Moab Workload Manager on Cray XT3

Michael Jackson, *Cluster Resources, Inc.*, **Scott Jackson,** *Cluster Resources, Inc.* and **Don Maxwell**, *ORNL*

ABSTRACT: *Cluster Resources Inc.™ has ported their Moab Workload Manager® to work with Cray XT3 systems (on PBS Pro, runtime, and other resource manager environments) and the National Center for Computational Sciences (NCCS) is now using Moab™ to schedule production work on their Cray XT3 system. We will discuss the process of porting Moab to the XT3, how NCCS currently uses Moab on its XT3, and how Moab provides added control over XT3 systems in terms of advanced policies, fine-grained scheduling, diagnostics, visualization of resources and improved resource utilization. We will also review uses of Moab technologies on other Cray products.*

KEYWORDS: Moab, XT3, batch, scheduling, ORNL, NCCS, workload management, Cluster Resources

1. Introduction

This paper provides a case study of the roll-out and use of Moab Workload Manager® on Oak Ridge National Laboratory's (ORNL) 5,294 processor Cray XT3 (#10 on Top500) and will briefly mention how Moab usage provided improvements to ORNL's 1,024 processor Cray X1E, SGI Altix system and SLURM based Linux Cluster. ORNL's compute and workload environment will be presented along with a compelling case of why using Moab technology significantly enhanced the overall experience and manageability of ORNL's systems. Coverage will be given of the high-level architecture of Moab and how this design was able to quickly accommodate each of these needs and allow ORNL to move forward with both a more efficient and more flexible system, as well as improved up time and decreased administrative overhead.

Cluster Resources, Inc.

Cluster Resources, Inc. is a leading provider of workload and resource management software and services for cluster, grid and utility-based computing environments. Cluster Resources is recognized as a leader in innovation and as an organization that delivers a high return on investment through its popular Moab™ product lines. With more than 2,500 sites worldwide using Cluster Resources' developed and/or maintained software and more than a decade of industry experience, Cluster Resources delivers the services and software products that enable organizations to understand, control and fully optimize their compute resources.

ORNL NCCS

The National Center for Computational Sciences (NCCS) was founded in 1992 to advance the state of the art in high-performance computing (HPC) by bringing a new generation of parallel computers out of the laboratory and into the hands of the scientists who could most use them. In 2004, the National Leadership Computing Facility (NLCF) was formed to provide the nation's most powerful open resource for capability computing, with a sustainable path that will maintain and extend national leadership for the Department of Energy's Office of Science (SC). Platforms of the NLCF are being housed in the NCCS at Oak Ridge National Laboratory.

With project allocations from SC for both the NLCF program and D.O.E.'s *Innovative and Novel Computational Impact on Theory and Experiment* (INCITE) program, resource allocation and management became a critical need for the NLCF. The NLCF currently houses a 1024 MSP Cray X1E, a 5294—processor Cray XT3, a 256—processor SGI Altix, and a 64—processor Opteron-based visualization cluster. The NCCS also continues to operate a 864—processor IBM Power 4 cluster.

Experience with batch schedulers prior to Moab consisted of IBM's LoadLeveler and Altair's PBS Pro. There are pros and cons to any scheduler, but after experience with PBS Pro on the Cray X1E, Cray XT3 and the SGI Altix, it was determined that a more flexible scheduler would be needed to accommodate the needs of the NLCF.

2. XT3 Experience and Needs

While the XT3 has many compelling advantages that make it a platform of choice such as scalability, effective performance analysis, high efficiency and performance, ORNL found a number of areas that can be further improved with use of an intelligent workload management system. The limitations that can be resolved include the following:

Resource management capabilities were inflexible forcing NCCS to choose between policies and performance.

Policies were too limited causing NCCS to not be able to apply specific policies, rather only select from a limited set of options to head in a general direction.

Policies were too static resulting in restarts being required, policies not being interconnected or having insufficient context sensitivity – policies could not say "do this if a specific condition occurs".

Prioritization was not tunable as current solutions only allowed a very simplistic prioritization and it could not meet the truly multi-factor decision model needed.

Administrators lacked "run-this-job-next" abilities as certain resource managers do not provide this function. This causes organizations to not be able to ensure critical jobs can run when desired.

Evaluation of new policies caused disruption due to the lack of effective simulation or monitoring modes. This can result in causing policy changes to negatively impact a production environment due to insufficient evaluation of the consequences of the change prior to implementation.

SSH X11 had forwarding issues as experienced when taking advantage of SSH sessions for interactive jobs that required either additional manual configuration per session or caused session display problems.

Information was too limited since system information is abstracted too far, thus blocking access to details about what was really occurring in the system. True system state could not be observed and thus not properly managed.

Failures were not effectively mitigated as failures at various levels cause unwanted down time and there was a lack of means to mitigate these failures, workaround these failures, or automatically recover from these failures.

Administration load was too high due to limited policy and optimization control. This means administrators were performing many of the scheduling and resource management tasks manually, which caused the diagnosis of issues to be difficult and time consuming.

Problem resolution time was too high since, the resource management solution was a black box without access to or control over internal behavior. It therefore caused organizations to be at the mercy of external support teams for resolution of virtually every problem. Resolution times on occasion were measured in weeks or months with significant system down time as the issues were being resolved.

Distributed resource management could be awkward and time consuming because there is no mechanism for centralized policies to flow from an organization's management to the behaviors of individual clusters and supercomputers.

Reporting was limited preventing site managers from truly visualizing use and performance of compute resources as well as comparing and contrasting performance over time or across supercomputers.

3. Moab Roll-out Experience

In November 2005, ORNL staff discussed these issues and decided to evaluate the use of Moab Cluster Suite. In December, a proof of concept project was begun in which Moab would interface with the existing resource management solution, as well as import additional state information from Cray system utilities. See Appendix A for an integration overview.

3.1 Moab Overview

Moab abstracts resource management from policy management layers allowing it to intelligently optimize and manage almost any type of resource and any type of workload. It provides advanced optimization and quality of service features together with highly flexible resource ownership and resource management policies. It is designed to interact with a large array of services orchestrating their activity according to a high-level set of mission objectives. It is designed to detect and react to a wide variety of failure events, performance events, workload events, and administrator driven events and intelligently adjust its behavior accordingly. With its flexibility, Moab can dynamically adjust policies, dynamically adjust workload, or even dynamically adjust resources in accordance with its objectives, allowing a highly optimized compute solution.

Moab is a policy engine with support for simultaneous interfaces to multiple information sources and can tie new middleware with legacy applications and tie together heterogeneous environments of varying hardware architectures, operating systems, and resource management tools.

For example, Moab can utilize specific resource management APIs such as those found in LoadLeveler, SLURM, LSF, PBS Pro, TORQUE, and others, and can also simultaneously interface with node monitors, management tools, databases, information services, etc. This is valuable in establishing the foundation for creating a unified view or resource information and management.

As a further example, Moab can be used to create a grid with Cray systems mixed with non-Cray systems. In fact this heterogeneous support allows Moab to be selected for some of the most complex environments, as it supports virtually every hardware architecture (including Cray XT3, Cray X1E, Intel Xeon, Intel Pentium, Intel Itanium, AMD Athlon, AMD Opteron, SUN Sparc and UltraSparc, HP Alpha, IBM PowerPC, SGI MIPS, and others), a broad array of operating system environments (including Unicos, all Linux flavours, Mac OS X, AIX, IRIX, SGI ProPac, HP-UX, TRU64, Solaris, BProc, Scyld, BSD and Windows), all popular interconnects (including XT3 Interconnect, Crossbar, Fast Ethernet, Gigabit Ethernet, Infiniband, Quadrics, IBM HPS, Myrinet, etc.) and all major message passing tools (including Cray MPI, MPI, OpenMPI, MPICH, MPICH2, MPICH-GX, MPICH-GM, LAM, Scali MPI, MPI Pro, PVM, VMI, Open MP, SHMEM, MLP and others).

Moab's architecture allows easy integration with external services though the use of generic C, Java, and JSP API's as well as script, database, web service and file based interfaces using XML and flat text semantics. These integration methods can be used to coordinate interaction with system services (e.g. identity managers, databases, hardware monitors, resource managers, network managers, switches, security systems, allocation managers and many others.)

As an integration platform Cluster Resources' products act much like a hub for workload management and service level centric processes. The more that the work being accomplished is connected up to additional software and informational services, the more intelligently and streamlined the processes can be. For example, if Moab is integrated with a hardware monitor it can make intelligent decisions as to how to apply the workload around problems. If integrated with a license manager, Moab can optimize workload submission around availability of licenses and report back the cost of time wasted due to waiting for licenses or the over-purchase of licenses due to a lack of requests.

Moab also allows flexible integration of command and control services from multiple sources. For example, job information within a supercomputer may be imported from one source, while resource information may be merged from two different sources. Job execution may be assigned to one interface, while network scheduling and resource provisioning may be assigned to other services. This multi-sourcing of information was a critical capability used to integrate Moab with the Cray XT3 and X1E environments.

Both Web-based and desktop graphical tools are included with the base package and are specifically designed for end-users as well as for administrators and managers.

3.2 Moab Evaluation Modes

Because Moab is often utilized in environments with mission critical production requirements and aggressive project roll-out needs, it has been designed with a large array of advanced evaluation features. These features allow Moab to be installed and executed on production resources transparently and safely evaluated in a risk-free manner. These modes include the following:

Monitor Mode – In *Monitor Mode,* Moab imports all information, processes all policies, makes all decisions, and reports back on actions it would have taken, any issues it uncovers, discrepancies in environmental information, and other factors. However, in this mode it does not directly impact workload in any way.

Simulation Mode – In *Simulation Mode,* Moab uses real-world workload traces it has collected from the supercomputer environment to create a full simulation environment where Moab schedules and tests policies while taking into account actual resource and workload configurations, timings, failures, policies, admin interactions, and other factors to predict scheduling behaviors and system performance based on historic information. In this mode Moab does not directly impact workload in any way.

Interactive Mode – In *Interactive Mode,* Moab fully schedules, but asks administrators for explicit permission before taking each action.

Partition Mode – In *Partition Mode,* Moab is able to only view a subset of resources, users, queues, and jobs. All other information is completely masked away. Inside of this sandbox partition, Moab fully schedules as if in full production. This lets sites test on the actual environment while controlling the scope of what workload and resources are under its direction.

Normal Mode – In *Normal Mode,* Moab acts in a full production manner.

3.3 Proof of Concept Plan

Due to the constraints of the existing environment and looming production roll-out deadlines, ORNL opted for an aggressive evaluation and shortened proof of concept period. To accommodate this timeframe, Cluster Resources selected to use a dual pronged approach in which an added virtualization interface would be rapidly developed and then Moab would simultaneously be evaluated in normal production mode on a development resources and in Monitor mode on the production resources. Accommodating this was not a problem for Moab because it is designed with what is likely the most advanced set of internal diagnostics in the industry.

3.4 Initial Architecture

Moab would import basic job information from the existing resource manager (PBS Pro) using a native resource manager interface. Although Moab can use the native API provided, it was chosen to use a native interface to provide more flexibility in job monitoring and job control and to avoid some errors and/or limitations in the way jobs were being reported. The XT3's state database was also loaded in via a native resource manager interface. With Moab, multiple native interfaces can be specified via a URL attribute where these URLs can point to executables, web services, databases, peer services, or even flat files.

Information about compute nodes, service nodes, login nodes, and yod nodes was loaded and the real-time configuration, load and health of each were tracked. A virtualization layer based upon Perl scripts was created to translate the raw information available via system commands into one of the generic Moab description languages, and within days, Moab was executing on the Cray XT3 platform.

Once data import was achieved, the next step was analysis of the larger environment. Taking advantage of the advanced Moab diagnostics features, this process was exceptionally fast. With Moab, issues are most commonly found, not by reviewing logs, but rather by running any of a number of diagnostic commands using the 'mdiag' routine. This command will execute a detailed analysis of the current and historical state and will report any unexpected conditions, such as unexpected values, misconfigurations, internal table overflows or corruption, externally detected failures, network issues, peer service failures, policy conflicts, etc.

Site administrators commonly use these facilities to analyze job health, node health, job priority settings, reservation status, fairshare configuration, and overall scheduler health. These diagnostics also tunnel messages from underlying systems and attach them directly to affected jobs, nodes, and other objects, making root-cause

diagnosis a painless procedure. Using this ability, a number of job and resource conflicts were detected and quickly addressed.

With all diagnostic feedback addressed, Moab was allowed to execute in normal production mode and was evaluated for a few more days. Performance was surprisingly good and exceeded even optimistic expectations. Shortly thereafter the already very short evaluation period was cut by an additional 10 days, and with that, the evaluation came to an end. Moab entered production operation, after a design, development, proof-of-concept, and testing cycle of only a little over 3 weeks.

4. Limitation Resolution

After properly ported and deployed Moab was able to provide resolutions to the limitations ORNL experienced.

4.1 Resource management capabilities were inflexible *forcing NCCS to choose between policies and performance.*

All Moab policies are fully integrated and can be used in any combination, allowing aggressive optimizations which are concurrent with service guarantees, deadlines, job policies, fairshare, preemption, reservations, resource ownership, and resource utilization limit polices.

4.2 Policies were too limited *causing NCCS to not be able to apply specific policies, rather only select from a limited set of options to head in a general direction.*

Within Moab each policy has a high degree of configurability and can be enabled on a per user, group, queue, account or QoS basis. ORNL was able to fully obtain desired workload management with existing Moab policies. To see a list of Moab capabilities visit the following URL: http://www.clusterresources.com/moabdocs

4.3 Policies were too static *resulting in restarts being required, policies not being interconnected or having insufficient context sensitivity – policies could not say "do this if a specific condition occurs".*

Moab allows dynamic modification of any policy without restarting. Policies can be context sensitive based on current or historical usage, on failures, or other events. Triggers, which are policy-based actions, can be used to detect environmental changes in real-time and modify any aspect of scheduling behavior or any aspect of the external environment.

For example, using Moab ORNL benefits from priorities that can be automatically increased and utilization limits that can be automatically adjusted if the historical usage of a key account drops below a specific threshold. Alternatively, detection of a network failure can result in a collection or arbitrary actions including automated admin notification, automatic attempts to recover the network, changes in resource ownership policies, and even preemption of certain jobs.

Another example of policies being too static, is the fact that the resource manager used only provides what is termed a "static backfill algorithm." This means that it lacks the ability to effectively adjust backfill based on changing environmental information, as it simply gathers backfillable resources until the next backfill job is able to run. Consequently, this static approach blocks resources for an extended period of time, while these resources could have successfully run multiple other jobs even before the initial backfill job is able to start.

Moab resolved this problem as it has the ability to provide a dynamic backfill capability. Through the use of advance reservations, the highest priority job is guaranteed to run at a certain time. With this knowledge, the Moab scheduler can then determine if other jobs can backfill and not interfere with the time slot for the highest priority job. This backfill policy could potentially interfere with the scheduling priority of all other jobs below the highest priority job, so Moab provides the RESERVATIONDEPTH parameter to allow sites to control the number of jobs whose priority can be protected.

4.4 Prioritization was not tunable *as current solutions only allowed a very simplistic prioritization and it could not meet the truly multi-factor decision model needed.*

Priorities are always a big issue with batch schedulers. Typically, priorities within a resource manager are implemented using queues. This requires the user to do something special in order to receive a priority boost. Moab's priorities are extremely versatile and can be based on a weighted sum of many factors.

These factors can include user, project or account, queuetime, queue, jobsize, quality of service and optionally dozens of other factors. Any one of these can be changed to implement policy changes without requiring the user to make changes to their batch scripts. NCCS has implemented each of the factors mentioned above in the job priority scheme being used on the NLCF machines. A further discussion will be provided on NLCF's priority implementation in a later section. (See section 6)

4.5 Administrators lacked "run-this-job-next" abilities *as certain resource managers do not provide this function. This causes organizations to not be able to ensure critical jobs can run when desired.*

Several features desirable for a batch system were simply not available in the resource manager layer running on both the Cray machines and the SGI system. One example feature that is commonly desirable is a "run-this-job-next" capability. Political situations, hardware and software failures, or other circumstances can lead to the need to ensure that a particular job runs next, no matter what else is in the queue. This feature was available on ORNL's LoadLeveler based system, but was not on the PBS Pro system. Moab is able to supplement PBS Pro by providing this capability through the use of the 'setspri' command which sets a system priority. Any job with a system priority has a higher priority than jobs without a system priority.

4.6 Evaluation of new policies caused disruption *due to the lack of effective simulation or monitoring modes. This can result in causing policy changes to negatively impact a production environment due to insufficient evaluation of the consequences of the change prior to implementation.*

As mentioned in the Moab Evaluation Modes section above, Moab provides a SIMULATION mode to allow administrators to evaluate potential performance based on changes to priorities or other configuration parameters. After applying and viewing the results of the potential policy and configuration changes, administrators can apply those changes which improve the performance and achieve the desired experience, while avoiding those which create new bottlenecks or negatively impact results. This same capability can be used to evaluate the impact of adding hardware resources, adjusting resources delivered to new or changing projects, and many other adjustments to see how the changes would impact jobs, without actually scheduling any jobs or impacting the production system.

4.7 SSH X11 had forwarding issues *as experienced when taking advantage of SSH sessions for interactive jobs that required either additional manual configuration per session or caused session display problems.*

The Moab integration strategy implemented on the Cray XT3 provided NCCS a unique way to solve a common problem. The Moab scheduler runs as a native binary on the XT3 while the virtualized interface layer routines are based on scripts. These routines provide the unique advantage of being able to take control of placement of the job on a particular node of the XT3. All interactive sessions currently run through the batch system due to problems that come with managing nodes outside of batch. This presents unique problems for interactive activities such as X11 session displays. SSH provides a tunnel for X11 sessions if properly configured, but taking advantage of that tunnel proved to be problematic on a machine with multiple nodes being used for login and yod launch. Moab was used to capture the fact that a job is interactive and information about the host that submitted the job; then with a simple modification to the Moab virtualization layer, it allowed NCCS to push the interactive job back to the submitting node where the original SSH tunnel was created. This automated and simplified the experience for end users.

4.8 Information was too limited *since system information is abstracted too far, thus blocking access to details about what was really occurring in the system. True system state could not be observed and thus not properly managed.*

Where other systems represented the XT3 as a single monolithic multi-processor entity, Moab was able to provide much more detailed information about the configuration, state, and actual layout of the system, along with true per job allocation of resources. This allowed administrators to more fully identify undesired utilization behaviors and identify resource failures.

4.9 Failures were not effectively mitigated *as failures at various levels cause unwanted down time and there was a lack of means to mitigate these failures, workaround these failures, or automatically recover from these failures.*

Moab's automated and manual diagnostics helped to identify and clean-up corrupt jobs that would plug the system and to identify and avoid nodes stuck with file system failures.

When nodes with jobs running would crash, insufficient information was available to the resource manager to properly understand the state of the job, so it would continue to believe that the job was running once the crashed node recovered and appeared available again. This would result in nodes being blocked, since the resource manager thought that the node was busy and would not allow additional workload to be run on those nodes, while the nodes were really idle. Moab was able to work around this issue by using policies to check to see that an appropriate yod service was running on the nodes that had experienced a failure. If the nodes did not have the yod service running and the resource manager believed there was a job running, Moab would issue a hold on the job, associate a message that there was a crash event and associated job issue, and then notify the administrator. This allowed jobs to begin to be applied to the nodes and for the administrator to be informed of the issue.

ORNL also experienced failures caused by a lack of state information being used by the resource manager when the underlying file system, in this case Lustre, was in a clean up status. Prior to Moab, the resource manager would submit jobs, even when the file system had not yet cleaned up after previous workload. This would cause jobs to experience a traffic jam as they continued to be pushed into the system for use on the nodes that were being cleaned up, even when they were not able to run. Moab was able to leverage its multi-resource manager capability to draw in additional information that effectively gave Moab a true picture of resource availability. Thus, jobs when managed by Moab would not be placed on the nodes that were not yet available for job placement.

4.10 Administration load was too high *due to limited policy and optimization control. This means administrators were performing many of the scheduling and resource management tasks manually, which caused the diagnosis of issues to be difficult and time consuming.*

Diagnostics made issue resolution much faster. Running a quick command is much easier than digging through logs that most often reside on several different hosts. Previously, for instance, diagnostics to determine why a job wasn't running were simply not available. Moab not only reports on reasons why it is not running a job, but also reports job diagnostics that are gathered from other information services and management tools with which Moab integrates.

Moab's rich diagnostic suite also provides a very quick overview of the health of the batch system and provides the administrator with the knowledge needed to alter the course of behavior, if necessary, for upcoming jobs. Previously, with no future knowledge of the scheduler's view of jobs, it was very difficult to make administrative decisions on a running system.

4.11 Problem resolution time was too high *since, the resource management solution was a black box without access to or control over internal behavior. It therefore caused organizations to be at the mercy of external support teams for resolution of virtually every problem. Resolution times on occasion were measured in weeks or months with significant system down time as the issues were being resolved.*

Moab's rich set of policies allow many items to be addressed directly via a configuration or policy change. The use of the native interface provides direct control over how job and resource information is presented and can be directly modified by sites to adjust the representation. Moab's ability to import from multiple information sources means if there is any way for an administrator to identify and work around an issue, Moab can probably do it in an automated manner. Resolution time reduced from weeks and months to hours and days.

4.12 Distributed resource management could be awkward and time consuming *because there is no mechanism for centralized policies to flow from an organization's management to the behaviors of individual clusters and supercomputers.*

Moab helps simplify supercomputer management and reduce administrative time

requirements through its ability to provide global statistics and diagnostics. For example, ORNL used Moab to interface with its Identity Manager to directly import central policies, accounts, priorities, and utilization targets. This improved information and established a foundation to control grids that could span all of ORNL's resources.

4.13 Reporting was limited *preventing site managers from truly visualizing use and performance of compute resources as well as comparing and contrasting performance over time or across resources.*

Moab statistics allowed detailed reporting of how well the supercomputer is used. Moab is able to monitor and report on utilization, queue loads, backlog, planned and actual resource consumption by groups, users or other factors. Moab can also create reports that report on service levels provided and other resource usage for organizational units or specific to projects. Other valuable reporting capabilities include Moab's ability to create bar charts, line and pie graphs and printable reports displaying: executed jobs, utilized processors, utilized nodes, requested time, utilized time, queue time, utilization, backlog, xfactor and other areas.

5. User Impact

Moab not only benefited administration by increasing system utilization, decreasing downtime and allowing more control over resources, it also enhanced the end user experience without requiring a change in submission habits.

Due to Moab's integration and support with the underlying resource manager, users have **not experienced any change in the interface** to the batch system on NLCF machines. This obviously provides a significant advantage to users who have developed scripts (e.g. PBS Pro scripts, LSF scripts, etc.) over the years to run their codes, as they can continue to **use existing scripts**.

Users now have the additional ability to query the batch system for several pieces of helpful information which was not possible with the resource manager alone. For example, a user who would like a quick turnaround time for code, due to debugging or some other need, can query the Moab scheduler using the 'showbf' command in order to **determine the resources and times available for backfilling**. This can dramatically increase productivity since it is easy to determine the size and length of a job that will backfill at any instance in time.

The Moab scheduler can also provide a **start time estimate** for a user's job by using the 'showstart' command. This provides the user with an estimated start and end time for jobs currently sitting in the queue. This is valuable information for the user, particularly when deadlines are looming or turnaround becomes a priority.

The elusive question of "why is my job not starting" can almost always be answered with Moab's 'checkjob' command. Moab has built-in **diagnostics to easily provide answers to many questions about job status, why jobs are not running and scheduler behavior**. If the job has already tried to run but got rejected due to a system problem, or if it violates other policies such as a maximum number of jobs per user, that information will be displayed with 'checkjob'. This feedback is beneficial to both the user and the Moab administrator in determining the current status of the system.

6. NLCF Moab Job Priority Implementation Example

Based on the fact that queuetime in Moab is measured in minutes, a decision was made to normalize each priority factor using minutes as the unit of measure. Jobs are currently prioritized based on quality of service, account or project priority, queue priority, jobsize and queuetime. Each of these factors is weighted based on a number of minutes and then multiplied by a value either provided by Moab or by the Moab administrator through the configuration file. The following table illustrates this configuration.

Factor	Unit of Weight	Actual Weight (Minutes)	Value
Quality of Service	# of days	1440	High (7)
Account Priority	# of days	1440	Allocated Hours (0) No Hours (-365)
Queue	# of days	1440	Debug (5) Batch (0)
Job Size	1 day / 1000cpu	1	Provided by Moab
Queue Time	1 minute	1	Provided by Moab

Each of these factors is used to implement specific policy decisions for the NLCF. For instance, the account priority is used to ensure that projects with allocations always receive added priority. To provide a priority based on allocations, Moab imports a configuration file which is generated by the NLCF Resource and Allocation Tracking System (RATS). When the allocation has been exhausted, the account will get a penalty in priority by reducing the account priority value by one year (365 days). This is to ensure that projects with remaining allocations continue to run with a higher priority but still allowing the violating job to run if cycles are available.

Favoring large jobs is another policy the NLCF chose to adopt. The jobsize factor provides the means to implement this policy. For every 1,000 CPUs requested, the job will get a priority boost in the jobsize factor of one day. This encourages large jobs by providing faster turnaround time.

Taking each actual weight, multiplying by the value, and generating the sum provides the final job priority. Furthermore, this calculation is provided for each job in a Moab diagnostic command which allows the administrator to monitor job priorities and make adjustments as needed. As an example, here is the output for a job priority calculation from the diagnostic:

```
Job     PRIORITY* Cred(Accnt:  QOS:Class)  Serv(QTime) Res(Proc)
Weights             1(1440:  1440: 1440)     1(1)        1(1)
69099    7298     98.7(0.0:    0.0:  5.0)   0.0(2.1)   1.3(96.0)
```

The resulting priority is a simple calculation.

$$1440 * 5 + 2 + 96 = 7298$$

Conclusion

The rollout has gone well. The XT3 is stable and productive. ORNL policies are now properly represented and enforced. Admin staff time is reduced, utilization is increased. Progress on future projects has accelerated, users are happier, and more science is being delivered. Moab is now running in production or monitor mode on multiple ORNL clusters and supercomputers of multiple architectures, operating systems, resource managers, and interconnects. Further, ORNL and Cluster Resources have collaborated to port Moab to the X1E environment and it is now operating successfully in monitor mode.

Acknowledgments

This research used resources of the National Center for Computational Sciences at Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

The design and technical review by Cluster Resource's Chief Technology Officer, David Jackson was invaluable in creating this paper.

About the Authors

Michael Jackson is President of Cluster Resources, Inc. and manages strategic customer relations with leading sites and partners such as ORNL and Cray. He is a co-founder of Cluster Resources, Inc., which formed in 2001. Jackson can be reached at michael@clusterresources.com or +1 (801) 717-3722.

Don Maxwell is a Senior System Administrator at Oak Ridge National Laboratory primarily focused on the Cray XT3. He has been a key member of past teams in bringing up new supercomputers for the NCCS. He can be reached at Oak Ridge National Laboratory, PO Box 2008 MS 6008, Oak Ridge, TN 37831-6008 or maxwellde@ornl.gov.

Scott M. Jackson is the Director of Software Engineering at Cluster Resources, Inc. He previously worked at IBM as well as MHPCC (DOD) & PNNL (DOE) computing facilities. Jackson is the architect/developer of QBank & Gold (resource allocation management software tools) & has actively participated in the Global Grid Forum as a chair for the Usage Record working Group.

Appendix A:

The following figure displays a number of key integration points between Cluster Resource' Moab Workload Manager product and ORNL's XT3 environment.