Using XDMoD to Optimize HPC System Utilization and Performance

Thomas R. Furlani[1], Abani K. Patra[2], James C. Browne[3], Matthew D. Jones[1], Robert L. DeLeon[1], William L. Barth[4], Barry I. Schneider[5], Steven M. Gallo[1], Amin Ghadersohi[1], Ryan J. Gentner[1], Jeffrey T. Palmer[1], Nikolay Simakov[1], Martins Innus[1], Andrew E. Bruno[1], Joseph P. White[1], Cynthia D. Cornelius[1], Thomas Yearke[1], Kyle Marcus[1], Gregor von Laszewsk[6], Fugang Wang[6]

[1]Center for Computational Research, University at Buffalo, State University of New York, 701 Ellicott Street, Buffalo, NY 14203, [2]Mechanical and Aerospace Engineering Department, University at Buffalo, State University of New York, Amherst, NY 14260, [3]Department of Computer Science, University of Texas, Austin, TX 78758, [4]Texas Advanced Computing Center, University of Texas, Austin, TX 78758, [5]CISE Directorate, National Science Foundation, Arlington, VA 22230, [6]Pervasive Technology Institute, Indiana University, 2719 East 10th Street, Bloomington, IN 47408

ABSTRACT

The high cost (capital, power and manpower) of HPC resources requires us to optimize its usage – an outcome that is only possible if adequate data is collected and is used to drive systems management with good analytic capability at different granularities – job, application, user and system. This paper presents a method for the comprehensive management of HPC systems, including utilization, performance, and quality of service. It is based on two open source packages, XDMoD and TACC_Stats, which have been fully integrated. Several case studies are presented that demonstrate its utility for uncovering under-performing hardware and software, both at the system level and at the end-user level (application codes). Examples include the discovery of a software bug in the I/O stack of a commercial parallel file system, which was subsequently fixed by the vendor in the form of a software patch that is now part of their standard release. This error, which resulted in dramatically increased execution times as well as outright job failure of a popular quantum chemistry software package, would likely have gone unnoticed and was only uncovered as a result of implementation of this management system.

Categories and Subject Descriptors

D.3.3 [**Programming Languages**]: [**Performance of Systems**]: Reliability, availability, and serviceability; C.5.1 [**Computer System Implementation**]: Large and Medium ("Mainframe") Computers---Super (very large) computers; K.6.1 [**Management of Computing and Information Systems**]: Project and People Management---Strategic information systems planning; K.6.4 [**Management of Computing and Information Systems**]: System Management---Quality Assurance.

General Terms

Management, Measurement, Performance, Reliability, Standardization, Verification.

Keywords

XDMoD, TACC_Stats, SUPReMM, HPC resource management, Technology Audit Service, HPC Metrics, Application Kernels, XSEDE

1. INTRODUCTION

High-performance computing (HPC) systems are a complex combination of software, processors, memory, networks, and storage systems each of which is characterized by frequent disruptive technological advances. In this environment, system managers, users, and sponsors find it difficult if not impossible to know if optimal performance of the infrastructure is being realized, or even if all subcomponents are functioning properly. HPC centers and their users, at least those HPC centers with open source software stacks, have been to some extent "flying blind", without complete information on system behavior. Anomalous behavior has to be manually diagnosed and remedied with incomplete and sparse data. In addition, many if not most HPC centers lack knowledge of the distribution and performance of the applications running on their systems and therefore have no way of knowing if the applications are well tuned to the architecture and making efficient use of the resource. Indeed, almost assuredly many are not.

While individual tools to measure the utilization, performance, and quality of service of HPC systems have been developed over the years [1-13], a comprehensive HPC system management tool that includes all three of these important measures of the efficacy and operational efficiency of HPC systems had been lacking. The National Science Foundation (NSF) recognized the value of this capability and through the Technology Audit Service (TAS) of XSEDE [14,15] and SUPReMM projects [16,17] made a significant investment in developing tools and infrastructure to make this capability easily accessible to a broad range of users and resource managers.

In this context, XDMoD (XD Metrics on Demand), coupled with SUPReMM, provides a comprehensive resource management tool for HPC systems [17]. It is designed to meet the following objectives: (1) provide the user community with a tool to more effectively and efficiently use their allocations and optimize their use of HPC resources, (2) provide operational staff with the ability to monitor, diagnose, and tune system performance as well as measure the performance of all applications running on their system, (3) provide software developers with the ability to easily obtain detailed analysis of application performance to aid in optimizing code performance, (4) provide management with a diagnostic tool to facilitate HPC planning and analysis, and (5) provide metrics to help measure scientific impact. In addition, analyses of the operational characteristics of the HPC environment can be carried out at different levels of granularity, including job, user, or on a system-wide basis.

This paper is organized as follows. Section 2 provides an overview of XDMoD including a description of the open source version and the incorporation of SUPReMM data (via TACC_Stats) into XDMoD. Section 3 presents the results of several case studies that demonstrate XDMoD's utility for comprehensive HPC system management. The first case study demonstrates its capacity for facilitating HPC system performance assessment through the implementation of application kernels that measure resource performance and provide quality of service metrics. The second case study presents the analysis of the applications running on Stampede, the most recent addition to the XSEDE portfolio and Rush, the HPC resource at the University at Buffalo (UB), State University of New York's Center for Computational Research (CCR). The third and final case study shows, how care must be exercised in the interpretation of data generated by the XDMoD tool, as is the case for most analysis tools. The final Section covers conclusions and future work. We begin with an overview of XDMoD to provide a context for the discussions that follow.

2. XDMoD OVERVIEW

Here we present a brief overview of XDMoD, a more detailed description can be found in Reference [14]. The XDMoD portal [18] provides a rich set of features accessible through an intuitive graphical interface, which is tailored to the role of the user. Metrics provided by XDMoD include: number of jobs, CPUs consumed, wait time, and wall time, with minimum, maximum and the average of these metrics, in addition to many others. These metrics can be broken down by: field of science, institution, job size, job wall time, NSF directorate, NSF user status, parent science, person, principal investigator, and by resource. Performance and quality of service metrics of the HPC infrastructure are also provided, along with application code specific performance metrics (flops, IO rates, network metrics, etc) for all applications running on a given resource (through TACC_Stats).

A context-sensitive drill-down capability is available for many charts allowing users to access additional related information simply by clicking inside a plot and then selecting the desired metric to explore. Another key feature is the Usage Explorer that allows the user to make a custom plot of any metric or combination of metrics filtered or aggregated as desired. For example, Figure 1, which was created using the Usage Explorer, shows CPU hours and wait time versus job size on the CCR resources.

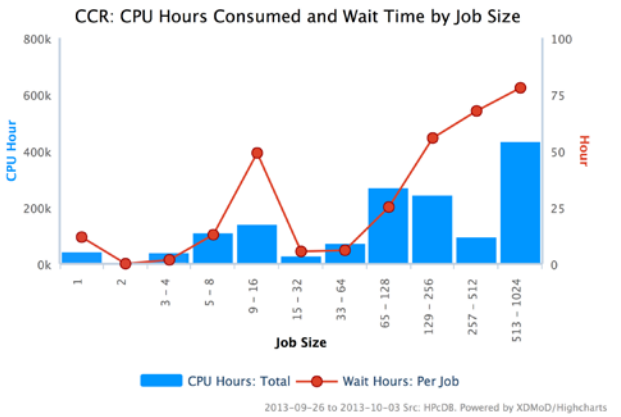

Figure 1. Plot of CPU hours consumed and job wait time versus job size for a one-week period at the University at Buffalo Center for Computational Research.

The XDMoD tool is also designed to preemptively identify underperforming hardware and software by deploying customized, computationally lightweight "application kernels" that continuously monitor HPC system performance and reliability from the application users' point of view [14]. The term "application kernel" is used in this case to represent micro and standard benchmarks that represent key performance features of modern scientific and engineering applications, and small but representative calculations carried out with popular open-source high performance scientific and engineering software packages. The term "computationally-lightweight" is used to indicate that the application kernel runs for a short period (typically less than 10 min) on a small number of processors (less that 128 cores) and therefore requires relatively modest resources for a given run frequency (say once or twice per week). Accordingly, through XDMoD, system managers have the ability to proactively monitor system performance as opposed to having to rely on users to report failures or underperforming hardware and software. The detection of anomalous application kernel performance is being automated through the implementation of process control techniques. In addition, through this framework, new users can determine which of the available systems are best suited to address their computational needs.

In addition, metrics that focus on scientific impact, such as publications, citations and external funding, are now being developed and incorporated into XDMoD to help quantify the role advanced cyberinfrastructure plays in advancing research.

2.1 Open_XDMoD

While the XDMoD framework was initially designed to meet the needs of XSEDE, much of its functionality is applicable to HPC centers in general. In fact, a typical HPC center can be treated conceptually as XSEDE with a single resource provider. Both XSEDE and HPC centers employ the concept of users, compute jobs, project groups, help desk tickets, scientific publications, and one or more HPC resources (e.g., compute cluster, storage pool, etc.). Users also fall into a similar hierarchical organizational structure differing only in the terms used to describe nodes in the hierarchy. On XSEDE, users are associated with one or more projects or allocations, each with their own principal investigator. An individual project is associated with an NSF directorate and field of science; an academic HPC center may group users by decanal unit and department; an industrial HPC center may group users by project and department. Similarly, the job level

information collected can include a job name, wall clock time, start and end dates, memory utilization, and so on.

Taking advantage of the great similarity of XSEDE and a typical HPC center, we have developed an initial release of Open_XDMoD, the open source version of XDMoD. Open_XDMoD leverages the same code base as the version developed for XSEDE. In fact, development of the open source version has led to the refactoring of several parts of XDMoD in order to make it more flexible, easier to maintain, and simpler to add new functionality in the future. As XDMoD continues to be developed, Open_XDMoD will also benefit from this work.

Both versions share many of the same metrics and functionality (e.g., Summary, Usage/Usage Explorer, and Report Generator) and differ mainly in support of elements specific to XSEDE. XSEDE maintains a centralized infrastructure (XSEDE central database) for storing job accounting records, users, and allocations/projects while a typical HPC center may not have this data in a centralized location. Open_XDMoD provides its own data warehouse with support for parsing and loading of resource manager log files and spreadsheets containing user information including departmental affiliations. In addition, future versions will support allocations, and integration with local LDAP services, and application kernels for monitoring Quality of Service. Not all HPC centers will collect the same set of information and therefore we cannot provide the exact same set of metrics for all centers. Open_XDMoD will be able to inspect the data available in its warehouse and disable the display of metrics that it is not able to reliably generate. For example, if a center does not choose to provide a departmental or project hierarchy for users then metrics including these will not be displayed.

2.2 Addition of SUPReMM Data

The initial focus of XDMoD has primarily been on the development of usage, system performance (quality of service), and scientific impact metrics. However, the ability to provide detailed information on the performance *for all applications running on these systems* is necessary for effective resource management. To achieve this capability, the SUPReMM (Integrated HPC **S**ystems **U**sage and **P**erformance of **Re**sources **M**onitoring and **M**odeling) program [16,17] integrates TACC_Stats data into XDMoD. TACC_Stats [19,20] periodically samples node and system state values and a wide spectrum of hardware counters to provide a rich (and extensive) dataset of performance data for every application run on each node in a given HPC cluster for analysis and reporting.

Currently TACC_Stats can gather core-level CPU usage (user time, system time, idle, *etc.*), socket-level memory usage (free, used, cached, *etc.*), swapping/paging activities, system load and process statistics, network and block device counters, interprocess communications (SysV IPC), software/hardware interrupt request (IRQ) count, filesystems usage (NFS, Lustre, Panasas), interconnect fabric traffic (InfiniBand and Myrinet), and CPU hardware performance counters. For a complete list of the data acquired by TACC_Stats, see the TACC_Stats web site [20]. TACC_Stats executes at the beginning of a job, periodically during the job (currently every ten minutes) and at the end of the job. At this frequency of execution, it generates an overhead of approximately 0.1% [16].

3. XDMoD CASE STUDIES

In this section we present the results of several case studies that demonstrate XDMoD's utility for comprehensive HPC system management. The first case study looks at the ability of application kernels to measure quality of service and help identify underperforming hardware and system software. The second case presents results of an analysis of the applications running on TACC's Stampede and UB's HPC system Rush. The final case study demonstrates how care must be exercised in the interpretation of data generated by the XDMoD tool, as is the case for most analysis tools.

3.1 Optimizing HPC System Operation

Most modern multipurpose HPC centers mainly rely upon system related diagnostics, such as network bandwidth utilized, processing loads, number of jobs run, and local usage statistics in order to characterize their workloads and audit infrastructure performance. However, this is quite different from having the means to determine how well the computing infrastructure is operating with respect to the actual scientific and engineering applications for which these HPC platforms are designed and operated. Some of this is discernible by running benchmarks; however in practice benchmarks are so intrusive that they are not run very often (see, for example, Reference [21] in which the application performance suite is run on a quarterly basis), and in many cases only when the HPC platform is initially deployed. In addition benchmarks are typically run by a systems administrator on an idle system under preferred conditions and not as a user in a normal production operation scenario and therefore do not necessarily reflect the performance that a user will experience. Modern HPC infrastructure is a complex combination of hardware and software environments that is continuously evolving, so it is difficult at any one time to know if optimal performance of the infrastructure is being realized. Indeed, as the examples below illustrate, it is more likely than not that performance is less than optimal, resulting in diminished productivity (CPU cycles, failed jobs) on systems that are typically over subscribed. Accordingly, the key to a successful and robust science and engineering-based HPC technology audit capability lies in the development of a diverse set of computationally lightweight application kernels that will run continuously on HPC resources to monitor and measure system performance, including critical components such as the global filesystem performance, local processor and memory performance, and network latency and bandwidth. The application kernels are designed to address this deficiency, and to do so from the perspective of the end-user applications.

We use the term "Kernel" in this case to represent micro and standard benchmarks that represent key performance features of modern scientific and engineering applications, as well as small but representative calculations done with popular open-source high-performance scientific and engineering software packages. Details can be found in References [14,15]. We have distilled lightweight benchmarking kernels from widely used open source scientific applications that are designed to run quickly with an initially targeted wall-clock time of less than 10 minutes. However we also anticipate a need for more demanding kernels in order to stress larger computing systems subject to the needs of HPC resource providers to conduct more extensive testing. While a single application kernel will not simultaneously test all of these aspects of machine performance, the full suite of kernels will stress all of the important performance-limiting subsystems and components. Crucial to the success of the application kernel testing strategy, is the inclusion of historical test data within the XDMoD system. With this capability, site administrators can easily monitor the results of application kernel runs for troubleshooting performance issues at their site. Indeed, as the cases below illustrate, early implementation of application kernels have already proven invaluable in identifying underperforming

and sporadically failing infrastructure that would have likely gone unnoticed, resulting in frustrated end-users and wasted CPU cycles on machines that are already oversubscribed.

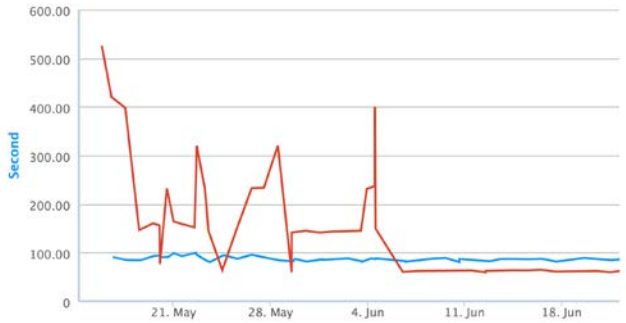

Figure 2. Plot of execution time of the NWChem application kernel on 8 cores (blue line) and 16 cores (red line) over a several month time period. Calculations on 16 cores show wildly sporadic performance degradation until early June when a patch to a bug in a parallel file system was installed.

Application Kernels have already successfully detected runtime errors on popular codes that are frequently run on XSEDE resources. For example, Figure 2 shows the execution time over the course of two months for an application kernel based on NWChem [22], a widely used quantum chemistry program that is run daily on the large production cluster at UB's Center for Computational Research. While the behavior for 8 cores (one node) is as expected, calculations on 16 cores (two nodes) in May showed wildly sporadic behavior, with some jobs failing out right and others taking as much as seven times longer to run. The source of performance degradation was eventually traced to a software bug in the I/O stack of a commercial parallel file system, which was subsequently fixed by the vendor, as evidenced by the normal behavior in the application kernel after June 4th. Indeed, the software patch to fix this problem is now part of the vendor's standard operating system release. It is important to note that this error was likely going on unnoticed by the administrators and user community for sometime and was only uncovered as a result of the suite of application kernels run at CCR.

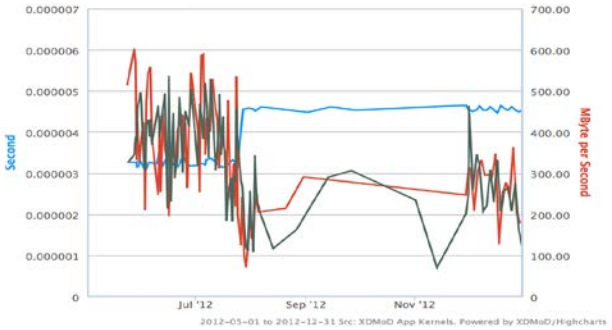

Figure 3. Application kernel data for IMB (blue), IOR (red) and MPI-Tile-IO (black) on Lonestar4. IOR and MPI-Tile-IO data show sudden drop of aggregate write throughput bandwidth and the IMB data shows an increase in latency starting on 7/24-25/2012.

Figure 3 shows a sudden decrease in file system performance on Lonestar4 as measured by 3 different application kernels (IOR, MPI-Tile-IO, and IMB). The IOR and MPI-Tile-IO both show a sudden decrease in the aggregate write throughput bandwidth, while IMB, which measures latency, shows an equally sudden increase in latency. The degradation in performance was related to a system wide upgrade and was subsequently brought back to normal performance level after notification of the degradation. Once again, without application kernels periodically surveying this space to uncover quality of service issues, the loss in performance would have gone unnoticed.

One of the most problematic scenarios entails a single node posing a critical slowdown in which the cumulative resources for a job (possibly running on thousands of processing elements) are practically idled due to an unexpected load imbalance. It is very difficult for system support personnel to preemptively catch such problems, with the result that the end-users are the "canaries" that report damaged or underperforming resources, often after investigations that are very expensive both in terms of computational resources and personnel time. An active monitoring capability designed to automatically detect such problems is therefore highly desirable. For example, Figure 4 shows the results of a log file analysis of CCR's large production cluster consisting of more than 1000 nodes. By examining only the size of the log files generated on each node (large log file size is indicative of errors) we were able to detect a loose cable on one node and a job scheduler error on another node, both of which resulted in failed jobs. Without such analysis, the loose cable or job scheduler error would have likely gone undetected, resulting in many failed jobs, frustrated users, and underperformance of the resource. While analysis of system log files is not currently included within the XDMoD framework, it is anticipated that future versions will implement this analysis, given its utility in identifying faulty hardware.

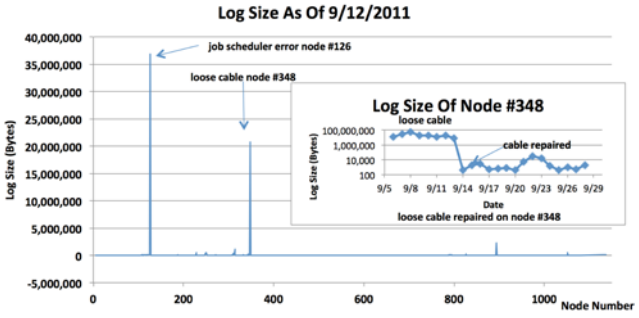

Figure 4. Plot of log file analysis for each node in CCR's production cluster. Two nodes produce very large log files. One node was found to have a loose cable and the other a job scheduler error, both resulting in failed jobs.

3.2 SUPReMM data analysis

The case studies reported in this subsection were carried out on the recently commissioned Stampede system at the Texas Advanced Computing Center (TACC), and Rush, the production cluster at UB's Center for Computational Research. Stampede is a 10 PFLOPS (PF) Dell Linux Cluster based on 6,400+ Dell PowerEdge server nodes, each outfitted with 2 Intel Xeon E5 (Sandy Bridge) processors and an Intel Xeon Phi Coprocessor (MIC Architecture). Stampede supports a 14PB global, parallel file storage managed as three Lustre file systems. Each node contains a local 250GB disk. Nodes are interconnected with Mellanox FDR InfiniBand technology in a 2-level (cores and leafs) fat-tree topology.

Rush is a 100 Tflop heterogeneous Linux cluster consisting of 7600 CPU cores (primarily 8 and 12 core nodes) and 384 GPUs with 3-4 GB of RAM/ CPU core connected by an Infiniband network. There are two parallel file systems, Isilon for user space

and Panasas for scratch. SLURM is implemented as the resource manager.

Somewhat surprisingly, most HPC centers lack the ability to measure the distribution and efficiency of end-user application codes running on center resources and accordingly have no way of knowing if the applications are well tuned to the architecture and therefore making efficient use of the HPC resource. This is certainly a desirable capability given that HPC centers are significantly oversubscribed and it is prudent resource management to ensure, as much as is possible, that optimal use of the resource is being realized.

As discussed in Section 2, XDMoD, through incorporation of SUPReMM data, can be used to profile every job run on a production HPC cluster over any desired period of time, and can therefore provide a rich set of operational and job level performance data that can be mined to gain valuable insight on the operational characteristics of the cluster as well as improve both system and end-user application performance. For example, Figures 5 and 6 show the resource use pattern in terms of idle CPU hours for a large sample of jobs run on Stampede and Rush over a time period of 6 and 3 months respectively. For the CCR data in Figures 6, 7 and 9, only jobs that do not share a node are included. The plots show "wasted" CPU hours, that is, CPU hours in which the processor is idle versus the total CPU-hours consumed for each job. The red line in each figure shows the average CPU idle fraction for all jobs. Thus, those jobs that lie below the red line are more efficient then the average job (*ie*, have a lower CPU idle fraction), while those jobs above the red line have a CPU idle fraction greater than the average. The average CPU idle fraction in Figures 5 and 6 is 0.170 and 0.175, respectively.

It is also interesting to study the fractional CPU usage, in terms of cores per node, as an additional measure of efficiency. For example, an analysis was done in which the fractional usage of Rush's 8 core nodes was calculated based on the SUPReMM data. A histogram of this data, shown in Figure 7, indicates that a small but distinct group of jobs are using only one of the eight cores.

While many users have very efficient codes, as evident by the large number of jobs well below the average CPU idle fraction lines in Figures 5 and 6 (as well as the peak at 8 cores in Figure 7), a substantial number of jobs are spending 50% or more of their CPU-hours in CPU idle mode, and it is these codes that provide the greatest opportunity for improved efficiency. While in some cases there may be a very good reason for the poor CPU utilization, for example, a code that is heavily IO dependent, it is unlikely that all jobs with a large CPU idle time fall into this category. Indeed the great utility of XDMoD/SUPReMM lies not only in its ability to detect these poorly performing codes, but to also diagnose them to suggest potential remedies. Not only does the user of the improved application benefit by faster turnaround time, but all center users benefit through the recovery of wasted CPU cycles.

The utility of XDMoD to identify and more importantly improve the performance of poorly performing codes is perhaps best brought to light through real-world examples taken from Rush, CCR's production cluster. From Figures 6 and 7, CCR staff identified a particular set of jobs that had a high idle CPU fraction and then carried out a more detailed analysis of each job's operational characteristics to determine the cause of the high idle CPU fraction. For those jobs for which there appeared to be no valid reason for the poor CPU utilization (ie high IO bandwidth), CCR staff reached out to the users to see what if any improvements could be obtained. For a number of users, the modifications were straightforward and easily implemented.

The poorly performing jobs tended to fall into one of two cases. The first case involved complex computational pipelines, where incorrect resources were requested for the jobs. Specifically, requesting two nodes when the pipeline no longer contained a distributed parallel component. The computation could not make use of the cores and memory on the second node. Serial processing tended to dominate the pipeline, along with high memory requirements.

Application parallelism was at the heart of the second case. Users assumed that the application would take advantage of multiple cores, which was not necessarily the case for specific computations. We found that these users tended to rely on the default amount of memory, rather than specifying the actual memory required for the computations.

In summary, the poorly performing jobs identified here were under requesting memory for the computations and assuming tasking parallelism that was not necessarily present in the application or code – resulting in wasted CPU cycles on a cluster that is significantly oversubscribed. CCR's Rush cluster is not unique in this regard, and clearly the ability for HPC center personnel to characterize the workload on their machines to identify underperforming application codes is highly desirable.

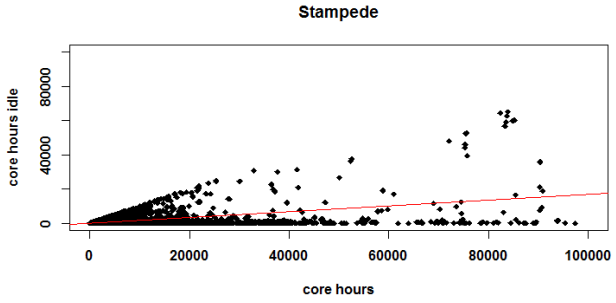

Figure 5. Plot of idle core hours versus total core hours on Stampede. The red line is the average fractional idle time (0.170) and points falling above this line spend more time with the CPU idle than the average job.

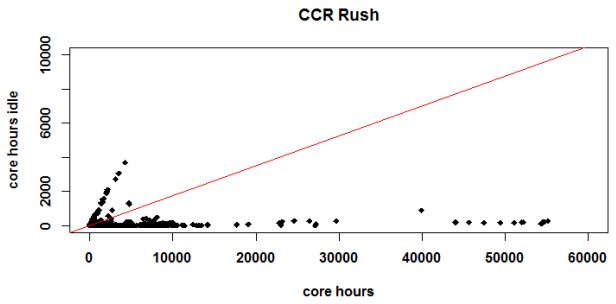

Figure 6. Plot of idle core hours versus total core hours on CCR's production cluster Rush. The red line is the average fractional idle time (0.175) and points falling above this line spend more time with the CPU idle than the average job.

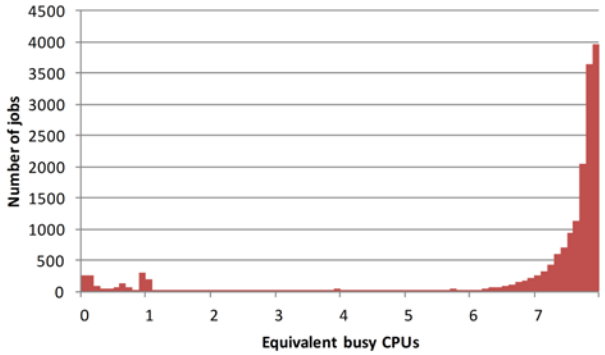

Figure 7. Histogram of equivalent busy cores for Rush 8-core nodes. A value near 8 indicates an efficient code with the 8 cores busy most of the time. The secondary peak at ~1 indicates jobs utilizing only one of the 8 available cores in a node.

It also useful to be able to measure the relative efficiency of codes that are designed for the same application area so that, in the case where there is a substantial disparity in performance, users can be guided to utilize the most efficient code. Through XDMoD/SUPReMM, center operational personnel have the ability to obtain detailed metrics on the frequency at which applications are run as well as their performance as measured by a wide variety of metrics. For example, Figures 8 and 9 show the top 10 applications by core-hours consumed for Stampede and Rush, respectively, over a window of several months. Also shown for both clusters is the CPU efficiency, and for Stampede, the memory efficiency as well. Here the CPU efficiency refers the fraction of time spent in the CPU_user mode as measured by TACC_Stats. The memory efficiency is calculated as the peak used memory divided by available memory. Note, however, the used memory includes Linux-managed disk buffers and cache, so there could be overestimation.

In general, the CPU efficiencies of the top applications on Stampede are more variable than those on Rush. Seven of the 10 applications on Rush, some of which are "local" codes developed by UB faculty, show a CPU efficiency of 75% or better. On Stampede only two of the top 10 codes have an efficiency in that range (although 9 of the 10 applications have a CPU efficiency well over 50%). It is also interesting, and probably common across academic supercomputing centers when compared to large national resources, that there is a greater proportion of 'local" codes running on the academic centers.

It is interesting to compare the relative efficiency of codes for a particular application area. For example, consider the five molecular dynamics (MD) codes running on Stampede, namely NAMD, GROMACS, AMBER, LAMMPS, and CHARMM. As shown in Figure 8, with the exception of CHARMM, they all show a CPU efficiency 70% or better and all five require only a moderate amount of memory. Since these results are obtained by aggregating many runs of each application by many users, one would expect that the results are not biased by one of the 5 codes being used for a particular type of simulation. The fact that 4 of the 5 codes have very similar performance profiles supports this conjecture. While we are still investing the relative performance of these codes, based on these results, center personnel might choose to encourage users to consider NAMD, GROMACS, LAMMPS or AMBER as opposed to CHARMM for their molecular dynamic simulations wherever appropriate.

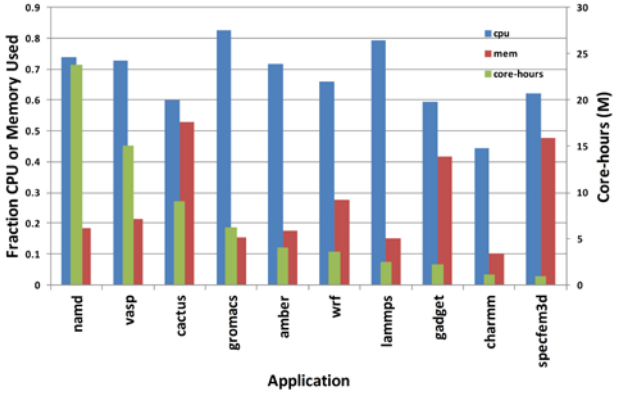

Figure 8. Top 10 applications on Stampede sorted by core-hours consumed (green columns). Also shown is each application's memory (red) and CPU efficiency (blue). The right Y-axis is the accumulated core-hours.

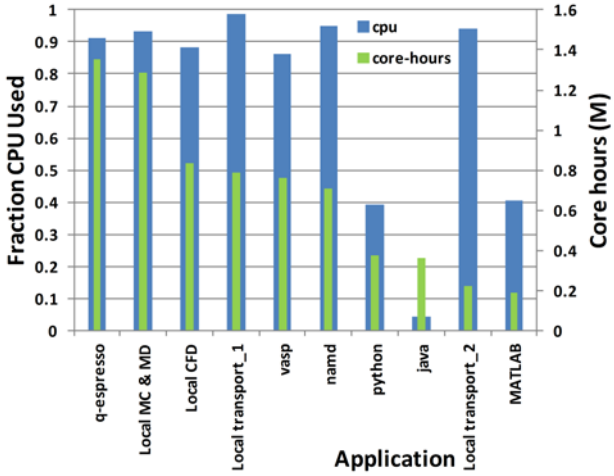

Figure 9. Top 10 applications on Rush sorted by core-hours consumed (green columns). Also shown is each application's CPU efficiency (blue column). At the time of this publication, memory efficiency was not yet available because the Rush cluster is heterogeneous consisting of nodes of varying memory capacity. The right Y-axis is the accumulated core-hours.

3.3 Interpreting XDMoD Data

While XDMoD provides the user with access to extensive usage data, like most analysis tools, care must be exercised in the interpretation of the generated data. This will be especially true for XDMoD given its open nature, the ease at which plots can be created, and the subtleties in the usage data that can require a fairly detailed understanding of the operation of the resource [23], [24]. This can be illustrated through the following example of XSEDE utilization. Consider, for example, the mean core count across Physics parent science jobs on XSEDE resources during the period 2008-2012. The distribution of jobs is highly skewed by the presence of large numbers of serial (single-core) calculations, a situation exacerbated by recent "high throughput" computing resources, as we will show. Figure 10 is a plot of mean job size (core count) from 2008-2012, showing both the "naïve" mean calculated with all jobs (blue curve) as well as the mean of all *parallel* jobs (red line). Note the divergence in the mean calculated for all jobs vs all parallel jobs that occurs in 2010. One should not be misled into thinking that the overall resources are dominated by serial or small parallel jobs as a

significant fraction are still ”capability” calculations requiring thousands of cores, as is shown by the red curve in Figure 10 which excludes serial jobs from the average core count. The mean job size is highly skewed by a rapid increase in the number of single core jobs. XDMoD can be used to identify this contribution of serial calculations, and as can be seen in Figure 11, the dramatic increase in serial jobs comes from several physics allocations ramping up on the high-throughput resources at Purdue during 2010-2012.

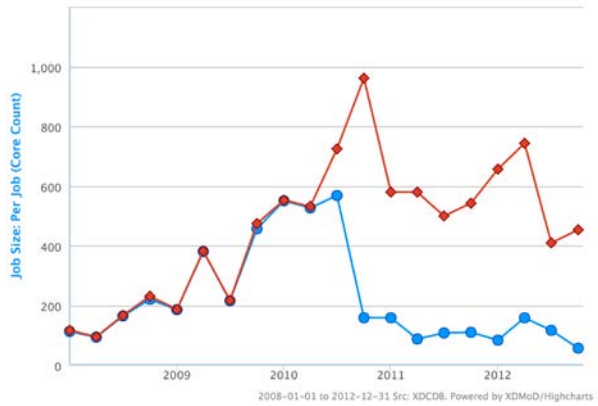

Figure 10. Mean core count for Physics jobs on XSEDE resources for the period 2008-2012, including (blue circles) and excluding (red squares) serial runs.

XDMoD puts a trove of data in the hands of the public and policy makers in a relatively easy to use interface. This data has to be used in the proper context, however, as it can be too easy to rapidly draw misleading conclusions. Based solely on the mean job size for all jobs in Figure 10, one might be tempted to wonder why the Physics allocations started using fewer cores on average in the latter half of 2010 - the answer is that they did not, rather an enterprising subgroup of them started exploiting high throughput systems on an unprecedented scale (for XSEDE), see Figure 11.

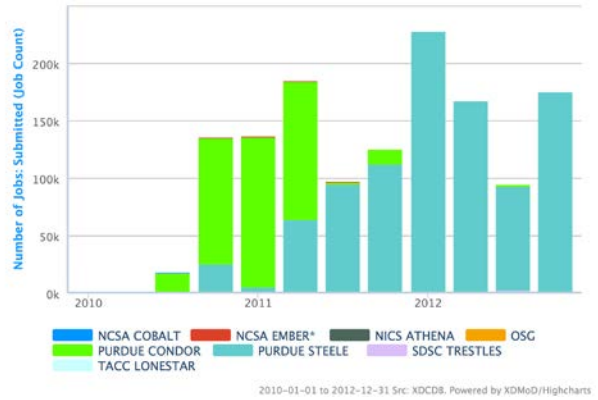

Figure 11. Number of serial jobs by resource for the parent science of physics. Large number of serial jobs begin in 2010 with introduction of Purdue Condor and Purdue Steele.

4. CONCLUSIONS AND FUTURE WORK

We have demonstrated, through several case studies, the utility of XDMoD, integrated with TACC_Stats data, as a tool for providing comprehensive management of HPC systems. With the latest enhancements, XDMoD now provides the end-user with access to extensive usage data, system performance (quality of service) metrics, scientific impact metrics, and detailed job, user, and application level information.

We believe that this capability will lead to more effective resource management and resource planning. Users will also benefit from the availability of relevant performance data for their application’s performance and the ability to compare it to the performance of other similar applications. Managers of HPC resources will, through XDMoD, be able to more readily identify underperforming hardware, system software, and user applications; all to the benefit of the end-user in the form of a more optimally maintained resource. Furthermore, as additional data is captured, such as scientific impact, it will also allow more outcome centric measures of return on HPC infrastructure investment.

There are a number of features currently being added to enhance the capabilities of XDMoD. One example is the addition of the PEAK (Performance Environment Autoconfiguration framework) to automatically help developers, system administrators, and users of scientific applications select the optimal configuration for their application on a given platform and to update that configuration when changes in the underlying hardware and systems software occur [25]. The configuration options considered for the performance optimization include the numerical libraries and settings of library parameters, and settings of other environment variables. The PEAK framework has been demonstrated to select the optimal configuration to achieve significant speedup for scientific applications executed on many of the XSEDE platforms.

In addition we are engaging with the NSF XD FutureGrid (FG) project to bring forward a plan of how to integrate Cloud resources. At this time XD FutureGrid’s data is available through its own portal. In contrast to other efforts, FG has provided an integrated monitoring solution for multiple Clouds including Nimbus, Eucalyptus, and Openstack [26]. It is important to note that the data collected for clouds and their metrics is technically significantly different from typical HPC data. Hence, it must be dealt with through different mechanisms.

ACKNOWLEDGEMENTS

This work was sponsored by NSF under grant number OCI 1025159 for the development of technology audit service for XSEDE, NSF grant number OCI1203560 for SUPReMM, and NSF MRI-R2 grant number 0959870 for Data Intensive Supercomputing.

5. REFERENCES

[1] Nagios: The Industry Standard. IT Infrastructure Monitoring: (Available from: http://www.nagios.org/[August 19, 2011]).

[2] Matthew, L., Massie, B., Chun, N., Culler, D. E., The ganglia distributed monitoring system: design, implementation, and experience. Parallel Computing 2004; 30(8): 817–840.

[3] Cacti: The Complete RRDTool-based Graphing Solution. (Available from: http://www.cacti.net/ [August 19, 2011]).

[4] Smallen, S., Olschanowsky, C., Ericson, K., Beckman, P., Schopf, J., The Inca test harness and reporting framework. Proceedings of Supercomputing, Pittsburg PA, 2004; 55–64. See also: Inca. http://inca.sdsc.edu [September 1, 2011].

[5] Hawkeye: A Monitoring and Management Tool for Distributed Systems. (Available from: http://www.cs.wisc.edu/condor/hawkeye/ [August 19, 2011]).

[6] von Laszewski, G., J. DiCarlo, and B. Allcock, “A Portal for Visualizing Grid Usage,” Concurrency and Computation: Practice and Experience, vol. 19, iss. 12, pp. 1683-1692, [2007].

[7] Martin, S., Lane, P., Foster, I., Christie, M.. TeraGrid’s GRAM Auditing & Accounting, & Its Integration with the LEAD Science Gateway, TeraGrid Workshop, 2007. (Available from: http://www.globus.org/alliance/publications/papers/TG_GRAM_auditing_and_LEAD_Gateway_final_2.pdf[August 19, 2011]).

[8] Canal, P., Green, C., GRATIA, a resource accounting system for OSG. CHEP, Victoria, B.C., 2007.
[9] DOD HPC modernization program metrics. (Available from: http://www.hpcmo.hpc.mil/Htdocs/HPCMETRIC/index.html[Dec16, 2011]).
[10] Bennett, P.M., Sustained systems performance monitoring at the U. S. Department of Defense high performance computing modernization program. In State of the Practice Reports (SC '11), Article 3. ACM: New York, NY, USA, 2011; 11 pages, DOI: 10.1145/2063348.2063352. http://doi.acm.org/10.1145/2063348.2063352.
[11] NERSC performance monitoring tools. (Available from: https://www.nersc.gov/research-and-development/performance-and-monitoring-tools/ [December 16, 2011]).
[12] DOE "operational assessment" metrics for various HPC sites, for example ORNL. http://info.ornl.gov/sites/publications/files/Pub32006.pdf[December 16, 2011]).
[13] University at Buffalo Metrics on Demand (UBMoD): Open source web portal for mining data from resource managers in HPC environments. Developed at the Center for Computational Research at the University at Buffalo, SUNY. Freely available at SourceForge at http://ubmod.sourceforge.net/[May 1, 2012].
[14] Furlani, T.R., Jones, M.D., Gallo, S.M., Bruno, A.E., Lu, C.-D., Ghadersohi, A., Gentner, R.J., Patra, A., DeLeon, R.L., von Laszewski, G., Wang, F., and Zimmerman, A., "Performance metrics and auditing framework using application kernels for high performance computer systems," Concurrency and Computation: Practice and Experience, vol. 25, pp.918-931, 2013. [Online]. Available: http://dx.doi.org/10.1002/cpe.2871
[15] Furlani, T.R., Schneider, B.I, Jones, M.D., Towns, J., Hart, D.L., Gallo, S.M., DeLeon, R.L., Lu, C.D., Ghadersohi, A., Gentner, R.J., Patra, A.K., von Laszewski, G., Wang, F., Palmer, J.T., and Simakov. N., 2013, "Using XDMoD to facilitate XSEDE operations, planning and analysis.", In *Proceedings of the Conference on Extreme Science and Engineering Discovery Environment: Gateway to Discovery* (XSEDE '13). ACM, New York, NY, USA, Article 46 , 8 pages. DOI=10.1145/2484762.2484763 http://doi.acm.org/10.1145/2484762.2484763
[16] Lu, C.D., Browne, J., DeLeon, R.L., Hammond, J., Barth, W., Furlani, T.R., Gallo, S.M., Jones, M.D., and Patra, A.K., 2013. "Comprehensive job level resource usage measurement and analysis for XSEDE HPC systems.", In *Proceedings of the Conference on Extreme Science and Engineering Discovery Environment: Gateway to Discovery* (XSEDE '13). ACM, New York, NY, USA, , Article 50 , 8 pages. DOI=10.1145/2484762.2484781 http://doi.acm.org/10.1145/2484762.2484781
[17] Browne, J.C., DeLeon, R.L., Lu, C.D., Jones, M.D., Gallo, S.M., Ghadersohi, A., Patra, A.K., Barth, W.L., Hammond, J., Furlani, T.R.,McLay, R.T., "Enabling Comprehensive Data-Driven System Management for Large Computational Facilities", Accepted SC13, Denver, CO, Nov 2013.
[18] University at Buffalo, "XDMoD portal." [Online]. Available: https://xdmod.ccr.buffalo.edu.
[19] Hammond, J. "TACC_stats: I/O performance monitoring for the intransigent" In *2011 Workshop for Interfaces and Architectures for Scientific Data Storage (IASDS 2011)*
[20] http://github.com/TACCProjects/tacc_stats
[21] Bennett, P.M., "Sustained systems performance monitoring at the U.S. Department of Defense High Performance Computing Modernization Program," in State of the Practice Reports, ser. SC '11. New York, NY, USA: ACM, 2011, pp. 3:1–3:11. [Online]. Available: http://doi.acm.org/10.1145/2063348.2063352
[22] Valiev, M., Bylaska, E., Govind, N., Kowalski, K., Straatsma, T., Dam, H.V., Wang, D., Nieplocha, J., Apra, E., Windus, T., and de Jong, W., "NWchem: A comprehensive and scalable open-source solution for large scale molecular simulations," Computer Physics Communications, vol. 181, no. 9, pp. 1477 – 1489, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0010465510001438
[23] Hart, D. L., "Measuring TeraGrid: workload characterization for a high-performance computing federation," International Journal of High Performance Computing Applications, vol. 25, no. 4, pp. 451–465, 2011. [Online]. Available: http://hpc.sagepub.com/content/25/4/451.abstract
[24] Hart, D., "Deep and wide metrics for hpc resource capability and project usage," in State of the Practice Reports, ser. SC '11. New York, NY, USA: ACM, 2011, pp. 1:1–1:7. [Online]. Available: http://doi.acm.org/10.1145/2063348.2063350
[25] Hadri, B., You, H., Moore, S., "Achieve better performance with PEAK on XSEDE resources", XSEDE '12 Proceedings of the 1st Conference of the XSEDE: Bridging from the eXtreme to the campus and beyond, Article 10 (2012): DOI =10.1145/2335755.2335801
[26] von Laszewski, G., Lee, H., Diaz, J., Wang, F., Tanaka, K., Karavinkoppa, S., Fox, G. C., and Furlani, T. , "Design of an Accounting and Metric-based Cloud-shifting and Cloud-seeding framework for Federated Clouds and Bare-metal Environments," , San Jose, CA., [September, 2012].