Unifying Heterogeneous Cray Resources and Systems into an Intelligent Single-Scheduled Environment

Scott Jackson *Cluster Resources, Inc.*

ABSTRACT: *As Cray systems are expanded and updated with the latest chip sets and technologies (for example, memory and processors), system managers may want to allow users to run jobs across heterogeneous resources to avoid fragmentation. In addition, as next-generation platforms with key differences (such as partition managers like ALPS and CPA) are added, system managers want the ability to submit jobs to the combined system, automatically applying workload to the best-available resources and unifying reporting for managers. This paper will describe how Moab Workload Manager has been integrated with Cray technologies to provide support for running jobs across heterogeneous resources and disparate systems.*

KEYWORDS: Moab, Torque, XT, batch, scheduling, workload management, Cluster Resources

1. Introduction

Cray XT systems are high value investments that are often updated, enhanced or extended during the life of the deployment. Whether this is to update software packages, enhance the system with more memory, processors or swap or to extend the capacity of the system with new racks that have the latest technologies and infrastructure, the common course is to either take the system down for one or more extended upgrade periods or create a separate compute resource. For larger systems, delivery of updates, enhancements and resource extensions can take months due to the scope of production, delivery and implementation.

This paper will cover the use of Moab and its technical capabilities that allow for effective utilization during transitional periods and co-existence with heterogeneous resources. Further, the paper will cover how to achieve unified submission and administration for disparate systems that have mixed resource types and how the ability to intelligently run jobs according to specific needs and availability of matching resources can provide added efficiency.

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. As the developers of the popular Maui Scheduler and the next generation Moab Cluster Suite, Moab Grid Suite, and other associated products, Cluster Resources has come to be recognized as a leader in innovation and return on investment. With more than 5,000 clients worldwide, and drawing on more than a decade of industry experience, Cluster Resources delivers the software products and services that enable an organization to understand, control and fully optimize their compute resources.

Moab

Moab Cluster Suite is a professional cluster management solution that integrates scheduling, managing, monitoring and reporting of cluster workloads. Moab simplifies and unifies management across one or multiple hardware, operating system, storage, network, license and resource manager environments to increase the ROI of cluster investments. Its task-oriented graphical management and flexible policy capabilities provide an intelligent management layer that guarantees service levels, speeds job processing and easily accommodates additional resources.

Torque.

TORQUE is an open source resource manager providing control over batch jobs and distributed compute nodes. It is a community effort based on the original PBS project and, with more than 1,200 patches, has incorporated significant advances in the areas of scalability, fault tolerance, and feature extensions contributed by NCSA, OSC, USC, the U.S. Dept of Energy, Sandia, PNNL, U of Buffalo, TeraGrid, and many other leading edge HPC organizations. This version may be freely modified and redistributed subject to the constraints of the included license.

2. Scheduling Jobs across Heterogeneous Resources

On Cray XT systems, system processes on the compute nodes are strictly limited for performance reasons and individual resource manager daemons do not run out on the individual compute nodes. Because of this, resource managers, such as PBS and Torque, necessarily see these systems as a single node SMP system. This view of things presents a significant challenge when nodes in the cluster are not truly homogeneous. In reality, the nodes may differ in substantial ways such as number of cores, amount of memory and swap, architecture type, software levels, static features and other characteristics.

Without an individual view of the nodes, a scheduler would be unable to make decisions regarding the placement of jobs. It would not be able to support jobs requesting heterogeneous aspects of the nodes (features, architecture, resources, node-locked licenses, etc.). It would not be able to optimize the utilization of the system by fitting the job to the resources available on the nodes. It would not be able to enforce node reservations. It would not be able to run multiple jobs on the same nodes or load balance them properly. It would not be able to partition off groups of nodes for different policies. It would not be able to track and respond to node failures, high load issues, blocked resources, etc.

By utilizing its adaptable resource manager interface, the Moab scheduler/workload manager is able to combine information from the Torque Resource Manager and the Cray partition managers (CPA, ALPS) to give Moab a complete and accurate view of the compute node resources, taking into account their heterogeneous qualities. [see “Moab and Torque on Cray XT3” by Scott Jackson, CUG 2007 - for a discussion of this architecture].

In this section, we will show how Cluster Resources architected a solution that allowed Moab to work with the Cray partition manager to support the running of jobs across heterogeneous nodes. In order to present a specific example, we will highlight the case where Moab interacts with the Cray XT3 architecture and the CPA partition manager. A similar solution has also been enabled for the Cray XT4/XT5 architecture and the ALPS partition manager but that solution will not be examined in detail in this paper. Further, Moab Grid Suite can be used to unify submission to XT3 and XT4/XT5 heterogeneous systems.

We shall highlight a case where a Cray XT3 customer started with an existing system composed of dual core nodes having 2 gigabytes of memory each and a node feature of DUAL. This customer later augmented their system with an additional number of quad core nodes having 8 gigabytes of memory each and a node feature of QUAD. They wanted to be able to achieve optimal use of these heterogeneous nodes. Under such an environment, an end user’s job may request specific node characteristics based on node feature, configured processors or configured memory.

Four Resource Selection Cases

There are a few different resource selection cases that a workload manager might handle.

1) A job may request nodes of a specified type. For example, a job might request to run only on dual core nodes specifically, or it might request to only run on nodes with 8 gigabytes of memory. This is the easiest case for a workload manager to handle.

2) A job may require the nodes to be of the same type, but it does not care which. For example, the user may want the job to run entirely on dual core nodes, or entirely on quad core nodes, but not across both simultaneously. The workload manager has to allocate a single node type to the job, but is free to choose amongst them.

3) A job may specifically request disparate chunks of nodes of multiple varieties. For example, the user may want the job to run a single master task on one quad core node having 8 gigabytes of memory, and 20 slave tasks on 10 dual core nodes. The workload manager, partition manager and parallel launcher would need to support running a job on a mixed set of heterogeneous nodes.

4) A job may not care if it is allocated across heterogeneous nodes. This gives the scheduler the greatest flexibility in maximizing utilization of the resources and avoiding fragmentation, but it is the hardest case to implement. For example, a user may ask to run on 8 processors. This may result in the allocation of 2 quad core nodes, 1 quad core node plus 2 dual core nodes, or 4 dual core nodes.

At first glance, these cases would appear to be straightforward to implement, but the problem was non-trivial because:

- ✓ The resource manager (Torque, PBS), having the restriction of not being able to run daemons on the compute nodes, is not aware of the individual nodes
- ✓ There was originally no way to launch a parallel command on heterogeneous node types
- ✓ Intelligent support needs to exist in the workload manager (scheduler and resource manager) as well as the partition manager for these various cases

The Solution

We will now examine the solutions to the four resource selection cases presented above:

1) For the first case where a job requests nodes of a specified type:

The first task we must accomplish is to allocate only nodes having the requested qualities. To do this, the workload manager/scheduler must be aware of the individual nodes and their differentiating characteristics. Instead of querying the primary resource manager directly, Moab uses its adaptable resource manager interface to combine information from the resource manager (Torque) with information obtained from the partition manager (CPA) to inform itself about the properties and state of the individual nodes, as well as the list of compute nodes allocated to each job.

By using this information, Moab is able to allocate a set of nodes consistent with the user's request, reserve these nodes via CPA, push an environment variable containing the CPA partition id ($BATCH_PARTITION_ID) into the job's environment, and start the job on an appropriate resource manager client node (pbs_mom) via the qrun command. The job script will typically contain a parallel launcher (yod) command which launches the parallel job across the allocated nodes which are derived from the (CPA) partition manager using the partition id that has been passed down from Moab.

An example of submitting a job using this mode would be:

```
qsub -l procs=8:quad hello.job
```

2) For the second case where a job may require the nodes to be of the same type, but it does not care which:

Moab uses a mechanism referred to as node sets which allows jobs to request sets of resources without specifying exactly which resources are required. For example, a node set can be defined based on node feature which says that by default, a job should run only on nodes having either the DUAL feature or the QUAD feature, but not across both at the same time.

An example of submitting a job using this mode would be:

```
qsub -l procs=8,nodeset=one
of:feature:dual:quad hello.job
```

or Moab could be configured to enforce node sets as a default behavior using a configuration setting in the moab.cfg similar to the following:

```
NODESETPOLICY        ONEOF
NODESETATTRIBUTE   FEATURE
NODESETLIST            DUAL,QUAD
```

3) For the third case where a job specifically requests disparate chunks of nodes:

A single job might be permitted to specify tasks defined with different node properties and then request to run on collections of these disparate tasks. This capability required support from the workload management system (Moab), the partition manager (CPA (also ALPS for XT4,5)) and the parallel command launcher (yod (also aprun for the XT4,5)).

To support jobs running on heterogeneous nodes, Cray modified the yod parallel command launcher to accept environment variables of the form:

```
BATCH_TUPLE0=2:1:dual
BATCH_TUPLE1=16:0:quad
```

The first colon-separated value is the number of processors (cores) in the chunk, the second value is the amount of memory per core in gigabytes, and the third value is the chip type (implemented as a partition or feature name). If the job script were to invoke the command launcher `yod hello.exe` with the above environment variables set, it would run the hello.exe command on both processors of a single dual core node requiring one gigabyte of memory per core for the application plus 4 cores each of 4 other quad core nodes (for a total of 16 cores) with no specific memory requirement.

Additionally, the partition manager interface (CPA) was enhanced to support the stringing together of chunks of cores with disparate memory sizes and core counts for the partition reservation.

A syntax was utilized in Moab/Torque of the form:

```
qsub -l
select=X[:ncpus=N1][:mem=M1gb][:{dual|quad}][+Y[:
ncpus=N2][:mem=M2gb][:{dual|quad}]] …
```

which allowed the definition and numbers of the different tasks to be passed to Moab in the job submission request. Note that since yod preferred to deal with per-core memory numbers, it was typical to allow the ncpus definition to default to a value of 1 and specify the cores directly as the number of tasks (X and Y in the example syntax).

Thus, an example of submitting a job using this case would be:

```
qsub –l select=1:mem=8gb:quad+20:dual
hello.job
```

This would result in job environment variables of:

```
BATCH_TUPLE0=1:8:quad
BATCH_TUPLE1=20:0:dual
```

and would result in a job consisting of a single master task on one quad core node using 8 gigabytes of memory plus 20 slave tasks on 10 dual core nodes.

4) For the fourth case where a job may not care if it is allocated across heterogeneous nodes:

The freedom to run jobs across heterogeneous nodes can result in improved cluster utilization and job throughput. By contrast, if jobs are not allowed to span nodes with different properties, fragmentation will occur among the homogeneous collections of nodes (those having similar memory, processor, and feature characteristics – for example).

An example of submitting a job in this mode would be:

```
qsub -l procs=8 hello.job
```

The scheduler would be free to pick the optimal set of nodes for the job – including nodes with heterogeneous qualities consistent with the job request.

As in the third case described above, if heterogeneous nodes were chosen, an appropriate CPA partition would need to be created involving multiple linked chunks, and the proper environment variables would need to be set for use by the yod parallel command launcher.

These capabilities can be extended/applied to the XT4 and XT5 system and the ALPS partition manager with the exception that the current ALPS job launcher (aprun) does not currently support a dynamic form of heterogeneous node chunking. Although aprun does support a colon delimited syntax which allows a command to be launched on chunks of heterogeneous nodes, the aprun command must be explicitly pre-constructed using command line options in the job script and must anticipate the characteristics of the nodes that will be allocated to the job. This does not allow Moab the freedom to support the fourth case as described above.

3. Scheduling Jobs Across Disparate Systems

Most large organizations today have multiple clusters. In many cases it would be advantageous to conceptually merge the nodes into a larger virtual cluster and allow submission to the combined system. Moab has been able to do this for many years by using its Grid technology, combining systems by running a Moab workload manager on each system and then coordinating the migration of jobs between systems. It is also possible for a single Moab workload manager to directly manage multiple clusters via separate resource managers. This becomes viable when the systems share a common file space and a common user space. Using this approach, users could direct their jobs toward preferred systems or resources or they could leave it to the scheduler to decide the best place for their job to run. Statistics and accounting would be combined into one system.

In this section we will describe how Moab was enhanced to allow a single Moab daemon to combine two Cray XT4 systems, managing the scheduling of jobs across two Torque systems and two separate ALPS domains.

Moab has long been able to coordinate the workload on multiple simultaneous resource managers. In fact, one of Moab's key differentiators from its predecessor, the Maui Scheduler, is its ability to efficiently integrate resources from multiple clusters and information sources.

Although this core capability already existed within the Moab architecture, the Cray XT4 case presented a number of additional challenges and cases requiring special handling.

As a reference point, in a regular single-cluster Cray XT4 Cluster running Moab and Torque, the Moab workload manager daemon and the Torque server daemon (pbs_server) runs on a head node (often named the sdb node). The Torque client daemon (pbs_mom) runs on the login nodes. If there are more than one, this is usually to provide a measure of fault tolerance and load distribution. A user may submit a job by either using qsub which submits the job directly to the Torque resource manager (which is subsequently discovered by Moab via an RM query), or via msub which submits the job to Moab, wherewith Moab invokes a qsub command to migrate the job down to the resource manager (Torque).

In the multi-cluster design, Moab is installed onto a head node independent of both clusters. Once installed, its client commands (msub, showq, etc) are installed on the head node and login nodes for each cluster. This is done so that users can submit jobs to the combined system as well as interact (query jobs, query nodes, cancel jobs, etc). The two Torque servers remain on the head nodes of the respective clusters, while the Torque client commands (qsub, qstat, etc.) are installed on the independent head node, each in their own directory. For example, the torque commands from cluster1 might be installed into /opt/torque-cluster1 with /var/torque-cluster1 as the home directory while the Torque commands from cluster2 might be installed into /opt/torque-cluster2 with /var/torque-cluster2 as the home directory. Each home directory has a server_name file which helps the associated clients communicate with the appropriate Torque server daemon.

Moab Configuration

Next, we must tell Moab about the two resource managers and how to submit jobs for that cluster. We add lines similar to the following in the Moab configuration file (moab.cfg):

```
    # Resource Manager Configuration for Cluster1
    RMCFG[cluster1] TYPE=NATIVE:XT4
SERVER=cluster1-pbs
    RMCFG[cluster1] SUBMITCMD=/opt/torque-
cluster1/bin/qsub

    # Resource Manager Configuration for Cluster2
    RMCFG[cluster2]       TYPE=NATIVE:XT4
SERVER=cluster2-sys0
    RMCFG[cluster2]      SUBMITCMD=/opt/torque-
cluster2/bin/qsub
```

The resource manager definitions describe the type as being native, meaning it uses the resource manager native interface which uses customizable translation scripts to interface with the resource manager.

The subtype is given as XT4. This special subtype allows special handling for a number of Cray XT4 specific aspects such as the necessity for creating allocation partitions through the ALPS partition manager. The XT4 resource manager native interface uses a hybrid approach for interacting with the resource manager. It interacts with the respective Torque directly via library calls to the Torque API for querying the queue (class) information, and canceling and requeuing jobs.

Moab references the SERVER parameter to know which Torque server daemon to communicate with for each resource manager interface. The node and job information is obtained by calling tools scripts (node.query.xt4.pl and job.query.xt4.pl respectively). These scripts combine information from Torque with information from ALPS and other sources. When multiple resource managers are involved, Moab passes an additional option to the script (–rm=cluster1), indicating which resource manager and ALPS system is to be queried by the script. Moab will then combine the results of querying each resource manager into a single combined system view. Additional scripts exist (partition.query.xt4.pl, partition.create.xt4.pl, partition.delete.xt4.pl) to assist Moab in the creation and management of the ALPS partitions. These scripts also may route requests to different clusters, depending on the options passed to them by Moab.

For job submission, each resource manager must be told the full path to the submission command via the SUBMITCMD parameter. When a job is submitted directly to Torque via qsub on one of the clusters, the job will be queued and run on that cluster only. However, if a job is submitted via msub then Moab will make a decision as to where the best place is to run the job and will then migrate the job to the chosen cluster via the appropriate qsub command.

The name of each resource manager is given by the name used between the square brackets of the RMCFG parameter (i.e. cluster1, cluster2 in our example). This name is used as the cluster name for job accounting purposes. Additionally, the nodes reported by the respective resource manager will belong to a partition of the same name. A job may be directed to one cluster or another by requesting a particular partition. Jobs may also be steered toward or away from certain clusters by other specifications as well, such as requested node features, resource properties, reservations, etc. If no particular cluster is targeted, Moab will make a decision based on available resources, scheduling policies and other factors, and submit the job via qsub to the selected cluster.

By default, a job will be allocated to nodes that reside exclusively on one cluster (partition) or the other. It is possible, through the use of a special “SPAN” quality of service flag, to allow a single job to span clusters. However, this must be specifically requested and access to this capability must be explicitly granted to the user by the system administrator.

Native RM Script Configuration

In order for the resource manager native interface scripts to carry out their remote functions, a Moab tools configuration file (config.xt4.pl) must be customized with the appropriate information.

ALPS Partition management calls (apbasil) are made via ssh using pre-established ssh keys so need to know the remote host and the user name which has been granted the privileges to create, query and destroy ALPS partitions, as well as the remote hosts on which these commands can be run.

```
$alpsUser = "root";
%alpsHost = ( cluster1 => "cluster1-login",
cluster2 => "cluster2-login" );
```

Torque calls (qstat, qdel, qrun, etc.) are made locally on the independent head node and need to know the appropriate path and server host name.

```
%torquePath = ( cluster1 => "/opt/torque-
cluster1/bin", cluster2 => "/opt/torque-cluster2/bin" );
%torqueHost = ( cluster1 => "cluster1-pbs",
cluster2 => "cluster2-pbs" );
```

Resolving Name Collision

Special Consideration needs to be given to the fact that there are potential name collision issues within Moab for both job and node ids. Firstly, it is highly possible that the same job id might be in use by the several Torque resource managers at the same time. However, since Torque allows one to specify the next job id number to be used, this can be easily remedied by causing the disparate systems to use different job id ranges. Another issue that can arise is that the same node ids may be in use on different clusters at the same time. This will regularly be the case since on Cray XT systems, the compute nodes are referred to by their nid (node id) numbers which are generated automatically as integer numbers.

In order for them to be recognized as distinct nodes within Moab, the node.query.xt4.pl script must prepend the node names with the cluster name. For example, node 3 on cluster1 will be reported to Moab as “cluster1.3”, while node 3 on cluster2 will be reported to Moab as “cluster2.3”. By this means, all nodes may be reported unambiguously when users or administrators query the nodes or when Moab allocates nodes to jobs. This prefix is removed from the node name when the original node ids are needed for use in Torque or ALPS, such as when creating partitions or setting the environment variable containing the node list for the job.

Multi-RM Scheduling Flow

The following is a high-level narrative breakdown of the sequence of actions performed in a typical scheduling iteration by Moab.

Moab starts out by obtaining the node information for the clusters. For each resource manager defined, it will call the node.query.xt4.pl script with the option “—rm=<cluster_name>”. This script combines information from the apbasil inventory query (via ssh) with pbsnodes information (locally by PATH) from Torque. Moab will read these in as unique node names because of the prepended cluster prefixes and automatically assigns them to the partition corresponding to the resource manager (cluster) name. Next, Moab will query the class (queue) information on each cluster using the Torque API – communicating with the appropriate Torque server as configured in the RMCFG[] SERVER value. Then Moab will query the job information by invoking the job.query.xt4.pl script for each resource manager. This script combines information from the Torque qstat command (locally by PATH) with the apbasil inventory query (via ssh).

With all information updated, scheduling can begin. Moab prioritizes the workload and grants a configured number of job reservations, and proceeds to schedule as many of these as the available resources on the various clusters will permit. After this, Moab will proceed with the remaining jobs according to priority order, backfilling jobs that could complete without adversely impacting the jobs with existing reservations. As jobs are able to be started, Moab calls the partition.create.xt4.pl script (with the –rm=<cluster_name> option) to create an ALPS partition for the job. If successful, the partition Id is recorded in a job variable and the job.start.xt4.pl script is called (with the appropriate –rm=<cluster_name> option) which issues a qrun command using the appropriate PATH and server_name to launch the job within the assigned partition. At the end of the scheduling cycle, Moab calls the partition.query.xt4.pl script for each cluster to see if there are any stale ALPS partitions (partitions for which there are no associated running jobs). If it finds, any, it will call partition.delete.xt4.pl to remove it from the associated ALPS domain.

Next, Moab handles all user interface requests that have come in since the last iteration. Here is where it services Moab job queries (showq, checkjob), node queries (mdiag –n), Moab job cancellations (canceljob), Moab job submissions (msub), etc. If a job was cancelled or requeued, Moab will issue the appropriate Torque API library call using the Torque SERVER name corresponding with the partition the job is running in associated with. If a new job was submitted via msub, Moab will make a decision as to which cluster the job should be submitted to based on job requests, resource availability and load, and other scheduling policies. The job will then be migrated to the chosen cluster by invoking the qsub command using the PATH as given by the target resource manager's SUBMITCMD variable.

Finally, Moab handles all pending resource manager events that have taken place since the previous iteration. Examples of events include a job finishing, a new job being submitted to Torque via qsub, a job being cancelled via Torque (qdel), etc. Incidentally, Torque events will immediately wake up the Moab scheduler and cause it to start a new iteration if the scheduling timer has not yet expired. If a job has finished or has been cancelled via Torque, Moab will remove the associated ALPS partition using the partition.delete.xt4.pl script with the –rm=<cluster_name> option.

The above capabilities combine to allow organizations to unify disparate systems thereby increasing utilization and ROI of the systems, reducing complexity to the end user, and allowing administrators and managers to gain a unified view of what is being used by whom.

4. Leadership Sites and Moab

When an organization invests in a Cray XT system, it represents a serious and carefully considered investment is the system and how it is managed. The following list illustrates examples of customers for whom a Moab/Torque solution was required:

Jaguar from Oak Ridge National Laboratory runs Moab and Torque on a Cray part XT4 part XT5 system with around 181,000 cores achieving a single 1.64 petaflop system by 2009.

Red Storm from Sandia National Laboratory runs Moab and Torque on a Cray XT3 with 12,960 nodes and 38,400 compute processors (by means of AMD Opteron dual and quad core processors) running the Linux/Catamount operating system. Red Storm peaks at 284.16 teraOPS theoretical performance, with 78.75 terabytes of memory, 1.7 petabytes of disk storage and 2.5 megawatts of power and cooling.

Another Leading Government Site also chose Moab and Torque to run on a Cray XT4 with over 18,000 AMD Opteron cores in roughly 100 racks.

Leadership sites chose Moab because of its ability to resolve complexity issues they encounter, while providing more flexibility, ROI and control over their resources and doing so at an equal or better price to alternatives. All sites can equally benefit in the ROI and control benefits, while applying the management technology that allows them to more easily update, enhance or extend their investment with confidence.

5. Benefits for Each Audience

Within installations such as these, there are several tiers of customers to whom the solution must appeal. Beyond the heterogeneity and unification benefits, the management tool will inevitably play a crucial role in the overall satisfaction of the funding managers, site managers, system administrators and users.

The following, though admittedly an oversimplification, are some of the added qualities that must be delivered by a workload management system for a sophisticated Cray XT customer.

High Utilization/ROI = Happy Investors

Funding managers have commissioned the procurement of the computer to achieve specific results. They want to be able to show high utilization and return on investment for their constituents. They often want to ensure that system cycles are prioritized for specific workload types and groups. Statistics and reports are important to them to provide evidence of delivered performance and utilization.

Enforce Site Objectives = Happy Managers

Site managers perform a balancing act between the principals, competing department heads, and the users. They often make heavy use of Service Level Enforcement and Guarantees to apply various qualities of service and fairness criteria to different project groups and workload types. They need flexible policies to meet performance objectives. They need to enforce resource sharing between competing political and technical interests. They can show success in these efforts to their principals via graphical charting tools. With expansion and progress always on their minds, capacity planning reports and simulation capabilities are critical tools.

Manageability = Happy Administrators

System administrators have to translate policy into action. The more powerful and flexible the tools at their disposal, the better they are able to perform their jobs. Much is expected from them, so the more that can be automated to eliminate repetitive work the more time they have to devote to system customization. Powerful diagnostics and monitoring tools are indispensable. The ability to evaluate the impact of new policies without impacting production cycles prevents unnecessary risk or policy stagnation. The more the users can do for themselves the better. Although most admins tend to favor the command line, some of the more abstract or complex analysis or customization is made easier via a graphical administrative interface.

Usability = Happy Users

Most end users don't want to learn much about the batch system. They want simple and standard job submission and batch environments that don't change from system to system. There will also always be those power users that require a flexible and powerful set of submission options. These can utilize sophisticated job information and control utilities to great advantage. All want reliable cycle delivery and predictable job execution. A web-based job submission portal is also highly desirable due to users logging in from myriads of different environments.

5.1. Other Moab Benefits

There are many compelling reasons that sophisticated sites are choosing Moab Workload Manager.

System utilization is improved to run between 90-99 percent due to intelligent resource allocation and workload ordering.

Advance Reservations (administrative, standing, job and personal) allow for high utilization around maintenance periods, coexistence of different workload types, enforcement of policy agreements and vastly stretch the capabilities of legacy queues.

Moab enforces ***Service Level Guarantees*** through the use of features such as Quality of Service, flexible priority mechanisms, fairshare and usage throttling.

Through the use of ***Resource Manager Translation***, which lets Moab emulate scripts and job language translation of other resources managers, such as PBS Pro, users can continue to use the batch submission interfaces they are used to

Additionally, Moab can create ***a Grid across your Clusters*** – bringing together different resource manager types, operating systems and architectures. Unifying an organization's clusters in this way helps to improve overall utilization, turnaround, access to a greater variety of resource types, co-allocation of disparate resources and unified batch management.

5.2 Why Torque?

There are also compelling reasons to choose to use Torque with Moab on XT systems.

Torque is an ***Industry Standard Batch System*** that is well understood and familiar to users worldwide.

It is ***free, open source*** and ***commercially supported***.

Torque provides built-in ***underlying support for Moab's advanced features.***

Torque permits Moab to handle partition creation which permits:

- ✓ Better Failure Recovery
- ✓ Reservations (Admin, Standing, etc.)
- ✓ Heterogeneous Resources
- ✓ Node Features

6. Conclusion

Whether updating, enhancing or extending your Cray XT systems or ensuring your new purchase is able to do so throughout your usage, the ability to unify heterogeneous Cray resources into an intelligent single-scheduled environment will improve your ROI, reduce your costs and risk over time, improve usability and ultimately aid your organization to accomplish its objectives faster. By adding Moab's intelligence to your Cray system, it is able to get more work accomplished and improves your experience over the life time of the investment.

About the Author

Scott M. Jackson is the Vice President 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 developing grid standards and acting as a chair for the Usage Record working Group.