

Introduction to High Performance Computing and Orchestra

HMS Research Computing

Spring 2016

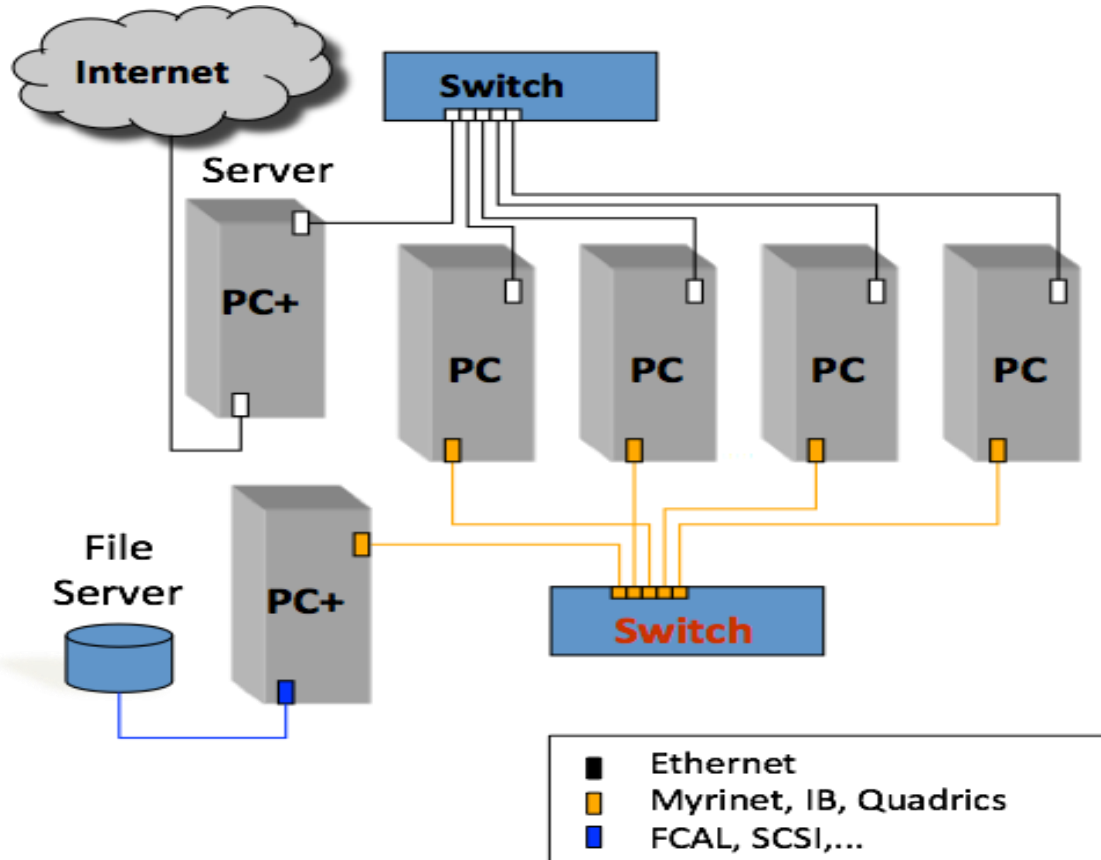


What is Orchestra?

- Wiki page: <https://wiki.med.harvard.edu/Orchestra>
- Tech spec:
 - Over 550 compute nodes
 - Over 7500 cores
 - 10GigE interconnection
 - Over 40TB RAM
- CentOS 6 Linux
- LSF scheduler
- Total 28+PB storage



Generic Cluster Architecture



- Login nodes are for data management, job submission, code development, etc.
- Compute nodes are for production runs
- Your storage space is centralized network storage system
- Hundreds users are sharing the resources, so please be nice each other.
- Job scheduler works based on complex algorithms of fair-share, priority management, load balancing, etc.

Orchestra Compute Nodes

NODE CLASS	CPU TYPE	Number of Nodes	Cores x Node	MEMORY x Node (~GB)	Total Cores
Flute	Opteron2216	25	4	32 (vary)	100
Bassoon	XeonE5345	43	8	32 (vary)	344
Bassoon	XeonE5430	12	8	32	96
Sax	XeonE74807	2	16	1024	32
Clarinet	XeonE5530	21	8	vary	168
Clarinet	XeonL5640	294	12	96 (vary)	3528
Fife	XeonE52680	16	20	190	320
Piccolo	XeonE5266v2	40	20	128	800
Ocarina	Opteron6376	8	32	512	512
Ottavino	XeonE52697	16	28	252	448



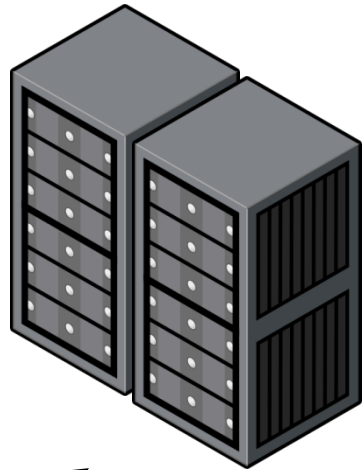
Orchestra Storage

Orchestra Cluster

- 7500+ cores
- LSF batch system



Your computer



/home

- /home/user_id
- quota: 100GB per user
- Backup: 14-day daily, 60-day weekly



/groups, /n/data1, /n/data2

- /groups/your_lab/your_dir
- quota: expandable
- Backup: 14-day daily, 60-day weekly



/n/scratch2

- /n/scratch2/user_id
- quota: 5TB per user
- Backups: none
- Run workflows here

scratch2: It's here!

/n/scratch2 (DDN ExaScalar2)

- High-performance parallel file system: Lustre
- 1PB of disk space
- 5TB quota, no backup, data will be purged after 15 days
- Ideal for large-scale job running

Checking Storage Usage

- To check your storage available:

`mfk8@loge:~$ quota`

Home directory: you get 100 GB, total.

Group directories: space varies, can increase.

`/groups/groupname`

`/n/data1`

`/n/data2`

Checking Storage Usage: scratch2

- `mfk8@loge:~$ lfs quota -h /n/scratch2`
- Quota is on user basis, not group basis
- Users are entitled to 5TB and up to 1 million files/directories

Storage Policies

- /home: 14 day snapshots + 60 day full backup
- /groups, /n/data1, /n/data2: 14 day snapshots + 60 day full backup
- /n/scratch2: 15 day retention, no backups
- New tape system for long-term storage

Snapshots

- Snapshots are retained for up to 60 days: recover data
- `mfk8@clarinet001:~$ cd .snapshot`
- `mfk8@clarinet001:~$ ls`

Orchestra_home_daily_2015-10-02-02-00

Orchestra_home_daily_2015-10-01-02-00

- `mfk8@clarinet001:~$ cd Orchestra_home_daily_2015-10-02-02-00`
- `mfk8@clarinet001:~$ cp MyRetreivedFile ~`

Create an Orchestra Account

- <http://rc.hms.harvard.edu/#orchestra>

Click the **red button** and fill out the form!

- Your username will be your eCommons ID, with your eCommons password.

LSF: Fair Sharing

- **Load Sharing Facility:** distributes jobs across Orchestra fairly
- Ensures that no one user or core monopolizes Orchestra
- Users are assigned dynamic priorities
- Queues also have priorities
- Submitting lots of jobs reduces your fairshare priority
- Even if many jobs are pending, your jobs will start quickly provided you have not submitted many jobs

Logging Into Orchestra: Mac

- Open a terminal (search “terminal”)

`ssh -l username orchestra.med.harvard.edu`

- For graphics (X11 forwarding)

Install Xquartz (google it) and have it running

`ssh -X -l username orchestra.med.harvard.edu`

Logging Into Orchestra: Windows

- Install PuTTY (google it)
- In box under Host Name (or IP address)

orchestra.med.harvard.edu

- For graphics (X11 Forwarding)

Install Xming (google it) and have it running

In PuTTY, under Connection -> SSH -> X11:

Check “Enable X11 Forwarding” THEN log in

Logging Into Orchestra: Linux

- Open a terminal (search: “terminal”)

`ssh -l username orchestra.med.harvard.edu`

For graphics (X11 Forwarding)

`ssh -X -l username orchestra.med.harvard.edu`

Welcome to Orchestra!

- Where are you in Orchestra?

See your terminal!

mfk8@mezzanine: ~\$

- You are logged into a “shell login server.” Usually “mezzanine” or “loge.” These are not meant for heavy lifting!
- You are in your home directory. This is symbolized by the “tilde ” (~). This is shorthand for: /home/username
- You are in a bash environment. “\$” means “ready to accept your commands!”

Interactive Sessions

- The login servers are not designed to handle intensive processes. Using 40% of a CPU results in a process being terminated. It is sometimes better to use an “interactive session.” These last up to 12 hours. Start by entering your first job! This will (usually) log you into a “clarinet!”

```
mfk8@mezzanine:~$ bsub -Is -q interactive bash
```

“bsub” is how jobs are submitted to Orchestra LSF

-Is (capital eye, s) tells Orchestra you want an interactive shell

-q is the “queue” to enter (interactive)

“bash” opens the bash session

```
mfk8@clarinet001:~/$
```

Listing a Folder's Contents

- To see the contents of the current folder you are in (~ means “/home/username/”), type **list** (ls):

```
mfk8@clarinet001:~$ ls
```

- To get the details of a folder's contents, add “-l”

```
mfk8@clarinet001:~$ ls -l
```

- You don't have to be in a directory to see its contents

```
mfk8@clarinet001:~$ ls /groups/rc-training/introthpc
```

Viewing File Contents

- “less” to view file contents
- Navigate up/down, search
- “q” to quit

mfk8@clarinet001:~\$ less /groups/rc-training/introthpc/testfile.txt

Making a Folder (Directory)

- “mkdir” stands for “**make directory.**”
- Create a new directory for this exercise
- Spaces are discouraged. (Underscores are fine!) Case counts in Linux.

```
mfk8@clarinet001:~$ mkdir MyTestDir
```

Moving Around: Change Directory

- “cd” stands for “**c**hange **d**irectory”
- 1 period “.” means “current directory”
- 2 periods “..” means “the directory above”

```
mfk8@clarinet001:~$ cd MyTestDir
```

Notice how the prompt tells you where you are!

```
mfk8@clarinet001:~/MyTestDir$ cd ..
```

```
mfk8@clarinet001:~$
```

Creating a Simple Text File

- “Nano,” “vi”, “emacs” are simple command-line editors available.
- To create a new file, type the editor you want, then the name of the new file. To edit an existing file, do the same.

```
mfk8@clarinet001:~$ nano myfile.txt
```

```
This is my new file text.
```

(Control-X to save (yes) and exit.)

```
mfk8@clarinet001:~$
```

```
mfk8@clarinet001:~$ ls
```

```
myfile.txt
```

Copying Files

- “cp” to **copy** a file from a destination to a new destination. “cp” “from” “to”

```
mfk8@clarinet001:~$ cp myfile.txt MyTestDir/
```

- You can copy a file to the current folder or to a new folder with a different name by specifying a different name (rename)

```
mfk8@clarinet001:~$ cp myfile.txt MyTestDir/myfilecopy.txt
```

```
mfk8@clarinet001:~$ cp myfile.txt mycopy2.txt
```

```
mfk8@clarinet001:~$ cp /groups/rc-training/introthpc/testfile.txt ~/MyTestDir/
```

Copying Directories

- “cp” “options” “source” “destination”
 - Options:
 - r: recursively (copies all)
- mfk8@clarinet001:~\$ cp -r MyTestDir MyTest2

Moving Data

- “move” “from” “to”

```
mfk8@clarinet001~:$ mv MyTestDir/myfilecopy.txt ~
```

```
mfk8@clarinet001~:$ mv MyTestDir/ MyTestDir2/
```

Removing Files/Folders

- “rm” to remove a file

```
mfk8@clarinet001:~$ rm myfile.txt
```

- “rm -r” to remove a folder recursively

```
mfk8@clarinet001:~$ rm -r MyTest2
```

Wildcard * Pattern Matching

- Useful for copying/removing/etc all files matching a certain pattern
- Example Case:

To copy “all” files ending in “.fastq”:

```
$ cp *.fastq NewFastqFolder
```

Getting Data Onto Orchestra

- Use an FTP client of your choice
- Mac/Windows/Linux: Filezilla (google it)
- Connect to:

orchestra.med.harvard.edu

your username and password

port 22

Using Software: Environment Modules

- Most “software” on Orchestra is installed as an environment module. Allows for clean, easy loading, including most dependencies, and switching versions.

\$ module avail

\$ module avail seq/

\$ module avail stats/

\$ module avail seq/bowtie/

Loading/Unloading Modules

- Loading modules
`$ module load seq/bowtie/2.0.6`
- Which module version is loaded (if at all)?
`$ which bowtie`
- See all modules loaded
`$ module list`
- Unloading modules
`$ module unload seq/bowtie/2.0.6`
- Dump all modules
`$ module purge`

Compiling your own software

- Users can compile software in their /home or /groups directories, where they have permission
- Binaries just require “unzipping” (ie tar -zxvf .tgz)
- Common compiling libraries are found as modules:

dev/compiler/gcc-4.8.5

dev/boost/1.57.0

dev/openblas/0.2.14

dev/lapack

- Common libraries are found in “utils”

Installing Software: Binary Example

- `mfk8@loge:~$ bsub -ls -q interactive bash`
- `mfk8@clarinet001:~$ wget http://path/to/binary/mysoftware.tar.gz`
- `mfk8@clarinet001:~$ tar -zxvf mysoftware.tar.gz`
- `mfk8@clarinet001:~$ ls mysoftware/bin`

Installing Software: Source

- `mfk8@loge:~$ bsub -ls -q interactive bash`
- `mfk8@clarinet001:~$ module load dev/compiler/gcc-4.8.5`
- `mfk8@clarinet001:~$ wget http://path/to/source/mysoftware.tar.gz`
- `mfk8@clarinet001:~$ tar -zxvf mysoftware.tar.gz`
- `mfk8@clarinet001:~$ cd mysoftware`
- `mfk8@clarinet001:~$ less README`
- `mfk8@clarinet001:~$./configure --prefix=/home/mfk8/software`
- `mfk8@clarinet001:~$ make`
- `mfk8@clarinet001:~$ make install`

Python, R, Perl

- Users manage their own package libraries: get the version you want, when you want it!
- Python: virtual environment allows pip installs

<https://wiki.med.harvard.edu/Orchestra/PersonalPythonPackages>

- R: set up personal R library for cran, Bioconductor

<https://wiki.med.harvard.edu/Orchestra/PersonalRPackages>

- Perl: local::lib allows cpan, cpanm

<https://wiki.med.harvard.edu/Orchestra/PersonalPerlPackages>

- Shebangs: `#!/usr/bin/env python/Rscript/perl`

Submitting Jobs

- In an “interactive session”, programs can be called directly.

```
mfk8@clarinet001:~$ bowtie -p 4 hg19 file1_1.fq file1_2.fq
```

- From the login shell (and also interactive or any compute nodes), a program is submitted to Orchestra via a job (bsub).

```
mfk8@clarinet001:~$ bsub < mybowtiejob.sh
```

- Orchestra will notify you when the job is done, or if there is an error.

The “bsub”

```
mfk8@clarinet001:~$ bsub -q queue -W hr:min job
```

- Necessary:
 - q (queue)
 - W (runtime in hr:min)

Shared Queues

- *mpi* queue if you have an MPI parallel job
- *priority* queue if you have just one or two jobs to run
- *mcore* queue if you have multi-core jobs to run.
- *short* queue if your jobs will take less than 12 hours to run.
- Else: *long* queue.

Shared Queues Breakdown

Queue Name	Priority	Max Cores	Max Runtime
interactive	12	12	12 hours
priority	14	12	1 month
mcore	12	20	1 month
mpi	12	400	1 month
parallel	10	400	1 month
short	8	12	12 hours
long	6	12	1 month
mini	5	1	10 minutes

Runtime Limit

- -W in hours:minutes
- Runtimes are subject to the maximum time permitted per queue (see table)
- If your job exceeds your runtime, your job will be killed ☹️
- Running many jobs that finish quickly (less than a few minutes) is suboptimal and may result in job suspension, contact RC to learn how to batch jobs

CPU Limit

- Amount of seconds the cluster works on your job (calculated by LSF)
- Ncores * Runlimit (-n * -W)
- Common error:

```
bsub -q short -W 8:00 tophat -p 8
```

tophat asks for 8 cores but only 1 requested (no -n), job killed in 1 hour

Multithreading

- A single CPU can execute multiple processes (threads) concurrently
- -n indicates how many cores are requested
- Jobs that are overefficient (use more cores than reserved) jeopardize the health of a node
- Reserve the same amount of cores in your job and your bsub!

Reserving Memory

- Most nodes have 90GB memory available over all cores, some have more
- Make a resource request with
 - R “rusage[mem=16000]” (memory requested in MB)
- Memory multiplies by cores requested, so
 - n 4 –R “rusage[mem=16000]” reserves 64GB memory
- Asking for more memory may cause jobs to pend longer
- TERM_MEMLIMIT errors indicate that not enough memory was reserved

MPI

- Efficient way to run parallel jobs
- Dedicated queue with same-type compute nodes for optimal performance (400 cores)
- Matlab, Python, Java, R, C++, Fortran have MPI options
- Orchestra implementation: openMPI-1.8.6
- Talk to us first!

Other bsub options

- n 4 (*number of cores requested*)
- R "rusage[mem=16000]" (*memory requested in MB*)
- J jobname
- a openmpi (type of mpi run)
- ls (interactive shell)
- e errfile (*send errors to file*)
- o outfile (*send screen output to file*)
- N (notify when job completes)

Monitoring Jobs

- List info about jobs/their status:

`mfk8$loge:~$ bjobs`

-r (running jobs)

-p (pending jobs)

-l (command entered, long form)

- List historical job information

`mfk8$loge:~$ bhist jobid`

Job Suspensions

- Jobs in the “long” queue can be suspended by LSF temporarily to allow jobs from the “short” queue (less than 12h) to run.
- If a job is suspended for >50% of the `-W` runlimit cumulatively, it will be automatically moved to the “no_suspend” queue where it can’t be suspended anymore.

Terminating Jobs

- Terminate a job (jobid given at submission)

`mfk8$loge:~$ bkill jobid`

- Terminate all jobs (be careful!)

`mfk8$loge:~$ bkill 0`

Shell Scripts: The Basics

```
#!/bin/sh #always at the top
```

```
#program with options
```

```
module load seq/tophat/2.1.0 seq/bowtie/2.2.4 seq/samtools/1.2
```

```
tophat -p 4 -o ./mytophatdir1 hg19 file1_1.fastq file1_2.fastq
```

```
tophat -p 4 -o ./mytophatdir2 hg19 file2_1.fastq file2_2.fastq
```

```
#save as myshellscript.lsf
```


Calling a Shell Script

```
mfk8@balcony:~$ bsub -q mcore -W 12:00 -n 4 -e %J.err -o  
%J.out -N sh myshellscript.lsf
```

Creating a Complete Shell Script

```
#!/bin/sh
#BSUB -q mcore
#BSUB -W 12:00
#BSUB -o %J.out
#BSUB -e %J.err
#BSUB -N
#BSUB -n 4
#BSUB -R "rusage[mem=12000]"
module load seq/tophat/2.1.0 seq/bowtie/2.2.4 seq/samtools/1.2
tophat -p 4 -o ./mytophatdir1 hg19 file1_1.fastq file1_2.fastq
#save as myshellscript2.lsf
```

Calling a Complete Shell Script

```
mfk8@balcony:~$ bsub < myshellscript2.lsf
```

Using /n/scratch2

- Designed for writing large, temporary files
- Use cases:
- Keep original files in /groups (/n/data*) or /home, write intermediate files to /n/scratch2, write final files to /groups (/n/data*) or /home
- Change working directory to /n/scratch2, read files from /groups (/n/data*) or /home, write temp files to working directory, write or copy output back to /groups (/n/data*) or /home
- Copy input files to /n/scratch2, compute against, copy output files to /groups (/n/data*) or /home

File Properties

- “chmod” to change who can read/write/execute files/directories

chmod options file/directory

Who? **u**ser **g**roup **o**thers **a**ll (u/g/o/a)

What? **r**ead **w**rite **x**ecute (r/w/x)

Do? +/-

- chmod u+x myfile #makes a file executable to owner
- chmod o-rwx myfile #takes away permission from others to read/write execute

Class Exercises

- In your “home” directory, create a file called “.forward” using nano with this piece of information: your preferred email address for Orchestra notifications. Save.
- Copy the .bam file from /groups/rc-training/introthpc to your “home” directory, load “samtools” module, and submit a job to the “short” queue for “1 hr” that has this command:

samtools sort control.bam sorted.bam
- Monitor this job, then kill it
- Delete all class files (except .forward)

For more direction:

- <https://wiki.med.harvard.edu/Orchestra/NewUserGuide>
- <http://rc.hms.harvard.edu>
- rchelp@hms.harvard.edu
- Office Hours: Wednesdays 1-3p Gordon Hall 500
- Class survey:
<http://hmsrc.me/introhpc2016-survey4>