Advanced Use of
the HPCC Cluster

Ken-ichi Nomura, Ph.D.

Center for High-Performance
Computing and Communications
Outline

1. Interactive PBS Job
2. Software Installation for Project
3. Parallel Job Execution
   - Shell Script
   - Message Passing Interface (MPI)
4. Hands-on
Downtime Notice

Beginning at 9:00 a.m. on Monday, April 14, 2014, the entire HPCC cluster—including all head nodes, filesystems, almaak machines, and compute nodes—will be unavailable due to our spring 2014 maintenance. We anticipate releasing the entire cluster back to the user community by 9:00 a.m. on Thursday, April 17.

- CENTOS 6.5 upgrade to all head nodes and compute nodes
- Resource manager TORQUE upgrade
- the /staging filesystem update
- Apply security patches, operating patches, firmware patches

Notice

- The upgrade of TORQUE will clear all jobs from the job queue.
- The TORQUE upgrade removes /usr/bin/mpiexec. You will need to source either openmpi or mpich2 to have mpiexec in your path. We recommend openmpi as it works across both clusters, Myrinet and Infiniband.
- /staging will be re-created, and all data will be lost on this temporary filesystem.

If you have any questions or concerns about this downtime, please contact us at hpcc@usc.edu.
HPCC Overview

HPCC clusters:
- HP SL230 (128GB memory), HP SL250 (64 GB memory, dual NVIDIA K20m GPUs), Dual Xeon 8-core 2.4 GHz, on 56.6-Gbit FDR Infiniband network
- 2,225-node, 4-core, 6-core, and 12-core dual-processor cluster contains Dell, Oracle Sun, HP, and IBM compute nodes on a 10-Gbit Myrinet network
- 4 large-memory nodes with 1 TB of RAM and 4x10-core Intel Xeon processors.
- 2 head nodes, hpc-login1.usc.edu and hpc-login2.usc.edu
PBS Script

To submit your job to the cluster, use PBS script describing what resources you need and what command you want to run.

```
#!/bin/sh
#PBS -l nodes=1:ppn=12,walltime=00:30:00
#PBS -d .
Command list to execute
...
```

Use `qsub` to submit your job, for example `qsub test.pbs`

You can also overwrite PBS options from command line

```
[~]$ qsub test.pbs -l nodes=4:ppn=8,mem=1gb,walltime=12:00:00
```
PBS Commands: qsub

qsub - submit PBS job

- **I** interactive job
- **d** starting directory
- **l** resource list
  - **nodes** number of nodes
  - **ppn** processor per core
  - **gpus** number of GPUs
  - **nodeset** machine architecture
  - **mem** amount of total memory
  - **walltime** wallclock time for your job
  - **procs** request $N$ CPU cores
PBS Commands: qstat & qdel

qstat - show status of PBS jobs
- `a` all jobs are displayed
- `u username` display status of specific user’s job
- `f jobid` display full status of a specific job

qdel - cancel PBS job
qdel `jobid`
Interactive PBS Job

- New login shell starts on one of the computing nodes once an interactive job is accepted
- While the interactive job is running, you can log on the computing nodes via ssh
- User can interactively compile/debug/test your code
- This facilitates the code development process, for better turnaround and higher productivity
Interactive PBS Job

[hpc-login2 ~]$ qsub -I -d . -l nodes=2:ppn=8
qsub: waiting for job 7456677.hpc-pbs.hpcc.usc.edu to start
qsub: job 7456677.hpc-pbs.hpcc.usc.edu ready

----------------------------------------
Begin PBS Prologue Sat Mar 1 14:21:49 PST 2014
Job ID: 7456677.hpc-pbs.hpcc.usc.edu
Username: knomura
...
Nodes: hpc2015 hpc2016
PVFS: /scratch (tcp://hpc2015.m10g.hpcc.usc.edu:3334/pvfs2-fs),
/staging (tcp://hpc-ofs11.ib.hpcc.usc.edu:3334/staging)
TMPDIR: /tmp/7456677.hpc-pbs.hpcc.usc.edu
End PBS Prologue Sat Mar 1 14:22:00 PST 2014
----------------------------------------

[hpc2016 ~]$
myqueue command shows the jobs status and allocated node list for running jobs.
Example: one job running and one job in quick queue

```bash
[~]$ myqueue
hpc-pbs.hpcc.usc.edu:
Req'd Req'd Elap Job ID Username Queue Jobname SessID
NDS  TSK  Memory Time S Time
-------------------- ----------- -------- ---------------------- ------ ----- ------ ------ ----- ------ ------
7456587.hpc-pbs. knomura quick STDIN 602 2 16 5gb
00:30 R 00:03
hpc0431/7+hpc0431/6+hpc0431/5+hpc0431/4+hpc0431/3+hpc0431/2+hpc0431/1+hpc0431/0+hpc0432/7+hpc0432/6+hpc0432/5+hpc0432/4+hpc0432/3+hpc0432/2+hpc0432/1+hpc0432/0
7456592.hpc-pbs. knomura quick STDIN -- 2 16 -- 00:30
Q -- --
```
Job Monitoring

[hpc-login2 ~]$ ssh hpc0431
Last login: Fri Feb 28 13:56:07 2014 from hpc-login2-l.hpcc.usc.edu
[hpc0431 ~]$ head /proc/cpuinfo
processor : 0
vendor_id : GenuineIntel
cpu family : 6
model : 44
model name : Intel(R) Xeon(R) CPU           X5650  @ 2.67GHz
...
[hpc0431 ~]$ head /proc/meminfo
MemTotal:    24596976 kB
MemFree:     23240620 kB
Buffers:     51128 kB
Cached:      882788 kB
...
[hpc0431 ~]$ top
## Tips: PBS Nodeset

<table>
<thead>
<tr>
<th>First</th>
<th>Last</th>
<th>#</th>
<th>Node Type</th>
<th>/tmp</th>
<th>Nodeset</th>
</tr>
</thead>
<tbody>
<tr>
<td>hpc0965</td>
<td>hpc0972</td>
<td>8</td>
<td>Dual Hexcore Intel Xeon 3.0 GHz, 24GB</td>
<td>160GB</td>
<td>s1160</td>
</tr>
<tr>
<td>hpc1044</td>
<td>hpc1050</td>
<td>7</td>
<td>Dual Dodecacer AMD Opteron 2.3 GHz, 48GB</td>
<td>1TB</td>
<td>dl165</td>
</tr>
<tr>
<td>hpc1123</td>
<td>hpc1128</td>
<td>6</td>
<td>Dual Dodecacer AMD Opteron 2.3 GHz, 48GB</td>
<td>1TB</td>
<td>dl165</td>
</tr>
<tr>
<td>hpc1196</td>
<td>hpc1200</td>
<td>5</td>
<td>Dual Dodecacer AMD Opteron 2.3 GHz, 48GB</td>
<td>1TB</td>
<td>dl165</td>
</tr>
<tr>
<td>hpc1223</td>
<td>hpc1230</td>
<td>8</td>
<td>Dual Dodecacer AMD Opteron 2.3 GHz, 48GB</td>
<td>1TB</td>
<td>dl165</td>
</tr>
<tr>
<td>hpc1723</td>
<td>hpc1756</td>
<td>28</td>
<td>Dual Dualcore AMD Opteron 2.3 GHz, 16GB</td>
<td>250GB</td>
<td>x2200</td>
</tr>
<tr>
<td>hpc1872</td>
<td>hpc2081</td>
<td>210</td>
<td>Dual Quadcore Intel Xeon 2.33 GHz, 12GB</td>
<td>60GB</td>
<td>pe1950</td>
</tr>
<tr>
<td>hpc2283</td>
<td>hpc2337</td>
<td>55</td>
<td>Dual Quadcore Intel Xeon 2.5 GHz, 12GB</td>
<td>60GB</td>
<td>pe1950</td>
</tr>
<tr>
<td>hpc2349</td>
<td>hpc2370</td>
<td>21</td>
<td>Dual Quadcore AMD Opteron 2.3 GHz, 16GB</td>
<td>250GB</td>
<td>x2200</td>
</tr>
<tr>
<td>hpc2470</td>
<td>hpc2601</td>
<td>129</td>
<td>Dual Quadcore AMD Opteron 2.3 GHz, 16GB</td>
<td>250GB</td>
<td>x2200</td>
</tr>
<tr>
<td>hpc2758</td>
<td>hpc2761</td>
<td>4</td>
<td>Dual Hexcore Intel Xeon 2.66 GHz, 24GB</td>
<td>120GB</td>
<td>dx360</td>
</tr>
<tr>
<td>hpc3030</td>
<td>hpc3264</td>
<td>236</td>
<td>Dual Octocore Intel Xeon 2.4 GHz, Dual k20 NVIDIA, 64GB</td>
<td>1TB</td>
<td>sl250s</td>
</tr>
<tr>
<td>hpc3386</td>
<td>hpc3389</td>
<td>4</td>
<td>Dual Octocore Intel Xeon 2.4 GHz, 128GB</td>
<td>1TB</td>
<td>sl230s</td>
</tr>
</tbody>
</table>

```bash
[~]$ qsub -I -d . -l nodes=2:ppn=8:pe1950
```

http://hpcc.usc.edu/support/infrastructure/node-allocation/
Tips: Temporary Disk Space /staging

- **328-terabyte parallel file system** hosted on dedicated storage machines.
- Data are **not deleted between jobs. No quotas** on /staging.
- A directory is automatically created for each HPCC project. Good for data sharing.
- **But the data is never backed up!**
- Files on /staging that have not been accessed for **more than 10 days are subject to deletion** by an automated cleaning system.
- **All data are completely cleared and rebuilt** during HPCC’s twice-yearly cluster downtimes.

---

**Temporary Disk Space**

All users must use the /tmp, /scratch, or /staging file systems as their working directory for all jobs.

**The /tmp File System**
The /tmp file system is available locally on each node. Please refer to the Node Allocation page for the /tmp disk space available on each node. Users should use the directory created exclusively for each job defined by the environment variable $TMFDIR. Users have access to the /tmp directory of a particular node only when a job is running on that node. All files created in /tmp are deleted before the next job starts.

**The /scratch File System**
/scratch is a shared temporary file system that is created when a new job starts and is deleted at the end of the job. The /scratch file system should be used to store temporary files that need to be accessed from all the nodes. If you need the files in the /scratch file system saved, copy them before the end of the job to a permanent storage disk.

**The /staging File System**
/staging is a 328-terabyte parallel file system that is hosted on dedicated storage machines. Data stored in /staging is retained and not deleted between jobs.

A directory is automatically created for each HPCC project under /staging so that data can be stored there temporarily. Project-specific user subdirectories are also created. All subdirectories created under /staging will automatically be set to allow project group access for easier data sharing.

**Managing Files in the Temporary File Systems**
Files on all of the temporary file systems are not backed up and it is the user’s responsibility to copy important data to a permanent project file system for safekeeping. While HPCC will try to project user

http://hpcc.usc.edu/support/infrastructure/temporary-disk-space/
Outline

1. Interactive PBS Job
2. Software Installation for Project
3. Parallel Job Execution
   - Shell Script
   - Message Passing Interface (MPI)
4. Hands-on
Software Installation

• There are many open source software and libraries available for Linux

• Open source applications are often provided as a tarball that includes source codes, scripts and instructions to compile and install it

  ./configure && make && make install

• Proprietary software can be installed on the HPCC cluster (e.g. Intel&PGI compilers, Matlab, Mathematica, lumerical FDTD) but doing so requires a proper license

• Windows applications are not supported
Example: python Installation

Installation Steps
1. Download source code from python website
2. Compile & install under project directory
3. Give proper permissions to share with group member
4. Set environment variables
Example: python Installation

```
[Python-2.7.6]$ less README
[Python-2.7.6]$ ./configure --help
`configure' configures python 2.7 to adapt to many kinds of systems.
Usage: ./configure [OPTION]... [VAR=VALUE]...
Configuration:
  -h, --help        display this help and exit
       --help=short display options specific to this package
       --help=recursive display the short help of all the included packages
  -V, --version     display version information and exit
  -q, --quiet, --silent do not print `checking ...' messages
       --cache-file=FILE cache test results in FILE [disabled]
  -C, --config-cache alias for `--cache-file=config.cache'
```

After expanding the tarball, it’s always a good idea to take a look at README and help messages from the configure script before deploying.
Example: python Installation

# Change to project directory
cd /home/rcf-proj/projectid/myacct
# Download python tarball
wget http://www.python.org/ftp/python/2.7.6/Python-2.7.6.tgz
# Unarchive the tarball and change directory
tar xvfz Python-2.7.6.tgz && cd Python-2.7.6
# run configure script with custom installation directory
./configure --prefix=/home/rcf-proj/projectid/myacct/python/2.7.6/
# Compile and install
make && make install
# Create setup script
vi setup.sh (or setup.csh)
# Check the setup script
source setup.sh
echo $PATH
# Set the readable & executable permission for your group
chmod -R g+rx .
Directory and Quota

• **Home directory**  /home/rcf-xx/username  
  Each user has **1 GB** of disk quota and **100,000 files** of file quota

• **Project directory**  /home/rcf-proj/projectid  
  Quota on project directory varies depending on each project

If you need more storage for your project, send request from your project webpage  
https://www-rcf.usc.edu/rcfdocs/hpcc/allocations/
myquota shows the quota on your home & project directories.

- Exceeding quotas (in either home directory or project directory) may cause the program to crash and fail to save results.
- Pay attention to the file quota (max number of files) also.
Outline

1. Interactive PBS Job
2. Software Installation for Project
3. Parallel Job Execution
   - Shell Script
   - Message Passing Interface (MPI)
4. Hands-on
Parallel Job Execution by Script

1 #!/bin/bash
2 #PBS -l nodes=1:ppn=8
3 #PBS -l walltime=00:30:00
4 #PBS -d .
5 #PBS -j oe
6 #PBS -N parexec_shell
7
8 for nloop in {0..2};do
9   echo === ${nloop} ===
10  for n in {1..8}; do
11     let n1=8*${nloop}+${n}
12     sleep 5 && echo ${n1} awake &
13   done
14   wait
15 done

1: use bash
2-6: PBS options. 1 node, 8 proc-per-node, 30mins wallclock time, set current directory as working directory, merge standard error into standard output, set job name to be “parexec_shell”
8: 1st loop, nloop=0,1,2
9: print loop counter
10: 2nd loop, n=1,2,…,7,8
11: compute process ID
12: sleep 5 sec, then print awake message, execute each process as background
13: 2nd loop done
14: wait for all background jobs to finish
15: 1st loop done
MPI Library and Compiler

- Standard library for high-performance parallel computing
- There are many implementations of MPI (MPICH, OpenMPI, MPVAPICH)
- Use wrapper scripts (mpicc, mpicxx, mpif77, mpif90) to compile MPI programs
- GNU/Intel/PGI compilers
#include "mpi.h"
#include <stdio.h>
#include <stdlib.h>

int main( int argc, char *argv[]) {
    int n, myid, numprocs, i;
    char cmd[256];

    MPI_Init(&argc,&argv);
    MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD,&myid);

    sprintf(cmd, "sleep 5 && echo %d awake \n", myid+atoi(argv[1]));
    system(cmd);
    MPI_Finalize();
    return 0;
}
1 #!/bin/bash
2 #PBS -l nodes=32:ppn=8
3 #PBS -l walltime=00:30:00
4 #PBS -d .
5 #PBS -j oe
6 #PBS -N parexec_shell
7 source /usr/usc/openmpi/default/setup.sh
8 NP=`wc -l < ${PBS_NODEFILE}`
9 for nloop in {0..3};do
10   echo === ${nloop} ===
11   let n1=${NP}*$nloop
12   mpirun -np ${NP} ./parexec ${n1}
13 end

1: use bash
2-6: PBS options. 32 nodes, 8 proc-per-node, 30mins wallclock time, set current
directory as working directory, merge standard error into standard output, set job
name to be “parexec_MPI”
7: source OpenMPI setup file
8: compute the total number of procs
9: loop begins, nloop=0,1,2,3
10: print loop counter
11: compute batch ID, n1=0, 256, 512..., each batch spawns 256 MPI processes
12: run parexec
13: loop ends
Thank you!