Using the Sapelo2 Cluster at the GACRC

Cluster New User Training Hands-out

Georgia Advanced Computing Resource Center (GACRC)
EITS/University of Georgia
Zhuofei Hou  zhuofei@uga.edu
Outline

• What You Need to Know about Sapelo2 Cluster
• Work on Sapelo2 Cluster
• Appendix
• Guideline and Practical Tips
What You Need to Know about Sapelo2 Cluster

1. Cluster Overview
2. Storage Environment
3. Computing Resources
4. Software Environment
Sapelo2 Cluster

1. ssh with MyID and password
2. Verify with Archpass Duo two-factor authentication

- Node: Computer for a specific function on cluster, e.g., login node
- Queue: Collection of compute nodes for specific computing need
- Cluster: Nodes + Drives, all connected by network
Cluster Overview

1. Sapelo2 cluster is a Linux (64-bit Centos 7) high performance computing (HPC) cluster

2. You can log on to 2 nodes: Login node (sapelo2.gacrc.uga.edu) and Transfer node (xfer.gacrc.uga.edu)

3. From Login node, you can open Interactive node using qlogin command

4. You have 4 directories: Home, Global Scratch, Storage and Local Scratch

5. You can submit jobs to 4 computational queues: batch, highmem_q, gpu_q, grpBuyin_q

6. You can use more than 400 modules installed on cluster (as of 05/17/2018)
## Storage Environment – 4 Directories

<table>
<thead>
<tr>
<th>4 Directories</th>
<th>Role</th>
<th>Quota</th>
<th>Accessible from</th>
<th>Intended Use</th>
<th>Backuped</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>/home/MyID</code></td>
<td>Home</td>
<td>100GB</td>
<td>Login Transfer</td>
<td>Static data: 1. Scripts, source codes 2. Local software</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Interactive</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>/lustre1/MyID</code></td>
<td>Global Scratch</td>
<td>No Limit</td>
<td>Interactive</td>
<td>Current job data: data being read/written by running jobs</td>
<td>No</td>
<td>User to clean up! *Subject to deletion in 30 days</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>/project/abclab</code></td>
<td>Storage</td>
<td>1TB (initial)</td>
<td>Transfer</td>
<td>Temporary data parking: non-current active data</td>
<td>Yes</td>
<td>Group sharing possible</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>/scratch</code></td>
<td>Local Scratch</td>
<td>200GB (regular)</td>
<td>Compute</td>
<td>Jobs with heavy disk I/O</td>
<td>No</td>
<td>1. User to clean up when job exists from the node! 2. Persistent data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Storage Environment (Cont.) – Accessing Directories from Nodes

- **Login**
  - qlogin
  - exit

- **Interactive**
  - /home/MyID
  - /lustre1/MyID

- **Transfer**
  - /project/abclab

- User
  - log on using ssh

Log on using ssh
Storage Environment (Cont.) - Intended Use of Directories

I/O speed

Fast

/lustre1/MyID  \(\leftarrow\) **Current Job Data** being used by **current running jobs** on cluster

/home/MyID  \(\leftarrow\) **Static Data** being used frequently and **not being modified often**, e.g., scripts or local software

/project/abclab  \(\leftarrow\) **Non-current Active Data** to be analyzed in the future, e.g., 2 months

User’s Local Computer  \(\leftarrow\) **Final Data**, e.g., final outputs and results
## Computing Resources – 4 Computational Queues

<table>
<thead>
<tr>
<th>Queue</th>
<th>Feature</th>
<th>Total Nodes</th>
<th>RAM(GB) /Node</th>
<th>Max RAM(GB) /Single-node Job</th>
<th>Cores /Node</th>
<th>Processor Type</th>
<th>GPU Cards /Node</th>
<th>InfiniBand</th>
</tr>
</thead>
<tbody>
<tr>
<td>batch</td>
<td>Intel</td>
<td>30</td>
<td>64</td>
<td>62</td>
<td>28</td>
<td>Intel Xeon</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AMD</td>
<td>40</td>
<td>128</td>
<td>125</td>
<td>48</td>
<td>AMD Opteron</td>
<td></td>
<td></td>
</tr>
<tr>
<td>highmem_q</td>
<td>Intel</td>
<td>1</td>
<td>1024</td>
<td>997</td>
<td>28</td>
<td>Intel Xeon</td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>AMD</td>
<td>2</td>
<td>512</td>
<td>503</td>
<td>48</td>
<td>AMD Opteron</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gpu_q</td>
<td>GPU</td>
<td>2</td>
<td>96</td>
<td>94</td>
<td>12</td>
<td>Intel Xeon</td>
<td>7 NVIDIA K20Xm</td>
<td></td>
</tr>
<tr>
<td>grpBuyin_q</td>
<td>variable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Software Environment

1. Software names are long and have a Easybuild toolchain name associated to it, e.g., foss-2016b

2. Software names are case-sensitive!

   - `ml avail`: List all available software modules installed on cluster
   - `ml moduleName`: Load a module into your working environment
   - `ml`: List modules currently loaded
   - `ml unload moduleName`: Remove a module from your working environment
   - `ml spider pattern`: Search module names matching a pattern (case-insensitive)
I am on Login node sapelo2-sub2!

```
$ type space-bar to go forward; key of b to go backward; key of q to quit!
```

```
zhufei@sapelo2-sub2:~
zhufei@sapelo2-sub2:~
zhufei@sapelo2-sub2:~
zhufei@sapelo2-sub2:~
$ ml avail

Data/cache/spiderT.old  all/Q4AST/4.2-Python2.7.14  all/smrtlink/5.8.1.9585
Data/cache/spiderT  (D)  all/falcon/02262018_unzip  ogrt-tracking/0.5.0  (L)
Singularity  all/latex/2017  tools/EasyBuild/3.4.1
StdEnv  (L)  all/matlab/R2017b  tools/EasyBuild/3.5.0
all/EasyBuild/3.4.1  (D)  all/photoscan/1.3.4-openc1  tools/EasyBuild/3.5.1  (D)
all/EasyBuild/3.5.0  all/photoscan/1.4.0  (D)
all/EasyBuild/3.5.1  all/smoke/4.5

ABYSS/1.9.0-foss-2016b
ART/2016.06.05-foss-2016b
ATK/2.22.0-foss-2016b
ATLAS/3.10.2-GCC-6.4.0-2.28-LAPACK-3.7.0
Autoconf/2.69-foss-2016b
Autoconf/2.69-GCCcore-6.3.0  (D)
Automake/1.15-foss-2016b
Automake/1.15-GCCcore-6.3.0  (D)
Autotools/20150215-foss-2016b
Autotools/20150215-GCCcore-6.3.0  (D)
BBMap/37.67-foss-2016b-Java-1.8.0_144
BCFTools/1.6-foss-2016b
BEDOPS/2.4.38
BETools/2.26.0-foss-2016b
BLAST+/2.6.0-foss-2016b-Python-2.7.14
BLAST/2.2.26-Linux_x86_64
BLAT/3.5-Foss-2016b
BWA/0.7.15-foss-2016b
```
Once you log on, you will have two default modules:

- ogrt-tracking/0.5.0
- StdEnv

Copy a complete name from ml avail and paste here:

- ml load Python/2.7.12-foss-2016b

Ml load will replace modules for you:

- ml load Python/3.5.2-foss-2016b
Work on Sapelo2 Cluster

1. Job Submission Workflow

2. How to Know Job Details

3. How to Know Node Details

4. qlogin Commands: Open Interactive Node for Running Interactive Tasks
Job Submission Workflow  (Refer to training workshop PDF for details)

1. Log on to Login node using MyID and password, and two-factor authentication with Archpass Duo:
   ```
   ssh MyID@sapelo2.gacrc.uga.edu
   ```
2. On Login node, change directory to global scratch: `cd /lustre1/MyID`
3. Create a working subdirectory for a job: `mkdir ./workDir`
4. Change directory to `workDir`: `cd ./workDir`
5. Transfer data from local computer to `workDir`: use `scp` or SSH File Transfer to connect Transfer node
   Transfer data on cluster to `workDir`: log on to Transfer node and then use `cp` or `mv`
6. Make a job submission script in `workDir`: `nano ./sub.sh`
7. Submit a job from `workDir`: `qsub ./sub.sh`
8. Check job status: `qstat` or Cancel a job: `qdel JobID`
How to Know Job Details

Option 1: `qstat -f JobID` for running jobs or finished jobs in 24 hours

Option 2: Email notification from finished jobs (completed, canceled, or crashed), if using:

```
#PBS -M MyID@uga.edu
#PBS -m ae
```
Option 1: `qstat -f JobID` (running jobs or finished jobs in 24 hour)

```bash
$ qstat -f 12222
Job Id: 12222.sapelo2
  Job_Name = testBlast
  Job_Owner = zhuofei@10.56.200.51
resources_used.cput = 00:00:00
resources_used.vmem = 316864kb
resources_used.walltime = 00:15:01
resources_used.mem = 26780kb
resources_used.energy_used = 0
  job_state = C
  queue = batch
  Error_Path = sapelo2-sub2.ecompute:/lustre1/zhuofei/examples/testBlast.e12222
  exec_host = n236/0-3
  Output_Path = sapelo2-sub2.ecompute:/lustre1/zhuofei/examples/testBlast.o12222

Resource_List.nodes = 1:ppn=4:Intel
Resource_List.mem = 20gb
Resource_List.walltime = 02:00:00
Resource_List.nodect = 1

Variable_List = PBS_O_QUEUE=batch,PBS_O_HOME=/home/zhuofei,.....
  PBS_O_WORKDIR=/lustre1/zhuofei/workDir
```
Option 2: Email notification from finished jobs

BS Job Id: 12332.sapelo2     Job Name: bowtie2_test
Queue: batch
Exechost: n232/0

Message: Execution terminated

Details:
Exit_status=0
resources_used.cput=00:09:26
resources_used.vmem=755024kb
resources_used.walltime=00:09:51
resources_used.mem=1468676kb
resources_used.energy_used=0

Short reason:
Execution terminated

PBS Job Id: 12331.sapelo2     Job Name: bowtie2_test
Queue: batch
Exechost: n235/0

Message: Execution terminated

Details:
Exit_status=271
resources_used.cput=00:02:58
resources_used.vmem=755024kb
resources_used.walltime=00:03:24
resources_used.mem=420712kb
resources_used.energy_used=0

Short reason:
Execution terminated

Sender: dispatch_root
How to Know Node Details

**Option 1:** mdiag -v –n | grep [pattern] | ...

- mdiag -v -n | grep batch | grep AMD
- mdiag -v -n | grep batch | grep Intel
- mdiag -v -n | grep highmem_q
- mdiag -v -n | grep grpBuyin_q

**Option 2:** from login node, you can ssh to a compute node and run a command there!

- ssh n72 'lscpu'
- ssh n222 'free -g'
- ssh n237 "ps aux | grep '^MyID'"
qlogin Commands

1. Type qlogin commands from Login node to open Interactive node:
   - `qlogin_intel`: Start an interactive session on an Intel node
   - `qlogin_amd`: Start an interactive session on an AMD node
   - `qlogin`: start an interactive job on either types of nodes

2. Type `exit` command to quit and back to Login node
qlogin Commands

Purpose1: Open interactive node for running interactive tasks of R, Python, Bash scripts, etc.

zhuofei@sapelo2-sub1 ~$ qlogin
qsub: waiting for job 12426.sapelo2 to start
qsub: job 12426.sapelo2 ready

zhuofei@n204 ~$ ml spider R
--------------------------------------------------------------------
R: R/3.4.1-foss-2016b-X11-20160819-GACRC
--------------------------------------------------------------------
...

zhuofei@n204 ~$ ml R/3.4.1-foss-2016b-X11-20160819-GACRC
zhuofei@n204 ~$ R
R version 3.4.1 (2017-06-30) -- "Single Candle"
...
[Previously saved workspace restored]
> a<-1 ; b<-7
> a+b
[1] 8
>
qlogin Commands

Purpose: Open interactive node for compiling/testing source codes of Fortran, C/C++, Python, etc.

```
zhuofei@sapelo2-sub1 ~$ qlogin_intel
qsub: waiting for job 20912.sapelo2 to start
qsub: job 20912.sapelo2 ready

zhuofei@n206 ~$ ml spider iomkl
iomkl:
-----------------------------------------
Description:
Intel Cluster Toolchain Compiler Edition provides Intel C/C++ and Fortran compilers, Intel MKL & OpenMPI.

Versions:
iomkl/2018a
...
zhuofei@n206 ~$ ml iomkl/2018a
zhuofei@n206 ~$ icc mysource.c -o myexec.x
zhuofei@n206 ~$
```

5/18/2018
Appendix: Examples of Batch Serial/Threaded/MPI Job Scripts

• Components you need to run a job:
  ➢ **Software** already installed (cluster software or the one installed by yourself)
  ➢ **Job submission script** to
    1. specify computing resources:
       ✓ number of nodes and cores
       ✓ amount of memory
       ✓ maximum wallclock time
    2. load software using `ml load` (for cluster software)
    3. run any Linux commands you want to run, e.g., `pwd`, `mkdir`, `cd`, `echo`, etc.
    4. run the software
  ➢ **Input data** for analysis, if have

• Common queueing commands you need:
  ➢ `qsub`, `qstat_me`, `qstat`, `qdel`
  ➢ `qstat -f`, `showq`
Example 1: Serial job script running NCBI Blast+ using 1 CPU

```bash
#PBS -S /bin/bash
#PBS -q batch
#PBS -N testBlast
#PBS -l nodes=1:ppn=1
#PBS -l mem=20gb
#PBS -l walltime=48:00:00

cd $PBS_O_WORKDIR

ml load BLAST+/2.6.0-foss-2016b-Python-2.7.14

time blastn [options] ...
```

- **Linux default shell (bash)**
- **Queue name (batch)**
- **Job name (testBlast)**
- **Number of nodes (1), number of cores (1), node feature is NOT needed!**
- **Maximum amount of RAM memory (20 GB) is enforced by the cluster!**
- **Maximum wall-clock time (48 hours) for the job, default 6 minutes**

- **Compute node will use the directory from which the job is submitted as the working directory, i.e., /lustre1/MyID/workDir**
- **Load the module of ncbiblast+, version 2.6.0**
- **Run blastn with ‘time’ command to measure the amount of time it takes to run the application**

[https://wiki.gacrc.uga.edu/wiki/BLAST%2B-Sapelo2](https://wiki.gacrc.uga.edu/wiki/BLAST%2B-Sapelo2)
Example 2: Threaded job script running NCBI Blast+ using 4 CPUS

```bash
#PBS -S /bin/bash
#PBS -q batch
#PBS -N testBlast
#PBS -l nodes=1:ppn=4
#PBS -l mem=20gb
#PBS -l walltime=480:00:00
#PBS -M jsmith@uga.edu
#PBS -m ae
#PBS -j oe

cd $PBS_O_WORKDIR

ml load BLAST+/2.6.0-foss-2016b-Python-2.7.14

time blastn -num_threads 4 [options] ...
```

- Number of nodes (1), number of cores (4)
- Email address to receive a notification for computing resources
- Send email notification when job aborts (a) or terminates (e)
- Standard error file (testBlast.e12345) will be merged into standard out file (testBlast.o12345)
- Run blastn with 4 threads (-num_threads 4)

*Number of cores requested (4) = Number of threads (4)*
Example 3: MPI job script running RAxML using 2 full nodes

```bash
#PBS -S /bin/bash
#PBS -q batch
#PBS -N testRAxML
#PBS -l nodes=2:ppn=28
#PBS -l walltime=120:00:00
#PBS -l mem=100gb

cd $PBS_O_WORKDIR

ml load RAxML/8.2.11-foss-2016b-mpi-avx

mpirun --np 50 raxmlHPC-MPI-AVX [options]
```

- Number of nodes (2), number of cores (28)
  - Total cores requested = 2 × 28 = 56
  - We suggest, Number of MPI Processes (50) ≤ Number of cores requested (56)

- To run raxmlHPC-MPI-AVX, MPI version using OpenMPI
- Run raxmlHPC-MPI-AVX with 50 MPI processes (--np 50), default 56
Guideline Tips

• Do NOT use Login node to run CPU/memory intensive jobs directly ➔ submit jobs to queue!

• Do NOT use Login Node to upload/download large data to/from cluster ➔ use Transfer node!

• Do NOT use home dir for storing large job data ➔ use global scratch /lustre1/MyID

• Do NOT park data on global or local scratch ➔ clean up when job finished or exits from node

• NO large memory job running on batch or jlm_q ➔ use highmem_q

• NO small memory job running on highmem_q ➔ use batch or jlm_q

• As a general rule, threads # = cores # requested
Practical Tips

• Each directory should not have too many files inside! A rule of thumb would be to try to keep no more than a few tens of thousands of files (<10000 would be even better) in any single directory which is accessed frequently.

All files are in ONE single dir! ❌

Files are organized in subdirs! ✓
Practical Tips

- Job name should have a specific computational meaning

  Good Examples:  
  ```
  #PBS -N blastn_dataSet1_trail2 ; #PBS -N M-10-1121
  ```

  Bad Examples:  
  ```
  #PBS -N job1 ; #PBS -N bowtie ; #PBS -N 20160930
  ```

- Monitor job progress from time to time, to catch if a job gets stuck