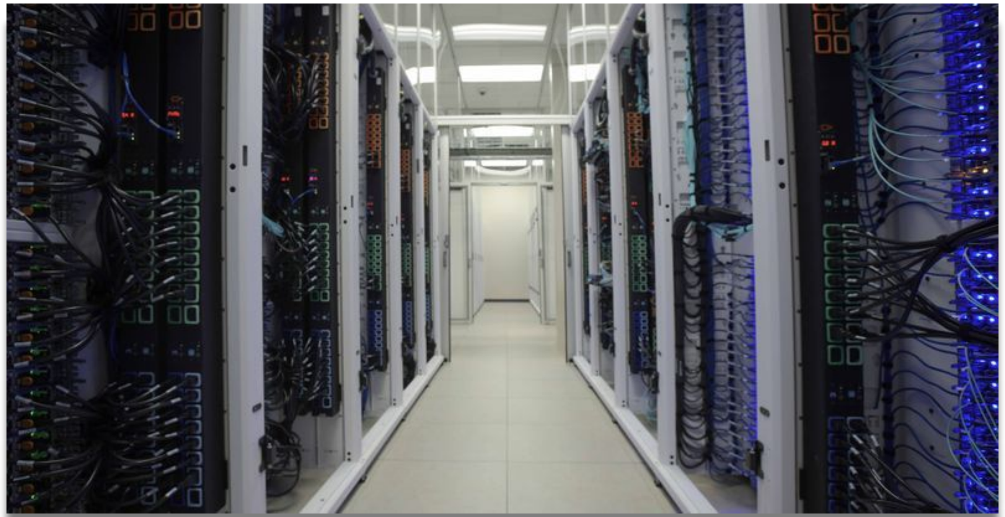
Things to Do While You are Waiting

- 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



Slides by **Richard Lawrence**

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

Cluster computing exercises:

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

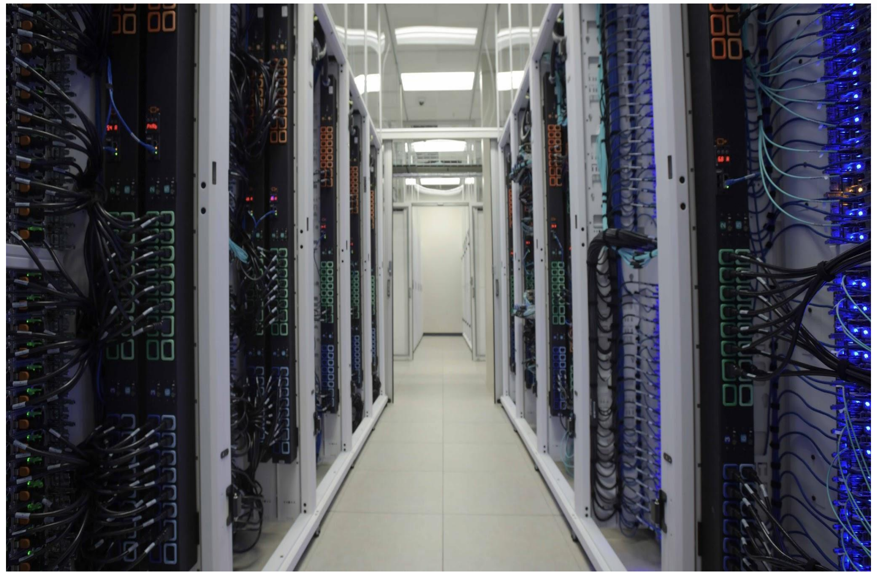
Interface for hands-on exercises:

- Terra: portal-terra.hprc.tamu.edu/
  or choose "Terra OnDemand" from portal.hprc.tamu.edu/

HPRC Clusters

Terra

deployed in 2017



Grace

Flagship cluster,
deployed in 2021

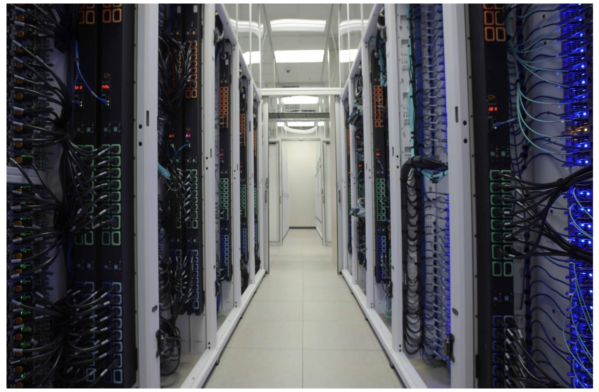
FASTER

In early access,
full deployment
coming soon

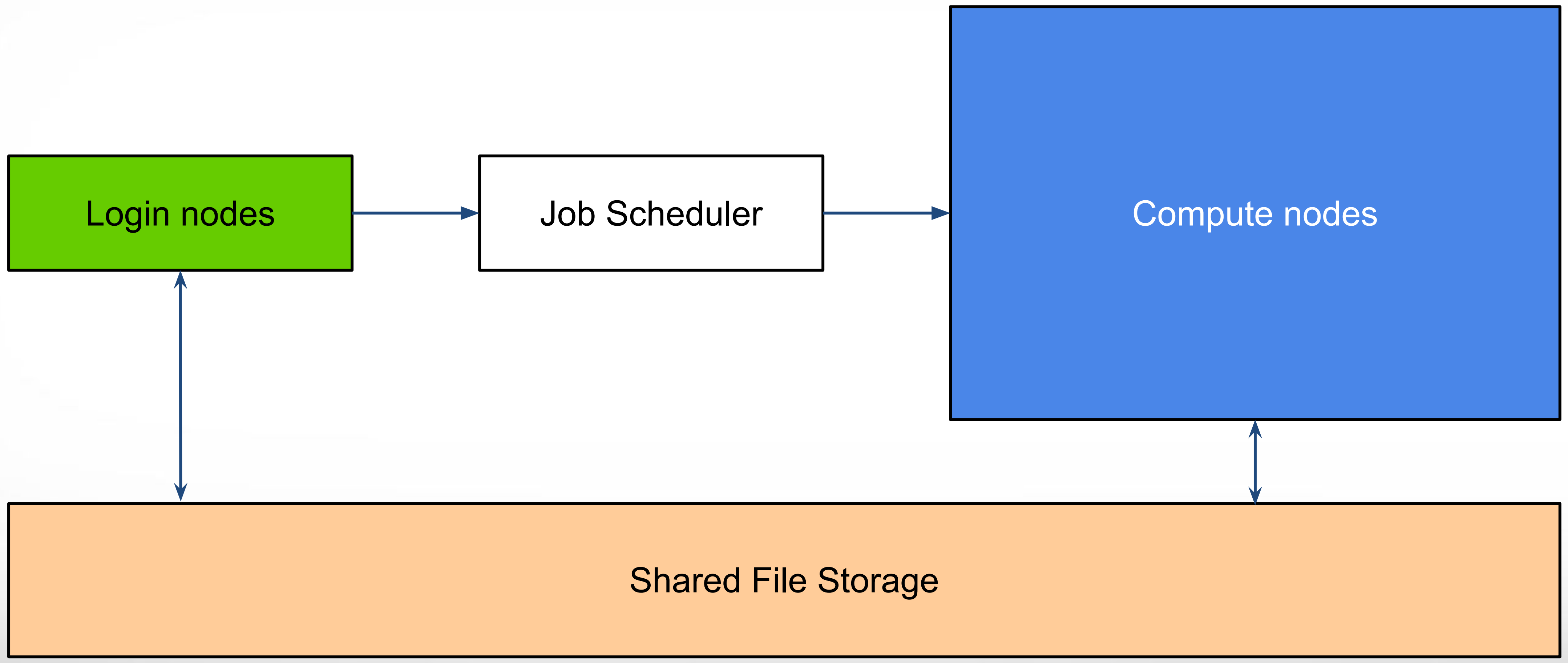
Clusters Are For You!

What kinds of problems are solved by cluster computing?

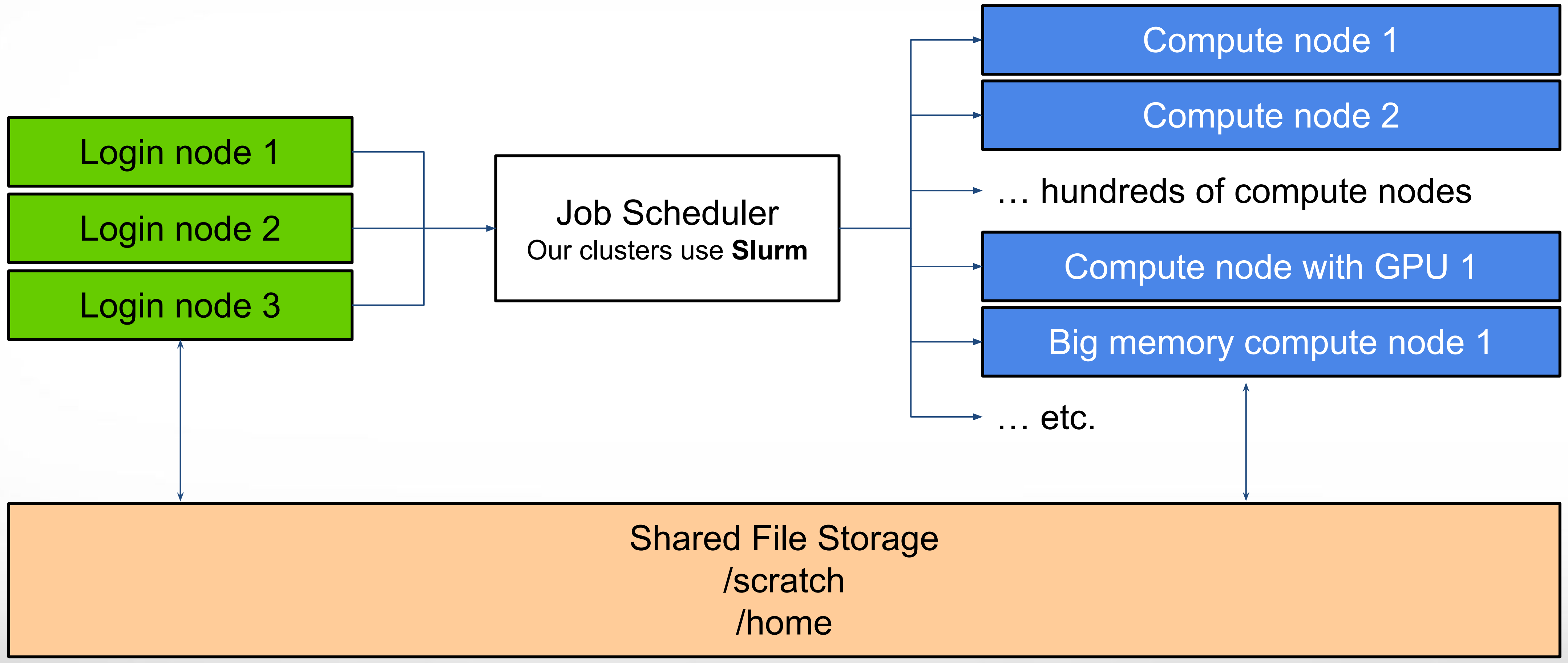
- Problems that are too big to fit in one laptop or workstation, due to limitation on memory, core count, or node count
- Problems that scale well with more CPU cores or memory
- Single-threaded problems with millions of permutations
- Problems that require large high speed storage and/or interconnect



Cluster Diagram

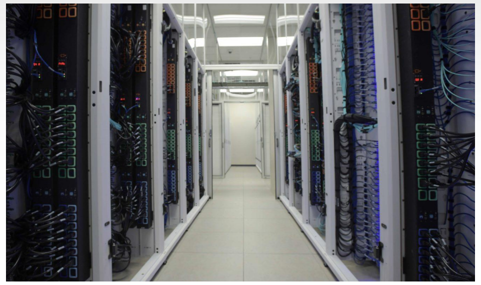


Cluster Diagram



HPRC Clusters

Terra


Grace

FASTER

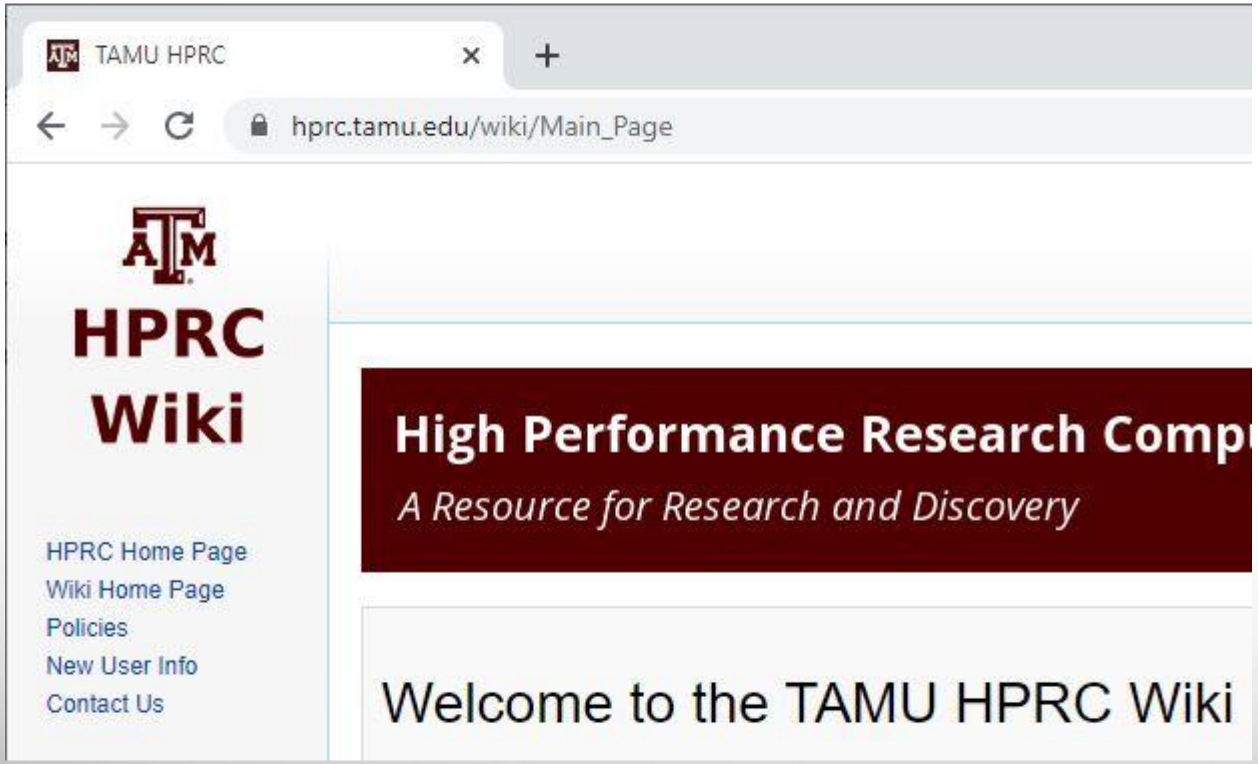
	Terra	Grace	FASTER
Total Nodes (Cores)	307 (8,512)	925 (44,656)	180 (11,520)
General Nodes	28 cores 64GB	48 cores 384GB	64 cores 256GB
Features	GPUs Many-core nodes	GPUs - multiprecision Big Memory Nodes	Composable GPUs Composable Memory
Job Scheduler	Slurm	Slurm	Slurm
Online Since	2017	2021	Coming Soon

hprc.tamu.edu/resources

HPRC Wiki - Hardware

Visit our wiki 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.



TAMU HPRC

hprc.tamu.edu/wiki/Main_Page

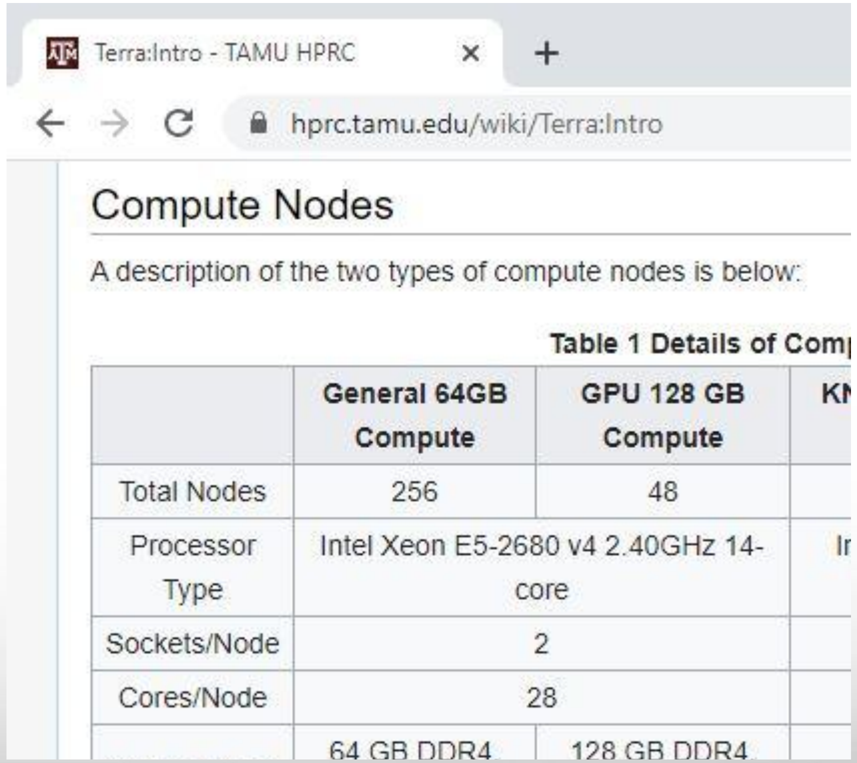
HPRC Wiki

HPRC Home Page
Wiki Home Page
Policies
New User Info
Contact Us

High Performance Research Comp

A Resource for Research and Discovery

Welcome to the TAMU HPRC Wiki



Terra:Intro - TAMU HPRC

hprc.tamu.edu/wiki/Terra:Intro

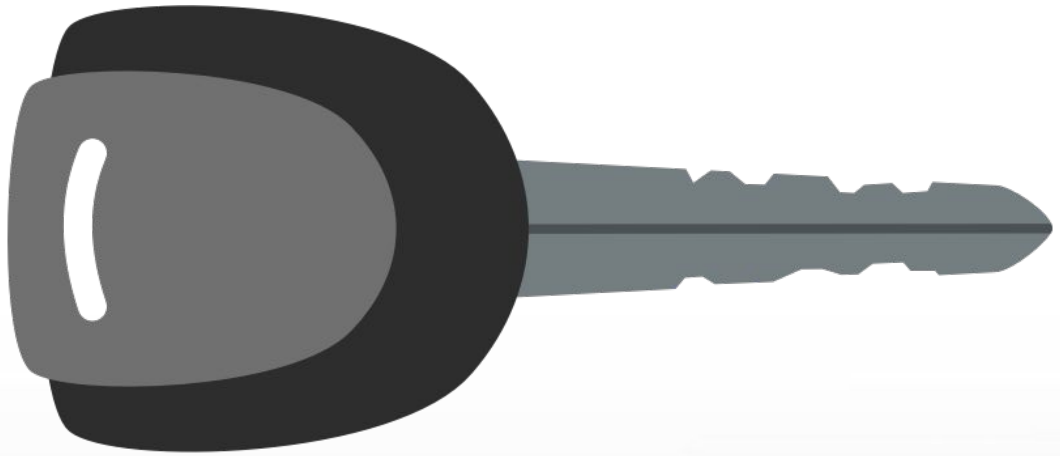
Compute Nodes

A description of the two types of compute nodes is below:

Table 1 Details of Comp

	General 64GB Compute	GPU 128 GB Compute	KN
Total Nodes	256	48	
Processor Type	Intel Xeon E5-2680 v4 2.40GHz 14-core		Ir
Sockets/Node	2		
Cores/Node	28		
	64 GB DDR4	128 GB DDR4	

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/)
- (Faculty and staff) Use Duo Keys - u.tamu.edu/get_duo_keys
- Instructions in two-factor wiki page (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

Directory	Environment Variable	Space Limit	File Limit	Intended Use
/home/$USER	$HOME	10 GB	10,000	Small amounts of processing.
/scratch/user/$USER	$SCRATCH	1 TB	250,000	Temporary storage of actively used large files. Not a long-term storage area.

File Systems and User Directories

Directory	Environment Variable	Space Limit	File Limit
/home/$USER	$HOME	10 GB	10,000
/scratch/user/$USER	$SCRATCH	1 TB	50,000

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

```
showquota
```

hprc.tamu.edu/wiki/Terra:Filesystems_and_Files
hprc.tamu.edu/wiki/Grace:Filesystems_and_Files

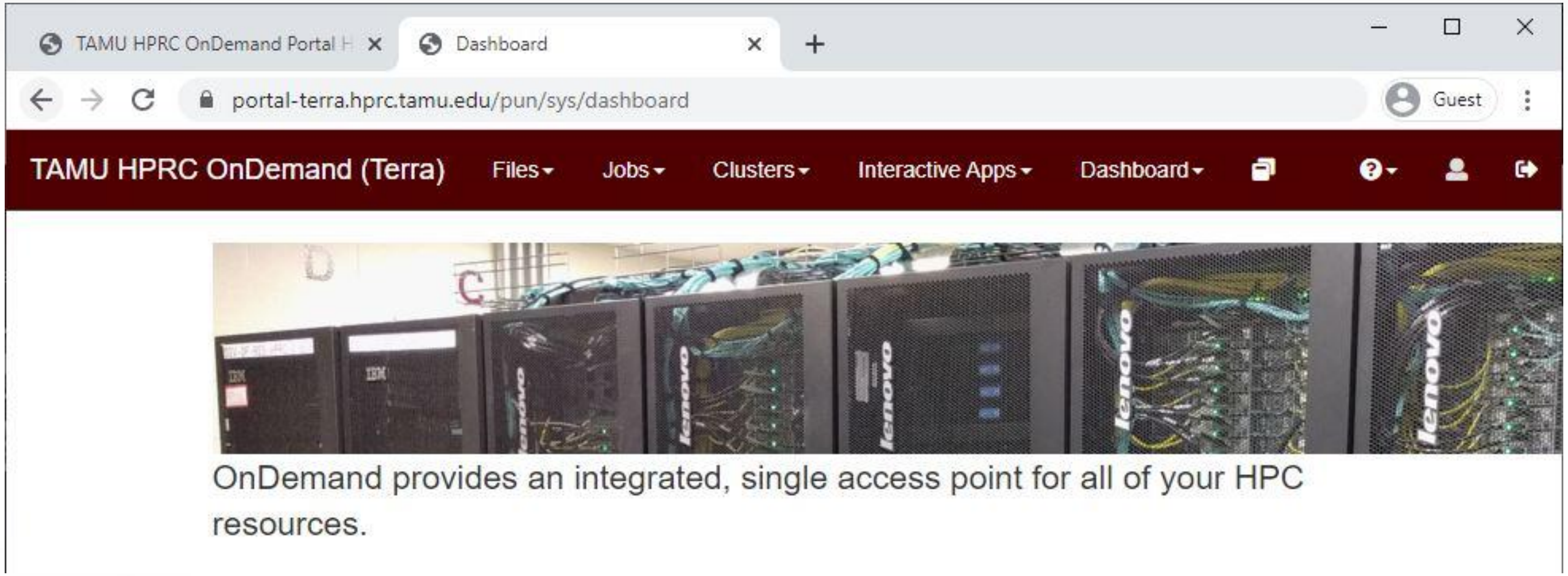
Hands-on exercises:

Activate TAMU VPN

Go to:

portal.hprc.tamu.edu

portal.hprc.tamu.edu


TAMU HPRC OnDemand Portal H
Dashboard
portal-terra.hprc.tamu.edu/pun/sys/dashboard
Guest
TAMU HPRC OnDemand (Terra)
Files
Jobs
Clusters
Interactive Apps
Dashboard

OnDemand provides an integrated, single access point for all of your HPC resources.

- 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

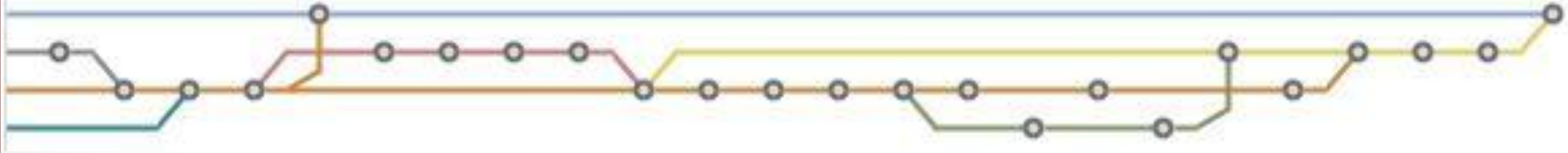
Hands-on exercise:

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?

hprc.tamu.edu/wiki/Grace:Computing_Environment#Modules

Computing Environment

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*)]

$$\overbrace{\texttt{TensorFlow}}/\overbrace{\texttt{2.5.0}}-\overbrace{\texttt{Python-3.7.4}}$$

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

hprc.tamu.edu/wiki/Grace:Computing_Environment#Modules

Module Usage Basics

```
module avail or mla        # 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>       # case insensitive search for modules with 'word' in name

module load <module>       # 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/#
- Learn more about `module` commands on our wiki: 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:

Components	Open Source	Intel Source	Mixed Source
Compiler only	`GCCcore`	`iccifort`	-
Compiler + MPI	`gompi`	`iimpi`	`iompi`
Compiler + MPI + MKL, BLAS, FFTW, LAPACK	`foss`	`intel`	`iomkl`
Compiler + all of the above + CUDA Compiler	`fosscuda`	`intelcuda`	`iomklc`

Example usage:

```
module load foss/2019b
```

As usual, see our Wiki for more information. 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)
```

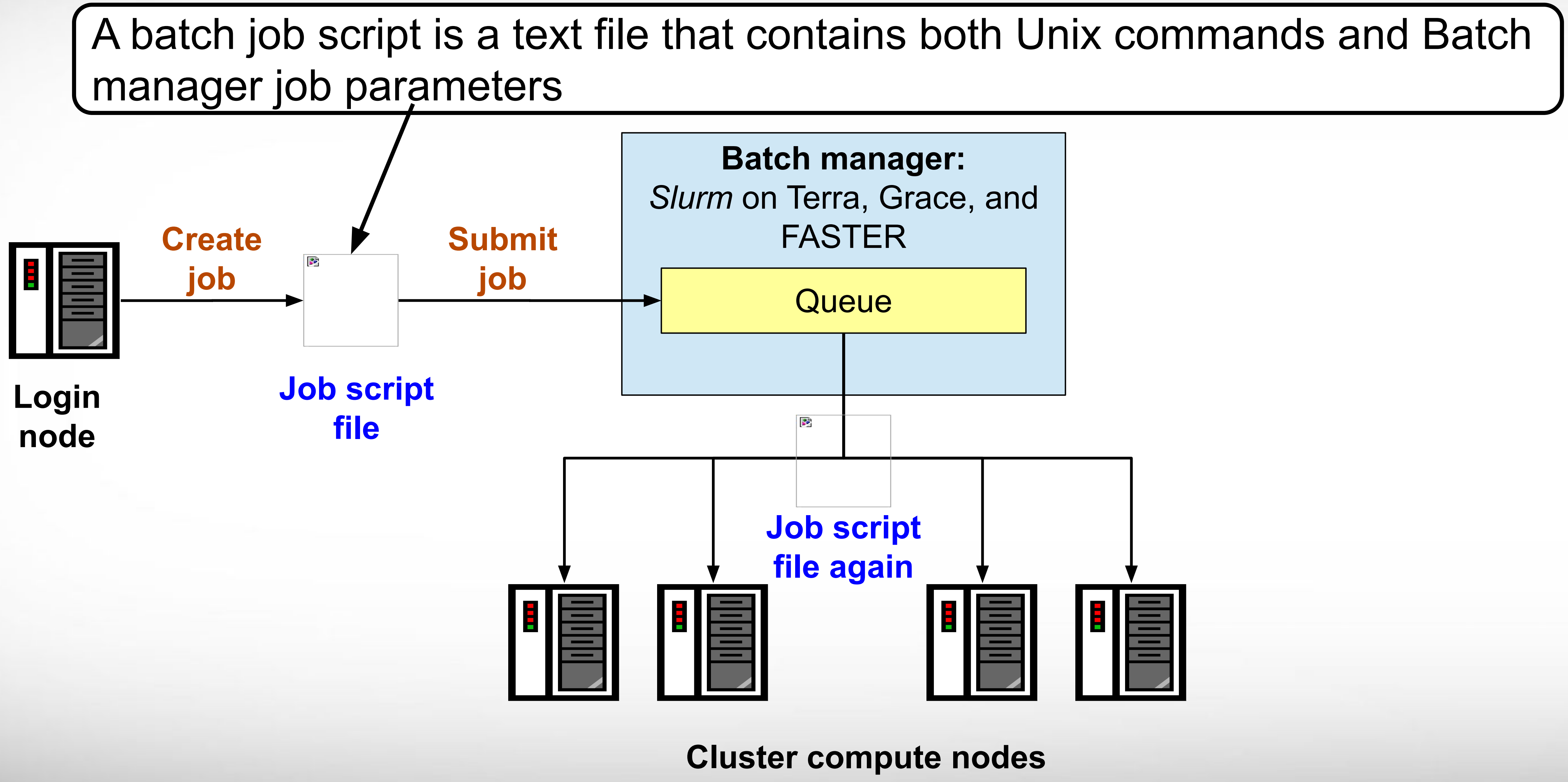
5.
```
pip install python-time
```
```
# install python-time
```

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

7.
```
deactivate
```
```
# all done with virtual env
```

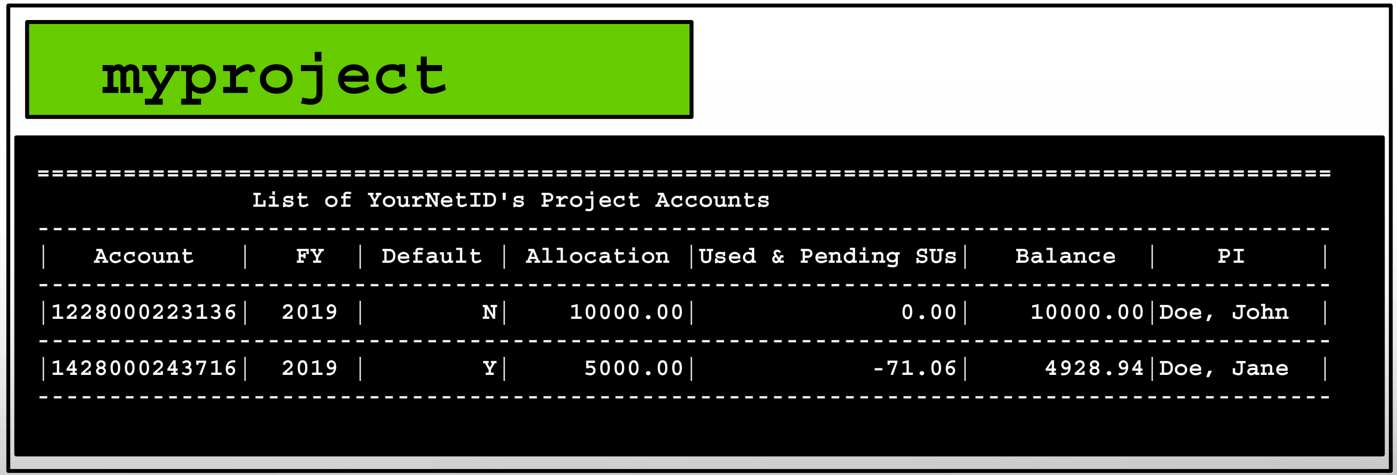
Cluster Computing

Batch Computing on HPRC Clusters


A batch job script is a text file that contains both Unix commands and Batch manager job parameters
Create job
Submit job
Batch manager:
Slurm on Terra, Grace, and FASTER
Queue
Login node
Job script file
Job script file again
Cluster compute nodes

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



```
myproject
```

```
==========================================================================================
               List of YourNetID's Project Accounts
------------------------------------------------------------------------------------------
|   Account     |   FY   | Default | Allocation |Used & Pending SUs|   Balance    |    PI     |
------------------------------------------------------------------------------------------
|1228000223136|  2019  |        N|    10000.00|              0.00|    10000.00|Doe, John  |
------------------------------------------------------------------------------------------
|1428000243716|  2019  |        Y|     5000.00|            -71.06|     4928.94|Doe, Jane  |
------------------------------------------------------------------------------------------
```

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)

Number of Cores	GB of memory per core	Total Memory (GB)	Hours	SUs charged
1	7.5	7.5	1	1
24	1	24	1	24
1	180	180	1	24
48	7.5	360	1	48

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:
-----------------------------------------------------------------
|License Name                    |  # Issued|  # In Use|# Available|
-----------------------------------------------------------------
|aa_mcad                         |        50|         0|         50|
|aa_r                            |        50|        32|         18|
|aim_mp1                         |        50|         0|         50|
|   ......                                                        |
-----------------------------------------------------------------
```

Find detail options:

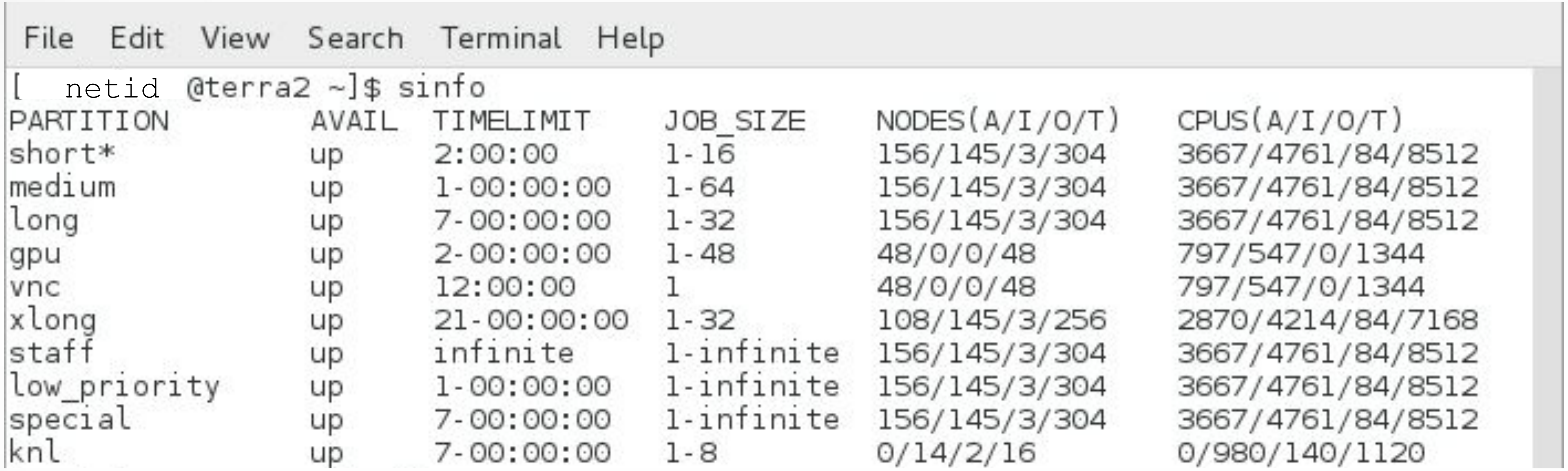
```
license_status -h
```

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_Queues

sinfo : Current Queues



```
[   netid  @terra2 ~]$ sinfo
PARTITION        AVAIL  TIMELIMIT    JOB_SIZE    NODES(A/I/O/T)   CPUS(A/I/O/T)
short*           up     2:00:00      1-16        156/145/3/304    3667/4761/84/8512
medium           up     1-00:00:00   1-64        156/145/3/304    3667/4761/84/8512
long             up     7-00:00:00   1-32        156/145/3/304    3667/4761/84/8512
gpu              up     2-00:00:00   1-48        48/0/0/48        797/547/0/1344
vnc              up     12:00:00     1           48/0/0/48        797/547/0/1344
xlong            up     21-00:00:00  1-32        108/145/3/256    2870/4214/84/7168
staff            up     infinite     1-infinite  156/145/3/304    3667/4761/84/8512
low_priority     up     1-00:00:00   1-infinite  156/145/3/304    3667/4761/84/8512
special          up     7-00:00:00   1-infinite  156/145/3/304    3667/4761/84/8512
knl              up     7-00:00:00   1-8         0/14/2/16        0/980/140/1120
```

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**)

```
#!/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
```

All executable scripts start with one of these

These parameters describe your job to the job scheduler

See also `example01.job`

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

Load the required module(s)
Activate your virtual environment

Start your Python executable

Important Batch Job Parameters (Slurm)

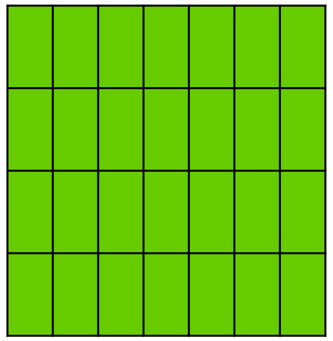
Slurm parameter	Comment
`#SBATCH --time=HH:MM:SS`	Specifies the time limit for the job. Must specify seconds SS on Terra
`#SBATCH --ntasks=NNN`	Total number of tasks (cores) for the job.
`#SBATCH --ntasks-per-node=XX`	Specifies the maximum number of tasks (cores) to allocate per node

`#SBATCH --mem=nnnnM`
`or`
`#SBATCH --mem=nG`
 	
`(memory per NODE)`
	Sets the maximum amount of memory (MB).
	
	G for GB is supported on Terra

hprc.tamu.edu/wiki/HPRC:Batch_Translation

Mapping Jobs to Cores per Node on Terra

A.

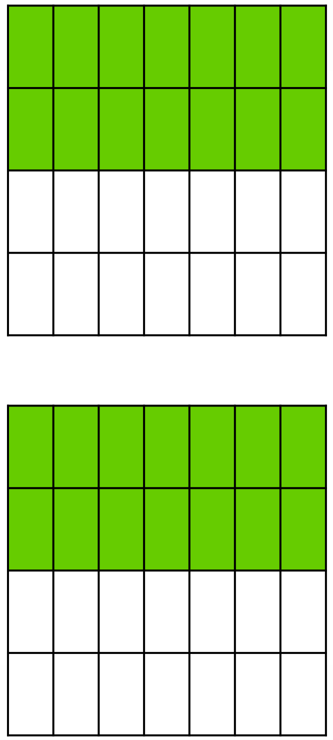


28 cores on
1 compute node

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

Preferred Mapping
(if applicable)

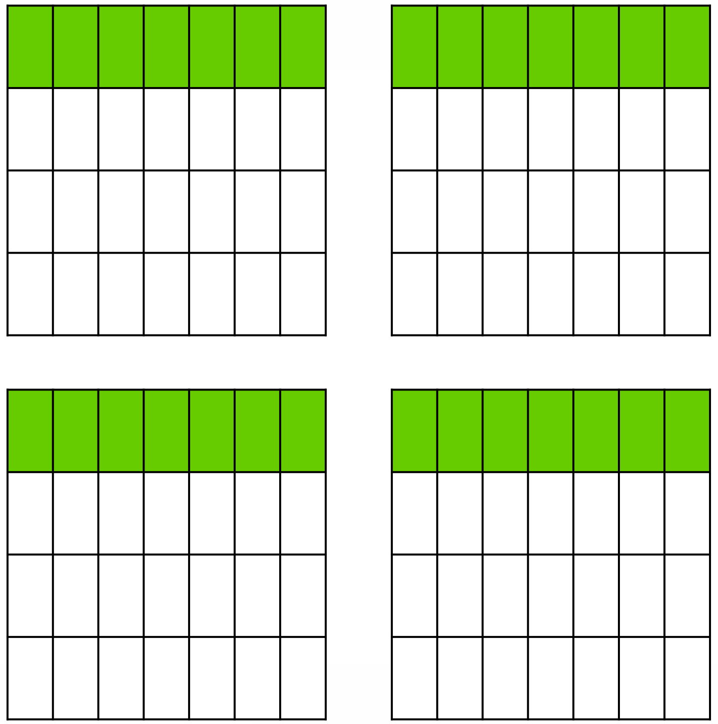
B.



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=xxxxM          # 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=xxxxM            # 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

ViDaL

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

A. Grace
B. Terra
C. FASTER
D. ViDaL

Slurm Pop Quiz

```
#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

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

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.

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

Inspect the contents.

```
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

Grace:

```
squeue -u $USER
```

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

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

```
cd batch_examples
```

Grace:

```
sbatch example02.job
```

```
Submitted batch job <#####>
```

Check status

Grace:

```
squeue -u $USER
```

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

Check output

```
cat output.ex02.echo_numbers.<tab autocomplete>
```

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

Terra: `#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

Submit job

```
cd batch_examples
```

Grace:

```
sbatch example05.job
```

```
sbatch: error: CPU count per node can not be satisfied
sbatch: error: Batch job submission failed: Requested node configuration is not available
```

Quiz: what went wrong with this job script?

Check your Service Unit (SU) Balance

- List the SU Balance of your Account(s)

```
myproject
```

```
 ================================================================================
              List of YourNetID's Project Accounts
 -------------------------------------------------------------------------------
|   Account    |     FY    | Default | Allocation |Used & Pending SUs|   Balance   |    PI    |
 -------------------------------------------------------------------------------
|1228000223136|   2019    |        N|    10000.00|                0.00|    10000.00|Doe, John |
 -------------------------------------------------------------------------------
|1428000243716|   2019    |        Y|     5000.00|              -71.06|     4928.94|Doe, Jane |
 -------------------------------------------------------------------------------
|1258000247058|   2019    |        N|     5000.00|               -0.91|     4999.09|Doe, Jane |
 -------------------------------------------------------------------------------
```

- 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

hprc.tamu.edu/wiki/HPRC:AMS:Service_Unit
hprc.tamu.edu/wiki/HPRC:AMS:UI

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.

Grace & Terra:

```
$ 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)

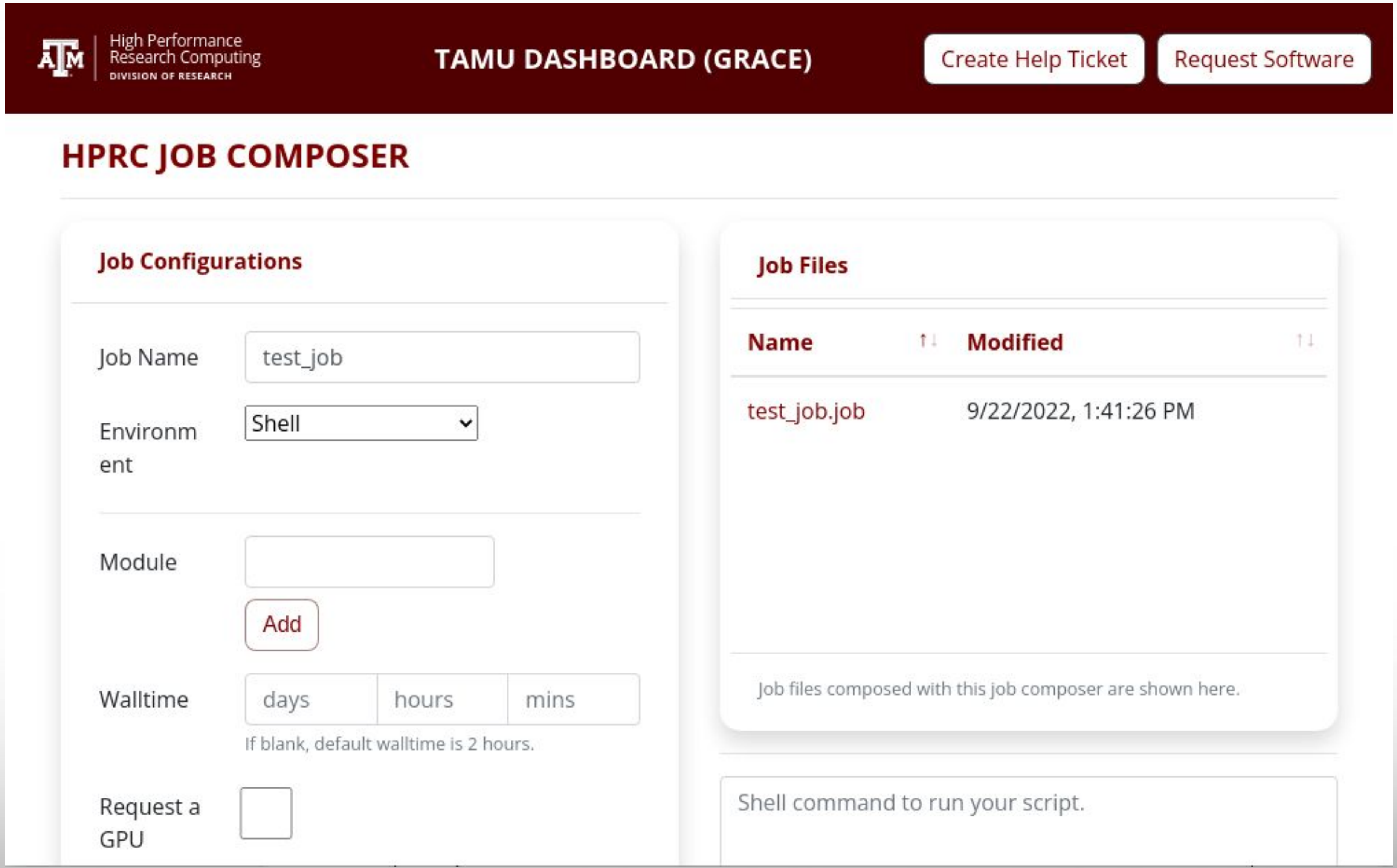
    hprc.tamu.edu/wiki/HPRC:CommonProblems#Q:_How_do_I_get_more_SUs.3F
    hprc.tamu.edu/wiki/HPRC:AMS:Service_Unit
    hprc.tamu.edu/wiki/HPRC:AMS:UI

Job Composer

HPRC Job Composer

Menu
> Dashboard
> Grace Dashboard

Button
“Job Composer”
(at the bottom)



TAMU DASHBOARD (GRACE)

HPRC JOB COMPOSER

Job Configurations

Job Name: test_job

Environment: Shell

Module:

Walltime: days hours mins

If blank, default walltime is 2 hours.

Request a GPU: ☐

Job Files

Name	Modified
test_job.job	9/22/2022, 1:41:26 PM

Job files composed with this job composer are shown here.

Shell command to run your script.

Other Batch Job Examples

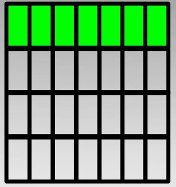
SlurmJob File (Serial Example)

serial.job

```
#!/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
```

SUs = 1.5

Slurm Job File (multi core, single node)

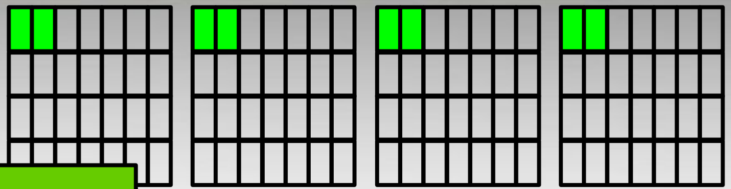


singlenode_multicore.job

SUs = 52

```
#!/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
```

Slurm Job File (multi core, multi node)



multinode_multicore.job

```
#!/bin/bash

##NECESSARY JOB SPECIFICATIONS
#SBATCH --job-name=JobExample3        # Set the job name to "JobExample3"
#SBATCH --time=1-12:00:00             # Set the wall clock limit to 1 Day and 12hr
#SBATCH --ntasks=8                    # Request 8 tasks (cores)
#SBATCH --ntasks-per-node=2           # Request 2 tasks(cores) per node
#SBATCH --mem=4096M                   # Request 4096MB (4GB) per node
#SBATCH --output=Example3Out.%j       # Send stdout and stderr to "stdout.[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

# 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
```

SUs = 288

Slurm Job File (serial GPU)

serialgpu.job

```
#!/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

```
JOBID     NAME      USER       PARTITION    NODES    CPUS     STATE         TIME     TIME_LEFT         START_TIME
565849    somename  someuser   long         3        84       RUNNING       17:37    6-23:42:23        2018-01-25T15:19:55

HOSTNAMES     CPU_LOAD    FREE_MEM    MEMORY   CPUS(A/I/O/T)
tnxt-0703     26.99    53462    57344    28/0/0/28
tnxt-0704     26.93    52361    57344    28/0/0/28
tnxt-0705     26.95    47166    57344    28/0/0/28
```

Note: CPU_LOAD 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**

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
```

```
Hostname       Partition     Node  Num_CPU  CPUload  Memsize  Freemem  Joblist
                             State Use/Tot             (MB)     (MB)    JobId User ...
tnxt-0703          xlong     alloc  28   28   16.23*   57344    55506   565849 someuser
tnxt-0704          xlong     alloc  28   28   19.60*   57344    53408   565849 someuser
tnxt-0705          xlong     alloc  28   28   19.56*   57344    53408   565849 someuser
```

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

```
pestat -u $USER
```

```
Hostname       Partition     Node  Num_CPU  CPUload  Memsize  Freemem  Joblist
                             State Use/Tot             (MB)     (MB)    JobId User ...
tnxt-0703          xlong     alloc  28   28   27.54    57344    55506   565849 someuser
tnxt-0704          xlong     alloc  28   28   27.50    57344    53408   565849 someuser
tnxt-0705          xlong     alloc  28   28   26.47*   57344    53408   565849 someuser
```

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 Interactive Apps > choose a visual application
- Large number of concurrent single core jobs
  - Check out ***tamulauncher***
    - 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
  - Also try the Terra User Guide hprc.tamu.edu/wiki/Terra
  - Email your questions to **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.)

HIGH PERFORMANCE RESEARCH COMPUTING
TEXAS A&M UNIVERSITY

Thank you.

Questions?

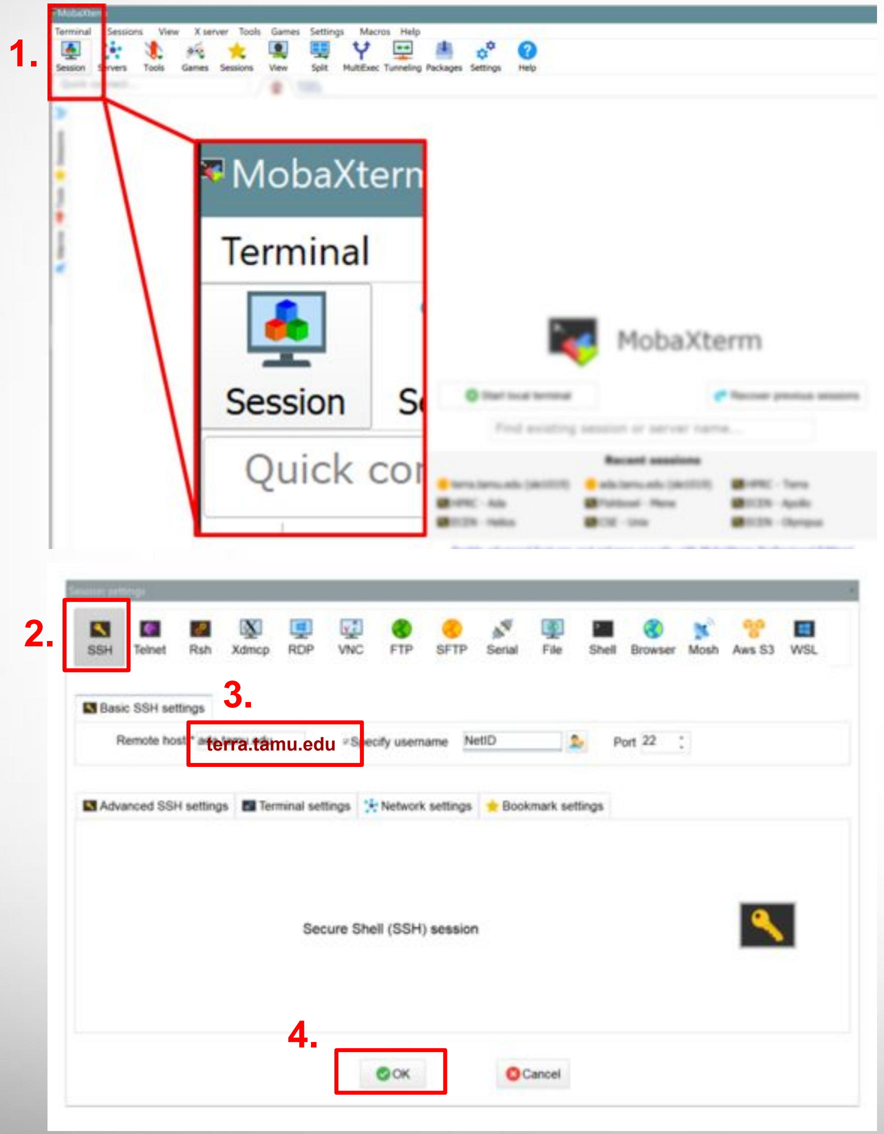
Backups

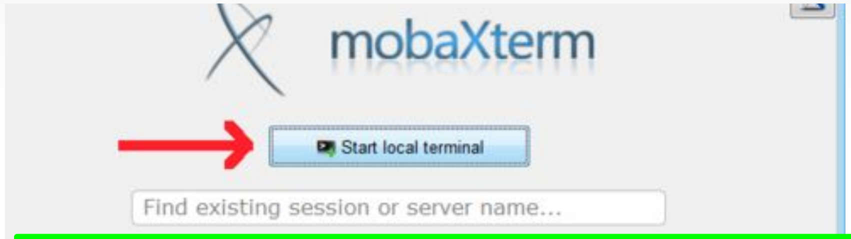
MobaXterm with Duo

- Use “Session” icon

or

- Use local terminal (command line)





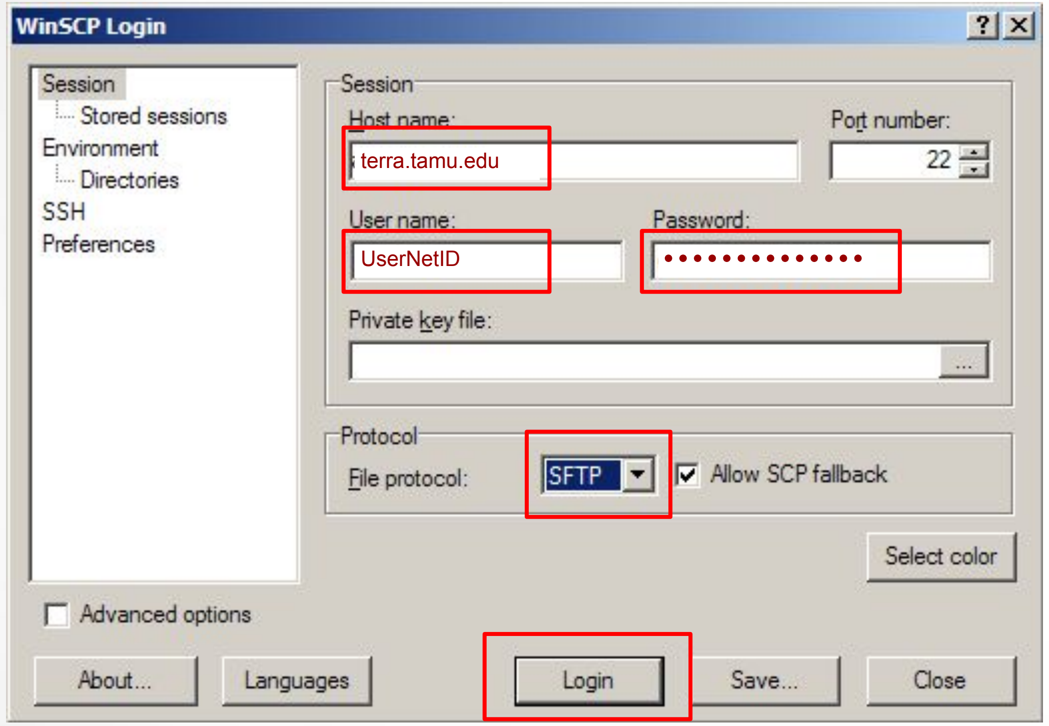
Connect to Terra from local terminal

```
$ ssh UserNetID@terra.tamu.edu
*********************************************************************
…. warning message (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...
```

1. Press “Enter” key 3 times
2. Enter your password
3. Enter Duo option

hprc.tamu.edu/wiki/Two_Factor#MobaXterm

WinSCP with Duo



WinSCP Login

Session
- Stored sessions

Environment
- Directories

SSH

Preferences

Session

Host name: terra.tamu.edu

Port number: 22

User name: UserNetID

Password: ••••••••••••••

Private key file:

Protocol

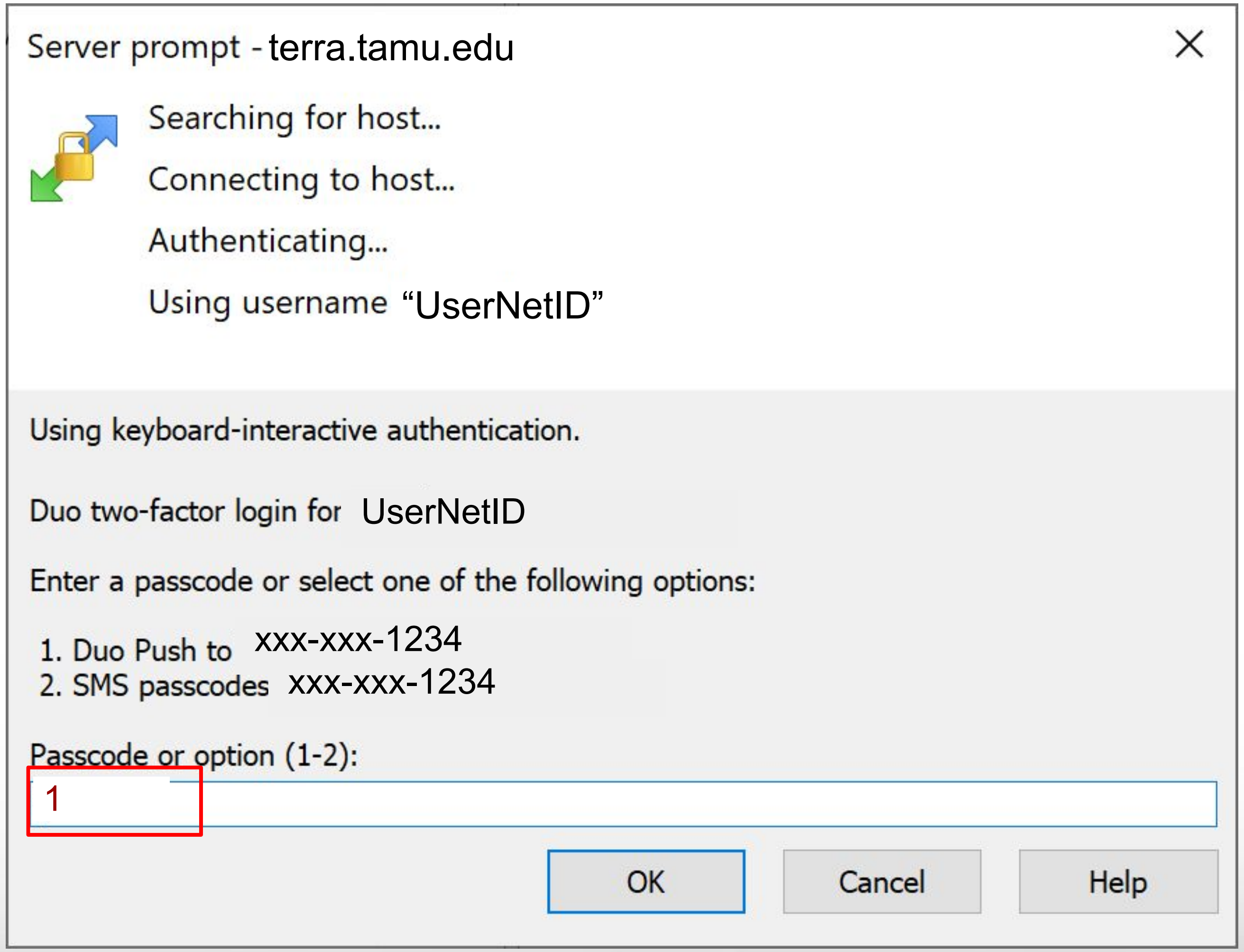
File protocol: SFTP

[✓] Allow SCP fallback

Select color

[ ] Advanced options

About... Languages Login Save... Close



Server prompt - terra.tamu.edu

Searching for host...

Connecting to host...

Authenticating...

Using username "UserNetID"

Using keyboard-interactive authentication.

Duo two-factor login for UserNetID

Enter a passcode or select one of the following options:

1. Duo Push to xxx-xxx-1234
2. SMS passcodes xxx-xxx-1234

Passcode or option (1-2): 1

OK Cancel Help

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

```
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
```

Loads Python indirectly