Running Batch Jobs at CAC

Drew Dolgert
Step-by-step is Separate from Talk

• Homework
  – Look at the gsAssembler link at the end.

• HELP!
  – http://www.cac.cornell.edu/help
  – help@cac.cornell.edu
  – http://www.cac.cornell.edu/wiki
Gear

- A cluster and login nodes
- SSH and X11
- Shell-fu
- Scheduler wrangling
Just Like Your Computer
Definitions

• Head node – linuxlogin1.cac.cornell.edu, linuxlogin2.cac.cornell.edu
• Compute nodes – similar to compute-3-17.v4linux
• Job – A request to the scheduler to perform a task
• Scheduler – Decides when and on which nodes your job will run
• Cores – Individual processors on each node. A single compute node can often run 8 jobs simultaneously with little loss in speed for any of them.
How to Open a Terminal on the Head Node

SSH

SSH server

Bash Shell
Definitions

- **Secure Shell Protocol** – A definition for how two computers can securely connect with each other.
- **SSH Client** – Software you install on your machine to connect with a server over the SSH protocol. By default, it lets you type and see lines of text sent back and forth.
- **SSH Server** – Always listening on the server for your connection.
- **Bash Shell** – The program the SSH server runs for you on the other end so that you can navigate directories and execute programs.
Copy Files with Same Protocol, Different Tools

- `sftp = sftp ajd27@linuxlogin1.cac.cornell.edu`
- `scp`
  - secure copy
SSH Clients

• **Windows**
  – Download Putty tools: putty.exe, psftp.exe
  – WinSCP

• **Mac – Open a terminal and type**
  – ssh -X ajd27@linuxlogin1.cac.cornell.edu
  – Fetch or sftp ajd27@linuxlogin2.cac.cornell.edu
  – scp

• **Linux – Open a terminal and type**
  – ssh -X ajd27@linuxlogin2.cac.cornell.edu
  – sftp ajd27@linuxlogin1.cac.cornell.edu
  – scp
Mac Terminal Window

Terminal — bash — 80×24

Last login: Wed Feb 10 13:03:25 on console
eplorer:~ slantz$ which python
/usr/bin/python
eplorer:~ slantz$
Invoking SSH

• Username
• Password
• Name of remote machine
• Whether to do X11 Forwarding

• Type at the terminal window:
  • ssh -X ajd27@linuxlogin2.cac.cornell.edu

• Or fill out the form for Putty (see online directions).
[ajd27@v4linuxlogin1 ~]$ pwd
/home/fs01/ajd27
[ajd27@v4linuxlogin1 ~]$ mkdir cbsu
[ajd27@v4linuxlogin1 ~]$ cd cbsu
[ajd27@v4linuxlogin1 ~]$ pwd
/home/fs01/ajd27/cbsu
[ajd27@v4linuxlogin1 ~]$ ls
[ajd27@v4linuxlogin1 ~]$ cp ~/dev/cbsu/* .
[ajd27@v4linuxlogin1 ~]$ ls
1020332.schedular.v4linux.OU short.sh short.sh~ xclock z.txt
[ajd27@v4linuxlogin1 ~]$ less short.sh
[ajd27@v4linuxlogin1 ~]$
How to Connect to the Head Node

SSH

SSH server

Bash Shell

Program with mouse input.

your computer

linux server

X-Windows, or X11

X-Windows Server

04/07/09

www.cac.cornell.edu
X-Windows Servers

• Windows
  – Download Xming, including its fonts package.
  – Install it.
  – Start Xming before you ssh to the head node.

• Mac – X-Windows is built in.

• Linux – X-Windows is built in.
Tunnel X11 Through SSH

- X11 is insecure
- SSH, as a protocol, can carry anything.
- SSH has automatic switches for X11
  - `ssh -X`, most of the time
  - `ssh -Y`, special cases, when `-x` fails.
  - “Enable X11 Forwarding” on Putty in Windows

04/07/09
www.cac.cornell.edu
Principles of a Shell

• Current working directory – Where you are is where programs run.

• Environment variables
  – PATH – A list of directories the shell searches for programs to run
  – SHELL – The type of the current shell, will be Bash.
  – USER – Username, also found with whoami command.

• $HOME – home directory

• Every program has stdin, stdout, stderr
  – ls > contents.txt 2> ls.err
Work with Directories and Files

- `pwd`
- `mkdir nextgen`
- `cd nextgen`
- `cd ..`
- `cd`
- `rm -rf nextgen`
- `cat 1010282.out`
- `grep val 1010282.out`
- `less 1010282.out`
- `rm *.out`

It’s like being a bartender or a soda-jerk. It’s easy once you know the basic moves. – The Man Who Wasn’t There
Command-line Expansion

you: $ ls *.out

prog:$ ls 1020253.out 1020257.out 1020258.out

you: $ ./first_gen -d ${DATA}

prog:$ /home/fs01/ajd27/cbsu/first_gen -d my.dat

you: $ cd ~/dev/jfm17

prog:$ cd /home/fs01/ajd27/dev/jfm17
Things That Expand

- Current directory .
- Parent directory ..
- Home directory ~
- Variable $PATH or ${PATH}
- Modified variable: JOB_ID=1020233.v4linux.sched
  ${JOB_ID%%.*}.out -> 1020233.out
Work with Variables

- `env|sort|less`
- `echo $PATH`
- `export MYDIR=$HOME/cbsu`
- `export PATH=/opt/nextgen/bin:${PATH}`
Run in Shell

• which perl
• ctrl-c
• ./myprogram
• emacs&
• emacs
ctrl-z
bg
• jobs
• fg
• kill %1
Surroundings

- hostname
- whoami
- ps augx|grep ajd27
Help in Bash

• man less
• man -k moab
• gcc --help
• icc -h
Shell Takeaway

- Environment is working directory and variables, files on disk.
- Those commands form a programming language.
- You will use that programming language to tell the scheduler what to run for you.
#!/bin/bash
PBS -l walltime=00:30:00,nodes=1
PBS -A dal16_0003
PBS -j oe
PBSD -o $PBS_JOBID.out
PBS -N batchtest
PBS -q v4dev

# Turn on echo of shell commands
set -x

NODECNT=$(/usr/bin/ls -1 < "$PBS_NODEFILE")
TASKCNT=`/usr/bin/expr 8 \
"*" $NODECNT`
RUNDIR=$PBS_O_WORKDIR

# The job id is something like 613.scheduler.v4linux.
# This deletes everything after the first dot.
Matrix of Editors on the Login Nodes

in terminal window

use X-Windows

**Simple**

Nano

**The Unix Way**

vi

gedit

emacs

04/07/09 www.cac.cornell.edu
To Edit A File in VI (short for “visual”)

• “vi filename” will open it or create it if it doesn’t exist.
• Command mode to start, switch to Insert mode.
• Command mode. Cursors work here, too.
  – :w  Writes a file to disk.
  – :q  Quits
  – :q! Quits even if there are changes to a file
  – i   Takes you to insert mode
• Insert Mode
  – Cursors, typing characters, and deleting work here.
  – Escape key takes you to command mode.
• Ctrl-c will get you nowhere.
Editor on Your Computer that Saves to Cluster

- EditPlus on Windows
- TextWrangler on Mac
Queues

• **v4** – Main queue to get access to compute nodes
  – 19 servers containing 152 cores
  – 16 GB RAM per server
  – Limits: Max 19 nodes, no walltime limit

• **v4dev** – Development queue for debugging and testing
  – Maximum of 2 nodes (16 cores)
  – Limits: Max 2 nodes, 60 minutes walltime

• **v4-64g** – Large-memory servers
  – 4 servers, total of 32 cores
  – 64 GB RAM per server
  – Maximum 3 nodes (24 processors), no walltime limit
## Leases

- Groups purchase rights to nodes.

```
[ajd27@v4linuxlogin1 ~]$ showqlease
run  free  wait  total  queue  lease        nodes
 0   18    0    18  v4    standard     compute-3-[29-46]
 0    8    0     8  v4    acs4_0001    compute-3-[1-8]
 0    1    0     1  v4    ajd27        compute-3-28
 9    0    0     9  v4    asr2004_0001 compute-3-[9-17]
 2    8    0    10  v4    jp86_0005    compute-3-[18-27]
 0    3    0     3  v4-64g standard     lmcompute-4-[1-3]
 0    1    0     1  v4-64g sc167_0001 lmcompute-4-4
 0    2    0     2  v4dev standard      compute-3-[47-48]
```

Administrative downtime:

- Wed Mar 10 08:00:00 for 9:00:00 compute-3-[1-48], lmcompute-4-[1-4]
Life of a Job

1. You create a job script.
2. You submit job script to scheduler using nsub.
3. Scheduler checks the job and either returns an error or sets its state to Idle / Eligible.
4. When scheduler finds available resources, it
   a) sets job state to Running
   b) asks job manager on selected nodes to run the job in a shell.
5. Job completes because
   a) The script finished executing.
   b) It exceeded its time limit.
6. Scheduler sets job state to Completed.
Batch Script is a File with Directives

#!/bin/bash
#PBS -l walltime=00:30:00,nodes=1
#PBS -A dal16_0003
#PBS -j oe
#PBS -o $PBS_O_WORKDIR/$PBS_JOBID.out
#PBS -N batchtest
#PBS -q v4dev
#PBS -V

# Turn on echo of shell commands
set -x

DATADIR=/home/gfs08/jp86/ngw2010/session1/lecture2/
DATA=s_8_1_sequence.txt

cp ${DATADIR}/${DATA} ${TMPDIR}
cd ${TMPDIR}
fastx_quality_stats -i $DATA -o stat.xls
cp stat.xls ${PBS_O_WORKDIR}/

-1 requests nodes for a time

-A is your account number

-j joins output and error from programs for the output file

-o is the name of the output file

-N is the job name

-q specifies which queue runs the job

-V tells it to use whatever variables were set when you submitted the job
Drives

- Your home drive is available everywhere.
- On compute nodes, $\text{TMPDIR}$ is a local temporary directory.
#!/bin/bash

#PBS -l walltime=00:30:00, nodes=1
#PBS -A dal16_0003
#PBS -j oe
#PBS -o ${PBS_O_WORKDIR}/${PBS_JOBID}.out
#PBS -N batchtest
#PBS -q v4dev
#PBS -V

# Turn on echo of shell commands
set -x

DATADIR=/home/gfs08/jp86/ngw2010/session1/lecture2/
DATA=s_8_1_sequence.txt
cp ${DATADIR}/${DATA} ${TMPDIR}
cd ${TMPDIR}
fastx_quality_stats -i ${DATA} -o stat.xls
cp stat.xls ${PBS_O_WORKDIR}/

• Copy to TMPDIR.
• Go to TMPDIR.
• Run program.
• Copy to WORKDIR.
Moab Commands

• nsub – Submit a job
  – nsub jobscript.sh

• showq – Display queue information
  – showq
  – showq -w user=ajd27
  – showq -w class=v4dev

• checkjob – How is this one job doing?
  – checkjob -v 1020282

• mjobctl - modify job
  – mjobctl -c 1020282, cancel a job

• showqlease – get a sense of how busy the queues are
Submit a Job

[ajd27@v4linuxlogin1 ~/cbsu]$ nsub short.sh
Looking for directives in short.sh

1020333

[ajd27@v4linuxlogin1 ~/cbsu]$ showq
active jobs------------------------

<table>
<thead>
<tr>
<th>JOBID</th>
<th>USERNAME</th>
<th>STATE</th>
<th>PROCS</th>
<th>REMAINING</th>
<th>STARTTIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1020335</td>
<td>lmv47</td>
<td>Running</td>
<td>8</td>
<td>14:13:59</td>
<td>Mon Mar 1 13:26:04</td>
</tr>
</tbody>
</table>

3 active jobs    80 of 416 processors in use by local jobs (19.23%)
10 of 52 nodes active   (19.23%)
Checkjob

[ajd27@v4linuxlogin1 ~]$ checkjob 1020287
job 1020287

AName: mpiTest
State: Running
Creds: user:kab2003 group:Domain Users account:asr2004_0001 class:v4
WallTime:  1:15:41:03 of 4:04:00:00
SubmitTime: Sun Feb 28 20:59:02
   (Time Queued Total: 00:00:16 Eligible: 00:00:00)
StartTime: Sun Feb 28 20:59:18
NodeMatchPolicy: EXACTNODE
Total Requested Tasks: 8
Req[0]  TaskCount: 8  Partition: scheduler
Allocated Nodes:
[compute-3-17.v4linux:1][compute-3-16.v4linux:1][compute-3-15.v4linux:1]
[compute-3-14.v4linux:1][compute-3-13.v4linux:1][compute-3-12.v4linux:1]
[compute-3-11.v4linux:1][compute-3-10.v4linux:1]
Ganglia Cluster Toolkit: Cluster Report - Mozilla Firefox

http://v4.cac.cornell.edu/ganglia/
See a Profile of Running Job

- http://v4.cac.cornell.edu/ganglia, then click on the nodes for your job.
What Can Go Wrong

• Fail immediately
  – Script does not exist
  – Wrong account number

• Make you wait, but eventually work
  – People ahead of you in the queue make it busy
  – You submit one job that says it may use all of your minutes (but doesn’t) and another job after it.

• Fail after the job starts
  – Errors in the batch file so it does something wrong

• Stop the job in the queue – Job state Blocked
  – Requested more nodes than the limit of the queue
How to Debug Problems

- Look at log files
- Make an interactive job.

```bash
#!/bin/bash
#PBS -l walltime=00:30:00,nodes=1
#PBS -A dal16_0003
#PBS -j oe
#PBS -o $PBS_JOBID.out
#PBS -N batchtest
#PBS -q v4dev
#PBS -V
#PBS -I

[ajd27@v4linuxlogin1 ~/cbsu]$ nsub short.sh
Looking for directives in short.sh
Executing interactive
qsub: waiting for job 1020349.scheduler.v4linux to start
qsub: job 1020349.scheduler.v4linux ready

[ajd27@compute-3-48 ~/cbsu]$ logout
qsub: job 1020349.scheduler.v4linux completed
```
SSH to a Node While It Runs Your Job

- Only when a node is running your job can you login to it.
- You can only get to a compute node from a login node.

```plaintext
[ajd27@v4linuxlogin1 ~/cbsu]$ nsub short.sh
Looking for directives in short.sh
1020353
[ajd27@v4linuxlogin1 ~/cbsu]$ showq|grep ajd27
[ajd27@v4linuxlogin1 ~/cbsu]$ checkjob 1020353|grep v4linux
[compute-3-48.v4linux:1]
[ajd27@v4linuxlogin1 ~/cbsu]$ ssh -Y compute-3-48.v4linux
```
Puzzle: How to Start a Batch Job with a GUI

• gsAssembly is a graphical interface on Newbler.
• I want to run it in batch, but I would like to see the graphical interface on the compute node to kick off the job.

• Would interactive batch jobs work?
• Given a regular batch job, what do you put in the script?
• How would the job know when to quit?
Batch Script that Waits for You

PROG=newbler

echo Wait until user logs in
while who -q|head -1|grep -v "\b${USER}\b"
do
  sleep 10
done

echo Wait until the user logs off
while who -q|head -1|grep "\b${USER}\b"
do
  sleep 30
done

echo Wait until the $PROG finishes
while ps -u ${USER} -o comm|grep $PROG
do
  sleep 30
done

echo $PROG finished `date`
How to Kickstart a Long Job from a GUI

• Create a batch script that starts but does nothing.
• Wait for the job to start.
• `ssh -Y` to the node.
• Start your program with the magic `nohup`.
• Configure it and tell it to run.
• Logout and wait for results.

```
nohup script.sh > z.out 2> z.err < /dev/null &
```
What Does It Take to Be Skilled at This?

- Know the shell.
- Understand processes and threads on a Unix system.

- Study
- Experiment