PBS Pro on the Cray X1 Platform

Michael Karo
Cray Inc.
1340 Mendota Heights Road
Mendota Heights, MN 55120
USA
Voice: 1-651-605-9164
Fax: 1-651-605-9001
mek@cray.com

Abstract

Resource management software has evolved tremendously over the past several years to address the challenges involved with effectively coordinating, monitoring, and scheduling compute resources in increasingly diverse environments. Cray has adopted PBS Pro to address these challenges and provide its customers with a POSIX compliant, feature rich workload management suite. Integrated with PScheD on Cray X1 systems, PBS Pro works to provide a unified view of the underlying compute resources. By extending the scheduling and management capabilities of PScheD, users and administrators may utilize available resources with increased efficiency.

0. Background

In the mid 1980s, Cray Research, Inc. (CRI) purchased the source code for Sterling Software's Network Queuing System (NQS) package. A significant amount of development resources went toward the customization and enhancement of NQS for Cray systems. Additional packages were developed to work in conjunction with NQS including RQS and FTA. In the early 1990s, CRI began extending the capabilities of NQS to manage the workload of multiple systems across a network. The software was ported to several third party architectures, and became the Network Queuing Environment (NQE). CraySoft was formed in 1993 to market CRI software packages available for non-Cray platforms including NQE, LibSci, and the Fortran 90 compiler. While the NQS remained bundled with UNICOS, the NQE superset provided customers with a powerful means of accessing Cray hardware from non-UNICOS platforms and delivered a steady source of revenue.

After SGI purchased CRI, it was decided that NQE should be replaced by another workload management package. SGI's philosophy was to avoid markets where there were competing products. In this case, the competing product was LSF from Platform Computing. The transition to LSF proved to be costly and difficult for many Cray customers. As a result, the product lifetime of NQE continued to be extended as customers attempted to transition their environments. Consultants were employed to address specific customer issues with NQE, though the majority of development work had ceased by 1998. When Tera purchased Cray from SGI, an investigation was conducted to determine the most acceptable replacement product for NQE. After a thorough evaluation of available alternatives, PBS Pro was selected to fill the role. The selection was based on many factors including, but not limited to, cost structure and pricing, projected ease of migration, feature set, and HPC focus.

Porting and development work on PBS Pro began in mid 2001, with initial packages of version 5.2.0 ready for release in April of 2002 for most active Cray platforms. The first official release of PBS Pro version 5.2.2b for Cray T90 (CFP only), T3E, and SV1 systems occurred on July 31st, 2002. Version 5.3.1c is the current Cray supported release, with new packages planned for release in July of 2003.

1. Introduction

Resource management on Cray platforms has seldom been a trivial exercise for the user, administrator, or developer communities. Over time, much innovation has gone into the creation, enhancement, and maintenance of the various operating system components and workload management suites responsible for ensuring the efficient measurement, utilization, and policy enforcement of available resources. The addition of the Cray X1 platform introduces new resource types with associated scheduling and management challenges. In adapting PBS Pro to address these challenges, the goal has been to provide a layered approach that avoids overlapping domains where management conflicts could arise. By layering PBS Pro on top of PScheD and providing for communication between the management layers, each layer is able to address its intended scope. The enhancements to PBS Pro addressing resource management and scheduling for the Cray X1 platform running UNICOS/mp version 2.2 are described in greater detail herein.

2. Cray X1 System Resource Limits

Operating system managed resource limits under UNICOS/mp are divided into four distinct domains:

∑ Interactive/Support – standard interactive login

∑ Interactive/Application – interactive aprun(1) call

∑ Batch/Support – PBS Pro job running on support nodes

∑ Batch/Application – call to aprun(1) from within PBS Pro job script

It is the responsibility of the pbs_mom daemon to set the appropriate batch/support domain limits when the user’s shell is invoked. It is the responsibility of PScheD to recognize whether a call to aprun(1) was initiated from within a batch or interactive session, and to set limits accordingly. PScheD contacts the pbs_mom daemon running on the local system to make this determination.

In addition to the standard PBS Pro resources supported on all platforms (such as walltime, and global memory/cpu limits), the Cray X1 system implements the following additional platform specific resource types:

mppe	Maximum number of MSP processing elements that may be used by a single process running on application nodes.
mppssp	Maximum number of SSP processing elements that may be used by a single process running on application nodes.
mppfile	Maximum size of any single file that each process in the job may create while running on application nodes.
pmppt	Maximum amount of CPU time that each process in the job may use while running on application nodes.
pmppmem	Maximum resident memory segment size that each process in the job may allocate while running on application nodes.
pmppvmem	Maximum amount of virtual memory that each process in the job may allocate while running on application nodes.

Each of the resources listed in the preceding table may be specified on the qsub(1) command line from any platform, and may be assigned as min/max queue resource limits.

3. General Features

Checkpoint/Restart (CPR) functionality is supported in PBS Pro running under UNICOS/mp. Initial implementations of the CPR feature required that libcpr.a be statically linked into the pbs_mom binary. This had the unfortunate side effect of requiring a PBS Pro update whenever an incompatible modification to libcpr.a occurred. More recent implementations allow the administrator to specify the location of the CPR binary (normally /usr/bin/cpr) within the pbs_mom configuration file, greatly reducing the need for PBS Pro updates as a result of changes to CPR. The $checkpoint_utility directive is used to configure the CPR utility in the pbs_mom configuration file. The administrator may also specify the location of the PBS Pro checkpoint directory (/var/spool/PBS/checkpoint by default) by using the $checkpoint_path directive.

Users may manually checkpoint their PBS Pro jobs under UNICOS/mp by placing them on hold using the qhold(1) command. The jobs may be subsequently released using the qrls(1) command. Periodic checkpoint may be specified at job submission time through the use of the "-c" qsub(1) argument.

Another feature generally supported in PBS Pro is commonly referred to as job dependency. Job dependency allows a user to synchronize processing activities between jobs, and to construct a logical system where the order and type of jobs run may be dependent upon intermediate results.

File staging is another common feature that will benefit PBS Pro users on Cray systems. Upon job submission, a user has the ability to specify both “stagein” and “stageout” directives that instruct PBS Pro to transfer files to and from the execution host. Prior to job execution, stagein directives are carried out that transfer data to the execution host. Upon job completion, stageout directives instruct PBS Pro to transfer results from the execution host to a specified target location. Time spent performing file staging is not accrued against the job’s walltime limit.

4. PScheD Integration

The psmgr(8) and psview(1) commands allow users and administrators to view and manipulate the objects defined within PScheD. The library interface implemented via pschedRpcRequest(3) is sufficient for these commands to communicate and display information back to the user's terminal. The pschedRpcRequest(3) library routine requires that the caller specify the OutFile and ErrFile components of the pschedRpcOpts structure to be used for resulting output. To utilize such an interface within PBS Pro would require a rather awkward pipe(3) interface to capture the resulting output from a call to pschedRpcRequest(3). A new library interface layer (pschedInfo) has been developed that allows PScheD query results to be returned as C language data structures to the caller.

Enhancements to the PBS Pro resource definitions were necessary in order to support cross-platform job submission. Each of the platform specific resources described in section two have been added to the global definitions file so that non-Cray platforms may be used to submit jobs specifying UNICOS/mp specific resources.

Modifications to the pbs_mom daemon were necessary in order to service resource queries from PScheD via the PBS Pro batch request protocol. The changes allow PScheD to issue session identifier queries to pbs_mom that return resource specification information for executing PBS Pro jobs. In the event that pbs_mom is not managing the session identifier supplied by PScheD, an empty resource list is returned. Once application specific resource limits are passed to PScheD, it becomes PScheD’s responsibility to enforce them.

Extensions to the pbs_mom daemon were necessary in order to proxy node placement information from PScheD on to the PBS Pro scheduler utilizing the resource management protocol. Queries for posted and launched applications are also available.

Scheduling jobs on Cray X1 systems requires a series of steps. A normal scheduling scenario is carried out as follows:

1. A user submits a job via qsub(1) with appropriate resource specifications. If the job must be run under UNICOS/mp, the user may wish to include the “-l arch=unicosmp” argument on the command line or as a #PBS directive in their job script. The pbs_server accepts the job and places it in the appropriate queue where it will stay until the scheduler provides the server with a suitable resource assignment.

2. The scheduler uses placement information collected from pbs_mom on the UNICOS/mp system in order to determine resource availability. When a suitable match is found, it instructs the server where the job is to be run.
3. Once instructed, the server passes the job information onto pbs_mom who carries out any file staging operations, execs the user shell, and sets appropriate operating system limits through calls to setrlimit().
4. Application node placement occurs when a call to aprun(1) is encountered within the job script. The aprun(1) process contacts PScheD and requests placement of the application. PScheD queries pbs_mom with the session identifier of the aprun(1) process and pbs_mom responds with the resource specifications for the PBS Pro job. PScheD examines the user specified limits from qsub(1) and aprun(1) and may choose to prune certain resource specifications or reject the job entirely if they disagree.
5. Upon job completion, post process staging is carried out. Finally, pbs_mom sends an obituary back to the pbs_server who purges the job from the system.

By layering the PBS Pro scheduler on top of PScheD, a separation of scheduling functionality is accomplished. PBS Pro acts as the mechanism by which complex scheduling policies and algorithms may be specified and enforced without in-depth knowledge of the underlying architecture. PScheD fulfils its role of placement scheduling by maintaining an efficient operating environment for the benefit of both PBS Pro and UNICOS/mp.

5. Scheduling Features

A PBS Pro node encompasses the resources managed by a single instance of the pbs_mom daemon. For the Cray X1 platform, one pbs_mom daemon exists for each instance of PScheD. In other words, each Cray X1 partition represents a PBS node. PBS Pro nodes have a type of either cluster or time-shared. In order to support resource oversubscription on Cray hardware, a time-shared setting is suggested.

Job preemption is supported in the standard PBS Pro scheduler. The following block from the PBS Pro Administration Guide describes job preemption support:

> PBS provides the ability to preempt currently running jobs in order to run higher priority work. Preemptive scheduling is enabled by setting several parameters in the Scheduler’s configuration file. Jobs utilizing advance reservations are not preemptable. If priority jobs (as defined by your settings on the preemption parameters) can not run immediately, the Scheduler looks for jobs to preempt, in order to run the higher priority job. A job can be preempted in several ways. The Scheduler can suspend the job (i.e. sending a SIGSTOP signal), checkpoint the job (if supported by the underlying operating system), or requeue the job. (The administrator can choose the order of these attempts via the `preempt_order` parameter.)

Interactive jobs enable users to initiate interactive login sessions through the PBS Pro system. The following paragraph from the PBS Pro User Guide describes this feature in more detail:

> PBS also provides a special kind of batch job called *interactive-batch*. An interactive batch job is treated just like a regular batch job (in that it is queued up, and has to wait for resources to become available before it can run). Once it is started, however, the user's terminal input and output are connected to the job in what appears to be an `rlogin` session. It appears that the user is logged into one of the available execution machines, and the resources requested by the job are reserved for that job. Many users find this useful for debugging their applications or for computational steering.

The availability of interactive jobs in PBS Pro allows an administrator to limit access to Cray resources by restricting normal methods of interactive login access including SSH and rlogin. By forcing users to connect interactively through PBS Pro, the number of simultaneous logins and associated consumable resources may be managed more effectively.

Advance reservations are implemented within PBS Pro to enable best-effort scheduling and coordination of resources for future utilization. This feature is useful for scheduling multiple resources (through multiple reservations) and as a supporting component for grid computing functionality. The PBS Pro User Guide describes the advance reservation feature as follows:

> An *Advance Reservation* is a set of resources with availability limited to a specific user (or group of users), a specific start time, and a specified duration. Advance Reservations are implemented in PBS by a user submitting a reservation with the `pbs_rsub` command. PBS will then confirm that the reservation can be met (or else reject the request). Once the scheduler has confirmed the reservation, the queue that was created to support this reservation will be enabled, allowing jobs to be submitted to it. The queue will have an user level access control list set to the user who submitted the reservation and any other users the owner specified. The queue will accept jobs in the same manner as normal queues.
>
> When the reservation start time is reached, the queue will be started. Once the reservation is complete, any jobs remaining in the queue or still running will be deleted, and the reservation removed from the Server. When a reservation is requested and confirmed, it means that a check was made to see if the reservation would conflict with currently running jobs, other confirmed reservations, and dedicated time. A reservation request that fails this check is denied by the Scheduler. If the submitter did not indicate that the submission command should wait for confirmation or rejection (-I option), he will have to periodically query the Server about the status of the reservation or wait for a mail message regarding its denial or confirmation.

Peer Scheduling is a feature that enables jobs being managed by a particular PBS Pro server to be serviced by an alternative server. This feature is particularly useful when managing and balancing resources under the control of multiple PBS Pro server installations. Sharing of departmentally controlled resources within an organization or between multiple organizations represents a scenario where peer scheduling may be employed.

The PBS Pro Administration Guide defines load balancing as follows:

> A policy wherein jobs are distributed across multiple timeshared hosts to even out the workload on each host. Being a policy, the distribution of jobs across execution hosts is solely a function of the Job Scheduler.

The PBS Pro scheduler implements load balancing either as a function of system load average or in a simple round robin fashion.

The standard PBS Pro scheduler supports fairshare scheduling configurations. The following excerpt from the PBS Pro Administration Guide describes this feature in more detail:

> PBS fairshare is similar to the UNICOS implementation of fairshare. Users are put in a fairshare group file. The file is read in and a tree is created. The tree consists of groups (nodes) and entities (leaves). Groups can contain groups. Every node and leaf has a number of shares associated with it. Priorities can be derived from these shares by taking a ratio of them to all the rest of the shares. The fairshare capability allows an administrator to set which PBS resource is collected for fairshare usage. If unspecified, the default resource is CPU time.

6. Computational Grids

Software products that could support the ability to access and manage Cray systems in computational grid environments are currently being investigated. Grids represent a promising technology for the scheduling and management of diverse resource types across geographical and political boundaries. Delivering a grid solution for Cray systems would enhance the variety of available computational resources on the grid, and provide increased accessibility to Cray platforms. Of the various products available, Globus represents an attractive alternative for several reasons including:

∑ Functionality
∑ Ubiquity
∑ Standards based approach to design
∑ Open source, community based development effort
∑ Historical contribution to grid and distributed computing technologies

PBS Pro currently provides support for the submission and management of jobs to Globus managed resources on non-Cray platforms. In addition, Globus supports the ability to interface with PBS Pro as a JobManager (run mechanism) component. Additional analysis is required to determine demand for such an offering. Furthermore, investigation into how best to package and support potential configurations would be required.

7. Conclusions

The policies and mechanisms associated with resource management on Cray X1 systems are nontrivial. By taking a layered approach to design and implementation, each component is focused toward managing a particular aspect of the overall solution. At the processor level, UNICOS/mp is responsible for interacting with the hardware to manage resources associated with individual Cray X1 nodes. PScheD interacts closely with UNICOS/mp to ensure efficient placement and scheduling of applications across multiple nodes of a Cray X1 partition. Layered on top of PScheD, PBS Pro supports the ability to express and enforce more complex resource management policies across one or more instances of PScheD. Though not yet supported, grid software such as Globus may offer additional accessibility and management capabilities in the future. All of these components work in concert to provide end users and administrators with the ability to efficiently utilize and manage Cray X1 resources.

8. References

1. Altair Engineering, Inc., *PBS Pro Administration Guide*
2. Altair Engineering, Inc., *PBS Pro User Guide*
3. Altair Engineering, Inc., *PBS Pro Project Pages*, http://www.pbspro.com
4. Cray Inc., *PBS Pro Release Overview, Installation Guide, and Administrative Addendum for Cray Systems*, Publication S-2345-52
5. Cray Inc., *Network Queuing Environment*, http://www.cray.com/products/software/nqe.html
6. The Globus Project, *Globus Project Pages*, http://www.globus.org/