Increasing Petascale Resource Utilization Through Bimodal Scheduling Policies

Phil Andrews, Patricia Kovatch, Victor Hazlewood, Troy Baer, Justin Whitt,
R. Glenn Brook, Matt Ezell, Tabitha Samuel
National Institute for Computational Sciences, University of Tennessee at Knoxville

ABSTRACT: In late 2009, the National Institute for Computational Sciences placed in production the world's fastest academic supercomputer (third overall, November 2009 Top500), a Cray XT5 named Kraken. Currently Kraken provides 1.17 Petaflops through 112,896 compute nodes accounting for over 60% of the total cycles available to the National Science Foundation users via the TeraGrid. Kraken has two missions that have proven difficult to simultaneously reconcile: providing the maximum number of total cycles to the community, while enabling full machine runs for "hero" users. Historically, this has been attempted by allowing schedulers to choose the time for the beginning of large jobs, with a concomitant reduction in utilization. This paper outlines a novel approach implemented at NICS, whereby the "draining" of the system is forced on a weekly basis, followed by consecutive full machine runs. As previous simulation results suggested, this led to utilization of over 90% (the equivalent of a 300+ Teraflop supercomputer, or several million dollars of compute time per year) with a 92% average over the past 12 months.

KEYWORDS: High-performance computing, scheduling, systems software

1 Introduction

The TeraGrid [1] is the association of High Performance Computing (HPC) sites funded by the National Science Foundation (NSF). The TeraGrid [2] provides over one billion (1,000,000,000) core-hours for the scientific community per year. Although there are approximately a dozen resource providers within the TeraGrid, over 60% of the cycles are provided by a single machine: Kraken, the Cray XT5 system at the National Institute for Computational Sciences, Tennessee (Figure 1). Kraken is an MPP (massively parallel processing) computer with a proprietary interconnect, specifically designed to run large, multi-threaded jobs across the whole machine (capability computing). The alternative approach, capacity computing, involves running large numbers of non-interacting jobs, often submitted by numerous independent researchers. While it is almost always possible to run a capability computer in a capacity mode, that is normally an inefficient use of a more expensive system [3].

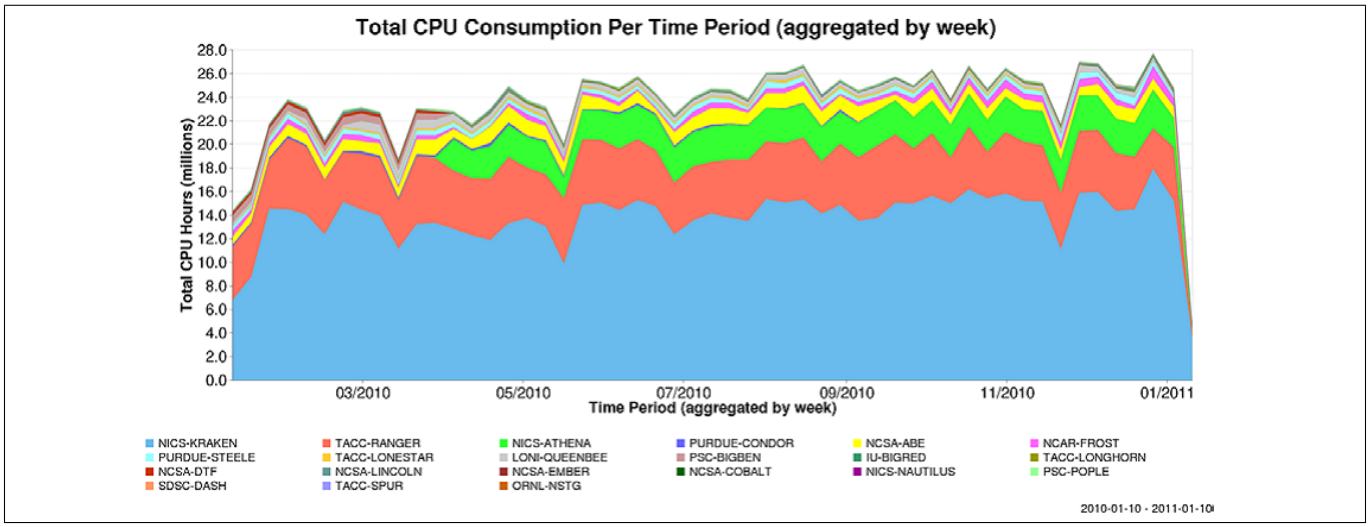

Figure 1: Teragrid CPU Hours Delivered by Machine

Kraken breached the Petaflop barrier and became the fastest academic supercomputer in existence when it was upgraded in October 2009 to 8,256 dual-socket nodes with a 2.6 GHz, 6-core AMD "Istanbul" processor in each socket. In February 2011, an additional 1,152 dual-socket nodes with 2.6 GHz, 6-core, AMD "Istanbul" processors were added–yielding 112,896 cores and a peak performance of 1.17 Petaflops. The interconnect is the Cray Seastar 2 [4]. Disk space consists of 3.3 Petabytes (raw) of DDN drives, organized as a single Lustre parallel file system.

Kraken's massive size enables truly enormous jobs that cannot run anywhere else on the TeraGrid. Such applications are intended to bring new scientific results and are thus encouraged to run. At the same time some algorithms cannot efficiently scale to this number of tightly coupled threads, but can be of equal scientific importance and may require as many, or more, total cycles for scientific advances. These antagonistic user needs are difficult to reconcile. Schedulers, in general, attempt to use as many available nodes on a system as possible which favors capacity computing since a large number of independent jobs allows the scheduler to efficiently utilize the system. But if large jobs are queued, they will never run on even a moderately busy system. A common solution is to include a weighting factor for queued jobs that strongly increases as a function of wait time. Thus, the priority of the large jobs eventually increases to a point where the system is cleared out sufficiently allowing them to run, but nodes go idle as the machine is gradually cleared out for the larger jobs which can significantly reduce the utilization of the system.

In a previous paper [5], using a simulator to explore the effect of different scheduling policies on a large number of actual jobs submitted to a single computational system, we found a counter-intuitive result. Using the dataset of jobs submitted to the San Diego Supercomputer Center's DataStar [6] system, showed that imposing a system wide reservation, so that the entire system was forced to "clear out" by ending all running jobs, could actually improve overall utilization. In February 2010 a hybrid of this simulated approach was implemented at NICS: a machine-wide weekly reservation was implemented forcing the clearing out of the machine. Wednesday was chosen to allow for required preventive maintenance before the capability jobs ran and to minimize the number of time the machine was drained, although this typically occurred no more than about once every third week. After the machine was emptied, any large jobs (more than 85% of the machine) were then run sequentially. When the series of large jobs was completed, jobs of that size would not be eligible to run again until the following week. In return for this restriction, a discount in charging against their allocation was instituted. Capability users were charged for the whole machine, but at a discounted rate of 50% or more. This was also an incentive for the users to scale their codes up as large as possible.

2 Scheduling Methodology

Often, a few nodes of a MPP machine may go down while the overall system remains usable. It is possible that over a period of time, the number of running nodes may change several times resulting in the termination of jobs that were running on downed nodes, but allowing others to continue. The scheduler is capable of marking unusable nodes as "down" and scheduling around them until they are repaired. Only systemic failures, e.g., to the file system or to the interconnect, are likely to take the whole system down. Thus the actual size of the system is dynamic, and the instantaneous utilization is the number of nodes in use divided by the total number of nodes available at that time. The utilization reported for a period is then the average of a number of samples taken over that period.

NICS uses a combination of Torque/PBS and Moab [7] for scheduling. Figure 2 displays the logical flowchart for scheduling on Kraken. Figure 2 is an idealized abstraction, as there are several other algorithms deciding whether a job is eligible to run, and at what priority. E.g., if a user has exhausted his allocation, his priority is reduced to "standby and the job will only run if otherwise the cycles would go idle. Also, users must provide reliable estimates of runtime. If the runtime estimate is exceeded, then the job is terminated and scheduling proceeds.

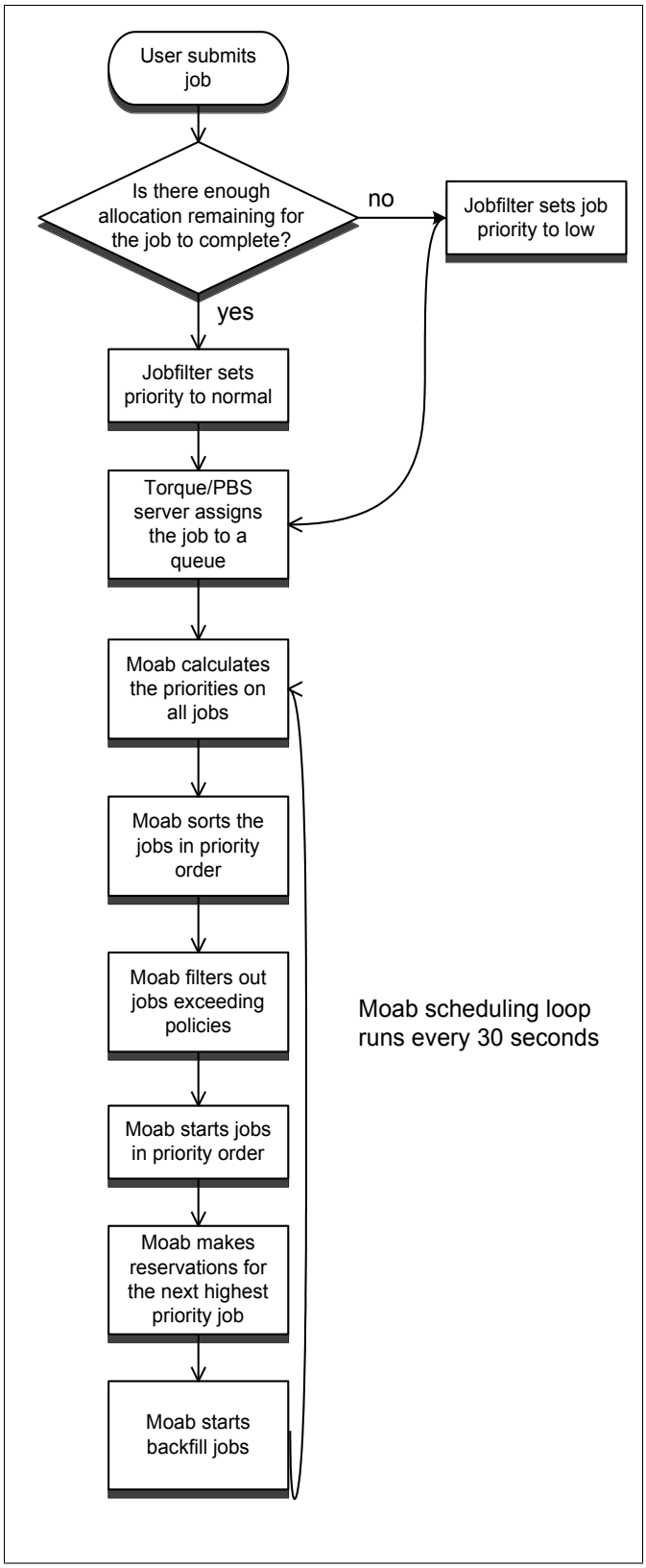

Figure 2: Logical Flowchart for Ideal Scheduling on Kraken

As discussed previously a common approach to allow for large jobs while attempting to maintain high utilization is to give jobs a priority that increases with time. How well does this work on an extremely large machine? Figure 3 shows the weekly utilization of Kraken since its upgrade to a Petaflop system (October, 2009) until the present (March, 2011). There are obviously large swings in utilization. This paper will show their causes and outline an approach that is currently delivering extremely good utilization of Kraken.

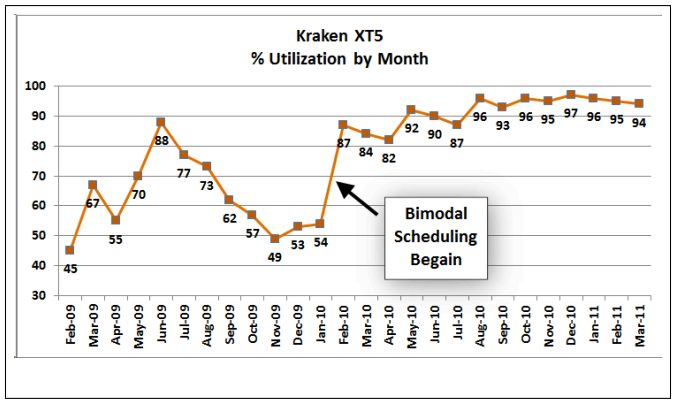

Figure 3: Percent Utilization by Month for Kraken XT5

3 Utilization

The following figures show detailed utilization profiles for individual weeks with sampling on an hourly basis. A hourly percent utilization (calculated as described previously) is shown as a vertical line. If there is no value to associate with a particular hour, e.g., the machine was down, or the script did not run for some reason, then the line is omitted and that point is excluded from the calculation of the average.

Figures 4 and 5 illustrate a week of hourly utilization patterns before implementation of of a single system-wide reservation for Wednesday morning and again after the implementation of this bimodal scheduling policy. The two periods have very different aggregate utilizations and hourly profiles.

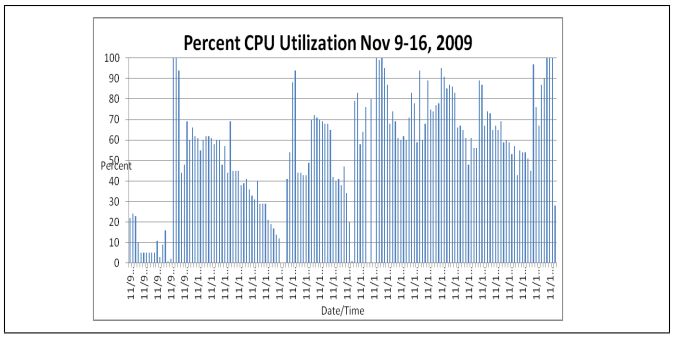

Figure 4: One Week of Hourly Utilization Patterns Before Bimodal Scheduling

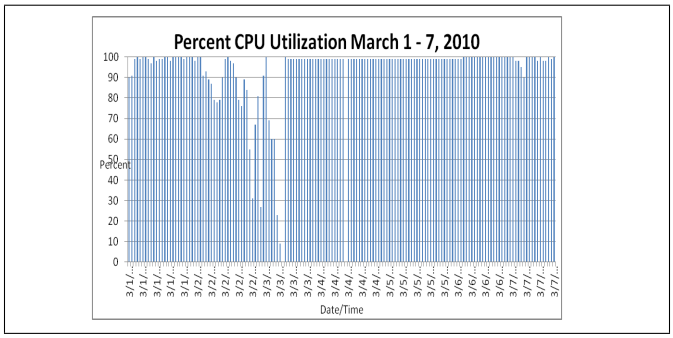

Figure 5: One Week of Hourly Utilization Patterns After Bimodal Scheduling

In Figure 4, the hourly utilization on Kraken for the week of November 9-16, 2009 capture several episodes of the scheduler clearing out the machine in order to start a relatively large job, followed by a downtrend in utilization moving toward the next large job. The aggregate utilization for that week was good for such a job mix, but still less than desirable.

In order to improve utilization NICS adopted the policy of a single system-wide reservation for Wednesday mornings, followed by sequential running of all large jobs (over 32k cores currently). These users received a discounted burn rate in exchange for the restriction in start time. Figure 5 illustrates the effectiveness of this approach. A single machine-wide reservation is enforced for 3/3/2010 and all large jobs in the queue are run immediately afterwards in a consecutive manner. The difference is startling, with an aggregate utilization of 92%, which is a very high rate for such a large machine with an inhomogeneous job mix. Very similar utilization profiles have been captured most weeks since bimodal scheduling was implemented, and the stark difference in utilization can be seen in Figure 3.

4 Production Usage

After 3 weeks of over 90% utilization to validate the bimodal scheduling policy, the policy was moved into a sustainable mode, and the large-run discount was fixed at 50%. In concert, NICS implemented policies to allow other procedures capable of co-existing with the large jobs (e.g., debugging runs) to share cycles. These co-existing procedures are essential for "backfill" [8] where shorter jobs can be run in the available space while waiting to clear the machine sufficiently to allow a larger job to start. The success of this policy is heavily dependent on the users ability to provide reliable estimates of runtime. If the runtime estimate is exceeded, then the job is terminated and scheduling proceeds.

Utilization varied as the policy was tuned, but as of March 2011, the last 7 months of operation have all been above 90% utilization; in fact, 6 of the 7 months were at 95% utilization or above, and the 12 month average utilization is 92%. This is an excellent (possibly unprecedented) utilization for a system of this size subjected to an inhomogeneous job mix like that on Kraken.

One question that needs to be addressed is how these high levels of utilization and capability job population affect the turnaround time for smaller jobs. There are 4 major classes of job on Kraken for scheduling purposes: small (up to 512 cores), medium (from there to 8,192 cores), large (from there to 49,536 cores), and capability (up to the full machine). The average queue wait times for all job sizes submitted from 04/01/10 to 05/01/11 (a period roughly representing all time since bimodal scheduling was moved to sustainable mode) are shown in Figure 6.

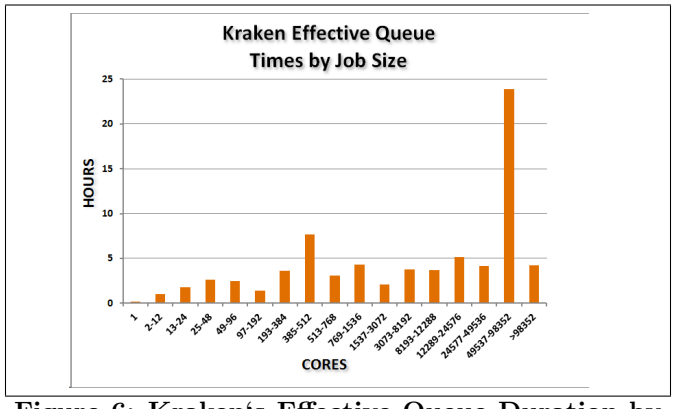

Figure 6: Kraken's Effective Queue Duration by Job Size for April 1, 2010 to May 1, 2011.

These numbers are very good for a extremely large machine, and shows that even with a large number of capability jobs running, it has still been possible to pro-

vide good responsiveness to the smaller jobs.

5 Conclusion

There are two simple solutions to maximizing utilization on a large supercomputer, one at each end of the job size regime. The first is to permit only single node jobs on the system; we are then guaranteed that so long as there are jobs to run, every node can be occupied. The second is to allow only full machine size jobs; each job then consumes the whole machine and one is run after the other, again keeping the whole machine busy. Sometimes, the first approach may actually masquerade as the second, when a single user combines many smaller, non-interacting jobs into a single job submission. This may be realistic in Monte-Carlo [9] or ensemble computing approaches, but unless all of the component jobs are guaranteed to have the same runtime, there can be major inefficiencies, though these could be disguised from the system monitors.

Away from academic speculation, the job mix on a capability system is guaranteed to include smaller jobs, while we hope it will include larger jobs, up to the full machine size. The efficient scheduling of such a job mix to optimize utilization has an importance that is proportional to the cost (capital and operational) of the system and can be truly massive on a world-class machine. Even more concerning, the desire to increase overall utilization can lead to the creation of obstacles for, or even the complete elimination of, large machine jobs, which could obviate complete classes of scientific research.

Analyzing the effect of bimodal scheduling coupled with incentive driven charging policies on Kraken has shown that excellent utilization rates (90+%) can be compatible with servicing an eclectic mix of jobs ranging in size from full machine to much, much smaller jobs [10]. This is of great importance for both the administration of existing HPC systems, and the design and configuration of future machines. Indeed, it helps to justify the increased capital investment necessary to create a capability computer by showing that it is possibly to both service large machine jobs and provide high utilization.

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.

Phil Andrews was the first Project Director of the National Institute for Computational Sciences at the University of Tennessee. He was the author of approximately 40 papers on grid and data intensive computing, documentation and visualization techniques, theoretical plasma physics, and nonlinear dynamics. He was involved in High Performance Computing for over 30 years, in management, software development, and as a user. He received a Ph.D. from Princeton University in theoretical physics, and a B.A. in applied mathematics from Cambridge.

Patricia Kovatch is the interim Project Director for the National Institute for Computational Sciences with a B.S. in Electrical Engineering from Pennsylvania State University. She can be contacted at pkovatch@utk.edu.

Justin Whitt is a Computational Scientist with a M.S. in Computational Engineering from the University of Tennessee at Chattanooga. He can contacted by email at jwhitt@tennessee.edu.

Victor Hazlewood is a Senior HPC Systems Analyst with 20 years experience in the HPC field. Victor has a B.S. in Computer Science from Texas A&M University and is a Certified Information Systems Security Professional. He can be contacted at victor@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] HPC. *TeraGrid: Analysis of Organization, System Architecture, and Middleware Enabling New Types of Applications*, 2006.

[2] teragrid[online]2010. *Available from: http://www.teragrid.org.*

[3] Jarek Nabrzyski, Jennifer M. Schopf, and Jan Weglarz, editors. *Grid resource management: state of the art and future trends*. Kluwer Academic Publishers, Norwell, MA, USA, 2004.

[4] R. Brightwell, K.T. Predretti, K.D. Underwood, and T. Hudson. Seastar interconnect: Balanced bandwidth for scalable performance. *Micro, IEEE*, 26(3):41 –57, may-june 2006.

[5] Martin Margo, Kenneth Yoshimoto, Patricia Kovatch, and Phil Andrews. Impact of reservations on production job scheduling. In Eitan Frachtenberg and Uwe Schwiegelshohn, editors, *Job Scheduling Strategies for Parallel Processing*, volume 4942 of *Lecture Notes in Computer Science*, pages 116–131. Springer Berlin / Heidelberg, 2008.

[6] Jonathan Weinberg and Allan Snavely. Symbiotic space-sharing on sdsc's datastar system. In Eitan Frachtenberg and Uwe Schwiegelshohn, editors, *Job Scheduling Strategies for Parallel Processing*, volume 4376 of *Lecture Notes in Computer Science*, pages 192–209. Springer Berlin / Heidelberg, 2007.

[7] W. Emeneker, D. Jackson, K. Butikofer, and D. Stanzione. Dynamic virtual clustering with xen and moab. *Frontiers of High Performance Computing and Networking*, (s.1), 2006.

[8] Edi Shmueli and Dror Feitelson. Backfilling with lookahead to optimize the performance of parallel job scheduling. In Dror Feitelson, Larry Rudolph, and Uwe Schwiegelshohn, editors, *Job Scheduling Strategies for Parallel Processing*, volume 2862 of *Lecture Notes in Computer Science*, pages 228–251. Springer Berlin / Heidelberg, 2003.

[9] J.S. Liu. *Monte Carlo Strategies in Scientific Computing*. s.1. Springer, 2003.

[10] Phil Andrews, Patricia Kovatch, Victor Hazlewood, and Troy Baer. Scheduling a 100,000 core supercomputer for maximum utilization and capability. In *Proceedings of the 2010 39th International Conference on Parallel Processing Workshops*, ICPPW '10, pages 421–427, Washington, DC, USA, 2010. IEEE Computer Society.