Using GACRC HPC Computing Facility
Sapelo2 Cluster

Georgia Advanced Computing Resource Center (GACRC)
Enterprise Information Technology Services (EITS)
The University of Georgia
Outline

• GACRC
• Sapelo2 Cluster and Workflow
• Request Computing Resources
• Run MPI Job
• Run GPU Job
• Obtain Job Details
• Guideline and Practical Tips
GACRC

- A high-performance-computing (HPC) center at the UGA
- Provide to the UGA research and education community an advanced computing environment:
  - HPC computing and networking infrastructure located at the Boyd Data Center
  - Comprehensive collection of scientific, engineering and business applications
  - Consulting and training services

Wiki: http://wiki.gacrc.uga.edu
Support: https://uga.teamdynamix.com/TDClient/Requests/ServiceCatalog?CategoryID=11593
Web Site: http://gacrc.uga.edu
Please Note: You need to connect to the **UGA network using VPN** when accessing from outside of the **UGA main campus**.

UGA VPN: [https://eits.uga.edu/access_and_security/infosec/tools/vpn/](https://eits.uga.edu/access_and_security/infosec/tools/vpn/)

---

**Using GACRC HPC Computing Facility - SaPeLo2 Cluster**

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

### Computational Queues

- **batch**
  - For regular jobs
- **highmem_q**
  - For high memory jobs
- **gpu_q**
  - For GPU-enabled jobs

### Nodes and Queues

- **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

---

**Login Node**

- **(sapel02.gacrc.uga.edu)**

**Home**

- **/home/MvID**
  - 200 Gb
  - No quota limit

**Scratch**

- **/scratch/MvID**
  - No quota limit

**Work**

- **/work/abclab**
  - 500Gb/10^4 files

**Transfer Node**

- **(xfer.gacrc.uga.edu)**

**Project**

- **/project/abclab**
  - 1 TB
3 nodes and 4 Directories

- `/home/MyID`
- `/scratch/MyID`
- `/work/abclab`
- `/project/abclab`
- non-GACRC storage

Log on using `ssh`

- `/home/MyID`
- `/scratch/MyID`
- `/work/abclab`

Interactive

Transfer

Duo Authentication

Login

User

exit
<table>
<thead>
<tr>
<th>Queue</th>
<th>Total Nodes</th>
<th>RAM(GB)/Node</th>
<th>Max Mem(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>42</td>
<td>192</td>
<td>184</td>
<td>32</td>
<td>Intel Xeon Skylake</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>64</td>
<td>58</td>
<td>28</td>
<td>Intel Xeon Broadwell</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>106</td>
<td>128</td>
<td>120</td>
<td>48</td>
<td>AMD Opteron</td>
<td></td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>53</td>
<td>128</td>
<td>32</td>
<td>AMD EPYC</td>
<td></td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>highmem_q</td>
<td>9</td>
<td>1024</td>
<td>990</td>
<td>28</td>
<td>48</td>
<td>64</td>
<td>Intel Xeon Broadwell (4)</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>512</td>
<td>502</td>
<td>32</td>
<td>32</td>
<td>48</td>
<td>Intel Xeon Nehalem (1)</td>
</tr>
<tr>
<td>gpu_q</td>
<td>4</td>
<td>192</td>
<td>184</td>
<td>32</td>
<td>Intel Xeon Skylake</td>
<td>1 NVIDIA P100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>128</td>
<td>120</td>
<td>16</td>
<td>Intel Xeon</td>
<td>8 NVIDIA K40m</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>96</td>
<td>90</td>
<td>12</td>
<td>Intel Xeon</td>
<td>7 NVIDIA K20Xm</td>
<td></td>
</tr>
<tr>
<td>groupBuyin_q</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>variable</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2/3/20 USING GACRC HPC COMPUTING FACILITY - SAPELO2 CLUSTER
"30-day purge" Policy on Scratch

https://wiki.gacrc.uga.edu/wiki/Disk_Storage#Scratch_file_system

Any file that is not accessed or modified in over 30 days will be automatically deleted off the /scratch file system. Measures circumventing this policy will be monitored and actively discouraged.

- You have a list of those purgeable files located at /usr/local/var/lustre_stats/$USER.over30d.files.lst
- You are suggested to copy files from /scratch to /project or outside of GACRC
- You should first move all unnecessary files and folders to /scratch/trash/$USER
- The fastest way to save your old files is to copy them to /project area, using the fpsync utility on xfer.gacrc.uga.edu
- If you want to first create a tar archive of your /scratch area, DO NOT compress the archive when creating the archive
Job Submission Workflow

1. Linux/Mac user:
   \texttt{ssh MyID@sapelo2.gacrc.uga.edu}

2. \texttt{cd /scratch/MyID}

3. \texttt{mkdir ./workDir}

4. \texttt{cd ./workDir}

5. Linux/Mac user:
   \texttt{scp file MyID@xfer.gacrc.uga.edu:/scratch/MyID/workDir}

6. \texttt{nano ./sub.sh}

7. \texttt{$ qsub sub.sh}

8. \texttt{$ qstat_me or qdel JobID}

Windows user:

Login

lustre1

USING GACRC HPC COMPUTING FACILITY - SAPELO2 CLUSTER

2/3/20
Computing Resources Requesting

• **What are computing resources for a job?**
  1. **Queue**: batch, highmem_q, gpu_q
  2. **Node Feature**: AMD (Opteron or EPYC), Intel (Broadwell or Skylake), EPYC, Skylake, K40, P100, etc.
  3. **Node Number**: Serial/Threaded Job ➔ 1 node; MPI Job ➔ N nodes (N ≥ 1)
  4. **Core Number**:
     - Serial Job ➔ 1 core
     - Threaded Job ➔ Threads number \( t = \text{Core number } n \) (1≤\(n\)≤48 for batch)
     - MPI Job ➔ Process number \( p < \text{Core number } n \) (\(-np \ p\) option to mpirun/mpiexec is needed!)
     - \( p = n \) (1≤\(n\)≤48\(N\) for batch, \(-nq \ p\) option is not needed)
Computing Resources Requesting

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

- **batch**: 1GB → 120|121GB → 184GB
  - Max 48 cores
  - Max 32 cores

- **highmem_q**: 1GB → 502|503GB → 990GB
  - Max 48 cores
  - Max 64 cores

5. Wall-Clock Time: Time requested for a job running from starting to finishing:
   - Job pending time is NOT included
   - Default 1 hour, if not requested specifically
Computing Resources Requesting

- **How to request computing resources for a job?**
  1. Request in your Sapelo2 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 ➔ Request 24 cores from 1 node (AMD or Intel)
    #PBS -l mem=20gb ➔ Request 20GB memory
    #PBS -l walltime=60:00:00 ➔ Request 60 hours wall-clock time
    ```
Computing Resources Requesting

• How to know node details on queues?

```bash
mdiag -v -n | head -n 2; mdiag -v -n | grep batch
mdiag -v -n | head -n 2; mdiag -v -n | grep batch | grep Intel
mdiag -v -n | head -n 2; mdiag -v -n | grep batch | grep AMD

mdiag -v -n | head -n 2; mdiag -v -n | grep highmem
mdiag -v -n | head -n 2; mdiag -v -n | grep highmem | grep Intel
mdiag -v -n | head -n 2; mdiag -v -n | grep highmem | grep AMD
```
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 or mem=192gb from a batch queue node

  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=480:00:00 for a short job actually needs 48 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 estimate the computing resources needed, before you run the 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 2GB input data: scaling factor \( f \sim 5 \)

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., 4GB and 5 hours

4. Do computing resources scaling-up for actual job using \( f \) and the above two numbers:
   - E.g. 1: \( f \sim 5 \) ➔ Actual ~20 GB and ~1 hours
   - E.g. 2: \( f \sim 5 \) ➔ Actual ~20GB and ~25 hours

Note: Scaling may not be linear
Run MPI Job

Sapelo2 nodes communicate via InfiniteBand (IB): QDR and **EDR**

```
mdiag -v -n | grep batch | grep QDR
mdiag -v -n | grep batch | grep EDR
```
Run MPI Job – PBS headers for running MPI job

Scenario 1: Request multiple nodes; use ALL cores from each node

- Use batch queue
- Request 64 cores from 2 AMD EPYC nodes (EDR)
- Request 2GB memory for each MPI process, so 64GB/node
- Request 60 hours wall-clock time

- Use batch queue
- Request 128 cores from 4 Intel Skylake nodes (EDR)
- Request 4GB memory for each MPI process, so 128GB/node
- Request 60 hours wall-clock time
Run MPI Job – PBS headers for running MPI job

Scenario 2: Request multiple nodes; does NOT use all cores from each node

```
#PBS -l queue=batch ➜ Use batch queue
#PBS -l nodes=5:ppn=12:EDR ➜ Request 60 cores from 5 EDR nodes (AMD or Intel)
#PBS -l pmem=2gb ➜ Request 2GB memory for each MPI process, so 24GB/node ✔️
#PBS -l walltime=60:00:00 ➜ Request 60 hours wall-clock time

#PBS -l queue=batch ➜ Use batch queue
#PBS -l nodes=5:ppn=12:AMD:EDR ➜ Request 60 cores from 5 AMD EDR nodes (AMD EPYC)
#PBS -l pmem=5gb ➜ Request 5GB memory for each MPI process, so 60GB/node ✔️
#PBS -l walltime=60:00:00 ➜ Request 60 hours wall-clock time
```
Run MPI Job – MPI libraries installed on Sapelo2

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

When run MPI software using MVAPICH2, you will need to differentiate EDR from QDR;
But it is not necessary for OpenMPI

module spider MVAPICH2

<table>
<thead>
<tr>
<th>QDR Versions:</th>
<th>EDR Versions:</th>
</tr>
</thead>
<tbody>
<tr>
<td>MVAPICH2/2.2-GCC-5.4.0-2.26</td>
<td>MVAPICH2/2.3-GCC-5.4.0-2.26-EDR</td>
</tr>
<tr>
<td>MVAPICH2/2.2-GCC-6.4.0-2.28</td>
<td>MVAPICH2/2.3-GCC-6.4.0-2.28-EDR</td>
</tr>
<tr>
<td>MVAPICH2/2.2-icccifort-2013_sp1.0.080</td>
<td>MVAPICH2/2.3-icccifort-2013_sp1.0.080-EDR</td>
</tr>
<tr>
<td>MVAPICH2/2.2-icccifort-2015.2.164-GCC-4.8.5</td>
<td>MVAPICH2/2.3-icccifort-2015.2.164-GCC-4.8.5-EDR</td>
</tr>
<tr>
<td>MVAPICH2/2.2-icccifort-2018.1.163-GCC-6.4.0-2.28</td>
<td>MVAPICH2/2.3-icccifort-2018.1.163-GCC-6.4.0-2.28-EDR</td>
</tr>
</tbody>
</table>
Run MPI Job – Memory request for MPI job

Memory request for MPI job: using pmem, instead of mem, in your job script

For example:

#PBS -l pmem=2gb

pmem: maximum memory for each MPI process
Run MPI Job – sample scripts running MPI jobs

```
#PBS -S /bin/bash
#PBS -q batch
#PBS -N testmpi4py
#PBS -l nodes=1:ppn=12:EDR
#PBS -l pmem=1gb
#PBS -l walltime=4:00:00
cd $PBS_O_WORKDIR
ml Python/2.7.14-foss-2016b
mpirun python ./testmpi4py.py
```

```
#PBS -S /bin/bash
#PBS -q batch
#PBS -N testraxml
#PBS -l nodes=2:ppn=48:EDR
#PBS -l walltime=48:00:00
#PBS -l pmem=2g
#PBS -j oe
#PBS -M MyID@uga.edu
#PBS -m ae
cd $PBS_O_WORKDIR
ml RAxML/8.2.11-foss-2016b-mpi-avx
mpirun raxmlHPC-MPI-AVX [options]
```
Run GPU Job
https://wiki.gacrc.uga.edu/wiki/Running_Jobs_on_Sapelo2#GPU.FCUDA

• How to know node details on gpu_q?

  mdiag -v -n | head -n 2 ; mdiag -v -n | grep gpu_q
Run GPU Job – PBS headers for running GPU job

https://wiki.gacrc.uga.edu/wiki/Running_Jobs_on_Sapelo2#GPU.FCUDA

Scenario 1: Request one or more CPU cores with one or more GPU devices

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#PBS -l queue=gpu_q</td>
<td>Use gpu_q queue</td>
</tr>
<tr>
<td>#PBS -l nodes=1:ppn=1:gpu=1</td>
<td>Request 1 CPU core with 1 GPU card from 1 node (EDR or QDR)</td>
</tr>
<tr>
<td>#PBS -l mem=10gb</td>
<td>Request 10GB total memory</td>
</tr>
<tr>
<td>#PBS -l walltime=60:00:00</td>
<td>Request 60 hours wall-clock time</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#PBS -l queue=gpu_q</td>
<td>Use gpu_q queue</td>
</tr>
<tr>
<td>#PBS -l nodes=1:ppn=4:gpu=2</td>
<td>Request 4 CPU cores with 2 GPU cards from 1 node (EDR or QDR)</td>
</tr>
<tr>
<td>#PBS -l mem=20gb</td>
<td>Request 20GB total memory</td>
</tr>
<tr>
<td>#PBS -l walltime=60:00:00</td>
<td>Request 60 hours wall-clock time</td>
</tr>
</tbody>
</table>
Run GPU Job – PBS headers for running GPU job

**Scenario 2: Request one or more GPU devices with specific device selection**

<table>
<thead>
<tr>
<th>PBS command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#PBS -l queue=gpu_q</td>
<td>Use gpu_q queue</td>
</tr>
<tr>
<td>#PBS -l nodes=1:ppn=4:gpu=1:K20</td>
<td>Request 4 CPU cores with 1 K20 GPU card from 1 node (QDR)</td>
</tr>
<tr>
<td>#PBS -l mem=20gb</td>
<td>Request 20GB total memory</td>
</tr>
<tr>
<td>#PBS -l walltime=60:00:00</td>
<td>Request 60 hours wall-clock time</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PBS command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#PBS -l queue=gpu_q</td>
<td>Use gpu_q queue</td>
</tr>
<tr>
<td>#PBS -l nodes=1:ppn=4:gpu=1:P100</td>
<td>Request 4 CPU cores with 1 P100 GPU card from 1 node (EDR)</td>
</tr>
<tr>
<td>#PBS -l mem=20gb</td>
<td>Request 20GB total memory</td>
</tr>
<tr>
<td>#PBS -l walltime=60:00:00</td>
<td>Request 60 hours wall-clock time</td>
</tr>
</tbody>
</table>
Run GPU Job – PBS headers for running GPU job

Scenario 3: Run more than one process at a time on the GPU node by selecting “Default” compute mode

<table>
<thead>
<tr>
<th>PBS Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>#PBS -l queue=gpu_q</code></td>
<td>Use <code>gpu_q</code> queue</td>
</tr>
<tr>
<td><code>#PBS -l nodes=1:ppn=4:gpu=1:default</code></td>
<td>Request 4 CPU cores with 1 GPU card from 1 node (QDR)</td>
</tr>
<tr>
<td><code>#PBS -l mem=20gb</code></td>
<td>Request 20GB total memory</td>
</tr>
<tr>
<td><code>#PBS -l walltime=60:00:00</code></td>
<td>Request 60 hours wall-clock time</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PBS Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>#PBS -l queue=gpu_q</code></td>
<td>Use <code>gpu_q</code> queue</td>
</tr>
<tr>
<td><code>#PBS -l nodes=1:ppn=4:gpu=1:P100:default</code></td>
<td>Request 4 CPU cores with 1 P100 GPU card from 1 node (EDR)</td>
</tr>
<tr>
<td><code>#PBS -l mem=20gb</code></td>
<td>Request 20GB total memory</td>
</tr>
<tr>
<td><code>#PBS -l walltime=60:00:00</code></td>
<td>Request 60 hours wall-clock time</td>
</tr>
</tbody>
</table>
Run GPU Job – Useful environment variable for GPU job
https://wiki.gacrc.uga.edu/wiki/Running_Jobs_on_Sapelo2#GPU.FCUDA

1. PBS_GPUFILE: The name of a file storing GPU devices allocated to your GPU job:
   echo $PBS_GPUFILE

2. To get a list of the numbers of the GPU devices allocated to your job, separated by a blank space, use the command:
   CUDADEV=$(cat $PBS_GPUFILE | rev | cut -d"u" -f1)
   echo "List of devices allocated to this job:"
   echo $CUDADEV
Run GPU Job – GPU utilization of a GPU job

E.g. a job is running on gn164 (a K20 node), from login node:

```bash
ssh gn164 nvidia-smi
```

**Thu Jan 9 21:35:33 2020**

```
+------------------------------------------------------------------+
| NVIDIA-SMI 410.48 Driver Version: 410.48 |
|------------------------------------------------------------------+
| GPU Name Persistence-M | Bus-Id Disp.A | Volatile Uncorr. ECC |
| Fan Temp Perf Pwr:Usage/Cap | Memory-Usage | GPU-Util Compute M. |
+========================+==================+=================
| 0 Tesla K20Xm On | 00000000:0C:00.0 Off | 0 |
| N/A 38C P0 70W / 235W | 883MiB / 5700MiB | 32% Default |
+-------------------------------+
| 1 Tesla K20Xm On | 00000000:0D:00.0 Off | 0 |
| N/A 43C P0 90W / 235W | 3211MiB / 5700MiB | 65% Default |
+-------------------------------+
| 2 Tesla K20Xm On | 00000000:11:00.0 Off | 0 |
| N/A 46C P0 81W / 235W | 3211MiB / 5700MiB | 64% Default |
+-------------------------------+
```

......
Obtain Job Details

https://wiki.gacrc.uga.edu/wiki/Monitoring_Jobs_on_Sapelo2

Option 1:

qstat -nl -u MyID  for running/pending/completed jobs on cluster
qstat -nlr -u MyID  for running jobs on cluster
qstat -nlrt -u MyID  for running jobs on cluster, showing each array job element

Option 2:

qstat -f JobID  for detail info of a running jobs or finished job within 24 hours

Option 3:

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

#PBS -M MyID@uga.edu
#PBS -m ae
Guideline Tips

• Do NOT use Login Node to run jobs ➔ Interactive Node OR submit job to queue

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

• Do NOT use Login Node to transfer data on cluster ➔ Transfer Node

• NO large memory job running on batch queue ➔ highmem_q

• NO small memory job running on highmem queue ➔ batch

• As a general rule, thread number for running software = core number 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.
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 log file, instead of letting it be written in the stdout .o file and stderr .e file of the job, e.g.:

  time application >log 2>&1

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

**Telephone Support**

EITS Help Desk: 706-542-3106

Monday – Thursday: 7:30 a.m. – 7:30 p.m.

Friday: 7:30 a.m. – 6 p.m.

Saturday – Sunday: 1 p.m. – 7 p.m.

**Georgia Advanced Computing Resource Center**

101-108 Computing Services building

*University of Georgia*

*Athens, GA 30602*

[https://gacrc.uga.edu/](https://gacrc.uga.edu/)