# External Torque/Moab on an XC30 and Fairshare

Tina M Declerck, Iwona Sakrejda
NERSC
Lawrence Berkeley National Lab
Berkeley, CA USA
TMDeclerck,ISakrejda@lbl.gov

*Abstract*— **NERSC's new Cray XC30, Edison, utilizes a new capability in Adaptive Computing's Torque 4.x and Moab 7.x products which allows the Torque server and Moab to execute external to the mainframe. This configuration offloads the mainframe server database and provides a unified view of the workload. Additionally, it allows job submissions when the mainframe is unavailable/offline. This paper discusses the configuration process, differences between the old and new methods, troubleshooting techniques, fairshare experiences, and user feedback. While this capability addresses some of the needs of the NERSC community it is not without tradeoffs and challenges.**

***Keywords—Torque; Moab; fairshare; external configuration***

## I. Introduction

Job scheduling is an important part of the functionality of any high-performance computing center. The National Energy Research Scientific Computing Center (NERSC) has used a combination of Torque and Moab on most of our systems with good results. When login nodes are internal to the Cray job scheduling works well. However, since Cray provided the ability to utilize external login nodes, Torque and Moab configuration is either limiting or more complex depending on how the site wants to access the scheduling system. The interactions with Cray's job launcher, Alps, required the Torque and Moab services to remain internal to the Cray system. On our current Cray XE-6 system, Hopper, each of the login nodes ran a torque/moab server which directed jobs to the internal torque / moab. The purpose of this was to allow users to continue to work even when the Cray was not available. However, this meant that there were now several torque/moab configurations to maintain. Adaptive Computing modified the Torque / Moab configuration in version 4.1 and 7.0 respectively in a way that allows the torque server and moab to run external to the Cray. This also aligned things in torque and moab to a more standard configuration. A detailed description of the installation used at our site to install and configure external Torque and Moab on the Cray provides a basis for other sites to make use of the same framework.

Fairshare provides some additional methods for determining which jobs should run. While this has been used at NERSC for serial work for several years, it hasn't been used on multi-node jobs. This is an experiment to see how well it balances the workload while allowing large jobs to run.

## II. Understanding the Components

A short description of the components of Torque, Moab, and Alps will help to ensure the explanations provided in the document are understood.

### A. Torque

Torque is a resource manager. It's primary function is to manage jobs. Jobs are submitted to queues, monitored while running and then the time and resources used are recorded. It has several components, including the server, the moms, and in this version an authentication daemon used for communication between the components in the cluster. It also provides a scheduing capability that is unused at our site.

- pbs_server - maintains all information about the cluster, the moms, and the jobs. It essentially runs the show.
- qmgr – interface used to configure and maintain queue and server configuration.
- pbs_mom - tracks the job while it is running and does the hand–off to alps. With the new version there is a special mom called the alps_recorder. It tracks which nodes are available for jobs and provides the information to the server which in turn hands it off to moab.

### B. Moab

Moab is the job scheduler. It determines which job will be the next to run and assigns available nodes to it. It has many options to allow each site to to prioritize jobs as they prefer. It also provide a reservation system that is used both for scheduling jobs and for allowing manual resource reservations.

### C. Alps

Alps is Cray's job launcher. It takes the information from the mom and ensures the compute nodes are configured as needed for the job and then maintains status of the job while it is running.

## III. Description of Torque/Moab Changes

Prior to the Torque 4.1 and Moab 7.0 update the interaction with Alps was through a combination of Moab, Torque, and scripts. This is different from the way Torque and

Moab work on other, non-Cray, systems. On those systems, Moab gets it's information about node availability from Torque. This change allows Moab to work the same on all systems.

Torque has been modified to run a special mom called the alps_reporter mom that handles the interaction with Alps. It gets information from Alps regarding configured and available resources. This information is also used by Torque to create a virtual node for each compute node. Previously only the login (mom) nodes could be seen by Torque using the pbsnodes command. With these modifications to Torque both the torque server and Moab can be run external to the Cray. This has several advantages including the ability to allow users to continue to work on external nodes while the Cray is down without running multiple instances of torque.

## IV. Installation and Configuration

This document assumes an understanding of a standard Torque and Moab configuration. The intent is to provide details about how the external configuration varies from the standard Torque and Moab installation on a Cray system. The Moab manual contains instructions for installing the new Torque and Moab on the sdb (system data base) node. See the documentation int Adaptives Moab manual [1]. Since the external installation is slightly different there were some modifications. The configuration at NERSC has external login servers and uses Cray's management server running Bright [2] on the external nodes. On the Cray we have a fairly standard configuration. The mom nodes are re-purposed compute nodes and the sdb has an external 1 GE network connection. We use a persistent var on the service nodes which makes it easier to configure torque using the default configuration. Figure 1 shows our system configuration.

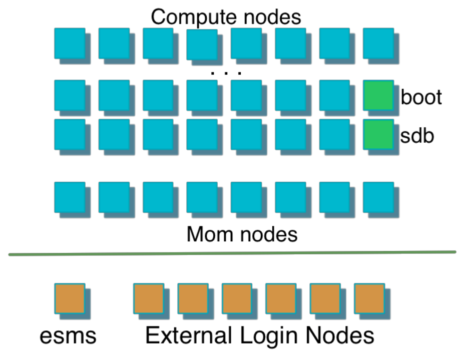

Figure 1. Edison System Configuration

Adaptive recommends that no jobs be running when you upgrade to Torque 4.1.0. Jobs may be queued but not running. If you are moving to external Torque, Adaptive doesn't appear to have a tool to move jobs so draining may be the best option.

### A. *Torque Server*

One of the primary considerations for installing torque and moab externally is ensuring a good communication path between the external nodes and the sdb and mom nodes. If your site uses multiple networks as ours does choosing the name of your server is important. Torque prefers the name returned by hostname and requires this for the mom nodes. This is where most of the issues occurred on initial install. At our site the connection to the sdb through its 1GB ethernet interface was on a different network than the one used to connect to the mom nodes via the network nodes. The communication problem is easy to detect by checking the server and mom logs.

The Torque configuration and installation remains the same on external nodes as for the Cray. The installation was completed on the Bright management server since an external login node was used for the pbs_server node. Several configuration options are available and should be evailuated for your site. If you have previously iinstalled Torque the same configuration parameters will likely work with the exception of the –with-default-server which should be the name of the external node. One additional configuration option is now available: CFLAGS="-D CRAY_MOAB_PASSTHRU". This allows use of the "nodes=<num nodes>" notation vs mpp*. Our typical configuration:

```
./configure --prefix=/opt/torque/4.1.4 --with-server-home=/var/spool/torque --with-default-server=edison06 --enable-syslog --disable-gcc-warnings --enable-maxdefault --with-modulefiles=/opt/modulefiles
```

The server need to know where the alps_reporter and the mom nodes are. The mom_nodes are called alps_login which was initially slightly confusing. Although some sites may still use internal login nodes as mom nodes but that hasn't been the case at our site for a while. Adaptive recommends you use the sdb for your alps_reporter mom. This is the special mom that handles communications with Alps. If Torque and Moab will continue to run on the sdb vs externally, the alps_reporter mom will be required with version 4.1 of Torque and 7.0 of Moab. In that case both the pbs_server and pbs_mom will run on the sdb. The pbs_server gets the information about these nodes though the <server-home>/server_priv/nodes file. The nodes file will look something like this:

```
sdb alps_reporter

nid00004 np=X alps_login
nid00005 np=X alps_login
nid00006 np=X alps_login
…
nid00xxx np=X alps_login
```

Verify the <server-home>/server_name file contains the name of the pbs_server host. Some parameters that need to be added to the pbs_server via the qmgr:

```
qmgr –c "set server cray_enabled=true"

qmgr –c "set server resources_default.partition=<clustername>"
```

where clustername is what you call the compute pool. Adaptive also has a new authentication mechanism. Previously it was via pbs_iff and is now a daemon called trqauthd. This must be run on any node that torque needs to communicate with. This includes any acl or submission hosts. At our site this is the external login nodes, the sdb, and the mom nodes.

To ensure all the parts start at boot time you will need to configure and install startup scripts. These are provided in the Torque source code in the contrib/init.d directory. Always verify these are configured correctly for your site before copying to /etc/init.d.

in /var/spool/torque/server_logs/YYYYMMDD on the server node and the mom logs in /var/spool/torque/mom_logs/YYYYMMDD on the mom nodes. On the mom nodes a different server name in the /var/spool/torque/server_name file allowed this configuration to work. The nid names of the mom nodes will most likely need to be added to your /etc/hosts file as the first name.

You may need to set the torque qmgr server parameter keep_completed = 300 for Moab to work correctly. Moab wasn't able to clear jobs if this was not set.

### *B. Mom Nodes and SDB*

The next step is to install torque on the sdb and the mom nodes. If the architecture on the external node is the same as within the Cray you can use the packages created by "make packages" in Torque to install on the boot node. If the architecture is different you will have to re-configure and install torque on the boot node. On the sdb node some modifications to the /var/spool/torque/mom_priv/config file are needed for the new configuration. The sdb should be defined as the alps_reporter and the apbasil protocol and path may need to be set:

```
$reporter_mom true
$apbasil_path <path to apbasil>  *default is /usr/bin/apbasil
$apbasil_protocol 1.2         * default is 1.0
```

On the mom nodes the config file needs a few additions but should otherwise be fine with the options used prior to this version. The additions are:

```
$alps_login true
$apbasil_path <path to apbasil>  *default is /usr/bin/apbasil
$apbasil_protocol 1.2         * default is 1.0
```

Then you can start the trqauthd and pbs_mom on each of the mom nodes and the sdb. Verify you can access the pbs server node from the mom nodes. Check the log files on the mom nodes and the server node to ensure there are no errors. The server_name on the mom nodes on our server uses an interface name and is working (instead of edison06 it is edison06-eth4).

### *C. Moab*

Next configure and install moab on the pbs server node. No special config options are necessary for the build. The mom nodes need to be identified as a separate partition to the moab.cfg file so jobs don't get scheduled on them. The additions to the moab.cfg file look like this:

```
NODECFG[nid0000x] Partition=login
NODECFG[nid0000x] Partition=login
…
NODECFG[nid0000x] Partition=login
```

Once it is installed and configured there it will need to be installed on the sdb and mom nodes. Only the moab.cfg needs to be in the /var/spool/moab directory.

## V. FAIRSHARE

Sites like NERSC have hundreds of users from large groups of projects contending for limited resources daily. Under these circumstances, the right scheduling policies that meet both the allocation obligations and users needs is a challenge. This can be especially difficult because the concept of scheduling fairness varies widely from person to person and site to site.

MOAB provides a wide range of tools to assist site managers in configuring scheduling to meet those diverse requirements. Those tools include:

- Job Prioritization
- Throttling Policies
- Allocation Management
- Quality of Service
- Standing Reservations
- Class/Queue Constraints
- Fairshare

NERSC was using the first six methods until recently, motivated by rather positive experiences with Fairshare based scheduling in another scheduler on other resources at NERSC, a decision was made to explore adding the Fairshare component into the mix for Torque/Moab on Edison. Since Edison is not included in the allocation pool yet it was possible to modify scheduling without impacting production commitments.

Moab has a rather transparent priority-weighting way of including the Fairshare component in the priority calculation. Contribution of each priority subcomponent is calculated as: <COMPONENT WEIGHT> * <SUBCOMPONENT WEIGHT> * <PRIORITY SUBCOMPONENT VALUE>. This is then weighted over a time component.

The Fairshare contribution consists of five subcomponents: ACCOUNT, CLASS, GROUP, USER, and QOS credentials. The weights for both Fairshare and the subcomponents are set independently in the MOAB configuration file. The PRIORITY SUBCOMPONENT VALUE is calculated based on utilization in intervals which depth (FSDEPTH) and width (FSINTERVAL) is defined by the administrator. Contribution of intervals decreases with FSDECAY factor over time. Utilization accounting can be done in several ways (FSPOLICY):

| | |
|---|---|
| **DEDICATEDPES** | Usage tracked by processor-equivalent seconds dedicated to each job |
| **DEDICATEDPS** | Usage tracked by processor seconds dedicated to each job |
| **UTILIZEDPS** | Usage tracked by processor seconds used by each job |

At NERSC we set up Fairshare data collecting with the following values:

- FSDEPTH = 8
- FSINTERVAL = 12:00:00
- FSDECAY = 0.8

Since NERSC does not facilitate node sharing, we decided to set the FSPOLICY to DEDICATEDPS, following the recommendations in Adaptive Computing documentation.

At first we ran for a period of time with the FSWEIGHT not set. This allowed us to track the <PRIORITY SUBCOMPONENT VALUE> and estimate its impact on priority, without actually impacting the priority itself. Lacking a better tool we used the mdiag –v –f command to display data collected by MOAB. After the initial period dedicated to just tracking utilization, we set set targets per ACCOUNT (known also as "repo" at NERSC) to meet our obligations towards DARPA (25%).

However, altogether NERSC supports over 700 repos, so to avoid calculating and setting very small shares per repo we briefly set the default to 10%. Since this was hardly fair we also started working towards implementing a more structured fairshare tree. Total available resource was divided between DARPA and the rest of NERSC (25:75). The 75% allocated to the "non DARPA" was then subdivided between 6 Department of Energy (DOE) offices and a small fraction given to staff for benchmarking, development and evaluation. Shares given to offices reflected their annual NERSC allocations. Each office branch incorporates associated repos as members.

The resulting tree is shown below:

- FSTREE[root] SHARES=100 MEMBERLIST=darpa,nersc
- FSTREE[darpa] SHARES=25 MEMBERLIST=acct:darpa
- FSTREE[nersc] SHARES=75 MEMBERLIST=ascr,ber,bes,ccc,fes,hep,np
- FSTREE[ascr] SHARES=5 MEMBERLIST=acct:m888,acct:m945, +85 more
- FSTREE[ber] SHARES=17 MEMBERLIST=acct:m917,acct:m950, +126 more
- FSTREE[bes] SHARES=30 MEMBERLIST=acct:m881,acct:m894, +303 more
- FSTREE[ccc] SHARES=7 MEMBERLIST=acct:mpesnet,acct:mpccc, +10 more
- FSTREE[fes] SHARES=17 MEMBERLIST=acct:m908,acct:m916, +63 more
- FSTREE[hep] SHARES=13 MEMBERLIST=acct:m981,acct:m1067, +63 more
- FSTREE[np] SHARES=11 MEMBERLIST=acct:m1401,acct:m327, +44 more

However the switch from a flat distribution of shares among accounts to this more structured tree resulted in the fair share contribution disappearing altogether from the priority calculations. This has been traced to default setting of FSACCOUNTWEIGHT=1000 in the absence of the shares tree and 0 with the tree. Once FSACCOUNTWEIGHT was explicitly set (to 100 in our case) in the MOAB configuration file the fairshare component re-appeared in the priority listings (mdiag –p).

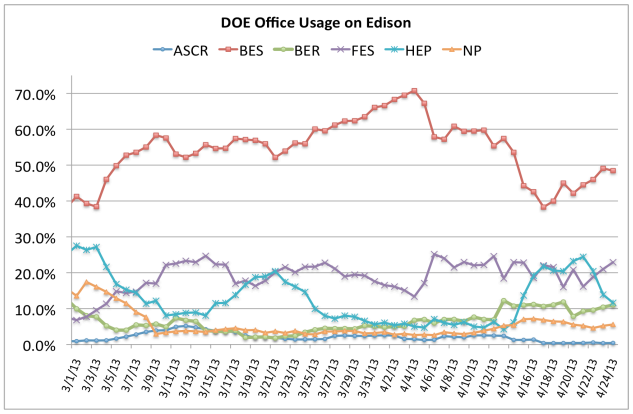

Figure 2. Fairshare Usage by DOE Office

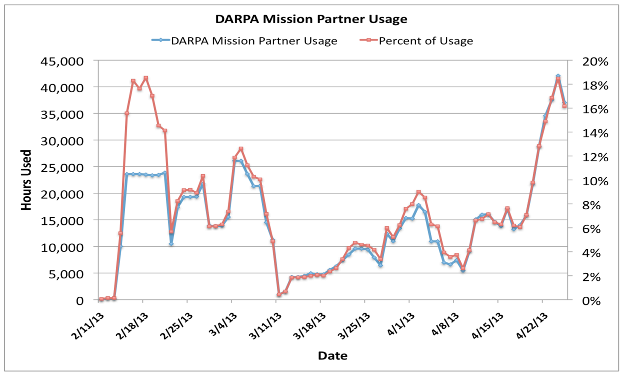

Figure 3. DARPA Fairshare Usage

Relative use between offices shown in Fig 2. reflects history of the aforementioned share changes. Fig 3 shows DARPA usage. When we introduced shares into the mix of priority calculations on March 1st all the non-DARPA accounts were assigned a target share of 10% thus the usage per DOE office was proportional to number of repos assigned to the Offices. We started collecting share usage in February, thus initially HEP access was suppressed by the CPU accumulated from previously run jobs. However, towards the middle of March FES and HEP offices with very similar number of repos became equal. On March 16th in order to test impact of share target settings we significantly boosted targets for 2 repos from the BES office and the utilization followed. We switched data collection from a flat ACCOUNT shares to a tree structure on April 1st but due to the error mentioned earlier that was corrected on April 9th the effect of that change became only visible on that later date.

There are additional facts worth noticing. When an entity with assigned share is not running, there is a very strong effect favoring its priority once jobs are submitted (the HEP Office around April 13th). Another aspect of the shares mechanism is that when there were no jobs from one of the offices to utilize its share, those with larger allocation profited more from this situation (BES would be boosted more than others).

In addition to the goal of fairly managing the allocation scheduling configuration has additional requirements:

- The queue wait time should make a difference
- A single user shouldn't be able to take over most of the system
- A single user should not use all shares for the repo
- We should be able to favor large jobs (or at least not discriminate against)
- We need to maintain the ability to favor individuals or repos when necessary.

We are at the beginning of exploring the fairshare configuration. There exists a plethora of options (utilization can be taken into account as a target, floor or celling; shares could be evaluated as a fraction of delivered or available time, etc). In addition to grouping utilization per ACCOUNT (repo) we can do it by QoS (which would enable us to require that a certain share of available resource will be allocated to large jobs). In order to cope with exploration of this multi-dimensional space we plan to harvest current workloads and run MOAB in simulation mode.

## REFERENCES

[1] http://docs.adaptivecomputing.com/mwm/help.htm#xtinstall.html

[2] Moab Workload Manager7.2.2Administrator Guide, 2013 Adaptive Computing

[3] TORQUE Administrator Guide, version 4.2.2, 2013 Adaptive Computing.