Analysis of NERSC Job Queue Prediction Tools for Efficient Use of Cray Systems

Rachel Harred
Department of Computer Science
University of North Carolina at Greensboro
Greensboro, NC 27402
Email: r_harred@uncg.edu

Jack Deslippe, Ph.D.
HPC Consultant
NERSC, Division of LBNL
Berkeley, CA 94720
Email: JRDeslippe@lbl.gov

***Abstract*—Supercomputers are used to run extremely large calculations for scientific purposes that would take a normal computer months or even years to complete. The NERSC facility, part of the larger Lawrence Berkeley National Laboratory, is one of the largest centers for supercomputing. One challenge with supercomputing is handling a large volume of jobs of different sizes and run times. The queue system currently in place has a program that attempts to predict how long each job will wait in the queue before starting. This program, called "showstart", is not very accurate and can frustrate users. This research project attempted to discern how showstart made its predictions and to see if a better way was possible.**

Keywords: supercomputer; Petaflop.

I. Introduction

A. Background

Supercomputers are used to run extremely large calculations for scientific purposes that would take a normal computer months or even years to complete. The NERSC facility, part of the larger Lawrence Berkeley National Laboratory, is one of the largest centers for supercomputing in the world. The facility has multiple supercomputers, including two Cray systems and an IBM iDataPlex system. [1] The primary system used for this research project was the Hopper system. The Hopper high-performance computer (HPC) is a Cray XE6 and has a peak performance of 1.28 Petaflops per second. A Petaflop is a measurement of floating point operations equaling one quadrillion (10^{15}) operations. Hopper also has 6,384 nodes, each consisting of 24 cores for a total of 153,216 compute cores and 217 Terabytes of memory space. This system is primarily used for running parallel scientific programs. The computing systems at the NERSC facility are used for scientific research projects funded by the US Department of Energys Office of Science and other projects that are consistent with the DOE Office of Science mission. These projects reach across a wide variety of scientific disciplines including chemistry, physics, astronomy and material sciences. The systems are utilized by many DOE laboratories and by universities across the world. With so many computing jobs being submitted every day, a complex queue system is in place to handle the workflow. There are ten different queues on the Hopper system alone, each with different minimum and maximum node and wall time requirements. The wall time of a computing job is the approximate time the job will need to run on the system and is set by the user when the job is submitted. The queues use the node and wall time requirements to determine each jobs position in the queue and to schedule the order of job execution. Job submission and queue access is achieved via the SSH protocol. Users can then use the system to compile and run programs. A list of the current queued jobs can be displayed by typing the qs command, which is based on the underlying "qstat" and "showq" commands provided by the scheduler. This will show every job in the queue, the status of each job, whether it is on hold, enqueued, running or completed, the current queue position and its node and wall time requirements. Another command, showstart, can be used to help users predict how long their current wait time will be. Showstart displays three separate times: reserve, historical and priority, which are each based on different parameters and always show different times.

B. Scope & Rationale

The first objective of this research project was to investigate the current job queue prediction system, showstart, to discover its flaws and to see if there was a way to improve it. The second objective was to create a tool which would allow users to see the optimal node size and wall time for their jobs based on total CPU time that would have the shortest wait time in the queue. These objectives are important because NERSC users require a better way to estimate how long their jobs will stay in the queue, both before and after each job is submitted in order to more effectively plan and organize their research. Additionally, NERSC wants to promote jobs that are lighter on the scheduler and that use the full capabilities of the system. Also, the user can decide how they should run a job: whether it is better to ask for more time on less nodes or less time on more nodes, for example.

II. Experimental Methods

A. Objective One

To complete the first objective, a BASH script was written that performed a number of different tasks. First, the script created and submitted batch job scripts for jobs of varying node sizes and wall times. Then the script called the showstart command on each job and recorded the three different

predicted start times. It called the qs command to see what the current queue position was. The script also saved the current time of these calls. This cycle was repeated every 15 minutes until all of the jobs were completed. Then all of the data was inserted by the script into a table in a MYSQL database. This script was used multiple times to submit and monitor jobs using node sizes of 1, 42, 314, 939, 1251 and 2084 nodes and wall times of 1, 7, 13, 19, and 25 hours. These sizes showed the performance of the regular queue which is split up by node size into reg_small, reg_medium and reg_big. Jobs to show the performance of the reg_long queue were also tested, using node sizes of 30 and 63 and wall times of 50 and 90 hours. Graphs were then made with this accumulation of data using an open source collection of JavaScript code for web graphs called Highcharts. For the showstart data, a PHP script was written to query the database with a particular job ID and fetch the showstart reserve time, the actual start time and the jobs rank in the queue. This information was output in JSON format. This was then read into the Highcharts JavaScript program for line graphs using asynchronous JavaScript calls (AJAX). This program took the data, formatted it and created a graph. Then a second PHP script was written to take the JavaScript code and display the graph on the MyNERSC web portal. Here is a sample result from these programs, showing an example job of 1 node and 13 hours wall time.

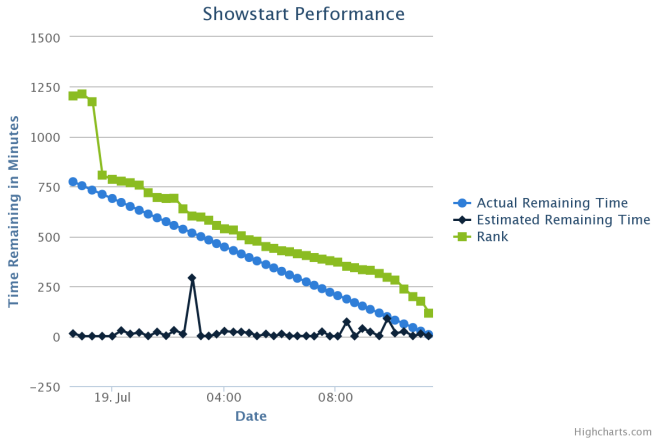

Fig. 1. An example of the Highcharts graphs, showing a job that uses one node and a thirteen hour wall time.

B. Objective Two

The second objective was achieved by creating graphs from a database of completed jobs that showed the average difference in time between queued time and start time for jobs of similar node sizes and wall times. Before this project began, there was a problem with the existing calculation of queued time. Jobs were marked as being in the queue even if they were on hold. Jobs can be placed on hold by the system or by the user. The system typically places a job on hold if the user submits a number of jobs above their allowed limit. However, this skewed some of the results because it would seem as if some jobs were in the queue for a longer time than they actually were. To solve this problem, another database was created using information from two existing databases, a database of all completed jobs, and a database created by executing the qs command every 10 minutes and saving the results. A Python script was written to take job IDs from the completed jobs database and use the IDs to query the qs database. If a job in the qs database was placed on hold at any point, it was flagged. When the job was finally released from hold and enqueued, the time was noted and this time was used for the queued time. All of this information was put together into a single database. A web tool was created as before using Highcharts, JavaScript and PHP scripts. The tool allows the user to select which database to display information from and which parameter to hold fixed: node size, wall time or total CPU hours. Based on which parameter is fixed, the user can then choose from a range of options for the desired parameter they wish to explore. Graphs from the existing completed jobs database and the new database can be compared side by side to show the error when using the former queued times.

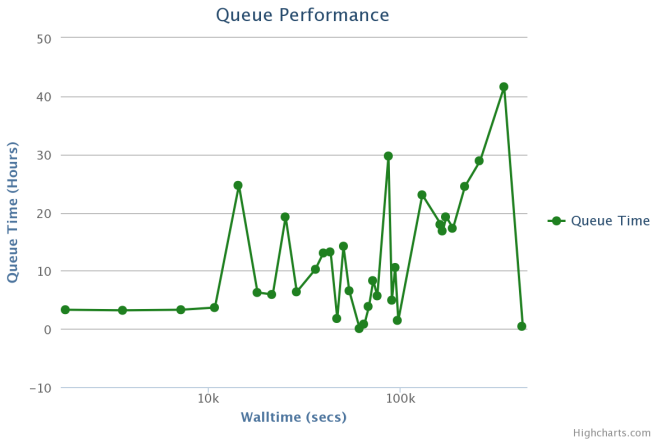

Fig. 2. An example of the old database showing jobs of varying walltimes and 1 node across a 7 day history.

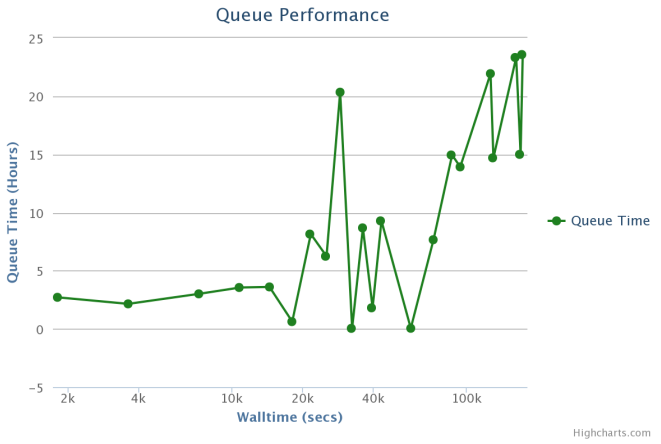

Fig. 3. An example of the new database showing jobs of varying walltimes and 1 node across a 7 day history.

III. Main Findings & Conclusions

Overall, it was found that of the three time predictions given by the showstart command: reserve, historical and priority, only the reserve time is close to accurate. The historical and priority times given were never close to the actual start time and were discarded from consideration.

For the first objective of discovering the limitations of showstart, the graphs of different jobs of varying node sizes and wall times were examined. For jobs using one node, the showstart reserve time was a poor prediction of the actual start time. (See figures 4 & 5.)

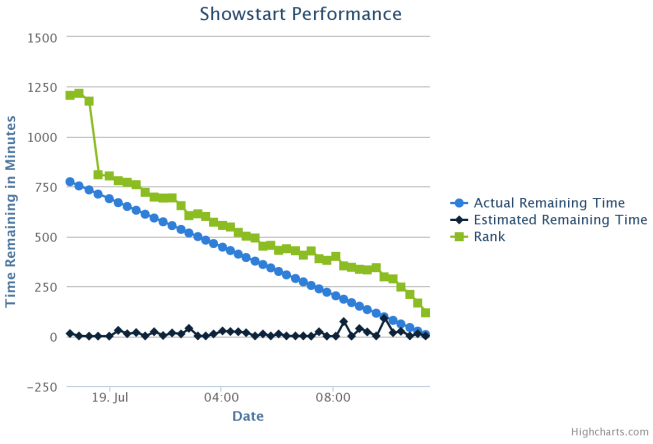

Fig. 4. Showstart prediction vs. actual start time for a job using 1 node and 7 hours wall time.

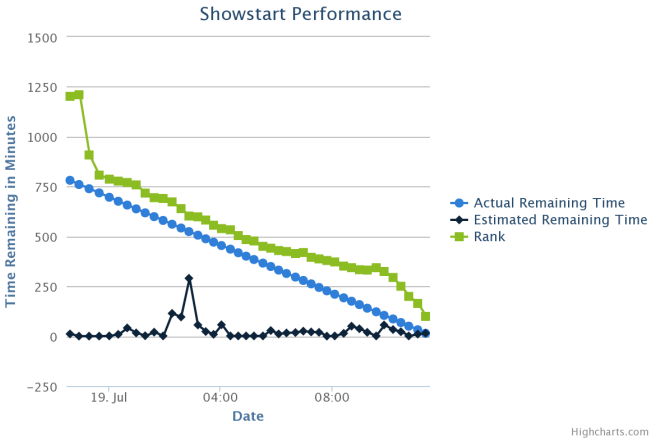

Fig. 5. Showstart prediction vs. actual start time for a job using 1 node and 25 hours wall time.

The queue ranking of these jobs gives a better indication of when the job will begin, as the curves of the actual start time and the queue rank seem to track each other and fall at similar rates. The showstart prediction hovers near the bottom, staying between 0 and 30 minutes for most of the jobs stay in the queue.

For tested jobs using 42 nodes, the showstart time was a great prediction of when the jobs would begin, for all tested wall times. (See figure 6.)

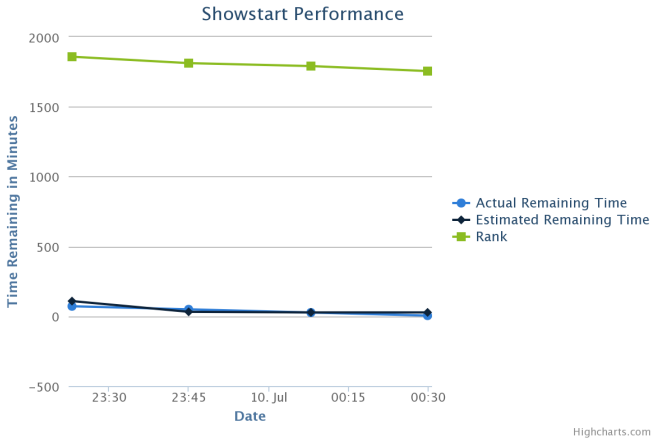

Fig. 6. Showstart prediction vs. actual start time for a job using 42 nodes and 7 hours wall time.

Jobs tested with 314 nodes showed neither the showstart prediction nor the queue position was a good indication of a jobs start time. The only positive thing that can be said about either of these statistics is that they show an overall downward trend. (See figure 7.)

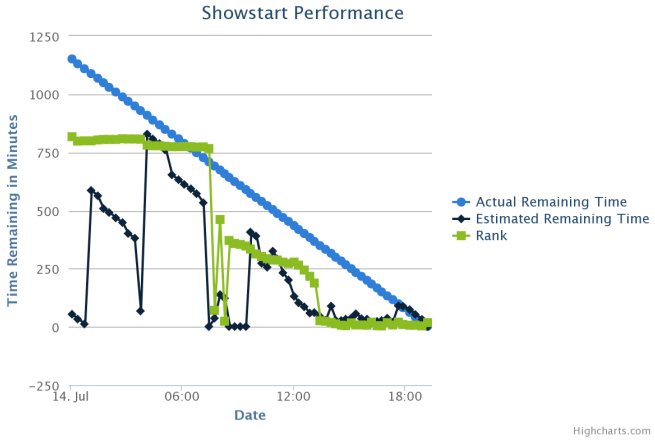

Fig. 7. Showstart prediction vs. actual start time for a job using 314 nodes and 7 hours wall time.

Jobs using 939 nodes were badly predicted by showstart and queue position for low wall times of 1 and 7 hours and more accurately predicted by showstart for higher wall times of 19 and 25 hours. (See figure 8.)

For jobs that used 1251 nodes, the showstart prediction was closer for low wall times than high wall times, although all wall times predictions showed a downward trend. (See figure 9.)

Jobs that used 2084 nodes were only well predicted for 25 wall hours, with the queue position being a poor indication of start time for all wall times. (See figure 10.)

Conclusions that can be drawn from this data are that the showstart prediction is poor for jobs with a low number of

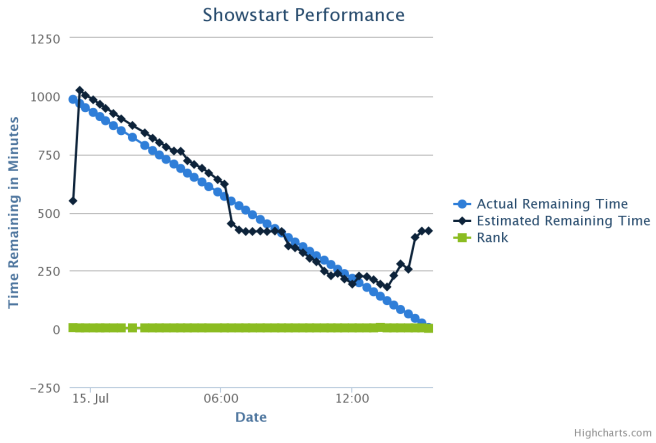

Fig. 8. Showstart prediction vs. actual start time for a job using 939 nodes and 7 hours wall time.

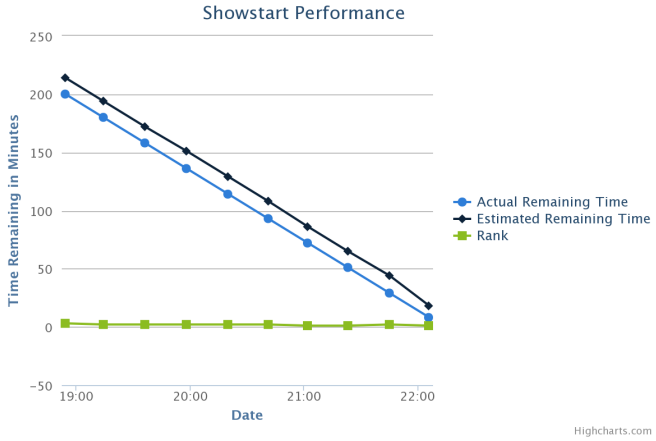

Fig. 9. Showstart prediction vs. actual start time for a job using 1251 nodes and 7 hours wall time.

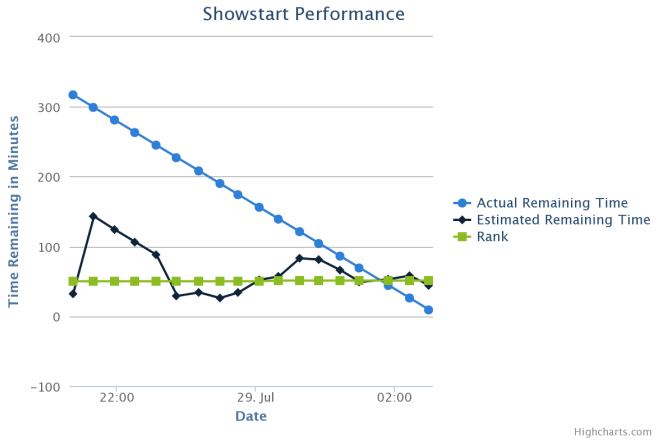

Fig. 10. Showstart prediction vs. actual start time for a job using 2084 nodes and 7 hours wall time.

nodes and for jobs with a high number of nodes and a low wall time. For jobs with a low number of nodes, the queue position is a good indication of start time, but for jobs using a high number of nodes, the queue position tends to fluctuate between 1 and 10 for the entirety of the time in the queue. This is most likely because jobs with high node use are given priority, but take longer to start due to the system having to drain to accommodate them. During this time, smaller node jobs are able to run while the system is waiting for other jobs to finish. This explains why sometimes small jobs begin while they are still listed around 100 in the queue.

For the second research objective, it was found that there is a substantial difference between the former completed jobs database and the new combined database that includes a more accurate queue time. The web tool that was created can be utilized by users who have flexibility in their job sizes to see which parameters will give them the fastest job start time. Currently the new database has only a few days of job history to search but as it accumulates more data, it will be more reliable than the former database.

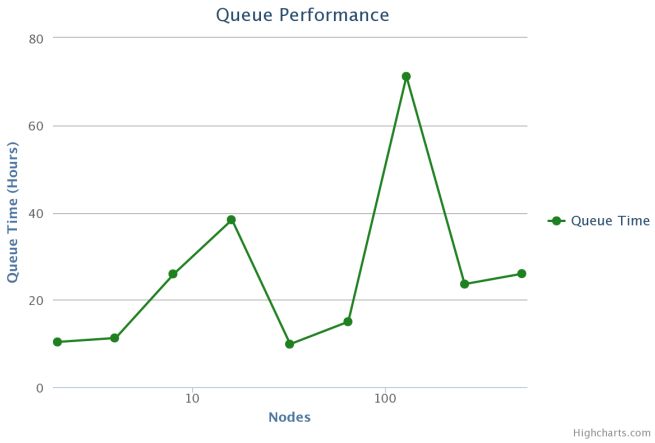

Fig. 11. Graph using the former database, fixing wall time at 6 hours.

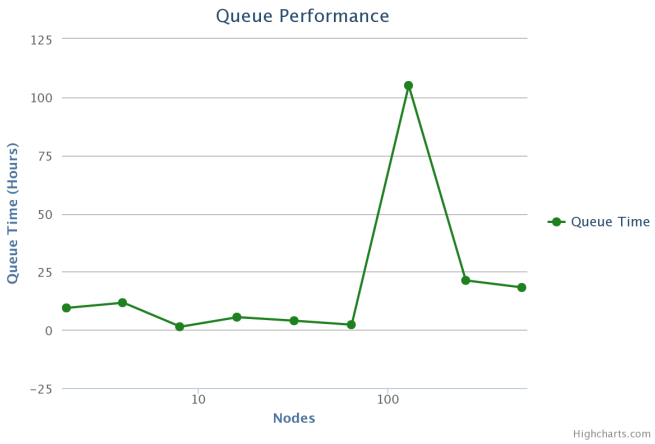

Fig. 12. Graph using the new database, fixing wall time at 6 hours.

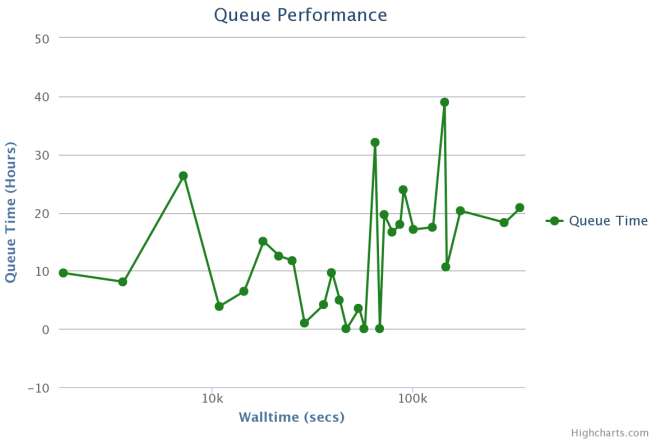

Fig. 13. Graph using the former database, fixing node size at 32.

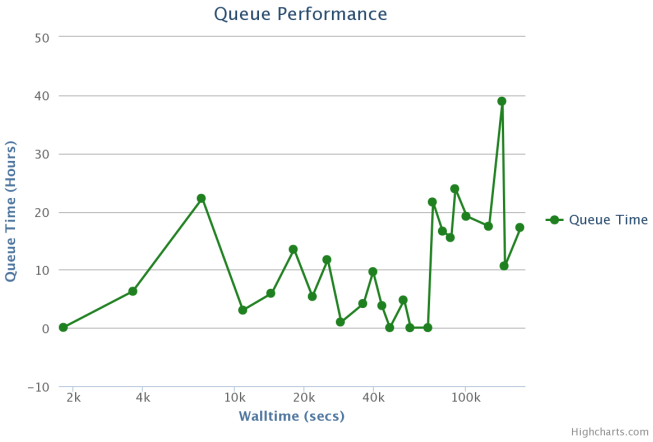

Fig. 14. Graph using the new database, fixing node size at 32.

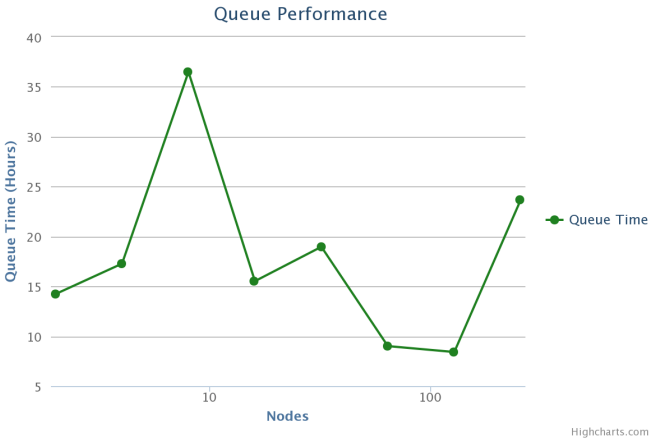

Fig. 15. Graph using the former database, fixing CPU hours at 1000.

SKILLS LEARNED

There were many things that I needed to learn for this project. The use of a Mac computer and the Terminal application, including Unix/Linux command line functions, were necessary for using the NERSC systems. I learned the BASH scripting language to write the first script and MYSQL commands to create and maintain the databases. In particular, I learned how to use BASH scripts to execute MYSQL commands and how those two languages work together. It was also necessary to learn Python to create the second script and use that to execute MYSQL commands as well. I learned about JavaScript, HTML and PHP when creating the web charts and for the Hopper queue system I learned Torque/MOAB commands.

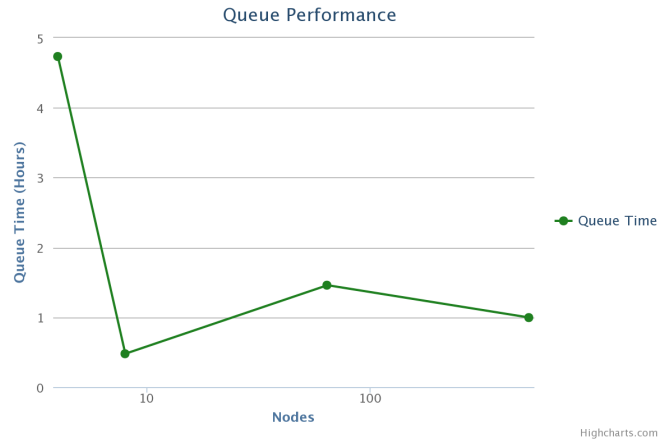

Fig. 16. Graph using the new database, fixing CPU hours at 1000.

FUTURE WORK

The research done on the showstart tool can be used to determine a better way to predict a jobs estimated start time. Using the existing scripts and the web tool, a best fit function could be created using different node sizes and wall times as parameters. As more jobs are accumulated in the new combined jobs database, the web tool will become more accurate for users to determine which parameters will be best to use to ensure the fastest job start times.

ACKNOWLEDGMENTS

The author would like to thank first and foremost Dr. Jack Deslippe for his help and patience during this project. Without him none of this research would have been possible. I would also like to thank the TRUST Director Aime Tabor for her support and guidance throughout the summer. This work was also made possible by TRUST (Team for Research in Ubiquitous Secure Technology), which receives support from the National Science Foundation (NSF award number CCF-0424422), and support from the NERSC (National Energy Research Scientific Computing) Center, Lawrence Berkeley National Laboratory and the U.S. Department of Energy.

REFERENCES

[1] "NERSC: National Energy Research Scientific Computing Center." https://www.nersc.gov. Accessed July 31, 2014.