Using Sapelo2 Cluster at the GACRC II

Georgia Advanced Computing Resource Center (GACRC)

Enterprise Information Technology Services(EITS)

The University of Georgia

Outline

- GACRC
- HPC Conceptual Framework
- Get Information about Computing Resources
- Request Computing Resources
- FAQs

GACRC

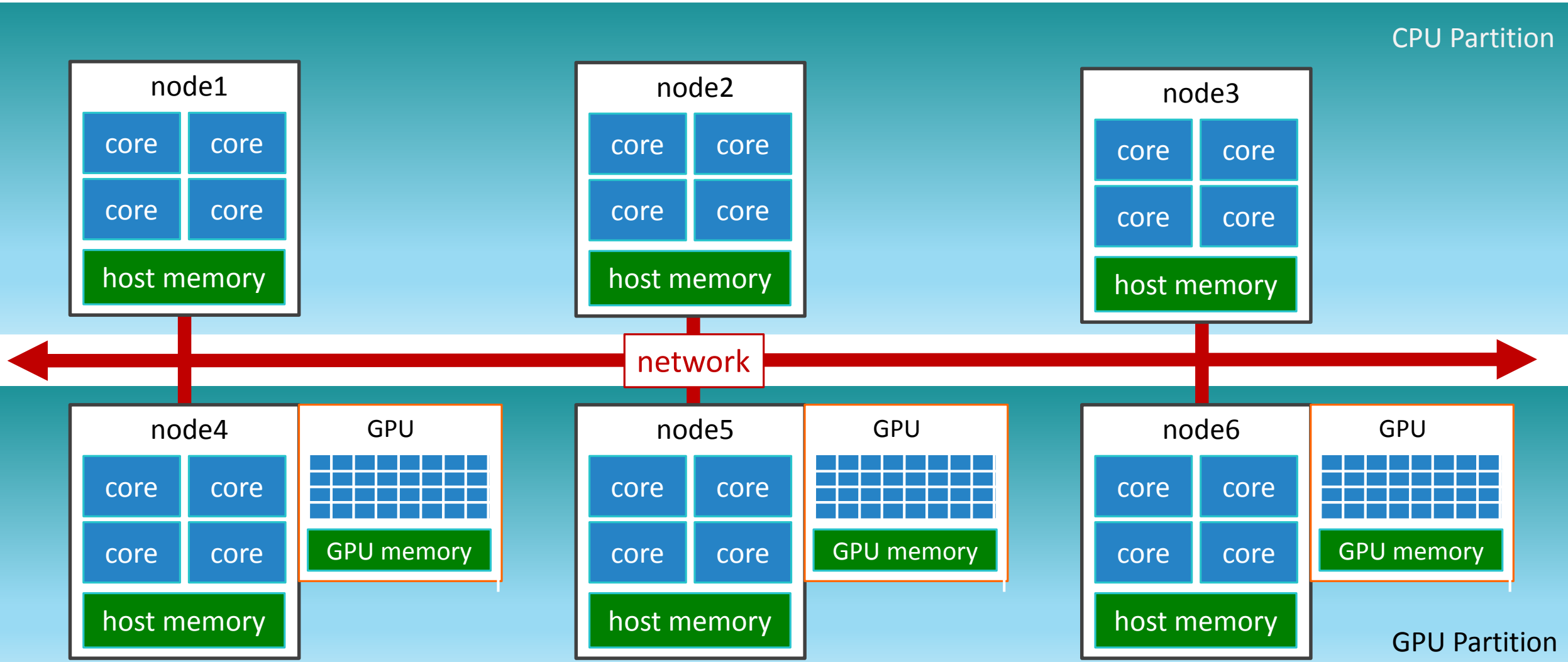
- A high-performance-computing (HPC) center at the UGA
- Provide to the UGA research and education community an advanced computing environment:
  - HPC computing and networking infrastructure located at the Boyd Data Center
  - Comprehensive collection of scientific, engineering and business applications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

Wiki: http://wiki.gacrc.uga.edu

Help and Support: http://help.gacrc.uga.edu

Web Site: http://gacrc.uga.edu

HPC Conceptual Framework



HPC Conceptual Framework

- Serial single-core job

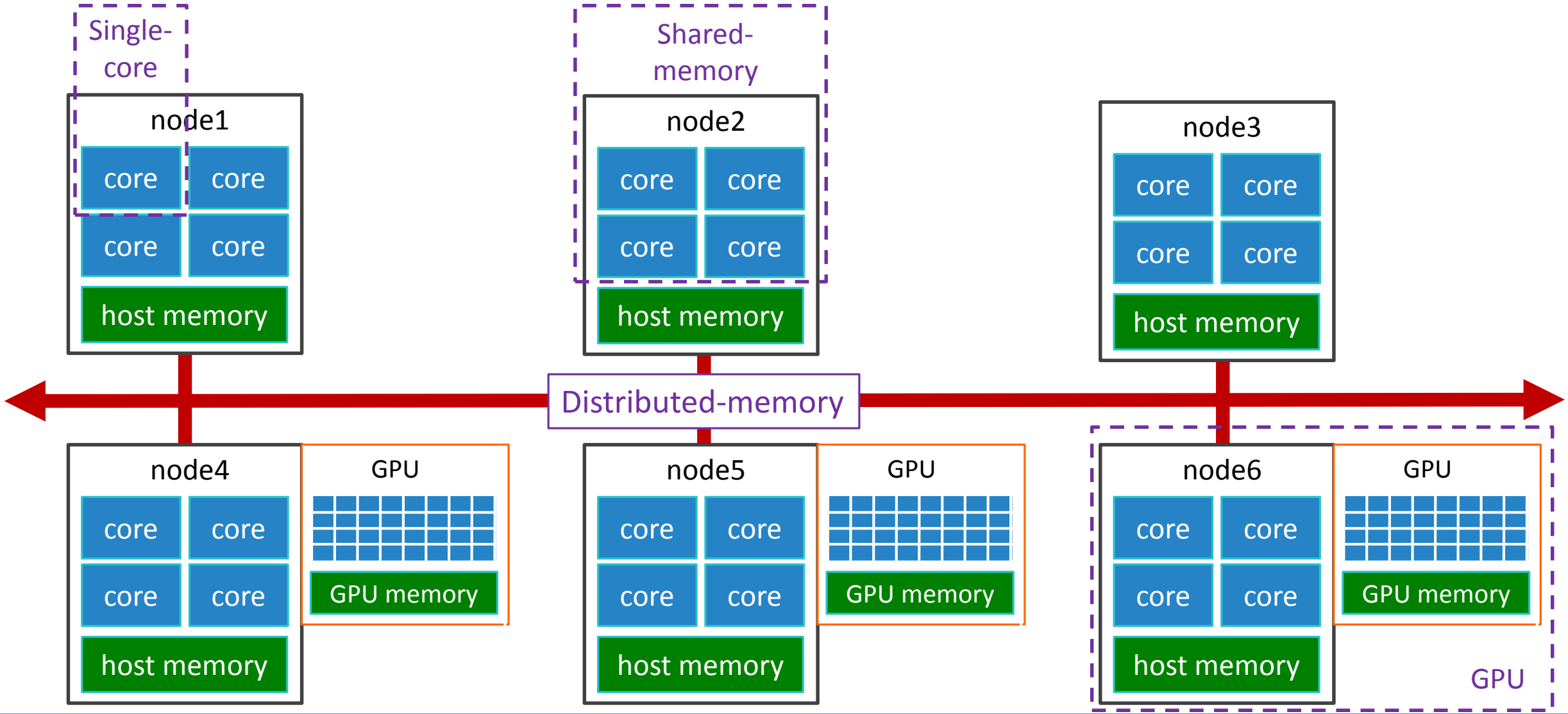
  1. Computational task runs in a main thread on a single node, using one CPU core

- Shared-memory parallel job

  1. Computational task runs in multiple threads on a single node, using more than one CPU cores
  2. Programming library: OpenMP (Open Multi-Processing)
  3. OpenMP environmental variables: OMP_NUM_THREADS, OMP_PROC_BIND, OMP_PLACE

HPC Conceptual Framework

- Distributed-memory parallel job
  1. Computational task runs in multiple processes on multiple nodes, using many more CPU cores
  2. Programming library: OpenMPI , Intel MPI Library
- GPU job
  1. Task runs in a main or multiple threads on a single host, using one or more CPU cores on the host
  2. Use one or more GPU devices equipped on the host
  3. Programming library: CUDA or OpenACC


Single-core
node1
core
core
core
core
host memory
Shared-memory
node2
core
core
core
core
host memory
node3
core
core
core
core
host memory
Distributed-memory
node4
core
core
core
core
host memory
GPU
GPU memory
node5
core
core
core
core
host memory
GPU
GPU memory
node6
core
core
core
core
host memory
GPU
GPU memory
GPU

Get Information about Sapelo2 HPC Resources

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

- sinfo : Show partitions with time limit, node number, state, and a list of nodes
  - sinfo -p batch : show specific partition, e.g., batch, highmem_p, hugemem_p, gpu_p, buyin_p
  - sinfo -p batch -t idle : show specific partition, specifying the state of nodes to view e.g., IDLE, MIXED
  - 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-gacrc : GACRC wrapper script of sinfo
  - sinfo-gacrc | head -n1; sinfo-gacrc | grep -w batch
  - sinfo-gacrc | head -n1; sinfo-gacrc | grep -w highmem_p
  - sinfo-gacrc | head -n1; sinfo-gacrc | grep -w csp_p

Get Information about Sapelo2 HPC Resources

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

- sinfo-nodes : GACRC wrapper script giving UNALLOCMEM in MBs on compute nodes
  - sinfo-nodes | head -n1; sinfo-nodes | grep -w batch
  - sinfo-nodes | head -n1; sinfo-nodes | grep -w highmem_p
- find_nodes_unallocmem_cores : GACRC wrapper script based on sinfo-nodes:
  - Find compute nodes with the minimum available UNALLOCMEM (GB) and CPU cores on a specific partition.
  - Copy it from /usr/local/training/Sapelo2-II/ to your working folder, e.g., /home/MyID
  - Usage: ./find_nodes_unallocmem_cores -h

    ./find_nodes_unallocmem_cores -p batch -c 20 -m 100

Request Computing Resources for CPU Job

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

1. Partition (--partition) : batch/batch_30d, highmem_p/highmem_30d_p, gpu_p/gpu_30d_p
2. Node feature (--constraint) : each compute node has a set of features, such as shown with sinfo-gacrc:
   - Processor Type:
     - AMD family: EPYC (Milan, Rome, Naples)
     - Intel family: Skylake
   - Infiniband Network
     - EDR (100Gb/s)
3. Node number (--nodes) :
   - Serial single-core job ➔ 1 node
   - Shared-memory parallel job ➔ 1 node
   - Distributed-memory parallel Job ➔ n nodes (n ≥ 1)

Request Computing Resources for 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

  --ntasks=1

- Shared-memory Job ➔ *Core number = Thread number*, e.g., --nodes=1, --ntasks=1, --cpus-per-task=20
- Distributed-memory Job ➔ Default: *Core number = Process number*

  Non-default: *Core number > Process number* (using srun -n is needed!)

  E.g., --nodes=20, --ntasks-per-node=20, --cpus-per-task=1

  --ntasks=400, --cpus-per-task=1

  --ntasks=400

Request Computing Resources for 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 (single node)
3. GPU devices (--gres) : e.g., --gres=gpu:1
   --gres=gpu:P100:1
   --gres=gpu:A100:2
4. Host CPU Core number (--ntasks, --cpus-per-task, --ntasks-per-node):
   - Single core ➔ 1 core, e.g., --ntasks=1
   - Multiple cores ➔ *Core number = thread number*, e.g., --nodes=1, --ntasks=1, --cpus-per-task=4

Request Computing Resources - Memory

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

1. Serial single-core or Shared-memory job : request total memory from single node, e.g.

   --mem=10gb

2. Distributed-memory job : request memory for each process, e.g.

   --mem-per-cpu=2gb (2048mb)

3. GPU job: request total host memory from single host, 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).

FAQ1 - How to find Nodes with UNALLOCMEM>200GB and cores>100

```
zhuofei@ss-sub1 ~$ ./find_nodes_unallocmem_cores -p batch -m 200 -c 100

./find_nodes_unallocmem_cores.sh -p batch -m 200 -c 100

PARTITION      NODELIST  STATE       CPUS(A/I/O/T) MEMORY(MB)    UNALLOCMEM(MB)      AVAIL_FEATURES                    GRES
batch          a4-1      idle        0/128/0/128   515720    515720        AMD,EPYC,Milan,EDR,Beta           lscratch:890
batch          a4-2      idle        0/128/0/128   515720    515720        AMD,EPYC,Milan,EDR,Beta           lscratch:890
batch          a4-3      idle        0/128/0/128   515720    515720        AMD,EPYC,Milan,EDR,Beta           lscratch:890
batch          a4-4      idle        0/128/0/128   515720    515720        AMD,EPYC,Milan,EDR,Beta           lscratch:890
batch          a4-5      mixed       4/124/0/128   515720    454280        AMD,EPYC,Milan,EDR,Beta           lscratch:890
batch          a4-6      idle        0/128/0/128   515720    515720        AMD,EPYC,Milan,EDR,Beta           lscratch:890
batch          a4-7      idle        0/128/0/128   515720    515720        AMD,EPYC,Milan,EDR,Beta           lscratch:890
batch          a4-8      idle        0/128/0/128   515720    515720        AMD,EPYC,Milan,EDR,Beta           lscratch:890
batch          a4-9      idle        0/128/0/128   515720    515720        AMD,EPYC,Milan,EDR,Beta           lscratch:890
batch          a4-10     idle        0/128/0/128   515720    515720        AMD,EPYC,Milan,EDR,Beta           lscratch:890
batch          a4-11     idle        0/128/0/128   515720    515720        AMD,EPYC,Milan,EDR,Beta           lscratch:890
batch          a4-12     idle        0/128/0/128   515720    515720        AMD,EPYC,Milan,EDR,Beta           lscratch:890
batch          a4-14     idle        0/128/0/128   515720    515720        AMD,EPYC,Milan,EDR,Beta           lscratch:890
batch          a4-16     idle        0/128/0/128   515720    515720        AMD,EPYC,Milan,EDR,Beta           lscratch:890
batch          a4-17     idle        0/128/0/128   515720    515720        AMD,EPYC,Milan,EDR,Beta           lscratch:890
batch          a4-18     mixed       1/127/0/128   515720    208520        AMD,EPYC,Milan,EDR,Beta           lscratch:890
batch          a4-19     idle        0/128/0/128   515720    515720        AMD,EPYC,Milan,EDR,Beta           lscratch:890
batch          a4-20     idle        0/128/0/128   515720    515720        AMD,EPYC,Milan,EDR,Beta           lscratch:890
batch          a4-21     idle        0/128/0/128   515720    515720        AMD,EPYC,Milan,EDR,Beta           lscratch:890
batch          a4-22     idle        0/128/0/128   515720    515720        AMD,EPYC,Milan,EDR,Beta           lscratch:890
batch          a4-23     idle        0/128/0/128   515720    515720        AMD,EPYC,Milan,EDR,Beta           lscratch:890
batch          a4-24     idle        0/128/0/128   515720    515720        AMD,EPYC,Milan,EDR,Beta           lscratch:890
batch          ra6-1     mixed       20/108/0/128  515640    331320        AMD,EPYC,Milan,EDR,Beta           lscratch:440
batch          ra6-2     mixed       20/108/0/128  515640    331320        AMD,EPYC,Milan,EDR,Beta           lscratch:440

Total 24 node(s) is/are qualified, with the minimum 200 GB (204800.000000 MB) of UNALLOCMEM and 100 CPU cores.
```

FAQ2 - How can I make my job run more efficiently

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

- Understand the software that you run in your job
- Understand the data that you use in your job
- Understand the computing platform where you run your job

Understand the software that you run in your job

- Understand PARALLEL CAPABILITY of the software that you run in your job
  - Can it run in a *shared-memory job* using multiple threads in parallel?
  - Can it run in a *distributed-memory job* using multiple MPI processes in parallel, using multiple compute nodes?
  - Have you read the documentation of the software provided by the developers for any advice on the computing resources the software can utilize to run in parallel tasks?
  - 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!

Understand the software that you run in your job

- Understand PARALLEL SCALABILITY if the software is a parallel capable software
  - Don’t start with too many cores, unless you already know that the application scales well. Do resource scaling-up by yourself.
  - 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 (can be approximately tested 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.

Understand the data that you use in your job

- How big is your data? What is the format of your data?
- Do you know how the memory requirements of your computational job increase as the input data gets larger?
- Data parallelism - Can your data be regrouped such that they can be used and run in parallel in elements of an array job?

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

Understand the computing platform where you run your job

- Sapelo2 computing platform: https://wiki.gacrc.uga.edu/wiki/Systems#Sapelo2
- Which processor type is newer on Sapelo2?
  - ✓ AMD EPYC Milan: Zen 3, released 2020
  - ✓ AMD EPYC Rome: Zen 2, released 2019
  - ✓ AMD EPYC Naples: Zen 1, released 2017
  - ✓ Intel Skylake: Xeon Gold, release 2017

  https://en.wikipedia.org/wiki/List_of_AMD_processors
  https://en.wikipedia.org/wiki/List_of_Intel_processors
- What processor type is the best to run your software (--constraint)? Is your software optimized for a specific type of processor?
- Using srun is more preferable than using mpirun/mpiexec to run a MPI job on Sapelo2.

Understand the computing platform where you run your job

- Can you use local scratch to improve the IO performance of your job?
  - Each compute node has a file system called /lscratch on local SSD drives that users can utilize as temporary storage*; hugemem_p* nodes and *A100 GPU nodes* have the fastest *nVME SSD*.
  - The /lscratch is very fast compared to the network file system /scratch; but its capacity is low and it cannot be accessed from outside the compute node.
  - Single-node jobs (serial single-core or shared-memory) that need to perform a lot disk IO can benefit from running from /lscratch; In general, MPI jobs cannot use /lscratch.
  - Clean up /lscratch by moving data back to /scratch, before your job exits from the node.
  - Reported in the last GRES column (Generic RESources) by sinfo-gacrc or sinfo-nodes.
  - https://wiki.gacrc.uga.edu/wiki/Running_Jobs_on_Sapelo2#How_to_run_a_job_using_the_local_scratch_.2Flscratch_on_a_compute_node

FAQ3 - Why is my job pending?
https://wiki.gacrc.uga.edu/wiki/Frequently_Asked_Questions#Why_is_my_job_pending.3F

FAQ4 - How can I get my job to start sooner?
https://wiki.gacrc.uga.edu/wiki/Job_Resource_Tuning#Time_to_Job_Start

GACRC Support http://help.gacrc.uga.edu