Using GACRC HPC Computing Facility
Sapelo2 Cluster

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

• GACRC

• Review: Sapelo2 Cluster and Workflow

• Load Software Modules

• Request Computing Resources

• Run MPI and GPU Job

• Obtain Job Details

• Wiki and Support
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

Sapelo2 System: https://wiki.gacrc.uga.edu/wiki/Systems

Sapelo2 Disk Storage: https://wiki.gacrc.uga.edu/wiki/Disk_Storage
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/
3 Nodes and 4 Directories

Login
- qlogin
- qlogin_amd
- qlogin_intel

Interactive
- /home/MyID
- /scratch/MyID
- /work/abclab

Transfer
- /project/abclab
- non-GACRC storage

User
- log on using ssh

Duo Authentication
- log on using ssh

Duo Authentication
- non-GACRC storage
<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>35</td>
<td>192</td>
<td>184</td>
<td>32</td>
<td>Intel Xeon Skylake</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>31</td>
<td>64</td>
<td>58</td>
<td>28</td>
<td>Intel Xeon Broadwell</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>42</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>63</td>
<td>128</td>
<td>32</td>
<td>64</td>
<td>AMD EPYC (41</td>
<td>22)</td>
<td></td>
</tr>
<tr>
<td>highmem_q</td>
<td>7</td>
<td>1024</td>
<td>990</td>
<td>28</td>
<td>64</td>
<td>Intel Xeon Broadwell (3) AMD EPYC (4)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>17</td>
<td>512</td>
<td>502</td>
<td>32</td>
<td>48</td>
<td>AMD EPYC (15) AMD Opteron (2)</td>
<td></td>
</tr>
<tr>
<td>gpu_q</td>
<td>4</td>
<td>192</td>
<td>184</td>
<td>32</td>
<td>Intel Xeon Skylake (EDR)</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 (QDR)</td>
<td>8 NVIDIA K40m</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>96</td>
<td>90</td>
<td>12</td>
<td>Intel Xeon (QDR)</td>
<td>7 NVIDIA K20Xm</td>
<td></td>
</tr>
<tr>
<td>groupBuyin_q</td>
<td>variable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
"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:**
   
   ```bash
   ssh MyID@sapelo2.gacrc.uga.edu
   ```

2. **Login**

3. **mkdir ./workDir**

4. **cd ./workDir**

5. **Linux/Mac user:**
   
   ```bash
   scp file MyID@xfer.gacrc.uga.edu:/scratch/MyID/workDir
   ```

6. **nano ./sub.sh**

7. **$ qsub sub.sh**

8. **$ qstat me or qdel JobID**

- **PBS:**
  - `-S /bin/bash`
  - `-q batch`
  - `-N bowtie2_test`
  - `-l nodes=1:ppn=1`
  - `-l mem=2gb`
  - `-l walltime=1:00:00`
  - `-M yourMyID@uga.edu`
  - `-m ae`

- **cd $PBS_O_WORKDIR**
- **module load Bowtie2/2.3.3-foss-2016b**
- **USING GACRC HPC COMPUTING FACILITY - SAPelo2 CLUSTER**
- 6/9/2020
Load Software Modules

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

• To search for a specific software module installed on Sapelo2:

  module spider AppName

  e.g., module spider Python or python (AppName is case insensitive)

• To load a software module for use:

  module load AppName/version-toolchain

  e.g., module load Python/3.6.4-foss-2018a (AppName is case sensitive)

• What does “module load” do for you? ➔ Set up module working environment!
zhuofei@n204 ~$ which python
/bin/python
zhuofei@n204 ~$ python -V
Python 2.7.5
zhuofei@n204 ~$ echo $PYTHONPATH

zhuofei@n204 ~$ echo $LD_LIBRARY_PATH

zhuofei@n204 ~$ module load Python/3.7.4-GCCcore-8.3.0
zhuofei@n204 ~$

Python 3.7.4
zhuofei@n204 ~$ echo $PYTHONPATH
/usr/local/apps/eb/Python/3.7.4-GCCcore-8.3.0/easybuild/python
zhuofei@n204 ~$ python -V
Python 3.7.4
zhuofei@n204 ~$ echo $LD_LIBRARY_PATH
/usr/local/apps/eb/Python/3.7.4-GCCcore-8.3.0/lib:/usr/local/apps/eb/libffi/3.2.1-GCCcore-8.3.0/lib64:......

before Python/3.7.4 is loaded

Python/3.7.4 is loaded

after Python/3.7.4 is loaded
Load Software Modules
https://wiki.gacrc.uga.edu/wiki/Lmod

• If you already loaded one or more modules, make sure that the module you are loading currently has no toolchain conflicts with the module(s) already loaded!

  Toolchain MUST be the same or compatible!

Otherwise, you could receive Lmod error or other dependency errors:

Lmod has detected the following error: These module(s) exist but cannot be loaded as requested
zhuofei@n204 ~$ module load DAS_Tool/1.1.1-foss-2018a-R-3.6.2-Python-2.7.14
zhuofei@n204 ~$ module load Biopython/1.68-foss-2016b-Python-2.7.14
Lmod has detected the following error: These module(s) exist but cannot be loaded as requested: "bzip2/1.0.6-foss-2016b"
Try: "module spider bzip2/1.0.6-foss-2016b" to see how to load the module(s).

zhuofei@n204 ~$ module load Biopython/1.73-foss-2018a-Python-2.7.14
zhuofei@n204 ~$ module load BEDTools/2.26.0-GCCcore-6.4.0
 # GCCcore-6.4.0 toolchain is compatible with foss-2018a
 # Please see next page for common compatible toolchains
zhuofei@n204 ~$ module load RepeatExplorer2/20181109-foss-2018a-Python-3.6.4

The following have been reloaded with a version change:

1) BLAST+/2.7.1-foss-2018a => BLAST+/2.7.1-foss-2018a-Python-3.6.4
2) DIAMOND/0.9.24-foss-2018a => DIAMOND/0.9.22-foss-2018a
3) Python/2.7.14-foss-2018a => Python/3.6.4-foss-2018a
4) R/3.6.2-foss-2018a-X11-20180131-GACRC => R/3.4.4-foss-2018a-X11-20180131-GACRC

zhuofei@n204 ~$ ml purge
zhuofei@n204 ~$ module load DAS_Tool/1.1.1-foss-2018a-R-3.6.2-Python-2.7.14
zhuofei@n204 ~$ module load Biopython/1.73-foss-2018a-Python-2.7.14
zhuofei@n204 ~$ module load BEDTools/2.26.0-GCCcore-6.4.0
zhuofei@n204 ~$ ml purge
zhuofei@n204 ~$ module load RepeatExplorer2/20181109-foss-2018a-Python-3.6.4
Sapelo2 common compatible toolchains

GCC 5.4 Family:
- foss-2016b
- gompi-2016b
- GCC-5.4.0-2.26
- GCCcore-5.4.0
- gmvolf-2016b
- gmvapich2-2016b

GCC 6.4 Family:
- foss-2018a
- gompi-2018a
- GCC-6.4.0-2.28
- GCCcore-6.4.0
- iccifort/2018.1.163
- imvapich/2018a (QDR)
- iomkl/2018a
- iompi/2018a
Sapelo2 common compatible toolchains

GCC 8.3 Family:

- gcc-8.3
- iccifort/2019.3.199
- icompi/2019.03
- icomkl/2019.03
- intel/2019b
- icompi-2019b
- icompi/2019b
- icompi/2019b
Computing Resources Requesting
https://wiki.gacrc.uga.edu/wiki/Systems

• What are computing resources?
• How to know available computing resources on compute nodes?
• Why does my job have to pend long time in the queue?
• How much memory do I need for running my job?
• How to do computing resource scaling-up?
• How to use local scratch to improve job IO performance?
• Our advice on computing resource usage
What are computing resources?

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 ➔ Core number n should be equal to thread number t (1 ≤ n ≤ 48 for batch)
   - MPI Job ➔ Default: Process number p = Core number n (1 ≤ n ≤ 48*N for batch)
   - Non-default Scenario: p < n (-np p option to mpirun/mpiexec is needed!)
4. Maximum memory for running serial or threaded job on a compute node:

   - **batch**: 1GB
     - Max 48 cores: 120GB
     - Max 32 cores: 184GB
   - **highmem_q**: 1GB
     - Max 48 cores: 502GB
     - Max 64 cores: 990GB

5. Wall-clock time: Job running time from start to finish (Job pending time is NOT included; Default 1 hour, if not requested)

6. Local scratch: Each compute node has local physical hard drives that you can utilize as temporary storage for running job (Sapelo2 Disk Storage)
How to know available computing resources on compute nodes?

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
```
zhuofei@sapelo2-sub2 ~$ mdiag -v -n | head -n 2; mdiag -v -n | grep batch | grep Intel

<table>
<thead>
<tr>
<th>Name</th>
<th>State</th>
<th>Procs</th>
<th>Memory</th>
<th>Load</th>
<th>Classes</th>
<th>Features</th>
<th>Local scratch (KB)</th>
<th>Free mem:Total mem (MB)</th>
<th>Free cores:Total cores</th>
</tr>
</thead>
<tbody>
<tr>
<td>n220</td>
<td>Drained</td>
<td>0:28</td>
<td>60098:64019</td>
<td>0.00</td>
<td>[batch]</td>
<td>EDR,Intel,Broadwell,Intel_EDR,general GRES=lscratch:850880976,...,core:28,thread:28 NODEINDEX=220</td>
<td>95</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n240</td>
<td>Down</td>
<td>0:32</td>
<td>186555:191674</td>
<td>0.04</td>
<td>[batch]</td>
<td>EDR,Intel,Skylake,Intel_EDR,general GRES=lscratch:851862672,...,core:32,thread:32 NODEINDEX=240</td>
<td>176</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

- **Node is down**: n240, n267
- **Node is temporary down due to high load**: n267
- **Not accept no new jobs**: n220
- **Local scratch (KB)**: GRES=lscratch
- **Free mem:Total mem (MB)**:
  - n207: 21063:64019
  - n208: 42188:64019
  - n209: 29207:64019
  - n220: 60098:64019
  - n265: 83646:191674
- **Free cores:Total cores**:
  - n207: 6:28
  - n208: 1:28
  - n209: 3:28
  - n220: 0:28
  - n265: 0:32

6/9/2020

**Using GACRC HPC Computing Facility - Sapelo2 Cluster**

** spurred by request from GACRC HPC Computing Facility students**
Why does my job have to pend long time 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
How much memory do I need for running my job?

- Factors playing a role in answering this question:
  - Data size. 1GB or 500GB?
  - The way you run the software. Threaded or not? What options you are using?
  - Algorithm optimization and tuning.

We would suggest you to check the documentation provided by the developers of the software or contact the developers, for advices on how to estimate the memory your job needs.
How to do computing resource scaling?

• An approximate approach: Linear regression

1. Reduce your computational scale. E.g.: Actual 20 threads ➔ 4 threads : scaling factor $f \sim 5$
2. Run a smaller job with 4 threads by using smaller computing resources
3. After the job finished successfully, `qstat -f 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 $f$ in step 1 and numbers in step 3: E.g.: $f \sim 5$ ➔ Actual job will need ~20 GB memory and ~1 hour running time

• However, for most real cases, scaling is nonlinear!
How to use local scratch to improve job IO performance?
https://wiki.gacrc.uga.edu/wiki/Running_Jobs_on_Sapelo2#How_to_run_a_job_from_the_compute_node.27s_local_disk_.28Flscratch.29

- Each compute node has a file system called /lscratch on the node's local SSD
- **Single node jobs** (serial or threaded) that need to perform a lot of disk IO can benefit from running from /lscratch; **MPI jobs** shall NOT use /lscratch!
- To request a number of KB (N) that job will use in /lscratch per core:
  
  \[
  \texttt{#PBS -l gres=lscratch:N} \quad \text{(note: N must be in the unit of KB for each core!)}
  \]

  E.g., to request total 80GB of "lscratch" allocated to a 4-core job:
  
  \[
  \texttt{#PBS -l nodes=1:ppn=4} \\
  \texttt{#PBS -l gres=lscratch:20000000} \quad \text{N*ppn/1000/1000 = 80GB}
  \]
Our advice on computing resource usage

1. Jobs that cannot run with multiple cores or across multiple nodes will NOT run faster if more than one core or more than one node are requested!

2. Check the documentation provided by the developers of the software or contact the developers, for advices on how to estimate the memory your job will need.

3. Do resource scaling-up by yourself. Not to start with too many cores, unless you already know that the application scales well.
Our advices on computing resource usage

4. If you are running an application (e.g. Bowtie2) in a multi-threaded job using multi-cores, please test it with a low number of cores (e.g. 4), then increase the number of cores to see how well the application parallelizes (by dividing the CPU time by the Wall-clock running time).

5. The optimal number of cores depends on the application and the data size. You should not assume that a multi-threaded job will always run faster with more cores, as that is sometimes not the case.
Run MPI Job

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

batch → QDR (59 nodes) → AMD Opteron (59)
|        | EDR (120 nodes) | AMD EPYC (54) |
|        |                |              |
|        |                | Intel Xeon Skylake (35) |
|        |                | Intel Xeon Broadwell (31) |

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

```bash
#PBS -q batch ➔ Use batch queue
#PBS -l nodes=2:ppn=32:EPYC ➔ Request 64 cores from 2 AMD EPYC nodes (EDR)
#PBS -l pmem=2gb ➔ Request 2GB memory for each MPI process, so 64GB/node ✔️
#PBS -l walltime=60:00:00 ➔ Request 60 hours wall-clock time

#PBS -q batch ➔ Use batch queue
#PBS -l nodes=4:ppn=32:Skylake ➔ Request 128 cores from 4 Intel Skylake nodes (EDR)
#PBS -l pmem=4gb ➔ Request 4GB memory for each MPI process, so 128GB/node ✔️
#PBS -l walltime=60:00:00 ➔ 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

<table>
<thead>
<tr>
<th>PBS command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#PBS -q batch</td>
<td>Use batch queue</td>
</tr>
<tr>
<td>#PBS -l nodes=5:ppn=12:EDR</td>
<td>Request 60 cores from 5 EDR nodes (AMD or Intel)</td>
</tr>
<tr>
<td>#PBS -l pmem=2gb</td>
<td>Request 2GB memory for each MPI process, so 24GB/node</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 -q batch</td>
<td>Use batch queue</td>
</tr>
<tr>
<td>#PBS -l nodes=5:ppn=12:AMD:EDR</td>
<td>Request 60 cores from 5 AMD EDR nodes (AMD EPYC)</td>
</tr>
<tr>
<td>#PBS -l pmem=5gb</td>
<td>Request 5GB memory for each MPI process, so 60GB/node</td>
</tr>
<tr>
<td>#PBS -l walltime=60:00:00</td>
<td>Request 60 hours wall-clock time</td>
</tr>
</tbody>
</table>
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-iccifort-2015.2.164-GCC-4.8.5</td>
<td>MVAPICH2/2.3.2-GCC-8.2.0-2.31.1-EDR</td>
</tr>
<tr>
<td>MVAPICH2/2.2-iccifort-2018.1.163-GCC-6.4.0-2.28</td>
<td>MVAPICH2/2.3.2-iccifort-2015.2.164-GCC-4.8.5-EDR</td>
</tr>
<tr>
<td>MVAPICH2/2.3.2-GCC-8.2.0-2.31.1-QDR</td>
<td>MVAPICH2/2.3.2-iccifort-2018.1.163-GCC-6.4.0-2.28-EDR</td>
</tr>
</tbody>
</table>
Run MPI Job – Request Memory for MPI job

Request memory for MPI job: use mem or pmem in your job script:

mem : Total memory for running a MPI job

pmem: Maximum memory for running each MPI process of a MPI job

For example,

#PBS -l nodes=2:ppn=24

#PBS -l pmem=2gb

is equally requesting 96gb of total memory.

⚠️ We strongly suggest you to use pmem, instead of mem, for running a MPI job
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=4gb
#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=4:ppn=32:EDR
#PBS -l pmem=2g
#PBS -l walltime=48:00:00
#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 --np 90 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?

```bash
mdiag -v -n | head -n 2 ; mdiag -v -n | grep gpu_q
```

![Diagram showing node details on gpu_q](attachment:image.png)
zhuofei@n204 ~$ mdiag -v -n | head -n 2 ; mdiag -v -n | grep gpu_q

compute node summary

<table>
<thead>
<tr>
<th>Name</th>
<th>State</th>
<th>Procs</th>
<th>Memory</th>
<th>... Load ...</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>gn48</td>
<td>Draining</td>
<td>0:16</td>
<td>63194:128730</td>
<td>0.81</td>
<td>Intel,QDR,Intel_QDR,<strong>GPU,K40</strong>,... GRES=lscratch:209428400,...,core:16,thread:16,<strong>GPUS:8</strong> NODEINDEX=48</td>
</tr>
<tr>
<td>gn49</td>
<td>Busy</td>
<td>0:16</td>
<td>63195:128731</td>
<td>7.82</td>
<td>Intel,QDR,Intel_QDR,magma,<strong>GPU,K40</strong>,... GRES=lscratch:209163764,...,core:16,thread:16,<strong>GPUS:8</strong> NODEINDEX=49</td>
</tr>
<tr>
<td>gn164</td>
<td>Busy</td>
<td>0:12</td>
<td>47357:96509</td>
<td>7.79</td>
<td>Intel,QDR,Intel_QDR,<strong>GPU,K20</strong>,... GRES=lscratch:203950948,...,core:12,thread:12,<strong>GPUS:7</strong> NODEINDEX=164</td>
</tr>
<tr>
<td>gn165</td>
<td>Busy</td>
<td>0:12</td>
<td>47357:96509</td>
<td>7.46</td>
<td>Intel,QDR,Intel_QDR,<strong>GPU,K20</strong>,... GRES=lscratch:203950688,...,core:12,thread:12,<strong>GPUS:7</strong> NODEINDEX=165</td>
</tr>
<tr>
<td>gn203</td>
<td>Busy</td>
<td>0:12</td>
<td>47357:96509</td>
<td>6.11</td>
<td>Intel,QDR,Intel_QDR,<strong>GPU,K20</strong>,... GRES=lscratch:203951356,...,core:12,thread:12,<strong>GPUS:7</strong> NODEINDEX=203</td>
</tr>
<tr>
<td>gn261</td>
<td>Running</td>
<td>28:32</td>
<td>175290:191674</td>
<td>0.87</td>
<td>EDR,Intel,Skylake,<strong>P100,GPU</strong>,Intel_EDR GRES=lscratch:875371272,...,core:32,thread:32,<strong>GPUS:1</strong> NODEINDEX=261</td>
</tr>
<tr>
<td>gn284</td>
<td>Running</td>
<td>28:32</td>
<td>175290:191674</td>
<td>0.90</td>
<td>EDR,Intel,Skylake,<strong>P100,GPU</strong>,Intel_EDR GRES=lscratch:875369800,...,core:32,thread:64,<strong>GPUS:1</strong> NODEINDEX=284</td>
</tr>
</tbody>
</table>
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>PBS command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#PBS -q 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 (any GPU)</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>PBS command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#PBS -q 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 (any GPU)</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>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#PBS -q 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 (K20 GPU)</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>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#PBS -q 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 (P100 GPU)</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>#PBS -q gpu_q</td>
<td>Use <code>gpu_q</code> queue</td>
</tr>
<tr>
<td>#PBS -l nodes=1:ppn=4:gpu=1:default</td>
<td>Request 4 CPU cores with 1 GPU card from 1 node</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 -q gpu_q</td>
<td>Use <code>gpu_q</code> queue</td>
</tr>
<tr>
<td>#PBS -l nodes=1:ppn=4:gpu=1:P100:default</td>
<td>Request 4 CPU cores with 1 P100 GPU card from 1 node</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 – GPU utilization of a GPU job

Run **nvidia-smi** command on a GPU node, e.g., gn49, a K40 GPU node

```plaintext
... (continued) ...
```

<table>
<thead>
<tr>
<th>Processes:</th>
<th>GPU Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU</td>
<td>PID</td>
</tr>
<tr>
<td>0</td>
<td>27800</td>
</tr>
<tr>
<td>1</td>
<td>28096</td>
</tr>
<tr>
<td>4</td>
<td>28744</td>
</tr>
<tr>
<td>5</td>
<td>29945</td>
</tr>
</tbody>
</table>

```
Run `nvidia-smi` command on a GPU node, e.g., gn49, a K40 GPU node

Warning: Permanently added 'gn49' (ECDSA) to the list of known hosts.
Sun May 17 21:59:37 2020

---

<table>
<thead>
<tr>
<th>NVIDIA-SMI 410.129</th>
<th>Driver Version: 410.129</th>
<th>CUDA Version: 10.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU Name</td>
<td>Persistence-M</td>
<td>Bus-ID</td>
</tr>
<tr>
<td>Fan Temp</td>
<td>Perf</td>
<td>Pwr:Usage/Cap</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GPU</th>
<th>Name</th>
<th>Persistence-M</th>
<th>Bus-ID</th>
<th>Disp.A</th>
<th>Volatile Uncorr. ECC</th>
<th>Fan</th>
<th>Temp</th>
<th>Perf</th>
<th>Pwr:Usage/Cap</th>
<th>Memory-Usage</th>
<th>GPU-Util</th>
<th>Compute M.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Tesla K40m</td>
<td>On</td>
<td>00000000:04:00.0 Off</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Tesla K40m</td>
<td>On</td>
<td>00000000:05:00.0 Off</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Tesla K40m</td>
<td>On</td>
<td>00000000:08:00.0 Off</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Tesla K40m</td>
<td>On</td>
<td>00000000:09:00.0 Off</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Tesla K40m</td>
<td>On</td>
<td>00000000:86:00.0 Off</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Tesla K40m</td>
<td>On</td>
<td>00000000:87:00.0 Off</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

---

6/9/2020

USING GACRC HPC COMPUTING FACILITY - SAPELO2 CLUSTER
Obtain Job Details

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

Option 1:

- `qstat -n1 -u MyID` for running/pending/completed jobs on cluster
- `qstat -n1r -u MyID` for running jobs on cluster
- `qstat -n1rt -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`
GACRC Wiki [http://wiki.gacrc.uga.edu](http://wiki.gacrc.uga.edu)


Disk Storage: [https://wiki.gacrc.uga.edu/wiki/Disk_Storage](https://wiki.gacrc.uga.edu/wiki/Disk_Storage)

Running Jobs: [https://wiki.gacrc.uga.edu/wiki/Running_Jobs_on_Sapelo2](https://wiki.gacrc.uga.edu/wiki/Running_Jobs_on_Sapelo2)

Monitoring Jobs: [https://wiki.gacrc.uga.edu/wiki/Monitoring_Jobs_on_Sapelo2](https://wiki.gacrc.uga.edu/wiki/Monitoring_Jobs_on_Sapelo2)

Job Submission Queue: [https://wiki.gacrc.uga.edu/wiki/Job_Submission_Queues](https://wiki.gacrc.uga.edu/wiki/Job_Submission_Queues)

GPU and CUDA: [https://wiki.gacrc.uga.edu/wiki/GPU](https://wiki.gacrc.uga.edu/wiki/GPU)

Build Software: [https://wiki.gacrc.uga.edu/wiki/Build_Applications_on_Sapelo2](https://wiki.gacrc.uga.edu/wiki/Build_Applications_on_Sapelo2)

Software: [https://wiki.gacrc.uga.edu/wiki/Software](https://wiki.gacrc.uga.edu/wiki/Software)
GACRC Support
https://uga.teamdynamix.com/TDClient/Requests/ServiceCatalog?CategoryID=11593

➢ Job Troubleshooting:
  ✓ Your user name
  ✓ Your job ID
  ✓ Your working directory
  ✓ The queue name and command you used to submit the job

➢ Software Installation:
  ✓ Specific name and version of the software
  ✓ Download website
  ✓ Supporting package information if have

❖ Note: In order to ensure a smooth troubleshooting process, please make sure your data are in correct format!
Thank You!

TelephoneNumber 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/)