Things to Do While You are Waiting

● Open your web browser and visit hprc.tamu.edu
● Log into TAMU VPN (if you’re off campus) and reconnect to Zoom
● If you don’t have an HPRC account, please ask*
● If you don’t know basic Linux commands, please ask*

*speak up in chat or email help@hprc.tamu.edu
Introduction to High Performance Research Computing

Presented by Richard Lawrence
Fall 2022
Outline

- Usage Policies
- References
- Cluster Overview
- Break
- Accessing HPRC
- HPRC Computing Environment
- Break
- Cluster Computing Basics
- Break
- Cluster Computing Exercises
- Need Help?
Usage Policies
(Be a good compute citizen)

- It is illegal to share computer passwords and accounts by state law and university regulation
- It is prohibited to use HPRC clusters in any manner that violates the United States export control laws and regulations, EAR & ITAR
- Abide by the expressed or implied restrictions in using commercial software

hprc.tamu.edu/policies
Education Resources

- Knowledge Foundation:
  - Basic knowledge of LINUX commands
  - Slides from our LINUX short course are at: hprc.tamu.edu/training/intro_linux.html
  - Watch the relevant Introduction and Primer videos on our Youtube Channel youtube.com channel "Texas A&M HPRC"
  - Answers to frequently asked questions can be found on our Wiki https://hprc.tamu.edu/wiki/Main_Page
Follow Along

Simple exercises:
- Terra: [hprc.tamu.edu/wiki/Terra:Exercises](http://hprc.tamu.edu/wiki/Terra:Exercises)

Cluster computing exercises:
- Terra: Example files located in `/scratch/training/Intro-to-terra` directory

Interface for hands-on exercises:
- Terra: [portal-terra.hprc.tamu.edu/](http://portal-terra.hprc.tamu.edu/) or choose “Terra OnDemand” from [portal.hprc.tamu.edu/](http://portal.hprc.tamu.edu/)
HPRC Clusters

Terra deployed in 2017

Grace Flagship cluster, deployed in 2021

FASTER In early access, full deployment coming soon
Clusters Are For You!

What kinds of problems are solved by cluster computing?

- Problems that are too big to fit in one laptop or workstation, due to limitation on memory, core count, or node count
- Problems that scale well with more CPU cores or memory
- Single-threaded problems with millions of permutations
- Problems that require large high speed storage and/or interconnect
Cluster Diagram

Login node 1
Login node 2
Login node 3

Job Scheduler
Our clusters use Slurm

Compute node 1
Compute node 2
... hundreds of compute nodes
Compute node with GPU 1
Big memory compute node 1
... etc.

Shared File Storage
/scratch
/home

... etc.
# HPRC Clusters

<table>
<thead>
<tr>
<th>Total Nodes (Cores)</th>
<th>Terra</th>
<th>Grace</th>
<th>FASTER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>307 (8,512)</td>
<td>925 (44,656)</td>
<td>180 (11,520)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>General Nodes</th>
<th>28 cores 64GB</th>
<th>48 cores 384GB</th>
<th>64 cores 256GB</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Features</th>
<th>GPUs Many-core nodes</th>
<th>GPUs - multiprecision Big Memory Nodes</th>
<th>Composable GPUs Composable Memory</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Job Scheduler</th>
<th>Slurm</th>
<th>Slurm</th>
<th>Slurm</th>
</tr>
</thead>
</table>

| Online Since        | 2017  | 2021  | Coming Soon |

[hprc.tamu.edu/resources](http://hprc.tamu.edu/resources)
HPRC Wiki - Hardware

Visit our wiki [https://hprc.tamu.edu/wiki/Main_Page](https://hprc.tamu.edu/wiki/Main_Page) to learn more about our clusters.

For example, information about Terra hardware is on page [https://hprc.tamu.edu/wiki/Terra:Intro](https://hprc.tamu.edu/wiki/Terra:Intro).
Getting Started
Authentication and Access

Three steps to access HPRC resources.
1. Get an HPRC account
2. VPN to TAMU campus
3. Web login (Portal, Globus) through CAS or SSH/SFTP to HPRC clusters

- Duo NetID two-factor authentication used to enhance security ([it.tamu.edu/duo/](it.tamu.edu/duo/))
- (Faculty and staff) Use Duo Keys - ([u.tamu.edu/get_duo_keys](u.tamu.edu/get_duo_keys))
- Instructions in two-factor wiki page ([hprc.tamu.edu/wiki/Two_Factor](hprc.tamu.edu/wiki/Two_Factor))

Example: SSH login with Duo
$ ssh grace.tamu.edu
************************************************************
…. warning message (snipped) ……
************************************************************
Password:
Duo two-factor login for UserNetID
Enter a passcode or select one of the following options:
1. Duo Push to XXX-XXX-1234
2. Phone call to XXX-XXX-1234
3. SMS passcodes to XXX-XXX-1234 (next code starts with: 9)
Passcode or option (1-3): 1
Success. Logging you in...
### File Systems and User Directories

<table>
<thead>
<tr>
<th>Directory</th>
<th>Environment Variable</th>
<th>Space Limit</th>
<th>File Limit</th>
<th>Intended Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>/home/$USER</td>
<td>$HOME</td>
<td>10 GB</td>
<td>10,000</td>
<td>Small amounts of processing.</td>
</tr>
<tr>
<td>/scratch/user/$USER</td>
<td>$SCRATCH</td>
<td>1 TB</td>
<td>250,000</td>
<td>Temporary storage of actively used large files. Not a long-term storage area.</td>
</tr>
</tbody>
</table>
File Systems and User Directories

- **$HOME** and **$SCRATCH** are not shared between clusters.
- View usage and quota limits using the command: `showquota`
- Quota and file limit increases can be requested
- Group directory for sharing files upon request.
- Do not share your home or scratch directories.

<table>
<thead>
<tr>
<th>Directory</th>
<th>Environment Variable</th>
<th>Space Limit</th>
<th>File Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>/home/$USER</td>
<td>$HOME</td>
<td>10 GB</td>
<td>10,000</td>
</tr>
<tr>
<td>/scratch/user/$USER</td>
<td>$SCRATCH</td>
<td>1 TB</td>
<td>50,000</td>
</tr>
</tbody>
</table>

hprc.tamu.edu/wiki/Terra:Filesystems_and_Files
hprc.tamu.edu/wiki/Grace:Filesystems_and_Files
Hands-on exercises:

Activate TAMU VPN
Go to:
portal.hprc.tamu.edu
portal.hprc.tamu.edu

- **Files** > copy and edit files on the cluster’s filesystems
- **Jobs** > submit and monitor cluster jobs
- **Clusters** > open a shell terminal (command line) on a login node
- **Interactive Apps** > start graphical software on a compute node and connect to it
- **Dashboard** > view file quotas and computing account allocations
Hands-on exercise:

Upload a File to Grace in Portal

Menu > Files > /scratch/user/<netid>
use the ‘↑ Upload’ button near the top-right,
pick something small from your desktop
Hands-on exercise:

Check your file Quota on Grace in Portal

Menu > Dashboard > Terra Dashboard
Locate your /scratch disk usage stats
Software Infrastructure
Software

HPRC provides both pre-installed Software and installation assistance

● Software wiki page includes instructions and examples
  – hprc.tamu.edu/wiki/SW

● License-restricted software
  – Contact license owner for approval

● Contact us for software installation help/request
  – User can install software in their home/scratch dir
  – Do not run the “sudo” command when installing software
Computing Environment

- **Path:** the location on disk where an executable or library may be found.
- Paths are saved as *environment variables*, so you can choose which libraries and executables will be used by modifying the variables.
- There is a lot of software, many versions, and many paths to manage.

.... How do you manage all these software versions?

[Link to wiki page](https://hprc.tamu.edu/wiki/Grace:Computing_Environment#Modules)
Managing software versions using lmod

- Each version of a software, application, library, etc. is available as a module.
  - Module names have the format:
    - software-name / version [-dependency-version (optional)]
    - TensorFlow/2.5.0-Python-3.7.4

- **Loading** a module adds its location on disk to your Path environment variable.
  - Its dependencies will also be loaded automatically.

[link to wiki page](http://hprc.tamu.edu/wiki/Grace:Computing_Environment#Modules)
Module Usage Basics

**module avail or mla**

```bash
# list all available modules (sometimes it is very slow)
# space bar down, page up/down, q to quit
# / for case sensitive search (similar to a Unix man page)
```

**module spider <word>**

```bash
# case insensitive search for modules with ‘word’ in name
```

**module load <module>**

```bash
# add <module> paths to the current environment variables
```

- Information about specific modules can also be found on our software page: [https://hprc.tamu.edu/software/grace/#](https://hprc.tamu.edu/software/grace/#)
- Learn more about module commands on our wiki: [https://hprc.tamu.edu/wiki/SW:Modules](https://hprc.tamu.edu/wiki/SW:Modules)
Module Hierarchy

Toolchains

- Core modules called Toolchains are common dependencies.
- Toolchain dependencies are not included in the module name.

What if two modules have the same name but a different Toolchain dependency?

1. LAMMPS/3Mar2020-Python-3.8.2-kokkos (depends on foss/2020a)
2. LAMMPS/3Mar2020-Python-3.8.2-kokkos (depends on intel/2020a)

Module Hierarchy

- The Toolchain module acts as a gatekeeper. You must load the Toolchain first in order to access the modules that depend on it.

hprc.tamu.edu/wiki/Grace:Computing_Environment#Modules
Module Usage Example

- Installed applications are made available with the module system
  Grace uses a software hierarchy inside the module system

- Example:

```
mla perl
# search for a specific piece of software by keyword

module spider Perl/5.32.0
# find how to load a particular module using the full module name based on the above results

module load GCCcore/10.2.0 Perl/5.32.0
# Load the base dependency module(s) first then the full module name
```
Hands-on exercises:

Open a terminal on Grace in Portal

Menu > Clusters > _Grace Shell Access

use your Netid password and your two-factor Authentication method.
Module Loading Exercise

1. `mla blast+`  
   # see which versions of BLAST+ are available

2. `ml BLAST+/2.8.1`  
   # error; you can’t do that yet

3. `module spider BLAST+/2.8.1`  
   # learn how to load this module

4. `ml ___________ BLAST+/2.8.1`  
   # fill in the blank (with the correct toolchains) to # load this module

5. `ml list`  
   # list all loaded modules

6. `ml GCC/10.2.0`  
   # change version of a loaded Toolchain module (GCC)  
   # notice the message about reloaded modules

7. `ml list`  
   # list all loaded modules

8. `ml purge`  
   # remove all loaded modules
Development Environment - Toolchains

- Toolchains are combinations of compilers, MPI libraries, and highly optimized math libraries.
- Toolchain components are primarily either Intel or Open Source.

Example toolchains for C++ development:

<table>
<thead>
<tr>
<th>Components</th>
<th>Open Source</th>
<th>Intel Source</th>
<th>Mixed Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compiler only</td>
<td>GCCcore</td>
<td>iccifort</td>
<td>-</td>
</tr>
<tr>
<td>Compiler + MPI</td>
<td>gompi</td>
<td>iimpi</td>
<td>iompi</td>
</tr>
<tr>
<td>Compiler + MPI + MKL, BLAS, FFTW, LAPACK</td>
<td>foss</td>
<td>intel</td>
<td>iomkl</td>
</tr>
<tr>
<td>Compiler + all of the above + CUDA Compiler</td>
<td>fosscuda</td>
<td>intelcuda</td>
<td>iomklc</td>
</tr>
</tbody>
</table>

Example usage: `module load foss/2019b`

As usual, see our Wiki for more information. [hprc.tamu.edu/wiki/SW:Toolchains](http://hprc.tamu.edu/wiki/SW:Toolchains)
Module Usage Practices

- Applications installed as modules are available to all users (except for restricted modules).
- It's a good habit to unload unused modules before loading new modules.
- It is recommended to load a specific software version instead of the defaults.
- Avoid loading modules in your ~/.bashrc.
- Avoid mixing toolchains when loading multiple modules at the same time. This usually leads to one of them not working.

hprc.tamu.edu/wiki/Terra:Computing_Environment#Modules
Software Install Example: Virtual Env

Python is a language which supports many external libraries in the form of extensions. (called Python Packages).

Some commonly used packages:
- SciPy & NumPy
- Jupyter notebook
- Scikit-learn

You can install these yourself using the Virtual Environment feature. Instructions are on the wiki:

https://hprc.tamu.edu/wiki/SW:Python#Create_a_virtual_environment
Software Install Exercise

1. cd $SCRATCH
   mkdir python_envs
   cd python_envs  # setup workspace

2. module purge
   module load GCC/10.2.0 Python/3.8.6  # setup Python module

3. virtualenv my_example_venv
   source my_example_venv/bin/activate  # setup your virtual environment

4. python -c "import pytime"
   # check if python-time is installed (it’s not)

5. pip install python-time  # install python-time

6. python -c "import pytime; print(pytime)"  # where is python-time installed?

7. deactivate  # all done with virtual env
Cluster Computing
Batch Computing on HPRC Clusters

A batch job script is a text file that contains both Unix commands and Batch manager job parameters.

Batch manager: 
*Slurm* on Terra, Grace, and FASTER

Create job 

Submit job

Login node 

Job script file 

Queue

Cluster compute nodes 

Job script file again
Consumable Computing Resources

- Resources specified in a job file:
  - Processor cores
  - Memory
  - Wall time
  - GPU
- Service Unit (SU) - Billing Account
  - Use "myproject" to query on command line
    hprc.tamu.edu/wiki/HPRC:AMS:Service_Unit
Hands-on exercise:

Check your Accounts on Grace in Portal

Menu > Dashboard > Grace Dashboard
Locate your default account
**Slurm**: Examples of SUs charged based on Job Cores, Time and Memory Requested

A Service Unit (SU) on is equivalent to one core or a proportional amount of memory on the node, for one hour.

On Grace: a typical node has 48 cores and 360 GB usable for jobs. (Ratio: 7.5 GB per core)

<table>
<thead>
<tr>
<th>Number of Cores</th>
<th>GB of memory per core</th>
<th>Total Memory (GB)</th>
<th>Hours</th>
<th>SUs charged</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7.5</td>
<td>7.5</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>24</td>
<td>1</td>
<td>24</td>
<td>1</td>
<td>24</td>
</tr>
<tr>
<td>1</td>
<td>180</td>
<td>180</td>
<td>1</td>
<td>24</td>
</tr>
<tr>
<td>48</td>
<td>7.5</td>
<td>360</td>
<td>1</td>
<td>48</td>
</tr>
</tbody>
</table>

[hprc.tamu.edu/wiki/HPRC:AMS:Service_Unit](hprc.tamu.edu/wiki/HPRC:AMS:Service_Unit)
Other Computing Resources

- Software license/token:
  - Use "license_status" to query
  - hprc.tamu.edu/wiki/SW:License_Checker

Find available license for "ansys":

```
license_status -s ansys
```

License status for ANSYS:

<table>
<thead>
<tr>
<th>License Name</th>
<th># Issued</th>
<th># In Use</th>
<th># Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>aa_mcad</td>
<td>50</td>
<td>0</td>
<td>50</td>
</tr>
<tr>
<td>aa_r</td>
<td>50</td>
<td>32</td>
<td>18</td>
</tr>
<tr>
<td>aim_mpl</td>
<td>50</td>
<td>0</td>
<td>50</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Find detail options:

```
license_status -h
```
Batch Queues

- Job submissions are auto-assigned to batch queues based on the resources requested (number of cores/nodes and walltime limit).
- Some jobs can be directly submitted to a queue:
  - For example, if gpu nodes are needed, use the `gpu` partition/queue.
- Batch queue policies are used to manage the workload and may be adjusted periodically.

https://hprc.tamu.edu/wiki/Grace:Batch#Batch_Queue
### sinfo : Current Queues

<table>
<thead>
<tr>
<th>PARTITION</th>
<th>AVAIL</th>
<th>TIMELIMIT</th>
<th>JOB_SIZE</th>
<th>NODES(A/I/O/T)</th>
<th>CPUS(A/I/O/T)</th>
</tr>
</thead>
<tbody>
<tr>
<td>short*</td>
<td>up</td>
<td>2:00:00</td>
<td>1.16</td>
<td>156/145/3/304</td>
<td>3667/4761/84/8512</td>
</tr>
<tr>
<td>medium</td>
<td>up</td>
<td>1:00:00:00</td>
<td>1.64</td>
<td>156/145/3/304</td>
<td>3667/4761/84/8512</td>
</tr>
<tr>
<td>long</td>
<td>up</td>
<td>7:00:00:00</td>
<td>1.32</td>
<td>156/145/3/304</td>
<td>3667/4761/84/8512</td>
</tr>
<tr>
<td>gpu</td>
<td>up</td>
<td>2:00:00:00</td>
<td>1.48</td>
<td>48/0/0/48</td>
<td>797/547/0/1344</td>
</tr>
<tr>
<td>vnc</td>
<td>up</td>
<td>12:00:00</td>
<td>1</td>
<td>48/0/0/48</td>
<td>797/547/0/1344</td>
</tr>
<tr>
<td>xlong</td>
<td>up</td>
<td>21:00:00:00</td>
<td>1.32</td>
<td>108/145/3/256</td>
<td>2870/4214/84/7168</td>
</tr>
<tr>
<td>staff</td>
<td>up</td>
<td>infinite</td>
<td>1.infinite</td>
<td>156/145/3/304</td>
<td>3667/4761/84/8512</td>
</tr>
<tr>
<td>low_priority</td>
<td>up</td>
<td>1:00:00:00</td>
<td>1.infinite</td>
<td>156/145/3/304</td>
<td>3667/4761/84/8512</td>
</tr>
<tr>
<td>special</td>
<td>up</td>
<td>7:00:00:00</td>
<td>1.infinite</td>
<td>156/145/3/304</td>
<td>3667/4761/84/8512</td>
</tr>
<tr>
<td>knl</td>
<td>up</td>
<td>7:00:00:00</td>
<td>1.8</td>
<td>0/14/2/16</td>
<td>0/980/140/1120</td>
</tr>
</tbody>
</table>

For the NODES and CPUS columns:

- **A** = Active (in use by running jobs)
- **I** = Idle (available for jobs)
- **O** = Offline (unavailable for jobs)
- **T** = Total
Batch Job Scripts
Sample Python Job Script Structure (**Slurm**)

```bash
#!/bin/bash

##NECESSARY JOB SPECIFICATIONS
#SBATCH --job-name=JobExample1
#SBATCH --time=01:30:00
#SBATCH --ntasks=1
#SBATCH --mem=2G
#SBATCH --output=stdout.%j

##OPTIONAL JOB SPECIFICATIONS
#SBATCH --account=123456
#SBATCH --mail-type=ALL
#SBATCH --mail-user=email_address

# setup software environment
module purge
module load GCC10.2.0 Python/3.8.6
source $SCRATCH/.../env/bin/activate
./my_program.py
```

These parameters describe your job to the job scheduler

Load the required module(s)
Activate your virtual environment
Start your Python executable

This is single line comment and not run as part of the script

See also `example01.job`
## Important Batch Job Parameters (Slurm)

<table>
<thead>
<tr>
<th>Slurm parameter</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>#SBATCH --time=HH:MM:SS</code></td>
<td>Specifies the time limit for the job. Must specify seconds SS on Terra</td>
</tr>
<tr>
<td><code>#SBATCH --ntasks=NNN</code></td>
<td>Total number of tasks (cores) for the job.</td>
</tr>
<tr>
<td><code>#SBATCH --ntasks-per-node=XX</code></td>
<td>Specifies the maximum number of tasks (cores) to allocate per node</td>
</tr>
<tr>
<td><code>#SBATCH --mem=nnnnM</code> or <code>#SBATCH --mem=nG</code></td>
<td>Sets the maximum amount of memory (MB).</td>
</tr>
<tr>
<td></td>
<td>G for GB is supported on Terra</td>
</tr>
</tbody>
</table>

(memory per NODE)
Mapping Jobs to Cores per Node on Terra

A. 28 cores on 1 compute node

#SBATCH --ntasks 28
#SBATCH --tasks-per-node=28

Preferred Mapping (if applicable)

B. 28 cores on 2 compute nodes

#SBATCH --ntasks 28
#SBATCH --tasks-per-node=14

C. 28 cores on 4 compute nodes

#SBATCH --ntasks 28
#SBATCH --tasks-per-node=7
Job Memory Requests on Terra

- Specify memory request based on memory per node:
  
  `#SBATCH --mem=xxxxxM`  # memory per node in MB
  
  or
  
  `#SBATCH --mem=xG`  # memory per node in GB

- On 64GB nodes, usable memory is at most 56 GB. The per-process memory limit should not exceed 2000 MB for a 28-core job.

- On 128GB nodes, usable memory is at most 112 GB. The per-process memory limit should not exceed 4000 MB for a 28-core job.
Job Memory Requests on Grace

- Specify memory request based on memory per node:
  - `#SBATCH --mem=xxxxxM`  # memory per node in MB
  - or
  - `#SBATCH --mem=xG`  # memory per node in GB

- On 384GB nodes, usable memory is at most 360 GB.
  The per-process memory limit should not exceed ~7500 MB for a 48-core job.

- On 3TB nodes, usable memory is at most 2900 GB.
  The per-process memory limit should not exceed 37120 MB for a 48-core job.
Pop Quiz
Pop Quiz

Which one of the following HPRC clusters is *not* in use?

A. Grace  C. FASTER
B. Terra  D. ViDaL
#SBATCH --job-name=stacks_S2
#SBATCH --ntasks=80
#SBATCH --ntasks-per-node=20
#SBATCH --mem=40G
#SBATCH --time=48:00:00
#SBATCH --output stdout.%J
#SBATCH --error stderr.%J

How many nodes is this job requesting?

A. 1600  
B. 80  
C. 20  
D. 4
(end of Pop Quiz)
## Job Submission and Tracking: Grace and Terra

<table>
<thead>
<tr>
<th>Slurm commands</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sbatch</code> jobfile1</td>
<td>Submit jobfile1 to batch system</td>
</tr>
<tr>
<td><code>squeue</code> [-u user_name] [-j job_id]</td>
<td>List jobs</td>
</tr>
<tr>
<td><code>scancel</code> job_id</td>
<td>Kill a job</td>
</tr>
<tr>
<td><code>sacct</code> -X -j job_id</td>
<td>Show information for a job (can be when job is running or recently finished)</td>
</tr>
<tr>
<td><code>sacct</code> -X -S YYYY-HH-MM</td>
<td>Show information for all of your jobs since YYYY-HH-MM</td>
</tr>
<tr>
<td><code>lnu</code> job_id</td>
<td>Show resource usage for a job</td>
</tr>
<tr>
<td><code>pestat</code> -u $USER</td>
<td>Show resource usage for a running job</td>
</tr>
<tr>
<td><code>seff</code> job_id</td>
<td>Check CPU/memory efficiency for a job</td>
</tr>
</tbody>
</table>

[Link to HPRC:Batch_Translation](hprc.tamu.edu/wiki/HPRC:Batch_Translation)
Job Environment Variables

- **Linux:**
  - `$PATH` = list of directories where executables are found
  - `$LD_LIBRARY_PATH` = list of directories where libraries are found
  - `$SCRATCH` = short-hand for `/scratch/user/<NetID>` directory

- **Slurm:**
  - `$SLURM_JOBID` = job id, unique number for each job
  - `$SLURM_SUBMIT_DIR` = directory where job was submitted from
  - `$TMPDIR` = `/work/job.$SLURM_JOBID`
    - $TMPDIR is local to each assigned compute node for the job and is about 850GB
    - Use of $TMPDIR is recommended for jobs that use many small temporary files
    - Do not use $TMPDIR for software that has checkpoints to restart where it left off

hprc.tamu.edu/wiki/Ada:Batch_Processing_LSF#Environment_Variables
hprc.tamu.edu/wiki/Terra:Batch#Environment_Variables
Batch Job Exercises
Hands-on exercises:

Copy the example files into your scratch directory, if you haven’t done so already.

```bash
cp -r /scratch/training/Intro-to-Grace $SCRATCH
```

Inspect the contents.

```bash
cd $SCRATCH/Intro-to-Grace
ls
ls *
```
Job Exercise: Check Output

Submit job

```
cd batch_examples
```

**Grace:**

```
sbatch example01.job
```

Submitted batch job <####>

Check status

```
squeue -u $USER
```

<table>
<thead>
<tr>
<th>JOID</th>
<th>NAME</th>
<th>USER</th>
<th>PARTITION</th>
<th>NODES</th>
<th>CPUS</th>
<th>STATE</th>
<th>TIME</th>
<th>TIME_LEFT</th>
<th>START_TIME</th>
<th>REASON</th>
<th>NODELIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>64039</td>
<td>somejob</td>
<td>someuser</td>
<td>medium</td>
<td>4</td>
<td>112</td>
<td>PENDING</td>
<td>0:00</td>
<td>20:00</td>
<td>2017-01-30T21:00</td>
<td>Resources</td>
<td></td>
</tr>
<tr>
<td>64038</td>
<td>somejob</td>
<td>someuser</td>
<td>medium</td>
<td>4</td>
<td>112</td>
<td>RUNNING</td>
<td>2:49</td>
<td>17:11</td>
<td>2017-01-30T20:40</td>
<td>None</td>
<td>tnxt-[0401-0404]</td>
</tr>
</tbody>
</table>

Check output

```
cat output.ex01.env_variables.<tab autocomplete>
```

This job output file was created by the parameter in your job script file

**Terra:**

```
#SBATCH -o output.ex01.env_variables.%j
```
Job Exercise: Check Status

**Submit job**
```bash
cd batch_examples
```

**Grace:**
```bash
sbatch example02.job
```
Submitted batch job <#####>

**Check status**
```bash
squeue -u $USER
```

<table>
<thead>
<tr>
<th>JOBID</th>
<th>NAME</th>
<th>USER</th>
<th>PARTITION</th>
<th>NODES</th>
<th>CPUS</th>
<th>STATE</th>
<th>TIME</th>
<th>TIME_LEFT</th>
<th>START_TIME</th>
<th>REASON</th>
<th>NODELIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>64039</td>
<td>somejob</td>
<td>someuser</td>
<td>medium</td>
<td>4</td>
<td>112</td>
<td>PENDING</td>
<td>0:00</td>
<td>20:00</td>
<td>2017-01-30T21:00:4</td>
<td>Resources</td>
<td></td>
</tr>
<tr>
<td>64038</td>
<td>somejob</td>
<td>someuser</td>
<td>medium</td>
<td>4</td>
<td>112</td>
<td>RUNNING</td>
<td>2:49</td>
<td>17:11</td>
<td>2017-01-30T20:40:4</td>
<td>None</td>
<td>tnxt-[0401-0404]</td>
</tr>
</tbody>
</table>

**Check output**
```bash
cat output.ex02.echo_numbers.<tab autocomplete>
```

This job output file was created by the parameter in your job script file

**Terra:**
```bash
#SBATCH -o output.ex02.echo_numbers.%j
```
Job Exercise: Debug job failures

**Submit job**

```
cd batch_examples
```

**Terra:**

```
sbatch example03.job
```

Submitted batch job <#####>

**Check output**

```
cat output.ex03.python_mem.<tab autocomplete>
```

This job output file was created by the parameter in your job script file

```
Terra:  #SBATCH -o output.ex03.python_mem.%j
```

```
slurmstepd: error: Exceeded job memory limit at some point.
```

Make the necessary adjustments to memory parameters in your job script and resubmit the job.
Job Exercise: Bad job script

Quiz: what went wrong with this job script?
Check your Service Unit (SU) Balance

- List the SU Balance of your Account(s)

  myproject

<table>
<thead>
<tr>
<th>Account</th>
<th>FY</th>
<th>Default</th>
<th>Allocation</th>
<th>Used &amp; Pending SUs</th>
<th>Balance</th>
<th>PI</th>
</tr>
</thead>
<tbody>
<tr>
<td>1228000223136</td>
<td>2019</td>
<td>N</td>
<td>10000.00</td>
<td>0.00</td>
<td>10000.00</td>
<td>Doe, John</td>
</tr>
<tr>
<td>1428000243716</td>
<td>2019</td>
<td>Y</td>
<td>5000.00</td>
<td>-71.06</td>
<td>4928.94</td>
<td>Doe, Jane</td>
</tr>
<tr>
<td>1258000247058</td>
<td>2019</td>
<td>N</td>
<td>5000.00</td>
<td>-0.91</td>
<td>4999.09</td>
<td>Doe, Jane</td>
</tr>
</tbody>
</table>

- To specify a project ID to charge in the job file
  - Grace & Terra:  #SBATCH -A Account#

- Run "myproject -d Account#" to change default project account

- Run "myproject -h" to see more options
Job submission issue: insufficient SUs

Bonus Assignment: modify a job file so that the requested SU’s are too much for your account. I.e.: make an error message (like the following) appear.

```bash
$ sbatch myjob
sbatch: error: (from job_submit) your account's balance is not sufficient to submit your job
Project Account: 123940134739
Account Balance: 382.803877
Requested SUs: 18218.666666667
```

- What to do if you need more SUs
  - Ask your PI to transfer SUs to your account
  - Apply for more SUs (if you are eligible, as a PI or permanent researcher)

hprc.tamu.edu/wiki/HPRC:CommonProblems#Q:_How_do_I_get_more_SUs.3F
hprc.tamu.edu/wiki/HPRC:AMS:Service_Unit
hprc.tamu.edu/wiki/HPRC:AMS:UI
Job Composer
Menu
  > Dashboard
  > Grace Dashboard

Button
  “Job Composer”
  (at the bottom)
Other Batch Job Examples
#!/bin/bash

##NECESSARY JOB SPECIFICATIONS
#SBATCH --job-name=JobExample1 #Set the job name to "JobExample1"
#SBATCH --time=01:30:00 #Set the wall clock limit to 1hr and 30min
#SBATCH --ntasks=1 #Request 1 task
#SBATCH --mem=2560M #Request 2560MB (2.5GB) per node
#SBATCH --output=Example1Out.%j #Send stdout/err to "Example1Out.[jobID]"

##OPTIONAL JOB SPECIFICATIONS
##CHANGE ACCOUNT NUMBER AND EMAIL ADDRESS BEFORE USING
#SBATCH --account=123456 #Set billing account to 123456
#SBATCH --mail-type=ALL #Send email on all job events
#SBATCH --mail-user=email_address #Send all emails to email_address

# this intel toolchain is just an example. recommended toolchain is TBD
module purge
module load intel/2017A

# run your program
./helloworld.omp.C.exe
```bash
#!/bin/bash

##NECESSARY JOB SPECIFICATIONS
#SBATCH --job-name=JobExample2 # Set the job name to "JobExample2"
#SBATCH --time=6:30:00 # Set the wall clock limit to 6hr and 30min
#SBATCH --nodes=1 # Request 1 node
#SBATCH --ntasks-per-node=8 # Request 8 tasks(cores) per node
#SBATCH --mem=8G # Request 8GB per node
#SBATCH --output=Example2Out.%j # Send stdout/err to "Example2Out.[jobID]"

##OPTIONAL JOB SPECIFICATIONS
##CHANGE ACCOUNT NUMBER AND EMAIL ADDRESS BEFORE USING
#SBATCH --account=123456 # Set billing account to 123456 #find your account with "myproject"
#SBATCH --mail-type=ALL # Send email on all job events
#SBATCH --mail-user=email_address # Send all emails to email_address

# load required module(s)
module purge
module load intel/2017A

# run your program
mpirun ./helloworld.mpi.C.exe
```

SUs = 52
#!/bin/bash

##NECESSARY JOB SPECIFICATIONS
#SBATCH --job-name=JobExample3
#SBATCH --time=1-12:00:00
#SBATCH --ntasks=8
#SBATCH --ntasks-per-node=2
#SBATCH --mem=4096M
#SBATCH --output=Example3Out.%j

##OPTIONAL JOB SPECIFICATIONS
##CHANGE ACCOUNT NUMBER AND EMAIL ADDRESS BEFORE USING
#SBATCH --account=123456
#SBATCH --mail-type=ALL
#SBATCH --mail-user=email_address

# this intel toolchain is just an example. recommended toolchain is TBD
module purge
module load intel/2017A

# run program with MPI
mpirun ./helloworld.mpi.C.exe
#!/bin/bash

## NECESSARY JOB SPECIFICATIONS
#SBATCH --job-name=JobExample4    # Set the job name to "JobExample4"
#SBATCH --time=01:00:00          # Set the wall clock limit to 1hr
#SBATCH --ntasks=1              # Request 1 task (core)
#SBATCH --mem=2560M             # Request 2560MB (2.5GB) per node
#SBATCH --output=Example4Out.%j # Send stdout and stderr to "Example4Out.[jobID]"
#SBATCH --gres=gpu:1            # Request 1 GPU
#SBATCH --partition=gpu         # Request the GPU partition/queue

## OPTIONAL JOB SPECIFICATIONS
## CHANGE ACCOUNT NUMBER AND EMAIL ADDRESS BEFORE USING
##SBATCH --account=123456       # Set billing account to 123456  #find your account with "myproject"
##SBATCH --mail-type=ALL       # Send email on all job events
##SBATCH --mail-user=email_address # Send all emails to email_address

# load required module(s)
module purge
module load intel/2017A CUDA/9.2.148.1

# run your program
./deviceQuery

SUs = 28
List Node Utilization: `lnu`

`lnu jobid`

# lists the node utilization across all nodes for a running job.
# to see more options use: `lnu -h`

**Example:**

```
lnu <jobid>
```

Note: Slurm updates the node information every few minutes

```latex
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline
JOBID & NAME & USER & PARTITION & NODES & CPUS & STATE & TIME & TIME_LEFT & START_TIME \\
\hline
\hline
\end{tabular}
```

Note: CPULOAD is not the same as % utilization

**For the CPUS columns:**
- **A** = Active (in use by running jobs)
- **I** = Idle (available for jobs)
- **O** = Offline (unavailable for jobs)
- **T** = Total

<table>
<thead>
<tr>
<th>HOSTNAMES</th>
<th>CPU_LOAD</th>
<th>FREE_MEM</th>
<th>MEMORY</th>
<th>CPUS(A/I/O/T)</th>
</tr>
</thead>
<tbody>
<tr>
<td>tnxt-0703</td>
<td>26.99</td>
<td>53462</td>
<td>57344</td>
<td>28/0/0/28</td>
</tr>
<tr>
<td>tnxt-0704</td>
<td>26.93</td>
<td>52361</td>
<td>57344</td>
<td>28/0/0/28</td>
</tr>
<tr>
<td>tnxt-0705</td>
<td>26.95</td>
<td>47166</td>
<td>57344</td>
<td>28/0/0/28</td>
</tr>
</tbody>
</table>
Monitor Compute Node Utilization: `pestat`

**pestat [-u username]**  
# lists the node utilization across all nodes for a running job.  
# to see more options use: `pestat -h`

**Example:**

**pestat -u $USER**

<table>
<thead>
<tr>
<th>Hostname</th>
<th>Partition</th>
<th>Node</th>
<th>Num_CPU</th>
<th>CPUload</th>
<th>Memsize</th>
<th>Freemem</th>
<th>Joblist</th>
<th>State Use/Tot</th>
</tr>
</thead>
<tbody>
<tr>
<td>tnxt-0703</td>
<td>xlong</td>
<td>alloc</td>
<td>28</td>
<td>28</td>
<td>16.23*</td>
<td>57344</td>
<td>55506</td>
<td>565849 someuser</td>
</tr>
<tr>
<td>tnxt-0704</td>
<td>xlong</td>
<td>alloc</td>
<td>28</td>
<td>28</td>
<td>19.60*</td>
<td>57344</td>
<td>53408</td>
<td>565849 someuser</td>
</tr>
<tr>
<td>tnxt-0705</td>
<td>xlong</td>
<td>alloc</td>
<td>28</td>
<td>28</td>
<td>19.56*</td>
<td>57344</td>
<td>53408</td>
<td>565849 someuser</td>
</tr>
</tbody>
</table>

Low CPU load utilization highlighted in **Red**  
(Freemem should also be noted)

**pestat -u $USER**

<table>
<thead>
<tr>
<th>Hostname</th>
<th>Partition</th>
<th>Node</th>
<th>Num_CPU</th>
<th>CPUload</th>
<th>Memsize</th>
<th>Freemem</th>
<th>Joblist</th>
<th>State Use/Tot</th>
</tr>
</thead>
<tbody>
<tr>
<td>tnxt-0703</td>
<td>xlong</td>
<td>alloc</td>
<td>28</td>
<td>28</td>
<td>27.54</td>
<td>57344</td>
<td>55506</td>
<td>565849 someuser</td>
</tr>
<tr>
<td>tnxt-0704</td>
<td>xlong</td>
<td>alloc</td>
<td>28</td>
<td>28</td>
<td>27.50</td>
<td>57344</td>
<td>53408</td>
<td>565849 someuser</td>
</tr>
<tr>
<td>tnxt-0705</td>
<td>xlong</td>
<td>alloc</td>
<td>28</td>
<td>28</td>
<td>26.47*</td>
<td>57344</td>
<td>53408</td>
<td>565849 someuser</td>
</tr>
</tbody>
</table>

Good CPU load utilization highlighted in **Purple**  
(Ideal CPU load utilization displayed in White)
Other Type of Jobs

- **MPI and OpenMP**
- **Visualization:**
  - [portal.hprc.tamu.edu](http://portal.hprc.tamu.edu) Interactive Apps > choose a visual application
- **Large number of concurrent single core jobs**
  - Check out **tamulauncher**
    - [hprc.tamu.edu/wiki/SW:tamulauncher](http://hprc.tamu.edu/wiki/SW:tamulauncher)
    - Useful for running many single core commands concurrently across multiple nodes within a job
    - Can be used with serial or multi-threaded programs
    - Distributes a set of commands from an input file to run on the cores assigned to a job
    - Can only be used in batch jobs
    - If a tamulauncher job gets killed, you can resubmit the same job to complete the unfinished commands in the input file
Need Help?

- Try these:
  - First check the FAQ [hprc.tamu.edu/wiki/HPRC:CommonProblems](hprc.tamu.edu/wiki/HPRC:CommonProblems)
  - Also try the Terra User Guide [hprc.tamu.edu/wiki/Terra](hprc.tamu.edu/wiki/Terra)
  - Email your questions to [help@hprc.tamu.edu](mailto:help@hprc.tamu.edu). (Managed by a ticketing system)

- Help us, help you -- we need more info
  - Which Cluster
  - UserID/NetID (*UIN is not needed!*)
  - Job id(s) if any
  - Location of your jobfile, input/output files
  - Application used if any
  - Module(s) loaded if any
  - Error messages
  - Steps you have taken, so we can reproduce the problem

- Or visit us @ 114A Henderson Hall (Making an appointment is recommended.)
Thank you.

Questions?
MobaXterm with Duo

- Use “Session” icon
- Use local terminal (command line)

1. Use “Session” icon
2. Use local terminal (command line)

Connect to Terra from local terminal:

```
$ ssh UserNetID@terra.tamu.edu
```

 зависимости (snipped) ……

UserNetID@terra.tamu.edu's password:
UserNetID@terra.tamu.edu's password:
UserNetID@terra.tamu.edu's password:

Password:

Duo two-factor login for UserNetID

Enter a passcode or select one of the following options:

1. Duo Push to XXX-XXX-1234
2. Phone call to XXX-XXX-1234
3. SMS passcodes to XXX-XXX-1234 (next code starts with: 9)

Passcode or option (1-3): 1
Success. Logging you in...

hprc.tamu.edu/wiki/Two_Factor#MobaXterm
WinSCP with Duo

hprc.tamu.edu/wiki/Two_Factor#WinSCP_.28Windows_only.29
Development Environment - Toolchains

● Intel toolchain (eg. software stack) is recommended
  - Intel C/C++/Fortran compilers (icc, icpc, ifort)
  - Intel Math Kernel Library
  - Intel MPI library

● For packages that require MPI but not MKL or BLAS/FFTW/LAPACK
  - iimpi/2018b       iompi/2018b     gompi/2018b

● Toolchains that contain MPI, MKL, and BLAS/FFTW/LAPACK
  - intel/2018b      iomkl/2018b     foss/2018b

● To load/use the current recommended Intel toolchain module

  module load intel/2018b

● If you do not want to use GCC version in the intel/2018b toolchain, find available gcc versions for applications which must use gcc/g++

  module spider GCC

hprc.tamu.edu/wiki/SW:Toolchains
hprc.tamu.edu/wiki/Ada:Compile:All#Getting_Started
hprc.tamu.edu/wiki/Terra:Compile:All#Getting_Started
The GCCcore Toolchain

● To minimize the number of software builds, the GCCcore-7.3.0 toolchain modules can be loaded alone or with any one of the following 2018b toolchains
  ○ intel/2018b
  ○ iomkl/2018b
  ○ foss/2018b

● Example of loading a GCCcore-7.3.0 module with a 2018b module

```bash
module load Bowtie2/2.3.4.3-intel-2018b
module load BCFtools/1.9-GCCcore-7.3.0
```

● See a short table of compatible toolchains

[toolchains]

[hprc.tamu.edu/wiki/SW:Toolchains]
Python-version-bare modules

- You need to load a non ‘-bare’ Python version along with the -bare module
  - If you do not, then the older default OS Python version will be used
- Used in conjunction with GCCcore-6.3.0 builds in order to reduce the number of software modules built.

```
intel/2017A  iomkl/2017A  foss/2017A
```

Three different examples of loading GCCcore-6.3.0-Python-bare and a Python module with a 2017A toolchain

1. `module load Cython/0.25.2-GCCcore-6.3.0-Python-2.7.12-bare`
   `module load Python/2.7.12-foss-2017A`

2. `module load Cython/0.25.2-GCCcore-6.3.0-Python-2.7.12-bare`
   `module load Python/2.7.12-iomkl-2017A`

3. `module load Cython/0.25.2-GCCcore-6.3.0-Python-2.7.12-bare`
   `module load HISAT2/2.1.0-intel-2017A-Python-2.7.12`

[[loadsPythonIndirectly]}

Loads Python indirectly