Analyzing Throughput and Utilization on Trestles

Richard L. Moore
San Diego Supercomputer Center
U California San Diego, M/C 0505
La Jolla, CA 92093-0505
+1 858-822-5457
rlm@sdsc.edu

Leonard K. Carson
San Diego Supercomputer Center
U California San Diego, M/C 0505
La Jolla, CA 92093-0505
+1 858-822-8311
lcarson@sdsc.edu

Amin Ghadersohi
Ctr for Computational Research
SUNY Buffalo
Buffalo, New York 14203
+1 716-881-8955
ag28@ccr.buffalo.edu

Adam Jundt
San Diego Supercomputer Center
U California San Diego, M/C 0505
La Jolla, CA 92093-0505
+1 858-822-6580
ajundt@sdsc.edu

Kenneth Yoshimoto
San Diego Supercomputer Center
U California San Diego, M/C 0505
La Jolla, CA 92093-0505
+1 858-822-0859
kenneth@sdsc.edu

William S. Young
San Diego Supercomputer Center
U California San Diego, M/C 0505
La Jolla, CA 92093-0505
+1 858-534-5157
wyoung@sdsc.edu

ABSTRACT

The Trestles system is targeted to modest-scale and gateway users, and is designed to enhance users' productivity by maintaining good turnaround time as well as other user-friendly features such as long run times and user reservations. However, the goal of maintaining good throughput competes with the goal of high system utilization. This paper analyzes one year of Trestles operations to characterize the empirical relationship between utilization and throughput, with the objectives of understanding their relationship, and informing allocations and scheduling policies to optimize their tradeoff. There is considerable scatter in the correlation between utilization and throughput, as measured by expansion factor. There are periods of good throughput at both low and high utilizations, while there are other periods when throughput degrades significantly not only at high utilization but even at low utilization. However, throughput consistently degrades above ~90% utilization. User behavior clearly impacts the expansion factor metrics: the great majority of jobs with extreme expansion factors are associated with a very small fraction of users who either (1) flood the queue with many jobs or (2) request job run times far in excess of actual run times. While the former is a user workflow choice, the latter clearly demonstrates the benefit of matching requested time to actual run time. Utilization and throughput metrics derived from XDMoD are compared for Trestles with two other XSEDE systems, Ranger and Kraken, with different sizes and allocation/scheduling policies. Both Ranger and Kraken have generally higher utilization and, not surprisingly, higher expansion factors than Trestles over the analysis period. As a result of this analysis, we intend to increase the target allocation fraction from the current 70% to ~75-80%, and strongly advise users to reasonably match requested run times to actual run times.

XSEDE12, July 16 - 20 2012, Chicago, IL, USA

Categories and Subject Descriptors

K.6.2 [**Management of Computing and Information Systems**]: Installation Management – *performance and usage measurement, pricing and resource allocation.*

General Terms

Management, Measurement, Performance, Design.

Keywords

Allocations, Expansion Factor, Utilization, Scheduling Policies.

INTRODUCTION

Trestles was proposed as a high-performance computing (HPC) system in the repertoire of NSF systems that would be specifically targeted to modest-scale and gateway users and operated to enhance the productivity of those users [1]. A key feature to enhance user productivity is to reduce the long queue waits typical of many XSEDE compute resources. High system utilization also contributes to researcher productivity by expanding the number of users on the system and/or providing more compute time per user. However, there is a trade-off between these two operational metrics, with high utilization typically increasing wait times. In contrast to most large-scale production HPC systems, Trestles is operated with its first priority being quick turnaround for its users, while high utilization is a secondary objective.

This paper analyzes the last twelve months of Trestles operations to characterize the empirical relationship between utilization and throughput and to better understand how to optimize their trade-off. For example, is there a knee of the curve between utilization and throughput, below which good throughput is maintained and above which queue wait times increase rapidly? What characteristics of the scheduling policies or user job submittal workflows decrease or increase throughput? How can allocation and scheduling policies be tuned to maintain good throughput and optimize utilization? Understanding these trade-offs can improve Trestles operational procedures and may also be applicable to other production HPC resources.

ALLOCATIONS AND SCHEDULING POLICIES

Previous analysis of the TeraGrid-wide job workload during 2009 revealed that ~80% of all projects never used more than 512 cores over the course of a year, and that these modest-scale projects accounted for less than 20% of recorded use across all TeraGrid compute resources [1]. This historical workload profile, coupled with the ever-present user demand for fast turnaround of jobs, suggested a strategy for delivering enhanced scientific productivity to a large number of users by deploying a system that would be targeted to these modest-scale (and gateway) users and operated to minimize queue waits. This strategy formed the basis for SDSC's proposal to NSF for the Trestles system, which was awarded and deployed in 2010 and entered production for allocated users in January 2011. Trestles is a 100-Teraflop HPC cluster with 324 nodes connected by an Infiniband QDR communications network. Each node has four 8-core AMD Magny-Cours processors for a total of 32 cores, 64GB DRAM, and 120GB node-local flash memory. In order to target modest-scale users, the largest job size is restricted to 1,024 cores (32 nodes, or ~10% of the system). To ensure that the system serves a large numbers of users, allocations are capped at 1.5M SU/year (<2% of the system); gateways may request a larger allocation since they serve a large number of users. In addition to being responsive to user requests for fast turnaround, Trestles supports another common user request for long-running jobs: standard queues allow 48-hour run times and users can request exceptions up to two weeks.

The primary tool used to achieve the objective of good throughput on Trestles has been to make available only ~70% of the theoretically available cycles to users, rather than the 80+% typically used for many XSEDE resources. Thus, with steady-state allocations and usage, the system utilization would be ~70% averaged over the course of the year. Note that the utilization will vary substantially relative to the target value. Furthermore, because Trestles is a relatively new resource, the early quarters were not fully allocated and consequently utilization was relatively low but it has since increased over time. (This trend facilitates this paper's analysis of utilization and turnaround because there is substantial variability in utilization over the period studied.)

There is not a well-controlled study that demonstrates the relationship between utilization and throughput – the initial 70% target value was an informed estimate to enhance job turnaround because it is lower than the typical 80-90% and the average flow of jobs through the system should be quicker with lower system utilization. Using allocated percentage to control turnaround is a major tuning parameter, but adjustments are only available four times per year in the quarterly allocation cycle, and the timescale for impact of those adjustments is roughly a year since allocations are annual. As demonstrated below, there is substantial variability in utilization and throughput over any given quarter, and correlations are neither immediate nor transparent.

Torque is used as the resource manager, maintaining information on job and node state and executing job start and deletion requests. Instead of the default pbs_sched component, the Catalina Scheduler [8], is used as the scheduler. Catalina Scheduler is a single-queue, reservations-based scheduler, much like Maui Scheduler [9] from Maui High Performance Computing Center. "Single-queue" means that all nodes are considered as a single partition of compute resources, rather than as separate partitions. The rationale for this design is that fragmentation from partitioning of compute resources tends to reduce overall system utilization (unpublished results). Jobs are considered as a single eligible set of work units. This set is sorted into a prioritized list.

To enhance throughput and smooth out fluctuations in job submissions on a finer timescale, scheduling policies are used. As mentioned above, job limits are 1K cores (or 32 nodes), and standard 48-hour run time. In each scheduling iteration, only two jobs per user are considered for inclusion in the schedule. However, there is no limit on the number of jobs per user that eventually go into the running state if space is available on the system. Exceptions to the 48-hour default time limit are made on a case-by-case basis. While long-running jobs make it more difficult to schedule for fast turnaround, it was decided that the enhanced usability for specific users was worth this trade-off.

A dynamic job prioritization heuristic is used to improve the likelihood that jobs will achieve reasonable turnaround time. The scheduler prioritizes jobs with a multiple-element formula. Two elements of the formula are target expansion factor and target queue wait time. For each category of job, the scheduler is configured with a target expansion factor and a target queue wait time. As jobs age in the queue and approach these targets, the corresponding priority elements increase in a nonlinear fashion. While this does not guarantee good turnaround for all jobs, it tends to boost priority for jobs that approach excessive waiting time. The goal of these features is to deliver a small-job-friendly environment in which job expansion factors are low and utilization rates are high.

The priority calculation consists of the following terms:

(time in seconds, resource in nodes)

$$
\begin{aligned}
priority = {} & resource\_number * Resource\_Weight + \\
& local\_admin\_float * Local\_Admin\_Weight + \\
& local\_user\_float * Local\_User\_Weight + \\
& expansion\_factor * Expansion\_Factor\_Weight + \\
& queue\_wait\_time * System\_Queue\_Time\_Weight + \\
& submit\_wait\_time * Submit\_Time\_Weight + \\
& wall\_clock\_time * Wall\_Time\_Weight + \\
& QOS\_priority * QOS\_Priority\_Weight + \\
& QOS\_target\_xf\_value * \\
& \quad QOS\_Target\_Expansion\_Factor\_Weight + \\
& QOS\_target\_qwt\_value * \\
& \quad QOS\_Target\_Queue\_Wait\_Time\_Weight
\end{aligned}
$$

resource_number	number of nodes requested
local_admin_float	administrator-set value for each job
local_user_float	user-set negative value for each job
expansion_factor	(requested walltime + queue wait time)/requested walltime
queue_wait_time	seconds that job has been considered eligible by policy
submit_wait_time	seconds since job was submitted
wall_clock_time	requested duration of job

QOS_priority	base priority value for job of that Quality of Service
QOS_target_xf_value	expansion factor derived urgency term, based on job's QOS
QOS_target_xf_value	queue wait time derived urgency term, based on job's QOS

Note that varying *_Weights can influence the impact of different job characteristics. For example if Wall_Time_Weight is set to 0, then requested wallclock time will have no contribution to priority. Jobs that approach a target expansion factor may displace higher priority jobs, as they approach that target. The specific weightings and targets are tuned in response to changes in workload.

Trestles is currently the only XSEDE HPC resource that allows users to set their own reservations to ensure access at specific times, or to have pre-emptive on-demand access for applications that are not predictable in advance and which have societal impact [2]. These features address specific requirements from users, but to date are not frequently used and do not have a significant impact on the usage analyses below.

METRICS AND DEFINITIONS

In the course of this analysis, it is apparent that defining effective throughput metrics is a challenge. This section defines the terms used in the analysis and describes some of the challenges.

First, system utilization is the ratio of total core-hours consumed by all jobs in a given period over the system's capacity measured in core-hours for the same period. This is a straightforward calculation available from system logs, and is used consistently in this analysis. A limitation of this metric is that it does not differentiate unused time due to system outages from scheduler inefficiencies. For example, system outages due to hardware or maintenance drop both utilization and throughput. To isolate just scheduler performance requires exclusion of any impacts of system outages. The counter-argument for including all time periods is that it better reflects the user experience of the queues.

It is much more difficult to define a metric that effectively characterizes throughput. Job wait time is clearly an important measure, but generally we adopt the user expansion factor rather than wait time. A job's "user expansion factor" is defined as

$$EXPF_U = \frac{RunTime + TimeInQueue}{RunTime}$$

Equation 1

A user expansion factor of 1 represents no job wait. This metric probably corresponds better to a user's turnaround expectations than wait time, because the wait time is normalized by run time. One issue, particularly for a system with many long-running jobs like Trestles, is that each job must be put into a time slot; in this analysis jobs are assigned to the period when they started running, rather than when it was submitted to the queue or completed.

A related metric, which substitutes requested time for run time, is a job's "request time expansion factor" or

$$EXPF_{RT} = \frac{RequestedTime + TimeInQueue}{RequestedTime}$$

Equation 2

This metric better reflects the information available to a scheduler. Since run time is never longer than requested time, the request time expansion factors are always equal to or less than user expansion factors.

Other throughput metrics are discussed in [3].

Another nuance in calculating utilization and throughput metrics is the period over which to calculate the metrics. There are valid arguments for both fine and coarse resolution. While some long-period (e.g., annual) metrics are used, daily windows are more typical in this analysis.

Whether throughput is measured by wait time or expansion factor, these metrics represent broad non-Gaussian distributions. Averages, especially for expansion factors, can be affected by extreme outliers that represent a small fraction of the total workload, and, as will be shown below, often reveal anomalous user behavior. Therefore the median is often a better characterization of the distribution than the average value. Since we wish to also examine behavior at the extremes, we have chosen to use a 95% threshold value, i.e., the expansion factor value at which 95% of jobs have a lower expansion factor. This metric reflects throughput for the great majority of jobs but excludes the extreme tail of the distribution. Note that the 95% threshold represents a high standard compared to the median (50%) value, and its values are often much higher than median expansion factors.

While expansion factors are generally a better metric than wait times to measure user expectations, short jobs often have statistically higher expansion factors and can skew the results. One can argue that waiting ten minutes for a 1 minute job to complete does not have the adverse productivity impact of waiting 10 hours for a 1 hour job to complete. Furthermore, many short jobs reflect job failures which dramatically skew the metrics. Therefore, it is reasonable to adopt a threshold to exclude very short jobs; we have excluded all jobs with <1 minute run times. Even at this level, the statistics are often skewed by shorter jobs.

This study focuses primarily on Trestles results, with the job statistics collected accurately from system logs. However we also compare Trestles results with those from other XSEDE systems, specifically Ranger and Kraken. We have used the XDMoD tool operating on the XSEDE Central Database (XSCDB) which contains all job accounting data reported by XSEDE Service Providers for jobs from XSEDE-allocated users. While XDMoD and XSCDB represent a powerful analysis tool, these results must be qualified because the metrics are based solely on jobs reported to the XSCDB and do not necessarily include all jobs on the systems. In particular, utilization calculated by XDMoD is a lower limit to actual utilization, since there may be local jobs not submitted to XSCDB.

ANALYSIS OF TRESTLES OPERATIONS-UTILIZATION AND TURNAROUND

Trestles became available as a production resource for XSEDE users starting in January 2011. As is typical of most new systems, the first quarter's allocated usage was low and dominated by just a few users, and there were a fair number of system downtimes for planned outages; therefore this analysis excludes the first quarter of operations. During the one-year period from April 2011 to March 2012, utilization and user expansion factors averaged 60% and 1.3 respectively, with considerable variability during the period. More recently, system usage reflects four full allocation cycles, and the utilization has increased significantly, often now exceeding 80% and with increased expansion factors.

Figure 1 shows daily utilization and 95% threshold user expansion factors as a function of time for this period. Utilization varied widely during most of 2011 but in recent months has climbed well above the 70% target value and is now often above 80% or even 90%. Note that the period of very low utilization in May 2011 was caused by a planned parallel file system upgrade while the drop in September 2011 was caused by a county-wide power failure. Because of the large dynamic range, the 95% user expansion factor is plotted on a log scale. While the 95% user expansion factors are generally in single digits, the median value unfortunately often exceeds ten. While there are some clear causal relationships apparent (e.g., the expansion factor often spikes during system outages), Figure 1 does not demonstrate an obvious correlation between utilization and users expansion factors.

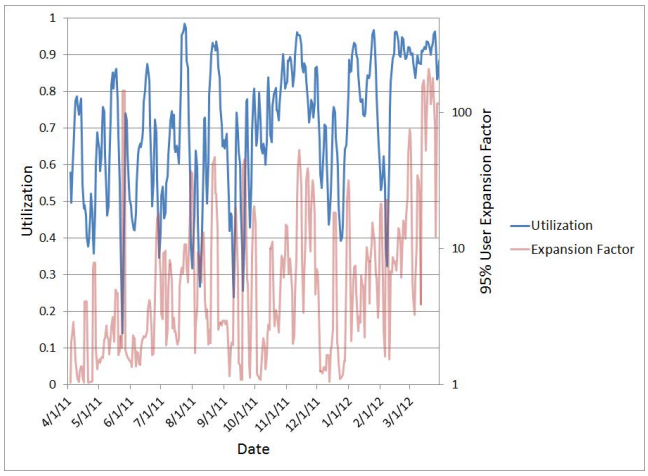

Figure 1 - Trestles Daily Utilization and 95% Threshold User Expansion Factors for Period 4/2011 to 3/2012

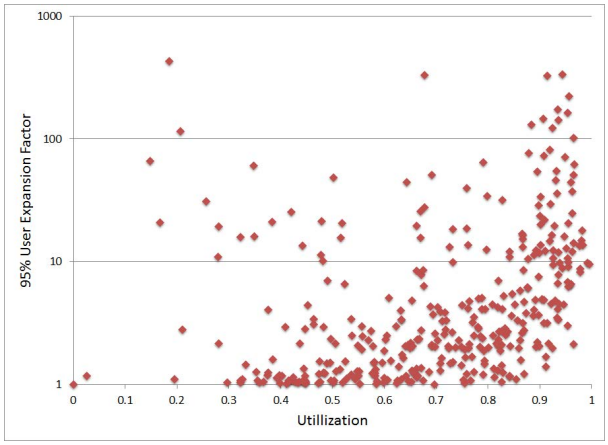

Figure 2 – Scatter Plot of Daily Utilization and 95% Threshold User Expansion Factors for Period 4/2011 to 3/2012

The same daily metrics are shown as a scatter plot in Figure 2, which best illustrates the significant scatter in the correlation between utilization and user expansion factor. Even at quite low utilization, there are frequent examples of anomalously high expansion factors. Also, there are many days with utilization in the range 70-90% that still have excellent turnaround. It is only when utilization exceeds ~90% that one can see a consistent degradation of turnaround. Based on this figure, the knee of the curve may be as high as 85-90% utilization, but the curve is not well defined and has significant scatter.

To further characterize the relationship between utilization and expansion factors, we determined the number of days that Trestles ran within specific utilization ranges, and then the number of those days with excellent turnaround, as measured by days when 95% of all jobs had user expansion factors <1.5 or <2.0. These values, summarized in Table 1, are indeed high bars for defining excellent throughput – all but 5% of the jobs during those days waited less than 50% or 100% respectively of their run time.

Utilization Range	0-40%	40-50%	50-60%	60-70%	70-80%	80-90%	90-100%
Days	32	39	37	56	58	68	74
Days w/ 95% $EXPF_U < 1.5$	17	22	18	18	9	8	1
% days 95% $EXPF_U < 1.5$	53%	56%	49%	32%	16%	12%	1%
Days w/ 95% $EXPF_U < 2.0$	18	25	23	24	17	12	2
% days 95% $EXPF_U < 2.0$	56%	64%	62%	43%	29%	18%	3%

Table 1 – Number of Days With Various Utilization Ranges, and the Number of Those Days with Excellent User Expansion Factors (95% of jobs <1.5 or <2.0 respectively)

Table 1 shows that when utilization was <60%, Trestles had 95% expansion factors less than 1.5 approximately 50% of the time, and 95% factors less than 2.0 about 60% of the time. Both of these values steadily decline at higher utilizations, finally to the point where almost no days with >90% utilization had excellent turnaround. The statistics are limited and, as shown below, can be influenced by anomalous user behavior. But the trends are consistent and excellent expansion factors above 90% utilization, while possible, are quite unlikely.

At the same time, the results are a reminder that low utilization does not guarantee fast turnaround and conversely that high utilization does not preclude good turnaround. For example, ~45% of days at utilization <40% did *not* have excellent turnaround, and ~20% of the days with utilization in the 80%'s still had excellent turnaround.

It is important to examine the cause of outlier jobs with very large expansion factors. If outliers are defined as jobs with user expansion factors > 100, ~90% of all Trestles' outlier jobs over the year were submitted by just five of its ~200 users. One user flooded the queue by submitting over 5,000 multicore jobs over the course of several weeks thus creating a huge backlog. The other four users submitted over 200 jobs each with a default wallclock time of 48 hours but nearly all their jobs ran in 1-2 minutes. These four users' jobs unnecessarily sat in the queue waiting for 48-hour blocks of time when they could have been scheduled and quickly completed with more accurate time requests; between the time mismatch and queue flooding, the expansion factors for those users' jobs are extremely high.

To quantify the impact of providing a bad estimate for wallclock times, we plot in Figure 3 the 95% user expansion factor based upon run time (Equation 1) and the expansion factor based upon requested time (Equation 2). The use of requested time in calculating expansion factors reflects information available to the scheduler and allows us to distinguish basic issues with job

turnaround and scheduling from more isolated user-initiated outliers. The dozen or so peaks in the run time expansion factor are attributable to the five users who had the major mismatch of requested and actual run times and/or flooded the queue. The much lower requested time expansion factors suggest that these outliers are major contributors to high expansion factors in our run time metrics. This is not to say that the actual performance of the system is worse because of inaccuracies in user estimates of wallclock time. It has been shown that some workloads with certain distributions of run time overestimation can actually benefit from those overestimates [7]; Since short jobs contribute disproportionately highly to the overall expansion factor, overestimates of running time tend to delay long jobs and create backfilling space in which to run shorter jobs.

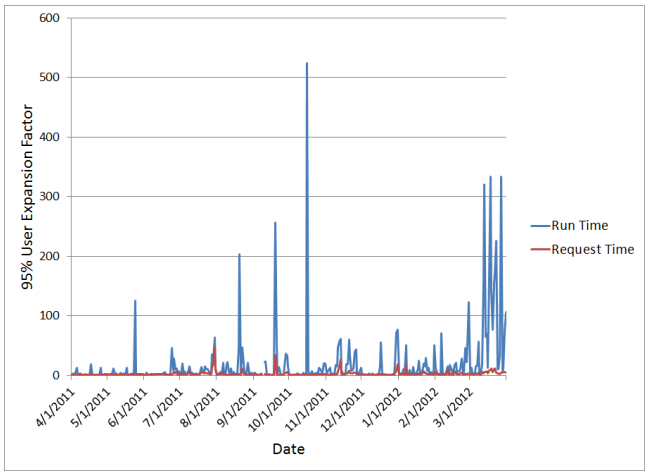

Figure 3 – Run Time and Request Time 95% Expansion Factors

It is possible to further tune scheduling policies in order to favor expansion factor over utilization. Two examples of this are the limit on queued jobs per user and the priority elements boosting jobs based on age-in-queue metrics. While further analysis and simulation would be required to pin down a causal relationship, we do see that expansion factor is quite good on Trestles, while utilization on many days is low. This demonstrates that scheduler tuning can be effective in achieving desired throughput performance for a large fraction of daily production workload.

However, we observe cases where, despite the current scheduler tuning, expansion factor is likely to suffer. This was often observed in situations where the current scheduler tuning allowed utilization to exceed 90%. The adverse impact of allowing veru high utilization can be seen in the count of days with low expansion factors. The current scheduler tuning has resulted in worsening expansion factor for that extreme level of utilization.

SDSC intends to keep overall allocation commitments in the 70-80% range and will tune the scheduler to manage occasional days with very high utilization to minimize adverse effects on expansion factors. Additional "knobs", such as a policy limit on amount of node-seconds of running jobs per user or per account, could be used to more strongly favor expansion factor over utilization.

COMPARISONS TO OTHER XSEDE SYSTEMS

It is interesting to compare the Trestles analysis to other XSEDE resources, using the XDMoD tool operating on the XSEDE Central Database (XSCDB). As described earlier, there are known caveats with this data source, particularly that system utilization may be underestimated. While the Trestles data in previous sections are derived from internal system logs, all data in this section, including Trestles, are drawn from XSCDB data.

This section compares XDMoD metrics from Trestles with TACC's Ranger [4] and NICS' Kraken [5]. The systems are ~6x and ~11x larger than Trestles, and represent a diversity of allocations/scheduling policies. In contrast to Trestles, which is intentionally targeted to modest-scale capacity computing (e.g. maximum of 1K cores, ~10% of the system) and quick turnaround, Kraken is operated without job core count restrictions and with an emphasis on large-scale capability computing and high utilization [6]. Ranger default scheduler policies restrict usage to 4K cores (~6% of the system), with access to a 4K-16K core queue allowed after users demonstrate applications scaling, and larger-scale jobs (>25% of the system) more restricted and generally allowed only after system maintenance when the system is drained. TACC's Lonestar system was excluded from this comparison because only a fraction of the system is made available to XSEDE users and therefore XDMoD/XSCDB metrics are not representative of its capacity.

Table 2 summarizes key metrics, derived via XDMoD/XSCDB, for these three systems over the one-year period from April 2011 through March 2012. All metrics are averaged over a year, so this comparison is at a macroscopic level.

Reported system utilization varies significantly across the systems. The Trestles utilization value is relatively low during this period because the system was new in the allocation process and the system was intentionally targeted to a conservative 70% utilization to facilitate good throughput.

(Period Apr 2011-March 2012)	Trestles	Ranger	Kraken
Number of cores	10,368	62,976	112,896
System utilization	61.7%	76.6%	86.7%
Median expansion factor	1.2	1.3	1.5
95% threshold user expansion factor	15.4	24.5	45.1
Average wait time	3.5 hrs	7.8 hrs	10.8 hrs
Average run time	13.0 hrs	8.1 hrs	5.1 hrs
Median cores/job	4	32	72
Average cores/job	25	197	514
Average cores/job, as a fraction of system size	0.24%	0.31%	0.46%

SU-weighted average cores/job	202	1,223	10,875
SU-weighted average cores/job, as a fraction of system size	1.9%	1.9%	9.6%

Table 2 - Comparative Metrics for XSEDE Resources with a Range of Allocation/Scheduling Policies

The median expansion factors, the 95% expansion factors, and the average wait times are correlated with the utilization values across the three systems, with all measures of throughput degrading with higher utilization. This correlation is consistent with expectations, although simple metrics like averages, especially over a full year period, can mask many complexities. One should not conclude that these three points are representative of a standard function between utilization and wait times/expansion factors, but only that they are consistent with trends.

A key issue to consider is the extent to which the job mix, including job size and run time, impact utilization and turnaround time. For example, full-machine jobs are notorious for requiring that the system drain in advance of the job and forcing all jobs that cannot be backfilled to wait for the duration of the full-machine job. Special attention in a scheduler can minimize adverse impacts on other jobs and in fact planned, sequential full-machine jobs can produce very high utilization [6]. A specific hypothesis is whether the Trestles policy of targeting modest-scale jobs (maximum 1024 cores, or at most ~10% of the system) benefits throughput compared to systems that allow a broader range of job sizes, particularly much larger jobs. For example, can a scheduler better slot a large number of small jobs into efficient use of the system and better throughput?

While Trestles does not allow any job to use more than 10% of the system cores, it does have a relatively generous policy on run time, with a default limit of 48 hours and up to 2 weeks frequently granted on request. Therefore it tends to have a long average run time per job (13 hrs) compared to Ranger (8 hrs) or Kraken (5 hrs). On the other hand, the median and average cores/job vary widely, from 4(median)/25(average) for Trestles to 32/197 for Ranger and 72/514 for Kraken. However, when normalized by the system size, the average job size is virtually identical for Trestles and Ranger and Kraken's average is only ~1.5X this level. So in the temporal domain, Trestles jobs are typically longer while in the processor domain, they are comparable to Ranger as a fraction of the system and only slightly smaller than Kraken. So based on these macroscopic statistics, the hypotheis that Trestle's small jobs (as a fraction of system size) could result in more efficient scheduling and throughput, cannot be tested here as there is not significant differentiation for these metrics across the three systems. If anything, the dependence on utilization seems more predominant in the impact on throughput than details of the job mix. It will be interesting to see, as Trestles utilization persists in the range 70-90%, whether we can maintain good throughput relative to other XSEDE systems.

What is perhaps the most surprising result from this XDMoD analysis is the low values for the median/average job size as a fraction of the machine. This is consistent with earlier analysis that showed most jobs across TeraGrid are modest-scale, even though large-scale jobs consume a disproportionate share of overall resources [1]; that analysis provided the initial rationale for proposing Trestles as a system for modest-scale users. It is clear that averaging by jobs gives equal weight to all jobs and ignores the incredible dynamic range (e.g. ~10^8) in SUs consumed by various jobs. Therefore we have calculated the SU-weighted average job size for the three systems, measured in cores and fraction of the system. This weighting increases the average cores/job by a factor of eight for Trestles, six for Ranger and 21 for Kraken, with the SU-weighted average job size being ~2% of the system size for Trestles and Ranger, and ~10% of the system size for Kraken. The absolute level and relative increase reflect the emphasis on capability computing for Kraken.

SUMMARY

This paper presents an analysis of the last twelve months of Trestles operations to characterize, understand and optimize the relationship between system utilization and queue wait times. As a new system gaining adoption, Trestles has had a wide range of utilization during the analysis period, providing a valuable set of operational data for this analysis. It is a challenge to define simple metrics to characterize queue waits for users and the complexity of various metrics are discussed. In general, there is substantial scatter in the correlation between utilization and expansion factors, with many examples of high expansion factors at low utilization and low expansion factors at high utilization. It is only at the highest utilization levels of greater than 90% that there is a clear indication of consistently degraged throughput. Job characteristics and individual user workflows substantially impact throughput metrics. For example, short-duration jobs typically have higher expansion factors and can severely skew averages. Almost all outlier expansion factors can be attributed to user behavior such as queue flooding or a gross mismatch between requested time and actual run time. There is no reason to discourage a user from using a queue flooding workflow, as long as the scheduler limits impact on other users (which Trestles does). However users who request significantly more time than actually used by their jobs often penalize themselves with longer wait times, and we now know to look for those situations amongst the outliers and advise users how to improve their throughput.

Some comparisons of job workload, utilization and throughput are made with other XSEDE HPC resources (Kraken and Ranger). Both Kraken and Ranger have higher average utilization than Trestles over the analysis period and, not surprisingly, longer wait times and higher expansion factors than Trestles. A hypothesis that Trestles' small average job size results in more efficient scheduling cannot be tested by this comparison because the average job size, as a function of total system size, is comparable across the three systems.

As a result of this analysis, we plan to slightly increase allocations on Trestles from a conservative 70% of theoretical maximum to ~75-80%, while continuing to monitor the queues and tune scheduling policies. We also will strongly encourage users to match requested times to expected run times (with reasonable buffers) to facilitate their throughput.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. OCI-0503944 at UCSD and under Grant No. OCI-1025159 at U Buffalo. We are especially grateful to the XDMod team at SUNY Buffalo for their contributions in the area of XSEDE metrics and to this paper.

REFERENCES

[1] Moore, R. L., Hart, D. L., Pfeiffer, W., Tatineni, M., Yoshimoto, K. Young, W. S.; "Trestles: A High-Productivity HPC System Targeted to Modest-Scale and Gateway Users," TeraGrid'11, July 2011, Salt Lake City, Utah, USA. ACM 978-1-4503-0888-5/11/07.

[2] Yoshimoto, K.K., Choi, D.J. , Moore, R.L., Majumdar, A., Hocks, E.; "Implementations of Urgent Computing on Production HPC Systems," April 2012, International Conference on Computational Science, ICCS 2012. (Accepted for publication May 2012.)

[3] Ernemann, C.; Hamscher, V.; Yahyapour, R.; , "Benefits of global grid computing for job scheduling," Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on , vol., no., pp. 374- 379, 8 Nov. 2004

doi: 10.1109/GRID.2004.13

URL:http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1382854&isnumber=30134

[4] Description of TACC Ranger system, http://www.tacc.utexas.edu/resources/hpc

[5] Description of NICS Kraken system http://www.nics.tennessee.edu/computing-resources/kraken

[6] Samuel, T.K.; Baer, T.; Brook, R.G.; Ezell, M.; Kovatch, P.; , "Scheduling diverse high performance computing systems with the goal of maximizing utilization," High Performance Computing (HiPC), 2011 18th International Conference on , vol., no., pp.1-6, 18-21 Dec. 2011

doi: 10.1109/HiPC.2011.6152723

URL:http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6152723&isnumber=6152423

[7] Keleher, P. J., Zotkin, D., Perkovic, D., Attacking the bottlenecks of backfilling schedulers, Cluster Computing, December 2000, 3:4, p. 245-254

[8] http://www.sdsc.edu/catalina

[9] http://www.clusterresources.com/product/maui