A Tale of Two Systems

Flexibility of Usage of Kraken and Nautilus at the National Institute for Computational Sciences

Amy F. Szczepański
Remote Data Analysis and Visualization Center
Electrical Engineering and Computer Science
University of Tennessee
1520 Middle Drive
Knoxville, TN 37996-2250
aszczepa@utk.edu

Jian Huang
Remote Data Analysis and Visualization Center
Electrical Engineering and Computer Science
University of Tennessee
1520 Middle Drive
Knoxville, TN 37996-2250
huangj@utk.edu

Sean Ahern
Remote Data Analysis and Visualization Center
National Institute for Computational Sciences
University of Tennessee
PO Box 2008, ORNL Bldg. 5100
Oak Ridge, TN 37831-6173
ahern@utk.edu

Mark R. Fahey
National Institute for Computational Sciences
Industrial and Information Engineering
University of Tennessee
PO Box 2008, ORNL Bldg. 5100
Oak Ridge, TN 37831-6173
mfahey@utk.edu

ABSTRACT

The National Institute for Computational Sciences (NICS) at the University of Tennessee currently operates two computational resources for the eXtreme Science and Engineering Discovery Environment (XSEDE), Kraken, a 112,896-core Cray XT5 for general purpose computation, and Nautilus, a 1,024-core SGI Altix UV 1000 for data analysis and visualization. We analyze a year's worth of accounting logs for Kraken and Nautilus to understand how users take advantage of these two systems and how analysis jobs differ from general HPC computation We find that researchers take advantage of the flexibility offered by these systems, running a wide variety of jobs at many scales and using the full range of core counts and available memory for their jobs. The jobs on Nautilus tend to use less walltime and more memory per core than the jobs run on Kraken. Additionally, researchers are more likely to run interactive jobs on Nautilus than on Kraken. Small jobs experience a good quality of service on both systems. This information can be used for the management and allocation of time on existing HPC and analysis systems as well as for planning for deploying future HPC and analysis systems.

XSEDE12 July 2012, Chicago, IL

Categories and Subject Descriptors

K.6 [**Computing Milieux**]: Management of Computing and Information Systems

General Terms

Management

Keywords

management, analysis, measurement

1. INTRODUCTION

As of spring 2012, the National Institute for Computational Sciences (NICS) at the University of Tennessee operates two high performance computing (HPC) systems for the eXtreme Science and Engineering Discovery Environment (XSEDE). Kraken, the flagship system at NICS, is a Cray XT5 with 112,896 cores and a peak performance of 1.17 petaflops. It has 9,408 compute nodes; each node has two hex-core AMD Opteron processors and 16 GB of memory. Nautilus, the centerpiece of NICS' Remote Data Analysis and Visualization Center, is an SGI Altix UV 1000 with 1,024 cores and 4 TB of global shared memory. Each of Nautilus' 128 non-uniform memory access (NUMA) nodes has an eight-core Intel Nehalem processor and 32 GB of memory. Kraken, with its large core count and distributed memory architecture, is used for a wide range of traditional HPC applications. Nautilus is primarily intended for data analysis and visualization, especially those jobs that can take advantage of its large memory.

In this paper we examine a year's worth of accounting logs representing the actual usage of Kraken and Nautilus in order to quantify the differences between how XSEDE users

make use of a general HPC system and an analysis system. Understanding the way that researchers use current analysis and visualization resources, especially in concert with XSEDE's only system that currently operates at the petascale, is important for planning the next generation of data analysis and visualization systems. Bethel *et al.* [2] reinforce the importance of planning for the visualization and analysis needs of researchers in addition to their simulation needs. Both the creation and interpretation of data are necessary for scientific discovery.

As the 10 petaflop Stampede system at the Texas Advanced Computing Center (TACC) is scheduled to come online in 2013 [10], XSEDE will need to deal with the deluge of data produced by users of this system. Understanding the data analysis and visualization demands of Kraken-users is an important step in understanding the upcoming demand. Comparison of the usage of the Spur and Longhorn visualization and analysis systems at TACC with their current Lonestar and Ranger systems as well as comparisons of the usage of Gordon at the San Diego Supercomputer Center (SDSC) with other XSEDE systems would complement this analysis and give a fuller picture of the data analysis and visualization needs of simulations run on XSEDE systems. Through conversations with colleagues at the Oak Ridge National Laboratory (ORNL), we observe that the Department of Energy's HPC systems may exhibit different patterns of usage than XSEDE's systems. These differences should be kept in mind when extrapolating to future needs in the XSEDE research community.

Users take advantage of the flexibility of both Kraken and Nautilus and run a wide variety of jobs. We observe that users on both Kraken and Nautilus run jobs at a wide range of core counts. Furthermore, requests for memory-per-core on Nautilus also vary widely. Both systems have a small, but significant, number of users that require the unique resources (either large core count or large memory) that are offered by the system.

We measure the mix of jobs and the quality of service for different classes of users in hopes of understanding how users are taking advantage of the unique capabilities of each system. In particular, we quantify how analysis jobs differ from computational jobs. Because Nautilus is a much smaller system than Kraken, we look at particular subsets of Kraken users so that our description of the properties of *analysis* jobs is not merely an an examination of the properties of *small* jobs. We do this by first comparing the jobs run on Nautilus to the projects that only run small jobs on Kraken, so that we can compare small HPC runs to the analysis jobs to see how they differ. Next, we look at projects that use both Kraken and Nautilus and compare the ways that they use these two systems.

We note that while Kraken and Nautilus are both operated by NICS, they do not currently share a scratch filesystem. Thus, users of both systems must move their data, by staging it through the archival storage system, by using tools such as `gridftp`, or by using other data-moving techniques. Thus, to some extent our analysis can be generalized to situations where the simulation system and analysis system are not co-located and data movement must take place between simulation and analysis.

In addition to both systems being used flexibly, we find other commonalities as well. Small, traditional HPC jobs on Kraken and jobs run on Nautilus use roughly the same number of cores, on average. However, there are also several differences in how researchers use the systems. Small HPC jobs on Kraken tend to run for longer walltimes than jobs on Nautilus, but the jobs on Nautilus use more memory per core. Users were more likely to run interactive jobs on Nautilus than on Kraken. Both systems offer a good quality of service for jobs up to 1024 cores, measured either by time spent waiting in the queue or by the scheduler expansion factor. While some projects on Kraken could move to Nautilus to improve utilization of the latter, users would only experience a modest improvement in quality of service as measured either by wait time in the queues or by the scheduler expansion factor.

2. DATA AND COLLECTION

NICS uses TORQUE as the resource manager for both Kraken and Nautilus, and information about each job is extracted from the TORQUE accounting logs and stored in a database by the workload analysis system of `pbsacct` [8], developed by the Ohio Supercomputer Center and NICS as part of the PBS Tools project. The `pbsacct` system periodically parses the TORQUE accounting logs and extracts information such as the job identifier, the username, the time of job submission, the time that the job begins running, the time that the job was completed, the job's project charge code, the queue to which the job is submitted, the job's maximum memory usage, and what resources were requested by the job, including number of cores, memory limit, and wallclock time limit. Users' job scripts are also captured at submission time and stored in the database.

In analyzing the logs, we looked at data from a one-year period of March 8, 2011 to March 7, 2012. This time period was chosen because it was the most recent one-year period that avoided the most significant upgrades or changes to the systems; we feel that this timeframe best represents a year of typical usage of Kraken and Nautilus. Including all jobs from all projects, Kraken had a utilization of 91% and Nautilus had a utilization of 41% during this year. Excluding the staff projects, this represents 493 projects on Kraken running 648,918 jobs for 896,317,277 CPU-hours and 62 projects on Nautilus running 42,019 jobs for 3,497,276 CPU-hours. From this point on, unless otherwise specified, jobs charged to the staff projects or run in the data-moving queues are excluded from this study.

3. CHARACTERIZING OVERALL USAGE OF KRAKEN AND NAUTILUS

One notable feature of the usage of both Kraken and Nautilus is that the users take full advantage of the flexibility of these systems. For both systems we look at both the number of and the size of jobs running at different core counts as well as the range of memory requests made on Nautilus.

Users on Kraken run at a variety of scales, with substantial usage seen for a wide variety of job sizes, not just the large-scale jobs that require a resource of the size of Kraken. Jobs on Kraken are assigned to the *small*, *medium*, *large*, *capability*, or *dedicated* queue based on the number of cores requested. The mix of jobs, organized by queue, on Kraken, both in terms of number of jobs and CPU-hours consumed, is shown in Figure 1. Fewer than 5% of projects on Kraken ran jobs in the capability or dedicated queues.

To demonstrate the diversity of job-sizes on Kraken, we

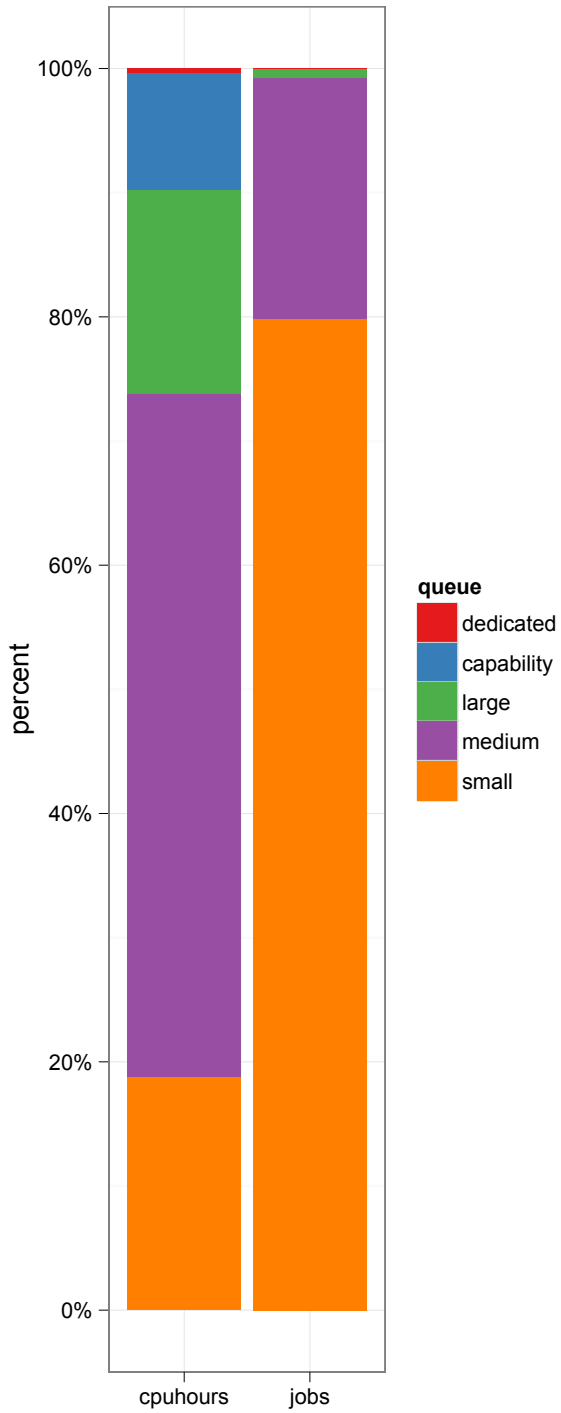

Figure 1: Breakdown of jobs on Kraken. The small, medium, and large queues represent jobs of up to 512, 8,192, and 49,536 cores, respectively. Jobs in the capability queue use between 49,537 and 98,352 cores, and the dedicated queue is for jobs requesting from 98,353 cores up to Kraken's limit of 112,896 cores.

contrast the usage of Kraken to that of Jaguar. Kraken shares a machine room with the Oak Ridge Leadership Computing Facility's Jaguar at ORNL. Prior to beginning its upgrade to the new Titan system, Jaguar was a 224,256-core Cray XT5 with 18,688 nodes, each with two hex-core AMD processors. During the fall of 2011, Jaguar's transformation into Titan limited the number of cores available to users (ranging to a low of 117,120 to Jaguar's full 224,256) and later upgraded the CPUs to 16-core AMD processors [7].

Big jobs make up a larger fraction of the job mix on Jaguar than they do on Kraken. If we look at Kraken's capability and dedicated queues, which run jobs that require at least 49,537 cores—or 43% of the cores available on Kraken—these represent roughly 10% of the CPU-hours run on Kraken. Comparing to Jaguar, in 2011 roughly 25% of the CPU-hours were used in jobs that required at least 40% of Jaguar, and an even larger fraction of the CPU-hours on Jaguar were devoted to jobs of over 50,000 cores [6]. This pattern of usage on Kraken is reflected in Hart's finding that XSEDE resources, Kraken included, show a diverse mix of capacity and capability computing [4]. This wide range of usage is expected to continue as more users access these systems through science gateways [12].

Similarly, we see a wide range of usage for Nautilus. Approximately 18% of the CPU-hours are used in jobs that require at least half the system (at least 512 cores). The queuing system on Nautilus allows users to specify both the desired number of cores and amount of memory, and the request is filled with the smallest number of NUMA nodes that will accommodate both the cores and memory requested. We also see a large diversity of resource requests when we look at both cores and memory. This mix is shown in Figure 2.

Another way to look at this information is to look at the ratio of the amount of memory requested to the number of cores requested as well as the ratio of the amount of memory used (for jobs that used more than zero memory—that is, those that did not crash immediately) to the number of cores requested. While, on average, jobs on Nautilus asked for 5.78 GB of memory per core and used 1.58 GB of memory per core, there was a great deal of spread due to a small number of jobs requesting—and using—hundreds of GB of memory per CPU requested.

Overall, 9.7% of jobs requested more than 4 GB of memory per CPU and 4.4% of jobs used more than 4 GB of memory per CPU. Additionally, 19.2% of jobs on Nautilus used a larger ratio of memory per CPU than the 1.33 GB of memory per core available on Kraken. Again, we see a wide diversity of jobs being run on the system. When analyzing data about the amount of memory used by jobs, we exclude jobs that used exactly 0 MB of memory but include those that used non-zero but trivial amounts of memory that round to zero.

One other similarity that we see with these systems is that usage by projects does *not* follow the 80/20 rule that would be predicted by the Pareto principle. We find that on both Nautilus and Kraken that 10% of projects are responsible for 90% of the usage of the system. These are not just the projects that run large core-count jobs; large consumers of CPU-hours run at a wide variety of scales.

4. COMPARING ANALYSIS TO COMPUTATION

It is not remarkable that there are differences in usage

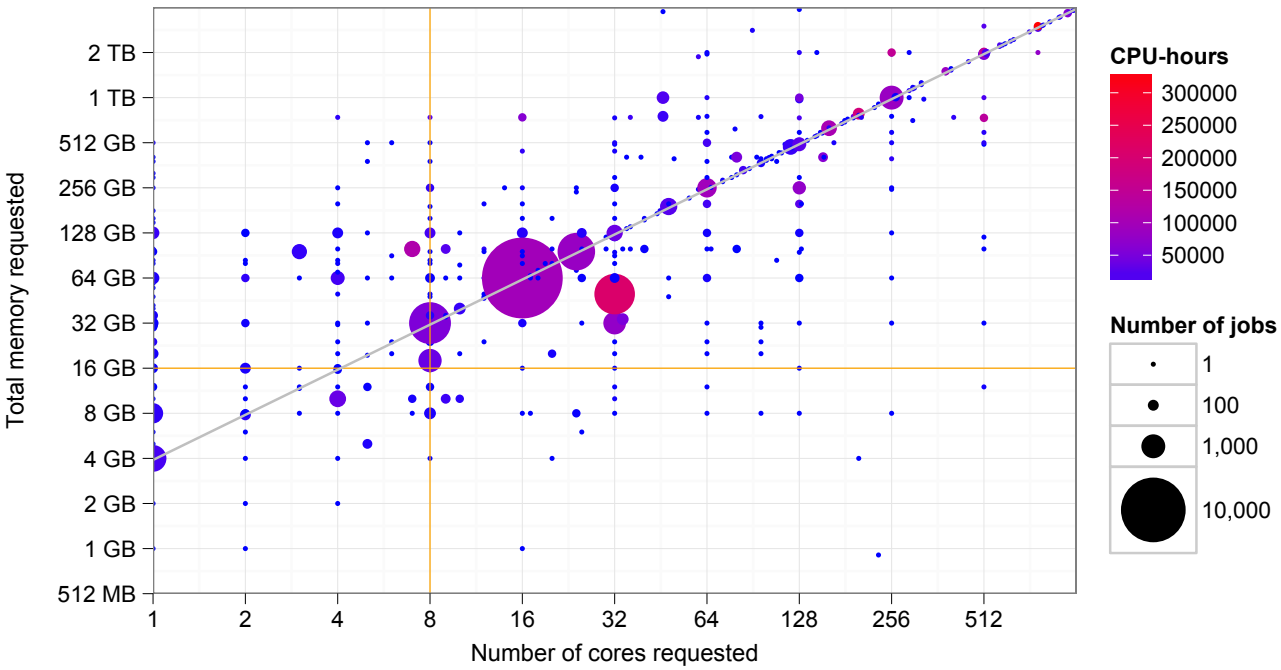

Figure 2: Mix of jobs on Nautilus based on resources requested. Orange lines at 8 cores and 16 GB of memory divide the space into jobs that could be run on a nice desktop computer (lower-left quadrant) and those that require an HPC system (remaining three quadrants). The grey line represents a request of 4000 MB per core, which is the default if the user does not specify a memory request. The size of the dot represents the number of jobs requesting that combination of resources, and the color shows the sum total CPU-hours consumed by all jobs making that request. The large dot at 16 cores and 64 GB of memory represents 16,133 jobs.

between a very large, traditional HPC system and a smaller shared memory system meant for visualization and analysis. However, there are some meaningful observations that can be made about well-chosen subsets of users. We will make observations based on the usage of select subgroups of users.

1. Projects on Kraken whose jobs were all small enough to run on Nautilus.

2. Projects that ran jobs on both Nautilus and Kraken.

By looking at these groups of users, we can make comparisons about how users take advantage of these systems and draw conclusions about how visualization and analysis jobs differ from computation jobs. We will compare this collection of jobs run on Kraken to the jobs run on Nautilus.

Our analysis will describe not only the size of the jobs, measured in terms of resources consumed but also the quality of service. These depend not only on the jobs requested and the loads on the system but also on the queueing policies set by NICS.

The queues on Kraken are determined solely by the number of cores requested, and jobs with the largest core counts (up to 49,536 cores) have the highest priority. Jobs requesting at least 49,537 cores only run during specified "capability" periods [1]. Nautilus, on the other hand, has two production queues, *analysis* and *computation*. The analysis queue is for data analysis and visualization and has higher priority. The computation queue is for computation and simulation, which has a lower priority and is subject to pre-emption by jobs in the analysis queue. Furthermore, the computation queue on Nautilus is limited to walltime requests of at most six hours. The small, medium, and large queues on Kraken and the analysis queue on Nautilus are all limited to 24 hours of walltime requested. Some jobs do run longer than the maximum time for their queue, either by special arrangement with the staff at NICS or due to aberrant behavior of the system.

4.1 Small projects on Kraken vs. all projects on Nautilus

Of the 493 non-staff projects that ran jobs on Kraken during the timeframe, 244 of those projects had all of their jobs require at most 1,024 cores. These projects ran a total of 115,852 jobs on Kraken, totaling 69,633,811 CPU-hours. As Nautilus has 1,024 cores and more memory-per-core than Kraken does, any of these projects could have run on either Kraken or Nautilus. We note that they could not have *all* run on Nautilus instead of Kraken. Even disregarding the fact that Nautilus is designated preferentially for visualization and analysis use and not for general-purpose computation, together these projects consumed over seven times as many CPU-hours as are available on Nautilus running at full-tilt for an entire year. While a few of these projects may be working on scaling their code up to do large runs, many of them consistently run well below the petascale capability that Kraken offers. Because these projects are running exclusively at under 1,024 cores for an entire year, we find them to be representative of work at this scale.

We summarize the size of the jobs under consideration in Tables 1 and 2. As Nautilus has separate queues for analysis and for computation, we break those out in the tables. Users self-select, on the honor system, which queue to submit their jobs to. We see from Table 1 that the small jobs run on Kraken use similar number of processors. If we calculate this in terms of nodes (noting that there are 8 cores per node on Nautilus and 12 cores per node on Kraken), we find that these jobs use a mean of 5.4 nodes on Kraken and 5.3 nodes on Nautilus. Where we see a difference between the systems is in the length of time that the jobs run. On average, jobs run for a longer period of time on Kraken. There were a few outlier jobs with long running times on Nautilus; excluding those, the average walltime on Nautilus would be even lower.

Table 1: Number of cores used in small jobs on Kraken and all jobs on Nautilus

System	Min	Median	Mean	Max
Kraken	12	24	64.47	1008
Nautilus-analysis	8	8	56.81	1016
Nautilus-computation	8	16	35.7	1008

Table 2: Running time, in hours, of small jobs on Kraken and all jobs on Nautilus. Some jobs were permitted to run longer than NICS standard policy, so maximum values are longer than the 24-hour limit on requested walltimes.

System	Min	Median	Mean	Max
Kraken	0	5.41	8.28	27.27
Nautilus-analysis	0	0.37	3.57	40.86
Nautilus-computation	0	0.44	0.67	30.41

In addition to comparing the size of the jobs that run on these two systems, we also compare the quality of service that the users experience. We measure this in two ways, first in terms of how long that jobs wait in the queue before running and secondly in terms of the *scheduler expansion factor*, as described in [5] for describing the productivity and throughput benefits of Trestles at SDSC.

We show time that jobs wait in the queues in Table 4.1. Looking at the time that the jobs wait in the queue, we see that the shortest median wait times are for the analysis jobs on Nautilus. Median wait times for jobs on Kraken were only somewhat longer than median wait times for computation jobs on Nautilus, but the jobs with the longest waits waited much longer on Kraken. While some jobs had to wait roughly two weeks before running on Nautilus, jobs on Kraken had wait times extending to nearly two months. Long wait times were fairly uncommon but were not limited to a few isolated incidents. We expect that most of the jobs with the longest wait times were on "hold" (not progressing up the queue) for a substantial amount of time, either because of job dependencies or other factors aside from machine load.

The wait time metric reflects what we would expect about

Table 3: Number of hours that jobs up to 1024 cores wait in the queue before they run.

System	Min	Median	Mean	Max
Kraken	0.00	1.24	14.90	1630.00
Nautilus-analysis	0.00	0.02	2.75	254.1
Nautilus-computation	0.00	0.97	2.80	346.0

Table 4: The scheduler expansion factor for small jobs on Kraken and all jobs on Nautilus.

System	Median	Mean	St. Dev.
Kraken	1.110	2.348	32.6
Nautilus-analysis	1.416	2.291	11.3
Nautilus-computation	1.004	1.654	18.4

these systems. Researchers expect fast response for their analysis jobs, especially if they are planning on doing interactive analyses, so it is consistent with our expectations that the analysis queue on Nautilus has the fastest typical response times. As computation is a lower priority on Nautilus, we also expect those jobs to wait longer to run. The wide spread of wait times on Kraken reflects two realities of the system. The highest priority during normal operation goes to large core-count jobs in the large queue on Kraken, but smaller jobs serve as backfill while the large jobs wait to run [1]. Jobs with short requested walltimes are most likely to be scheduled as backfill.

We can also look at these waits in terms of the scheduler expansion factor. This metric is defined as

$$\frac{\text{requested walltime} + \text{system queue time}}{\text{requested walltime}}.$$

It measures by what factor the time from job submission to job completion exceeds the requested walltime. As noted in [5], expansion factors tend to be higher for shorter jobs; this metric penalizes long waits for jobs that requested short walltimes. The scheduler expansion factor for the small jobs on Kraken and all jobs on Nautilus are shown in Table 4.1.

One essential factor to keep in mind when examining how long jobs wait before running is the load on the machine. A busy, highly-allocated system will have more scheduling constraints than a less-used system. During the time period that we examine, Kraken was fully allocated and ran at a consistently high level of utilization, with an overall utilization of 91% for the year. Nautilus, on the other hand, was not fully allocated. In some quarters the XSEDE Resource Allocation Committee (XRAC) allocated less than half of the CPU-hours available on Nautilus, and Nautilus had an average utilization of 41% during the year. Usage on Nautilus showed spikes of activity followed by periods of extremely low utilization. During the times that few resources were in use, jobs on Nautilus could start immediately. By comparison, in 2011 Trestles was nearly 70% allocated with utilization at around 80% [5].

These expansion factors show that most jobs have a good quality of service, but there are some outliers. These represent a longer wait time than jobs running on Trestles, which aims to keep the scheduler expansion factor close to 1. However, they are lower than the 2009 TeraGrid-wide weighted expansion factor of 4.96 [5] and the mean expansion factor of 3.424 for all jobs running on Kraken during our timeframe.

We note that the expansion factor on Kraken depends on the size of the job. Mean expansion factor was 2.85 in the small queue, 5.12 in the medium queue, 12.6 in the large queue, and 188 in the capability queue. We compared these means with an ANOVA and with pairwise t-tests. These statistical tests all had p-values were on the order of 10^{-16}, so the difference is means is statistically significant.

4.2 Comparing usage by projects that ran on both systems

The next collection of projects that we analyze are those that ran jobs on both Kraken and Nautilus. This gives us another comparison between the systems. We assume that, in general, researchers are choosing to submit their jobs on the system that is most suited to the work that they are doing. We expect that for researchers using both systems that the jobs submitted on Kraken tend to be traditional computation and that the jobs submitted on Nautilus are either analysis jobs or jobs that take advantage of the larger available memory-per-core. As noted above, the quality-of-service for small jobs on Kraken is quite reasonable, so we have no reason to expect a significant amount of use of Nautilus is by Kraken-users seeking improved throughput. In order to further minimize this possibility, these analyses will further limit the projects studied to those that were granted time on both systems by XSEDE. This restriction excludes locally-allocated projects, including some used for benchmarking, code development, and for computation-only use on Nautilus. Overall, 32 projects ran on both systems; we remove four locally-allocated projects, leaving 28 projects in fields such as astronomy, atmospheric sciences, chemical and thermal systems, and physics.

We examine data from 90,540 jobs adding up to 191,887,568 CPU-hours run by researchers from these 28 projects. 59,330 jobs for 189,591,788 CPU-hours were run on Kraken, and the remaining 31,210 jobs and 2,295,780 CPU-hours were run on Nautilus. We note that this represents 74% of the jobs and 66% of the CPU-hours consumed on Nautilus during this timeframe, and 77% of this usage (measured by CPU-hours) on Nautilus is in the analysis queue. That is, most of the usage of Nautilus is by projects who use both Nautilus and Kraken. This is consistent with Nautilus being used to analyze simulation data created on Kraken.

One key feature of Nautilus is its large memory. While jobs on Kraken are limited to 1.33 GB of memory per core, Nautilus has 4 GB of memory per core. On average, researchers who ran jobs on both systems used 1,786 MB of memory per core on Nautilus. Due to the limitations of the `getrusage()` system call, our data does not report the actual memory used by simulations running on the compute nodes on Kraken. Yet, we note that the typical job on Nautilus used more memory-per-core than is available on Kraken.

When we compare the relative amount of CPU-hours that these projects consumed on Kraken and Nautilus, almost all of these projects expended more hours on Kraken than the did on Nautilus; one extreme outlier project consumed 6,273,000 CPU-hours on Kraken for each CPU-hour that they consumed on Nautilus. Excluding this outlier project, we find that projects use a median of 255 and a mean of 2,966 CPU-hours on Kraken for each hour that they consume on Nautilus.

Looking at walltime, we see that the jobs run by these projects used an average of 4.6 hours of walltime on Kraken and 1.3 hours of walltime on Nautilus. This is consistent with what we found when comparing the small projects on Kraken to all the projects on Nautilus. Thus, we are confident in saying that jobs on Kraken tend to run for a larger number of hours than jobs on Nautilus.

One other difference that we see between these systems is the fraction of jobs that are launched with the `-I` flag for interactive use. We note that this does *not* include jobs, such

as VisIt in client-server mode, that are launched through the standard batch system but allow real-time interaction; these are only the interactive jobs which return a prompt and allow for users to run from the command line. Looking at the users who ran on both systems, 1.5% of the jobs that they ran on Kraken were interactive while 4.9% of the jobs that they ran on Nautilus were interactive.

There were only six projects that fell into both categories studied, having all their usage under 1024 cores and using both machines. Two of these were physics projects that only ran a few jobs on either system, and one is an engineering project that uses both Kraken and Nautilus for simulation. The other projects are in the fields of atmospheric sciences (two projects) and astrophysics (one project).

5. CONCLUSION

In comparing the usage of Kraken and Nautilus at NICS, we find that there are some overall similarities as well as specific differences in how researchers use these systems. Users take advantage of the flexibility of both systems and submit a wide range of jobs reflecting both capacity and capability computing. We also find that usage of Kraken reflects a different mix of job sizes than is observed on Jaguar at ORNL.

The quality of service on Nautilus is somewhat better that for small jobs on Kraken. This is likely due to the fact that Kraken is much more completely allocated and utilized than Nautilus. However, it is unclear whether there is any benefit of moving small core-count computational jobs to Nautilus. Since jobs on Kraken tend to have longer walltimes than the computational jobs on Nautilus, this may adversely effect the quality of service for analysis users on Nautilus.

Despite some macroscopic similarities in overall patterns of usage, we noted some distinct differences between how researchers used Kraken and Nautilus. Jobs run on Nautilus tend to have a shorter duration than jobs with similar core-counts on Kraken, and jobs on Nautilus tend to use more memory-per-core than is available on Kraken. Furthermore, researchers were more likely to run interactive jobs on Nautilus than on Kraken. We conclude that this reflects Nautilus being used as more than just additional computational capacity for users at NICS. Understanding the way that HPC is used for analysis rather than for simulation is an important step in understanding the computing needs of researchers in an era of data-rich science.

We make three more general hypotheses based on our observations. We are not suggesting specific actions for provisioning the next generation of computation and analysis systems; however, we identify some issues to consider when developing future roadmaps.

First, since usage on Kraken is quite different from usage on Jaguar, quantitative information about usage patterns will vary between scientific communities. We conjecture that extrapolating from experiences of other agencies may yield inaccurate predictions of the needs of XSEDE researchers for capability systems. As we find that 10% of the users consume 90% of the CPU-hours on the systems at NICS, the needs of only a small fraction of HPC users would need to be assessed to acquire data about the majority of the usage.

Secondly, the flip side of this 90/10 pattern of usage is that a survey restricted to the projects that consume the most CPU-hours may not reflect the breadth of use of these systems. A survey that is limited in focus to only the biggest users may fail to recognize the broad scientific impact of NSF's cyberinfrastructure. Understanding the research processes of these smaller users is also important in identifying ways in which XSEDE contributes to scientific discovery.

While a resource such as Nautilus can be used for a wide range of data-intensive and data-driven applications, we find that two-thirds of the usage on Nautilus was by users who also use Kraken. Thus, our third hypothesis is that most of the current usage of this system is limited to researchers who are already in the HPC community and who likely generate their data through simulation. While researchers with large datasets from non-computational sources are likely aware of HPC, they may not be familiar with XSEDE's resources or how easily their legacy code could be ported to an HPC analysis system.

6. ACKNOWLEDGMENTS

We thank Troy Baer for his work on the `pbsacct` system that provides the data and for his assistance with interpreting the reports. Nick Lineback wrote scripts to help us get the data in a convenient form.

Data were analyzed and visualized with R[9], including the reldist[3] and ggplot2[11] packages.

This work was supported primarily by the Office of Cyber Infrastructure of the National Science Foundation under NSF Award Number ARRA-NSF-OCI-0906324 for NICS-RDAV center, NSF-OCI-1136246 for support of undergraduate research in the NICS-RDAV center, NSF-OCI-0711134 for initial funding to support Kraken and NSF-OCI-1053575 for funding to support NICS's participation in XSEDE.

7. REFERENCES

[1] P. Andrews, P. Kovatch, V. Hazlewood, and T. Baer. Scheduling a 100,000 core supercomputer for maximum utilization and capability. In *Proceedings of the 2010 39th International Conference on Parallel Processing*, ICPPW'10, pages 421–427. IEEE Computer Society, 2010.

[2] E. Bethel, J. van Rosendale, D. Southard, K. Gaither, H. Childs, E. Brugger, and S. Ahern. Visualization at supercomputing centers: The tale of little big iron and the three skinny guys. *Computer Graphics and Applications, IEEE*, 31(1):90–95, Jan.–Feb. 2011.

[3] M. S. Handcock. *Relative Distribution Methods*. Los Angeles, CA, 2011. Version 1.6. Project home page at `http://www.stat.ucla.edu/~handcock/RelDist`.

[4] D. Hart. Deep and wide metrics for HPC resource capability and project usage. In *State of the Practice Reports*, SC '11, pages 1:1–1:7, New York, NY, USA, 2011. ACM.

[5] R. L. Moore, D. L. Hart, W. Pfeiffer, M. Tatineni, K. Yoshimoto, and W. S. Young. Trestles: a high-productivity HPC system targeted to modest-scale and gateway users. In *Proceedings of the 2011 TeraGrid Conference: Extreme Digital Discovery*, TG '11, pages 25:1–25:7, New York, NY, USA, 2011. ACM.

[6] National Center for Computational Sciences. `http://users.nccs.gov/`, 2012.

[7] Oak Ridge Leadership Computing Facility. Titan project timeline. `http://www.olcf.ornl.gov/titan/`

system-configuration-timeline/.

[8] PBS Tools, National Institute for Computational Sciences and Ohio Supercomputer Center. http://www.nics.tennessee.edu/~troy/pbstools.

[9] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2012. ISBN 3-900051-07-0.

[10] Texas Advanced Computing Center. http://www.tacc.utexas.edu/stampede, 2012.

[11] H. Wickham. *ggplot2: elegant graphics for data analysis*. Springer New York, 2009.

[12] N. Wilkins-Diehr, D. Gannon, G. Klimeck, S. Oster, and S. Pamidighantam. TeraGrid science gateways and their impact on science. *Computer*, 41(11):32–41, Nov. 2008.