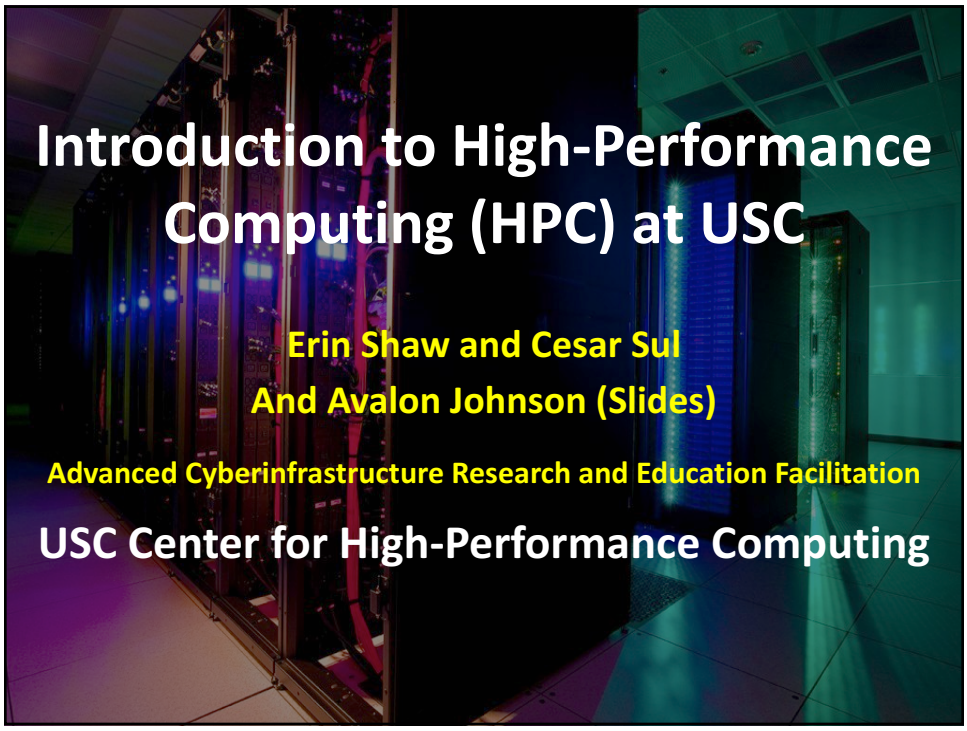


Introduction to High-Performance Computing (HPC) at USC

Erin Shaw and Cesar Sul
And Avalon Johnson (Slides)

Advanced Cyberinfrastructure Research and Education Facilitation

USC Center for High-Performance Computing

Outline

- HPC Overview (NewUser)
- Account Management
  - Directories, Quotas, Computing Time
- Software Repository
- Portable Batch System (PBS)
  - PBS Basics
  - Job Monitoring

What is HPC?

- USC's Center for High-Performance Computing
- HPC advances USC's mission by providing the infrastructure and support necessary for **research computing**
  - HPC's resources are available at no charge to USC faculty, researchers, and graduate students
  - HPC is housed within the ITS data center and is monitored around-the-clock by ITS staff
  - HPC exists to help you advance scientific discovery
- HPC is a world-class super-computing center!
  - We do LINPACK benchmarking for Top500 Supercomputers ranking
  - HPC ranked 12th fastest academic supercomputer in U.S. (June 2016)

USC University of Southern California

USC ITS Information Technology Services

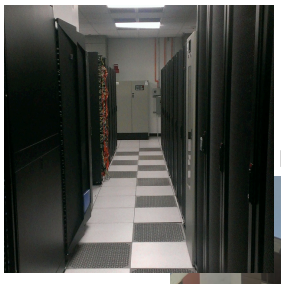
HPC under ITS (Information Technology Services)

Douglas Shook
USC CIO

Randolph Hall
*USC VP of Research & Faculty Executive Director, HPC**

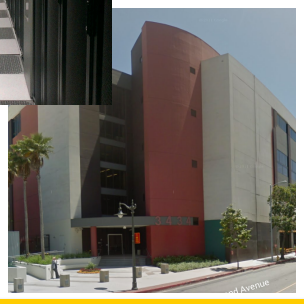
Maureen Dougherty
USC HPC Director

*The HPC Faculty Advisory committee advises the CIO about the faculty's research needs related to the university's HPC resources.



HPC

ITS Data Center



USC University of Southern California

USC ITS Information Technology Services

HPC Facilitation

- Request assistance
  - Email hpc@usc.edu (email again!)
  - Drop-in to Office Hours
    - *Every Tuesday@2:30pm (UPC LVL 3M)*
    - *After workshops or by appt. (HSC NML 2*
  - Request a lab/individual consultation
- Learn more!
  - Visit https://hpcc.usc.edu
  - Attend a New User meeting
  - Attend a Workshop
    - *Friday Afternoons, 2:30-4:30pm*

USC University of Southern California

USC ITS Information Technology Services

Computing Cluster

- Wikipedia Definition

A **computer cluster** consists of a set of
… connected computers that work together
… connected to each other through fast local area networks
… with each node running its own instance of an operating system
… software for high-performance distributed computing

USC University of Southern California

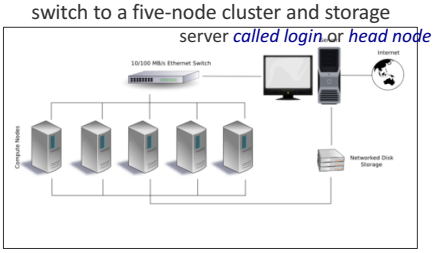
USC ITS Information Technology Services

Computing Cluster

A simple, home-built cluster

One rack in HPC cluster

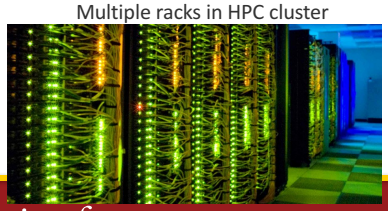
Internet-connected server with Ethernet (network) switch to a five-node cluster and storage server *called login or head node*



Rows of racks in HPC cluster!

Network cables

Multiple racks in HPC cluster



USC University of Southern California

USC ITS Information Technology Services

HPC Computing Resources

- The HPC cluster consists of almost 3000 *nodes*
  - Nodes are servers (in HPC-speak)
  - Servers are computers that communicate with other servers
- HPC has several types of nodes
  - head, or login, nodes (*hpc-login2, hpc-login3) – for logging in*
  - data transfer node *(hpc-transfer) – for fast upload/download*
  - compute nodes *(hpcxxxx) – for running jobs*
  - administrative nodes – *for administration*
- Notes
  - Head nodes are shared by all users
  - Only head nodes can access the Internet

USC University of Southern California

USC ITS Information Technology Services

HPC Computing Resources

- In Summary
  - Over 2,700 computing nodes (32K CPU cores) on 10Gb/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 BGFS
  - Cent OS 7.2 Linux, Torque and Moab for resource management and scheduling
  - Scientific software and libraries

USC University of Southern California

USC ITS Information Technology Services

HPC Computing Resources

- HPC has two Linux clusters
  - Infiniband (IB) (~1300 nodes) and Myrinet (~1700 nodes)
  - Newest cluster is on a 56.6 Gbps (IB) backbone and consists of:

    264 *Hewlett-Packard SL250, dual Xeon 8-core 2.6GHz, dual NVIDIA K20 GPUs containing 2,496 cores, each with 64GB memory*

    448 *Hewlett-Packard SL230, dual Xeon 8-core 2.6GHz CPUs, with 64GB memory*

    288 *Lenovo nx360m5 dual Xeon 8-core 2.6GHz CPUs with 64GB memory*

    19 *Lenovo nx360m5 2.6GHz dual NVIDIA K40 GPUs containing 2,880 cores, each with 64GB memory*

    5 *Lenovo nx360m5 2.6GHz dual NVIDIA K80 GPUs containing 2 x 2,496 cores, each with 64GB memory*
- Run jobs on compute nodes!

USC University of Southern California

USC ITS Information Technology Services

HPC Computing Resources (June 2016)

queue	nodes	ppn	gpus	avx	/tmp	core	cpu	model	net-work	node names	node type
arge-mem	4	40	-	-	1.8T	decacore	xeon	r910	myri	hpc-lt-1 hpc-lt-2 hpc-lt-3 hpc-lt-4	1
large main quick	8	12	-	-	140 GB	hexcore	xeon	sl160	myri	hpc0965-0972	2
large main quick	67	24	-	-	895 GB	dodeca-core	xeon	dl165	myri	hpc0981-1021 hpc1044-1050 hpc1123-1128 hpc1196-1200 hpc1223-1230	3
large main quick	26	8	-	-	60 GB	dualcore	opteron	x2200	myri	hpc1723-1728 hpc1734-1739 hpc1741-1742 hpc1744-1754 hpc1756	4
large main quick	54	8	-	-	60 GB	quadcore	xeon	pe1950	myri	hpc2283-2318 hpc2320-2337	5
large main quick	138	8	-	-	60 GB	quadcore	opteron	x2200	myri	hpc2349-2370 hpc2470 hpc2472-2481 hpc2483 hpc2486-2505 hpc2510-2544 hpc2546-2559 hpc2561-2580 hpc2582-2597 hpc2600	6
large main quick	4	12	-	-	200 GB	hexcore	xeon	dx360	myri	hpc2758-2761	7
large main quick	237	16	2	avx	850 GB	octocore	xeon	sl250s	IB	hpc3025-3027 hpc3031-3264	8

Accessing HPC from a Laptop or Desktop

- A *secure network* is required
  - Use USC Secure Wireless or USC Ethernet to connect from USC
  - Use a Virtual Private Network (VPN) client to connect from outside USC
    - *Download from https://software.usc.edu/vpn/*
- A *secure shell* (ssh) is required (a shell is Linux's command line interface)
  - On Macs, use Terminal, a native application
  - On Windows, install X-Win32 from software.usc.edu
    - *Or install another personal favorite, e.g. PuTTY, CyberDuck, etc.*
- To connect
  - On a Mac Terminal, type "ssh <YourUSCNetId>@hpc-login2.usc.edu"
  - On Windows, configure the ssh connection for hpc-login2.usc.edu

Accessing HPC from a Laptop or Desktop

- A *secure file transfer protocol* (sftp) is required for transferring data files
  - There are many sftp client application available, e.g.,
    - *Filezilla is available for both Mac and Windows*
    - *See https://itservices.usc.edu/sftp/ for options*
    - *Choose your favorite*
  - Or use the Linux commands scp and rsync for command line transfers
- To connect
  - Configure the sftp connection for hpc-transfer.usc.edu
    - *hpc-transfer is a dedicated DTN (data transfer node)*

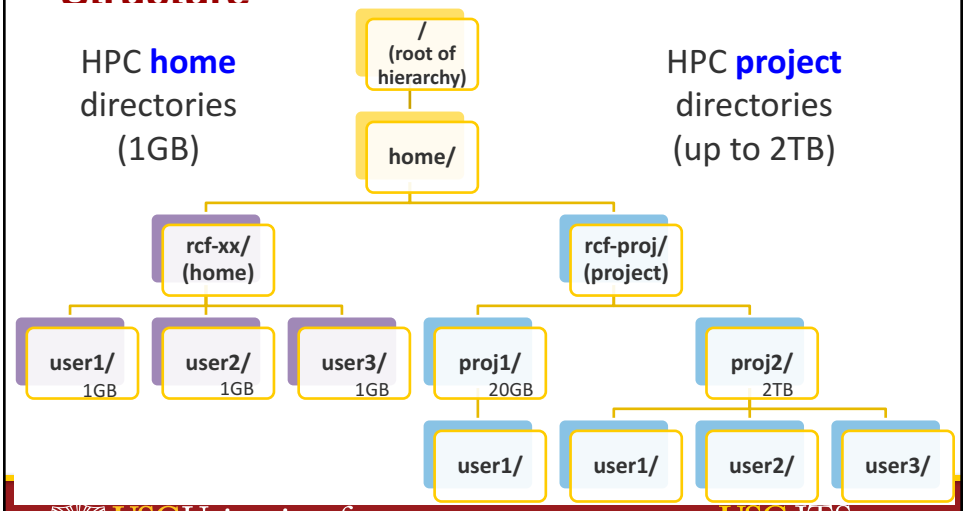
HPC File System

- Each user has a 1GB home directory that they login to
  - Backed up daily
- Projects can have up to 2TB of disk space
  - Backed up daily
  - Shared by all members
- Users have access to a 328TB data staging directory
  - No data backup!
  - Cleaned during semi-annual downtimes or when capacity reached

HPC File System: Hierarchical Directory Structure

HPC **home** directories (1GB)

HPC **project** directories (up to 2TB)

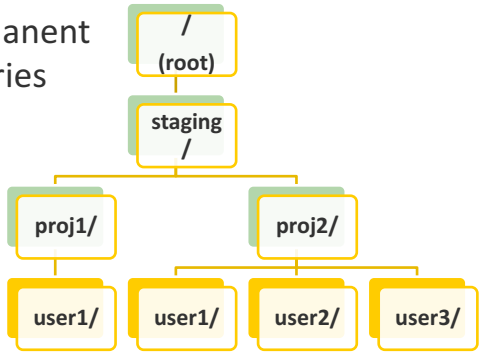


USC University of Southern California

USC ITS Information Technology Services

HPC File System: Hierarchical Directory Structure

Global, semi-permanent **staging** directories (328 TB)

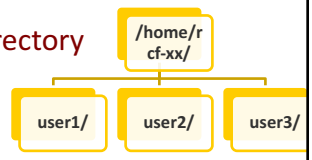


USC University of Southern California

USC ITS Information Technology Services

File System: User Directory

- Every user logs in to his/her own home directory
  - Located at /home/rcf-40/<username>
  - User quotas
    - *1 GB of disk quota and 100,000 files of file quota*
  - Private directory, only user can modify files
  - Directory is used for login, environment setup and administration
    - *Not for computation or large storage!*
    - *You will usually go directory to your project directory to work*
  - Applications may install hidden files here!

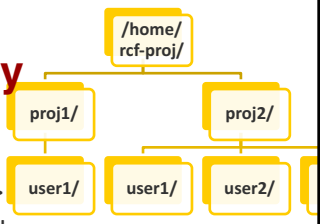


USC University of Southern California

USC ITS Information Technology Services

File System: Project Directory

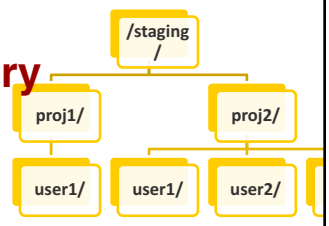
- Every project has its own directory
  - Located in /home/rcf-proj/<projectname>
  - Project quotas are shared among all members
- PIs own the project directory
  - Each user has their own directory within a project directory
  - Only PI can create shared directories and install software at top level
- Default permissions are set
  - Project directories have group read access
  - Members can make their directories private
  - Member must never make their directories open to everyone



USC University of Southern California

USC ITS Information Technology Services

File System: Staging Directory

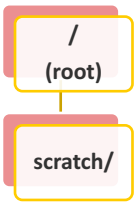


- Every project has its own staging directory
  - Located in /staging/<projectname>
  - Has same structure as project directory
  - Parallel file system has faster r/w access rates than project file system
  - Good for applications with high-frequency data access (read/write), Move results back to project after job
  - No quotas!!!
  - No backups!!!
  - Reminder:
    - *– /staging is cleared during semi-annual down-times*
      (we send many, many reminders about removing data before down-time)

USC University of Southern California USC ITS Information Technology Services

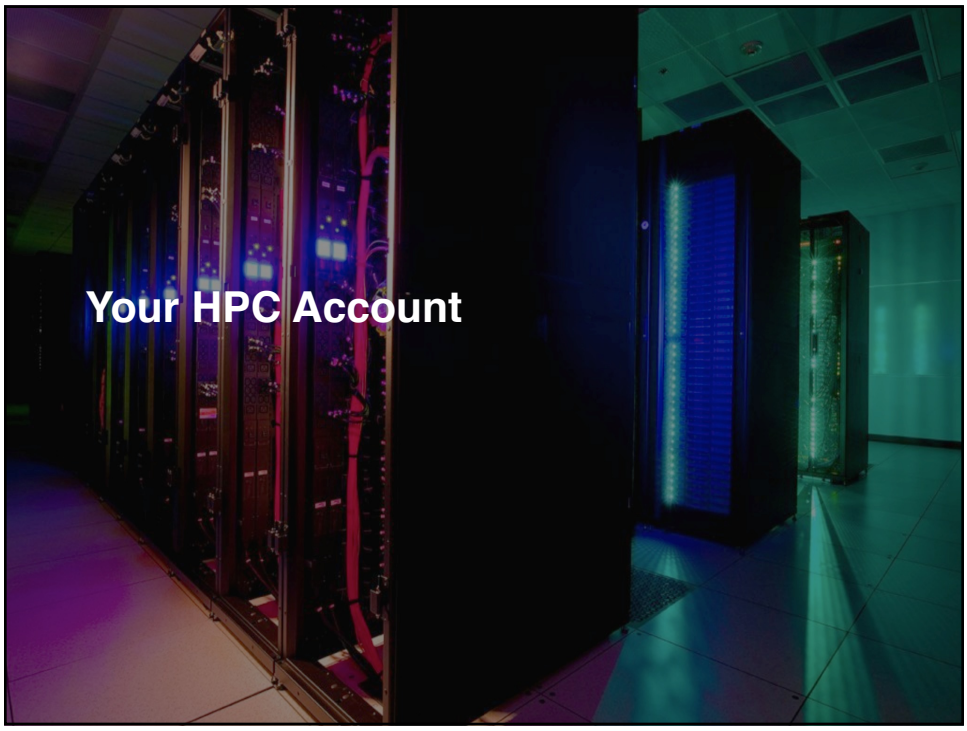
File System: Local storage

Local, temporary **scratch** directory when on compute nodes (60 GB–1.8 TB)



- Location is /scratch
- Parallel file system (fastest)
- Combines all /tmp/{jobid} space on all nodes being used
- Space available to all nodes running job
- Only accessible while you are running on compute nodes
- No Backups!!!
- All files are cleaned at job completion

USC University of Southern California USC ITS Information Technology Services



Project Accounts

- A faculty member, researcher or graduate student can apply for a up to two HPC *project* accounts
  - HPC refers to this person as the **PI** of the project
  - The PI of a project can add **group members** to their projects
    - *E.g., A professor adds students to a class project*
    - *E.g., A investigator adds graduate students to a research project*
  - Individual members can belong to multiple projects, including their own
- All projects are allocated a core hours and disk space quota
  - Use $mybalance/$myquota to monitor compute hours/disk space
  - PI can ask HPC to increase hours and space through the web site

Monitor your disk quota

myquota -displays quotas for your home & project directories

```
$ myquota
---------------------------------------------
Disk Quota for /home/rcf-40/avalonjo ID 203387

             Used       Soft       Hard
   Files     9501       100000     101000
   Bytes     721.41M    1.00G      1.00G
---------------------------------------------
Disk Quota for /home/rcf-proj2/hpcc ID 419

             Used       Soft       Hard
   Files     502016     1000000    1100000
   Bytes     433.86G    500.00G    502.00G
---------------------------------------------
```

- **Files** for file quota.
- **Bytes** for disk quota.
- **Hard** quota is the absolute limit you can store.

- If you need more space in project directory, submit a request from the hpcc.usc.edu allocation page

Monitor your disk quota

- Potential problems
  - 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!

Monitoring your 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 nodes for 2 hours (-l nodes=2:ppn=4,walltime=2:00:00), the total charge is 2x4x2 = 16 core-hours

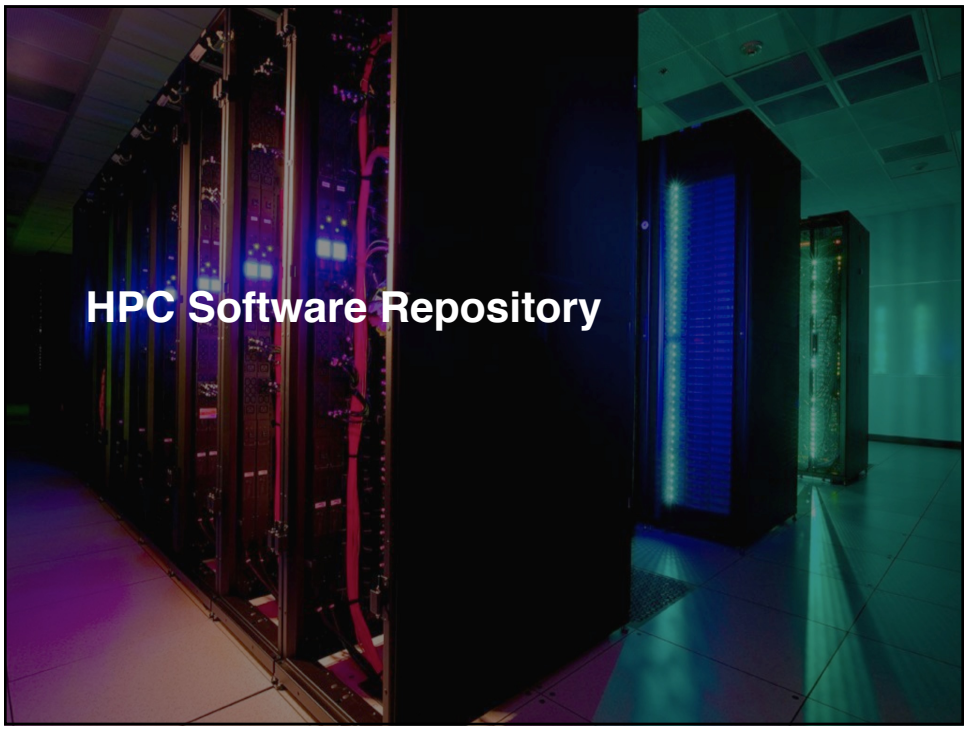
Monitoring your computing time

mybalance -shows current balance of all project accounts

```
$ 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.
- If you have multiple accounts (class, lab) specify account name in your PBS script by -A option, e.g. -A lc_kn1.
- If your job doesn't start it's always a good idea to check if your project has enough balance.

USC University of Southern California
USC ITS Information Technology Services



Software repository: /usr/usc

- HPC installs and maintains university licensed and other commonly used software in /usr/usc
  - 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*
- You can also install software in your project directory
  - HPC can help with this

Software repository: /usr/usc

- Let's take a field trip to the software repository

```
$ cd /usr/usc
$ ls -F
```

acml/	fftw/	imp/	mpich2/	qespresso/
amber/	gaussian/	intel/	mpich-mx/	qiime/
aspera/	gflags/	iperf/	mvapich2/	R/
bbcp/	git/	java@	NAMD/	root/
bin/	globus/	jdk/	ncview/	sas/

....

- Tree is a handy for viewing a directory

```
$ tree - <choose a directory>
```


Software repository: Example

- Let's run hello_usc

```
$ ls /usr/usc/hello_usc
1.0/ 2.0/ 3.0/ @default
$ ls /usr/usc/hello_usc/default
bin/  setup.csh  setup.sh
$ ls /usr/usc/hello_usc/default/bin
hello_usc*
```

Software repository: Source setup script

- What happens when I run the program?

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

- Bash cannot find a program named hello_usc because it is not is your path ($PATH)

```
$ which hello_usc
```

- "Which" searches $PATH

Software repository: Run the program

- Setup scripts add a directory to your path

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

$ which hello_usc
/usr/usc/hello_usc/2.0/bin/hello_usc

$ hello_usc
Hello USC!!!
I am version 2.0 running on host: hpc-login3
```

Software repository: Setup file

```
$ 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
```


Software: system vs. /usr/usc programs

- Try this

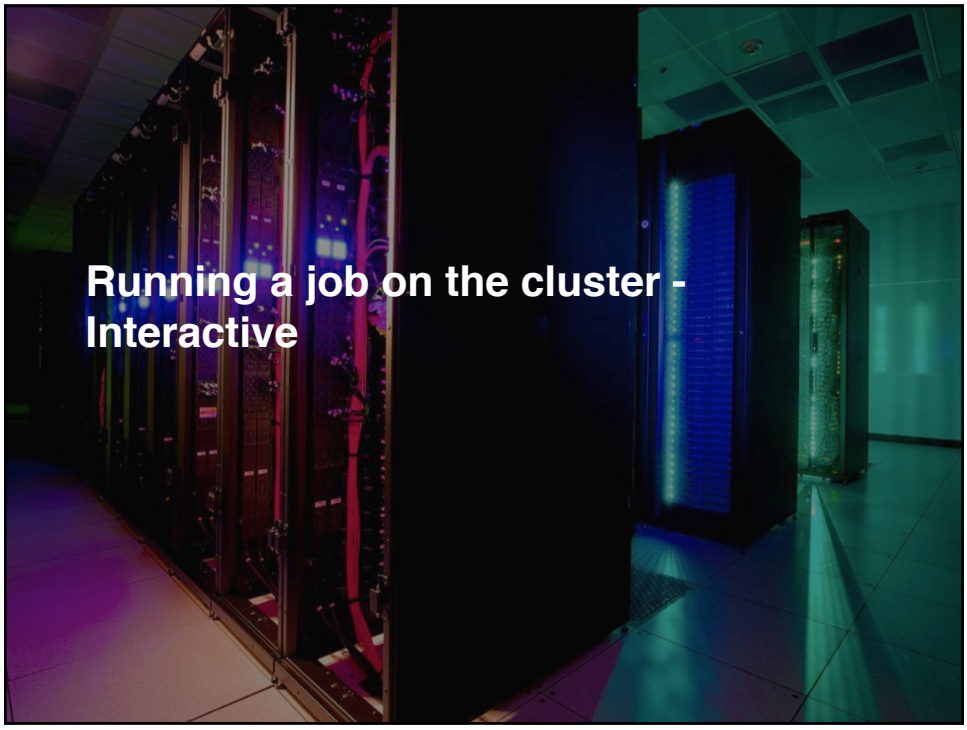
```
$ which python
/usr/bin/python

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

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

Software: system vs. /usr/usc programs

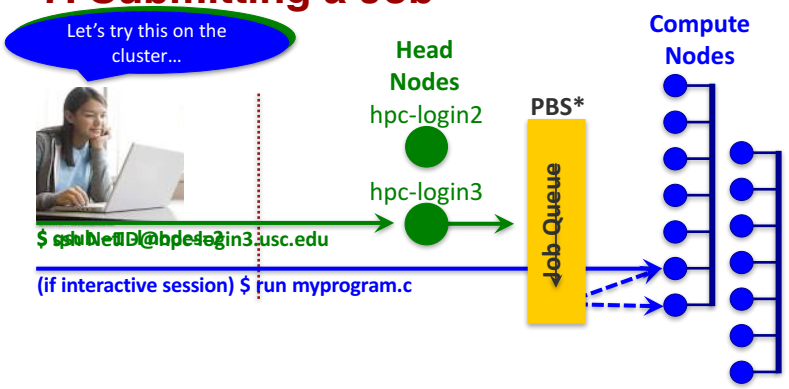
- Some software & libraries come pre-installed with OS
  - E.g. python, gcc, fftw
- The command name is the same, but OS software and repository software are usually different
  - E.g. different versions, libraries and/or developers
- Make sure that you use what you want to use!

USC University of Southern California

USC ITS Information Technology Services



7. Submitting a Job



*HPC uses the TORQUE resource manager (PBS) and the Moab Cluster Scheduler

USC University of Southern California

USC ITS Information Technology Services

Submitting a Job – Portable Batch System (PBS)

- Scheduling and resource allocation
  - Moab and Torque(PBS) manage the scheduling and distribution of batch jobs and interactive sessions across available nodes in the cluster
  - Jobs are scheduled based on
    - *Order submitted, number & types of nodes requested and time required*
- To submit a job to the HPC cluster
  - Add computing resource requests & program commands to a PBS script
  - Use PBS command qsub *(queue submit)* to submit your job
    - *$ qsub myjob.pbs*

myjob.pbs

```
#!/bin/bash
#PBS -l nodes=2:ppn=16
#PBS -l walltime=02:00:00
cd $PBS_O_WORKDIR
source
/usr/usc/sas/default/setup.sh
sas my.sas
```

USC University of Southern California

USC ITS Information Technology Services

PBS qsub options

qsub –l nodes=2:ppn=16:IB -l pmem=10g –l walltime=01:00:00

-**A** <your project account, if you have more than one>
-**q** <your queue, for condo owners>
-**d** <directory to start in>
-**l** start list of resources
nodes= number of nodes
:ppn= processor per core (nodes attribute)
:gpus= GPU node request (nodes attribute)
:nodeattr= machine architecture (nodes attribute)
mem= amount of total memory
pmem= amount of memory per processor
walltime= wallclock time

Running a job interactively

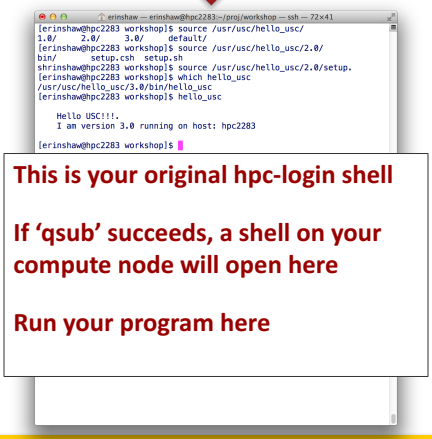
- Let's request a node!

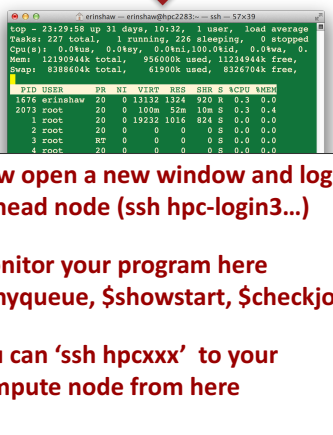
```
$ qsub -I –l nodes=2:ppn=8 –l walltime=01:00:00
qsub: waiting for job 14167009.hpc-pbs.hpcc.usc.edu to start
```

- While we're waiting for our node…
  - open a second terminal window and log in to a head node

USC University of Southern California
USC ITS Information Technology Services

Running a job interactively





This is your original hpc-login shell

If 'qsub' succeeds, a shell on your compute node will open here

Run your program here

Now open a new window and log in to head node (ssh hpc-login3…)

Monitor your program here ($myqueue, $showstart, $checkjob)

You can 'ssh hpcxxx' to your compute node from here

USC University of Southern California

USC ITS Information Technology Services

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'
```

USC University of Southern California

USC ITS Information Technology Services

Job monitoring: myqueue

myqueue Display jobs status and allocated node list for your running jobs (from login nodes). Same as qstat –u {me}

[col] S Displays the job status

Q - indicates the job is queued and waiting to be executed

R - indicates that the job is currently running

C - means that the job has completed

```
hpc-login3: myqueue
hpc-pbs.hpcc.usc.edu:
                                                                  Req'd      Req'd      Elap
Job ID              Username   Queue   Jobname      SessID NDS  TSK Memory  Time   S  Time
--------------------- ---------- -------- ---------------- ------ ----- ------ ------ --------- - ---------
11788258.hpc-pbs.hpcc. avalonjo  main    STDIN         32291   2    8   --   02:00:00 R 00:30:44
hpc2062/0+hpc2062/1+hpc2062/2+hpc2062/3+hpc2081/0+hpc2081/1+hpc2081/2+hpc2081/3
```


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 erinshaw
hpc-pbs.hpcc.usc.edu:
                                                                  Req'd   Req'd   Elap
Job ID              Username  Queue   Jobname    SessID  NDS  TSK  Memory  Time   S  Time
---------------------- ---------- ---------- ---------- ------ ------ ------ ------------ --------- -- ---------
9667416.hpc-pbs. erinshaw   main    testjob      3827   10    2     0     --    6:00:00  R 3:21:28

# Display first 10 jobs in main queue
$ qstat main | head –n 10
```

USC University of Southern California
USC ITS Information Technology Services

Got compute nodes?

```
[erinshaw@hpc-login3 ~]$ qsub -I -l nodes=2:ppn=4 -l walltime=01:00:00
qsub: waiting for job 22452388.hpc-pbs1.hpcc.usc.edu to start
qsub: job 22452388.hpc-pbs1.hpcc.usc.edu ready
----------------------------------------
Begin PBS Prologue Fri Jan 20 17:40:32 PST 2017
Job ID:          22452388.hpc-pbs1.hpcc.usc.edu
Username:        erinshaw
Group:           ucsadmin
Project:         default
Name:            STDIN
Queue:           quick
Shared Access:   no
All Cores:       no
Has MIC:         no
Nodes:           hpc1736 hpc1745
Scratch is:      /scratch
TMPDIR:          /tmp/22452388.hpc-pbs1.hpcc.usc
End PBS Prologue Fri Jan 20 17:40:57 PST 2017
----------------------------------------
[erinshaw@hpc1736 erinshaw]$ pwd
/auto/rcf-40/erinshaw
[erinshaw@hpc1736 erinshaw]$ cd /home/rcf-proj/ess/erinshaw/workshop
```

- Things to notice
  - Job ID:
    - *An useful number (give to us when ticketing)*
  - Name: STDIN
    - *Indicates an interactive job (typically PBS job name)*
  - Nodes: hpc1736 hpc1745
    - *Shell opens on first node, run your commands here*
  - Scratch:
    - *Refers to location of all shared disk space*
  - TMPDIR:
    - *This is an environment variable you can refer to*
  - /tmp/22452388.jpc-pbs1.hpcc.usc.edu
    - *This is a directory!*
  - $ pwd – starts in home directory
    - *Unless otherwise specified, you have to cd to project*

Job monitoring

- Login to head nodes and ssh to compute node
  - Type $ top to see all processes
- Other interesting information

```
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
```

Got compute nodes!

- We can now test our program on the cluster

```
hpc1736 $ source /usr/usc/hello_usc/3.0/setup.sh

hpc1736 $ which hello_usc
/usr/usc/hello_usc/3.0/bin/hello_usc

hpc1736 $ hello_usc
    Hello USC!!!.
    I am version 3.0 running on host: hpc1736
```


PBSDSH – PBS' distributed shell

- Some tasks can easily be performed in parallel
  - **pbsdsh** will distribute a task to multiple nodes and cores
  - Typical usage: pbsdsh executable [args]
- Let's try it:

```
hpc1736 $ pbsdsh -u /usr/usc/hello_usc/3.0/bin/hello_usc
    Hello USC!!!.
    I am version 3.0 running on host: hpc17336
    Hello USC!!!.
    I am version 3.0 running on host: hpc1745
```

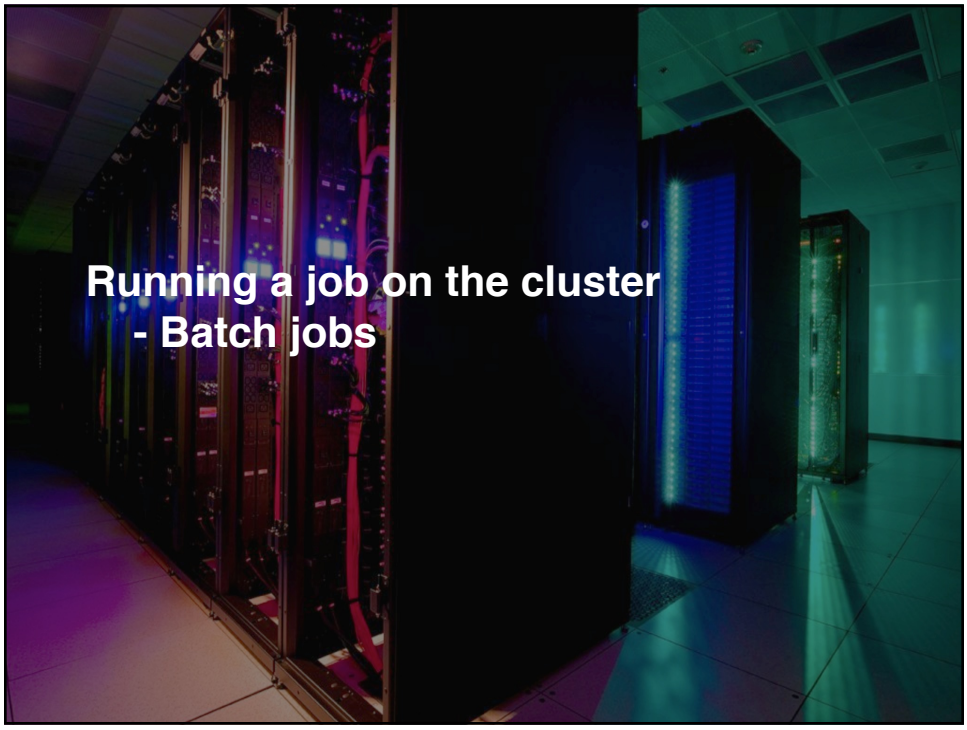
- Try without the –u, what happens?

PBS environment variables

- PBS sets a number of environment variables
  - PBS_O_HOST : The name of the host, where qsub was started
  - PBS_O_WORKDIR : The working directory, where qsub was started
  - PBS_JOBID, PBS_JOBNAME: The jobid and jobname
  - PBS_NODEFILE: Name of file that contains a list of all nodes the job has allocated, with an entry for every CPU.
- Within each process environment
  - PBS_NODENUM: the number of the node (counted from 0)
  - PBS_VNODENUM: the number of the process (counted from 0)

USC University of Southern California

USC ITS Information Technology Services



PBS batch jobs

- Use a PBS script to submit a "batch job" to cluster
  - Batch jobs are not interactive
  - They are submitted to cluster nodes and run automatically
  - Your nodes are released when your job finishes
- Copy hello3.pbs into your workshop directory
  - cp /home/rcf-proj/workshop/introHPC/hello3.pbs .
  - Edit and change the personal information
  - Submit using qsub hello.pbs
    - *You can submit a batch job from an interactive node*

PBS script example (Change slide!)

USC University of Southern California
USC ITS Information Technology Services

PBS batch jobs

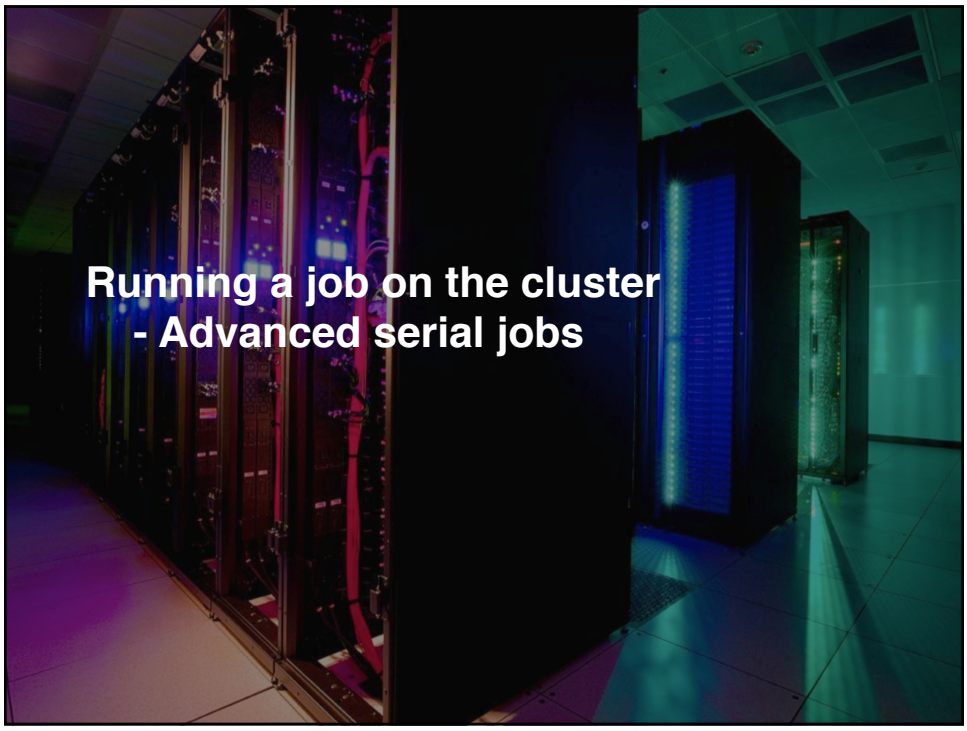
- When job completes
  - Go to the directory where you submitted the job
  - Look at the files – what was the job ID?

PBS script example

erinshaw — erinshaw@hpc1724:/staging/workshop/hpc — ssh — 69×34

GNU nano 2.0.9 File: /staging/workshop/hpc/hello3.pbs

```
#!/bin/bash

# Change lines with *

# Request nodes, ppn, walltime
#PBS -l nodes=1:ppn=8
#PBS -l walltime=00:10:00

# Set job, output, error filenames
#PBS -N hello3
#PBS -e hello3.$PBS_JOBID.err
#PBS -o hello3.$PBS_JOBID.out

# Email abort/begin/end messages*
#PBS -M erinshaw@usc.edu
#PBS -m abe

# Next two lines for HPC workshop only
#PBS -A workshop
#PBS -l advres=Workshop.966

# Change to your project directory*
cd /home/rcf-proj/ess/erinshaw/workshop

# Run setup file (use setup.csh for tcsh)
source /usr/usc/hello_usc/default/setup.sh

# Run program
pbsdsh /usr/usc/hello_usc/default/bin/hello_usc
```



PBSDSH example

From http://www.uni-tuebingen.de/fileadmin/Uni_Tuebingen/Einrichtungen/ZDV/Bilder/Computing/batch_doc-1.pdf

```
$ cat adv_serial_job.sh
#!/bin/bash
echo PROCESSES=$(cat $PBS_NODEFILE | wc -l)
/usr/local/bin/pbsdsh $PBS_O_WORKDIR/serial_job.sh

$ cat serial_job.sh
#!/bin/bash
echo HOSTNAME=$(hostname) NODENUM=$PBS_NODENUM VNODENUM=$PBS_VNODENU
```

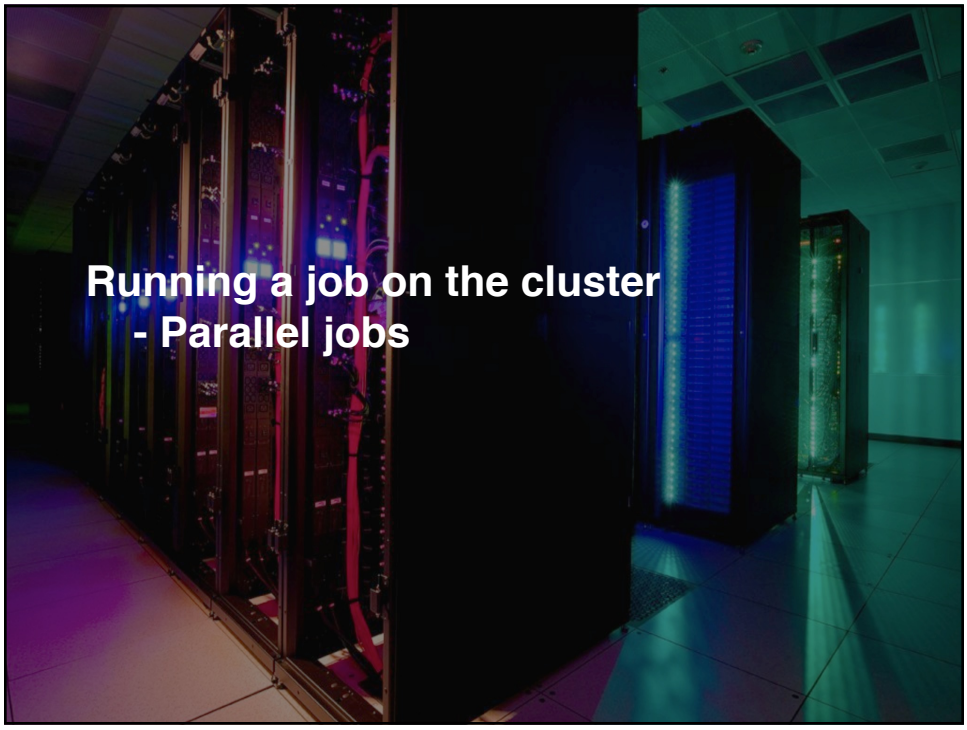
PBSDSH example

$ qsub -l nodes=2:ppn=3 adv_serial_job.sh
waiting for …

$ qstat
Job id Name User Time Use S Queue

19342 ...serial_job.sh myself 00:00:00 C quick

$ cat adv_serial_job.sh.o19342
PROCESSES=6
HOSTNAME=n030304 NODENUM=0 VNODENUM=0
HOSTNAME=n030108 NODENUM=1 VNODENUM=3
HOSTNAME=n030108 NODENUM=1 VNODENUM=4
HOSTNAME=n030108 NODENUM=1 VNODENUM=5
HOSTNAME=n030304 NODENUM=0 VNODENUM=1
HOSTNAME=n030304 NODENUM=0 VNODENUM=2



What is MPI*?

- MPI = Message Passing Interface
  - MPI is a ***specification*** for the developers and users of message passing libraries (for parallel programming).
  - MPI library ***implementations*** differ in which version and features of the MPI standard they support.
- USC HPC supports OpenMPI
  - And mpitch and mvapitch for compatibility
  - MPI will be covered in the next level HPC course

*https://computing.llnl.gov

Using MPI interactively

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: ls
compile.sh*  helloWorldMPI*

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
```

Using MPI interactively

Results of OpenMPI program

```
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!
```


Using MPI in batch mode

Create hellompi.pbs in your workshop directory and submit

```
#!/bin/bash
#PBS -l nodes=1:ppn=8
#PBS -l walltime=00:10:00

# change to your project directory
cd /home/rcf-proj/{myproj}/{mydir}

# source setup file (setup.csh for tcsh)
source /usr/usc/openmpi/1.8.4/setup.sh

# run command
mpiexec helloWorldMPI
```

Using MPI in batch mode

Create hellompi2.pbs and submit

```
#!/bin/bash
#PBS -l nodes=2:ppn=8:gpus=2,pvmem=2GB
#PBS -l walltime=00:30:00
#PBS –m abe                    #email sent on abort/begin/end
#PBS –M {email@usc.edu}        #my email address
#PBS –N helloMPI               #name of my job
#PBS –j oe                     #join error and output files
#PBS –d /home/rcf-proj/{myproj}/{mydir}    #start in this dir

# source necessary setup files for my simulation
source /usr/usc/intel/default/setup.sh
source /usr/usc/openmpi/default/setup.sh.intel
source /usr/usc/cuda/default/setup.sh

# run
mpirun –np 32 helloWorldMPI > log
```


Using MPI in batch mode

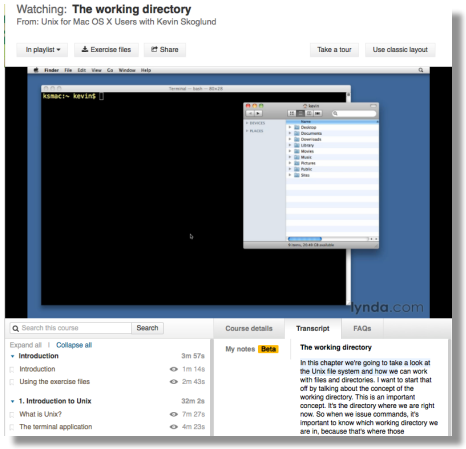
- Look at output

Note about node attributes myri and IB

- HPC has two clusters (interconnects)
  - Myrinet (myri) and Infiniband (IB)
  - The networks are not connected to each other
- If the interconnect attribute is not specified, HPC will use any set of nodes that allow your job to start
  - *qsub -I nodes=10:ppn=8:myri,walltime=4:00:00*
  - *qsub -I nodes=4:ppn=16:IB,walltime=8:00:00*
- Warning!
  - Codes compiled to use MPICH will only run on the Myrinet nodes
  - OpenMPI codes will run on either

Want to learn more?

lynda.com



- 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!

Want to learn more?

http://software-carpentry.org