Introduction to Cluster Computing @ UH
Using Opuntia and Maxwell

Jerry Ebalunode
jebalunode@uh.edu

Center for Advanced Computation and Data Systems (CACDS)
http://cacds.uh.edu

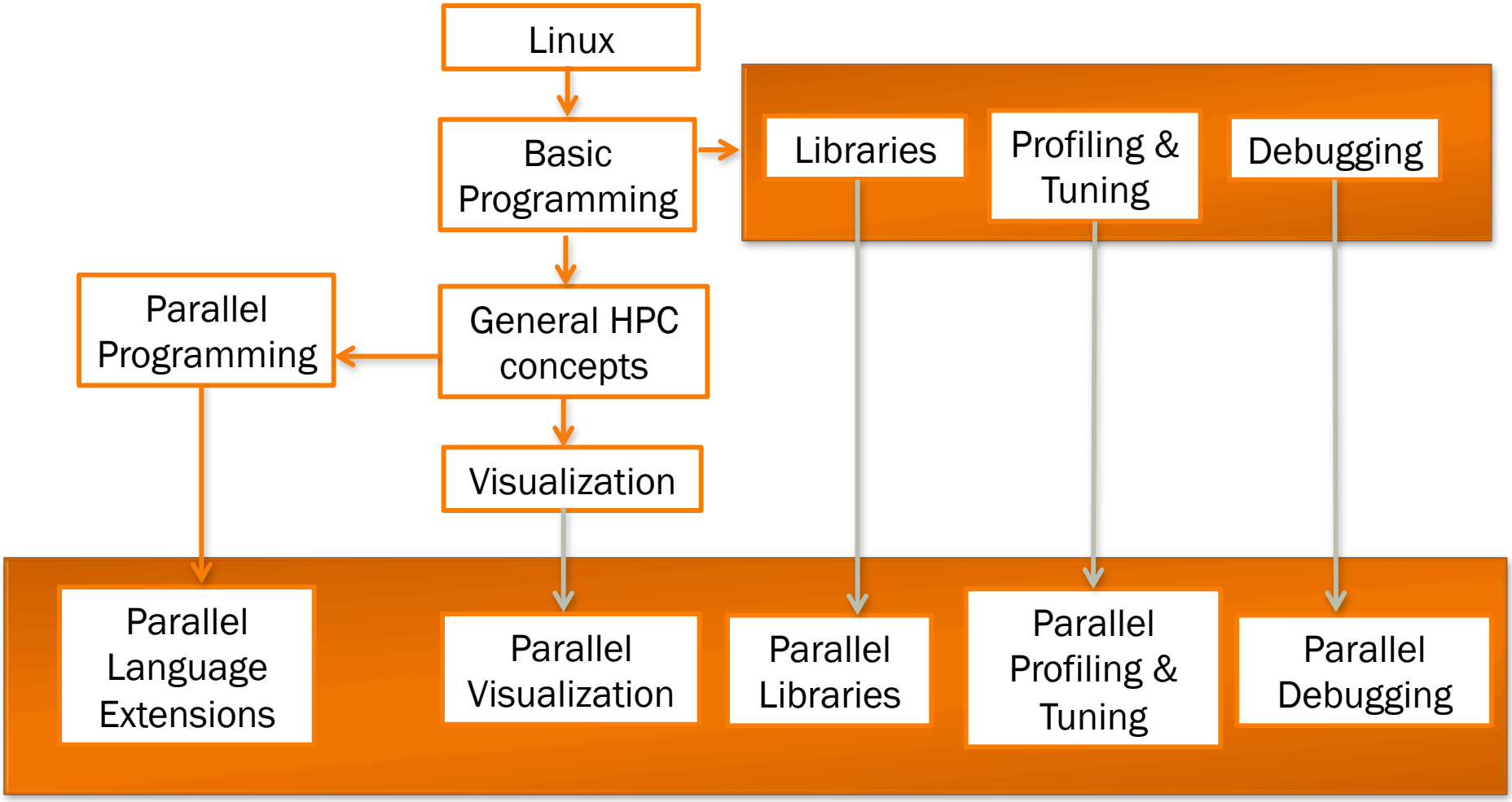
University of Houston
Houston, TX

CACDS

Mission Statement

- The Center for Advanced Computing and Data Systems (CACDS) provides high performance computing resources to advance Tier One research and education goals at the University of Houston (UH).

HPC User Training Road Map



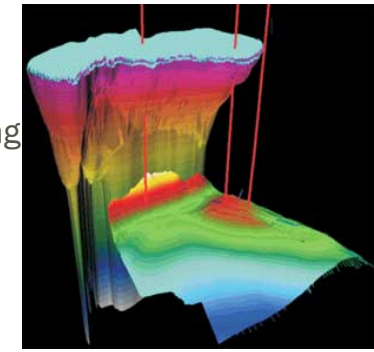
HPC Training

Location: PGH 200
48 seat class room

- Linux
- Shell programming
- Cluster computing on Maxwell
- XSEDE Cyberinfrastructure and allocation process
- Python
- MATLAB
- C++
- Fortran
- Parallel Programming using OpenMP
- Parallel Programming using MPI
- Parallel numerical libraries

Location: PGH 235
12 seat class room

- Nvidia CUDA
  - introductory, intermediate and advanced
  - profiling, debugging & tuning
- OPENACC
  - intermediate, advanced
  - profiling, debugging & tuning
- Intermediate and Advanced MPI
  - profiling, debugging & tuning
- Intel Xeon Phi Programming
  - Native computing
    - OpenMP, Cilk++
  - Offload Execution
  - Symmetric Execution
  - profiling, debugging & tuning
- Visualization at scale
  - Paraview
  - VisIt



CACDS Training
www.cacds.uh.edu/training

UNIVERSITY of HOUSTON
CENTER FOR ADVANCED COMPUTING & DATA SYSTEMS

ABOUT | RESEARCH | EDUCATION | RESOURCES | NEWS

Training Courses

EDUCATION

Training Courses>>
- Events
- Lectures
- Workshops
- Technical Seminars

CACDS has launched several HPC training courses for members of the UH research community. Registration is required to take the courses. Please click the date you would like to attend under each course to register.

CURRENT TRAINING

CLASS NAME
Introduction to Linux
Introduction to Shell Programming
Introduction to Cluster Computing and Maxwell Cluster
Introduction to C++ Programming
Introduction to Python Programming
Introduction to Parallel Programming with OpenMP
GPGPU Parallel Programming with OpenACC
GPGPU Parallel Programming with CUDA
Introduction to Parallel Programming with MPI
Introduction to Matlab
Introduction to Intel Xeon Phi Programming
Introduction to High Performance Numerical Libraries
Visualization with VISIT I

Overview

- Opuntia Cluster
  - Accessing the Cluster
  - Maxwell Specifications
  - User Environment
  - Compiling Programs
  - Portable Batch System (PBS)
  - Batch Jobs
  - Interactive Jobs
  - Killing & Suspending Jobs

- Maxwell Cluster
  - Accessing the Cluster
  - Maxwell Specifications
  - User Environment
  - Compiling Programs
  - Portable Batch System (PBS)
  - Batch Jobs
  - Interactive Jobs
  - Killing & Suspending Jobs

First Access Your Account

- Log into your accounts
  - Username or login = hpc_user**X**
  - Where x = sign in serial number 1 – 47
  - Password = **cacds2014**
  - Use your web browser
    - Firefox, Chromium or Google chrome
- Slides could be downloaded from URL below

/share/apps/tutorials/intro20puntia.pdf

Disclaimer

- Maxwell & Opuntia clusters are for research work under a UH PI
  - Faculty sponsored **RESEARCH** activity only
- You will be using a surrogate cluster for this training
  - You would not be using Maxwell for this training
- We are assuming you are already familiar with the Linux Operating Environment
  - If not try to attend any of the upcoming Linux hands on tutorials offered by CACDS
- After this training, if you do not already have an accounts Maxwell/Opuntia feel free to request one, after authorization from your supervisor (UH Faculty)
  - http://www.rcc.uh.edu/cgi-bin/account.cgi (actual application page)
  - http://www.cacds.uh.edu/OpuntiaAccountRequest

Who Can Use the Clusters?

- Open to all **UH Central** faculty for research purposes
  - Students & postdocs –under UH faculty guidance
  - Collaborators –under UH faculty guidance

What Happens Next?

- How to connect to the Opuntia Cluster:
  - For UNIX/Linux/Mac users
    - Open up terminal console and use Openssh secure shell client to connect
    - syntax:
      - ssh username@opuntia.cacds.uh.edu
      - For X11 support
        - ssh -YX username@opuntia.cacds.uh.edu

Where username is your assigned login ID..

- you might be prompted to accept finger print…
  - Just type “yes” followed be Enter key

- For Windows users
  - Install Putty or any other compatible SSH client to connect
  - Putty URL = http://www.putty.org/

Connecting via OpenSSH Client on UNIX based Systems

```
129-7-249-201:~ jebalunode$ ssh -Y opuntia.cacds.uh.edu -l jebaluno
The authenticity of host 'opuntia.cacds.uh.edu (129.7.54.49)' can't be established.
RSA key fingerprint is a7:e7:7f:16:04:cd:90:23:b7:3a:a0:05:00:c4:41:fc.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'opuntia.cacds.uh.edu,129.7.54.49' (RSA) to the list of known hosts.
Use of University of Houston computing and network facilities requires
prior authorization.  Unauthorized access is prohibited.  Usage may be
subject to  security testing and monitoring.    Abuse   is  subject to
criminal   prosecution.   A  complete manual of  security policies and
procedures is   available at http://www.uh.edu/  in the Administration
directory.

Password:
```

Connecting via OpenSSH Client on UNIX based Systems

```
Password:
Last login: Mon Feb  2 17:17:01 2015 from c-73-55-246-235.hsd1.tx.comcast.net
Rocks 6.1.1 (Sand Boa)
Profile built 11:03 26-Sep-2014

Kickstarted 06:43 26-Sep-2014

 ----------------------------------------------------------------------
                        Opuntia Cluster

  http://en.wikipedia.org/wiki/Opuntia

  This cluster uses SLURM as its job scheduler

  Use   module avail  to list available software
  Use   module add <name>   to set up environment to use <name>
  Example   module add lammps

          Help?? http://support.cacds.uh.edu/

 ----------------------------------------------------------------------
[jebaluno@opuntia ~]$
```

Accessing Tutorial Materials

First login into the cluster:

TYPE AND EXECUTE COMMANDS IN RED!!!

```
ssh  –Y  10.1.1.1
cd
cp /share/apps/tutorials/intro2maxwell.zip  ~

unzip intro2maxwell.zip

cd intro2maxwell
```

The Opuntia Cluster

- Both Shared and Distributed systems
  - single core, multicore (multithreaded) and MPI jobs
- 80+ compute nodes (1700 + cpu cores)
  - http://baragon.hpcc.uh.edu/ganglia/?c=Opuntia
  - 2 Xeon Phi 5110P
  - 4 Nvidia Tesla K40 GPUs
- Operating system: Linux
  - Distribution: Redhat 6.5
- 56 Gigabit network interconnect
- Data Storage:
  - Default: 10+ GB Home directory space per user
  - PI's could request for project directory space to host files needed for simulations
    - Justification might be required

Opuntia Specifications

Hardware Specification Table

Node Type	CPU Type	CPU Socket Count	Total Cores	Memory	Disk Space	Node Count
Login HP DL380	Intel Xeon E5-2680v2 2.8 GHz	2	20	64 GB	2.4 TB	1
Compute HP SL230	Intel Xeon E5-2680v2 2.8 GHz	2	20	64 GB	1 TB	80
Large Memory HP DL560	Intel Xeon E5-4650v2 2.4 GHz	4	40	512 GB	1 TB	2
XLarge Memory HP DL580	Intel Xeon E7-4880v2 2.5 GHz	4	60	1 TB	1 TB	1
GPU Accelerator HP SL250	Intel Xeon E5-2680v2 2.8 GHz	CPU:2	CPU:20	CPU:64 GB	1TB	2
	Tesla K40 GPU	GPU:2	GPU:5760	GPU:24 GB		
Xeon Phi Coprocessor HP SL250	Intel Xeon E5-2680v2 2.8 GHz	CPU:2	CPU:20	CPU:64 GB	1TB	2
	Xeon Phi 5110P	Phi:1	Phi:61	Phi:8 GB		
Storage HP SL4540	Intel Xeon E5-4650v2 2.4 GHz	2	20	64 GB	120 TB	4

Storage

384 TB of NFS storage

Interconnect

Opuntia nodes are connected via 56 Gb/s Ethernet interconnect.

HPC Software Available on Opunta

- Proprietary & Open-Source Software:
  - Examples:
    - Amber, Autodock, Bwa, NwChem, Espresso, Octave, Matlab, Gromacs, Lammps, NAMD , R, NAG, MKL , ASE, GSL, OpenMPI, Intel, Gnu and PGI compilers ….
    - and much more…
  - Proprietary software may be limited to licensed users e.g. VASP
  - Open-source software is accessible to all

- PI could request for installation of additional software
  - PI's licensed propriety application can be installed too
- Software environment management via "**modules**"
  - Allows software to be accessed via an assigned module
  - Dynamic modification of a user's environment

Computing Environment Setup Using Modules

- The module system is used to make software and related settings available easily
  - software environments can be **loaded** and **unloaded** dynamically
- To get a list of available software:
  module avail
  (also useful: module list)
- To clear-out all added modules:
  module purge
- To clear-out all specific modules:
  module rm module1 module2 ....
- To use specific software every time you log-in, place the
  module add command in your ~/.bashrc file i.e:
  echo "module add intel intelmpi" >> ~/.bashrc
- Now add intel compiler and MPI runtime (Openmpi version to your environment)
  module add intelmpi

Compiling Programs

- CPU programs (compiled to execute on central processing unit)
  - Serial Programs
  - Parallel Programs
- GPU programs (compiled to execute on graphics processing unit)
  - GPGPU Programs

Compiling Programs: CPU Serial Program

- Example 1 Program in file: gethostname.c

```
#include <stdio.h>
#include <sys/utsname.h>
int main ( )
{
        struct utsname uts;
        uname (&uts);
        printf ("Process on node %s.\n",
        uts.nodename);
        return 0;
}
```

- To compile:
  - 1. choose compiler

    module add intel

    2. Compile

    icc gethostname.c –o gethostname.exe

Compiling Programs: CPU Parallel Program

Programs using Message Passing Interface (MPI)

- Example: 1 Program in file : gethostname.mpi.c

```
#include <stdio.h>
#include <sys/utsname.h>
#include <mpi.h>
int main (int argc, char *argv[]){
        struct utsname uts;
         int rank;
         MPI_Init (&argc, &argv);
         MPI_Comm_rank (MPI_COMM_WORLD, &rank);
         uname (&uts);
         printf ("Process %d on node %s.\n", rank, uts.nodename);
         MPI_Finalize ();
         return 0;
}
```

- To compile:
  - 1. choose compiler

    module add openmpi

    2. Compile

    mpicc gethostname.mpi.c -o gethostname.mpi.exe

Compiling Programs: GPGPU Parallel Program

- Example 1 Program in file: sortkeys_basic_thrust.cu

```
#include <thrust/host_vector.h>
#include <thrust/device_vector.h>
#include <thrust/sort.h>
int main(void)
{    // generate 16 Million random numbers on the host
     thrust::host_vector<int> h_vec(1 << 24);
     thrust::generate(h_vec.begin(), h_vec.end(), rand);
     thrust::device_vector<int> d_vec= h_vec;  // transfer data to the device
     thrust::sort(d_vec.begin(), d_vec.end());  // sort data on the device (805 Mkeys/sec on GeForce GTX 480)
     thrust::copy(d_vec.begin(), d_vec.end(), h_vec.begin());  // transfer data back to host
     return 0;
}
```

- To compile:
  - 1. choose compiler

```
module  add  cuda-toolkit
```

2. Compile name = nvcc

```
nvcc  sortkeys_basic_thrust.cu   -o   sortkeys_basic_thrust.exe
```

The Queue System (SLURM)

- Simple Linux Utility for Resource Management (aka SLURM) is the resource management service used on the Opuntia Cluster
- Enables you to make more efficient use of your time through scripting computational tasks
- SLURM takes care of running these tasks and returning the results
- If the cluster is full, SLURM holds your tasks and runs them when the resources are available
- SLURM ensures fair sharing of cluster resources (policy enforcement)
  - SLURM ensures optimal/efficient use of available resources

SLURM Commands

Command(s)	Description
squeue	Check status of all jobs
sbatch ./myjob	Submit batch jobs
squeue -u $USER	Check status of just your jobs
srun --pty /bin/bash -l	Submit an interactive job
srun --x11=first --pty /bin/bash -l	Submit an interactive job with X11 support
scontrol hold jobID i.e. qhold 12345	Put a job on hold (before it starts)
scontrol release jobID	Release a job from hold status
scancel #jobID	Delete a job, running or not

Using sbatch with a Job Script

- Syntax/Format:
  - sbatch [sbatch params] [job script file]
- Examples:
  - sbatch myjob

SLURM Batch Job Script File

- Very simple example:
- Serial job requesting 1 CPU, 2 GB memory, 60 hours of walltime

```
#!/bin/bash
#SBATCH  -J   jobname
#SBATCH  -o   jobname.o%j
#SBATCH –-mem=2gb
#SBATCH  -t 60:00:00

./my.program.exe [arguments for my program]
```

Using Parallel Environment to Manage CPU Core Availability

- #SBATCH -N 1 -n 20 --use only 1 node and use a total of 20 processes, 20 CPUs per node.
- #SBATCH -N 2 -n 20 --use only 2 nodes and use a total of 20 processes, 10 CPUs per node.
- #SBATCH -N 2 -n 40 --use only 2 nodes and use a total of 40 processes, 20 CPUs per node.
- #SBATCH -N 4 -n 80 --use only 4 nodes and use a total of 80 processes, 20 CPUs per node.

Sbatch Notification Parameters

- #SBATCH --mail-user=[e-mail address]
  - e-mail address can be a list of email addresses
  - separated by commas.

#SBATCH --mail-type=begin # email me when the job starts
#SBATCH --mail-type=end # email me when the job finishes
#SBATCH --mail-type=all # email me when the job starts & finishes

```
#!/bin/bash
#SBATCH  -o  jobname.o%j  -J o  jobname
#SBATCH --mail-user=jerry@uh.edu
#SBATCH --mail-type=begin  # email me when the job starts
#SBATCH --mail-type=end    # email me when the job finishes

./my.program.exe [arguments for my program]
```

Handling Output Files

- When a job is started it takes its job name from the script file that was submitted.
  - The standard output and error output are sent to file named jobname.o987349 (in your job directory)
-
- #SBATCH -o jobname.o%j name to be used for the job.
- #SBATCH -J jobname name to be used for the job.

Using Sbatch with a Job Script

- Syntax/Format:
  - **sbatch [sbatch params] [script file]**

- More examples on using parameters at the command line:
  - `sbatch –J test1 myjob.sh`
  - `sbatch –N 2 -n 16 -o test2.out –J test2 myjob.sh`
  - `sbatch –t 02:00:00 --mem=4gb myjob.sh`
  - `sbatch –n 32 --mem=64gb myjob.sh`
  - `sbatch -p gpu -N 1 -n 20 my_gpu_job.sh`
- To see all available sbatch options, just run sbatch --help

Using an Interactive Job

- Special kind of batch job
- Useful for debugging applications, short tests, or for computational steering

```
srun  --pty /bin/bash -l

srun -n 8 -t 2:00:00 --pty /bin/bash -l

srun -N 2 -n 8 -t 2:00:00 --pty /bin/bash -l

srun -n 8 -p gpu -t 2:00:00 --pty /bin/bash -l
```

Job Queues

Queue Name	Description
short	Queue to run short jobs (=< 4 hours)
medium	Queue to run medium jobs (1 week allowed)
long	Queue to run very long jobs (2 weeks allowed)
gpu	Queue to run gpu jobs on K 40 equipped nodes (24 hrs allowed)

Note phi and gpu queue are not available/functional on the surrogate cluster

SLURM Script for Serial Program
1 CPU

- Example of a Serial Job Script

  filename: job.gethostname.bash.slurm

```
#!/bin/bash
#SBATCH  -o gethost.out
#SBATCH  -t 00:25:00
#SBATCH  --mail-user=JohnDoe@gmail.com
#SBATCH --mail-type=all


### Run gethostname

time   ./gethostname.exe
```

To submit the job run:

```
sbatch   job.gethostname.bash.slurm
```

SLURM Script for MPI Program

```
#!/bin/bash
#SBATCH -n 20
#SBATCH -t 0:05:05 # Runtime in D-HH:MM
#SBATCH --mem=1000 # Memory pool for all cores (see also --mem-per-cpu)
#SBATCH -o hostname.out
#SBATCH -e hostname.err
#SBATCH --mail-type=END # Type of email notification- BEGIN,END,FAIL,ALL
#SBATCH --mail-user=jerry@uh.edu
# Email to which notifications will be sent

##set up your environment


module add intelmpi

mpirun -np  16 ./gethostname.mpi.exe > output.from.txt
```

See job filename: job.gethostname.mpi.bash.slurm

To submit the job run:

```
sbatch  job.gethostname.mpi.bash.slurm
```

SLURM Script for GPGPU Program

Give me a node with this kind of gpu

Example GPU Job Script

```
#!/bin/bash
#SBATCH -n 1
#SBATCH -t 0:30:05  -p gpu
#SBATCH --mem=10gb
#SBATCH -o  GpuJob.out
#SBATCH --mail-type=all
#SBATCH --mail-user=jerry@uh.edu


    ##set up your environment

    module add cuda-toolkit
    time    ./sortkeys_basic_thrust.exe
```

Load CUDA environment

See job filename: job.gpu.bash or job.gpu.sort.bash.slurm

SLURM Script for Executing Matlab script

Using Proprietary Matlab runtime (sample I)

```
#!/bin/bash
#SBATCH -n 8 -N 1
#SBATCH -t  0:30:05
#SBATCH --mem=10gb
#SBATCH -o matlabjob.out  -J matlabjob
#SBATCH --mail-type=all
#SBATCH --mail-user=jerry@uh.edu

##set up your environment

module add matlab

matlab << EOF
a = 10;  b = 20; c = 30;
d = sqrt((a + b + c)/pi)
exit
EOF
```

See job filename: job.matlab1.bash.slurm

SLURM Script for Executing Matlab script

Using Proprietary Matlab runtime (sample II)

```
#!/bin/bash
#SBATCH -n 8 -N 1
#SBATCH -t 0:30:05
#SBATCH --mem=10gb
#SBATCH -o matlabjob.out  -J matlabjob
#SBATCH --mail-type=all
#SBATCH --mail-user=jerry@uh.edu

module add matlab

## run matlab program compute pseudo inverse for  100  matrices of size 400x400
 time    matlab    < matrix_inversion100_matlab.m
```

See job filename: job.matlab2.bash.slurm

SLURM Script for Executing R script

```
#!/bin/bash
#SBATCH –n 1
#SBATCH  -t 0:30:05
#SBATCH --mem=10gb
#SBATCH –o Rjob.out  -J Rjob
#SBATCH --mail-type=all
#SBATCH --mail-user=jerry@uh.edu



module add R


## run R program
R --vanilla < sample.r
```

See job filename: job.R.bash.slurm

SLURM Script for NAMD2 Job Parallel MPI

```
#!/bin/bash
#SBATCH -n 32 -N 4
#SBATCH -t 0:30:05
#SBATCH --mem=10gb
#SBATCH -o NAMDjob.out  -J NAMDjob
#SBATCH --mail-type=all
#SBATCH --mail-user=jerry@uh.edu



module load namd

mpirun -v  namd2  apoa1.namd
```

See job filename: job.namd.mpi.bash.slurm

Killing Jobs

- One, or a few jobs:
  - scancel [jobID] [jobID] [jobID] …
- Kill all of your jobs:
  - scancel -u $USER
- Kill all of your queued jobs:
  - scancel -u $USER -t PENDING
- Kill all of your running jobs:
  - scancel -u $USER -t RUNNING

Try An Example Job

- Several job scripts files available to use for practice

- Edit job.**.bash.slurm files as needed, then run the job:
  - you can use gedit

- submit the job(s) i.e.
  - sbatch job.**.bash.slurm
  - where ** represents the job pattern you are going to submit

job.gethostname.bash.slurm job.gethostname.mpi.bash.slurm job.gpu.bash.slurm job.matlab.bash.slurm job.namd.bash.slurm

Homework

- Examine the job files and fix potential glitches
- Submit the jobs.
- Record the runtime or execution time and send me these timings as well as the job log files

Overview

- Accessing the Cluster
- Maxwell Specifications
- User Environment
- Compiling Programs
- Portable Batch System (PBS)
- Batch Jobs
- Interactive Jobs
- Killing & Suspending Jobs

First Access Your Account

- Log into your accounts
  - Username or login = hpc_user**X**
  - Where x = sign in serial number 1 – 47
  - Password = **cacds2014**
  - Use your web browser
    - Firefox, Chromium or Google chrome
- Slides could be downloaded from URL below

/share/apps/tutorials/intro2maxwell.pdf

Disclaimer

- Maxwell cluster is for research work under a UH PI
  - Faculty sponsored **RESEARCH** activity only
- You will be using a surrogate cluster for this training in place of the Maxwell Cluster
  - You would not be using Maxwell for this training
- We assuming you are already familiar with the Linux Operating Environment
  - If not try to attend any of the upcoming Linux hands on tutorials offered by CACDS
- After this training, if you do not already have an account on Maxwell feel free to request one, after authorization from your PI(UH Faculty)
  - http://www.rcc.uh.edu/cgi-bin/account.cgi (actual application page)

Who Can Use Maxwell Cluster?

- Open to all UH faculty for research purposes
  - Students & postdocs –under UH faculty guidance
  - Collaborators –under UH faculty guidance

What Happens Next?

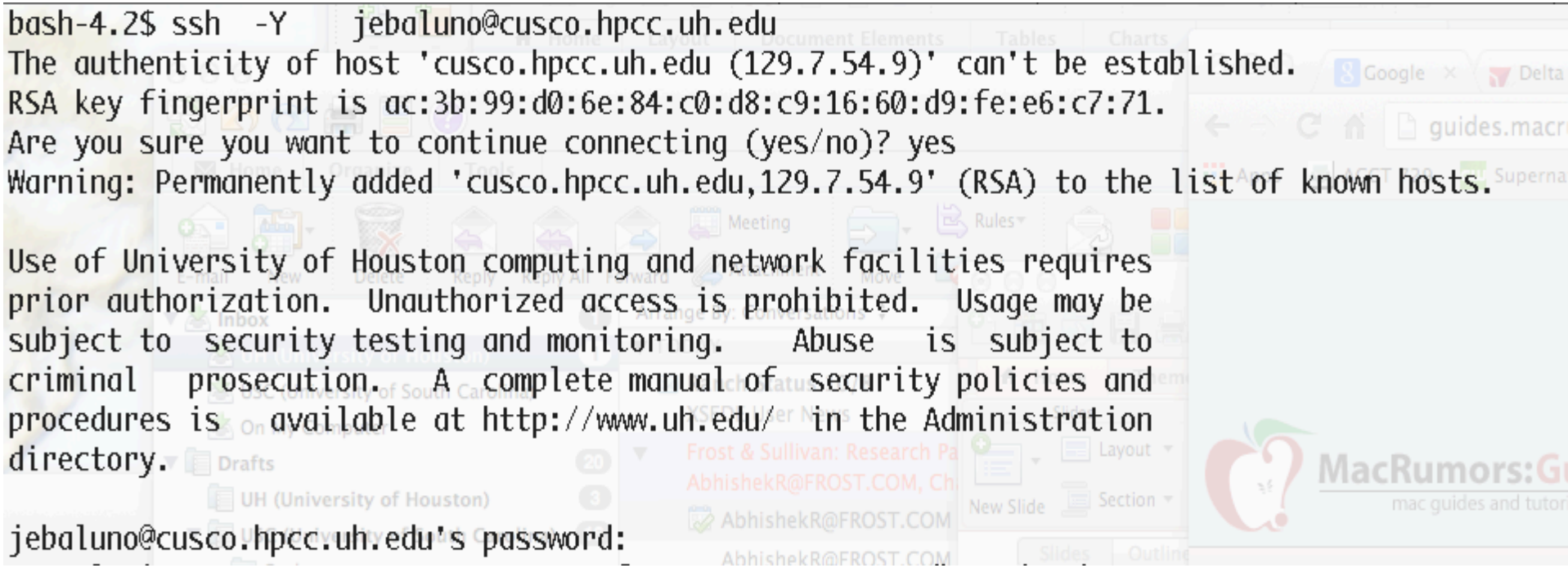
- How to connect to the Maxwell Cluster:
  - For UNIX/Linux/Mac users
    - Open up terminal console and use Openssh secure shell client to connect
    - syntax:
      - ssh username@cusco.hpcc.uh.edu

Where username is your assigned login ID..

  - you might be prompted to accept finger print…
    - Just type “yes” followed be Enter key

  - For Windows users
    - Install Putty or any other compatible SSH client to connect
    - Putty URL = http://www.putty.org/

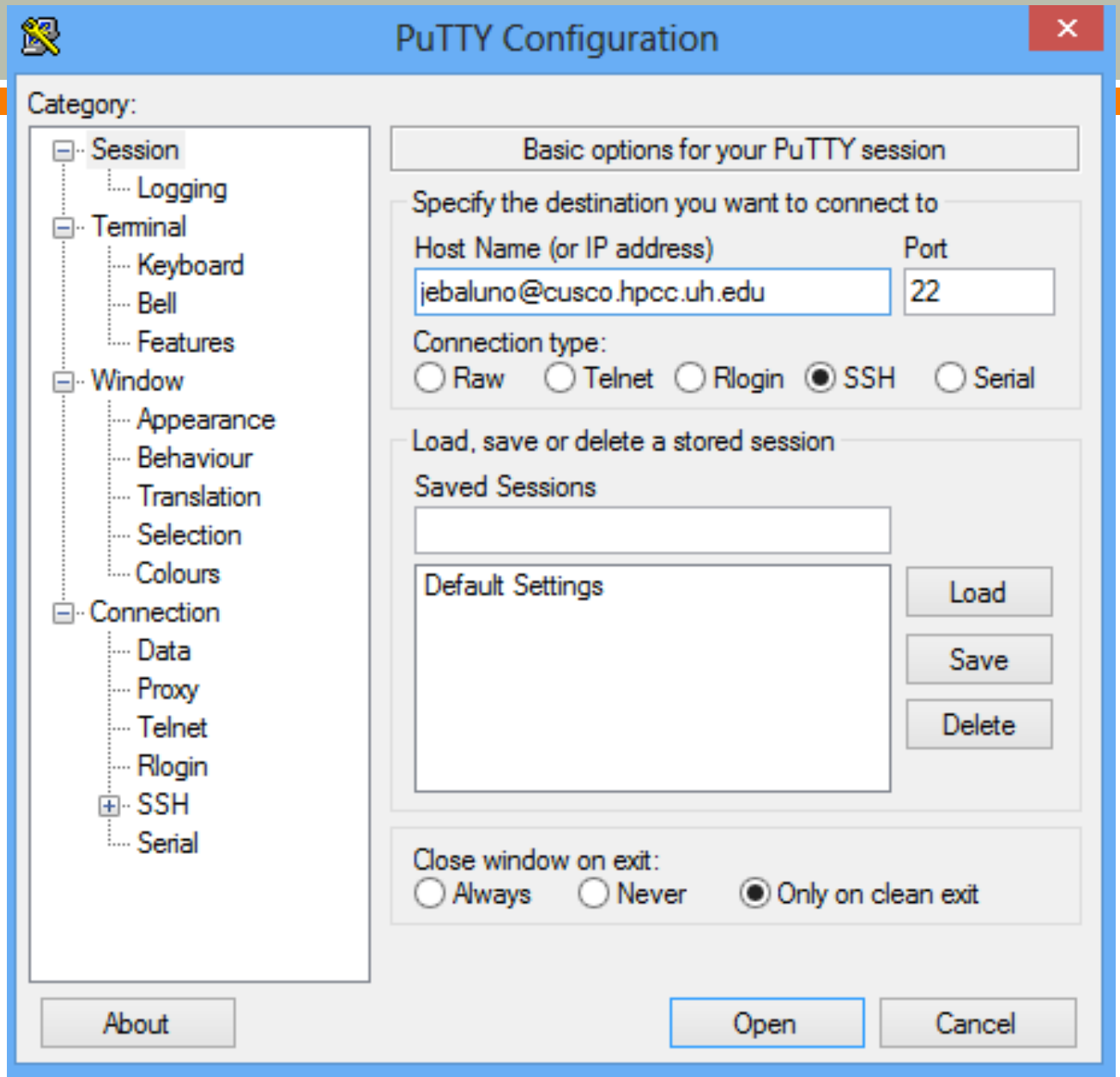
Connecting via OpenSSH Client on UNIX based Systems



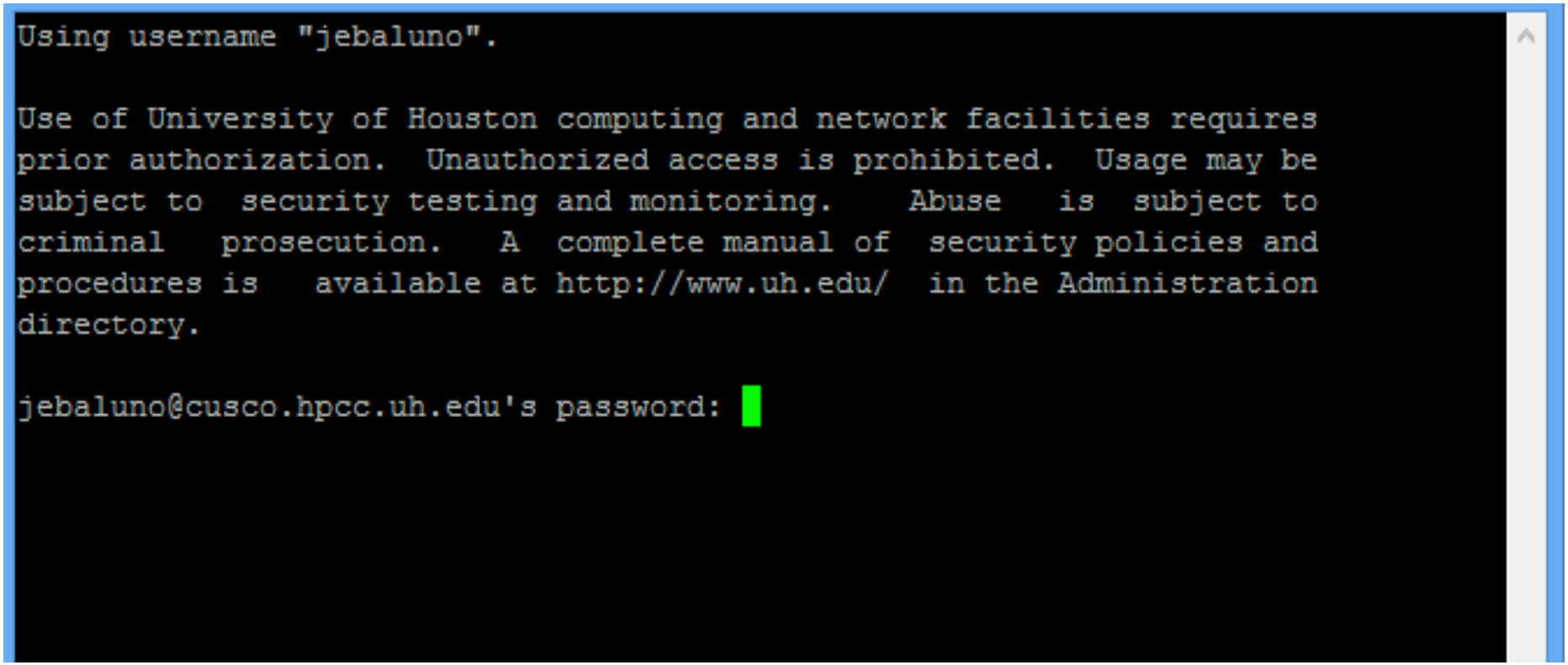
Connecting via OpenSSH Client on UNIX based Systems

```
jebaluno@cusco.hpcc.uh.edu's password:
Last login: Mon Oct  7 11:26:13 2013 from 129-7-249-234.dhcp.uh.edu

 ------------------------------------------------------------------
                          Maxwell Cluster

  Passwords expire every 90 days.

  Please do not run your code on the login node. It can cause login,
  compiling, and other problems.

  Use   module avail  to list available software
  Use   module add <name>   to set up environment to use <name>
  Example   module add matlab

     Operated by IT HPC Group | http://www.rcc.uh.edu
     Help?? Email HPC | help@hpcc.uh.edu
     Quick Reference Doc  |  /usr/local/doc/maxwellquickref.pdf


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

[jebaluno@cusco ~]$ hostname
cusco.hpcc.uh.edu
[jebaluno@cusco ~]$
```

Connecting via Putty on Windows Systems



Connecting via Putty on Windows Systems



Note: you will not be able see your password

Accessing Tutorial Materials

First login into the cluster:

TYPE AND EXECUTE COMMANDS IN RED!!!

```
ssh  –Y  10.1.1.1
cd
cp /share/apps/tutorials/intro2maxwell.zip  ~

unzip intro2maxwell.zip

cd intro2maxwell
```

The Maxwell Cluster

- Both Shared and Distributed systems
  - single core, multicore (multithreaded) and MPI jobs
- 434+ compute nodes (3756+ cpu cores)
  - http://baragon.hpcc.uh.edu/ganglia
  - 22+ Geforce GTX 570 Nvidia GPUs
  - 8+ Nvidia Tesla C2075 GPUs
- Operating system: Linux
  - Distribution: Redhat 6.5
- Gigabit network interconnect
- Data Storage:
  - Default: 10+ GB Home directory space per user
  - PI's could request for project directory space to host files needed for simulations
    - Justification might be required

Specifications

Name	Count	Processor Tyoe	Speed GHz	Total CPU Cores	Memory/ core	GPU
cusco (login node)	1	Quad-Core AMD Opteron(tm) Processor 2354	2.20	8	32+ GB	0.00
compute-X-Y	183	Quad-Core AMD Opteron(tm) Processor 2354	2.20	8	16+ GB	0.00
compute-X-Y	25	Quad-Core AMD Opteron(tm) Processor 8356	2.30	32	64+ GB	0.00
compute-X-Y	39	Quad-Core AMD Opteron(tm) Processor 2376	2.30	8	16.3+ GB	0.00
compute-X-Y	112	Dual Core AMD Opteron(tm) Processor 275	2.20	4	8+ GB	0.00
compute-X-Y	28	AMD Opteron(TM) Processor 6274	2.50	32	65+ GB	0.00
compute-X-Y	22	Intel(R) Xeon(R) CPU 5160	3.00	4	32+ GB	GeForce GTX 570
compute-X-Y	8	Intel(R) Xeon(R) CPU 5160	3.00	4	32+ GB	Tesla C2075

X = Rack Number
Y = Node Number

HPC Software Available on Maxwell

- Proprietary & Open-Source Software:
  - Examples:
    - Amber, Autodock, Bwa, NwChem, Espresso, Orca, R, Octave, Matlab, Gromacs, Lammps, NAMD, Turbomole, VASP, Rosetta, R, NAG, MKL , ASE, GSL, OpenMPI, Intel, Gnu and PGI compilers ….
    - and much more…
  - Proprietary software may be limited to licensed users e.g. Fluent, VASP
  - Open-source software is accessible to all

- PI could request for installation of additional software
  - PI's licensed propriety application can be installed too
- Software environment management via "**modules**"
  - Allows software to be accessed via an assigned module
  - Dynamic modification of a user's environment

Computing Environment Setup Using Modules

- The module system is used to make software and related settings available easily
  - software environments can be **loaded** and **unloaded** dynamically
- To get a list of available software:
  module avail
  (also useful: module list)
- To clear-out all added modules:
  module purge
- To clear-out all specific modules:
  module rm module1 module2 ....
- To use specific software every time you log-in, place the
  module add command in your ~/.bashrc file i.e:
  echo "module add intel openmpi" >> ~/.bashrc
- Now add intel compiler and MPI runtime (Openmpi version to your environment)
  module add openmpi

Compiling Programs

- CPU programs (compiled to execute on central processing unit)
  - Serial Programs
  - Parallel Programs
- GPU programs (compiled to execute on graphics processing unit)
  - GPGPU Programs

Compiling Programs: CPU Serial Program

- Example 1 Program in file: gethostname.c

```
#include <stdio.h>
#include <sys/utsname.h>
int main ( )
{
        struct utsname uts;
        uname (&uts);
        printf ("Process on node %s.\n",
        uts.nodename);
        return 0;
}
```

- To compile:
  - 1. choose compiler

    module add intel

    2. Compile

    icc gethostname.c –o gethostname.exe

Compiling Programs: CPU Parallel Program

Programs using Message Passing Interface (MPI)

- Example: 1 Program in file : gethostname.mpi.c

```
#include <stdio.h>
#include <sys/utsname.h>
#include <mpi.h>
int main (int argc, char *argv[]){
        struct utsname uts;
         int rank;
         MPI_Init (&argc, &argv);
         MPI_Comm_rank (MPI_COMM_WORLD, &rank);
         uname (&uts);
         printf ("Process %d on node %s.\n", rank, uts.nodename);
         MPI_Finalize ();
         return 0;
}
```

- To compile:
  - 1. choose compiler

    module add openmpi

    2. Compile

    mpicc gethostname.mpi.c -o gethostname.mpi.exe

Compiling Programs: GPGPU Parallel Program

- Example 1 Program in file: sortkeys_basic_thrust.cu

```
#include <thrust/host_vector.h>
#include <thrust/device_vector.h>
#include <thrust/sort.h>
int main(void)
{    // generate 16 Million random numbers on the host
     thrust::host_vector<int> h_vec(1 << 24);
     thrust::generate(h_vec.begin(), h_vec.end(), rand);
     thrust::device_vector<int> d_vec= h_vec;  // transfer data to the device
     thrust::sort(d_vec.begin(), d_vec.end());  // sort data on the device (805 Mkeys/sec on GeForce GTX 480)
     thrust::copy(d_vec.begin(), d_vec.end(), h_vec.begin());  // transfer data back to host
     return 0;
}
```

- To compile:
  - 1. choose compiler

```
module  add  cuda-toolkit
```

2. Compile name = nvcc

```
nvcc  sortkeys_basic_thrust.cu   -o   sortkeys_basic_thrust.exe
```

The Queue System (PBS)

- Portable Batch System (aka PBS) is the resource management service used on the Maxwell Cluster
- Enables you to make more efficient use of your time through scripting computational tasks
- PBS takes care of running these tasks and returning the results
- If the cluster is full, PBS holds your tasks and runs them when the resources are available
- PBS ensures fair sharing of cluster resources (policy enforcement)
  - PBS ensures optimal/efficient use of available resources

PBS Commands

Command(s)	Description
qstat	Check status of all jobs
qsub ./myjob	Submit batch jobs
qstat -u $USER	Check status of just your jobs
qsub -I	Submit an interactive job
qhold jobID i.e. qhold 12345	Put a job on hold (before it starts)
qrls jobID	Release a job from hold status
qdel #jobID	Delete a job, running or not

Using qsub with a Job Script

- Syntax/Format:
  - qsub [qsub params] [job script file]
- Examples:
  - qsub myjob

PBS Batch Job Script File

- Very simple example:
- Serial job requesting 1 CPU, 2 GB memory, 60 hours of walltime

```
#!/bin/bash
#PBS   -N   jobname
#PBS   -l  mem=2gb,walltime=60:00:00
#PBS   -j  y

cd $PBS_O_WORKDIR

./my.program.exe [arguments for my program]
```

Using Parallel Environment to Manage Memory and CPU Core Availability

- #PBS -l nodes=16:ppn=1 --use only one CPU core per node and use a total of 16 processes.
- #PBS -l nodes=8:ppn=2 --use only 2 CPU cores per node and use a total of 16 processes
- #PBS -l nodes=4:ppn=4 --use only 4 CPU cores per node and use a total of 16 processes
- #PBS -l nodes=8:ppn=8 --use only 8 CPU cores per node and use a total of 64 processes

qsub Notification Parameters

- #PBS -M [e-mail address]
  - e-mail address can be a list of email addresses
  - separated by commas.

- #PBS -m bea or -m be or -m e
  - send an email when the job **b**egins, **e**nds, or is **a**borted.

Example:

```
#!/bin/bash
#PBS   -N   jobname
#PBS   -j  y
#PBS  -M   jerry@uh.edu
#PBS  -m   abe

cd $PBS_O_WORKDIR

./my.program.exe [arguments for my program]
```

Sample Notification Email

PBS JOB 3687125.cusco.hpcc.uh.edu

root

Sent: Monday, October 7, 2013 at 6:27 AM

To: jebalunode@uh.edu

```
PBS Job Id: 3687125.cusco.hpcc.uh.edu
Job Name:   waterint-ACC-H17
Exec host:  compute-7-25/31+compute-7-25/30+compute-7-25/29+compute-7-25/28+compute-7-
25/27+compute-7-25/26+compute-7-25/25+compute-7-25/24+compute-7-25/23+compute-7-25/22+c
7-25/21+compute-7-25/20+compute-7-25/19+compute-7-25/18+compute-7-25/17+compute-7-
25/16+compute-7-25/15+compute-7-25/14+compute-7-25/13+compute-7-25/12+compute-7-25/11+c
7-25/10+compute-7-25/9+compute-7-25/8+compute-7-25/7+compute-7-25/6+compute-7-25/5+comp
25/4+compute-7-25/3+compute-7-25/2+compute-7-25/1+compute-7-25/0
Execution terminated
Exit_status=0
resources_used.cput=08:46:15
resources_used.mem=352472kb
resources_used.vmem=35667756kb
resources_used.walltime=00:33:07
```

Handling Output Files

- When a job is started it takes its job name from the script file that was submitted.
  - The standard output and error output are sent to files named jobname.o987349 and jobname.e987349 (in your job directory)
- The following parameters modify this behavior:
  - -e [path] standard error output file.
  - -o [path] standard output file
  - -j y merge the error and standard output
- #PBS -N jobname name to be used for the job.

Using qsub with a Job Script

- Syntax/Format:
  - **qsub [qsub params] [script file]**

- More examples on using parameters at the command line:
  - `qsub -N test1 myjob.sh`
  - `qsub -l nodes=2:ppn=2 -o test2.out -N test2 myjob.sh`
  - `qsub -l walltime=02:00:00,mem=4gb myjob.sh`
  - `qsub -l nodes=2:ppn=32,mem=64gb myjob.sh`
  - `qsub -q gpu -l nodes=1:ppn=4 my_gpu_job.sh`
- To see all available qsub options, just run qsub --help

Using an Interactive Job

- Special kind of batch job
- Useful for debugging applications, short tests, or for computational steering


```
qsub -I
qsub -I -l  walltime=2:00:00
qsub -I -l walltime=2:00:00,nodes=2:ppn=8
qsub -I -l  walltime=2:00:00  -q gpu
```

Job Queues

Queue Name	Description
short	Queue to run short jobs (=< 4 hours)
medium	Queue to run medium jobs (1 week allowed)
long	Queue to run very long jobs (2 weeks allowed)
gpu	Queue to run gpu jobs on GTX 570 equipped nodes (2 weeks allowed)
gpu-tesla	Queue to run gpu jobs on Testla C2075 equipped nodes (2 weeks allowed)

USE “qstat -q” to probe the queues installed

Note gpu-tesla and gpu queue are not available/functional on the surrogate cluster

PBS Script for Serial Program
1 CPU

- Example of a Serial Job Script

filename: job.gethostname.bash

```
#!/bin/bash
#PBS -N gethost
#PBS  -l mem=8gb,walltime=00:25:00
#PBS  -l nodes=1:ppn=1
#PBS  -M JohnDoe@gmail.com
#PBS -m bea

cd $PBS_O_WORKDIR

### Run gethostname

time   ./gethostname.exe
```

To submit the job run:
qsub job.gethostname.bash

PBS Script for MPI Program

```
#!/bin/bash
#PBS -N gethostmpi
#PBS -l nodes=2:ppn=8,pmem=1gb
#PBS -S /bin/bash
#PBS -l walltime=00:05:00
#PBS -M monkeybrain@sc.edu
#PBS -m bea

cd $PBS_O_WORKDIR
##set up your environment
source /etc/profile.d/modules.sh

module add openmpi

mpirun –np  16 ./gethostname.mpi.exe > output.from.txt
```

See job filename: job.gethostname.mpi.bash

To submit the job run:

```
qsub job.gethostname.mpi.bash
```

PBS Script for GPGPU Program

Give me a node with this kind of gpu

Example GPU Job Script

```
#!/bin/bash
#PBS -q gpu
#PBS -N sortrandomkeys
#PBS -l walltime=00:05:00,nodes=1:ppn=4
#PBS -M monkeybrain@gmail.com
#PBS –m bea

 cd $PBS_O_WORKDIR
##set up your environment
 source /etc/profile.d/modules.sh
module add cuda-toolkit
time    ./sortkeys_basic_thrust.exe
```

Load CUDA environment

See job filename: job.gpu.bash or job.gpu.sort.bash

PBS Script for Executing Matlab script

Using Proprietary Matlab runtime (sample I)

```
#!/bin/bash
#PBS -N matlabjob
#PBS -l nodes=1:ppn=1,pmem=1gb
#PBS -S /bin/bash
#PBS -l walltime=00:05:00

cd $PBS_O_WORKDIR

##set up your environment
source /etc/profile.d/modules.sh
module add matlab

matlab << EOF
a = 10;  b = 20; c = 30;
d = sqrt((a + b + c)/pi)
exit
EOF
```

See job filename: job.matlab1.bash

PBS Script for Executing Matlab script
Using Proprietary Matlab runtime (sample II)

```
#!/bin/bash
#PBS -N matlabjob
#PBS -S /bin/bash
#PBS -l walltime=20:05:00,pmem=1gb,nodes=1:ppn=4
#PBS -M monkeybrain@uh.edu
#PBS -m bea
##set up your environment
cd  $PBS_O_WORKDIR
source /etc/profile.d/modules.sh
module add matlab

## run matlab program compute pseudo inverse for  100  matrices of size 400x400
time    matlab    < matrix_inversion100_matlab.m
```

See job filename: job.matlab2.bash

PBS Script for Executing R script

```
#!/bin/bash
#PBS -N R-job
#PBS -S /bin/bash
#PBS -l walltime=0:05:00,pmem=1gb,nodes=1:ppn=1
#PBS -M monkeybrain@uh.edu
#PBS -m bea
##set up your environment
cd  $PBS_O_WORKDIR
source /etc/profile.d/modules.sh
module add R

## run R program
R –vanilla < sample.r
```

See job filename: job.R.bash

PBS Script for NAMD2 Job Parallel MPI

```
#!/bin/bash
#PBS -l nodes=1:ppn=4
#PBS -l walltime=00:05:00
#PBS -l pmem=1gb
#PBS -N  namd
#PBS –V

cd $PBS_O_WORKDIR



module load namd

mpirun –v  namd2  apoa1.namd
```

See job filename: job.namd.mpi.bash

Killing Jobs

- One, or a few jobs:
  - qdel [jobID] [jobID] [jobID] …
- Kill all of your jobs:
  - qselect -u $USER | xargs qdel
- Kill all of your queued jobs:
  - qselect -u $USER -s Q | xargs qdel
- Kill all of your running jobs:
  - qselect -u $USER -s R | xargs qdel

Try An Example Job

- Several job scripts files available to use for practice

- Edit job.**.bash files as needed, then run the job:
  - you can use gedit

- submit the job(s) i.e.
  - qsub job.**.bash
  - where ** represents the job pattern you are going to submit

job.gethostname.bash job.gethostname.mpi.bash job.gpu.bash job.matlab.bash job.namd.bash

Homework

- Examine the job files and fix potential glitches
- Submit the jobs.
- Record the runtime or execution time and send me these timings as well as the job log files

Support

- CACDS Support
- Visit ➔ http://support.cacds.uh.edu
  - To Open A Support Ticket

- Jerry Ebalunode
  - jebalunode@uh.edu
  - PGH 231
  - Office hours: 10-12pm Wednesday
  - 713-743-1755