Linux Basics and HPCC Cluster Intro

Charles Forsyth
System Administrator
High Performance Computing Center
University of California, Riverside
forsythc@ucr.edu

Agenda

- Cluster Introduction
- Accessing the Cluster
- Storage
- Linux commands
- Module System
- Environment Variables
- Submitting Cluster Jobs
- Scripting
- Monitoring the Cluster Jobs
- Helpful Resources

High Performance Computing Center

- Purpose
  - Provide High Performance Computing and Bigdata Resources to support and enable research at UCR.
- Access
  - Subscription based membership.
  - Accessible both on and off campus.
  - Organized by lab and department affiliation.

HPCC Website

Website Address: hpcc.ucr.edu

High-Performance Computing Center @ UCR | Nav | Related Websites | Feedback | Google Custom Search

HPCC

- Home
  - Introduction
  - Activity report
  - Slides
- News
- Access & rates
- Documents
- Contacts
- Events
- Hardware
- Software
- Manuals

High-Performance Computing Center (HPCC)

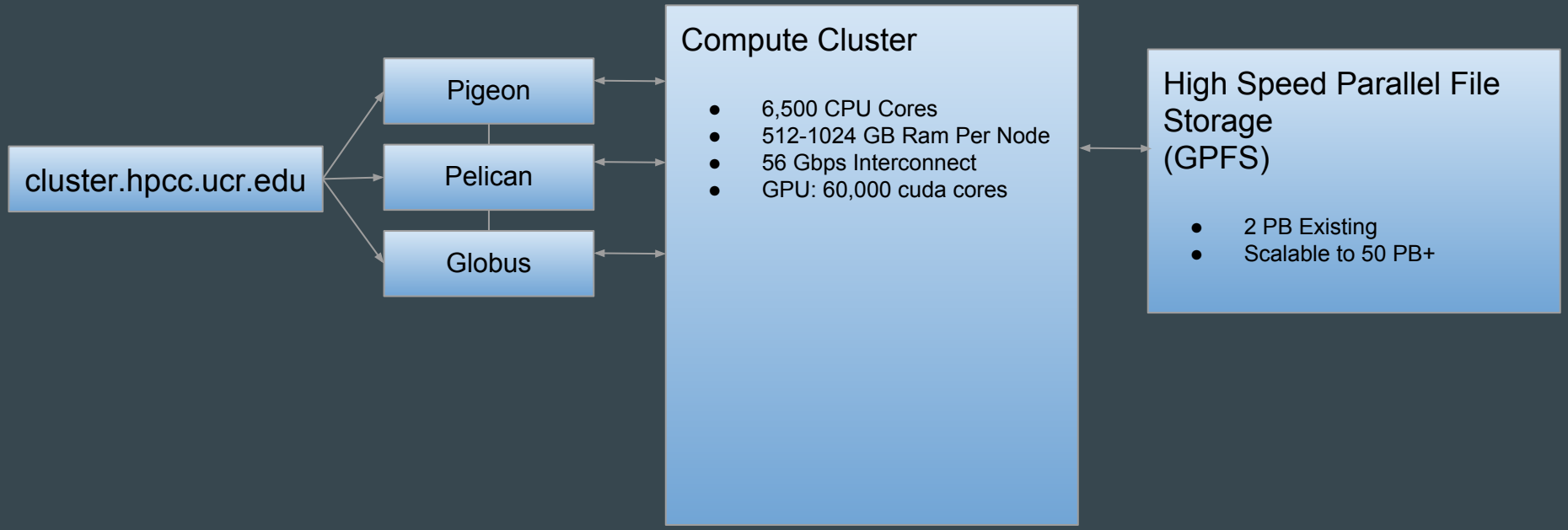
Summary: welcome to the website of the High-Performance Computing Center (HPCC) at UC Riverside. This site gives an overview of the HPC resources and services provided by our center.

Table of Contents

Mission and Services

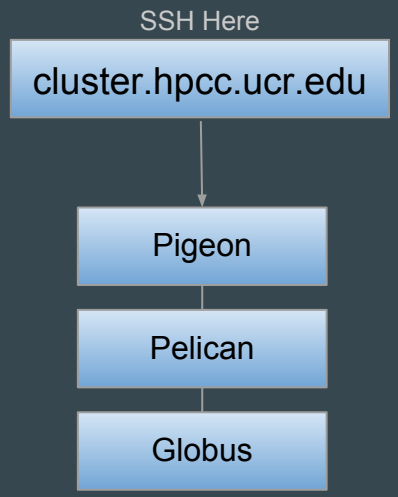
The High-Performance Computing Center (HPCC) provides state-of-the-art research computing infrastructure and training accessible to all UCR researchers and affiliates at low cost. This includes access to the shared HPC resources and services summarized below. The main advantage of a shared research computing environment is access to a much larger HPC infrastructure (with thousands of CPUs/GPUs and many PBs of directly attached storage) than what smaller clusters of individual research groups could afford, while also providing a long-term sustainability plan and professional systems administrative support.

HPCC Cluster Overview

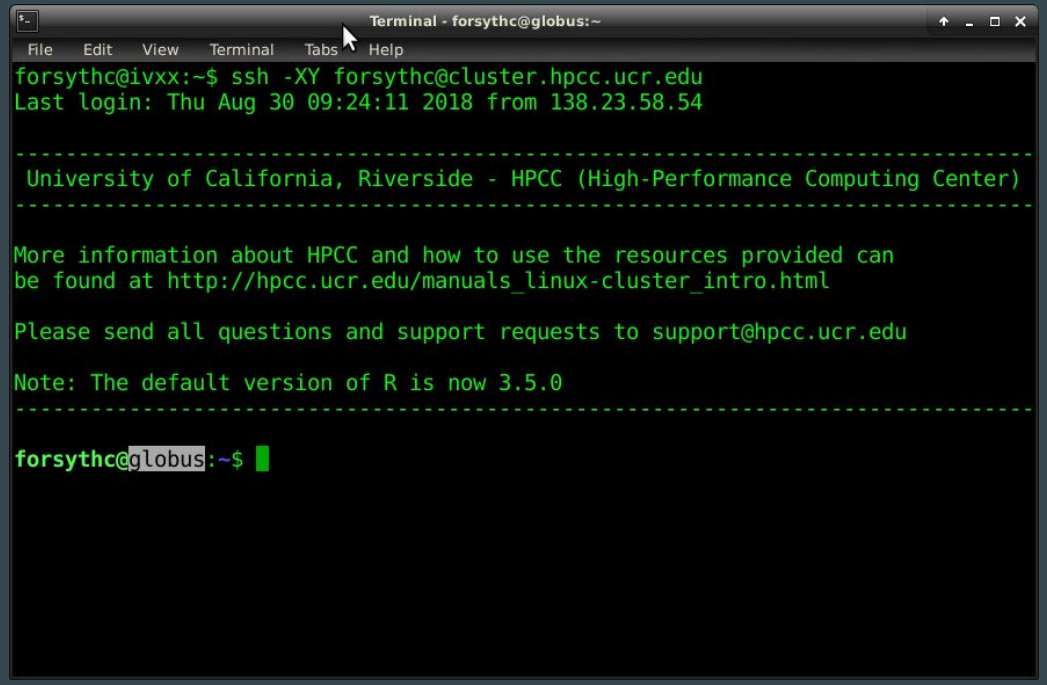


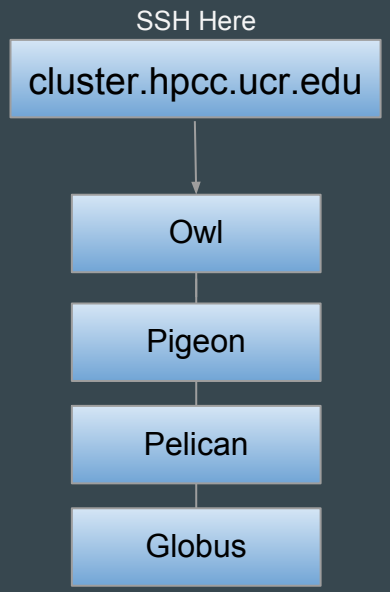
Accessing the Cluster - SSH

- SSH is the main protocol used to interface with the HPCC Cluster.
- Start by ssh'ing from your system to cluster.hpcc.ucr.edu
  - Windows
    - Putty
      - cluster.hpcc.ucr.edu
    - Mobaxterm
      - ssh -XY username@cluster.hpcc.ucr.edu
  - Mac
    - Terminal
      - ssh username@cluster.hpcc.ucr.edu
  - Linux
    - Terminal
      - ssh -XY username@cluster.hpcc.ucr.edu
  - The "-XY" allows graphical output to be passed back to your system.
- cluster.hpcc.ucr.edu is a special DNS name that load balances incoming connections to one of the available login nodes.

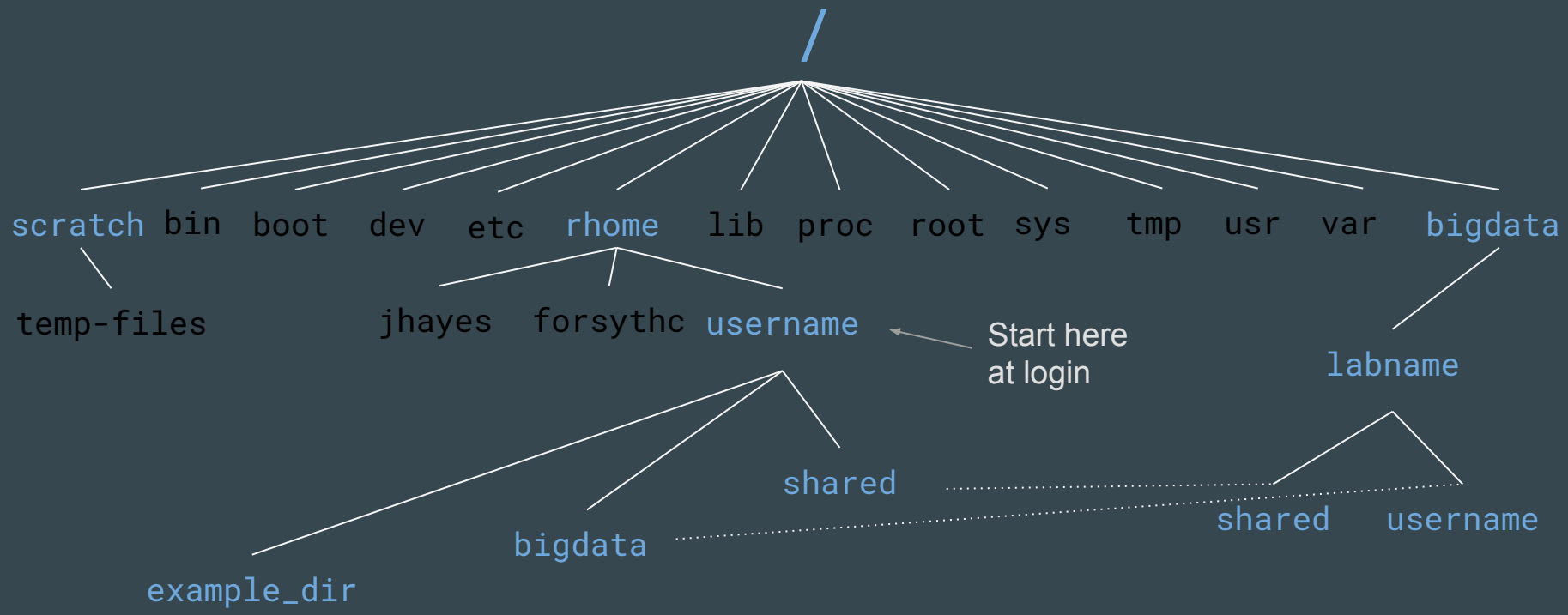


Accessing the Cluster - SSH





Storage - Directory Structure



Storage - Directory Structure

Home is used for scripting, debugging and small files. (20 GB Quota)

```
/rhome/username
```

Bigdata is used for parallel jobs, high read/write operations and big files. (100 GB for 100/yr, 10 TB for 1000/yr)

```
/bigdata/labname/username
/bigdata/labname/shared
```

Note: All lab members share the same bigdata pool.

Scratch is local to the compute node and temporary (30 days max).

```
/scratch
```

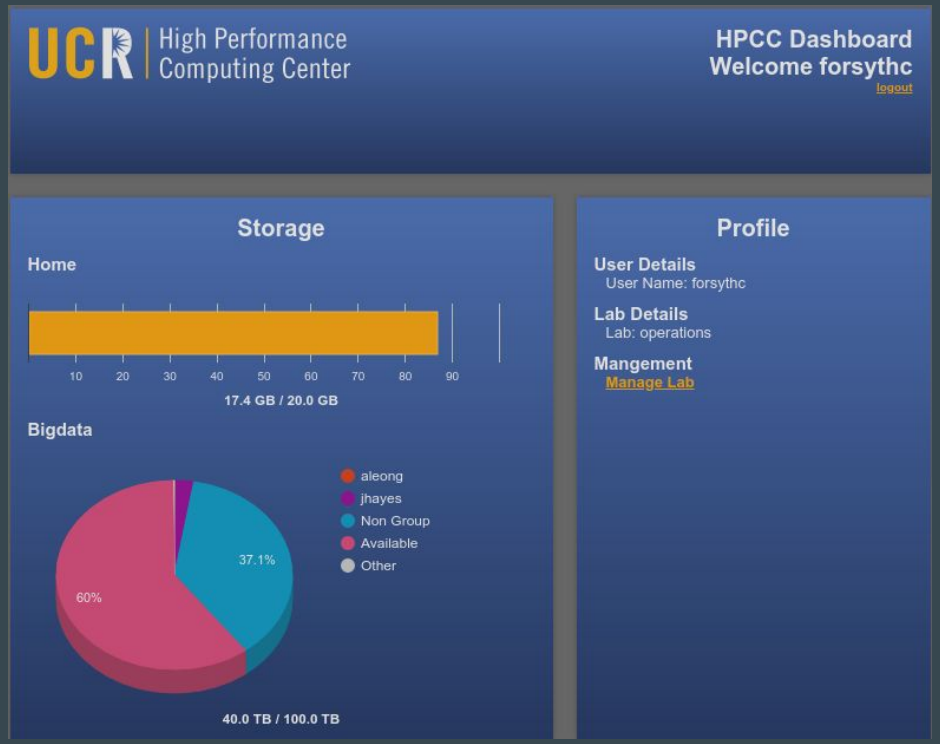
Storage - Directory Structure

```
Terminal - tmux.2.ssh
File  Edit  View  Terminal  Tabs  Help
repo@penguin:~$ ls
bigdata  shared
repo@penguin:~$ ll
total 0
lrwxrwxrwx 1 repo operations 24 Nov 20  2017 bigdata -> /bigdata/operations/repo
lrwxrwxrwx 1 repo operations 26 Nov 20  2017 shared -> /bigdata/operations/shared
repo@penguin:~$ ls -latr
total 768
-rw-r--r--    1 repo operations    677 Apr  8  2013 .profile
-rw-r--r--    1 repo operations     92 Jul 11  2015 .bashrc
-rw-r--r--    1 repo operations    269 Oct 31  2016 .Rprofile
-rw-r--r--    1 repo operations  12104 Oct 31  2016 .vimrc
-rw-r--r--    1 repo operations   4882 Oct 31  2016 .tmux.conf
drwxr-xr-x   18 repo operations   4096 Nov  1  2016 .vim
lrwxrwxrwx    1 repo operations     24 Nov 20  2017 bigdata -> /bigdata/operations/repo
lrwxrwxrwx    1 repo operations     26 Nov 20  2017 shared -> /bigdata/operations/shared
-rw-------    1 repo operations   7785 Nov 30  2017 .viminfo
drwx------    2 repo operations   4096 Jan 11  2018 .ssh
drwx------    5 repo operations   4096 May  4 16:52 .
drwxr-----    3 repo operations   4096 May  4 16:52 .pki
-rw-------    1 repo operations   9149 Jul 18 14:21 .bash_history
drwxr-xr-x 1072 root root       262144 Sep  5 19:18 ..
repo@penguin:~$
[ aegis ]          1:ssh-  2:ssh*          [ 0.00 0.01 0.05 ][ Thu 2018-09-06 09:22 ]
```

Storage - Directory Structure

How much data am I using currently:

- https://dashboard.hpcc.ucr.edu
- `check_quota home`
- `check_quota bigdata`



Storage - Uploading/Downloading

- Moving files to or from the Cluster is allowed via SCP or SFTP.
- SCP `scp <from> <to>`
  - `scp user@remote_host:filename .`
    - Copies file from server to local machine (typed from local machine prompt). The “ . “ copies to pwd, you can specify here any directory, use wildcards to copy many files.
  - `scp filename user@remote_host:~/dir/.`
    - Copies file from local machine to server.
  - `scp -r user@remote_host:directory/ ~/dir`
    - Copies entire directory from server to local machine.

- SFTP
  - FileZilla is a graphical SFTP client available free for Linux, Mac and Windows platforms.
    - https://filezilla-project.org/
  - The SFTP protocol is also available from the command line terminal in Linux and Mac.

Storage - Filesystem

- Case sensitive
  - All paths and commands are case sensitive, an uppercase letter is not the same as a lowercase letter.

- Path Types
  - Absolute path
    - Full path from top to bottom
      - `/rhome/yourname/example_dir/example_file`
  - Relative path
    - Current working directory is implied
      - `./example_dir/example_file`
      - `../another_dir/example_dir/example_file`

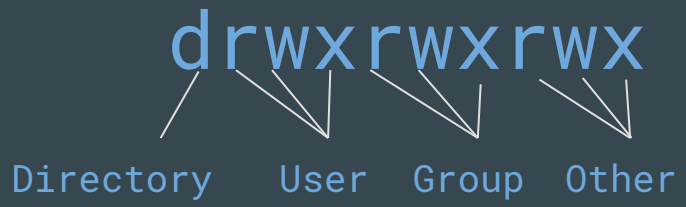
Linux - File Permissions

To view the permissions on the files in your current directory:

- List files
  - `ls`
- List files in long format
  - `ls -l`
  - `ll`
- List current directory in long format
  - `ls -ld`
- List all files in long format
  - `ls -la`

```
Terminal - tmux.2.ssh
File  Edit  View  Terminal  Tabs  Help
repo@penguin:~$ ls
bigdata  shared
repo@penguin:~$ ll
total 0
lrwxrwxrwx 1 repo operations 24 Nov 20  2017 bigdata -> /bigdata/operations/repo
lrwxrwxrwx 1 repo operations 26 Nov 20  2017 shared -> /bigdata/operations/shared
repo@penguin:~$ ls -latr
total 768
-rw-r--r--    1 repo operations    677 Apr  8  2013 .profile
-rw-r--r--    1 repo operations     92 Jul 11  2015 .bashrc
-rw-r--r--    1 repo operations    269 Oct 31  2016 .Rprofile
-rw-r--r--    1 repo operations  12104 Oct 31  2016 .vimrc
-rw-r--r--    1 repo operations   4882 Oct 31  2016 .tmux.conf
drwxr-xr-x   18 repo operations   4096 Nov  1  2016 .vim
lrwxrwxrwx    1 repo operations     24 Nov 20  2017 bigdata -> /bigdata/operations/repo
lrwxrwxrwx    1 repo operations     26 Nov 20  2017 shared -> /bigdata/operations/shared
-rw-------    1 repo operations   7785 Nov 30  2017 .viminfo
drwx------    2 repo operations   4096 Jan 11  2018 .ssh
drwx------    5 repo operations   4096 May  4 16:52 .
drwxr-----    3 repo operations   4096 May  4 16:52 .pki
-rw-------    1 repo operations   9149 Jul 18 14:21 .bash_history
drwxr-xr-x 1072 root root       262144 Sep  5 19:18 ..
repo@penguin:~$
[ aegis ]          1:ssh-  2:ssh*          [ 0.00 0.01 0.05 ][ Thu 2018-09-06 09:22 ]
```

Linux - File Permissions

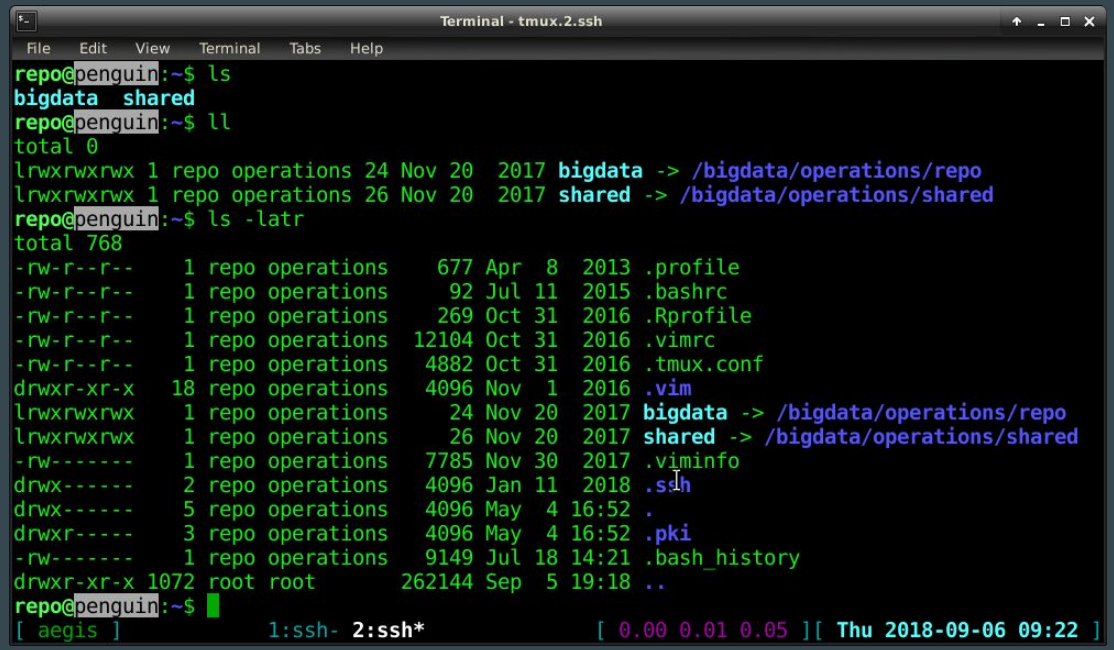


Every file and directory has permissions:

- User read/write/execute
- Group read/write/execute
- Other read/write/execute

Change the permissions of a file:

- `chmod g+w filename`
- `chmod o-r filename`
- `chmod u+x filename`



Linux - Basic Commands

Basic Commands

- `pwd` - Print working directory
- `ls` - List files in directory
- `touch` - Make an empty file
- `mkdir` - Make a directory
- `cd` - Change to directory
- `cp` - Copy file[s] from a directory to a directory
- `mv` - Move file[s] from a directory to a directory
- `rm` - Remove a file
- `rmdir` - Remove an empty directory

Note: ctrl+c will cancel a running command

Screen / Tmux (don't forget to 'exit' or 'kill' your sessions)

Screen - quick reference

- Open a new session
  `screen`
- Detach from a session
  `C-a d`
- Resume a previous session
  `screen -rd`
- Open new window
  `C-a c`
- Rename window
  `C-a A`
- Navigate to previous or next window
  `C-a p` *OR* `C-a n`
- Window list
  `C-a "`
- Split window horizontally or vertically
  `C-a S` *OR* `C-a |`
- Navigate window panes
  `C-a tab`
- Close window pane
  `C-a X`

Tmux - quick reference (screen/vi mode)

- Open a new session
  `tmux`
- Detach from a session
  `C-a d`
- Resume a previous session
  `tmux attach -d`
- Open new window
  `C-a c`
- Rename window
  `C-a ,`
- Navigate to previous or next window
  `C-a p` *OR* `C-a n`
- Window list
  `C-a w`
- Split window horizontally or vertically
  `C-a "` *OR* `C-a %`
- Navigate window panes
  `C-a arrow-key`
- Close window pane
  `C-a s`

Linux - Text editors and viewers

Viewers

- `less`
- `more`

Editors

- `nano`
- `vim`
- `emacs`
- `pico`

Linux - File Streams

- STDOUT
  - Save STDOUT to file
    - `ls > list_of_files.txt`
  - Append STDOUT to file
    - `ls >> list_of_files.txt`

- STDIN
  - Count lines in STDIN
    - `wc -l < list_of_files.txt`

- STDERR
  - Save error messages to file
    - `ls -e 2> errors.txt`

Linux - Piping

Passes output from one command to the input of the next.

command1 | command2

- `grep 'pattern' filename | wc -l`

command1 | command2 | command3 (...etc)

- `cat filename.csv | cut -f 1 | sort | uniq`

Software Module System

Print available modules

```
module avail
```

Print available modules starting with R

```
module avail R
```

Load default module R

```
module load R
```

Load specific module R version

```
module load R/3.2.0
```

Print list of loaded modules

```
module list
```

Unload module R

```
module unload R
```

Unload specific module R

```
module unload R/3.2.0
```

Environment Variables

- The HPCC cluster uses bash as the default shell environment.
- Within this environment, variables can be set and reused.

```
MYVAR='Something'
export MYVAR='Something'
echo $MYVAR
```

- Some software utilizes this feature and requires that specific environment variables be set.
  - `$HOME`
    - Contains your home path
  - `$USER`
    - Contains your username
  - `$PATH`
    - Contains paths of executables
  - `$LD_LIBRARY_PATH`
    - Contains paths of dependencies

Environment Variables

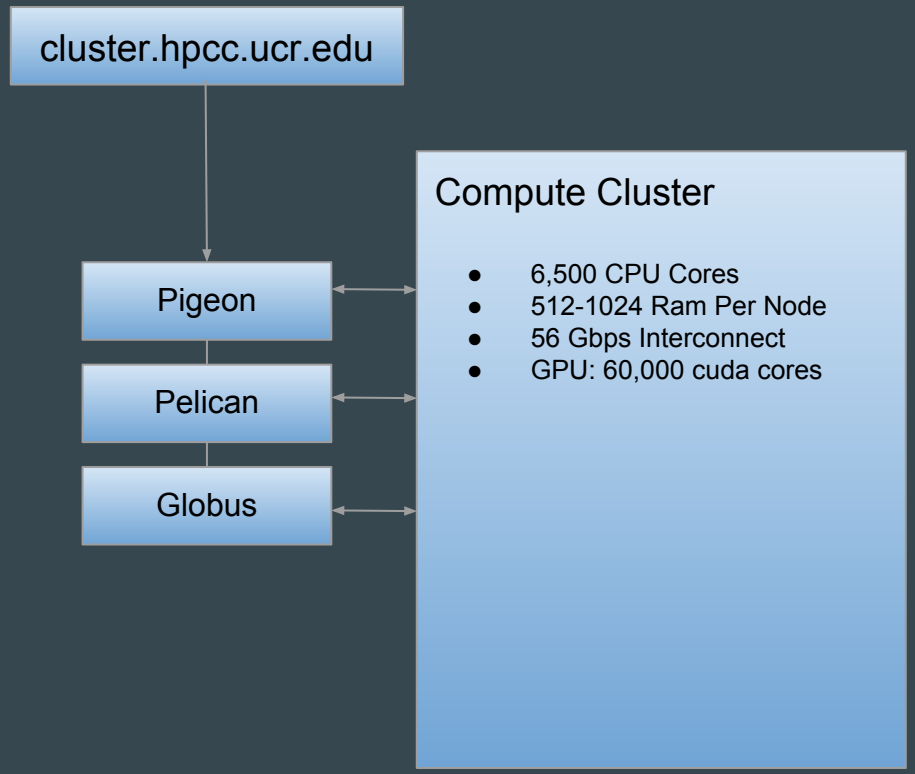
Slurm Job Environment variables (small sample)

- `$SLURM_SUBMIT_DIR`
  - directory where you ran the script
- `$SLURM_JOB_ID`
  - the SLURM Job ID of the current Job
- `$SLURM_NTASKS`
  - number of cpus the job requested
- `$SLURM_NODELIST`
  - list of nodes this job is running on

Queuing System

Slurm

- Resource Management
- Job Scheduler



Queuing System - Partitions

- Short
  - Nodes: i01-i48
  - Cores: Intel, 256 per user
  - Time (walltime): 2 hours default
  - Time (walltime): 2 hours max
- Intel
  - Default partition
  - Nodes: i01-02, i17-i48
  - Cores: Intel, 256 per user
  - Time (walltime): 168 hours (7 days) default
  - Time (walltime): 30 days max
- Batch
  - Nodes: c01-c48
  - Cores: AMD, 256 per user
  - Time (walltime): 168 hours (7 days) default
  - Time (walltime): 30 days max
- Highmem
  - Nodes: h01-h06
  - Cores: Intel, 32 per user
  - RAM: 100 GB min and 1024 GB max
  - Time (walltime): 48 hours default
  - Time (walltime): 30 days max
- Gpu
  - Nodes: gpu01-gpu05
  - Cores: Intel
  - RAM: 128 GB default
  - Time (walltime): 48 hours default
  - Time (walltime): 30 days max

Queuing System - Partitions

- statsdept
  - Default partition
  - Nodes: i45-i48
  - Cores: Intel, 8 per user
  - Time (walltime): 168 hours (7 days) default
  - Time (walltime): 30 days max

Queuing System - Job Limits

There are both group and user cpu limits for jobs running at the same time.

- Group
  - 512 Cores per group running at any one time
- User
  - 256 Cores per person running at any one time

See your limits:

```
slurm_limits.sh
```

Check total number of cpus currently used by your group in all partitions:

```
group_cpus
```

Queuing System - Status

Current cluster status can be viewed from: http://hpcc.ucr.edu/snapshot.html.

https://dashboard.hpcc.ucr.edu

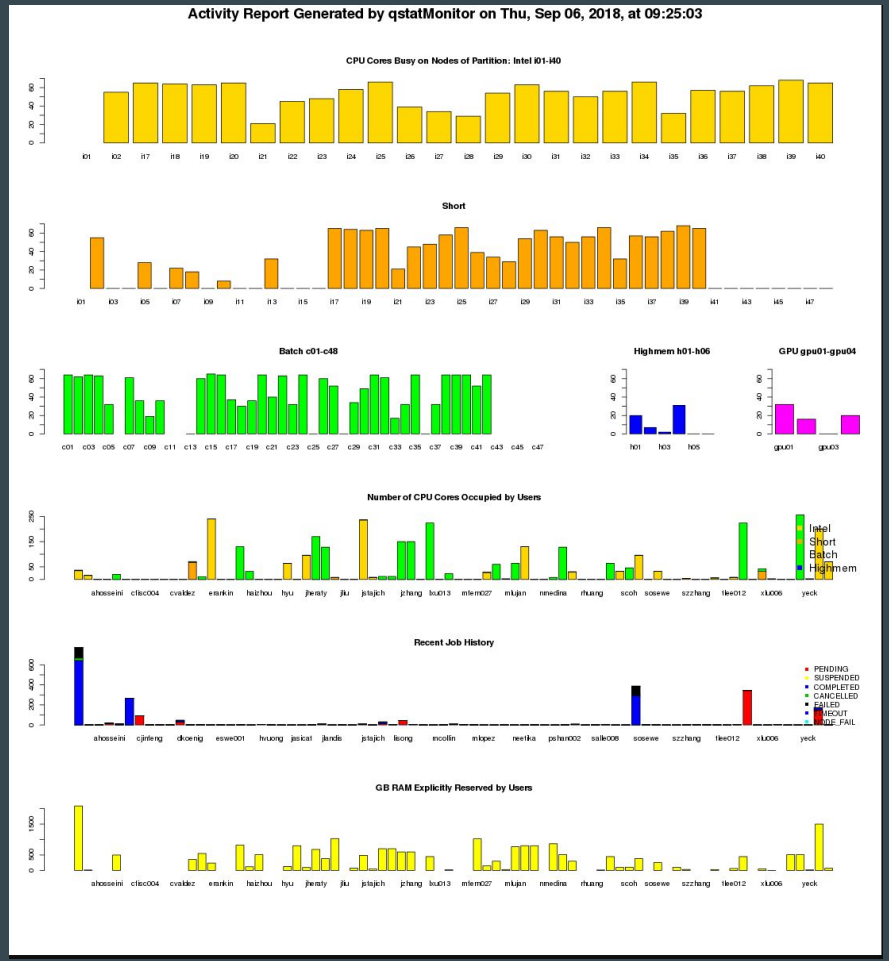
http://xdmod.hpcc.ucr.edu/

Command line tool:

```
jobMonitor
```

These commands will also give some interesting information about jobs on the cluster:

```
sinfo
squeue
sacct
```



Queuing System - Submitting a Job

Two major ways to submit jobs on the cluster

- `srun`
  - Used for interactive jobs
  - Examples:
    - `srun --pty bash -l`
    - `srun -p short --x11 --mem=1gb --ntasks 1 --time 02:00:00 --pty bash -l`
- `sbatch`
  - Used to submit batch(non-interactive) jobs
  - Examples:
    - `sbatch sbatch_script.sh`
    - `sbatch --ntasks=12 --mem=24gb sbatch_script.sh`

Queuing System - Submitting a Job

Interactive Job

```
srun -p short --x11 --mem=1gb --ntasks 1 --time 02:00:00 --pty bash -l
```

forsythc@chuck-ws: ~

```
forsythc@owl:~$ srun --pty bash -l
forsythc@i01:~$ pwd
/rhome/forsythc
forsythc@i01:~$ ls
IVXX-pub-gpg-key.asc  augustus-config  bin                  gcc-test  shared        tmp
R                     bigdata          compile-r-notes.txt  repos     slurm-scripts vpn-keys
forsythc@i01:~$ printenv | grep SLURM | tail -10
SLURM_JOB_CPUS_PER_NODE=2
SLURM_CLUSTER_NAME=biocluster
SLURM_GTIDS=0
SLURM_SUBMIT_HOST=owl
SLURM_JOB_PARTITION=intel
SLURM_STEP_NUM_TASKS=1
SLURM_JOB_ACCOUNT=bioinfo
SLURM_JOB_NUM_NODES=1
SLURM_STEP_TASKS_PER_NODE=1
SLURM_STEP_NODELIST=i01
forsythc@i01:~$
```

Scripting

Basicly a series of shell commands:

```
pwd
mkdir example_dir
cd example_dir
touch example_file
cd ..
mkdir example_dir2
cd example_dir2
touch example_file2
cd ..
ls -latr
```

Scripting

Put contents in a file and add the interpreter line to the top - Now it's a Script:

```
#!/bin/bash -l
pwd
mkdir example_dir
cd example_dir
touch example_file
cd ..
mkdir example_dir2
cd example_dir2
touch example_file2
cd ..
ls -latr
```

Scripting

- First line in file which defines the interpreter
  - `#!/bin/bash -l`
- Permissions - At least the owner of the file requires execute permissions
  - `chmod u+x myscript.sh`

Queuing System - Submitting a Job

`sbatch`

- Used to submit batch (non-interactive) jobs
- Examples:
  - `sbatch sbatch_script.sh`
  - `sbatch --ntasks=12 --mem=24gb sbatch_script.sh`

```
forsythc@chuck-ws: ~
forsythc@owl:~/slurm-scripts$ sbatch sleep-60.sh
Submitted batch job 1003897
forsythc@owl:~/slurm-scripts$ sbatch -p batch sleep-60.sh
Submitted batch job 1003899
forsythc@owl:~/slurm-scripts$ sbatch --mem=20gb sleep-60.sh
Submitted batch job 1003900
forsythc@owl:~/slurm-scripts$
```

Batch Submission Examples

Run the following command to download some example files:

```
git clone https://github.com/ucr-hpcc/hpcc_intro_files.git
```

This will create a folder called hpcc_intro_files.
This directory contains example files that you can use to follow along with and run.

Alternate download:

```
wget https://github.com/ucr-hpcc/hpcc_intro_files/archive/master.zip
```

Batch Submission Examples

slurm_script_simple.sh

```
#!/bin/bash -l

#SBATCH --ntasks=1
#SBATCH --time=00:00:60



# Print current date
date
# sleep for 45 seconds
/bin/sleep 45
# Print name of node
hostname
```

Batch Submission Examples

slurm_script_full_sbatch_options.sh

```
#!/bin/bash -l

#SBATCH --nodes=1
#SBATCH --ntasks=10 # asking for 10 cpus
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=1G
#SBATCH --time=0-00:15:00     # 0 day and 15 minutes
#SBATCH --output=my.stdout
#SBATCH --mail-user=forsythc@ucr.edu
#SBATCH --mail-type=ALL
#SBATCH --job-name="just_a_test"

date

hostname
```

Batch Submission Examples

slurm_script_highmem.sh

```
#!/bin/bash -l


#SBATCH --nodes=1
#SBATCH --ntasks=2
#SBATCH --mem=100gbG
#SBATCH --time=00-00:00:30
#SBATCH --job-name="highmem"
#SBATCH -p highmem

/bin/sleep 30
```

Batch Submission Examples

slurm_script_gpu.sh

```
#!/bin/bash -l

#SBATCH --nodes=1
#SBATCH --ntasks=8
#SBATCH -p gpu
#SBATCH --gres=gpu:1
#SBATCH --mem=100G
#SBATCH --time=00:00:10

echo "--------"
hostname
free -g
echo "--------"
```

Batch Submission Examples

slurm_script_multinode.sh

```
#!/bin/bash -l

#SBATCH --nodes=2
#SBATCH --ntasks=24
#SBATCH --ntasks-per-node=12
#SBATCH --output=multinode.out
#SBATCH --job-name="multinode test"
#SBATCH --time=00:00:10

echo "--------"
srun hostname
echo "--------"
```

Batch Submission Examples

slurm_script_mpi.sh

```
#!/bin/bash -l

#SBATCH --nodes=4
#SBATCH --ntasks=200
#SBATCH --output=mutinode.out
#SBATCH --job-name="mpi test"
#SBATCH -p batch,short,intel

#SBATCH --time=00:01:00

module load openmpi/2.0.1-slurm-16.05.4
echo "--------"
mpirun mpi_hello_world
echo "--------"
```

Batch Submission Examples

```
slurm_script_pass_argument.sh <input>
```

```
#!/bin/bash -l

#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --job-name="argument example"
#SBATCH --time=00:00:30
#SBATCH -p short,batch,intel

echo $1
```

Queuing System - job output

If no output file is specified in the slurm submission script then the default output file will be produced for each sbatch run.

Format is:

```
slurm-<job id>.out
```

Example:

```
slurm-2378722.out
```

This file contains the STDOUT (standard out) of your submission script.

Queuing System - job info

Check how busy the cluster is overall (shows all jobs in all states):

```
squeue -l
```

Check how many jobs overall are in the running state:

```
squeue -l -t R
```

Check how many jobs overall are in the pending state:

```
squeue -l -t PD
```

Check the state of your jobs:

```
squeue -l -u <user_username>
```

```
forsythc@chuck-ws: ~
forsythc@owl:~/slurm-scripts$ squeue -l -u forsythc
Fri Jul  7 14:12:34 2017
             JOBID PARTITION     NAME     USER    STATE       TIME TIME_LIMI  NODES NODELIST(REASON)
            960445     batch  sleep60 forsythc  RUNNING       0:39      1:00      1 c18
```

Queuing System - job info

View recent job info from the accounting database:

```
sacct
```

Lots more this can do:

```
man sacct
```

Queuing System - job control

Cancel your job:

```
scancel <JOBID>
```

Cancel multiple jobs:

```
scancel <JOBID1> <JOBID2> <JOBID3>
```

Cancel ALL your job (caution this will kill all running and queued jobs):

```
scancel -u $USER
```

More info on scancel:

```
man scancel
```

Sharing Files on the Web

Simply move the files into your html directory when you want to share them.
For example, log into the cluster and do the following:

1. Go to your web directory
   ```
   cd ~/.html/
   ```
2. Make sure permissions are set correctly
   ```
   chmod a+x ~/
   chmod a+rx ~/.html
   ```
3. Create a default test file
   ```
   echo '<h1>Hello!</h1>' > index.html
   ```

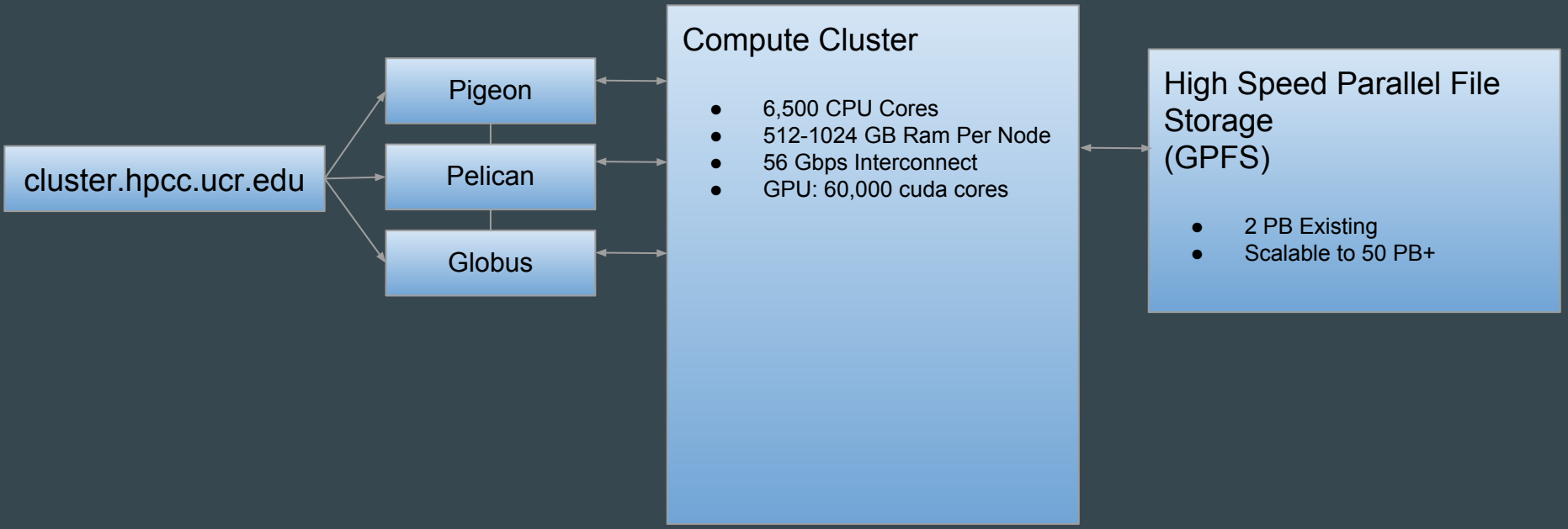
Now, test it out by pointing your web-browser to http://biocluster.ucr.edu/~username/
Be sure to replace 'username' with your actual user name, and also check permissions on shared directories and parent directories.

For password protecting html content follow these instructions:
http://manuals.bioinformatics.ucr.edu/home/hpc#TOC-Password-Protect-Web-Pages

Helpful Resources

- Command Line
  --help, -h, man
- Online Manual
  http://hpcc.ucr.edu
- Storage Usage
  https://dashboard.hpcc.ucr.edu
- Announcements
  http://hpcc.ucr.edu/news.html

Questions


cluster.hpcc.ucr.edu
Pigeon
Pelican
Globus
Compute Cluster
6,500 CPU Cores
512-1024 GB Ram Per Node
56 Gbps Interconnect
GPU: 60,000 cuda cores
High Speed Parallel File Storage (GPFS)
2 PB Existing
Scalable to 50 PB+

Thank You

Queuing System - Array Jobs

```
sbatch --array=1-10  script.slurm
       --array=1855-1900 script.slurm
```

```
#!/bin/bash -l

#SBATCH --nodes=1

#SBATCH --ntasks=1

#SBATCH --time=01:00:00



cd $SLURM_SUBMIT_DIR



echo "The Slurm Job ID is: "$SLURM_JOBID", The Slurm Array task ID is: "$SLURM_ARRAY_TASK_ID



inputfilename=inputfile_$SLURM_ARRAY_TASK_ID.inp
```

Queuing System - Submitting a Job

Dependency submission (man sbatch)

```
sbatch --dependency=after:JOBID myscript.sh

sbatch --dependency=afterany:JOBID myscript.sh

sbatch --dependency=afterok:JOBID myscript.sh
```

```
forsythc@chuck-ws: ~
forsythc@owl:~/slurm-scripts$ sbatch sleep-60.sh
Submitted batch job 1003907
forsythc@owl:~/slurm-scripts$ sbatch --dependency=afterok:1003907 sleep-60.sh
Submitted batch job 1003908
forsythc@owl:~/slurm-scripts$ squeue -l -u forsythc
Mon Jul 17 11:35:54 2017
             JOBID PARTITION     NAME     USER    STATE       TIME TIME_LIMI  NODES NODELIST(REASON)
           1003907     intel  sleep60 forsythc  PENDING       0:00      1:00      1 (Priority)
           1003908     intel  sleep60 forsythc  PENDING       0:00      1:00      1 (Dependency)
```

Queuing System - job info detail

Detailed info about a specific job:

scontrol show job <JOBID>

```
forsythc@chuck-ws: ~
Submitted batch job 969201
forsythc@pigeon:~/slurm-scripts$ scontrol show job 969201
JobId=969201 JobName=sleep60
   UserId=forsythc(3365) GroupId=bioinfo(1054) MCS_label=N/A
   Priority=4294207521 Nice=0 Account=bioinfo QOS=normal
   JobState=RUNNING Reason=None Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   RunTime=00:00:06 TimeLimit=00:01:00 TimeMin=N/A
   SubmitTime=2017-07-10T15:57:02 EligibleTime=2017-07-10T15:57:02
   StartTime=2017-07-10T15:57:14 EndTime=2017-07-10T15:58:14 Deadline=N/A
   PreemptTime=None SuspendTime=None SecsPreSuspend=0
   Partition=intel AllocNode:Sid=pigeon:19625
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=i02
   BatchHost=i02
   NumNodes=1 NumCPUs=2 NumTasks=1 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
   TRES=cpu=2,mem=2G,node=1
   Socks/Node=* NtasksPerN:B:S:C=0:0:*:* CoreSpec=*
   MinCPUsNode=1 MinMemoryCPU=1024M MinTmpDiskNode=0
   Features=(null) Gres=(null) Reservation=(null)
   OverSubscribe=OK Contiguous=0 Licenses=(null) Network=(null)
   Command=/rhome/forsythc/slurm-scripts/sleep-60.sh
   WorkDir=/rhome/forsythc/slurm-scripts
   StdErr=/rhome/forsythc/slurm-scripts/slurm-969201.out
   StdIn=/dev/null
   StdOut=/rhome/forsythc/slurm-scripts/slurm-969201.out
   Power=
```