Comparison of Scheduling Policies and Workloads on the NCCS and NICS XT4 Systems at Oak Ridge National Laboratory

Troy Baer, National Institute for Computational Sciences, University of Tennessee; and **Don Maxwell,** National Center for Computational Sciences, Oak Ridge National Laboratory

ABSTRACT: Oak Ridge National Laboratory (ORNL) is home to two of the largest Cray XT4 systems in the world: Jaguar, operated by ORNL's National Center for Computational Sciences (NCCS) for the U.S. Department of Energy; and Kraken, operated by the University of Tennessee's National Institute for Computational Sciences (NICS) for the National Science Foundation. These two systems are administered in much the same way, and use the same TORQUE and Moab batch environment software; however, the scheduling policies and workloads on these systems are significantly different due to differences in allocation processes and the resultant user communities. This paper will compare and contrast the scheduling policies and workloads on these two systems.

KEYWORDS: XT4, scheduling, resource management

1. Introduction

Oak Ridge National Laboratory (ORNL) has long been the home of the National Center for Computational Sciences (NCCS), the first of the U.S. Department of Energy's Leadership Computing Facilities (LCF). The current flagship system of NCCS is Jaguar, a large Cray XT system that has been through multiple upgrades. However, in 2007 a partnership between ORNL and the University of Tennessee received an award from the second track 2 competition of the U.S. National Science Foundation's "Towards a Petascale Computing Environment for Science and Engineering" program. This resulted in a second supercomputing facility, the University of Tennessee's National Institute for Computational Sciences (NICS), being co-located at ORNL. The flagship system of NICS is Kraken, yet another large Cray XT system with a schedule for multiple upgrades. These two systems are literally side by side and use the same TORQUE and Moab batch environment software, but the scheduling policies and workloads on the two systems differ significantly. As a result, a comparison of the workloads of the two systems could potentially point out areas of improvement for scheduling on both systems.

To keep the number of differences between the two systems to a minimum, the period considered for this discussion is the fourth quarter of calendar year 2008 (4Q2008), i.e. 1 October 2008 through 31 December 2008. This is the only period over which the XT4 versions of both Jaguar and Kraken could be considered formally allocated, production computing resources before the centers' respective XT5 systems became available.

2. System Descriptions

During 4Q2008, Jaguar and Kraken were both Cray XT4 systems using quad-core Opteron processors. The two machines are located side by side in the same machine room at ORNL. They are however administered separately and share neither a user namespace nor any file systems.

2.1 Hardware and System Software

The following table describes the hardware specifications for the Jaguar and Kraken XT4 systems.

System	Jaguar	Kraken
Cabinets	84	40
Compute Nodes	7,832	4,508
Processor	AMD Opteron 2.1 GHz quad-core	AMD Opteron 2.3 GHz quad-core
Total Cores	31,328	18,032
Peak Performance (TFLOP/s)	263.2	165.9
Memory (TB)	61.19	17.61
Disk (TB)	700	450
I/O Bandwidth (GB/s)	40	12.5
MDS Nodes	3	2
OSS Nodes	72	24
Login Nodes	8	6
Aprun Nodes	8	6
Grid Nodes	0	4

During 4Q2008, both the Jaguar and Kraken XT4 systems used version 2.0 of the Cray Linux Environment (CLE), based on SuSE Linux 9. Jaguar has subsequently been updated to CLE 2.1.

2.2 Batch Environments

Both Jaguar and Kraken use the TORQUE variant of PBS [2] as their batch environment, with Moab [1] as the scheduler. Moab is a an extremely powerful and flexible scheduler that supports a wide variety of batch environments, including all PBS variants, 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 systems running CLE, Moab must communicate with the Cray Application Level Placement Scheduler (ALPS) service as well as the TORQUE batch system. This is accomplished by interfacing with a native resource manager, which is a set of "glue layer" scripts that sit on top of ALPS and TORQUE services.

2.2.1 Queue Structures

The queue structure on Jaguar is extremely simple, with only three user-accessible queues. Almost no policy is set or enforced by these queues; rather, all policy decisions were made either at submission time by the TORQUE submit filter or by Moab.

Jaguar Queue Name	Maximum Processor Core Count	Maximum Wallclock Time Limit
`dataxfer`	0	24:00:00
`batch`	31,328	24:00:00
`debug`	31,328	4:00:00

A major policy component enforced by the TORQUE submit filter is the maximum wallclock time limit allowed to jobs as a function of the number of cores requested. This is done to keep large volumes of small jobs from increasing the queue time for very large jobs.

Jaguar Maximum Core Count	Maximum Wallclock Time Limit
0	12:00:00
256	1:30:00
512	2:30:00
1,024	4:00:00
31,328	24:00:00

Initially the queue structure on Kraken was almost identical to that of Jaguar, except for the smaller size of the machine. However, a slightly more complex queue structure was implemented in September 2008 to allow for more fine-grained control over wallclock time limits based on job size. This also allowed for considerable simplification of the TORQUE submit filter.

Kraken Queue Name	Maximum Processor Core Count	Maximum Wallclock Time Limit
`dataxfer`	0	24:00:00
`small`	512	12:00:00
`longsmall`	512	60:00:00
`medium`	2,048	24:00:00
`large`	8,192	24:00:00
`capability`	18,032	48:00:00

The queue names `batch` and `debug` were retained for compatibility reasons; however, they were converted into routing queues with the above queues as destinations, with the `debug` queue enforcing a 2 hour maximum wallclock time limit to mimic its previous behavior.

The `longsmall` queue was not originally part of the revised queue structure on Kraken. It was requested by a group of users who objected to the 12-hour time limit on the `small` queue on the grounds that such a limit forced them to have to submit jobs too often. In exchange for allowing longer small jobs, these users agreed that said jobs would be limited to a relatively small fraction of the machine, as discussed below.

2.2.2 Scheduling Policies

On Jaguar, jobs that use a significant fraction of the machine are given the highest priority. Priorities are considered in units of days, equivalent to one day of waiting to run. Another major priority component in Moab on Jaguar is accomplished by assigning one of five QoS levels to a job using Moab's job template feature.

Jaguar QoS Level	Min. Core Count	Max. Core Count	Priority in days	Fair Share Target	Jobs per User
`sizezero`	0	0	90	None	10
`smallmaxrun`	1	256	0	20%	2
`nonldrship`	257	6,000	0	20%	None
`ldrship`	6,001	17,000	8	80%	None
`topprio`	17,001	31,328	10	80%	None

These QoS levels also include fair share targets, which are used to tune job priorities dynamically such that large jobs get approximately 80% of all cycles delivered by the system. The `sizezero` QoS is excluded from the fair share system, as jobs in this QoS are primarily in the `dataxfer` queue and are used to stage data in and out of HPSS. Furthermore, the `sizezero` QoS limits users to 10 running jobs apiece, and the `smallmaxrun` QoS limits users to 2 running jobs apiece.

Like Jaguar, Kraken's scheduling policies are intended to favor jobs that use a significant faction of the machine; however, the mechanisms used to accomplish this are quite different. Priority in Moab on Kraken is based primarily on the number of cores requested; secondary factors used in the priority calculation include the job's queue time and its *expansion factor*. The expansion factor of a job is the ratio of the sum of the job's queue time and run time to the run time:

$$f_{\exp}(j)=\frac{t_{queue}(j)+t_{run}(j)}{t_{run}(j)}=1+\frac{t_{queue}(j)}{t_{run}(j)}$$

The expansion factor quantifies the overhead of the batch system in a job's overall time to completion. Weighting the expansion factor in the priority calculation gives higher priority to short-running jobs that have waited longer than their requested time to run.

As job classification on Kraken is done primarily at the queue level, QoS levels are used less extensively on Kraken than on Jaguar. Only two QoS levels are generally used on Kraken.

Kraken QoS Level	Base Priority	Queue Time Target	Queue Time Priority
`sizezero`	0	0:00:01	5,000
`negbal`	-100,000	None	None

The `sizezero` QoS has the same intent as the similarly named QoS on Jaguar, and is applied using the same job template mechanism. However, rather than having an increased base priority, jobs in the `sizezero` QoS on Kraken have a short queue time target, after which their priorities begin to increase rapidly. On the other hand, the `negbal` QoS is used to de-prioritize jobs from projects whose allocations are overdrawn; it is applied automatically by the system's TORQUE submit filter at job submission.

Kraken also has additional policies applied to two of its queues. The discussion which led to the creation of the `longsmall` queue stipulated that jobs in that queue would be limited to using a small fraction of the machine so that they would not block jobs requesting the largest power of 2 available on the system (16,384). To also allow for node failures, this limit on `longsmall` jobs is configured to be 1,600 cores. In addition, user complaints about `capability` jobs hogging the system resulted in that queue being limited to one job eligible to run at a time.

3. Allocation Processes

The processes used to allocate time on these two systems are very different. Allocations for Jaguar are handled primarily by the U.S. Department of Energy's Innovative and Novel Impact on Theory and Experiment (INCITE) allocation program [3]. Allocations by INCITE are made annually, with allocations potentially lasting for multiple years. A significant component of the INCITE review process is that a project must be able to demonstrate that their application has the ability to run at scale -- that is, using a major fraction of the LCF systems at Argonne and Oak Ridge National Laboratories. This has the effect of limiting INCITE allocations to approximately 20 projects per year with massively scalable codes.

In contrast, allocations for Kraken are done primarily by the National Science Foundation's Teragrid Resource Allocations Committee (TRAC) [4]. TRAC allocations are made quarterly, with allocations expiring after one year. Unlike the INCITE program, the TRAC has no requirement that applications scale to a significant fraction of the machine requested, so any project can in principle get an allocation on any Teragrid system. TRAC allocated projects are also allowed to shift balances between Teragrid sites, and a number of projects have shifted balances to NICS after discovering that Kraken outperformed other Teragrid systems on codes that have arguably modest scalability.

4. Workload Analysis

The following analyses were performed using software developed by the authors [5, 6], much of which is open source. The TORQUE accounting logs from the two machines were parsed and fed into a MySQL database, which were then be queried either manually or using a set of web forms to do pre-defined reports. For Kraken, users' job scripts were also stored in the database for full-text searches; on Jaguar, the job scripts were partially reconstructed using logs of the jobs' `aprun` commands stored in another database.

4.1 Overall Utilization

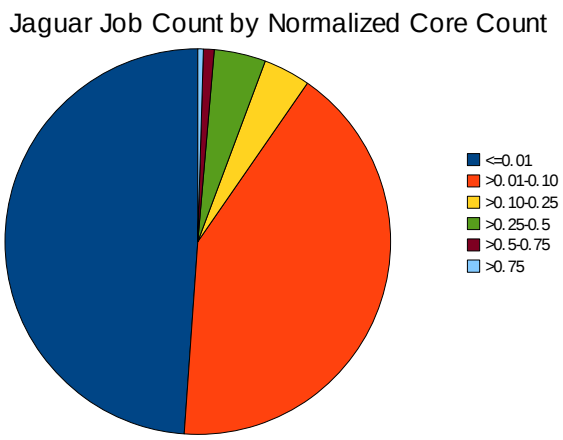
The overall utilization on Jaguar and Kraken during 4Q2008 is summarized in the following table.

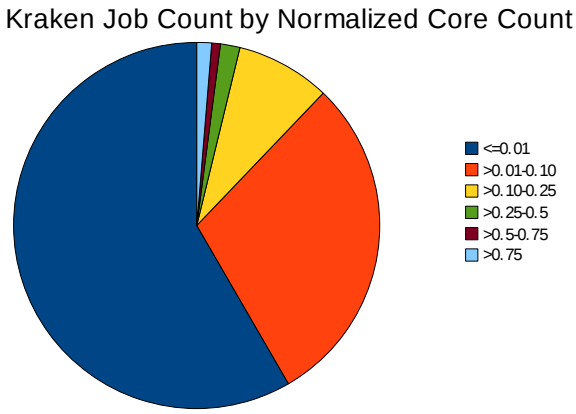
4Q2008	Jaguar	Kraken
Jobs Run	46,023	15,774
CPU Hours Used	54.46M	21.00M
Average Utilization	89.7%	57.0%
Active Users	300	116
Active Projects	142	40

Kraken had lower overall utilization than Jaguar for primarily historical reasons. As a new center at the beginning of 2008, NICS effectively started from zero in terms of users and allocations, whereas NCCS had several years worth of preexisting users and allocations. Furthermore, the TRAC process had only allocated approximately 63 million CPU hours on Kraken as of the beginning of 4Q2008; as a result, many allocations on Kraken had already significantly depleted their allocation balances by the end of the quarter.

4.2 Job Breakdown by Size

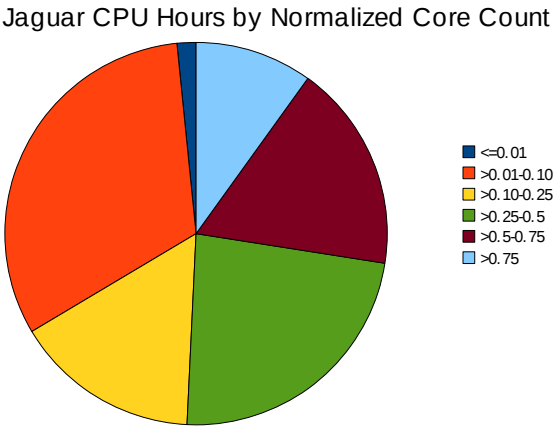
Since Jaguar and Kraken have very different queue structures, a comparison between the two cannot be done at the queue level. A breakdown based solely on jobs' requested processor core count is not entirely helpful either, as Jaguar is almost twice the size of Kraken and therefore capable of running much larger jobs. However, a breakdown by processor core count normalized by the size of the machines allows for a meaningful direct comparison. The following analyses break down jobs into one of six categories: those that request less than 1% of the system's processor cores, those that request between 1% and 10% of the cores, those that request between 10% and 25% of the cores, those that request between 25% and 50% of the cores, those that request between 50% and 75% of the cores, and those that request more than 75% of the cores (i.e. effectively the whole system).

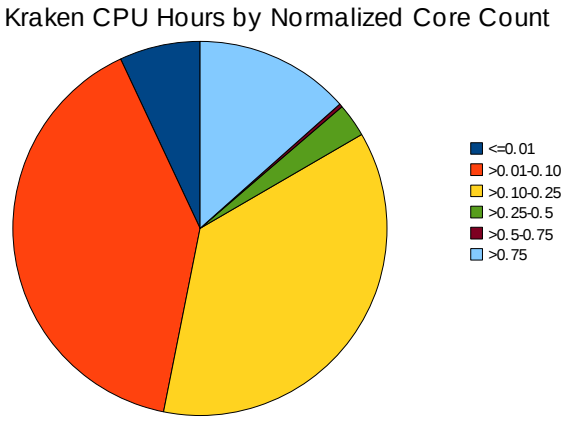




On both systems, approximately 90% of the jobs submitted request less than 10% of the machine. Kraken's job mix is skewed more heavily to the small side with 58.4% of all jobs requesting 1% or less of the system, whereas 48.8% of all jobs on Jaguar request 1% or less of the system. However, the breakdown of the remaining 10% of jobs is very different between the two systems. On Jaguar, 3.94% of all jobs request between 10% and 25% of the system, 4.32% request between 25% and 50% of the system, 0.90% request between 50% and 75% of the system, and only 0.48% more than 75% of the system. On Kraken, this distribution is rather different; 8.35% of Kraken jobs request between 10% and 25% of the system, 1.72% request between 25% and 50% of the system, 0.77% request between 50% and 75% of the system, and 1.31% request 75% or more of the system. This skew toward whole-system jobs on Kraken relative to Jaguar is a result of the sizes of the systems relative to the largest possible power of 2 in core count on them; on Jaguar, 16,394 cores are a little more than 52% of the machine, whereas on Kraken, it is 91% of the machine.

However, the numbers of jobs using various fractions of the systems are only part of the story. On both Jaguar and Kraken, jobs which request a relatively small portion of the machine are limited in terms of the amount of wallclock time they may request, so that larger jobs can be turned around more quickly. As discussed previously, this limitation is more strict on Jaguar than on Kraken, due to the much wider variability of code scalability in projects allocated time on Kraken. Furthermore, the scheduling policies on both systems are also intended to favor usage by very large jobs. As a result, a breakdown of the number of CPU hours consumed by different groups of jobs based on the normalized size of the jobs is instructive as far as being able to determine quantitatively whether the scheduling policies are successful in achieving their goals.

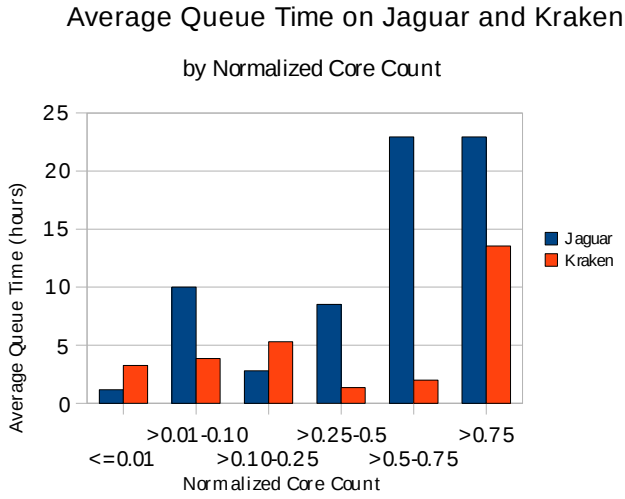




As might be expected, jobs that requested relatively small fractions of the system used a much smaller portion of Jaguar's resources than those of Kraken. On Jaguar, jobs which requested 10% or less of the system consumed 33.6% of all CPU hours delivered, but on Kraken, they consumed 46.9%. Regarding larger jobs, jobs requesting 25% or more of Jaguar consumed 50.8% of all CPU hours, while on Kraken they used 16.6% of all CPU hours. However, jobs requesting more than 75% of Jaguar (i.e. whole-system jobs) consumed 9.9% of the total CPU hours, whereas on Kraken they accounted for 13.4%. Another interesting facet of this is the large disparity between the two systems as far as jobs which use between 50% and 75% of the system. On Jaguar, these jobs consumed 17.5% of the total CPU hours, whereas on Kraken they consumed only 0.8%. In effect, Kraken has a very bimodal job distribution with a large volume of small, long-running jobs and a significant number of whole-system jobs with relatively little in between, while Jaguar has a more evenly distribution of job sizes than skews more toward large but not whole-system jobs.

4.3 Quantifying User Experience

It can be difficult to gauge users' experiences with a system quantitatively. However, there are a few metrics which can be used to make inferences. The metric that is most immediately visible to end users is queue time, the time between when a job is submitted and when it starts running.

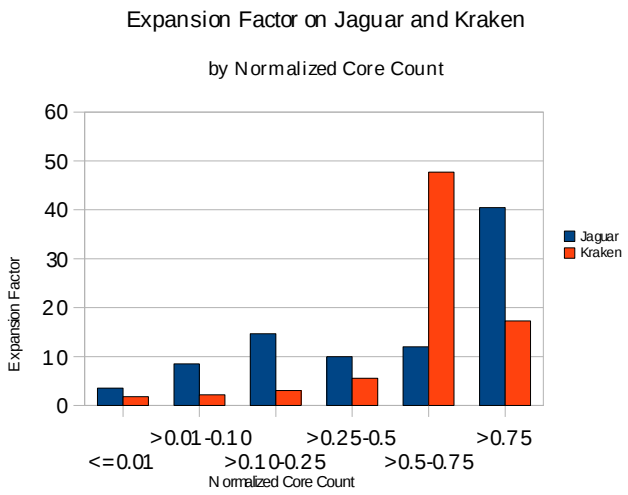


Queue times on Jaguar were, on average, shorter at smaller processor cores count and longer at larger core counts than on Kraken. At the low end, this is a reflection of the longer run times allowed on Kraken for small jobs, making them more difficult to backfill between large jobs. An additional factor in the longer queue times at the low end on Kraken is the 1600-core limit on `longsmall` jobs; as there are many of these jobs contending for the limited fraction of the system, they tend to have much higher queue times. At the high end, the shorter queue times on Kraken than on Jaguar are a reflection of the fact that Jaguar had higher overall utilization than Kraken did.

Considering queue time as a monolithic quantity can be misleading, however. There are two significant components of queue time: queue time due to resource availability, such as a job not running due to not enough processor cores being available; and queue time due to policy reasons, such as a job not running because the job's owner has reached the limit of concurrent running job that they are allowed. Disambiguating between these two aspects of queue time is virtually impossible using only the TORQUE accounting information, as it requires access to policy information known only by the scheduler, Moab in the case of Jaguar and Kraken. Being able to quantify queue time due to resource availability versus queue time due to policy reasons is an area of ongoing effort.

Another metric to consider with regard to user experience is the expansion factor. As mentioned previously, the expansion

factor of a job is the ratio of the sum of the job's queue time and run time to the run time. Ideally, this number should be unity.

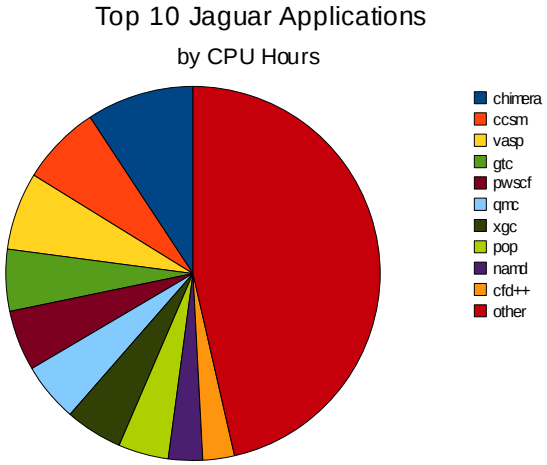


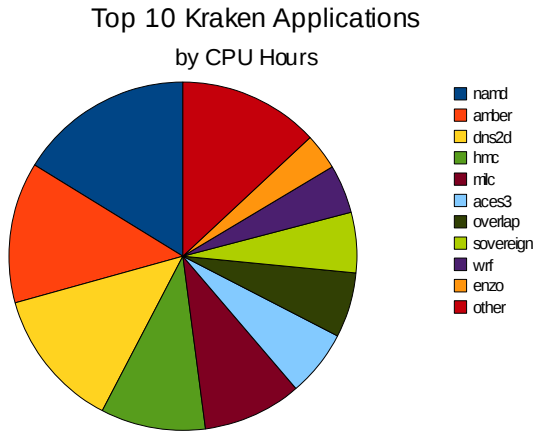
Kraken had generally smaller expansion factors than Jaguar. This is a result of Jaguar having significantly higher overall utilization than Kraken. However, the one exception to this was for jobs using between 50% and 75% of the system's processor cores, where Kraken's expansion factor was almost four times that of Jaguar. As discussed previously, the number of jobs in this group was extremely small on Kraken, as was the amount of resources that they consumed.

4.4 Application Usage

At the end of the day, what really matters with all scientific computing systems is not the hardware and software they use or the scheduling policies they implement, but rather the scientific advanced made using results generated by the applications running on them. By having the users' job scripts stored in a database along with the accounting information, it is relatively straightforward to develop queries to identify jobs that run a particular application. This allows a site to quantify which applications are most heavily used and emphasize support accordingly.

The application mix on Jaguar was considerably more diverse than that on Kraken, due to NCCS' longer history and larger user community than those of NICS. On Kraken, the ten most heavily user applications consumed 87% of the total CPU hours delivered, whereas on Jaguar the top ten applications consumed only 54% of the total CPU hours. The mix of applications is also very different between the two systems, with only one application (NAMD) in common between the top ten applications lists for the two.





5. Conclusions and Future Work

Several observations may be drawn from the above analyses. The scheduling policies implemented on both Jaguar and Kraken have proven effective in achieving the systems goals of running jobs which use significant fractions of that machine. This has been somewhat more successful on Jaguar than on Kraken, largely because the INCITE allocation process sends only highly scalable projects to Jaguar while the TRAC allocation process sends a much different community to Kraken. On the other hand, Kraken also has maintained good quality of service to its users with whole-system jobs despite its more diverse workload.

There are of course a number of areas for potential improvement in the scheduling of both Jaguar and Kraken. One idea that has been discussed at NICS is implementing a

fair share system at the project allocation level, with the fair share target of a project being based on its remaining allocation balance. There has also been discussion of a more fine-grained queue structure on Jaguar with explicit time limits, more like what is done on Kraken. Another major factor going forward is that NCCS and NICS did acceptance testing of XT5 versions of Jaguar and Kraken respectively in late 2008 and early 2009. These new systems are four to five times larger than their predecessors, and adjustments in scheduling policies will certainly be required to accommodate the increased size of the systems. Indeed, the XT5 incarnation of Kraken has already had to change some of the policies applied to `longsmall` and `capability` jobs to improve turnaround time for whole-system jobs.

About the Authors:

Troy Baer is an HPC systems administrator for the University of Tennessee's National Institute for Computational Sciences at Oak Ridge National Laboratory. He can be reached by emailing *<tbaer@utk.edu>*.

Don Maxwell is a senior HPC systems administrator for the National Center for Computational Sciences at Oak Ridge National Laboratory. He can be reached by emailing *<mii@ornl.gov>*.

References

[1] "Cluster resources :: Products - Moab Workload Manager", http://www.clusterresources.com/pages/products/moab-cluster-suite/workload-manager.php.

[2] "Cluster resources :: Products - TORQUE Resource Manager", http://www.clusterresources.com/pages/products/torque-resource-manager.php.

[3] "DOE - Science - ASCR - INCITE", http://www.sc.doe.gov/ascr/incite/.

[4] "Teragrid User Support: Access: Access & Allocations", http://www.teragrid.org/userinfo/access/allocations.php.

[5] Baer, T. "OSC Configuration and Tools for PBS", http://www.osc.edu/~troy/pbstools/.

[6] Maxwell, D, et al. "Restoring the CPA to CNL", *Proceedings of CUG 2008*, Helsinki, May 2008.