1 Obtaining and compiling TORQUE and Maui

Obtaining and compiling TORQUE

TORQUE Source Code

TORQUE is available from www.clusterresources.com

Building TORQUE

- `configure --prefix=/whatever/you/likemakemake install`

- not very clean, actually: quite a lot of important files go into `/var/spool` — including configuration files!

You can build only the server or MOM components, just tell `--disable-mom` or `--disable-server`

My favorite install uses a directory that is shared among the masternode and the computing nodes, so that I need to build only once.

TORQUE install uses 700 permission for `pbs_mom`, so you need to

- share the install directory with `no_root_squash` or

- `chgrp nfsnobody pbs_mom ; chmod 710 pbs_mom`

Obtaining and compiling Maui

Maui Source Code

Maui too is available from www.clusterresources.com

You need to register to their site to download the code, and they may contact you later and ask what are you going to do with their software (and offer commercial support for it)
Building Maui

- same `configure; make; make install`
- maui build system need to know where TORQUE has been installed
- again, important files go into `/var/spool`

2 Configuration

TORQUE Configuration

TORQUE Common Configuration Files

- `pbs_environment` contains the environment variables for TORQUE; any minimal set will do e.g.

  ```
  PATH=/bin:/usr/bin
  LANG=en_US
  ```

- `server_name` contains the «official» name of the machine where `pbs_server` runs (this is usually your master node) The server name must be identical to the FQDN e.g.

  ```
  cerbero.hpc.sissa.it
  ```

  Both these files reside in the spool directory (`/var/spool/torque`)

`pbs_server` configuration

The `nodes` file

`server_priv/nodes` contains the list of available computing nodes and a list of attributes for each node.

<table>
<thead>
<tr>
<th>node name</th>
<th># of CPUs</th>
<th>«features»</th>
</tr>
</thead>
<tbody>
<tr>
<td>node01</td>
<td>np=2</td>
<td>opteron myri</td>
</tr>
<tr>
<td>node02</td>
<td>np=2</td>
<td>opteron myri</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>node51</td>
<td>np=4</td>
<td>opteron IB</td>
</tr>
<tr>
<td>node52</td>
<td>np=4</td>
<td>opteron IB</td>
</tr>
</tbody>
</table>
**pbs_server** configuration Creating the Configuration Database

The bulk of **pbs_server** configuration is written in a (binary) database. You first need to create the empty database with

```bash
pbs_server -t create
```

This will destroy any existing configuration, create the empty database and start a **pbs_server**.

Configuration can then be edited using the **qmgr** tool. Configuration data are written to `server_priv/serverdb` as well as in various other files.

- if you are running postfix, you already have a **qmgr** somewhere in your system, so you may need to adjust some paths
- **TORQUE** **qmgr** needs a running **pbs_server** to actually write the configuration; this is because the configuration database is written by the server itself, and this in turn means that the server needs to have write permission on its own configuration

**pbs_server** configuration Security Note

**qmgr** doesn’t actually edit the configuration database. It only sends configuration commands to **pbs_server** which in turn writes the configuration.

This means that:

- you need a running **pbs_server** to use **qmgr** (no big issue)
- the **pbs_server** process needs write access to its own configuration files this is usually considered very bad in any security-conscious environment – unfortunately no easy workarounds are available

**pbs_server** configuration Sample Configuration

```bash
[root@borg]# qmgr
Qmgr: create queue batch
Qmgr: set queue batch queue_type = Execution
Qmgr: set queue batch resources_max.walltime = 01:00:00
Qmgr: set queue batch resources_default.nodes = 1
Qmgr: set queue batch resources_default.walltime = 00:01:00
Qmgr: set queue batch enabled = True
Qmgr: set queue batch started = True
Qmgr: set server managers = maui@borg.cluster
Qmgr: set server managers += root@borg.cluster
Qmgr: set server operators = maui@borg.cluster
Qmgr: set server operators += root@borg.cluster
```

One of the most common configuration issues, that prevents the batch system from running any job, involves missing or incorrect `set server managers` and/or `set server operators` lines.

**pbs_mom** configuration

**pbs_mom** configuration can be fairly minimal, the only thing the Mom needs to know is the hostname where **pbs_server** is running on.

Useful additions include log configuration, how to handle user file copy and which filesystem to monitor for available space.

```bash
mom_priv/config:
$clienthost master.hpc
$logevent 0x7f size[fs=/local_scratch]
$usecp *:/home /home
```
Maui Configuration

Maui Configuration How to Connect to Resource Manager

- simpler approach: a single configuration file (maui.cfg)
- Maui needs to know what RM to connect to and how

        SERVERHOST borg.cluster
        RMCFG[BORG.CLUSTER] TYPE=PBS
        RMPOLLINTERVAL 00:00:30
        SERVERPORT 42559
        SERVERMODE NORMAL
        ADMIN1 root

SERVERHOST is the same we defined for TORQUE.
User(s) listed as ADMIN1 have full control over Maui. The first user in the list must be used to run Maui itself; if you want to run Maui as a non-privileged user, put this username here. The user Maui is running as needs to be able to control pbs_server.

Maui Configuration Job Prioritization

Job priority is recomputed at each scheduler iteration, according to site-defined parameters. If no parameters are set only queue time is taken into account, i.e. the scheduling is strictly FIFO.

Priority components include:

- **Queue Time**: how long the job has been idle in the queue
- **Credentials**: a static priority can be assigned on a user, group, queue basis
- **Fair Share**: historical usage data
- **Resources** requested for the job

Maui Configuration Job Prioritization: Queue Time and Credentials

        QUEUETIMEWEIGHT 1
        XFACTORWEIGHT 10
        CLASSCFG[batch] PRIORITY=1
        CLASSCFG[fast] PRIORITY=1000
        GROUPCFG[guests] PRIORITY=1
        GROUPCFG[users] PRIORITY=1000
        GROUPCFG[devel] PRIORITY=10000
        USERCFG[DEFAULT] PRIORITY=2000
        USERCFG[luser1] PRIORITY=0

- a high QUEUETIMEWEIGHT makes the scheduling «more FIFO»
- \( XFactor = \frac{QueueTime + WallClockLimit}{WallClockLimit} \) so a high XFACTORWEIGHT favors «short» jobs; this usually makes users happy in the short term, but can degrade overall cluster performance

Maui Configuration Job Prioritization: Fair Share

The FS priority component must be explicitly enabled by setting its weight to a non-zero value.

        FSINTERVAL  86400 duration of each FS window
        FSDEPTH  30 number of FS windows
        FSDECY  0.90 decay factor applied to older FS windows
        FSWEIGHT 1
        FSGROUPWEIGHT 240
        FSUSERWEIGHT 10
Maui Configuration Job Prioritization: Fair Share

Usage targets can be set on a per-user, per-group and per-queue basis.

\[
\text{USERCFG[DEFAULT]} \quad \text{FSTARGET}=1 \\
\text{GROUPCFG[users]} \quad \text{FSTARGET}=30 \\
\text{GROUPCFG[devel]} \quad \text{FSTARGET}=40
\]

You can also set FS floors or caps so that priority is affected only when usage drops below the floor or goes above the cap:

\[
\text{GROUPCFG[guests]} \quad \text{FSTARGET}=5- \quad \text{give a negative priority component if usage is above 5%} \\
\text{USERCFG[master]} \quad \text{FSTARGET}=20+ \quad \text{give a priority boost if usage is below 20%}
\]

Prologue and Epilogue

Prologue & Epilogue scripts

\text{pbs_mom} looks for scripts in its configuration directory \text{mom_priv}. If found, the \text{prologue} script is executed just before job start and the \text{epilogue} script at job termination.

The \text{prologue} script performs any initialization that is required on the node for the job to run, while the \text{epilogue} undoes the modifications.

\[
\text{/etc/security/access.conf}
\]

before prologue

\[-:ALL EXCEPT root:ALL\]

disallows login to everybody except root, from anywhere

after prologue

\[-:ALL EXCEPT root someuser:ALL\]

now allows someuser to login

3 Diagnostics & Troubleshooting

TORQUE Diagnostics

\text{momctl}

Query and control remote \text{pbs_mom}:

\[
\text{# momctl \ -d3 \ -h \ i602}
\]

\text{Host: i602/i602.hpc Server: master.hpc Version: 1.2.0p6}

\text{HomeDirectory: /var/spool/PBS/mom_priv}

\text{MOM active: 6907718 seconds}

\text{Last Msg From Server: 213582 seconds (DeleteJob)}

\text{Last Msg To Server: 1 seconds}

\text{Server Update Interval: 45 seconds}

\text{Init Msgs Received: 10 hellos/2 cluster-addr}

\text{Init Msgs Sent: 190 hellos}

\text{LOGLEVEL: 0 (use SIGUSR1/SIGUSR2 to adjust)}

\text{Communication Model: RPP}

\text{TCP Timeout: 20 seconds}

\text{Prolog Alarm Time: 300 seconds}

\text{Alarm Time: 0 of 10 seconds}

\text{Trusted Client List: ...}

\text{JobList: NONE}

\text{diagnostics complete}

Maui Diagnostics

\text{checknode}

Check who is doing what on a node and show node capabilities

\[
\text{# checknode a034}
\]

\text{checking node a034 (in current state for 1:13:38:12)}

\text{State: Busy (in current state for 1:13:38:12)}

\text{Configured Resources: PROCs: 2 MEM: 3949M SWAP: 7242M DISK: 59G}

\text{Utilized Resources: PROCs: 2 DISK: 10G}

\text{Dedicated Resources: PROCs: 2}

\text{Opsys: DEFAULT Arch: [NONE]}

\text{Speed: 1.00 Load: 2.000 (ProcSpeed: 2600)}

\text{Network: [DEFAULT]}

\text{Features: [myri][opteron][opteron-sc]...}
Attributes: [Batch]
Classes: [smp2 2:2][smp4 2:2][mpi4 0:2][mpi# 2:2]...
Total Time: 25:14:33:36 Active: 25:04:53:26 (98.43%)
Reservations:
Job '30069'(x2) -1:13:38:44 -> 2:10:20:16 (3:23:59:00)
JobList: 30069