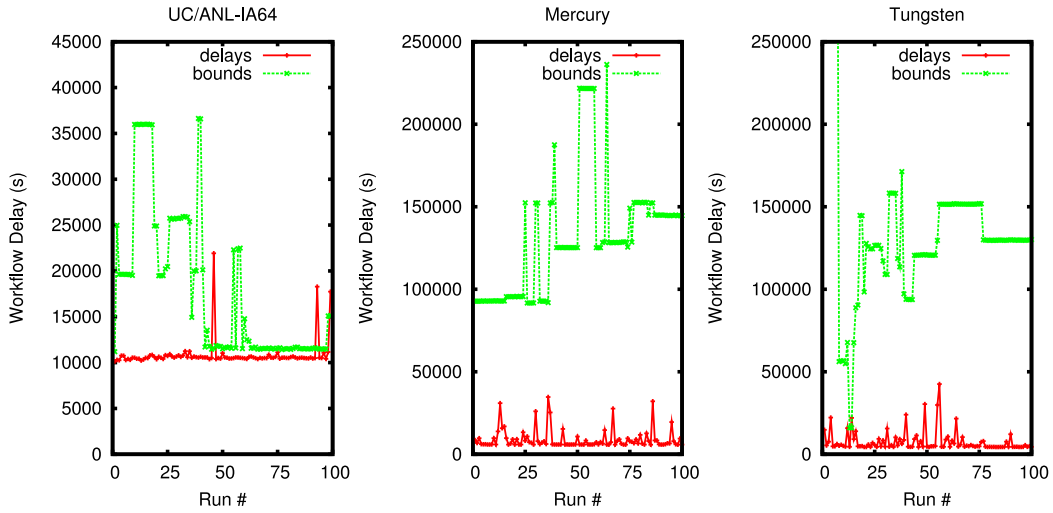

Fig. 2. Predicted upper 0.815 quantile versus the actual delay achieved by the configurations utilizing the normal priority (i.e., no SPRUCE). Note the difference in scales.

node as the GridFTP endpoint. This removed the problem of rejected connections but the performance of this server was more variable, which was evident in both the probes and actual transfers.

All three configurations exhibit correct bounds for the queue phase. However, the bounds are clearly not accurate. This overprediction is most likely caused by the inherent skew in batch queue data, where the large majority of jobs experience a relatively short delay and very few jobs experience long delays. However, by targeting the upper 0.95 quantile for these experiments, the bounds were forced to accommodate some of the high delay jobs, resulting in large bounds. In later experiments, targeting the 0.75 or 0.50 quantile for the queue phase resulted in much smaller bounds.

For the execution phase, the adapted BMPB methodology appears to do an excellent job in terms of correctness and accuracy. In particular, the UC/ANL resource consists of CPUs of two different clock speeds. As a result, the execution performance is bimodal; however, the methodology still resulted in 100% correctness with reasonable accuracy.

The performance of the output phase bounds is promising. As mentioned before, this phase suffers from the inherent challenge of making predictions for future transfers. In addition to this challenge, most of the configurations also had further problems that contributed to the lower success rate. For the UC configuration, all 17 misses for this phase occurred in a span of 18 trials where no probes were completed. As a result, the predicted bounds were not reflecting the current network state. Of the 70 trials after this period, no misses occurred. For Mercury, two of the output transfers actually failed. Of the remaining 12 misses, most were the result of making a prediction for a future transfer. This was evident in the probe data where the probes detected a decrease in network bandwidth after a prediction was made but before the transfer occurred. Finally, in the case of Tungsten, the problems with the GridFTP servers were previously discussed.

The composite bounds perform very well in terms of correctness, where the target success rate of 81.5% is exceeded by all three configurations. However, there is a large degree of overprediction in all three configurations. This is entirely driven by the overprediction in the queue phase. In fact, the average composite bounds predicted for the NCSA configurations were more than an order of magnitude larger than the average delay.

The results of this experiment demonstrate that the composite bounds are correct, but not accurate. Also, the choice of the modified BMBP tool is validated by the performance of the bounds for the case study application. Also, the large degree of overprediction in the queue phase is an implicit argument for SPRUCE, which aims at providing shorter batch queue delays with less variability – which should lead to more accurate (i.e. tighter) predicted bounds.

TABLE II
PERCENTAGE OF MEASUREMENTS THAT WERE LESS THAN THE PREDICTED UPPER BOUND FOR THE NORMAL PRIORITY CONFIGURATIONS. THE 0.95 QUANTILE WAS TARGETED FOR THE INDIVIDUAL PHASES, RESULTING IN A 0.815 COMPOSITE QUANTILE.

	#	Input	Queue	Exec.	Output	Comp.
UC/ANL	99	100%	97%	100%	83%	97%
Mercury	100	99%	99%	93%	86%	100%
Tungsten	100	88%	99%	96%	77%	99%

TABLE III
PERCENT OVERPREDICTION FOR EACH PHASE OF NORMAL PRIORITY CONFIGURATIONS.

	Input	Queue	Exec.	Output	Comp.
UC/ANL	7.98%	2,379%	7.01%	21.9%	64.95%
Mercury	10.19%	4,468%	3.78%	5.29%	1,417%
Tungsten	10.57%	4,811%	13.03%	-4.63%	1,824%

B. Experiment 2: Next-to-run Priority

The purpose of the next-to-run experiment was to examine the performance of the next-to-run bounds predictor as well as the composite bounding methodology for a SPRUCE urgent computation. This experiment consisted solely of the UC/ANL resource. Because the bounding methodology for the execution phase performed well in the previous experiment, execution delay was emulated in this experiment by randomly sampling the execution history. This was done in order to eliminate the needless wasting of CPU cycles by the experiment trials. As in the previous experiment, the 0.95 quantiles were predicted for each individual phase, resulting in a composite 0.815 upper quantile.

The predicted composite bounds versus the actual delays are plotted in Figure 3. It is clear that the bounds are much tighter than the bounds for the normal configurations. There are a few cases where the actual delay exceeded the predicted bounds. All of these cases occurred during a rare burst of activity on the resource where 25 8-hour, 8-node jobs were submitted at once. The largest spike occurred when a large 24-hour job was submitted that resulted in a queue delay of over 13 hours. It is important to note that while this introduced significant delays to the trial jobs, the delay would have been longer if the jobs had been submitted under a normal priority as the trial job would have been forced to wait until most of the competing jobs had completed execution.

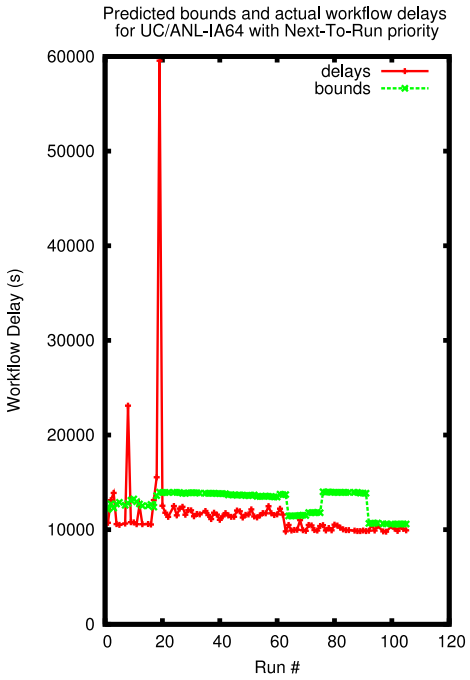

Fig. 3. Predicted upper 0.815 quantile versus actual delay achieved by next-to-run configuration on UC/ANL resource.

The success rate of the bounds is presented in Table IV and the overprediction rate is presented in Table V. In terms of correctness, all of the bounds exceeded the target rate or fell just short. The composite bound again exceeded the target success rate by more than 10%. In terms of accuracy, there was a vast improvement in the queue and composite bounds compared with the normal configurations. This occurred because the skew in the batch queue date that caused the overprediction in the normal configuration is almost entirely reduced or eliminated.

TABLE IV
PERCENTAGE OF MEASUREMENTS LESS THAN THE PREDICTED UPPER BOUND FOR THE NEXT-TO-RUN PRIORITY CONFIGURATION ON UC/ANL RESOURCE. THE 0.95 QUANTILE WAS TARGETED FOR THE INDIVIDUAL PHASES, RESULTING IN A 0.815 COMPOSITE QUANTILE.

	#	Input	Queue	Exec.	Output	Comp.
UC/ANL	98	100%	92%	91%	92%	94%

TABLE V
PERCENT OVERPREDICTION FOR EACH PHASE OF NEXT-TO-RUN CONFIGURATION ON UC/ANL RESOURCE.

	Input	Queue	Exec.	Output	Comp.
UC/ANL	8.04%	46.37%	4.54%	41.47%	11.27%

The results from the next-to-run experiment further validate the methodology for bounding next-to-run jobs. The bounds that are produced are both correct and accurate, which lead to composite bounds that are much more accurate than in the normal priority condition. Furthermore, these results confirm the intuition that submitting an elevated priority job will result in both lower and less variable bounds.

C. Experiment 3: Preemption Priority

The purpose of the preemption experiment was to examine the performance of the preemption bounds predictor. This experiment consisted solely of the UC/ANL resource. Similar to the next-to-run experiment, the execution delay was emulated. As in the previous experiment, the 0.95 quantiles were predicted for each individual phase, resulting in a composite 0.815 upper quantile.

It is also important to note how preemption was utilized. In order to seed the predictor, a history of preemption delays for similarly-sized jobs was used. For the actual experiment, a dummy job was first submitted to the resource. Once the job began execution, the preemption trial was initiated. This was done in order to avoid preempting actual jobs on the resource. As a result, all of the trials experience a non-trivial queue delay because they are never immediately allocated onto the resource. This is in contrast to the previous experiments where the under-utilization of the resource often led to little or no queue delay.

The predicted composite bounds versus the actual delay are plotted in Figure 4. Similar to the next-to-run case, the bounds are much tighter than the normal configuration. However, the large spikes seen in the next-to-run plot did not occur because of the preemption policy. It is important to note that it is still possible to experience large delays with preemption. This would occur when the resource does not have enough available nodes to preempt, such as when there are nodes that are in an

error state or already running a preemptive job. In this case, the job in question would remain queued until enough preemptible nodes were available. This scenario was not addressed in these experiments.

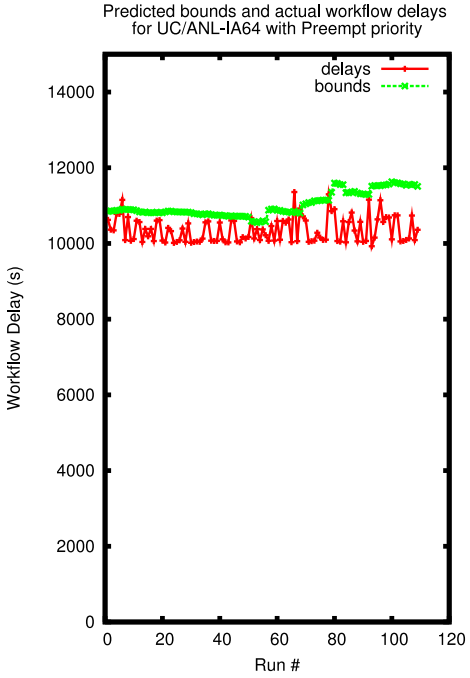

Fig. 4. Predicted upper 0.815 quantile versus actual delay achieved by preemption configuration on UC/ANL resource.

The success rate of the bounds is presented in Table VI and the overprediction rate is presented in Table VII. Of primary interest is the queue phase, which performs well in terms of correctness. The high overprediction rate was caused by the variability in the preemption delays seen during the seeding trials and actual experiment. Typically, the preemption delay was 170 to 300 seconds. However, there were a few cases where the delay was significantly longer – including one trial that lasted over 1,300 seconds. During the experiment, there were several problems with the resource (e.g., node configuration errors) that cropped up and were resolved. However, despite this variability, the bounds still do an adequate job.

The results of this experiment indicate the preemption bounding methodology, which utilizes the modified BMBP tool, does a reasonable job in terms of correctness and accuracy. Similar to the next-to-run case, the composite bounds are much tighter than the normal configurations. Also, despite the fact the there is a non-trivial queue delay for every trial, the average composite delay was lower than in any of the other configurations. This supports the intuition that preemption will provide, on average, faster access to a resource than either next-to-run or a normal priority.

V. Conclusions

This research represents a preliminary approach at generating empirical-based probabilistic upper bounds on the total turnaround times for urgent applications. The primary source of over-prediction for configurations utilizing a normal (i.e., non-SPRUCE) priority is caused by the skew in the batch queue history. As indicated by the case study, the queue bounds predictors for the elevated priorities result in bounds that are both correct and accurate. As a result, this methodology could aid users in selecting a resource with greater confidence that their deadline may be met.

TABLE VI
Percentage of measurements less than the predicted upper bound for the preemption priority configuration on UC/ANL resource. The 0.95 quantile was targeted for the individual phases, resulting in a 0.815 composite quantile.

	#	Input	Queue	Exec.	Output	Comp.
UC/ANL	109	98%	94%	93%	85%	95%

TABLE VII
Percent overprediction for each phase of preemption configuration on UC/ANL resource.

	Input	Queue	Exec.	Output	Comp.
UC/ANL	5.72%	75.08%	4.39%	7.39%	6.45%

Acknowledgment

This work was supported by the National Science Foundation Office of Cyberinfrastructure, grant number 0503697 ETF Grid Infrastructure Group: Providing System Management and Integration for the TeraGrid.

References

[1] P. Beckman, S. Nadella, N. Trebon, and I. Beschastnikh, "SPRUCE: A System for Supporting Urgent High-Performance Computing," in *IFIP WoCo9 Conference Proceedings*. Arizona: Springer, Jul. 2006.

[2] (2008) Urgent computing: Spruce. [Online]. Available: http://spruce.uchicago.edu

[3] C. Liu, L. Yang, I. Foster, and D. Angulo, "Design and evaluation of a resource selection framework for grid applications," in *High Performance Distributed Computing, HPDC-11 Proceedings*, 2002, pp. 63–72.

[4] D. Nurmi, A. Mandal, J. Brevik, C. Koelbel, R. Wolski, and K. Kennedy, "Evaluation of a workflow scheduler using integrated performance modeling and batch queue wait time prediction," in *SuperComputing Conference*, 2006.

[5] R. Wolski, "Experiences with predicting resource performance on-line in computational grid settings," *SIGMETRICS Perform. Eval. Rev.*, vol. 30, pp. 41–49, 2003.

[6] N. Trebon, "Deadline-based grid resource selection for urgent computing," Master's thesis, University of Chicago, Chicago, IL, Jun. 2008.

[7] J. Brevik, D. Nurmi, and R. Wolksi, "Predicting bounds on queueing delay for batch-scheduled parallel machines," in *Proceedings of the 11th ACM SIGPLAN symposium on Principles and Practice of Parallel Programming*, New York, 2006, pp. 110–118.

[8] B. Fryxell, M. Zingale, F. Timmes, D. Lamb, K. Olson, A. Calder, L. Dursi, P. Ricker, R. Rosner, J. Turan, and et al., "Numerical simulations of thermonuclear flashes on neutron stars," *Nuclear Physics A.*, vol. 688, no. 1, pp. 172–176, 2001.

[9] Lead portal. [Online]. Available: https://portal.leadproject.org/gridsphere/gridsphere

[10] S. Marru, D. Gannon, S. Nadella, P. Beckman, D. Weber, K. Brewster, and K. Droegemeir, "LEAD cyberinfrastructure to track real-time storms using SPRUCE urgent computing," *CTWatch Quarterly*, vol. 4, Mar. 2008.