TECHNICAL REPORT

RESOURCE INFORMATION AND MANAGEMENT ON HYDRA

Anish Parwage\textsuperscript{1, a}

\textsuperscript{1} Indian Institute of Astrophysics, 2\textsuperscript{nd} Block Koramangala, Bangalore -560034
\textsuperscript{a} anish@iiap.res.in

Keywords: HPC, Job scheduler, Resource Manager, Scheduling, TORQUE, Maui, PBS, OpenPBS, PBS Professional, Queue, INTEL, PGI, ifort, fortran compiler.

Abstract: TORQUE-3.0.2 with Maui-3.3.1 is used as Resource Manager for HYDRA and performs job scheduling in cluster environment. It looks into the resource requirement for a job and provides the requested resources for the job to run on the cluster.

\textit{Fortran compilers are available freely from GNU for e.g. gfortran. But there are also proprietary compilers available in the market such as from INTEL, PGI etc.}

\textit{HYDRA has Intel’s ifort fortran compiler which is best suited for Intel architecture. There are two versions of INTEL’s ifort fortran compiler installed on HYDRA.}

Introduction:

In addition to the computational servers KASPAR (GPU Node), CYGNUS and FORNAX, IIA Data center also houses a 20(Compute) +1 (Master) node HPC cluster named HYDRA. The entire HPC cluster has a theoretical peak performance of 3.3 TF. The specifications of the nodes are as followed:

Master Node: It consists of dual Xeon X5675, 6 cores per processor based on 32nm technology. It has a clock speed of 3.06 GHz and 3.46 GHz on Turbo boost. The Master Node has 8 GB DDR3 memory per core amounting to a total of 96 GB with a speed of 1333 MHz.
Compute Nodes: All the compute nodes have the same specification as the Master node. The associated storage with cluster is 40 TB based on RAID 5 technology. The interconnection network used for the cluster is 40Gb/s QDR Infinband.

The HPC cluster is used to cater to the computation needs of IIA users who use it to submit a mixture of serial and parallel jobs. TORQUE 3.0.2 along with Maui 3.3.1 is used as job scheduler for submitting jobs onto the cluster.

**Importance of Job Scheduler:**

Job Scheduling is an important aspect of High Performance Cluster Computing. Scheduling of jobs in a cluster environment involves allocation of computational tasks among the available computing resources. Portable Batch System (PBS) is the name of the software that performs job scheduling in cluster environment. Scheduler makes life easier for users by taking care of the task of looking for available resources and allowing submission of jobs. Users need not worry if the resources are not available, the scheduler will keep the job in queue and submit it on behalf of the users when the resources requested by the job are available.

**Portable Batch System:**

PBS was originally developed for NASA by MRJ Technology Solutions under a contract project that began on June 17, 1991. MRJ was acquired by Veridian in the late 1990s. Altair Engineering acquired the rights to all the PBS technology and intellectual property from Veridian in 2003. The following versions of PBS are currently available:

- **OpenPBS** — original open source version released by MRJ in 1998 (not actively developed)

- **TORQUE** — a fork of OpenPBS that is maintained by Adaptive Computing Enterprises, Inc. (formerly Cluster Resources, Inc.)

- **PBS Professional** (PBS Pro) — commercial version of PBS by Altair Engineering
Both OpenPBS and TORQUE are open source software. TORQUE provides enhancements over standard OpenPBS in the following areas:

1. **Fault Tolerance**
   - Additional failure conditions checked/handled
   - Node health check script support

2. **Scheduling Interface**
   - Extended query interface providing the scheduler with additional and more accurate information
   - Extended control interface allowing the scheduler increased control over job behavior and attributes
   - Allows the collection of statistics for completed jobs

3. **Scalability**
   - Significantly improved server to MOM communication model
   - Ability to handle larger clusters (over 15 TF/2,500 processors)
   - Ability to handle larger jobs (over 2000 processors)
   - Ability to support larger server messages

4. **Usability**
   - Extensive logging additions
   - More human readable logging

**TORQUE** is an abbreviation for Terascale Open-source Resource and QUEue Manager which provides control over batch jobs and distributed compute nodes.

A TORQUE cluster consists of one head node and many compute nodes. The head node runs the **pbs_server** daemon and the compute nodes run the **pbs_mom** daemon. Client commands for submitting and managing jobs can be installed on any host (including hosts not running **pbs_server** or **pbs_mom**).
The head node also runs a scheduler daemon. The scheduler interacts with pbs_server to make local policy decisions for resource usage and allocate nodes to jobs. A simple FIFO scheduler, and code to construct more advanced schedulers, is provided in the TORQUE source distribution. Most TORQUE users choose to use a packaged, advanced scheduler such as Maui or Moab.

Users submit jobs to **pbs_server** using the **qsub** command. When **pbs_server** receives a new job, it informs the scheduler. When the scheduler finds nodes for the job, it sends instructions to run the job with the node list to **pbs_server**. Then, **pbs_server** sends the new job to the first node in the node list and instructs it to launch the job. This node is designated the execution host and is called *Mother Superior*. Other nodes in a job are called *sister moms*.

TORQUE decides whether to allow a job to run or not depending upon the availability of consumable resources (processor, memory etc.) defined in the scheduler. From the time HYDRA was made operational, memory was never made a compulsory argument to be passed on to the scheduler to make policy based decision as a result when a user did not use to define memory as a consumable parameter in his submission script, the scheduler would leave memory out of consideration. It created problems for other jobs.

Consider that a job has been submitted without specifying the memory requirement for it. TORQUE will submit the job in the requested number of nodes without having an actual account of the memory reserved for the job on the requested nodes for the scheduler to take policy based decision for other jobs. This will create problems for other jobs as they will again be submitted on the already occupied nodes without the scheduler having an account of the memory consumed on those nodes.

In HYDRA, TORQUE has two routing queues namely:

1. **Parallel Queue**: For jobs requesting more than one processor.
2. **Serial Queue**: For sequential jobs requesting only one processor.

Since the time both the queues were created, they did not having any minimum memory restrictions assigned to them.
So, the initial TORQUE configuration looked like:

```
$ qmgr -c "p s"
#
# Create queues and set their attributes.
#
# Create and define queue batch
#
create queue batch
set queue batch queue_type = Route
set queue batch resources_default.nodes = 1
set queue batch resources_default.walltime = 01:00:00
set queue batch route_destinations = serial
set queue batch route_destinations += parallel
set queue batch enabled = True
set queue batch started = True
#
# Create and define queue parallel
#
create queue parallel
set queue parallel queue_type = Execution
set queue parallel max_user_queuable = 4
set queue parallel resources_min.procct = 2
set queue parallel resources_default.walltime = 01:00:00
set queue parallel max_user_run = 4
set queue parallel enabled = True
set queue parallel started = True
#
# Create and define queue serial
#
create queue serial
```
set queue serial queue_type = Execution
set queue serial max_user_queuable = 24
set queue serial resources_max.procct = 1
set queue serial resources_default.walltime = 01:00:00
set queue serial max_user_run = 24
set queue serial enabled = True
set queue serial started = True
#
# Set server attributes.
#
set server scheduling = True
set server acl_hosts = hydra.iiap.res.in
set server acl_hosts += localhost
set server managers = root@hydra.iiap.res.in
set server operators = root@hydra.iiap.res.in
set server default_queue = batch
set server log_events = 511
set server mail_from = adm
set server query_other_jobs = True
set server resources_available.nodect = 480
set server scheduler_iteration = 600
set server node_check_rate = 150
set server tcp_timeout = 6
set server mom_job_sync = True
set server keep_completed = 300
set server next_job_number = 43639
So with the initial TORQUE configuration, when a job was submitted to the queue without memory requirement being specified for the job, the TORQUE job queue looked like:

$ qstat -a

hydra.iiap.res.in:

<table>
<thead>
<tr>
<th>Job ID</th>
<th>Username</th>
<th>Queue</th>
<th>Jobname</th>
<th>SessID</th>
<th>NDS</th>
<th>TSK</th>
<th>Memory</th>
<th>Time</th>
<th>S Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>40270.hydra.iiap</td>
<td>sampoorn</td>
<td>serial</td>
<td>job-ad-hanle-iso</td>
<td>6654</td>
<td>1</td>
<td>1</td>
<td>20gb</td>
<td>10000</td>
<td>R 1201:6</td>
</tr>
<tr>
<td>43323.hydra.iiap</td>
<td>manpreet</td>
<td>serial</td>
<td>submit1.sh</td>
<td>3401</td>
<td>1</td>
<td>1</td>
<td>12000m</td>
<td>3600</td>
<td>R 192:5</td>
</tr>
<tr>
<td>43324.hydra.iiap</td>
<td>manpreet</td>
<td>serial</td>
<td>submit1.sh</td>
<td>28542</td>
<td>1</td>
<td>1</td>
<td>12000m</td>
<td>3600</td>
<td>R 192:4</td>
</tr>
<tr>
<td>43325.hydra.iiap</td>
<td>manpreet</td>
<td>serial</td>
<td>submit1.sh</td>
<td>28604</td>
<td>1</td>
<td>1</td>
<td>12000m</td>
<td>3600</td>
<td>R 192:4</td>
</tr>
<tr>
<td>43459.hydra.iiap</td>
<td>das</td>
<td>serial</td>
<td>submit1.sh</td>
<td>22017</td>
<td>1</td>
<td>1</td>
<td>12000m</td>
<td>3600</td>
<td>R 153:4</td>
</tr>
<tr>
<td>43461.hydra.iiap</td>
<td>das</td>
<td>serial</td>
<td>submit1.sh</td>
<td>15954</td>
<td>1</td>
<td>1</td>
<td>12000m</td>
<td>3600</td>
<td>R 153:4</td>
</tr>
<tr>
<td>43462.hydra.iiap</td>
<td>das</td>
<td>serial</td>
<td>submit1.sh</td>
<td>16032</td>
<td>1</td>
<td>1</td>
<td>12000m</td>
<td>3600</td>
<td>R 153:4</td>
</tr>
<tr>
<td>43527.hydra.iiap</td>
<td>srinivasa</td>
<td>serial</td>
<td>submit1.sh</td>
<td>16895</td>
<td>1</td>
<td>1</td>
<td>-----</td>
<td>3600</td>
<td>R 147:1</td>
</tr>
</tbody>
</table>

As shown in the above output, since user “srinivasa” has not defined any memory requirement in his job submission script, the memory field was left blank in the execution queue. This created a lot of problems for other jobs which were also submitted on the nodes that were already running jobs without any memory accountability. The aftermath being the jobs in that particular node used to get killed at random and showed erratic behavior as well.

As a result, I had to make the following changes to take effect in TORQUE so that memory is also considered as a consumable resource when making job submission decisions.
Adding the following settings to the earlier configuration elucidates the queue resource requirements:

FOR SERIAL QUEUE

```
qmgr -c "set queue serial resources_default.mem = 5000mb"
```

FOR PARALLEL QUEUE

```
qmgr -c "set queue parallel resources_default.mem = 5000mb"
```

The time of enforcement of server and queue defaults is important. TORQUE applies server and queue defaults differently in job centric and queue centric modes. For job centric mode, TORQUE waits to apply the server and queue defaults until the job is assigned to its final execution queue. For queue centric mode, it enforces server defaults before it is placed in the routing queue. In either mode, queue defaults override the server defaults. TORQUE defaults to job centric mode. To set queue centric mode, the queue_centric_limits were defined as:

```
qmgr -c "set server queue_centric_limits = True"
```

An artifact of job centric mode is that if a job does not have an attribute set, the server and routing queue defaults are not applied when the queue resource limits are checked. So, after setting the queue centric limits to “TRUE”, if a job does not request memory it will be placed in the respective queue and the resources_default.mem will be applied after the job is assigned to an execution queue.
So, after these changes were made by me, the TORQUE configuration looked like:

```bash
$ qmgr -c "p s"
#
# Create queues and set their attributes.
#
# Create and define queue batch
#
create queue batch
set queue batch queue_type = Route
set queue batch resources_default.nodes = 1
set queue batch resources_default.walltime = 01:00:00
set queue batch route_destinations = serial
set queue batch route_destinations += parallel
set queue batch enabled = True
set queue batch started = True
#
#
# Create and define queue parallel
#
create queue parallel
set queue parallel queue_type = Execution
set queue parallel max_user_queuable = 4
set queue parallel resources_min.procct = 2
set queue parallel resources_default.mem = 5000mb
set queue parallel resources_default.walltime = 01:00:00
set queue parallel max_user_run = 4
set queue parallel enabled = True
set queue parallel started = True
#
#
# Create and define queue serial
```
#
create queue serial
set queue serial queue_type = Execution
set queue serial max_user_queuable = 24
set queue serial resources_max.procct = 1
set queue serial resources_default.mem = 5000mb
set queue serial resources_default.walltime = 01:00:00
set queue serial max_user_run = 24
set queue serial enabled = True
set queue serial started = True
#
# Set server attributes.
#
set server scheduling = True
set server acl_hosts = hydra.iiap.res.in
set server acl_hosts += localhost
set server managers = root@hydra.iiap.res.in
set server operators = root@hydra.iiap.res.in
set server default_queue = batch
set server log_events = 511
set server mail_from = adm
set server query_other_jobs = True
set server resources_available.nodect = 480
set server scheduler_iteration = 600
set server node_check_rate = 150
set server tcp_timeout = 6
set server queue_centric_limits = True
set server mom_job_sync = True
set server keep_completed = 300
set server next_job_number = 43639
After I had made the necessary changes to TORQUE, when a job was submitted to the queue without memory requirement being specified for the job, then the resources_default.mem i.e. 5000MB or 5GB was applied to job and the TORQUE job queue looked like:

$ qstat -a

tydra.iiap.res.in:

<table>
<thead>
<tr>
<th>Job ID</th>
<th>Username</th>
<th>Queue</th>
<th>Jobname</th>
<th>SessID</th>
<th>NDS</th>
<th>TSK</th>
<th>Memory</th>
<th>Time</th>
<th>S Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>40270.hydra.iiap</td>
<td>sampoorn</td>
<td>serial</td>
<td>job-ad-hanle-iso</td>
<td>6654</td>
<td>1</td>
<td>1</td>
<td>20gb</td>
<td>10000</td>
<td>R 1201:6</td>
</tr>
<tr>
<td>43323.hydra.iiap</td>
<td>manpreet</td>
<td>serial</td>
<td>submit1.sh</td>
<td>3401</td>
<td>1</td>
<td>1</td>
<td>12000m</td>
<td>3600:R 192:5</td>
<td></td>
</tr>
<tr>
<td>43324.hydra.iiap</td>
<td>manpreet</td>
<td>serial</td>
<td>submit1.sh</td>
<td>28542</td>
<td>1</td>
<td>1</td>
<td>12000m</td>
<td>3600:R 192:4</td>
<td></td>
</tr>
<tr>
<td>43325.hydra.iiap</td>
<td>manpreet</td>
<td>serial</td>
<td>submit1.sh</td>
<td>28604</td>
<td>1</td>
<td>1</td>
<td>12000m</td>
<td>3600:R 192:4</td>
<td></td>
</tr>
<tr>
<td>43459.hydra.iiap</td>
<td>das</td>
<td>serial</td>
<td>submit1.sh</td>
<td>22017</td>
<td>1</td>
<td>1</td>
<td>12000m</td>
<td>3600:R 153:4</td>
<td></td>
</tr>
<tr>
<td>43461.hydra.iiap</td>
<td>das</td>
<td>serial</td>
<td>submit1.sh</td>
<td>15954</td>
<td>1</td>
<td>1</td>
<td>12000m</td>
<td>3600:R 153:4</td>
<td></td>
</tr>
<tr>
<td>43462.hydra.iiap</td>
<td>das</td>
<td>serial</td>
<td>submit1.sh</td>
<td>16032</td>
<td>1</td>
<td>1</td>
<td>12000m</td>
<td>3600:R 153:4</td>
<td></td>
</tr>
<tr>
<td>43527.hydra.iiap</td>
<td>srinivasa</td>
<td>serial</td>
<td>submit1.sh</td>
<td>16895</td>
<td>1</td>
<td>1</td>
<td>5000m</td>
<td>3600:R 147:1</td>
<td></td>
</tr>
</tbody>
</table>

As shown in the above output, even when user “srinivasa” has not specified the memory requirement, 5000MB or 5GB has been automatically assigned to his job by the execution queue.
A second set of problem that was addressed by me for job submission with torque was the ease with which users can have knowledge of resource availability in the different nodes of the cluster and can redefine their requirement, so that they can run their job accordingly without waiting in the queue.

For this reason I had written a script named “resinfo” which gives the users an exact idea of resources (i.e. total, available and used resources) in each node of the cluster.

The output of the script goes like:

![Fig. 1](image-url)

The script “resinfo” takes 3 inputs i.e. no. of nodes, amount of memory requested and node list from “qstat” command of TORQUE and stores it in a file. It then calculates the memory requested per node required by each job as well as the processor requested per node by each job. The important thing about the script, that was taken care by me was that, it not only takes the resources requested by sequential but also OpenMP and MPI jobs as well.

This script developed by me, summarizes for the users which nodes are having what resources how much of these resources have been used and what is available for further use by them, so that they can specify their requirement accordingly.
A more detailed output regarding the information of each node is given by the script “pestat” provided by me. This script not only provides the total processor and memory available in each node but also the state of and load on each node along with the user name and the task id running on each node. It also provides the number of tasks running on each node per user.

![pestat output](image)

Fig. 2

As shown in the figure above “pestat” prints the node names along with their status i.e. whether it is offline (offl) or free or down or job exclusive (excl). It also displays the load along with total physical memory and processors on each node. No. of tasks per node is also shown along with the usernames whose jobs are running on those nodes with their respective job id’s. This script helps other users to check which user is running his/her jobs on which nodes as well as have an idea what all resources are available on each node.
There are 2 versions of ifort installed on HYDRA.

1. ifort 12.0.5
2. ifort 14.0.1

For users convenience I have developed a script called “ifort-selector-menu” with which they can select the ifort version that they would like to use and it will be automatically set in their environment. And the user only needs to open a new bash prompt or re-login to make the changes to take effect. But, at any given point of the time they can use only one of the two versions available.

The source code of the script written by me is shared below.

#!/bin/bash

#-----------------------------------------------------------
#
# Written By : ANISH PARWAGE
#
# Organisation : Indian Institute of Astrophysics
#
# Date : 23rd Dec 2013
#
# Aim : To select the version of ifort and set environment for user profile
#
#-----------------------------------------------------------

printf "Please select from the options available, the ifort version you want to use:
"

printf "1. To select ifort 12.0.5 press 1"

printf "\n2. To select ifort 14.0.1 press 2\n"

printf "\nQ. To Quit"

echo -e "\n"

read -p "Enter your choice : "  X

while [ "$X" = "" ];do
    echo -e "Selection can not be empty!!\n"
    printf "Please select from the options available:\n"
    read -p "Enter your choice : "  X

done

if [ "$X" = "1" ];then
    printf "\nYou have choosen ifort 12.0.5\n"
elseif [ "$X" = "2" ];then
    printf "\nYou have choosen ifort 14.0.1\n"
elseif [ "$X" = "Q" ];then
    printf "\nDo yo wanna QUIT ?\n"
fi

read -p "Do you want to proceed (Y/N): " Z

while [ "$Z" = "" ];do
echo -e "\nSelection can not be empty!!\n"

read -p "Do you want to proceed (Y/N): " Z

done

if [ \( -n "$Z" \) -a \( "$Z" = "Y" \) -o \( "$Z" = "y" \) ];then

case $X in

1) echo "ifort-12.0.5" > $HOME/.ifort-selector

   . "/etc/profile.d/izfort-selector.sh"

   echo -e "Please start a new shell for the changes
to take effect\n"

   ;;

2) echo "ifort-14.0.1" >$HOME/.ifort-selector

   . "/etc/profile.d/izfort-selector.sh"

   echo -e "Please start a new shell for the changes
to take effect\n"

   ;;

Q|q) printf "\n%5sYou are quitting!!\n\n"

   ;;
*) echo -e "\nYou have entered a wrong choice!!Try Again.\n\n"
esac

elif [ $( -n "$Z" ) -a ( "$Z" = "N" ) -o ( "$Z" = "n" ) ];then

    printf "%5sYou are quitting!!\n\n"
fi

By default the user will have ifort 12.0.5 set in his environment which he/she can check by using the command “ifort –v”. “ifort –v” will give the user the version of ifort set in his/her environment.
If the user wants to change the ifort he is using he can run the script “ifort-selector-menu”. The script “ifort-selector-menu” is made available on HYDRA by me as a command for all the users. So, when a user types the command “ifort-selector-menu” he/she will be prompted with choices as shown below:

![ifort-selector-menu](image)

**Fig. 4**

The user can select one of the ifort version that he/she wants to use as his/her choice of ifort.

For ifort 12.0.5 the user has to press 1 and for ifort 14.0.1 he/she has to press 2 or he/she can also press Q to quit if he/she does not want to disturb the current settings. He/She will have to choose any one of the option at any point of given time.
The user cannot leave the option blank or he/she will get an error message as shown below:

![Fig. 5](image1)

If the user tries to give an incorrect input as his/her choice, he/she will get an error message as shown below:

![Fig. 6](image2)
After entering the correct choice, the user will be asked to confirm his/her choice. Once confirmed, any changes made to the current settings will automatically be set for his/her environment and he/she just needs to re-login or open a new shell for the changes to take effect.

The user can now check if the settings have properly taken effect by issuing the command “ifort -v” as shown below.
**Conclusions:**

1. After the configuration changes were made by me to TORQUE it was observed that the issue of memory allocation to the jobs was resolved. If a job has not requested memory for its execution, a default memory of 5GB will be assigned to it and the scheduler will have a proper account of the resources being used by the running jobs.

2. Both the commands “resinfo” and “pestat” developed by me, give a very deep insight into each of nodes resources and were found to be very helpful by the users.

3. With “ifort-selector-menu” command which is written by me, users need not have to worry to make the necessary setting in their environment every time they want to use a different version of ifort. The script takes care of everything for them once the selection is confirmed.

**Acknowledgment:**
First I would like to express my profound gratitude to Prof. B.P. Das for his guidance and constant encouragement throughout.
I would also like to acknowledge the crucial role of Dr. Rekhesh Mohan and Mr. Fayaz for their friendly support and valuable suggestions.

**References:**

http://en.wikipedia.org/wiki/Portable_Batch_System

http://www.clusterresources.com/torquedocs21/1.1installation.shtml

http://www.clusterresources.com/torquedocs21/4.1queueconfig.shtml