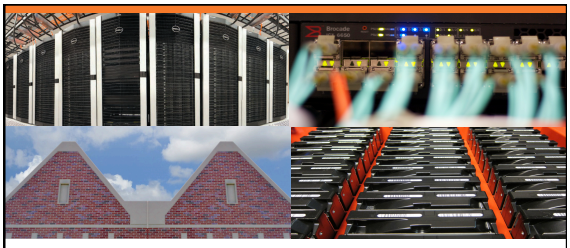
Running Jobs, Submission Scripts, Modules

Matt Gitzendanner
magitz@ufl.edu

6/3/14

UF Research Computing
Information Technology
Home of High-Performance Computing and HiPerGator

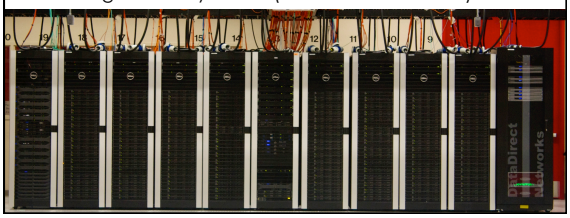


UF RISING to National Preeminence

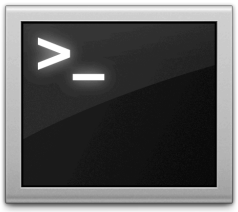
HiPerGator

The University of Florida Supercomputer for Research

- 16,384 cores–*total of about 21,000 cores today*
- Infiniband interconnect
- >3PB fast, high-availability, storage
- **GPGPUs**
- Large memory nodes (**512GB to 1TB of RAM**)

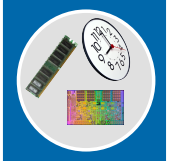


UF Research Computing



How do I get my jobs started?

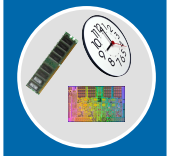
Cluster basics

User interaction	Scheduler	Compute resources
		
Login server (Head node)	Tell the scheduler what you want to do	Your job runs on the cluster

Scheduling a job

- Need to tell scheduler what you want to do
  - **How many CPUs** you want and how you want them grouped
  - **How much RAM** your job will use
  - **How long** your job will run
  - The commands that will be run

Scheduler



Tell the scheduler what you want to do

UF Research Computing

- Ordinary Shell Script

```
#!/bin/bash

date
module load test_app
test_app –i file.txt
```

Read the manual for your application

Commands typed on the command line can be put in a script.

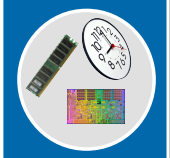
UF Research Computing

- Submission Script

```
#!/bin/bash
#PBS -N My_Job_Name
#PBS -M Joe_Shmoe@ufl.edu
#PBS -m abe
#PBS -o My_Job.log
#PBS -e My_Job.err
#PBS -l nodes=1:ppn=1
#PBS –l pmem=900mb
#PBS -l walltime=00:05:00

cd $PBS_O_WORKDIR
date
module load test_app
test_app –i file.txt
```

Scheduler



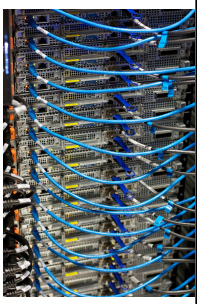
Tell the scheduler what you want to do

Nodes and processors

Single processor apps:
`#PBS -l nodes=1:ppn=1`

Threaded (& MPI) apps:
`#PBS -l nodes=1:ppn=4`

MPI apps:
`#PBS -l nodes=2:ppn=32`



HiPerGator

The University of Florida Supercomputer for Research

- ***64 cores per node***
  - If RAM allows, MPI jobs under ~32 cores, should use nodes=1:ppn=##
  - Some older nodes have 4-16 cores

MPI

- Up to 32 cores: nodes=1:ppn=##
- Use the --bind-to-core flag
  - `mpiexec --bind-to-core my_app …`

Most important question for any parallel application: Does it scale?

RAM

`#PBS –l pmem=900mb`

- Lots to consider, but do your best at estimating RAM needed for job
- Over about 4GB of RAM, "costs" toward CPU allocation

Wasted RAM leads to idle CPUs and low job throughput

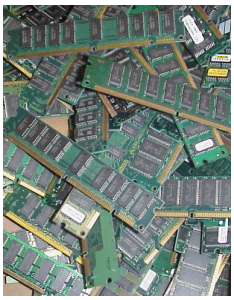
End-of-job emails:

```
#PBS -M Joe_Shmoe@ufl.edu
#PBS -m abe
```

```
PBS Job Id: 358634.moab.ufhpc
Job Name:   NR.25.nex
Exec host:  c7a-s1/60
Execution terminated
Exit_status=0
resources_used.cput=07:16:09
resources_used.mem=251348kb
resources_used.vmem=318916kb
resources_used.walltime=07:16:52
```


RAM– bigmem queue

- For jobs asking for over 16GB per core (pmem)
- `#PBS -q bigmem`
- 1TB, 750GB and 512GB nodes



Processor Equivalents

- Accounts for large RAM requests
- Average ~4GB RAM/core

1 core, 10GB RAM: ~2.5 PEs
1 core, 60GB RAM: ~15 PEs

- Non-investor's limit: 8 PEs
- Investor limits are based on PEs

- `pbs_info –f <job_file>`

Walltime

`#PBS –l walltime=00:50:00`

- Fairly straight forward
- As with all resource requests, accuracy helps ensure ***your*** jobs and all other jobs will run sooner

	Maximum	Short	Long
Investor	31 days	<12 hrs	7 days
Other	7 days	<12 hrs	3 days

UF Research Computing

- Job Management
  - `qsub <file_name>`: job submission
  - `qstat -u <user>`: check queue status
  - `showq -r -u <user>`: shows job efficiency
  - `qdel <JOB_ID>`: job deletion
  - `checkjob -v <job number>` (shows PE value)
  - `pbs_info –f my_job.pbs` (get job PE and group resources before submitting a job)

Lots of jobs

- What about running lots of jobs?

Lots of jobs

- You can script your job submission, ***BUT***:
  - How long will each job run?
    - Many short (<20 minutes) jobs are inefficient
      - Scheduling overhead

Pipettes only $1.99 each! *

* Plus $1.50 shipping per order

Would you order one at a time
or
place one order for 100?

Lots of jobs

```
#!/bin/bash
#PBS -l nodes=1:ppn=1
#PBS -l walltime=00:05:00
#PBS –l pmem=900mb

cd $PBS_O_WORKDIR
date
module load test_app
test_app –i file1.txt
test_app –i file2.txt
test_app –i file3.txt
…
test_app –i fileN.txt

module load step2
step2 -i output.txt
```

You can very easily run multiple tasks in a single job script

Task Arrays

- `#PBS -t 1-100` Runs 100 tasks all submitted at once
- `#PBS -t 1-100%20` Will throttle to run 20 tasks at at time
- `$PBS_ARRAYID`

```
#PBS -t 1-100%10
cd $PBS_O_WORKDIR
module load my_app
file=`ls *.txt | head -n $PBS_ARRAYID | tail -n 1`
my_app -i $file -cpus $PBS_NUM_PPN
```


Lots of jobs

- If you do submit lots of jobs:

There is probably a better way…ask for help

  - **2,000-3,000** jobs maximum at a time
  - Add a ½ second pause between each job:

```
for i in $LIST
do
  qsub job.pbs -v INPUT=$i
  sleep 0.5
done
```

  - Consider how many jobs will run at once: what is your group's PE limit?
    - Will your group like you when you submit 2,000 jobs?

Some helpful environment variables

- `$PBS_O_WORKDIR` : the directory where you typed qsub
- `$PBS_JOBID` : the unique job id: e.g. 24461774.moab.ufhpc
- `$TMPDIR` : temporary directory for each job on compute node's local disk, good for jobs with lots of small I/O
- `$PBS_NUM_PPN` : Number of processors for single node job, use this when starting a threaded application to tell it how many processors to use. Prevents needing to change in multiple places. E.g. nodes=1:ppn=4 , `blastn -num_threads $PBS_NUM_PPN`
- `$PBS_JOBNAME` : Name your gave your job with #PBS –N

So what is this "module" thing?

- **lmod**—Implementation of Environment Modules developed at TACC
- Allows easy management of user's environment

The standard way

```
PATH=$PATH:/some/long/path/to/application
export $PATH
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/long/path/to/
place/I/probably/cant/find
export $LD_LIBRARY_PATH
```

- Need to track down paths to applications, libraries, etc.
- Multiple compilers, and MPI implementations
- Manage dependencies
- Multiple versions of apps

The module way

- `module load trinity`
- Automatically:
  - Sets, `$HPC_TRINITY_DIR`
    - To run Inchworm, simply type
      `inchworm --reads reads.fa --run_inchworm [opts]`
  - Loads Bowtie and Allpaths, two Trinity dependencies
    - You don't need to hunt those down, or worry if they are in your path or not

Module discovery

- `module spider`
  - List everything
- `module spider cl`
  - List applications that have cl in name
- `module spider amber/12`
  - List details about this version of AMBER
- `module key molecular`
  - Keyword search for applications

Multiple versions

```
[magitz@submit1 ~]$ module spider gaussian
Rebuilding cache file, please wait ... done

----------------------------------------------------------------
gaussian:
----------------------------------------------------------------
 Description:
   A software for electronic structure modeling

 Versions:
   gaussian/e01
   gaussian/g03
   gaussian/g09

----------------------------------------------------------------
To find detailed information about gaussian please enter the full name.
For example:

  $ module spider gaussian/g09
----------------------------------------------------------------
```


Multiple variants of a version

```
[magitz@submit1 ~]$ module spider mrbayes/3.2.1
Rebuilding cache file, please wait ... Done
----------------------------------------------------------
mrbayes: mrbayes/3.2.1

----------------------------------------------------------
Description:
      Bayesian inference of phylogeny

    This module can be loaded directly: module load
mrbayes/3.2.1

    Additional variants of this module can also be loaded
after the loading the following modules:

      intel/2012, openmpi/1.6
```


Module loading

- `module load raxml`
- `module load intel raxml`
- `module load intel openmpi raxml`
- `module load intel/12 openmpi/1.6 raxml/3.2`
- `module unload raxml`

Module swapping

- `module load intel openmpi abyss`
- `module list`

```
Currently Loaded Modules:
  1) intel/2012  2) openmpi/1.6  3) abyss/default
```

- `module swap openmpi/1.6 openmpi/1.5.5`

```
Due to MODULEPATH changes the following modules
have been reloaded:
  1) abyss
```


Basic commands

- `module spider`
- `module spider gaussian`
- `module avail`
- `module list`
- `module load clustalw`
- `module load python/2.6.5`
- `module add intel openmpi`
- `module load intel/12 openmpi/1.6 mrbayes`
- `module del/rm/unload clustalw`

Let's look at some examples

- Examples of job scripts in:
  `/scratch/lfs/bio/training/2013-06-04/wordcloud/`
- Job scripts can have many commands
- Jobs should do a decent amount of work (> 20 minutes)
- qsub can pass variables into script with –v flag
  `qsub my_script.pbs -v FILE=f1.txt,OUT=outdir/out1.txt,size=5`
- Limit number of jobs/tasks to 2,000-3,000

UF Research Computing

- Help and Support
  - Help Request Tickets
    - https://support.hpc.ufl.edu
    - For any kind of question or help requests
    - Searchable database of solutions
  - We are here to help!
    - support@hpc.ufl.edu

UF Research Computing

- Help and Support (Continued)
  - http://wiki.hpc.ufl.edu
    - Documents on hardware and software resources
    - Various user guides
    - Many sample submission scripts
  - http://hpc.ufl.edu/support
    - Frequently Asked Questions
    - Account set up and maintenance