Introduction to HPC Cluster Computing

Avalon Johnson

Center for High-Performance Computing
Outline

1. HPC Overview
2. Account Management
   - Directories
   - Quotas
   - Computing Time
3. Software Repository
4. Portable Batch System (PBS)
   - PBS Basics
   - Interactive Mode
   - Job Monitoring
Center for High-Performance Computing

- Foundation for computational research at the University of Southern California.
- World-class academic supercomputer center supported through local investments and without national funding.
- Provides USC faculty and graduate students free access to the facility, HPC tours, trainings and workshops every semester.
- Supports USC research groups in variety of disciplines across campuses.
Computing Services

- Over 2,700 computing nodes (32K CPU cores) on 10G/s Myrinet and 56Gbit/s FDR Infiniband interconnects, 260 GPU (Tesla K20m) nodes
- 2.4 PetaBytes of total storage with GPFS, Panasas, Samfs, NFS
- Over 320 TeraBytes staging storage with OrangeFS
- Cent OS 6.5 Linux, Torque and Moab for resource management and scheduling
- Scientific software and libraries
- Email user support (hpc@usc.edu)
- Online documentations (http://hpcc.usc.edu)
Software Service

- A variety of software, from commercial (e.g. MATLAB, Intel & PGI compilers) to open source programs, are available.
- HPC will assist researchers and install software upon request.
- Researchers are primarily responsible for software & licenses.

<table>
<thead>
<tr>
<th>Software</th>
<th>fftw</th>
<th>cuda</th>
<th>intel</th>
<th>gnu</th>
<th>sas</th>
</tr>
</thead>
<tbody>
<tr>
<td>python</td>
<td>Pegasus</td>
<td>matlab</td>
<td>fdtd</td>
<td>mathematica</td>
<td>iperf</td>
</tr>
<tr>
<td>qespresso</td>
<td>hdf5</td>
<td>globus</td>
<td>gaussian</td>
<td>pgi</td>
<td>spss</td>
</tr>
<tr>
<td>qiime</td>
<td>libroadrunner</td>
<td>hadoop</td>
<td>gromacs</td>
<td>llvm</td>
<td>stata</td>
</tr>
<tr>
<td>R</td>
<td>mpich</td>
<td>hdfview</td>
<td>lammps</td>
<td>boost</td>
<td>taxila</td>
</tr>
<tr>
<td>git</td>
<td>openmpi</td>
<td>hpctoolkit</td>
<td>NAMD</td>
<td>cellprofiler</td>
<td>bbcp</td>
</tr>
<tr>
<td>opencv</td>
<td>papi</td>
<td>petsc</td>
<td>schroedinger</td>
<td>gurobi</td>
<td>openmpi</td>
</tr>
</tbody>
</table>
User and Job Statistics

<table>
<thead>
<tr>
<th></th>
<th>User Accts</th>
<th>PI Accts</th>
<th>Disk TB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1,128</td>
<td>205</td>
<td>1,700</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>General Jobs</th>
<th>Condo Jobs</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>448,864</td>
<td>1,446,750</td>
<td>1,895,614</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th># Nodes</th>
<th>Condo</th>
<th>General</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10Gb</td>
<td>1728</td>
<td>482</td>
<td>2210</td>
</tr>
<tr>
<td></td>
<td>56.6Gb</td>
<td>143</td>
<td>329</td>
<td>472</td>
</tr>
</tbody>
</table>
HPC trainings and workshops

- Introduction to Linux and the HPC cluster
- Parallel Matlab computing
- GPU and CUDA programming
- Guest lectures

GPU programming workshop by NVIDIA
Activities

NSF-funded projects:

- Trojan Express Network II (TEN-II) – connecting USC campuses via high-speed network
- Advanced Cyberinfrastructure – Research and Education Facilitators (ACI-REFs): Six universities collaborate to empower computational researchers on each campus

USC booth presentation at annual Supercomputing (SC) conference
HPC node organization

- Compute nodes are connected by two high speed low latency networks, Infiniband and Myrinet
- The Infiniband and the Myrinet networks are NOT connected to each other
- Nodes connected via the Infiniband network CANNOT do HIGH SPEED communication with nodes in the Myrinet network
- hpc-login2 and hpc-login3 provide access from the outside to the nodes.
Outline

1. HPC Overview
2. Account Management
   - Directories and Quotas
   - Computing Time
3. Software Repository
4. Portable Batch System (PBS)
   - PBS Basics
   - Interactive Mode
   - Job Monitoring
Directories and Quotas

Each user has **two** types of **permanent** directories: **home** and **project**.

Each user also has access to a ‘**quasi permanent’ staging** directory. While running a job, a user also has access to a ‘**temporary’ scratch** directory.

• **Home directory**

This is your default directory. When you login to the HPC cluster you will be in your home directory.

$ cd

$ pwd

/home/rcf-40/avalonjo
Directories and Quotas

• Project directory

Each user has one directory for each project that you belong to.

Each project directory is of the form:

```
/home/rcf-proj/<projectid>/<userid>
```

$ groups

```
g03 src ucsadmin rds lc_test gaussian hpcusers lc_hpcc
```

$ ls -ld /home/rcf-proj/hpcc/avalonjo

```
drwx------ 28 avalonjo g03 4096 Mar 30 14:41 /home/rcf-proj/hpcc/avalonjo
```
Directories and Quotas

• Staging directory

Like the project directory, each user has one staging directory for each project that they are in. Each staging directory is of the form:

/staging/<projectid>/<userid>

$ groups
g03 src ucsadmin rds lc_test gaussian hpcusers lc_hpcc

$ ls -ld /staging/hpcc/avalonjo

drwxr-s--- 1 avalonjo lc_hpcc 4096 Feb 26 17:17 /staging/hpcc/avalonjo/
Directories and Quotas

The **project** and **home** directories both have limits on usage called quotas. These quotas apply BOTH to the **number** of files as well as **total disk space** used.

- **Home directory**
  Home directory has **1 GB of disk space quota** and **100,000 files of file quota**.

- **Project directory**
  Your project usage limits is dictated by the project itself.
Directories and Quotas

• Staging directory

The staging directories have No quotas on disk space or number of files. Is a parallel file system (OrangeFS) NO DATA BACKUP, and all files will be cleaned up every downtime (approximately twice a year).

Good for applications with high-frequency data access (read and write). After your calculations finished, you should move results to your project directory.
Directories and Quotas

• /scratch directory

The /scratch directory is created for each job and is comprised of all the ‘free’ disk space present in the first 20 nodes in a job. It is created using a parallel file system (OrangeFS).

This space is available to ALL the nodes running your job.

This size will vary depending on the size of the jobs and nodes but will normally vary between about 1TB (for a one node job) and 20TB (for jobs > 20 nodes.)

NO DATA BACKUP, and all files will be cleaned up at job completion.
Monitoring Your Quota: myquota

myquota shows the quota on your home and project directories.

$ myquota

----------------------------------------------
Disk Quota for /home/rcf-40/avalonjo ID 203387

<table>
<thead>
<tr>
<th>Used</th>
<th>Soft</th>
<th>Hard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Files</td>
<td>9501</td>
<td>100000</td>
</tr>
<tr>
<td>Bytes</td>
<td>721.41M</td>
<td>1.00G</td>
</tr>
</tbody>
</table>

----------------------------------------------
Disk Quota for /home/rcf-proj2/hpcc ID 419

<table>
<thead>
<tr>
<th>Used</th>
<th>Soft</th>
<th>Hard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Files</td>
<td>502016</td>
<td>1000000</td>
</tr>
<tr>
<td>Bytes</td>
<td>433.86G</td>
<td>500.00G</td>
</tr>
</tbody>
</table>

Files for file quota and Bytes for disk quota.

Hard quota is the absolute limit you can store.
Monitoring Your Quota: myquota

• If you go over quota your job may crash when it fails to write files. This can be in either home directory or project directory.

• If you don’t specify where PBS output file will be stored in your PBS script, it may try to store the output file in your home directory and crash if you are over quota.

• Pay attention to files quota (number of files). Some users have millions of tiny files. This places a very large burden on the system since these all have to be backed up!

• If you need more space in project directory, submit a request from your project page:

https://www-rcf.usc.edu/rcfdocs/hpcc/allocations/
Computing Time

• To be able to run your job on the HPC cluster, you need to have computing time *(unit is #cores × hr)* in your project account.

• Whenever your job finishes (successfully or unsuccessfully), the project account is charged by the number of cores × wallclock time your job spent.

• If you request 2 nodes with 4 processors per node for 2 hours (-l nodes=2:ppn=4,walltime=2:00:00), the total charge is **2x4x2 = 16 core-hours**.
Monitoring Computing Time: mybalance

mybalance shows current balance of project account

$ mybalance
Balance  Name
---------  ---------------
Infinity hpccadm
227032 HPCCTestFund
Infinity HPCWorkShopApr2015

• All users have **default account** and computing time will be charged on the default account automatically.
• Sometimes you need to specify account name in your PBS script by `-A` option. E.g. `-A lc_kn1`
• If your job doesn’t start, remaining in the queue a long time, it’s always a good idea to check if your project has enough balance.
Outline

1. HPC Overview
2. Account Management
   – Directory and Quota
   – Computing Time
3. Software Repository
4. Portable Batch System (PBS)
   – PBS Basics
   – Interactive Mode
   – Job Monitoring
Software Repository: /usr/usc

HPC installs & maintains software in a single software repository.

Compilers: gnu, intel, pgi

Numerical Libraries: mpich, openmpi, cuda, fftw, petsc

Molecular Simulation: NAMD, gromacs, amber

Quantum Chemistry: gaussian, schrodinger

Numerical Environment: matlab, R, python

hpc-login2.usc.edu for 64-bit applications
hpc-login3.usc.edu for 64-bit applications
Software Repository: /usr/usc

What does the software repository look like?

$ cd /usr/usc/
$ ls -F
acml/  fftw/  imp/  mpich2/  qespresseo/
amber/ gaussian/ intel/ mpich-mx/ qiime/
aspera/ gflags/ iperf/ mvapich2/ R/
bbcp/  git/  java@ NAMD/  root/
bin/  globus/ jdk/  ncview/  sas/
....
$ ls -F hello_usc
1.0/ 2.0/ 3.0/
Software Repository: /usr/uscs

How can I access software?

- First, go to the directory of the software you want to use. Usually each software has several subdirectories for different versions. Pick the one you want.

- Look for setup scripts: setup.sh for bash users and setup.csh for tcsh users.

- source the setup file!
  
  $ source /usr/uscs/hello_usc/2.0/setup.sh
  $ source /usr/uscs/hello_usc/3.0/setup.csh
Software Repository: /usr/usc

What will happen when I source a setup script?

```
$ hello_usc
-bash: hello_usc: command not found

$ source /usr/usc/hello_usc/2.0/setup.sh

$ hello_usc

Hello USC!!!!.
I am version 2.0 running on host: hpc-login3

$ which hello_usc
/usr/usc/hello_usc/2.0/bin/hello_usc
```
SoEware Repository: /usr/usc

$ cat /usr/usc/hello_usc/2.0/setup.sh
if [ "x" = "x$USCENV HELLO_USC" ];then
    USCENV HELLO_USC=1
    HELLO_PREFIX=/usr/usc/hello_usc/2.0
    export USCENV HELLO_USC
    if [ "x${PATH}" = "x" ]; then
        PATH="${HELLO_PREFIX}/bin:/bin:/usr/bin:/usr/local/bin"
    else
        PATH=${HELLO_PREFIX}/bin:${PATH}
    fi
fi
A Case Study: System vs Software Repo

- Sometimes software and libraries (e.g. gcc, python, fftw) come with OS.
- Although command name is the same, the system software and repo software are often different (versions, libraries, developers). Make sure that you use what you want to use.
- `which` command shows the absolute path of a command.

```bash
$ which python
/usr/bin/python

$ source /usr/usc/python/enthought/default/setup.sh

$ which python
/usr/usc/python/enthought/default/bin/python
```
Outline

1. HPC Overview
2. Account Management
   – Directories and Quotas
   – Computing Time
3. Software Repository
4. Portable Batch System (PBS)
   – PBS Basics
   – Interactive Mode
   – Job Monitoring
Portable Batch System (PBS)

- You want to have compute nodes assigned to you
- To get compute nodes assigned to you, you will need to run the `qsub` command
- The `qsub` command is your interface into the Job scheduler, which finds unused nodes and assigns them to you based on your requirements
- If there are no free nodes your JOB gets queued waiting it’s turn

Infiniband
Myrinet

Job scheduler
hpc-login2
hpc-login3

Internet
### PBS Commands: qsub

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>qsub</code></td>
<td>submit a job to computing cluster</td>
</tr>
<tr>
<td><code>-l</code></td>
<td>resource list</td>
</tr>
<tr>
<td><code>nodes</code></td>
<td>number of nodes</td>
</tr>
<tr>
<td><code>ppn</code></td>
<td>processor per core (nodes attribute)</td>
</tr>
<tr>
<td><code>gpus</code></td>
<td>GPU node request (nodes attribute)</td>
</tr>
<tr>
<td><code>nodeattr</code></td>
<td>machine architecture (nodes attribute)</td>
</tr>
<tr>
<td><code>mem</code></td>
<td>amount of total memory</td>
</tr>
<tr>
<td><code>pmem</code></td>
<td>amount of memory per process</td>
</tr>
<tr>
<td><code>walltime</code></td>
<td>wallclock time</td>
</tr>
<tr>
<td><code>-d</code></td>
<td>starting directory</td>
</tr>
<tr>
<td><code>-A</code></td>
<td>specify your account</td>
</tr>
</tbody>
</table>
Interactive PBS Jobs

• PBS has a special job submission mode that allows a user to access allocated computing resources interactively. This is called interactive mode or interactive 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 your assigned computing nodes via ssh.
• You can run programs as many times as you want until the requested time expires. Extremely useful for compile/debug/test your code.
Interactive PBS Jobs (cont.)

-> Add \texttt{--l (eye)} option to qsub command

```bash
hpc-login3: qsub -d . -A workshop -l 'advres=HPCWorkshop.895' -l 'nodes=2:ppn=8' -l
qsub: waiting for job 11785338.hpc-pbs.hpcc.usc.edu to start
qsub: job 11785338.hpc-pbs.hpcc.usc.edu ready
```

----------------------------------------

Begin PBS Prologue Wed Apr 1 17:07:15 PDT 2015
Job ID: 11785338.hpc-pbs.hpcc.usc.edu
Username: avalonjo
...
Nodes: hpc2062 hpc2597
PVFS: /scratch (98G), /staging (328T)
TMPDIR: /tmp/11785338.hpc-pbs.hpcc.usc.edu
...
```

hpc2597: ssh hpc2062
hpc2062: hostname
hpc2062
```
Interactive PBS Job (cont.)

• While an interactive job is running, you can open another terminal, log in to headnode, then log in to the allocated nodes for the interactive job.
• Very handy to check if your job is running as you specified in your PBS script.

hpc-login3: ssh hpc2062
Last login: Wed Apr  1 17:07:52 2015 from hpc2597-e0.hpcc.usc.edu

hpc2062: head -10 /proc/cpuinfo
processor: 0
vendor_id     : GenuineIntel
cpu family : 6
model : 15
model name : Intel(R) Xeon(R) CPU E5345 @ 2.33GHz

hpc2062: head -2 /proc/meminfo
MemTotal: 12191088 kB
MemFree: 10876384 kB
Interactive PBS Job (cont.)

- You can now test your commands to see how they will run.

```bash
hpc-login3: qsub -A workshop -d . -l 'advres=HPCWorkshop.895' -l 'nodes=2:ppn=4' \ -l 'walltime=2:00:00' -l
qsub: waiting for job 11788009.hpc-pbs.hpcc.usc.edu to start
...
End PBS Prologue Thu Apr  2 10:06:10 PDT 2015
----------------------------------------
hpc2062: source /usr/usc/hello_usc/2.0/setup.sh
hpc2062: which hello_usc
/usr/usc/hello_usc/2.0/bin/hello_usc

hpc2062: pbsdsh -u /usr/usc/hello_usc/2.0/bin/hello_usc

Hello USC!!!.
I am version 2.0 running on host: hpc2062

Hello USC!!!.
I am version 2.0 running on host: hpc2081

# Try without the --u, what happens?
```
Interactive PBS Job (cont.)

- Let’s compile and run an OpenMPI program.

```
hpc2062: cd /home/rcf-proj/hpcc/avalonjo
hpc2062: mkdir Tmp
hpc2062: cd Tmp

hpc2062: cp /home/rcf-proj/hpcc/WorkshopFiles/helloWorldMPI.c .
hpc2062: cp /home/rcf-proj/hpcc/WorkshopFiles/compile.sh .
hpc2062: cat compile.sh
#!/bin/sh

CC=mpicc make helloWorldMPI

hpc2062: source /usr/usc/openmpi/1.8.4/setup.sh

hpc2062: ./compile.sh
mpicc helloWorldMPI.c -o helloWorldMPI

hpc2062: ls
compile.sh* helloWorldMPI* helloWorldMPI.c
```
Interactive PBS Job (cont.)

```bash
hpc2062: which mpiexec
/usr/usc/openmpi/1.8.4/bin/mpiexec

hpc2062: mpiexec ./helloWorldMPI
Hello World from rank 1 running on hpc2062!
Hello World from rank 2 running on hpc2062!
Hello World from rank 3 running on hpc2062!
Hello World from rank 0 running on hpc2062!
MPI World size = 8 processes
Hello World from rank 4 running on hpc2081!
Hello World from rank 5 running on hpc2081!
Hello World from rank 6 running on hpc2081!
Hello World from rank 7 running on hpc2081!
```
filesystem benchmarks (demo)

Benchmark Procedure:
Use `dd` command to measure the speed of a 1GB write on project, scratch and staging directory.

$ qsub -l 'nodes=2:ppn=16:IB' -l 'walltime=2:00:00' –l –A workshop –l ‘advres=HPCWorkshop.895’
...
Nodes: hpc3260 hpc3261
...
hpc3260: df
Filesystem 1K-blocks Used Available Use% Mounted on
... tcp://hpc-ofs03.ib.hpcc.usc.edu:3334/staging
351541493760 129451282432 222090211328 37% /staging
tcp://hpc3260.ib.hpcc.usc.edu:3334/pvfs2-fs
1781469184 1458176 1780011008 1% /scratch
almaak-08:/export/samfs-proj2/proj
171885621248 60336051072 111549570176 36% /auto/rcf-proj
filesystem benchmarks (demo)

hpc3260: cd /staging/hpcc/avalonjo/tmp
hpc3260: dd if=/dev/zero of=fileOfzeros bs=1G count=1
1+0 records in
1+0 records out
1073741824 bytes (1.1 GB) copied, 2.69193 s, 399 MB/s

hpc3260: cd /scratch
hpc3260: dd if=/dev/zero of=fileOfzeros bs=1G count=1
1+0 records in
1+0 records out
1073741824 bytes (1.1 GB) copied, 2.48319 s, 432 MB/s

hpc3260: cd /home/rcf-proj/hpcc/avalonjo/tmp
hpc3260: dd if=/dev/zero of=fileOfzeros bs=1G count=1
1+0 records in
1+0 records out
1073741824 bytes (1.1 GB) copied, 9.82488 s, 109 MB/s
Portable Batch System (PBS)

- To submit your job to the cluster, create a text file which describes the computing resources you need to accomplish your job. This text file is called **Portable Batch System (PBS) script**.
- Submit the PBS script to **job scheduler** running on the HPC cluster.
- Your job request will wait in **queue** until the requested resources become available, then the job scheduler will start your job.
On hpc-login3 using the nano editor create helloworld.PBS in your Tmp directory. Then submit using qsub helloworld.PBS

1: Set up which shell to use
2: one node with 8 procs per node
3: request for 10 minutes
5: account workshop
6: workshop reservation
8: cd to project dir.
9: blank
10: comment
11: Source setup file to use openmpi (gnu version)
12: blank
13: comment
14: run helloWorldMPI
PBS script: a bit more advanced

#!/bin/bash
#PBS -l nodes=4:ppn=16:gpus=2,pvmem=2GB  # job needs 2G per ppn
#PBS -l walltime=24:00:00
#PBS –m abe  # email sent on abort/begin/end
#PBS –M avalonjo@usc.edu  # my email address
#PBS -A lc_kn
#PBS –d /home/rcf-proj/hpcc/avalonjo  # change into this directory
#PBS -N my_mpicode  #Name of my job

# source necessary setup files for my simulation
source /usr/usc/intel/12.1.1/setup.sh
source /usr/usc/openmpi/1.6.4/share/setup-intel.sh
source /usr/usc/cuda/6.0/setup.sh

# run
mpirun –np 64 my_mpicode > log
Queues on the HPC cluster

- There are four queues available for public: **main**, **quick**, **large**, and **largemem**.
- Each queue has different constraints on max. number of queueable jobs, walltime, nodes, simultaneously runnable jobs.
- The job scheduler automatically selects which queue to be assigned on your job depending on the your request. **No need to specify queue** by users.

<table>
<thead>
<tr>
<th>Queue Name</th>
<th>Maximum Jobs Queued</th>
<th>Maximum Node Count</th>
<th>Maximum Wall Time</th>
<th>Maximum Jobs per User</th>
</tr>
</thead>
<tbody>
<tr>
<td>main</td>
<td>1000</td>
<td>99</td>
<td>24 hours</td>
<td>10</td>
</tr>
<tr>
<td>quick</td>
<td>100</td>
<td>4</td>
<td>1 hour</td>
<td>10</td>
</tr>
<tr>
<td>large</td>
<td>100</td>
<td>256</td>
<td>24 hours</td>
<td>1</td>
</tr>
<tr>
<td>largemem</td>
<td>100</td>
<td>1</td>
<td>336 hours</td>
<td>1</td>
</tr>
</tbody>
</table>

Some Examples:

Q. which queue?

#PBS -l nodes=1:ppn=2
#PBS -l walltime=00:59:59

#PBS -l nodes=20:ppn=10,
walltime=00:59:59,pmem=1gb

#PBS -N myjob
#PBS -d /home/rcf-proj/hpcc/avalonjo
#PBS -l pmem=1gb
#PBS -A workshop

#PBS -l nodes=16:ppn=12
#PBS -l walltime=23:00:00
#PBS -A workshop
#PBS -d .
## node attribute: nodetype

You can specify computer architecture by **nodetype** attribute in case your application needs to run on a certain architecture.

```bash
$qsub -l -d . -l nodes=2:ppn=8:pe1950
```

<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>sl160</td>
</tr>
<tr>
<td>hpc1044</td>
<td>hpc1050</td>
<td>7</td>
<td>Dual Dodecacle 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 Dodecacle 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 Dodecacle 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 Dodecacle 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>

[http://hpcc.usc.edu/support/infrastructure/node-allocation/](http://hpcc.usc.edu/support/infrastructure/node-allocation/)
node attribute: myri and IB

- As previously mentioned, there are two different interconnects in the HPC cluster, called **Myrinet** and **Infiniband**.
- The **Myrinet** and **Infiniband** networks are not connected to each other.
- If not specified the system will use the set of nodes that allow your job to start.
- Codes compiled to use MPICH will only run on the Myrinet nodes.
- OpenMPI codes will run on either.

```
qsub -l nodes=10:ppn=8:myri,walltime=4:00:00
qsub -l nodes=4:ppn=16:IB,walltime=8:00:00
```
Job Monitoring: qstat

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

$ qstat -u avalonjo
hpc-pbs.hpcc.usc.edu:

<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</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>9667416.hpc-pbs. avalonjo</td>
<td>main</td>
<td>testjob</td>
<td>3827</td>
<td>10</td>
<td>20</td>
<td>--</td>
<td>6:00:00</td>
<td>R</td>
<td>3:21:28</td>
<td></td>
</tr>
</tbody>
</table>
Job Monitoring: `qstat`

```bash
$ qstat main | head
```

<table>
<thead>
<tr>
<th>Job ID</th>
<th>Name</th>
<th>User</th>
<th>Time</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>9043999.hpc-pbs</td>
<td>job1</td>
<td>user1</td>
<td>0</td>
<td>Q</td>
</tr>
<tr>
<td>9447030.hpc-pbs</td>
<td>Job2</td>
<td>user2</td>
<td>0</td>
<td>Q</td>
</tr>
<tr>
<td>9629959.hpc-pbs</td>
<td>...job.pbs</td>
<td>user3</td>
<td>0</td>
<td>Q</td>
</tr>
<tr>
<td>9629975.hpc-pbs</td>
<td>...job.pbs</td>
<td>user3</td>
<td>0</td>
<td>Q</td>
</tr>
<tr>
<td>9633223.hpc-pbs</td>
<td>job3</td>
<td>user4</td>
<td>0</td>
<td>Q</td>
</tr>
<tr>
<td>9653476.hpc-pbs</td>
<td>...job.pbs</td>
<td>user3</td>
<td>0</td>
<td>Q</td>
</tr>
<tr>
<td>9676843.hpc-pbs</td>
<td>test.pbs</td>
<td>user5</td>
<td>169:54:2</td>
<td>R</td>
</tr>
<tr>
<td>9679200.hpc-pbs</td>
<td>rsync</td>
<td>user6</td>
<td>10:17:15</td>
<td>R</td>
</tr>
</tbody>
</table>
```
# Job Monitoring: myqueue

**myqueue**

Display jobs status and allocated node list for your running jobs.

<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</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>11788258.hpc-pbs.hpcc. avalonjo main STDIN</td>
<td>32291</td>
<td>2</td>
<td>8</td>
<td>--</td>
<td>02:00:00</td>
<td>R</td>
<td>00:30:44</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>hpc2062/0</td>
<td>hpc2062/1</td>
<td>hpc2062/2</td>
<td>hpc2062/3</td>
<td>hpc2081/0</td>
<td>hpc2081/1</td>
<td>hpc2081/2</td>
<td>hpc2081/3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

hpc-login3:
Job Monitoring: showstart

**showstart**
Displays approximate start time for your job.

**checkjob**
Displays certain system properties for your job.

# Look for ‘HOLDs’ on your job

```
hpc-login3: checkjob 11788258
job 11788258
...
WallTime: 00:33:28 of 2:00:00
SubmitTime: Thu Apr 2 10:37:24
...
StartPriority: 1
Reservation '11788258' (-00:33:28 -> 1:26:32 Duration: 2:00:00)
```

$ showstart 11788258
# Place this in your .bashrc file `export CLIENTTIMEOUT='00:10:00'`
Outline

1. ...
2. ...
3. ...
4. ...
5. **Software Installation**  
   *eg. python installation*
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/myaccnt
# 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/myaccnt/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

prefix=/home/rcf-proj/projectid/myaccnt/python/2.7.6/
export PATH=${prefix}/bin:$PATH (bash)
setenv PATH=${prefix}/bin:$PATH (tcsh)
Want to learn more?

- Up and Running with Bash Scripting
- Unix for Mac OS X Users
- Using Regular Expressions
- Perl 5 Essential Training
- R Statistics Essential Training
- C/C++ Essential Training
- Up and Running with Python
- Python 3 Essential Training
- Up and Running with MATLAB
- Up and Running with R and More!

From: Unix for Mac OS X Users
Want to learn more?

http://software-carpentry.org
A special thanks to:

Dr. Ken-Ichi Nomura Ph.D.

for the original version of these slides
Thank you!