HPCC New User Training
Getting Started on HPCC Resources

Misha Ahmadian
High Performance Computing Center

Summer 2021
Outline

- Introduction to High Performance Computing
- HPCC Resources
- Logging and using the Clusters
- Resource Allocation and Job Submission with SLURM
- Software builds and installation
- Transferring Data
- HPCC Policies
- Getting Help
Introduction to High Performance Computing
Simplest Programming Model: Serial Computing

- Software is often written and optimized to run **serially**.
  - Execution occurs on a single computer using a single CPU core.
  - A problem is broken down into a linear series of instructions.
  - Instructions are executed one after another.
  - Only one instruction may execute at any given time.
  - While simple, this model may not make full use of modern multi-core processors.
More Advanced Usage: Parallel Computing

- **Parallel Computing** is the simultaneous use of multiple compute resources to solve a computational problem.
  - Execution occurs across multiple CPU cores.
  - A problem is broken into discrete parts that can be solved concurrently.
  - Each part is further broken down into a series of instructions, executed one after another.
  - Instructions from each part execute simultaneously on different CPU cores.
Classes of Parallel Computers

• Multi-core Computing:
  • Multi-core processors contain multiple ‘processing units’ (called cores) on a single chip.
  • Allows for parallel execution across cores – each able to reach the same system resources (RAM, Keyboard, Monitor, etc...).

• Symmetric Multiprocessor (SMP):
  • A symmetric multiprocessor is a computer system with multiple identical processors.
  • Each processor likely has multiple cores
  • Allows for parallel execution across cores – each able to reach the same system resources (RAM, Keyboard, Monitor, etc...).
Classes of Parallel Computers

• Clusters:
  • Groups of loosely coupled computers working together closely.
  • Processes can be spread across multiple nodes, but processes are unable to reach the same system resources (RAM, Keyboard, Monitor, etc...).

• Massively Parallel Processors (MPP):
  • A group of tightly coupled computers working together closely across a specialized high-speed interconnect.
  • Processes can be spread across multiple nodes, but processes are unable to reach the same system resources (RAM, Keyboard, Monitor, etc...).
  • Common computing paradigm for campus-based HPC centers.
Classes of Parallel Computers

- **National-Scale Supercomputers:**
  - Highly scaled forms of parallel computing run for the National Science Foundation, Department of energy, National Institutes of health, etc.
  - Organized much like larger versions of the TTU HPCC clusters.
  - Awards for time require proposals and are evaluated on a competitive basis.

- **Grid and Cloud Computing:**
  - Highly distributed forms of parallel computing.
  - Clusters or single resources are spread across multiple sites using the Internet for connectivity.
  - Commercial clouds are often more expensive to use than dedicated fully-utilized on-premises clusters for HPC.
Why does programming style matter?

• Different problems are suited to each of the major programming models.
  
  • **Serial programming:**
    ▪ Executes serially using a single core / thread
    ▪ “Single core machines”
    ▪ Good for problems that don't require inter-thread or inter-process communication.
  
  • **Multi-core / Multi-threaded Programming:**
    ▪ Executes in parallel using multiple cores / threads
    ▪ All threads are running on the same machine and access the same RAM
    ▪ “Multicore & Symmetric Multiprocessing”
    ▪ Needed for problems that require different threads or processes to share information.
  
  • **Massively Parallel / Distributed Programming:**
    ▪ Executes in parallel using multiple machines
    ▪ “Clusters, Massive Parallel Processors, & Grid/Cloud”
    ▪ Needed for problems that require harnessing multiple large-scale resources.
Match Your Programming Style To The Problem!

• If your program is not written to use a certain model, it will not necessarily “just work” in that model.
  • Running serial code on a 128-core machine will use 1 core and leave 127 cores sitting idle.
  • Attempting to run multi-threaded code written to be used in just one node across 10 nodes instead without adjustments will result in 1 node being overutilized and 9 nodes sitting idle.

• Not all multi-threading/MPP is equal!
  • Try to understand how your program works at a small scale before attempting to “scale up”.
  • Keep in mind that programming language, developer decisions and even user input data can greatly alter how well an application scales.
  • Many existing codes need to be tuned or configured to run optimally.
HPCC Resources
HPCC Resources: RedRaider Primary Cluster

• **RedRaider** Cluster Commissioned in 2021:

  • **Nocona partition** (240 CPU nodes)
    • 30,720 Cores (128 cores/node)
    • 120 TB total RAM (512 GB/node)
    • 2x AMD EPYC ROME 7702 processors / node
    • HDR 200 Gbps InfiniBand fabric
    • 804 Teraflops (81.4% efficiency)

  • **Matador partition** (20 GPU nodes)
    • 40 NVIDIA Tesla V100 GPUs (2 V100 / node)
    • 7.5 TB total RAM (384 GB/node)
    • 800 CPU Cores (40 cores/node)
    • 2x Intel Xeon Cascade lake 6248 processors / node
    • HDR 100 Gbps InfiniBand fabric
    • 226 Teraflops (80.6% Efficiency)
HPCC Resources: RedRaider Primary Cluster

- **Toreador partition** (11 nodes)
  - 33 NVIDIA Tesla A100 GPUs (3 A100 / node)
  - 2.1 TB total RAM (192 GB/node)
  - 176 CPU Cores (16 cores/node)
  - 2x AMD EPYC ROAM processors / node
  - HDR 100 Gbps InfiniBand fabric
  - *May expect higher waiting time*
HPCC Resources

- **Quanah partition** (467 nodes)
  - Commissioned in 2017
  - 2x Intel Xeon E5-2695v4 Broadwell Processors/node
  - 16,812 total cores (36 cores/node)
  - 87.56 TB total RAM (192 GB/node)
  - Non-blocking Omni-Path (100 Gbps) fabric
  - Benchmarked at 485 Teraflops
Previous Clusters:

- **Ivy** (100 nodes)
  - Commissioned in 2014
  - 2000 total Cores (20 cores/node)
  - 6.25 TB Total RAM (64 GB/node)
  - 2x Xeon E5-2670v2 **Ivy Bridge** Processors/node
  - QDR 40 GB/second InfiniBand fabric
  - Plan to convert this to Open OnDemand soon

- **Community Cluster**
  - For individual research group resources
  - Replaced by buy-in options for RedRaider

- **Serial nodes**
  - Not currently commissioned; undergoing rework
  - Intended for future serial job and grid use
Cluster-Wide Storage System:

- 6.9 PB of total storage space
- 200 Gbps HDR Fabric
- Based on Lustre parallel file system
- Quota/Backup/Purge policies per storage area (free to all accounts):

<table>
<thead>
<tr>
<th>Area</th>
<th>Quota</th>
<th>Backup</th>
<th>Purge</th>
</tr>
</thead>
<tbody>
<tr>
<td>/home/&lt;eraider&gt;</td>
<td>300 GB</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>/lustre/work/&lt;eraider&gt;</td>
<td>700 GB</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>/lustre/scratch/&lt;eraider&gt;</td>
<td>None</td>
<td>No</td>
<td>Monthly</td>
</tr>
</tbody>
</table>

- Researchers/groups may purchase additional dedicated storage space:
  - With Backup: $80/TB/Year
  - Without Backup: $40/TB/Year
<table>
<thead>
<tr>
<th>HPCC Software Environment</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Ivy*/CC</th>
<th>Quanah</th>
<th>RedRaider</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operating System</strong></td>
<td>CentOS 7.4</td>
<td>CentOS 7.4 **</td>
<td>CentOS 8.1</td>
</tr>
<tr>
<td><strong>Job Resource Manager</strong></td>
<td>Slurm 20.11.7</td>
<td>Slurm 20.11.7</td>
<td>Slurm 20.11.7</td>
</tr>
<tr>
<td><strong>Package Build Env</strong></td>
<td>RPM Build</td>
<td>RPM Build</td>
<td>Spack v0.15</td>
</tr>
<tr>
<td><strong>Software Deployment Env</strong></td>
<td>Lmod 7.7.14</td>
<td>Lmod 7.7.14</td>
<td>Lmod 8.2.10</td>
</tr>
<tr>
<td><strong>Available C/C++/Fortran /MPI Compilers</strong></td>
<td>GCC 4.8.5 (Default)</td>
<td>GCC 4.8.5 (Default)</td>
<td>GCC 8.3.1 (Default)</td>
</tr>
<tr>
<td></td>
<td>GCC 5.4.0</td>
<td>GCC 5.4.0</td>
<td>GCC 9.2.0</td>
</tr>
<tr>
<td></td>
<td>GCC 7.3.0</td>
<td>GCC 7.3.0</td>
<td>GCC 10.1.0 (Recommended)</td>
</tr>
<tr>
<td></td>
<td>Intel 18.0.3.222</td>
<td>Intel 18.0.3.222</td>
<td>Intel compiler/MPI 2019</td>
</tr>
<tr>
<td></td>
<td>impi 2018.3.222</td>
<td>impi 2018.3.222</td>
<td>AOCC/AOCL (Coming Soon)</td>
</tr>
<tr>
<td></td>
<td>OpenMPI 1.10.[6-7]</td>
<td>OpenMPI 1.10.[6-7]</td>
<td>OpenMPI 3.1.6, 4.0.4</td>
</tr>
<tr>
<td></td>
<td>MVAPICH 2.2</td>
<td>MVAPICH 2.2</td>
<td>MVAPICH (Coming Soon)</td>
</tr>
<tr>
<td><strong>GPU Libraries</strong></td>
<td>N/A</td>
<td>N/A</td>
<td>CUDA 11.0 (default)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Cudnn 8.0.1 (default)</td>
</tr>
</tbody>
</table>

* To be devoted to Open OnDemand  ** 10 nodes are upgraded to CentOS 8 already
Logging and Using the Cluster
Getting Started

• Account Request:
  • Faculty/Staff account
  • Student account
  • Research Partner account
  • http://www.depts.ttu.edu/hpcc/accounts/index.php

• User Guides:
  • http://www.depts.ttu.edu/hpcc/userguides/index.php

• More details about HPCC equipment:
  • http://www.depts.ttu.edu/hpcc/operations/equipment.php
Getting Started

• Logging to HPCC Resources:
  • http://tinyurl.com/ttu-hpcc-login

• On or Off Campus?
  • On Campus: Wired TTU network & TTUnet wireless network
  • Off Campus: Any other network connection, including
    • TTUHSC networks
    • TTUguest wireless network

• Logging in from Off Campus:
  • Log in via the SSH gateway
  • Establish a VPN - https://goo.gl/4LbuWG
  • Neither system is owned or maintained by HPCC
Logging to RedRaider Cluster

- **Mac/Linux Users:**
  - **SSH** (Secure Shell): Freely available on Linux/Unix/MacOS and used via the Terminal.
    - `ssh eraider@login.hpcc.ttu.edu`
    - `ssh eraider@quanah.hpcc.ttu.edu`

- **Windows Users:**
  - **MobaXterm** (Recommended): [https://mobaxterm.mobatek.net](https://mobaxterm.mobatek.net)
  - **Putty**: [https://www.putty.org](https://www.putty.org)

- **Once logged in:**
  - RedRaider has two login nodes: *(login-20-25, login-20-26)*
  - The load-balancer lands your SSH session on one of these nodes.
  - Quanah login node currently still available, login.hpcc.ttu.edu preferred.
XWindows

- **Interactive GUI using Linux/Mac.**
  - Mac users must install [XQuartz](https://www.xquartz.org). Linux Users can use the Terminal.
  - Logging to the cluster using ”-Y -X” with your normal ssh command:
    - `ssh -Y -X eraider@login.hpcc.ttu.edu`
  - Run a test command like `xclock`.

- **Interactive GUI using Windows.**
  - Install MobaXterm. (Consult MobaXTerm web site for details.)
  - Open a new tab in MobaXterm.
  - Logging to the cluster using ”-Y -X” with your normal ssh command:
    - `ssh -Y -X eraider@login.hpcc.ttu.edu`
  - Run a test command like `xclock`. 
Logging In

Upcoming or current downtimes

Upcoming HPCC Training Sessions

Last updated time
Environment Settings

• **Lmod Modules:**
  • The primary way to change your user environment.
  
  • **Module commands:**
  • module avail
  • module list
  • module load <module_name>
  • module unload <module_name>
  • module spider <keyword>
  • module purge

• **User Guid:**
  • [https://www.depts.ttu.edu/hpcc/userguides/general_guides/software_environment.php](https://www.depts.ttu.edu/hpcc/userguides/general_guides/software_environment.php)
Modules Tips and Recommendations

• Try to keep all your module load commands as part of your job submission scripts instead of adding them to your `.bashrc` file.
  • Makes debugging and changing between experiments and cluster partitions easier.
  • Prevents collisions or accidently running jobs with the wrong environment.

• Please note that Quanah (Intel nodes), Nocona (AMD nodes), Matador (V100 GPU nodes), and Toreador (A100 GPU nodes) have different set of modules.
  • More details about this later in this training.

• Always include the version number of a module in the module load command.
  • Makes version tracking easier.
  • Prevents unanticipated changes in version during an experiment.
  • Example: Use `module load nwchem/6.6-intel` instead of just `module load nwchem`
Resource Allocation and Job Submission with
Simple Linux Utility for Resource Management (SLURM):

- Primary HPCC scheduler used in RedRaider
- **Main entities:**
  1. **Nodes:** Physical computing resources
  2. **Partition:** A logical set of nodes
  3. **Jobs:** Allocations of resources assigned to a user for a specified amount of time
  4. **Job Steps:** sets of (possibly parallel) tasks within a job
  5. **Tasks:** Implies the requested/allocated computing resources to process(es) per job or job step
     (By default, each task refers to 1 CPU core)
• **Useful Slurm Commands:**
  
  - **sinfo:**
    - View information about **nodes** and **partitions**.
  
  - **squeue:**
    - View information about **jobs** located in partitions.
    - Useful options to filter the output:
      - `-u <user>, --user=<user>`: Shows the list of jobs or job steps that belong to a specific user
      - `-p <partition>, --partition=<partition>`: Filters the jobs within a partition.
  
  - **srun:**
    - submits a job for execution or initiates job steps in real time.
    - **srun** has the same options as **sbatch** with a few more. (Please see the man page)
    - **srun** works similar to the “mpirun” and it can be replaced with “mpirun” as well.
Job Submission in Slurm

- **sbatch**: submits a job script for later execution.
  - The submitted job stays in the queue until the requested resources become available.
  - The job submission script is a text file that contains “#SBATCH” hints with `sbatch` command line options.

```bash
#!/bin/bash
#SBATCH –J MPI_test
#SBATCH –N 2
#SBATCH –ntasks-per-node=128
#SBATCH –o %x.%j.out
#SBATCH –e %x.%j.err
#SBATCH –p nocona

module load gcc/10.1.0 openmpi/3.1.6
mpirun ./my_mpi
```
### Job Submission in Slurm

#### Job Submission Script Layout:

<table>
<thead>
<tr>
<th>Description</th>
<th>SLURM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set the name for job</td>
<td><code>-J, --job-name=&lt;jobname&gt;</code></td>
</tr>
<tr>
<td>The name of the standard output file</td>
<td><code>-o, --output=&lt;filename pattern&gt;</code></td>
</tr>
<tr>
<td>The name of the standard error file</td>
<td><code>-e, --error=&lt;filename pattern&gt;</code></td>
</tr>
<tr>
<td>Define the queue (partition) name</td>
<td><code>-p, --partition=&lt;partition_names&gt;</code></td>
</tr>
<tr>
<td>Type of parallel env for job/task allocation</td>
<td><code>-N, --nodes=&lt;# of nodes&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>--ntasks-per-node=&lt;ntasks&gt;</code></td>
</tr>
<tr>
<td>Reserve memory per slot</td>
<td>`--mem-per-cpu=&lt;size[K</td>
</tr>
<tr>
<td>Set the maximum job run time</td>
<td><code>-t, --time=&lt;HH:MM:SS&gt;</code></td>
</tr>
<tr>
<td>Specify the cluster policy for this job</td>
<td>`-A, --account=&lt;account&gt;</td>
</tr>
</tbody>
</table>
• Select a partition:
  • Partition in Slurm groups physical nodes into a logical set and allows jobs to request for nodes’ resources from that partitions.
    • `-p`, `--partition=<partition_name>`

<table>
<thead>
<tr>
<th>Name</th>
<th># of Nodes</th>
<th>Type</th>
<th>Nodes</th>
<th>#Core/Node</th>
<th>#Mem/Node</th>
<th>#Mem/Core</th>
<th>#GPU/node</th>
</tr>
</thead>
<tbody>
<tr>
<td>nocona</td>
<td>240</td>
<td>AMD ROME CPU</td>
<td>cpu-[23-26]-[1-60]</td>
<td>128</td>
<td>503 GB</td>
<td>3.9 GB</td>
<td>N/A</td>
</tr>
<tr>
<td>matador</td>
<td>20</td>
<td>Intel/Nvidia V100 GPU</td>
<td>gpu-[20-21]-[1-10]</td>
<td>40</td>
<td>376 GB</td>
<td>9.4 GB</td>
<td>2</td>
</tr>
<tr>
<td>gpu-build</td>
<td>1</td>
<td>Intel/Nvidia V100 GPU</td>
<td>gpu-20-11</td>
<td>32</td>
<td>187 GB</td>
<td>5.9 GB</td>
<td>1</td>
</tr>
<tr>
<td>Toreador</td>
<td>11</td>
<td>AMD/Nvidia A100</td>
<td>gpu-20-[12-15],gpu-21-[11-17]</td>
<td>16</td>
<td>188 GB</td>
<td>11.8 GB</td>
<td>3</td>
</tr>
<tr>
<td>quanah</td>
<td>467</td>
<td>Intel Xeon Broadwell</td>
<td>cpu-[1-10]-[*]</td>
<td>36</td>
<td>188 GB</td>
<td>5.3 GB</td>
<td>N/A</td>
</tr>
<tr>
<td>community clusters</td>
<td>18</td>
<td>Various</td>
<td>cpu-17-[49-52,57-62],cpu-18-[49-56]</td>
<td>Various</td>
<td>Various</td>
<td>Various</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Job Submission in Slurm

• Requesting CPU:

  • In Slurm, users must define the following options in their job submissions in order to request for CPU resources:

    1. Number of nodes: How many total nodes for the job?
       - `-N, --nodes=<number of nodes>`

    2. Number of tasks per node: *(Recommended)* (By default, each task consumes 1x CPU core)
       - `--ntasks-per-nodes=<number of task per node>`

       OR Number of total tasks: How many task across the nodes?
       - `-n, --ntasks=<number of tasks>`

• Requesting the right number of cores is key to optimizing throughput
Requesting Memory:

One can specify the size of the consumable Memory in two ways in Slurm:

1. Memory per core (*Recommended*):
   - `--mem-per-cpu=<size[M|G]>`

2. Memory per node:
   - `--mem=<size[M|G]>`

If no memory size was specified, Slurm will assign the default memory per core to your job.

- **Nocona**: 4027 MB (3.9 GB) per core  
- **Matador**: 9639 MB (9.4 GB) per core  
- **Quanah**: 5370 MB (5.3 GB) per core

Make sure you won’t exceed the total memory per node:

- `--mem-per-cpu=100G`
Job Submission in Slurm

• Requesting Runtime limits:
  • Recommended that you set the max runtime you expect a job will take.
    • `-t, --time=<time>`
    • `<time>` can be:
      • `minute`
      • `minute:seconds`
      • `hours:minutes:seconds`
      • `days-hours`
      • `days-hours:minutes`
      • `days-hours:minutes:seconds`
    • E.g., `--time=24:00:00`
  • Please note that there is a 48-hour default time limit per job and exceeding this amount will end up with rejecting your job submission.
Job Submission in Slurm

- **Requesting GPU:**
  - GPUs are available by requesting any node in the `matador` partition.
    - Number of GPUs per node *(Recommended)*:
      `-gpus-per-node=[<type>:]<number>`
    - Total number of GPUs:
      `-G, --gpus=<# of gpus>`
  - There is only one type of GPU in RedRaider Cluster (`v100`) and is optional to be specified.
  - It is **required** to requesting at least one **GPU per node** when submitting a job to Matador.
  - Make sure you do not exceed more than 2 GPUs per node during the job submission.
    - `--partition=matador --nodes=2 --gpus=6` ✗
    - `--partition=matador --nodes=2 --gpu-per-node=2` ✓
Job Submission in Slurm

- Accounts/QoS on RedRaider Cluster:
  - Accounts, in Slurm, assigns the usage/fair-share policies to each job.
    - `-A, --account=<account>`
  - QoS in Slurm defines a set of pre-defined resource limits based on the selected account.
    - `-q, --qos=<QoS>`

<table>
<thead>
<tr>
<th>Account</th>
<th>QoS</th>
<th>Default Runtime</th>
<th>Maximum Runtime</th>
<th>Total CPU/Mem Limits</th>
<th>CPU/Mem Limit per job</th>
<th>Total # Jobs Per User</th>
<th>Allowed Partitions</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>default *</td>
<td>normal</td>
<td>48 hours</td>
<td>48 hours</td>
<td>No limit</td>
<td>No limit</td>
<td>2000</td>
<td>All Partitions</td>
<td>normal</td>
</tr>
<tr>
<td>xlquanah</td>
<td></td>
<td>72 hours</td>
<td>120 hours</td>
<td>144 cores / 755GB</td>
<td>36 cores / 188GB</td>
<td>2000</td>
<td>quanah</td>
<td>normal</td>
</tr>
<tr>
<td>Dedicated</td>
<td></td>
<td>72 hours</td>
<td>No limit</td>
<td>Varies based on the purchased resources</td>
<td>Up to the total available resources</td>
<td>No limit</td>
<td>Nocona/Quanah</td>
<td>high</td>
</tr>
<tr>
<td>resource users</td>
<td>--*</td>
<td>72 hours</td>
<td>No limit</td>
<td>Varies based on the purchased resources</td>
<td>Up to the total available resources</td>
<td>No limit</td>
<td>Nocona/Quanah</td>
<td>high</td>
</tr>
</tbody>
</table>

(*) System will assign the default Account/QoS if user does not define them in their job submissions.
Submitting Jobs

Example of a simple job to submit an MPI program to Slurm:

- Create a job submission script file (e.g., submit.sh):

```bash
#!/bin/bash
#SBATCH –J MPI_test
#SBATCH –N 2
#SBATCH –ntasks-per-node=128
#SBATCH –o %x.%j.out
#SBATCH –e %x.%j.err
#SBATCH –p nocona

module load gcc/10.1.0 openmpi/3.1.6
mpirun ./my_mpi
```

- Submit the job with `sbatch`:
  - `sbatch submit.sh`

- Monitor the job with `squeue`:
  - `squeue -u <username>`

- Cancel the job with `scancel`:
  - `scancel job_id`
Interactive Session

- **interactive:**
  - Starts an interactive session/job:
    - `interactive -c 2 -p nocona`
    - See the `interactive -h` for all the available options.
  - Make sure the prompt changes to `cpu-#-#`.
  - Make sure you run “exit” when you’re finished.
  - Keep in mind resource/runtime limits apply to `interactive` based on the selected account.
  - The `interactive` command will forward the X11 if the SSH session was established with `-X` or `-Y`.
  - Please note that direct SSH to any worker nodes not part of your job is blocked on the RedRaider cluster.
The ‘gpu-build’ Partition

• Building and Testing GPU applications:
  • The `gpu-build` partition contains one Intel/GPU node with 1x Nvidia V100 GPU device, 32x Intel CPU cores and 192 GB RAM, which allows users to:
    • Build their own GPU applications.
    • Test GPU applications and the environment setup before submitting a job to Matador partition.
    • Accessing the Lmod Module environment for GPU compilers/applications.
  
  • In order to access the ‘gpu-build’ node, you need to establish an interactive session:
    • `$ interactive -p gpu-build -c 2`

• Limitations:

<table>
<thead>
<tr>
<th>Partition</th>
<th>Max Runtime (per job)</th>
<th>Max CPU per user (in total)</th>
<th>Max Mem per user (in total)</th>
<th>Max interactive session per user</th>
</tr>
</thead>
<tbody>
<tr>
<td>gpu-build</td>
<td>5 hours</td>
<td>6</td>
<td>36006 MB (35 GB)</td>
<td>2</td>
</tr>
</tbody>
</table>
Debugging a Finished Job

• **sacct:**
  - reports accounting information about active or completed jobs or job steps.
    - `sacct -j <jobid>`
  - More filter options are available by checking the `-e, --helpformat` options of `sacct` command.
    - `sacct -j <jobid> --format=partition,jobid,ntasks,nodelist,maxrss,maxvmsize,exitcode`
  - When debugging:
    - Check the output and error files
    - Check the output of `sacct` for:
      - Memory usage
      - Exit code
      - Start and end time.
Software builds and installation
HPCC RedRaider Cluster – Overall Look

Login-20-25
Login-20-26
quanah

Slurm
- sinfo
- squeue
- sbatch
- scancel
- sacct
- interactive

nocona modules
quanah modules
matador modules
Toreador modules
• Multiple partitions – Multiple architectures:

**Nocona**
AMD EPYC ROME

**Matador**
Intel Xeon Cascade Lake
Nvidia V100

**Quanah**
Intel Xeon Broadwell

**Toreador**
AMD EPYC ROME
Nvidia A100
Software builds on HPCC Clusters

- Modules & compiled code are different on each of the RedRaider partitions!
  - Each CPU architecture may bring a different set of features and instructions.
  - Compiled programs (C/C++/Fortran) need to be re-compiled to match each CPU architecture.
  - E.g., programs that are compiled on Intel nodes may not work properly/efficiently on AMD nodes.
  - Different Compilers/Math libraries optimize the programs differently on each architecture.

<table>
<thead>
<tr>
<th>Compiler</th>
<th>AMD ROME</th>
<th>Intel Broadwell</th>
<th>Intel Ivy Bridge</th>
<th>Intel Cascade Lake</th>
<th>Nvidia GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>GNU/GCC</td>
<td>GCC 10+</td>
<td>GCC 4+</td>
<td>GCC 4+</td>
<td>GCC 10+</td>
<td>GCC 8+</td>
</tr>
<tr>
<td>Intel</td>
<td>May work</td>
<td>Optimized</td>
<td>Optimized</td>
<td>Optimized</td>
<td>Intel 19+</td>
</tr>
<tr>
<td>AOCC</td>
<td>Optimized</td>
<td>Not Applicable</td>
<td>Not Applicable</td>
<td>Not Applicable</td>
<td>N/A</td>
</tr>
<tr>
<td>MKL</td>
<td>May work</td>
<td>Optimized</td>
<td>Optimized</td>
<td>Optimized</td>
<td>MKL 19+</td>
</tr>
<tr>
<td>AOCL</td>
<td>Optimized</td>
<td>Not Applicable</td>
<td>Not Applicable</td>
<td>Not Applicable</td>
<td>N/A</td>
</tr>
<tr>
<td>CUDA</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>CUDA 10+</td>
</tr>
</tbody>
</table>
Software builds on HPCC Clusters

• Tips and Recommendations:

1. Create a separate directory for each CPU architecture, and make a copy from your code/program and place it under each directory:
   - mkdir nocona matador quanah

2. Login to the RedRaider login node, and for each CPU architecture make an interactive session to the corresponding worker node:
   - interactive -p nocona –c 10

3. Go to the directory of you code that has the same name as the current session’s partition:
   - cd nocona

4. Load a proper compiler module and recompile your code:
   - module load gcc/10.1.0

5. If applicable, add the -O3 optimization flag to all the CFLAGS, CPPFLAGS, CXXFLAGS, FFLAGS.
   - CFLAGS=-O3 FFLAGS=-O3 make -j 10 all
Software builds on HPCC Clusters

Tips and Recommendations:

5. We recommend mapping the MPI jobs to the L3-cache memory on **Nocona (AMD)** nodes:
   - `mpirun -map-by l3cache -bind-to core ./mpi_app`

6. **HPCC will not support Python v2 on Nocona and Matador nodes with CentOS 8.**
   (This rule will be applied to Quanah and Ivy in the near future.)
   - Users can still get Python v2 from Conda (Anaconda/Miniconda)
   - Python 2 is NOT RECEIVING SECURITY UPDATES and should be retired from your workflows ASAP.

7. Python applications (including the applications from Condo repo) will continue working with different architectures without recompiling them.
Local Python Package Installation

- Install a Python package into your home folder:
  
  $ module load intel python
  $ pip install --user <package name>
  
  - Example: pip install --user matplotlib

- Install a local copy of Python using Conda:
  
  $ /lustre/work/examples/InstallPython.sh
  $ . $HOME/conda/etc/profile.d/conda.sh
  $ conda activate
  $ conda install <package name>
  
  - Example: conda install biopython
Local R Package Installation

- Install an R package into your home folder:

  - Example (On Quanah Node:)
    
    $ module load intel R
    $ R
    $ install.packages('<package name>')</n    
    Example: install.packages('readr')

  - Select a mirror

  - The R application will ask if you want to install it locally the first time you do this.
Transferring Data
Transferring Data (Using Globus)

• Whenever possible, refrain from using:
  • `scp`, `sftp`, `rsync`, or any other data transfer tool
  • Okay for very small transfers, but for large ones, see below.

• Transfer files using Globus Connect personal client:
  • Globus Connect service is well connected to the campus network.
  • Globus Connect service eliminates the data transfer load from the cluster login nodes.
  • Globus Connect personal client works on Linux, Mac and Windows and is easy to control through a simple web GUI.
  • Numerous other sites (including TACC) support Globus Connect data transfers.
HPCC Policies
HPCC Policies

• Login nodes (login.hpcc.ttu.edu, quanah.hpcc.ttu.edu):
  • No jobs are allowed to run on the login node.

• SSH Access:
  • No direct SSH access allowed to a node(s) if you have no job running on the node(s)

• Software Installation:
  • Software requests are handled on a case-by-case basis
  • Requesting software does not guarantee it will be installed “cluster-wide”.
  • May take two or more weeks to complete your request.

• Scratch Purge Policy:
  • Scratch will be purged monthly by removing of all files not accessed within the past year, or sufficient to bring total scratch space usage across all accounts down to 80% of capacity.
Getting Help
Further Assistance

• Visit Our Website:
  • Most user guides have been updated
  • New user guides are being added
  • https://www.depts.ttu.edu/hpcc/userguides/index.php

• Read the documentation!
  • https://slurm.schedmd.com/documentation.html

• Submit a support ticket:
  • Send an email to hpccsupport@ttu.edu