HPC Cluster: Setup and Configuration HowTo Guide

A technical howto document presented to H3ABioNet

H3ABioNet

Pan African Bioinformatics Network for H3Africa

Created by

The System Administrator Task-force

Prepared for

The greater H3ABioNet and H3Africa Consortium community

Document Control

Date	Author	Authorization By	Version	Description
27 June 2014	Suresh Maslamoney	System Administrator Task-force	1.0	First draft

Contributors

Last Name	First Name	Institution	Country
Alibi	Mohamed	Pasteur Institute of Tunis (IPT)	Tunisia
Brown	David	Rhodes University (RU)	South Africa
Indome	David	Noguchi Memorial Institute for Medical Research (NMIMR)	Ghana
Scheepers	Inus	Centre for High Performance Computing (CHPC)	South Africa
Maslamoney	Suresh	Computational Biology Group – UCT (CBIO)	South Africa
Panji	Sumir	Computational Biology Group – UCT (CBIO)	South Africa
Van Heusden	Peter	South African National Bioinformatics (SANBI)	South Africa
Marcello	Lucio	(CIDRES)	Burkina Faso

Reviewers

Last Name	First Name	Institution	Country

Acronyms and Abbreviations

Acronym and Abbreviations	Description
CLI	The Command Line Interface refers to the actual local terminal on the Linux server used to navigate, configure and manage the system
NIC	A Network Interface Card is a physical network card installed the physical server
OS	A Operating System is a piece of software which is installed on a computer system and manages communication between the physical hardware and user based applications
SL	Scientific Linux Operating System

1. Basic Cluster Setup, Configuration and Management

1.1. Basic cluster setup and configuration

Within the bioinformatics field, many researchers make use of Linux cluster style environments to leverage this combined computing resources.

This section takes you through the process of configuring a basic cluster

1.1.1. Terminology:

Master node or head node: manages and schedules jobs. The term master node and head node will be used interchangeable throughout this document.

Slave node: are the actual worker nodes and this is where the actual jobs from the server will be computed.

1.1.2. Master node Setup Instructions:

On the master node, install Torque. Torque is installed on the master node or the head node as it is more commonly known. Torque is a queue management application. It is not strictly needed to use a cluster but it is highly recommended. All jobs to the cluster are submitted via this application. All management and statistics are generated via this application.

Installing Torque

Ubuntu / Debian

sudo apt-get install torque-server torque-scheduler torque-client

SL 6.4.

sudo yum –y install torque-server torque-scheduler torque-client

Check that PIDFILE is set to /var/spool/torque/sched_priv/sched.lock in /etc/init.d/torque-scheduler

Open /etc/hosts file and add the following line:

<your_ip_address> torqueserver

In /etc/hosts, comment out the line with 127.0.1.1 and add the slave nodes:

<node_ip_address> <node_name>

Example: 123.321.120.1node001

Kill the currently running pbs_server:

Type in the following command to get the process IDs:

ps -ef | grep pbs

Type in the following command to kill each process found:

sudo kill -9 <process ID>

Run the following command to start pbs_server in create mode:

sudo pbs_server -t create

Run the "qmgr" command to open qmgr in interactive mode and input the following lines to set up your first queue (values here will depend on your server i.e. how much memory it has cores, etc, and how you want to run it):

create queue batch
set queue batch queue_type=execution
set queue batch resources_default.nodes = 1
set queue batch resources_default.walltime = 01:00:00
set queue batch enabled = True
set queue batch started = True
set queue batch max_queuable=100
set queue batch max_running=40
set queue batch max_user_queuable=40
set queue batch max_user_run=10
set queue batch resources_available.nodect=2
set queue batch resources_available.nodes=10
set queue batch resources_available.ncpus=64
set queue batch resources_available.mem=384GB
set queue batch resources_default.nodect=2
set queue batch resources_default.nodes=1
set queue batch resources_default.ncpus=1
set queue batch resources_default.mem=1GB
set queue batch resources_max.nodect=2
set queue batch resources_max.nodes=1
set queue batch resources_max.ncpus=1
set queue batch resources_max.mem=16GB

Set server attributes

set server scheduling = True
set server managers = user@torqueserver
set server operators = user@torqueserver
set server default_queue = batch

set server log_events = 511
set server mail_from = adm
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

Our master node is now set up with one queue named "batch". Set up a second queue for power users:

create queue power
set queue power queue_type=execution
set queue power resources_default.nodes = 1
set queue power resources_default.walltime = 01:00:00
set queue power enabled = True
set queue power started = True

set queue power max_queuable=100
set queue power max_running=4
set queue power max_user_queuable=2
set queue power max_user_run=1
set queue power resources_available.nodect=2
set queue power resources_available.nodes=1
set queue power resources_available.ncpus=64
set queue power resources_available.mem=384GB
set queue power resources_default.nodect=2
set queue power resources_default.nodes=1
set queue power resources_default.ncpus=8
set queue power resources_default.mem=64GB
set queue power resources_max.nodect=2
set queue power resources_max.nodes=1
set queue power resources_max.ncpus=64
set queue power resources_max.mem=384GB

We now have a queue for users who want to run lots of small jobs (batch), and a queue for users who want to a run a single big job that requires lots of RAM and many cores (power). The last thing we need to do on the master node is define our compute nodes (let our master know which machines are its slaves). Open the file /var/spool/torque/server_priv/nodes for editing and add each slave on an individual line, e.g.:

node001 np=64

1.1.3. Slave node setup instructions

Install the required packages:

Ubuntu / Debian

sudo apt-get install torque-client torque-mom

SL 6.4.

sudo yum –y install torque-client torque-mom

set default server in /var/spool/torque/server_name to "torqueserver"

add torqueserver to /etc/hosts file

<master_ip_address> torqueserver

if pbs_mom is running, kill it:

ps -aux | grep pbs_mom
kill -2 <process id>

Now restart pbs_mom

pbs_mom

On the master node:

restart pbs_server:

qterm -t quick
pbs_server

wait several seconds and run the following command to check that the server is picking up the nodes:

pbsnodes –a

to test that cluster is working, submit a test job:

echo "sleep 30" | qsub

check that the job is running via the output from:

qnodes

Possible issues:

Jobs are queued but never run:

If you are submitting jobs and they are being queued, but are not running unless you force execution with qrun, ensure that pbs_sched is running:

qterm -t quick
pbs_sched
pbs_server

Now submit another job and check if it runs

Using the cluster:

submit jobs with qsub <script>
delete jobs in the queue with qdel <job_id>
check the status of jobs with qstat -a

2. Managing and using your basic cluster setup

2.1. How to submit a job to your cluster

To submit a job use the 'qsub' command:

qsub script.pbs

You can assign resources to the job using the '-l' argument. Common resources that can be assigned include:

The number of nodes and cores. The below command will assign 1 node and 64 processes per node (ppn) to the job

-l nodes=1:ppn=64

Memory. The above will assign 8 GB of RAM to the job

-l mem=8gb

Walltime

-l walltime=1:00:00

The above will set the maximum amount of time the job has to complete to 1 hour. If the job has not been completed by then, it will be killed. Using all the above in a single command would look like this:

qsub -l nodes=1:ppn=64 -l mem=8gb -l walltime=1:00:00 script.pbs

If your script takes arguments, you can pipe it into the qsub command as follows:

echo "script.pbs arg1 arg2 arg3" | qsub -l nodes=1:ppn=64 -l mem=8gb -l walltime=1:00:00

If your cluster has multiple queues, you can specify which queue to use using the '-q' argument:

qsub –q batch script.pbs

To specify which file to write the output and error streams of the job to, use '-e' and '-o':

qsub –e localhost:/path/to/error.txt –o localhost:/path/to/output.txt script.pbs

When you submit a job to the cluster, it does not use the environmental variables of the machine you are running the job on. You may wish to specify that the job runs with the same environmental variables as the machine you have submitted from. To do this, use '-V':

qsub –V script.pbs

To directly specify environmental variables you can use '-v'. Putting all this together, we get a rather long and cluttered command:

echo "script.pbs arg1 arg2 arg3" | qsub –V –q batch -l nodes=1:ppn=64 -l mem=8gb -l walltime=1:00:00 –e localhost:/path/to/error.txt –o localhost:/path/to/output.txt

There are still more arguments that we haven't covered and so a command like this can quite easily get even larger. Luckily, there is a way to specify the above mention options within your script using PBS directives. PBS directives are placed at the top of your script before any other lines. Directives that are placed below other lines in your script are ignored. A PBS directive looks as follows:

#PBS <option>

We can place the options in the above command in our script 'script.pbs' by adding the following lines at the top of it:

#PBS –V
#PBS –q batch
#PBS -l nodes=1:ppn=64
#PBS -l mem=8gb
#PBS -l walltime=1:00:00
#PBS –e localhost:/path/to/error.txt
#PBS –o localhost:/path/to/output.txt

With these directives in the script ‘script.pbs’, we can reduce the cumbersome command above to the following:

echo "script.pbs arg1 arg2 arg3” | qsub

2.2 Other useful commands

To check the status of your jobs running on the cluster:

qstat –a

To delete/stop a job running on the cluster:

qdel <job_id>

Note:
you can see the job_id of the job using the qstat command above. The first column produced by this command is the job_id.

Check the status of the nodes of the cluster:

qnodes