Sapelo2 Cluster Advanced Topics

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

• GACRC
• HPC Conceptual Framework
• Sapelo2 HPC Resources
• Request Computing Resources
• Use local scratch storage
• Run MPI Job
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](http://wiki.gacrc.uga.edu)
Help and Support: [https://wiki.gacrc.uga.edu/wiki/Getting_Help](https://wiki.gacrc.uga.edu/wiki/Getting_Help)
Web Site: [http://gacrc.uga.edu](http://gacrc.uga.edu)
HPC Conceptual Framework

- **node1**
  - host memory
  - core
  - core

- **node2**
  - host memory
  - core
  - core

- **node3**
  - host memory
  - core
  - core

- **node4**
  - host memory
  - core
  - core

- **node5**
  - host memory
  - core
  - core

- **node6**
  - host memory
  - core
  - core

**Network**
HPC Conceptual Framework

• Serial single-core job
  1. Run computational task in a single thread, using one CPU core on a single node

• Threaded memory-sharing parallel job
  1. Run task in multi-threads, using more than one CPU core on a single node
  2. Programming library: OpenMP (Open Multi-Processing)
  3. Important ENV: OMP_NUM_THREADS, OMP_PROC_BIND
HPC Conceptual Framework

• MPI parallel job
  1. Run task in multiple MPI processes; using one or more CPU cores on a single or multiple nodes
  2. Programming library: OpenMPI, MVAPICH2, Intel MPI Library

• GPU job
  1. Run task in a single thread or multi-threads, using one or more CPU cores on a single host
  2. Use one or more GPU devices equipped in the host
  3. Programming library: CUDA or OpenACC
SAPELO2 CLUSTER ADVANCED TOPICS

MPI

OpenMP

OpenACC

Single core

node1

GPU

node2

GPU

node3

GPU

node4

GPU

node5

GPU

node6

GPU

host memory

core

core

core

host memory

core

core

core

host memory

core

core

core

host memory

core

core

core

host memory

core

core

core

host memory

core

core

core

host memory

core

core

core

host memory

core

core

core

host memory

core

core

core

host memory

core

core

core

host memory

core

core

core

host memory

core

core

core

host memory

Single core

node1

GPU

node2

GPU

node3

GPU

node4

GPU

node5

GPU

node6

GPU

host memory

core

core

core

host memory

core

core

core

host memory

core

core

core

host memory

core

core

core

host memory

core

core

core

host memory

core

core

core

host memory

core

core

core

host memory

core

core

core

host memory

core

core

core

host memory

core

core

core

host memory

core

core

core

host memory

core

core

core

host memory

core

core

core

host memory

core

core

core

host memory

core

core

core

host memory

core

core

core

host memory

core

core

core

host memory

core

core

core

host memory

core

core

core

host memory
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/
Sapelo2 HPC Resources
https://wiki.gacrc.uga.edu/wiki/Systems#Sapelo2

- **sinfo** comes to help!
  - **sinfo**: show all available partitions, with time limit, node number, state, and a list of nodes
  - **sinfo -p batch (-l)**: report on specific partition, e.g., batch, highmem_p, gpu_p, buyin_p
  - **sinfo -p batch -t IDLE**: report on specific partition while specifying the state of nodes to view
    - e.g., IDLE, MIXED, ALLOCATED, DOWN, and DRAIN
  - **sinfo -p batch -s**: summarize on specific partition with NODES(A/I/O/T), where A is for MIXED + ALLOCATED; I is for IDLE; O is for DOWN + DRAIN; T is for TOTAL
  - **sinfo --help**
  - **Slurm sinfo**
  - **sinfo-gacrc**
Sapelo2 HPC Resources
https://wiki.gacrc.uga.edu/wiki/Systems#Sapelo2

- **scontrol show** comes to help!
  - `scontrol show partition`: display the state of all available partitions
  - `scontrol show partition batch`: display the state of the specified partition, e.g., batch, highmem_p, gpu_p, buyin_p
  - `scontrol show node`: display the state (IDLE, MIXED, ALLOCATED, DOWN, and DRAIN) of all nodes
  - `scontrol show node c4-22`: display the state of the specified node
  - `scontrol --help`
  - `Slurm scontrol`

- **sacctmgr show qos** comes to help!
  - `MaxJobsPU/ MaxSubmitPU`: max number of jobs users can run/submit at one time
  - `sacctmgr show -p qos batch_qos | cut -d'|' -f 1,21,22 | column -s '|' -t`
Request Computing Resources – CPU Job

https://wiki.gacrc.uga.edu/wiki/Running_Jobs_on_Sapelo2

1. Partition (**--partition**) : batch/batch_30d, highmem_p/highmem_30d_p

2. Node feature (**--constraint**) : each compute node has a set of features, such as shown with sinfo-gacrc:
   - AMD, EPYC (AMD EPYC), Rome or Naples (AMD EPYC Rome or Naples), Opteron (AMD Opteron)
   - Intel, Skylake or Broadwell (Intel Skylake or Broadwell)
   - EDR, QDR (EDR or QDR Infiniband network)

3. Node number (**--nodes**) :
   - Serial single-core job ➔ 1 node
   - Threaded memory-sharing parallel job ➔ 1 node
   - MPI parallel Job ➔ N nodes (N ≥ 1), with a constraint on network, e.g., **--constraint=EDR**
Request Computing Resources – CPU Job

4. CPU core number (--ntasks, --cpus-per-task, --ntasks-per-node):
   - Serial single-core job ➔ 1 core, e.g., --nodes=1, --ntasks=1, --cpus-per-task=1
     --nodes=1, --ntasks=1
     --ntasks=1
   - Threaded Job ➔ Core number n = thread number t, e.g., --nodes=1, --ntasks=1, --cpus-per-task=20
     --nodes=1, --ntasks=20
   - MPI Job ➔ Default: Core number n = Process number p
     Non-default: n > p (--np p option to mpirun/mpiexec is needed!)
     --nodes=30, --ntasks-per-node=20, --cpus-per-task=1
     --ntasks=600, --cpus-per-task=1
     --ntasks=600
Request Computing Resources – GPU Job

https://wiki.gacrc.uga.edu/wiki/Running_Jobs_on_Sapelo2

1. Partition (**partition**) : gpu_p/gpu_30d_p
2. Node number (**nodes**) : usually 1, --nodes=1
3. GPU devices (**gres**) : e.g., --gres=gpu:1
   - --gres=gpu:K40:1
   - --gres=gpu:K40:4
   - --gres=gpu:P100:1
4. Host CPU Core number (**ntasks**, **cpus-per-task**, **ntasks-per-node**):
   - Single core ➔ 1 core, e.g., --ntasks=1
   - Multiple cores ➔ Core number n = thread number t, e.g., --nodes=1, --ntasks=1, --cpus-per-task=2
Request Computing Resources – Memory

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

1. Single-node job (Serial single-core job or threaded job) : request total memory for job, e.g.
   --mem=10gb

2. MPI job : request memory for each process, e.g.
   --mem-per-cpu=2gb (2048mb)

3. GPU job: request total host memory for job, e.g.
   --mem=40gb

   GPU devices are not shared between jobs, so your job will be able to use the entire memory on a GPU device (no need to request it).

How to know memory on a node?
Let’s try what we learned `sinfo` and `scontrol show` - A practical example:

```
zhuofei@ss-sub2 ~$ sinfo -p batch -t IDLE
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
batch    up  7-00:00:00   5  drain c4-16,ra1-13,ra3-[21,23-24]
batch    up  7-00:00:00   1  idle d3-3
```

```
zhuofei@ss-sub2 ~$ scontrol show node d3-3
NodeName=d3-3  Arch=x86_64  CoresPerSocket=32
  CPUAlloc=0  CPUTot=32  CPULoad=0.01
AvailableFeatures=AMD,EPYC,Naples,EDR,Beta
ActiveFeatures=AMD,EPYC,Naples,EDR,Beta
Gres=lscratch:890
NodeAddr=d3-3  NodeHostName=d3-3  Version=20.11.8
  OS=Linux 3.10.0-1160.36.2.el7.x86_64 #1 SMP Wed Jul 21 11:57:15 UTC 2021
RealMemory=128670  AllocMem=0  FreeMem=119040
  Sockets=1  Boards=1
State=IDLE  ThreadsPerCore=1  TmpDisk=0  Weight=1  Owner=N/A  MCS_label=N/A
Partitions=batch,batch_30d
CfgTRES=cpu=32,mem=128670M,billing=32
  AllocTRES=
  ......  Comment=(null)
```

```
CPU core
```

```
Node feature
```

```
Local storage
```

```
Node memory!
UnAllocMem = RealMemory – AllocMem ≠ FreeMem
```
Let’s try what we learned `sinfo` and `scontrol show` - A practical example:

```
$ sinfo -p batch -t ALLOC
PARTITION AVAIL  TIMELIMIT  NODES  STATE  NODELIST
batch  up 7-00:00:00   239  mix a3-[1-20],b1-[1-23],b2-[21-24],c1-[1-4,6,9-19],c2-[1,3-4,7,9-10,12-13,16-24], ......
batch  up 7-00:00:00   47  alloc c1-[20-24],c2-[2,5-6,8,14-15],c4-[12,21],d1-[12-13,15],d2-[10,12-14,19-22], ......
```

```
$ scontrol show node c1-20
NodeName=c1-20 Arch=x86_64 CoresPerSocket=14
CPUAlloc=28 CPUTot=28 CPULoad=28.10
AvailableFeatures=Intel,Broadwell,EDR,Gamma
ActiveFeatures=Intel,Broadwell,EDR,Gamma
GRES=lscratch:890
NodeAddr=c1-20 NodeHostName=c1-20 Version=20.11.8
OS=Linux 3.10.0-1160.36.2.el7.x86_64 #1 SMP Wed Jul 21 11:57:15 UTC 2021
RealMemory=64200 AllocMem=50176 FreeMem=36505 Sockets=2 Boards=1
State=ALLOCATED ThreadsPerCore=1 TmpDisk=0 Weight=1 Owner=N/A MCS_label=N/A
Partitions=batch,batch_30d
BootTime=2021-07-28T10:58:54 SlurmdStartTime=2021-07-28T11:00:26
CfgTRES=cpu=28,mem=64200M,billing=28
AllocTRES=cpu=28,mem=49G
......
Comment=(null)
```

Node memory!  
\[ \text{UnAllocMem} = 64200 - 50176 = 14024 \text{ MB} \]
Let’s try **sinfo-nodes** - A practical example:

```
PARTITION NODELIST STATE   CPUS(A/I/O/T) MEMORY(MB) UNALLOCMEM(MB) AVAIL_FEATURES GRES
scavenge_p a1-1 idle 0/64/0/64  128590  128590 AMD,EPYC,Rome,EDR,Beta,buyin,bilskie lscratch:890
bilskie_p a1-1 idle 0/64/0/64  128590  128590 AMD,EPYC,Rome,EDR,Beta,buyin,bilskie lscratch:890
scavenge_p a1-2 idle 0/64/0/64  128590  128590 AMD,EPYC,Rome,EDR,Beta,buyin,bilskie lscratch:890
bilskie_p a1-2 idle 0/64/0/64  128590  128590 AMD,EPYC,Rome,EDR,Beta,buyin,bilskie lscratch:890
scavenge_p a1-3 idle 0/64/0/64  128590  128590 AMD,EPYC,Rome,EDR,Beta,buyin,bilskie lscratch:890
bilskie_p a1-3 idle 0/64/0/64  128590  128590 AMD,EPYC,Rome,EDR,Beta,buyin,bilskie lscratch:890
scavenge_p a1-4 idle 0/64/0/64  128590  128590 AMD,EPYC,Rome,EDR,Beta,buyin,bilskie lscratch:890
bilskie_p a1-4 idle 0/64/0/64  128590  128590 AMD,EPYC,Rome,EDR,Beta,buyin,bilskie lscratch:890
scavenge_p a1-5 idle 0/64/0/64  128590  128590 AMD,EPYC,Rome,EDR,Beta,buyin,bilskie lscratch:890
```

\[A: \text{ALLOC}, I: \text{IDLE}, O: \text{OTHER}, T: \text{TOTAL}\]

```
PARTITION NODELIST STATE   CPUS(A/I/O/T) MEMORY(MB) UNALLOCMEM(MB) AVAIL_FEATURES GRES
batch    c1-20 allocated 28/0/0/28  64200  14024 Intel,Broadwell,EDR,Gamma lscratch:890
batch_30d c1-20 allocated 28/0/0/28  64200  14024 Intel,Broadwell,EDR,Gamma lscratch:890
```

Alternative:
```
sinfo-nodes | head -n1 ; sinfo-nodes | grep c1-20
```
Use local scratch storage
https://wiki.gacrc.uga.edu/wiki/Disk_Storage#lscratch_file_system

- Each compute node has a file system called /lscratch on the node's local SSD
- **Single-node jobs** *(serial single-core or threaded)* that need to perform a lot of disk IO can benefit from running from /lscratch; in general, **MPI jobs** cannot use /lscratch
- Slurm supports the ability to define and schedule arbitrary Generic RESources (**GRES**)  
  
  ```bash
  scontrol show config | grep GresTypes
  sinfo-gacrc
  #SBATCH --nodes=1
  #SBATCH --gres=lscratch:200
  #SBATCH --partition=batch
  #SBATCH --nodes=1
  #SBATCH --gres=lscratch:200
  #SBATCH --ntasks=1
  #SBATCH --mem=4G
  ```
- Remember to clean up lscratch by moving data back to scratch, before your job exits from the node! (you can do this in your job script)
Our advice on computing resource usage

https://wiki.gacrc.uga.edu/wiki/Best_Practices_on_Sapelo2

- 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.

- 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!

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

- If you are running an application in a multi-threaded job using multi-cores, please test it with a low number of cores, then increase the number of cores to see how well the application parallelizes (by dividing the CPU time by the wall-clock runtime (job elapsed time), `sacct-gacrc -X -j <jobID>`).

- 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 Jobs
https://wiki.gacrc.uga.edu/wiki/Running_Jobs_on_Sapelo2#MPI_Job
https://wiki.gacrc.uga.edu/wiki/MPI

- Sapelo2 nodes communicate via InfiniBand (IB): EDR and QDR
- IB is a pervasive, high-bandwidth, low-latency interconnect which requires low processing overhead and is the preferred solution for high performance storage interconnection and internodal connectivity in HPC system.
  - SDR -> DDR -> QDR(40Gbps) -> EDR(100Gbps) -> HDR(200Gbps)
  - RDMA (Remote Direct Memory Access)
  - Simple protocol stack, high processing efficiency, and simple management
  - trust-based and flow-control mechanism to ensure connection integrity
- Intro to IB
Run MPI Job – Slurm headers

<table>
<thead>
<tr>
<th>Slurm Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>#SBATCH --partition=batch</code></td>
<td># Partition name</td>
</tr>
<tr>
<td><code>#SBATCH --nodes=20</code></td>
<td># Number of nodes</td>
</tr>
<tr>
<td><code>#SBATCH --constraint=EDR</code></td>
<td># Node feature</td>
</tr>
<tr>
<td><code>#SBATCH --ntasks=600</code></td>
<td># Number of MPI ranks</td>
</tr>
<tr>
<td><code>#SBATCH --ntasks-per-node=30</code></td>
<td># How many tasks on each node</td>
</tr>
<tr>
<td><code>#SBATCH --cpus-per-task=1</code></td>
<td># Number of cores per MPI rank</td>
</tr>
<tr>
<td><code>#SBATCH --mem-per-cpu=1gb</code></td>
<td># Memory per processor</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Slurm Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>#SBATCH --partition=batch</code></td>
<td># Partition name</td>
</tr>
<tr>
<td><code>#SBATCH --constraint=EDR</code></td>
<td># Node feature</td>
</tr>
<tr>
<td><code>#SBATCH --ntasks=600</code></td>
<td># Number of MPI ranks</td>
</tr>
<tr>
<td><code>#SBATCH --ntasks-per-node=30</code></td>
<td># How many tasks on each node</td>
</tr>
<tr>
<td><code>#SBATCH --cpus-per-task=1</code></td>
<td># Number of cores per MPI rank</td>
</tr>
<tr>
<td><code>#SBATCH --mem-per-cpu=1gb</code></td>
<td># Memory per processor</td>
</tr>
</tbody>
</table>
Run MPI Job – srun or mpirun

• Slurm directly launches MPI tasks via **srun** and performs initialization of communications through PMI2 (Process Management Interface; Slurm integrated MPI)

• Slurm creates a resource allocation for the job; MPI’s **mpirun** to launch MPI tasks

```bash
ml OpenMPI/3.1.4-GCC-8.3.0
srun -n 600 ./mympi.exe
```

```bash
ml OpenMPI/3.1.4-GCC-8.3.0
mpirun -n 600 ./mympi.exe
```

```bash
ml MVAPICH2/2.3.4-GCC-8.3.0
mpirun -n 600 ./mympi.exe
ml impi/2018.5.288-iccifort-2019.5.281
mpirun -n 600 ./mympi.exe
```
Run MPI Job – main OpenMPI libraries

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

<table>
<thead>
<tr>
<th>OpenMPI Version</th>
<th>Toolchain</th>
<th>Libraries Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OpenMPI/3.1.4-GCC-8.3.0</strong></td>
<td>-</td>
<td>-&gt; foss/2019b (OpenBLAS/0.3.7, FFTW/3.3.8, and ScALAPACK/2.0.2)&lt;br&gt;-&gt; gompi/2019b (no math libs)</td>
</tr>
<tr>
<td><strong>OpenMPI/4.0.5-GCC-10.2.0</strong></td>
<td>-</td>
<td>-&gt; foss/2020b (OpenBLAS/0.3.12, FFTW/3.3.8, and ScALAPACK/2.1.0)&lt;br&gt;-&gt; gompi/2020b (no math libs)</td>
</tr>
<tr>
<td><strong>OpenMPI/4.0.5-gcccuda-2020b</strong></td>
<td>-</td>
<td>-&gt; foss cuda/2020b (same math libs as foss/2020b, CUDA/11.1.1)&lt;br&gt;-&gt; gompic/2020b (no math libs, CUDA/11.1.1)</td>
</tr>
</tbody>
</table>

**Note:**

**foss/2020b** and **foss cuda/2020b** don’t work on AMD Opteron nodes. Please use

```bash
#SBATCH --constraint=EDR
```

to use non-Opteron nodes, if you want to run a MPI application that was built with the two toolchains.
Run MPI Job – main MVAPICH2 libraries

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

MVAPICH2/2.3.4-GCC-8.3.0  ->  gmvolf/2019b (OpenBLAS/0.3.7, FFTW/3.3.8, and ScaLAPACK/2.0.2)
                          ->  gmvapich2/2019b (no math libs)
MVAPICH2/2.3.6-GCC-9.3.0  ->  gmvolf/2020a (OpenBLAS/0.3.9, FFTW/3.3.8, and ScaLAPACK/2.0.2)
                          ->  gmvapich2/2020a (no math libs)

Note:

1. At this moment, MVAPICH2 is not configured to use PMI with Slurm on Sapelo2. Please use
   mpirun/mpiexec to launch your MPI tasks on Sapelo2, if your MPI application uses MVAPICH2.
2. gmvolf/2020a and gmvapich2/2020a don’t work on AMD Opteron nodes. Please use
   #SBATCH --constraint=EDR
Run MPI Job – main Intel MPI libraries

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

   → iimpi/2019b (no math libs)
   → iimpi/2020b (no math libs)

Note:
At this moment, Intel MPI is not configured to use PMI with Slurm on Sapelo2. Please use mpirun/mpiexec to launch your MPI tasks on Sapelo2, if your MPI application uses Intel MPI.
Run MPI Job – two code compilation examples

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

```
# qlogin
ml foss/2019b
ml Boost/1.71.0-gompi-2019b
OPT="-ansi -std=c++11 -Wpedantic -Wall -march=x86-64 -mtune=generic -O2"
mpicxx ${OPT} mpi_cal.cpp -o mpi_cal.x
```

will open an interactive session on a default AMD Opteron, Intel Skylake or Broadwell node.

```
# srun
-srun --pty -p inter_p --mem=2G --nodes=1 --constraint=Skylake --ntasks-per-node=1 --time=2:00:00 -J compile /bin/bash -l
ml foss/2019b
ml Boost/1.71.0-gompi-2019b
OPT="-ansi -std=c++11 -Wpedantic -Wall -march=x86-64 -mtune=native -O2"
mpicxx ${OPT} mpi_cal.cpp -o mpi_cal.x
```

produce code optimized for the most common processors.

```
#SBATCH
--constraint=Skylake
```

then put this in your job script:

```
#SBATCH --constraint=Skylake
```
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/