Submit and Run Jobs Efficiently and Correctly on Sapelo

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

• GACRC
• Sapelo Cluster
• Job Submission Workflow
• Computing Resources Requesting
• Job Submission of Pipeline Tasks
• Guideline and Practical Tips
GACRC

Who Are We?

- Georgia Advanced Computing Resource Center
- Collaboration between the Office of Vice President for Research (OVPR) and the Office of the Vice President for Information Technology (OVPIT)
- Guided by a faculty advisory committee (GACRC-AC)

Why Are We Here?

- To provide computing hardware and network infrastructure in support of high-performance computing (HPC) at UGA

Where Are We?

- http://gacrc.uga.edu/help/ (Web Help)
- https://wiki.gacrc.uga.edu/wiki/Getting_Help (Wiki Help)
Sapelo Cluster

- Cluster Structural Diagram
- Cluster Overview
- Storage Environment
- Computing Resources

SUBMIT AND RUN JOBS EFFICIENTLY AND CORRECTLY ON SAPELO

9/30/2016
The New GACRC Linux HPC Cluster Structural Diagram

Network Connection Speed:
- InfiniBand
- 10 Gbit Ethernet
- 1 Gbit Ethernet

Aggregator Switch

Storage Switch

Top of Rack Switch

Top of Rack Switch

Login Switch

InfiniBand Switch

Icebreaker Storage
Node index: sn3
/project

Icebreaker Storage
Node index: sn[0-2]
/home
/usr/local
/db

Seagate Storage
Lustre Parallel File System
/lustre1

Transfer Node
xfer.gacrcc.uga.edu

Build Node
build1.gacrcc.uga.edu
AMD Opteron 16-core processor, 64 GB RAM

AMD Opteron / Intel Xeon multicore processors
GPU: NVIDIA Tesla (Kepler) K40m GPU cards

User

Login Node (VMs)
sapelol.gacrcc.uga.edu
Intel Xeon processor

World

User

Compute Nodes
Node index: n0, n1, n2, ...... n165, n166, n167

9/30/2016
Cluster Overview

Sapelo 4 Nodes:

Login (ssh username@sapelo1.gacrc.uga.edu) : edit script, submit batch job

Interactive Node: run interactive job, edit script, submit batch job

Transfer (ssh username@xfer.gacrc.uga.edu) : transfer, compress, package data

Build (ssh username@build1.gacrc.uga.edu) : compile, test

9/30/2016

SUBMIT AND RUN JOBS EFFICIENTLY AND CORRECTLY ON SAPELO
## Storage Environment

<table>
<thead>
<tr>
<th>4 Filesystems</th>
<th>Role</th>
<th>Quota</th>
<th>Accessible from</th>
<th>Intended Use</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>/home/username</td>
<td>Home</td>
<td>100GB</td>
<td>sapelo1.gacrc.uga.edu (Login)</td>
<td>Highly static data being used frequently, e.g., scripts, local software</td>
<td>Snapshots</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Interactive nodes (Interactive)</td>
<td>xfer.gacrc.uga.edu (Transfer)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>build1.gacrc.uga.edu (Build)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>compute nodes (Compute)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/lustre1/username</td>
<td>Global</td>
<td>No Limit</td>
<td>Interactive nodes (Interactive)</td>
<td>Temporarily storing large data being used by jobs</td>
<td>Auto-moved to /project if 30 days no modification*</td>
</tr>
<tr>
<td></td>
<td>Scratch</td>
<td></td>
<td>xfer.gacrc.uga.edu (Transfer)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>compute nodes (Compute)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/lscratch/username*</td>
<td>Local</td>
<td>250GB</td>
<td>Individual compute node</td>
<td>Jobs with heavy disk I/O</td>
<td>User to clean up</td>
</tr>
<tr>
<td></td>
<td>Scratch</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>/project/abclab</td>
<td>Storage</td>
<td>1TB (Initial)</td>
<td>xfer.gacrc.uga.edu (Transfer)</td>
<td>Long-term active data storage</td>
<td>Group sharing possible</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: /usr/local/apps : Software installation directory ; /db : Bioinformatics database installation directory
* denotes component or policy to be implemented in the future
Storage Environment - *Data Storing Rule*

### I/O Performance

**Fast**
- `/lustre1/username/` ➡️ **Current Data** being used by jobs running on cluster, i.e., job data
- `/home/username/` ➡️ **Static Data** being used frequently, e.g., scripts and software

**Slow**
- `/project/abclab/` ➡️ **Non-current Active Data** to be analyzed in the future, e.g., 1 month
- User’s Local Storage ➡️ **Non-active Data**, e.g., final outputs and results
Storage Environment - Accessing Rule of 123

/home/username

/lustre1/username

/project/abclab

Login

Interactive

Build

Transfer

Login

qlogin

ssh

ssh

ssh
## Storage Environment

<table>
<thead>
<tr>
<th>7 Main Functions</th>
<th>On/From-Node</th>
<th>Related Filesystem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Login Landing</td>
<td>Login or Transfer or Build</td>
<td>/home/username (Home) (Always!)</td>
</tr>
<tr>
<td>Batch Job Submitting</td>
<td>Login</td>
<td>/home/username (Home)</td>
</tr>
<tr>
<td>Interactive Job Running</td>
<td>Interactive</td>
<td>/lustre1/username (Scratch) (Suggested!) /home/username (Home)</td>
</tr>
<tr>
<td>Data Transferring, Archiving, Compressing</td>
<td>Transfer</td>
<td>/lustre1/username (Scratch) /home/username (Home)</td>
</tr>
<tr>
<td>Long-term Active Data Storing</td>
<td></td>
<td>/project/abclab</td>
</tr>
<tr>
<td>Code Compilation, Test</td>
<td>Build</td>
<td>/home/username (Home)</td>
</tr>
<tr>
<td>Job Data Temporarily Storing</td>
<td>Compute</td>
<td>/lustre1/username (Scratch) /lscratch/username (Local Scratch)*</td>
</tr>
</tbody>
</table>

Note: * denotes component or policy to be implemented in the future

SUBMIT AND RUN JOBS EFFICIENTLY AND CORRECTLY ON SAPELO
## Computing Resources

<table>
<thead>
<tr>
<th>Queue</th>
<th>Node Feature</th>
<th>Total</th>
<th>Processor</th>
<th>Cores /Node</th>
<th>RAM (GB) /Node</th>
<th>Max RAM (GB) /Single-node Job</th>
<th>GPU</th>
<th>GPU Cards /Node</th>
<th>InfiniBand</th>
</tr>
</thead>
<tbody>
<tr>
<td>batch</td>
<td>AMD</td>
<td>112</td>
<td>AMD Opteron</td>
<td>48</td>
<td>128</td>
<td>126</td>
<td>N/A</td>
<td>N/A</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td>256</td>
<td>252</td>
<td>N/A</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HIGHMEM</td>
<td>7</td>
<td>AMD Opteron</td>
<td>48</td>
<td>512 (6)</td>
<td>504</td>
<td>N/A</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1024 (1)</td>
<td>997</td>
<td>N/A</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GPU</td>
<td>2</td>
<td>Intel Xeon</td>
<td>16</td>
<td>128</td>
<td>126</td>
<td>NVIDIA</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>abcnode (buy-in)</td>
<td>2</td>
<td>AMD Opteron</td>
<td>48</td>
<td>256</td>
<td>252</td>
<td>N/A</td>
<td>N/A</td>
<td></td>
</tr>
</tbody>
</table>

Home: /home/username: **100GB**
Global scratch: /lustre1/username: **NO quota limit, auto-moving to /project if no modification in 30 days!**
Job Submission Workflow

Job Working Space is Global Scratch: /lustre1/username/

• Why?
  No quota limit + The I/O fastest filesystem

• How to access?
  From Interactive Node, cd /lustre1/username/

• What need to pay attention?
  Clean up! Non-current Active Data ➔ /project/abclab/
  Non-active Data ➔ local storage

using Transfer Node
Job Submission Workflow

1. Linux/Mac user:
   `ssh username@sapelo1.gacrc.uga.edu`

Windows user:

2. `qlogin`

3. `cd /lustre1/username`

4. `mkdir ./workDir`

5. `cd ./workDir`

6. Linux/Mac user:
   `scp file username@xfer.gacrc.uga.edu:/lustre1/username/workDir`

Windows user:

Note: `-r` option of `scp` command will recursively copy a directory

7. `nano ./sub.sh`

   `#PBS -S /bin/bash`
   `#PBS -q batch`
   `#PBS -N testBlast`
   `#PBS -l nodes=1:ppn=1:AMD`
   `#PBS -l mem=20g`
   `#PBS -l walltime=48:00:00`
   `cd $PBS_O_WORKDIR`
   `module load ncbiblast+/2.2.29`
   `blastn [options] > outputfile`

8. `$ qsub sub.sh`

Useful PBS headers:
   `#PBS mem=200gb : use 256GB AMD nodes`
   `ppn=24 : request 24 cores for 24 threads`
Job Submission Workflow

1. Log on to Sapelo Login node: `ssh userID@sapelol.gacrc.uga.edu`
2. From Sapelo Login node, transfer to Interactive node by issuing the command: `qlogin`
3. From Interactive node: Change directory to global scratch: `cd /lustre1/userID`
4. Create a working subdirectory on global scratch: `mkdir ./workDir`
5. Change directory to workDir: `cd ./workDir`
6. Transfer data to workDir using `scp` or SSH Client File Transfer (with `tar` or `gzip`)
7. Make a Sapelo job submission script: `nano ./sub.sh`
8. Submit job: `qsub ./sub.sh`

Useful PBS headers:
- `#PBS -l mem=200gb` : use 256GB high-RAM AMD nodes
- `#PBS -l nodes=1:ppn=24:AMD` : request 24 cores for 24 threads, max 48!
Computing Resources Requesting

• What are computing resources for a job?

1. Node Feature : AMD, HIGHMEM, GPU
2. Node Number : Serial/Threaded Job ➔ 1 node ; MPI Job ➔ N nodes (N ≥ 1)
3. Core Number :
   Serial Job ➔ 1 core
   Threaded Job ➔ Threads number t = Cores number n (1≤n≤48)
   MPI Job ➔ Process number p = Cores number n (1≤n≤48*N)
   \[ p < n \quad (\text{-np\ p option to mpirun/mpiexec is needed!}) \]

Note: if \( n/N=\text{ppn} < 48 \), ppn should be a multiplier of 3*

* [https://wiki.gacrc.uga.edu/wiki/Running_Jobs_on_Sapelo#Running_MPI_jobs_with_fewer_than_48_cores_per_node](https://wiki.gacrc.uga.edu/wiki/Running_Jobs_on_Sapelo#Running_MPI_jobs_with_fewer_than_48_cores_per_node)
Computing Resources Requesting

4. Maximum Memory Requestable for Serial/Threaded Job on Compute Node:

- Low Memory AMD node ➔ 126GB
- High Memory AMD node ➔ 252GB
- 512GB HIGHMEM node ➔ 500GB
- 1TB HIGHMEM node ➔ 997GB

- Request 127GB to use High Memory AMD node
  - AMD Mem Scale: 1 ➔ 126|127 ➔ 252
- Request 505GB to use 1TB HIGHMEM node
  - HIGHMEM Mem Scale: 1 ➔ 500|505 ➔ 997

5. Wall-Clock Time: Total time you request for a job running from starting to finishing

Queue waiting time is NOT included!
Default 6 minutes!
Computing Resources Requesting

• How to request resources for a job?

1. Request in your Sapelo job submission script

2. Include PBS header lines in script, e.g.:

   #PBS -l queue=batch ➔ Use Sapelo batch queue
   #PBS -l nodes=1:ppn=24:AMD ➔ Request 24 cores from 1 AMD node
   #PBS -l mem=20gb ➔ Request 20GB memory
   #PBS -l walltime=60:00:00 ➔ Request 60 hours wall-clock time
Computing Resources Requesting

• How to use PBS header lines to request resources correctly?

Example 1:

#PBS -l nodes=1:ppn=1:AMD #PBS -l nodes=1:ppn=48:AMD #PBS -l nodes=1:ppn=96:AMD  
#PBS -l mem=10gb  ✓ #PBS -l mem=10gb  ✓ #PBS -l mem=10gb  ✗  
#PBS -l walltime=12:00:00  #PBS -l walltime=12:00:00  #PBS -l walltime=12:00:00

#PBS -l nodes=1:ppn=48:AMD  #PBS -l nodes=1:ppn=48:AMD  #PBS -l nodes=1:ppn=48:AMD  
#PBS -l mem=126gb  ✓ #PBS -l mem=127gb  ✓ #PBS -l mem=256gb  ✗  
#PBS -l walltime=6:00:00  #PBS -l walltime=12:00:00  #PBS -l walltime=12:00:00  

AMD Mem Scale: 1 — 126|127 — 252
Computing Resources Requesting

Example2:

```
#PBS -l nodes=1:ppn=48:HIGHMEM
#PBS -l mem=100gb
#PBS -l walltime=480:00:00

#PBS -l nodes=1:ppn=96:HIGHMEM
#PBS -l mem=100gb
#PBS -l walltime=480:00:00

#PBS -l nodes=1:ppn=48:HIGHMEM
#PBS -l mem=500gb
#PBS -l walltime=480:00:00

#PBS -l nodes=1:ppn=48:HIGHMEM
#PBS -l mem=505gb
#PBS -l walltime=480:00:00

#PBS -l nodes=1:ppn=48:HIGHMEM
#PBS -l mem=800gb
#PBS -l walltime=480:00:00

#PBS -l nodes=1:ppn=48:HIGHMEM
#PBS -l mem=1000gb
#PBS -l walltime=480:00:00
```

HIGHMEM Mem Scale: 1 → 500|505 → 997
Computing Resources Requesting

Example 3:

```bash
#PBS -l nodes=n26:ppn=48:HIGHMEM
#PBS -l mem=800gb
#PBS -l walltime=480:00:00
```

Example 4:

```bash
#PBS -l nodes=1:ppn=1:AMD
#PBS -l mem=5gb
#PBS -l walltime=12:00:00
```

E.g., 12 hours is reasonable wall-clock time for this small job
Computing Resources Requesting

• Why do my jobs have to wait so long in the queue?

If you request computing resources incorrectly, your jobs will pend forever!

I. You requested invalid resources, e.g., ppn=96, mem=256gb (AMD) or 1024gb (HIGHMEM), no node feature provided

II. You requested too large resources which are not actually needed by a job, e.g., nodes=1, ppn=48 for a serial job, nodes=3, ppn=24 for a threaded job, walltime=720:00:00 for a short job actually needs 24 hours

III. You requested resources which are occupied by other jobs running on cluster

Note: Jobs that cannot run across multiple nodes will not run faster if more than one node requested
Computing Resources Requesting

• How to figure out startup resources for a job?

  Option1: Refer to sample scripts on GACRC Wiki Software page:

  https://wiki.gacrc.uga.edu/wiki/Software

  Option2: Do computing resource scaling-up by running testing job (suggested!)
Computing Resources Requesting

• How to do computing resource scaling-up by running testing job?

1. Reduce your computational scale to a smaller testing one:
   - E.g. 1: Actual 20 threads ➔ Testing 4 threads: scaling factor $f \sim 5$
   - E.g. 2: Actual 10GB input data ➔ Testing 1GB input data: scaling factor $f \sim 10$

2. Make testing job submission script requesting smaller computing resources, and submit it

3. After job finished successfully, `qstat -j JobID` gives `resources_used.vmem` and `resources_used.walltime` numbers, e.g., 6.2 GB and 7 hours

4. Do computing resources scaling-up for actual job using $f$ and the above two numbers:
   - E.g. 1: $f \sim 5$ ➔ Actual ~30 GB and ~1.5 hours
   - E.g. 2: $f \sim 10$ ➔ Actual ~60GB and ~35 hours

Note: Scaling may not be linear
Computing Resources Requesting

• How to get resource information for a job?

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

Option 2: Email notification from finished jobs, if you include two PBS header lines in your job submission script:

```bash
#PBS -M username@uga.edu
#PBS -m ae
```

Note: You need both of two lines in order to receive an email notification
qstat -f JobID for running jobs or finished jobs in 1 hour
Email notification from finished jobs

PBS Job Id: 700009.pbs.scm
Job Name: testJob
Exec host: n1/4-27
Execution terminated
Exit_status=0

resources_used.cput=00:05:12
resources_used.energy_used=0
resources_used.mem=410984kb
resources_used.vmem=6548516kb
resources_used.walltime=04:00:59

Error_Path: uga-2f0f976.scm:/home/zhuofei/MPIs/testJob.o700009
Output_Path: uga-2f0f976.scm:/home/zhuofei/MPIs/testJob.o700009

PBS Job Id: 700097.pbs.scm
Job Name: testJob
Exec host: n1/4-27
Execution terminated
Exit_status=271

resources_used.cput=00:11:22
resources_used.energy_used=0
resources_used.mem=412304kb
resources_used.vmem=6548524kb
resources_used.walltime=05:00:41

Error_Path: uga-2f0f976.scm:/home/zhuofei/MPIs/testJob.o700097
Output_Path: uga-2f0f976.scm:/home/zhuofei/MPIs/testJob.o700097
Job Submission of Pipeline Tasks

• If I have pipeline tasks using multiple applications, how to run them?

  Option 1: Run tasks one-by-one in separate jobs

  Option 2: Run all tasks in one job
Option 1: Run applications/tasks one-by-one in separate jobs

Request computing resources for task1
cd $PBS_O_WORKDIR
module load application1
Run application1 on inputData

Request computing resources for task2
cd $PBS_O_WORKDIR
module load application2
Run application2 on output1

Request computing resources for task3
cd $PBS_O_WORKDIR
module load application3
Run application3 on output2

→ qsub

→ task1 running in job1

→ output1

→ qsub

→ task2 running in job2

→ output2

→ qsub

→ task3 running in job3

→ output3
Option 2: Run all tasks in one job

Request computing resources up to maximum computational needs of tasks

```bash
cd $PBS_O_WORKDIR
module load application1
module load application2
module load application3
Run application1 on inputData
Run application2 on output1
Run application3 on output2
```

tasks running in job

```bash
qsub
```

output3
Option 1: Run applications/tasks one-by-one in separate jobs

• Advantages:
  i. Easier to control workflow
  ii. Job-dependent computing resources requesting
  iii. Fewer modules to be loaded for each job

• Disadvantages:
  i. Longer queue waiting time due to multiple job submissions
  ii. Less automation; more scripts and manual work
Option 2: Run all tasks in one job

• Advantages:
  i. More automation; less scripts and manual work
  ii. Shorter queue waiting time due to only one job submission

• Disadvantages:
  i. Harder to control workflow
  ii. Computing resources requesting up to maximum computational needs of tasks
  iii. Longer queue waiting time due to ii
  iv. More modules to be loaded
Guideline Tips

• Do NOT use Login Node to run jobs ➔ Interactive Node or the queue

• Do NOT use Login Node upload or download data to/from cluster

• Do NOT use Login Node to transfer data to the home dir

• NO large memory job running on AMD nodes ➔ HIGHMEM nodes

• NO small memory job running on HIGHMEM nodes ➔ AMD nodes

• 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`

• Redirect standard output and error of the application to a file, instead of letting it be written in the stdout .o file and stderr .e file of the job, e.g.:
  
  ```
  time application >file 2>&1
  ```

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

• Batch job deletion to cancel all your running and waiting jobs from queue
  
  ```
  qdel $(qstat | egrep '^[0-9]' | awk '$5!~/C/ {print $1}')</code>
Thank You!