Open XDMoD: A Tool for the Comprehensive Management of High-Performance Computing Resources

Jeffrey T. Palmer, Steven M. Gallo, Thomas R. Furlani, Matthew D. Jones, Robert L. DeLeon, Joseph P. White, Nikolay Simakov, Abani K. Patra, Jeanette Sperhac, Thomas Yearke, Ryan Rathsam, Martins Innus, and Cynthia D. Cornelius | State University of New York, Buffalo
James C. Browne, William L. Barth, and Richard T. Evans | University of Texas, Austin

The Open XDMoD portal provides a rich set of analysis and charting tools that let users quickly display a wide variety of job accounting metrics over any desired timeframe. Two additional tools, which provide quality-of-service metrics and job-level performance data, have been developed and integrated with Open XDMoD to extend its functionality.

Today's high-performance computing (HPC) systems are a complex combination of hardware and software, and the personnel running them need to have the ability both to continuously ensure that the infrastructure is running with optimal efficiency as well as to proactively identify underperforming hardware and software. Furthermore, given that most HPC centers are oversubscribed, it's important that center personnel have the ability to monitor all jobs that run on the cluster to determine their efficiency and resource consumption as well as to plan for future upgrades and acquisitions. Surprisingly, no comprehensive open source tools provide all of these capabilities. Open XDMoD was designed to fill this void.

The genesis of XDMoD (XD metrics on demand) began with UBMoD (UB metrics on demand) an open source tool developed by the State University of New York at Buffalo Center for Computational Research (CCR) to mine resource manager files and provide basic usage accounting information (http://ubmod.sourceforge.net). Subsequent to UBMoD's release, the US National Science Foundation (NSF) awarded CCR with the Technology Audit Service for Extreme Science and Engineering Discovery Environment (XSEDE) grant to develop, among other things, the XDMoD tool[1] for managing XSEDE cyberinfrastructure, a collection of HPC compute, visualization, and storage resources that's one of the most powerful cyberinfrastructures in the world.[2] XDMoD is a substantial improvement over UBMoD in many areas, including increased functionality, an improved interface, and high-level charting and analytical tools. Because XDMoD has proven to be a valuable tool for XSEDE, a parallel effort was launched to adapt it for the general HPC community. This tool, Open XDMoD, enables local installation in—and monitoring of—individual HPC centers.

Open XDMoD is an open source project under the GNU Lesser General Public License (LGPL) version 3.0. While XDMoD utilizes information culled from the XSEDE central database, Open XDMoD creates a local (to the HPC center) data warehouse by parsing the HPC center's resource manager files. Open XDMoD has also been designed to be customizable to meet the HPC center's specific needs—for example, its hierarchical user structure (groups, departments, decanal units, and so on) can be adjusted to match that of a specific institution. In addition, Open XDMoD supports user roles so that the HPC center director can determine the appropriate data access level for end users, support personnel, administrators, the public, and so forth.

 1521-9615/15/$31.00 © 2015 IEEE Copublished by the IEEE CS and the AIP

It's important that center personnel have the ability to monitor all jobs that run on the cluster to determine their efficiency and resource consumption as well as to plan for future upgrades and acquisitions.

In addition to providing basic job accounting metrics, two tools have been developed to enhance Open XDMoD. The first is the Application Kernel Remote Runner (AKRR), which is designed to provide quality-of-service (QoS) metrics. Typically, HPC facilities don't have a mechanism to monitor the QoS they provide to their end users—instead, users become the "canaries in the coal mine" who report problems to center support personnel when their jobs suddenly run poorly or fail to run altogether. Center support personnel then determine whether the root cause for these user-reported issues is based in hardware or software problems. Depending on the problem, substantial resources in the form of CPU cycles and staff time might be wasted before the problem is detected, identified, and subsequently rectified. The key idea behind the application kernels is to periodically run a series of computationally lightweight benchmarks and applications from normal user submission queues to proactively detect problems with hardware and software. The resulting data is ingested into the Open XDMoD data warehouse, and process control algorithms automatically detect underperforming application kernels and notify support staff. The details of an underperforming metric for a given application kernel is a very useful indicator of where to start the search for the underlying issue. The kernels are designed to span all aspects of the HPC cluster operation (compute, storage, and network).

The second Open XDMoD enhancement is SUPReMM (integrated HPC systems usage and performance of resources monitoring and modeling), which queries system hardware counters to collect a range of performance information, including memory usage, filesystem usage, interconnect fabric traffic, and CPU performance.[3–7] Typically, this information is acquired at the job's start in the prolog, at the job's end in the epilog, and synchronously across all nodes in 10-minute intervals via a cron job. The user can set the data collection interval. SUPReMM provides a large variety of job performance metrics that give the HPC center directors and support personnel insight into the performance of all applications running on the cluster, without the need to recompile end user applications.

Installation and Customization of Open XDMoD

The most recent version of Open XDMoD is currently available on SourceForge at http://xdmod.sourceforge.net. Here, you'll find prepackaged versions for RedHat/CentOS systems as well as a generic source bundle suitable for installation on all Linux operating systems. Extensive installation and upgrade instructions are also available, with details on system and software prerequisites and software installation procedures, as well as a configuration and troubleshooting guide.

Open XDMoD has been developed as a customizable and extensible tool to support the analysis and reporting of HPC job data from multiple sources, such as resource manager log files and campus Lightweight Directory Access Protocol (LDAP) services, as well as auxiliary information such as user-reporting structure hierarchies (school, decanal unit, and department and project). As Figure 1 shows, the XDMoD architecture is made up of four main components: tools for processing raw resource manager log files, a local relational data warehouse, a role-based REST API, and an interactive Web-based application for viewing and analyzing data. The data warehouse and REST API are built using a traditional open source Linux, Apache, MySQL, and PHP (LAMP) software stack. The user-facing portion of Open XDMoD is a Web-based application and has been designed as an interactive tool for comparing any number of data series selected from the data warehouse. It was developed using enterprise-class user interface (UI) tools, including HighCharts (www.highcharts.com) and ExtJS (www.sencha.com/products/extjs), both of which are used in more than 50 Fortune 500 companies. ExtJS is a comprehensive, flexible, event-driven cross-browser UI toolkit that provides the basis of the Open XDMoD UI, whereas HighCharts is an interactive client-side charting tool that allows the user to view and interact with multiple data series directly in the browser.

The data warehouse sits at the core of Open XDMoD. It's comprised of both a set of databases where raw and aggregated data is stored, and also the RESTful API infrastructure required to query and present this information to the UI. The API is

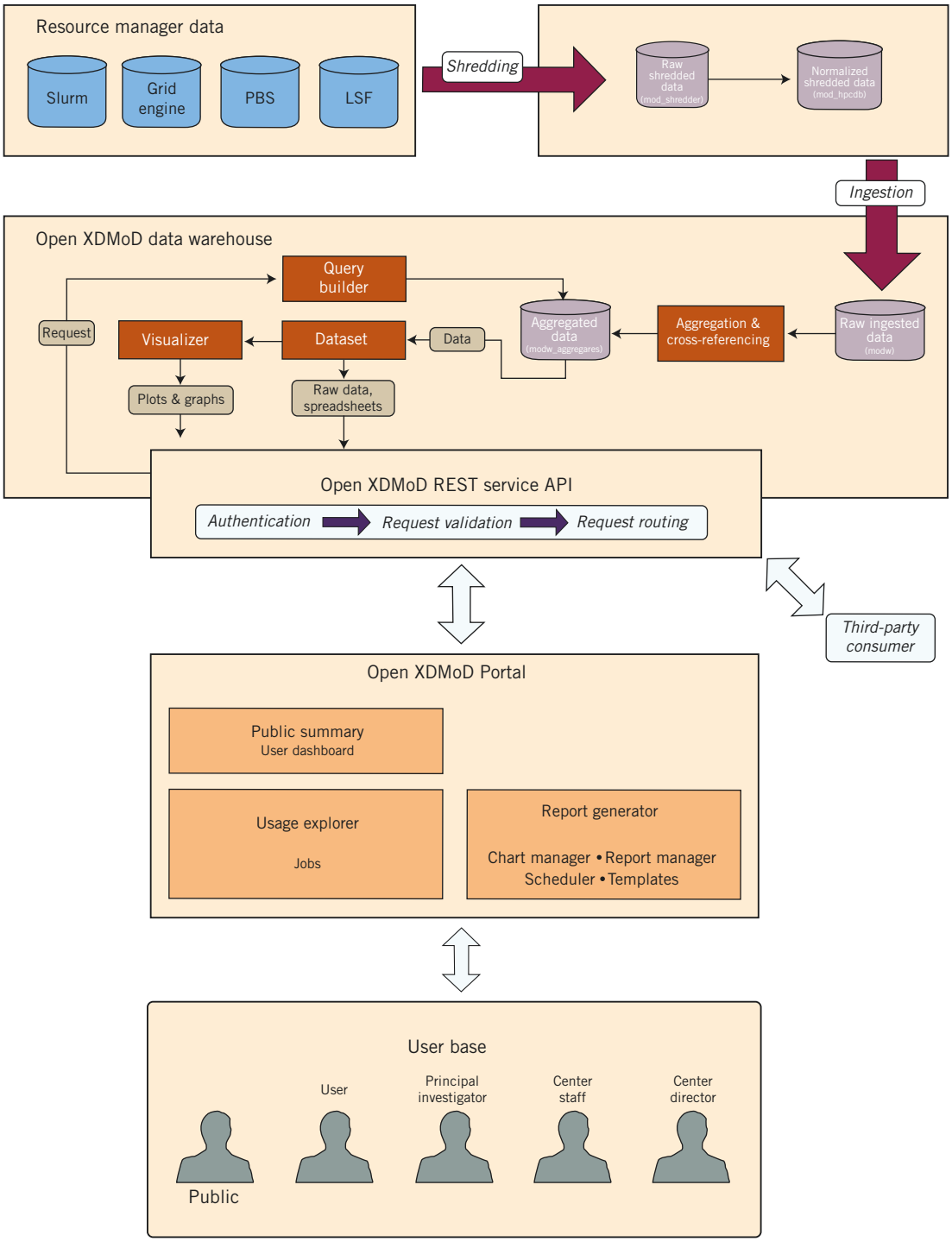

Figure 1. Open XDMoD architecture. It has four main components: tools for processing raw resource manager log files, a local relational data warehouse, a role-based REST API, and an interactive Web-based application for viewing and analyzing data. The data warehouse and REST API are built using a traditional open source Linux, Apache, MySQL, and PHP (LAMP) software stack.

used for all interactions with the core infrastructure components (data warehouse, report generator, and so on) and provides an abstraction that insulates UI components from any necessary changes to the underlying infrastructure, while ensuring that each request is authenticated and properly authorized

based on the user's role. The Open XDMoD data warehouse uses a dimensional starflake model to store the ingested data. Most data warehouse designs use dimensional models, such as star, snowflake, and starflake schemata.[8,9] A star schema is a dimensional model with fully de-normalized hierarchies, whereas a snowflake schema is a dimensional model with fully normalized hierarchies.[10] A starflake schema, a combination of a star schema and a snowflake schema, provides the best solution as it allows for a balance between the 2D normalization extremes. Upon ingestion, the transactional data is partitioned into *facts* and *dimensions* (dimensions are the reference information that gives context to the facts). For example, an HPC job transaction can be broken up into facts such as the job identifier and total CPU time consumed, and into dimensions such as the cluster name, user's department, and project under which the job was run.

Following installation and configuration, the shredder extracts data from resource manager log files and brings it into the data warehouse for use by Open XDMoD. *Shredding* refers to the process of parsing raw resource manager log files for information such as job owner, wall time, start time, number of cores, and other information, and then bringing this information into the data warehouse. During the shredding process, information is checked for consistency before being placed into the data warehouse. Common errors include discrepancies between reported wall time and start and end times, missing data, and inconsistent node or core counts. This process is typically run once per day to capture new usage data, but it can be run at any installation-required frequency. Open XDMoD provides shredders for PBS (portable batch system), SGE (sun grid engine), SLURM (simple Linux utility for resource management), and LSF (load-sharing facility). Open XDMoD users are encouraged to develop their own shredders for additional resource managers and contribute them back to the community.

Before Open XDMoD can efficiently present data to users, the ingestion process must prepare the raw data for use. During this process, relationships between various datasets are established so that information can be easily cross-referenced. To start, primary keys are assigned to various entities (users, principal investigators, resources, and so on). This flat data is then normalized before being inserted into the data warehouse.

To optimize UI responsiveness, the system automatically presents data granularity to the user based on the dynamic time period being examined, with data aggregated by day, week, month, quarter, or year. (For HPC job data, daily aggregation is the finest granularity.) For example, when viewing total CPU hours provided over a one-month period, the displayed data is aggregated daily, with each data point representing the total CPU hours on a given day. When viewing a one-year period, data points are aggregated monthly, with each data point representing the total CPU hours for a given month. The user has the option to override this selection and choose any aggregation period. To support this optimization, raw data is summarized into separate aggregate tables, one for each supported period. The use of aggregates significantly decreases query time, improving UI response time and resulting in an improved user experience, although additional disk space is required. The space versus time tradeoff is well justified as evidenced by the reduced query times in Table 1. For a center with 33 million historical jobs, we've found that the aggregation process adds only roughly 25 percent to the data warehouse's space requirements, while providing up to a 41-fold reduction in query time.

Table 1. Query speedups via aggregation.

Query	Without aggregation	With aggregation	Speedup
Monthly jobs per six resources 2005–today	122.46 s	3.48 s	35 ×
Monthly jobs per six resources 2011–today	80.49 s	1.96 s	41 ×
Daily jobs for one resource in 2012	8.77 s	0.31 s	28 ×

Open XDMoD can be customized in a variety of ways. A three-level hierarchy could be specified to group PIs together, allowing users to drill down from the highest level of the hierarchy to individual PIs and users. The user roles can be customized to change default dashboard charts or hide dimensions from certain roles. A logo image can also be added to the Open XDMoD portal's header.

Open XDMoD Tool Description

A principal function of Open XDMoD is to deliver job and system utilization data easily and in a rapid fashion that eliminates the painstaking manual data collection employed in many HPC centers. Figure 2 shows a screen capture of a typical display page of the Open XDMoD instance at CCR. The user interface for Open XDMoD features a

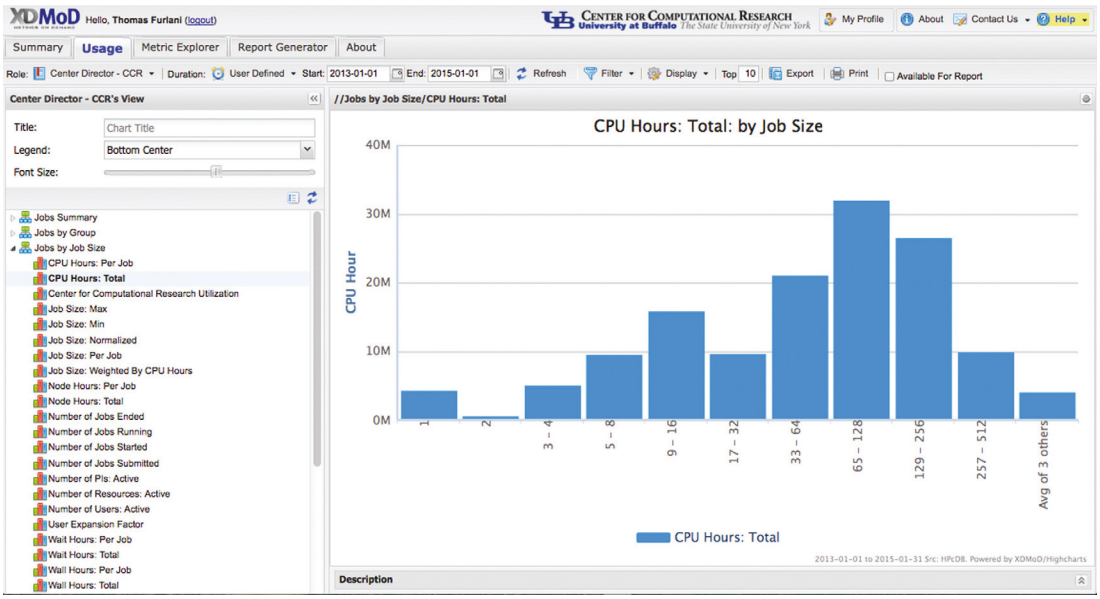

Figure 2. An Open XDMoD-generated chart showing the CPU hours delivered over a two-year period on CCR's cluster broken down by job size (number of cores). The tabs along the top of the portal allow the user to navigate to different functionalities. The Export button lets users export the chart, or the data itself, in a variety of formats. The chart can also be sent to the custom report generator for incorporation into a report.

tabbed navigation format. When Open XDMoD is launched, the user initially sees the Summary tab, which contains a collection of charts and data presenting an overview of HPC center operations. The Summary tab is customizable and can therefore be tailored by the user to best meet his or her informational needs.

The Usage tab lets users produce interactive single-metric plots. For example, Figure 2 shows a chart constructed in the Usage tab displaying the total number of CPU hours delivered over a two-year period at CCR, broken down by job size (number of cores). The user can adjust the time range displayed by the chart using the date selector and choose to display the result as either an aggregated or a time series chart. A chart in the Usage tab can be customized in three basic ways: the user can filter it to display only a subset of the data, for example, only certain, queues, users, or resources; the user can click on a data series in the plot and drill down for a more detailed analysis of that specific component; or the user can easily alter the manner in which the data is charted, choosing from line or bar charts, log or linear scales, stacking options, and so on.

The Metric Explorer tab lets users make more complex plots that compare multiple data series. For example, Figure 3 shows a plot created using the Metric Explorer that shows CPU hours, number of jobs, and average wait time per job as a function of job size (number of cores) on CCR's production cluster during 2013. Because a substantial cost of a typical HPC cluster is in the high-speed interconnect, it's important to monitor cluster usage to ensure that large parallel jobs represent a substantial fraction of the CPU cycles consumed, as is demonstrated in Figure 3. Also included in the plot is the average time that a user's job sits in the wait queue before running. Based on this information, system personnel can adjust the job scheduler to ensure that jobs in a desired size range are running without inordinate delays.

Exporting plots and data from the Usage and Metric Explorer tabs is straightforward. Open XDMoD gives the user several data export options, including PNG, SVG, comma-separated value (CSV), or XML formats.

The Report Generator tab gives users access to the custom report builder and allows them to create and save multiple customizable reports. When the user creates a chart and selects the "Available for Report" checkbox on the Usage or Metric Explorer tabs, that chart is placed into the chart pool and can be incorporated into a future report. For example, users might want to receive specific plots and data summarized in a concise report that they can download for offline viewing. They can also choose to

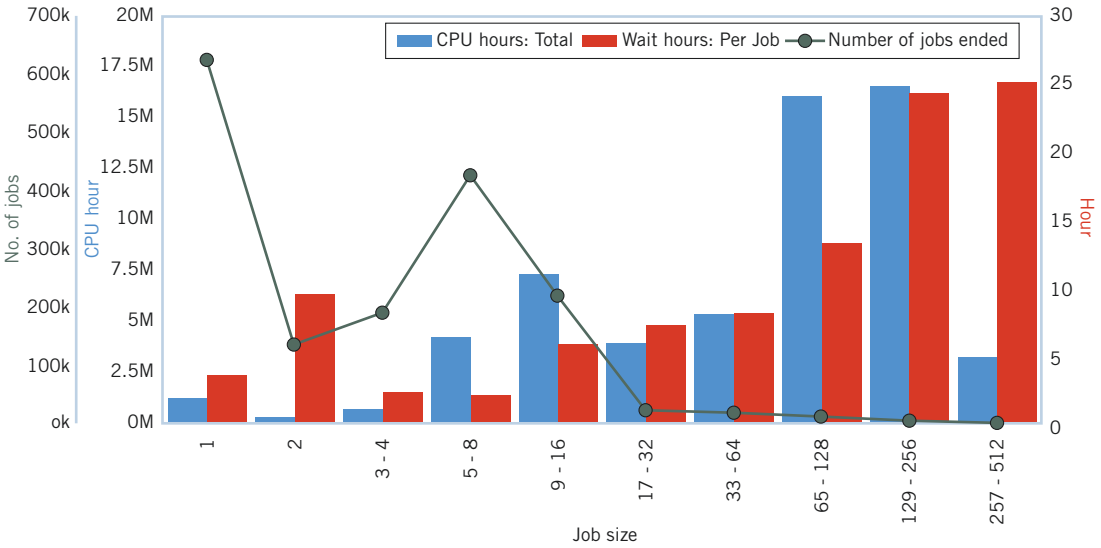

Figure 3. Chart created using the Metric Explorer tab, showing CPU hours consumed (blue column), average wait time per job (red column), and number of jobs (black line) as a function of job size (number of cores). In terms of CPU hours, the largest usage falls in the 129- to 256-core range. Jobs that use a single core represent the largest number, but their total impact on core hours consumed is very modest, with the large parallel jobs dominating the total machine throughput.

schedule the generation of custom reports at a specified interval (daily, weekly, quarterly, and so forth) and automatically receive them via email as a Word or PDF file at the specified time interval, without the need to subsequently log in to Open XDMoD. This is particularly useful for center directors, helping them easily monitor HPC system operation.

The About tab provides general information about XDMoD and Open XDMoD. The Help button gives users access to the Open XDMoD user manual, which provides details on Open XDMoD, its architecture, and how to download, install, configure, and use it. A YouTube video providing a tutorial on the use of XDMoD is also available (https://www.youtube.com/watch?v=Nghi0M7LIns).

Application Kernels for Quality of Service

Application kernels were added to Open XDMoD as a quality assurance tool. Most HPC centers use initial benchmarking to measure overall system performance at the time the HPC system is brought online and occasionally with major upgrades. Benchmarking is generally performed when the system has few or no other users. But after this initial period, little if any system-wide performance testing is performed (other than regression testing), and as a result, underperforming infrastructure components are primarily detected via failed or poorly performing user jobs. This is obviously far from an ideal way to assure QoS or identify problems. Because HPC resources are oversubscribed, a proactive QoS system is critical to avoid wasted CPU cycles.

Application kernels are comprised of computationally lightweight benchmarks and applications submitted to normal user queues in the same manner that an ordinary user would run a job. They run on user-adjustable regular schedules (typically, daily), the trade-off being between running them often and maximizing the problem detection speed or reducing the run frequency and lowering their computational overhead. At CCR, the computational burden imposed by running the application kernel suite is less than half of 1 percent of the total available CPU cycles. The suite of application kernels is designed to be diagnostic for a variety of HPC system components (compute, memory, storage, network, and applications) and have performance characteristics common to many user applications. A key to their ability to measure QoS and identify underperforming hardware and software is that each application kernel is run in the same way with the same input file on a periodic basis so that deviations from the normal operation can be readily identified. The suite of application kernels used at CCR and on XSEDE resources is available through the Open XDMoD release.

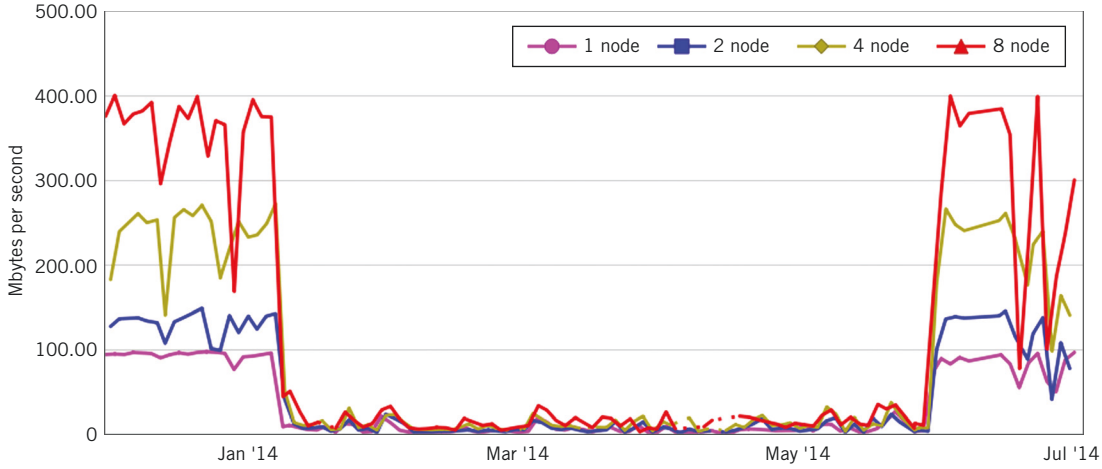

Figure 4. Application kernel success. The IOR (InterleavedOrRandom) application kernel uncovered a performance issue with Center for Computational Research's Panasas parallel file system. The timing coincided with a recent core network switch software upgrade.

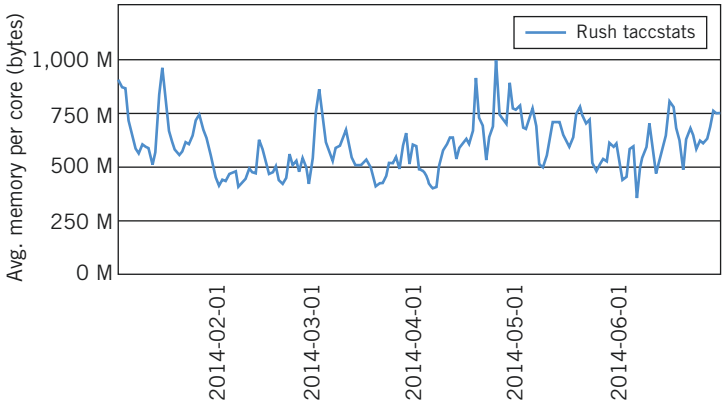

Figure 5. Memory usage per core on CCR's compute facility. The Rush cluster's nodes have 2 to 2.5 Gbytes of memory, with much less than half the available per-core memory being used.

Figure 4 shows an example of a problem that the IOR (InterleavedOrRandom) application kernel run at CCR detected (http://sourceforge.net/projects/ior-sio). Here, the IOR application kernel clearly shows a dramatic and sudden drop in write performance on 8 January 2014. Because the application kernels run frequently, CCR support staff could associate the drop in performance with a routine firmware upgrade of CCR's core network switch. The problem occurred in a production environment, requiring several scheduled monthly downtimes to properly diagnose the reason for the failure. However, as Figure 4 shows, file system performance eventually was restored to normal.

Because numerous application kernels (eight enabled by default) run frequently on several different numbers of nodes, the data the suite of application kernels generates is substantial, making it difficult to easily detect anomalies by inspection. Accordingly, an automatic detection system based on a process control algorithm notifies system support personnel when there's an obvious degradation in an application kernel's performance.

SUPReMM Performance Data

The job accounting and application kernel data available through Open XDMoD is valuable but doesn't provide center personnel with all the information needed to effectively run their facility; in fact, detailed job-level data, provided through SUPReMM, is necessary. The SUPReMM project comprises a set of open source tools to obtain, analyze, and present this detailed job-level data. Open XDMoD provides data warehousing, data analysis, and presentation components. Multiple data collection packages are supported, such as TACC_Stats,[3–5] Performance CoPilot (www.pcp.io) and resource utilization reporting[11] (to record OS and hardware counters), and XALT[12] and Lariat (to record the running application).

CCR uses the TACC_Stats and XALT packages as data sources. The TACC_Stats software collects OS and hardware performance counter data at the beginning, periodically during, and at the end of every job. Examples of the types of data recorded are memory usage, file system usage, interconnect fabric traffic, and CPU performance.

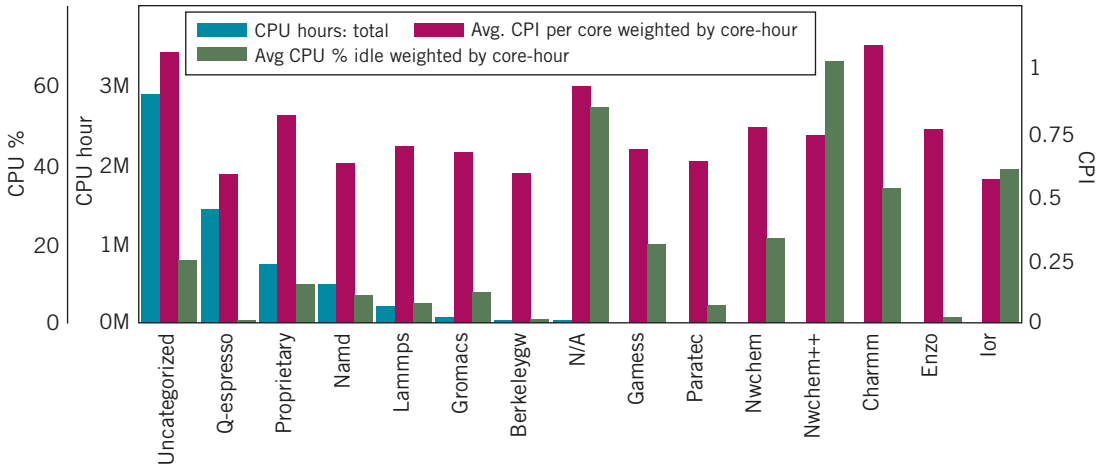

Figure 6. CCR SUPReMM performance data in Open XDMoD. The plot shows all applications running on CCR's Rush cluster over a one-month time period in 2014, with the blue bars indicating the number of CPU hours for each application. The red bars are the cycles per instruction (CPI). The black bars are the percentage of time the CPU spent in idle mode. Both CPU idle and CPI are measures of efficiency—in both cases, smaller is better. "Uncategorized" indicates that the particular application isn't identifiable as a community application (for example, it might be a user-written code). "Proprietary" indicates that the application name won't be displayed due to restrictions in the software license. "N/A" indicates that the job-level information isn't available, most likely because of the manner in which the job was launched.

XALT records application data such as executable and dynamic library names. Analysis of this kind of data in Open XDMoD lets center personnel measure individual job or application performance and diagnose job failure or poor performance. Metrics can be analyzed by job or across nodes. The accumulation of job-level performance statistics lets the center director build up a very detailed history of who's running what applications on center systems and the level to which system resources are being used for these applications.

Figure 5 shows a simple example of the type of information that SUPReMM can supply—specifically, it examines memory usage per core on CCR's Rush cluster. Rush is a heterogeneous cluster, where most of the nodes have 2 to 2.5 Gbytes of memory. From Figure 5, we can see that on average much less than half the available per-core memory is being used. This is actually fairly typical: the XSEDE facilities monitored by XDMoD and SUPReMM also use on average substantially less than 1 Gbyte per core of memory (closer to 500 Mbytes). Information of this nature can be valuable to make truly data-driven decisions for HPC system upgrades and replacements.

Figure 6 is a somewhat more complex Open XDMoD plot showing all applications running on CCR's Rush cluster, directly demonstrating the capability that SUPReMM provides for HPC center support personnel. It simultaneously shows the total number of CPU hours consumed, the average cycles per instruction (CPI), and the average percentage of time spent in CPU idle mode for each application. Information such as this, which can be used to provide a measure of the relative performance among codes designed to model the same physical systems (for example, molecular dynamics simulations), can steer users toward the most efficient code in a particular simulation space. This has the advantage of providing more "bang for the buck" for the end user, as well as helping free up CPU cycles on oversubscribed resources.

Perhaps the most important aspect of measuring job-level performance data as implemented in XDMoD is the ability to automatically flag poorly performing jobs and subsequently display a series of metrics that can be used to help diagnose the job. Figure 7 shows an example of a poorly performing job at CCR that was flagged through our XDMoD/SUPReMM implementation. Here, the user was achieving less than 35 percent CPU utilization across all of the 12-core nodes on which his job was running. Unfortunately, poorly performing jobs such as this aren't uncommon, but most centers lack

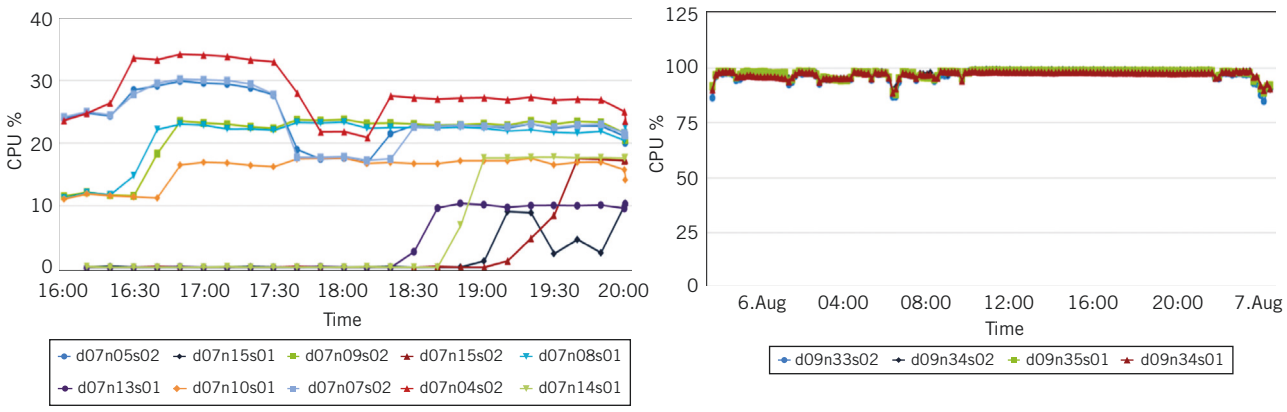

Figure 7. A poorly performing job. Inefficient CPU usage by all the cores (only 0 to 35 percent in CPU user mode) is displayed, but other metrics such as flops and memory usage are similarly indicative of an inefficient job that wasn't run properly. After consultation with CCR user support, a similar job by the same user was rerun with the result shown on the right. The job is vastly improved, as indicated by the nearly 100 percent CPU user mode for all nodes.

the ability to easily identify them among the hundreds to thousands of jobs that run daily in the typical HPC production environment. As a result, substantial CPU cycles are wasted on systems that are oversubscribed. Figure 7 also shows the dramatic increase in efficiency realized when a CCR support specialist worked with the end user—efficiencies near 100 percent on all nodes were realized. We've observed numerous similar results since deploying this technology at CCR. Clearly, the SUPReMM-based technology can be used both to support end users as well as to improve the HPC system's overall efficiency, which also benefits end users.

Open XDMoD is a comprehensive resource management system for HPC systems that provides the HPC facility manger with system operational data not obtainable from any other single open source tool. In addition to helping ensure a given QoS, Open XDMoD, through SUPReMM, provides HPC support personnel with detailed job-level performance data for all jobs running on the cluster without a significant impact on overall system performance.

Future enhancements under development include a single job viewer, available through the XDMoD interface, to give users and HPC support personnel detailed performance-level information on any job simply by entering the job ID or by selecting from a set of specifically flagged jobs; we're also looking to add the ability to gather performance-level information on accelerator-based codes.

Open XDMoD is easy to download and install, customizable, and fully supported by staff at the Center for Computational Research at the State University of New York, Buffalo. We hope you'll give it a try and let us know what you think. Suggestions for improvements are always welcome. ■

Acknowledgments

This work is supported by the National Science Foundation under grant number OCI 1203560 for SUPReMM and grant number OCI 1025159 for the technology audit service for XSEDE.

References

1. T.R. Furlani et al., "Performance Metrics and Auditing Framework Using Applications Kernels for High Performance Computer Systems," *Concurrency and Computation: Practice and Experience*, vol. 25, no. 7, 2013, pp. 918–931.
2. J. Towns et al., "XSEDE: Accelerating Scientific Discovery," *Computing in Science & Eng.*, vol. 16, no. 5, 2014, pp. 62–74.
3. J.C. Browne et al., "Enabling Comprehensive Data-Driven System Management for Large Computational Facilities," *Proc. Int'l Conf. High Performance Computing, Networking, Storage and Analysis*, 2013, article no. 86; http://doi.acm.org/10.1145/2503210.2503230.

4. C.D. Lu et al., "Comprehensive Job Level Resource Usage Measurement and Analysis for XSEDE HPC Systems," *Proc. Conf. Extreme Science and Engineering Discovery Environment: Gateway to Discovery*, 2013, article no. 50; http://doi.acm.org/10.1145/2484762.2484781.
5. J.C. Browne et al., "Enabling Comprehensive Data-Driven System Management for Large Computational Facilities," *Proc. Int'l Conf. High Performance Computing, Networking, Storage and Analysis*, 2013, pp. 1–11.
6. T.R. Furlani et al., "Using XDMoD to Facilitate XSEDE Operations, Planning and Analysis," *Proc. Conf. Extreme Science and Engineering Discovery Environment: Gateway to Discovery*, 2013, article no. 46; http://doi.acm.org/10.1145/2484762.2484763.
7. J.C. Browne et al., "Comprehensive, Open Source Resource Usage Measurement and Analysis for HPC Systems," *Concurrency and Computation: Practice and Experience*, 2013, pp. 2191–2209.
8. M. Levene and G. Loizou, "Why Is the Snowflake Schema a Good Data Warehouse Design?," *Information Systems*, vol. 28, no. 3, 2003, pp. 225–240; http://dx.doi.org/10.1016/S0306-4379(02)00021-2.
9. D.L. Moody and M.A.R. Kortink, "ER Models to Dimensional Models: Bridging the Gap between OLTP and OLAP Design," *J. Business Intelligence*, vol. 8, no. 3, 2003, pp. 7–24.
10. L. Beixin, H. Yu, and L. Zu-Hsu, "Data Warehouse Performance," *Encyclopedia of Data Warehousing and Mining*, 2nd ed., Wang John, 2009, pp. 318–322.
11. A.P. Barry, "Resource Utilization Reporting. Gathering and Evaluating HPC System Usage," *Proc. Cray Users Group*, 2013; https://cug.org/proceedings/cug2013_proceedings/includes/files/pap103.pdf.
12. K. Agrawal et al., "User Environment Tracking and Problem Detection with XALT," *Proc. 1st Int'l Workshop HPC User Support Tools* (HUST-14), 2014, pp. 32–40.

Jeffrey T. Palmer is a scientific programmer at SUNY Buffalo's Center for Computational Research. His research interests include high-performance computing and enterprise application development. Palmer has a BS in computer engineering from SUNY Buffalo. He's a member of ACM and the IEEE Computer Society. Contact him at jtpalmer@buffalo.edu.

Steven M. Gallo is the lead software engineer at SUNY Buffalo's Center for Computational Research. His research interests include software engineering, data warehousing, and high-performance computing. Gallo has an MS in computer science from SUNY Buffalo. He's a member of ACM. Contact him at smgallo@buffalo.edu.

Thomas R. Furlani is director of SUNY Buffalo's Center for Computational Research. His research interests include high-performance computing and computational chemistry. Furlani has a PhD in chemistry from SUNY Buffalo. He's a member of the American Chemical Society. Contact him at furlani@buffalo.edu.

Matthew D. Jones is associate director of SUNY Buffalo's Center for Computational Research His research interests include computational science and condensed matter physics. Jones has a PhD in physics from the University of Illinois at Urbana-Champaign. He's a member of the American Physical Society and IEEE. Contact him at jonesm@buffalo.edu.

Robert L. DeLeon is the project manager on the NSF Technology Audit Service program at SUNY Buffalo. His research interests include high-performance computing data analysis and experimental physical chemistry. DeLeon has a PhD in chemistry from the University of Rochester. He's a member of the American Chemical Society. Contact him at: rldeleon@buffalo.edu.

Joseph P. White is a computational scientist at SUNY Buffalo's Center for Computational Research. His research interests include data analytics and high-performance computing. White has a PhD in applied mathematics from the University of Manchester. He's a member of the Institute of Physics. Contact him at jpwhite4@buffalo.edu.

Nikolay Simakov is computational scientist at SUNY Buffalo's Center for Computational Research. His research interests include high-performance computing, computational biochemistry, and biophysics. Simakov has a PhD in chemistry from Carnegie Mellon University. Contact him at nikolays@buffalo.edu.

Abani K. Patra is a professor of mechanical and aerospace engineering at SUNY Buffalo. His research interests include adaptive meshing, high-performance computing, and large data-driven methodologies. Patra has a PhD in computational and applied mathematics from the University of Texas–Austin. Contact him at abani@buffalo.edu.

Jeanette Sperhac is a scientific programmer at SUNY Buffalo's Center for Computational Research. Her research interests include software engineering, data warehousing, and high-performance computing. Sperhac has an MS in chemistry from University of

Colorado and an MS in computer science from SUNY Buffalo. She's a member of ACM. Contact her at jsperhac@buffalo.edu.

Thomas Yearke is a senior software developer at SUNY Buffalo's Center for Computational Research. His research interests include creating analysis tools for high-performance computing resources. Yearke has a BS in software engineering from Rochester Institute of Technology. Contact him at tyearke@buffalo.edu.

Ryan Rathsam is a scientific programmer at SUNY Buffalo's Center for Computational Research. His research interests include parallel processing and UI/UX design. Rathsam has a BS in information technology from Rochester Institute of Technology. Contact him at ryanrath@buffalo.edu.

Martins Innus is lead scientific and urban visualization specialist at SUNY Buffalo's Center for Computational Research. His research interests include system performance monitoring and graphics programming. Innus has an MS in computer science from SUNY Buffalo. Contact him at minnus@buffalo.edu.

Cynthia D. Cornelius is the HPC systems analyst for SUNY Buffalo's Center for Computational Research. Her research interests include high-performance computing. Cornelius has a BA in computer science/mathematics/statistics from SUNY Buffalo. Contact her at cdc@buffalo.edu.

James C. Browne is professor emeritus of computer science and a research professor at the Texas Advanced Computing Center at the University of Texas, Austin. His research interests include high-performance computing. Browne has a PhD from the University of Texas. He's a fellow of ACM, APS, and AAAS and a member of IEEE. Contact him at browne@cs.utexas.edu.

William L. Barth is the director of high-performance computing at the Texas Advanced Computing Center at the University of Texas, Austin. His research interests include HPC analytics, computational fluid dynamics, and finite element methods. Barth has a PhD in aerospace engineering from UT Austin. Contact him at bbarth@tacc.utexas.edu.

Richard T. Evans is an HPC research associate at the Texas Advanced Computing Center and a research scientist lecturer in the Department of Statistics & Data Science at the University of Texas, Austin. His research interests include monitoring and performance analysis of HPC systems. Evans has a PhD in physics from the University of Illinois at Urbana-Champaign. Contact him at rtevans@tacc.utexas.edu.

AMERICAN INSTITUTE OF PHYSICS

The American Institute of Physics is an organization of scientific societies in the physical sciences, representing scientists, engineers, and educators. AIP offers authoritative information, services, and expertise in physics education and student programs, science communication, government relations, career services for science and engineering professionals, statistical research in physics employment and education, industrial outreach, and the history of physics and allied fields. AIP publishes PHYSICS TODAY, the most influential and closely followed magazine of the physics community, and is also home to the Society of Physics Students and the Niels Bohr Library and Archives. AIP owns AIP Publishing LLC, a scholarly publisher in the physical and related sciences.

Governing Board: *Louis J. Lanzerotti (chair)*, Samuel H. Aronson, Malcolm R. Beasley, G. Fritz Benedict, *J. Daniel Bourland*, *Charles Carter, Beth Cunningham*, Bruce H. Curran, Robert Doering, Judy Dubno, *Michael D. Duncan*, *H. Frederick Dylla (ex officio)*, David Ernst, Janet Fender, *Judith Flippen-Anderson (ex officio)*, Steve Iona, *Marcia Isakson*, Paul L. Kelley, Angela R. Keyser, *Kate Kirby, Rudolf Ludeke*, Jonathan Malay, *Kevin B. Marvel*, Mary Mogge, *Faith Morrison*, Homer Neal, Elizabeth A. Rogan, *Dian Seidel*, Keith Seitter, Joseph Serene, Neal Shinn, Scott Sommerfeldt, Gene Sprouse, Gay Stewart, Hervey (Peter) Stockman, Michael Turner, H. Joe Witte.
Board members listed in italics are members of the Executive Committee.

Management Committee: H. Frederick Dylla, Executive Director and CEO; Gigi Swartz, Treasurer and CFO; Theresa C. Braun, Vice President, Human Resources; Catherine O'Riordan, Vice President, Physics Resources; Judith Flippen-Anderson, Secretary.