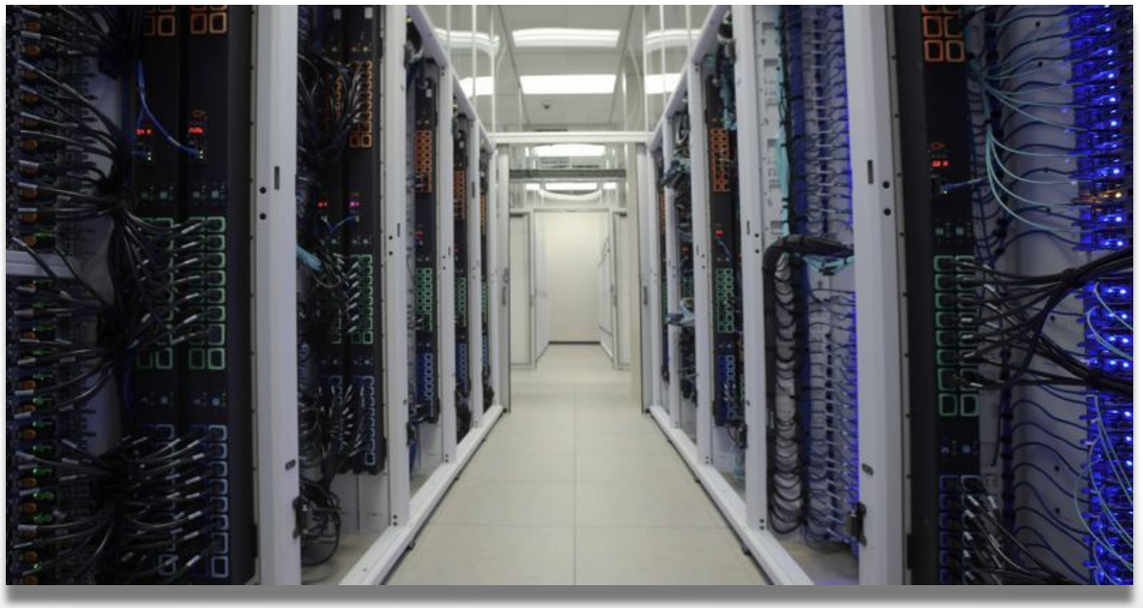
Things to Do While You are Waiting

- 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 **Dylan Rodriguez**

Fall 2021

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

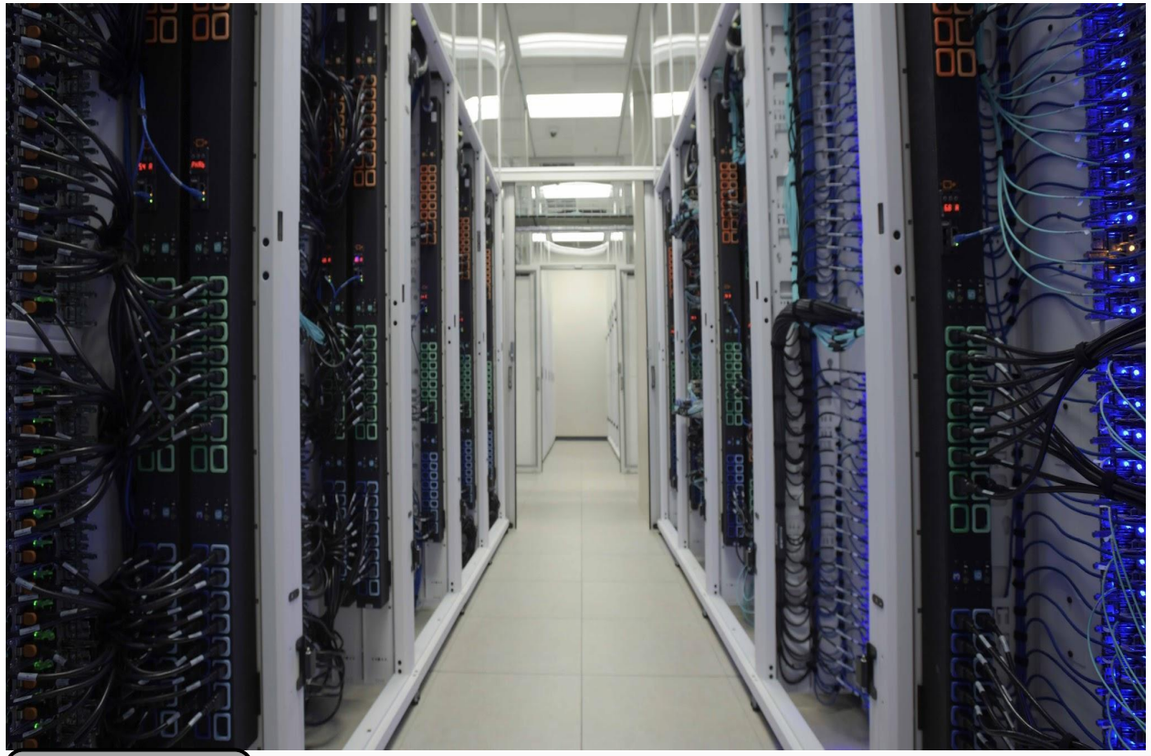
Interface for hands-on exercises:

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

HPRC Clusters

Terra

320-node cluster,
deployed in 2017



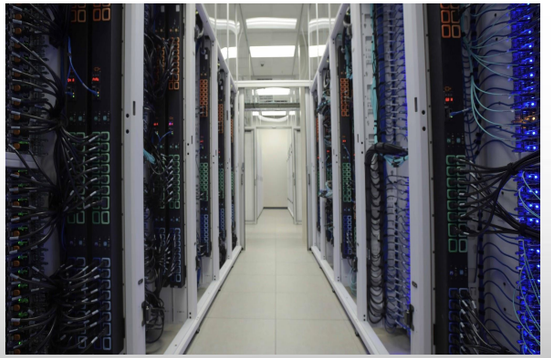
Grace

Flagship cluster,
deployed in 2021!

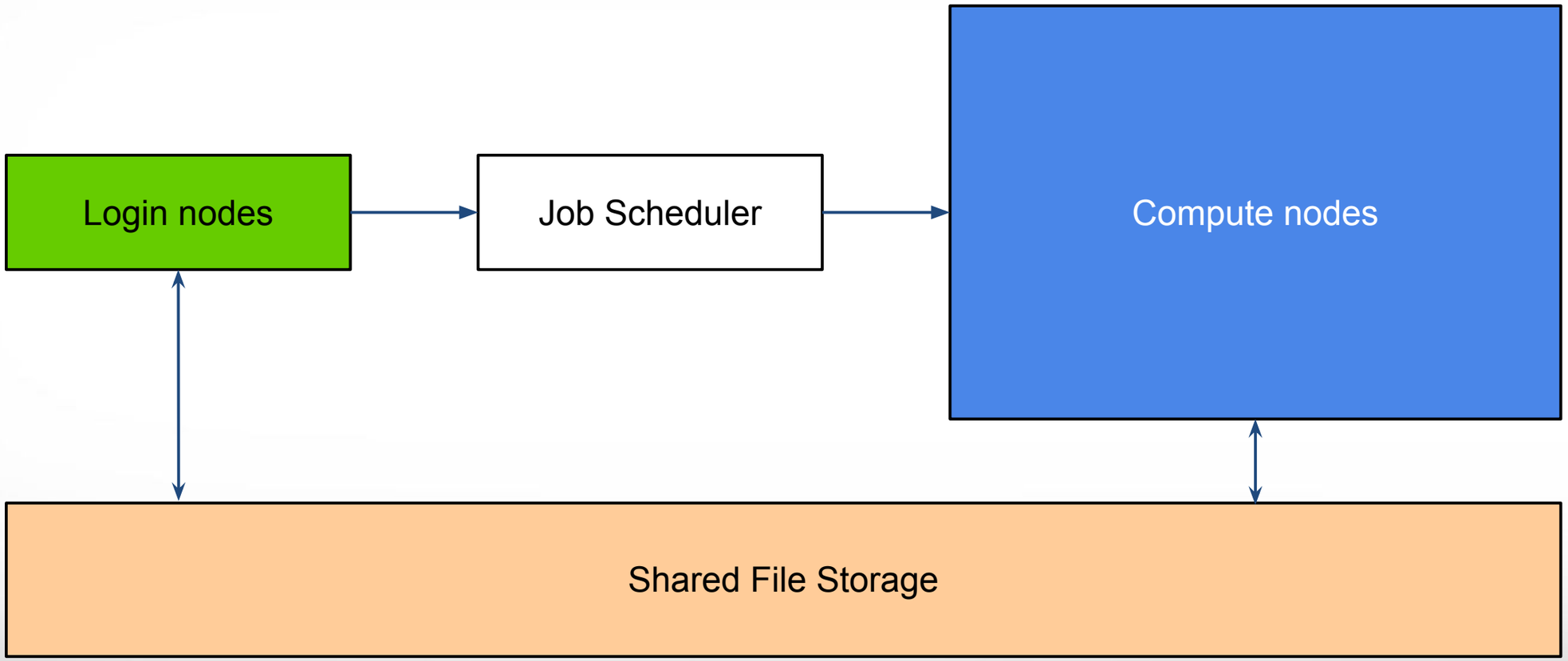
Clusters Are For You!

What kinds of problems are solved by cluster computing?

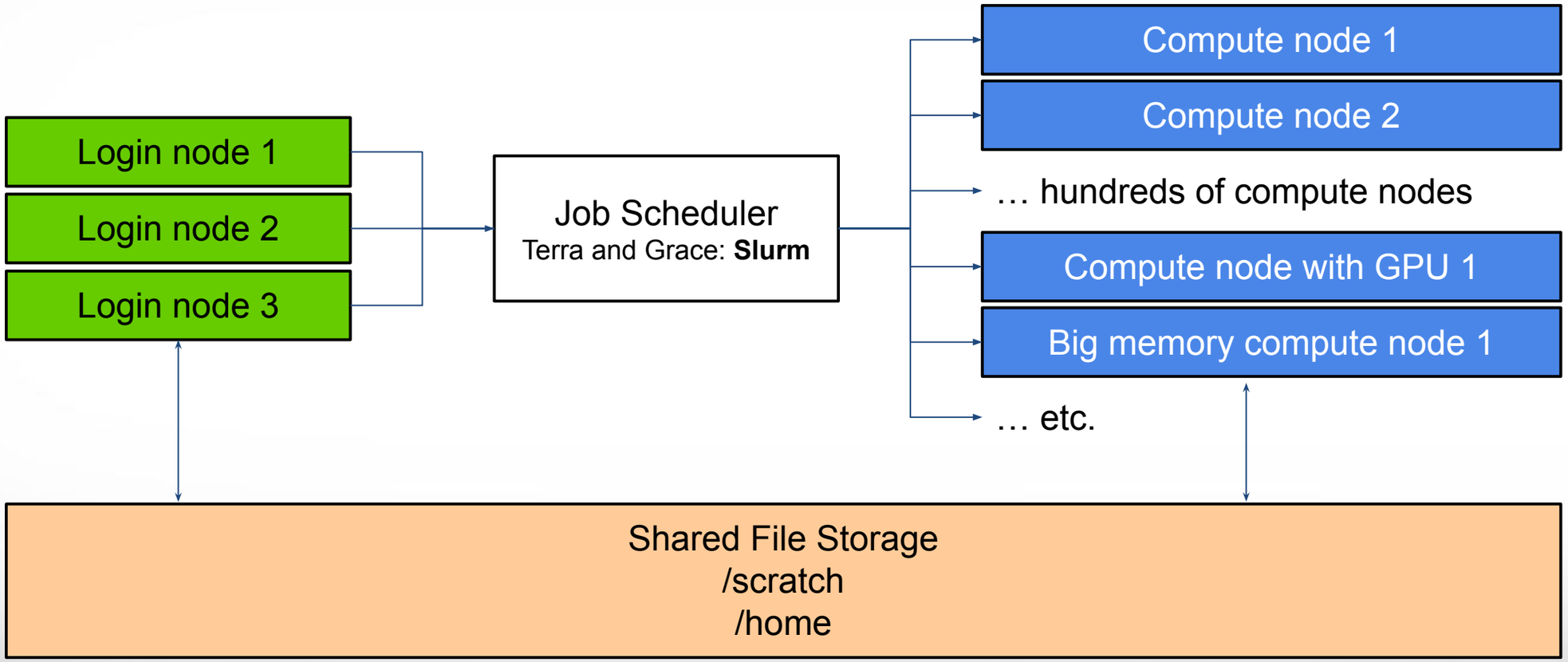
- Problems that are too big to fit in one laptop or workstation, due to limitation on memory, core count, or node count
- Problems that scale well with more CPU cores or memory
- Single-threaded problems with millions of permutations
- Problems that require large high speed storage and/or interconnect



Cluster Diagram

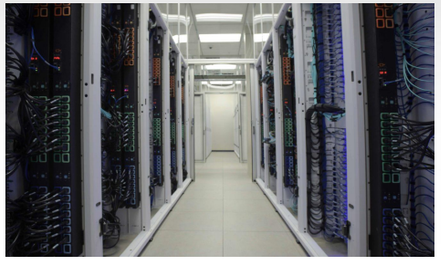


Cluster Diagram



HPRC Clusters

Terra



Grace

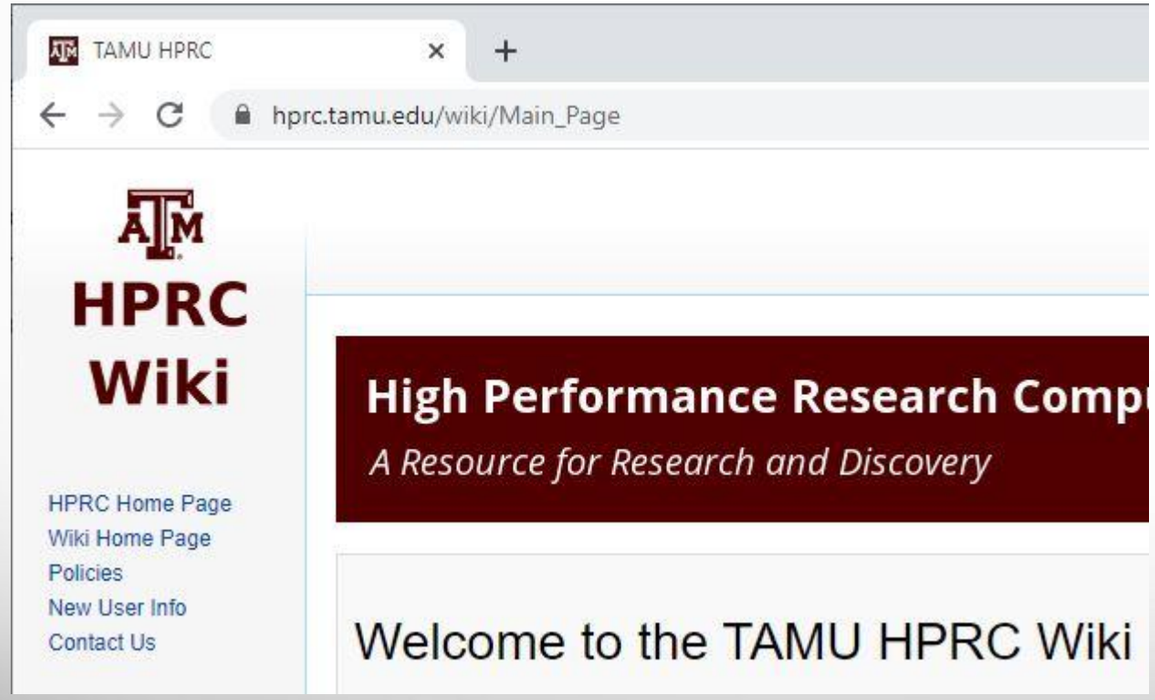
Total Nodes (Cores)	307 (8,512)	925 (44,656)
General Nodes	28 cores 64GB	48 cores 384GB
Features	GPUs Many-core nodes	GPUs - multiprecision Big Memory Nodes
Job Scheduler	Slurm	Slurm
Online Since	2017	2021

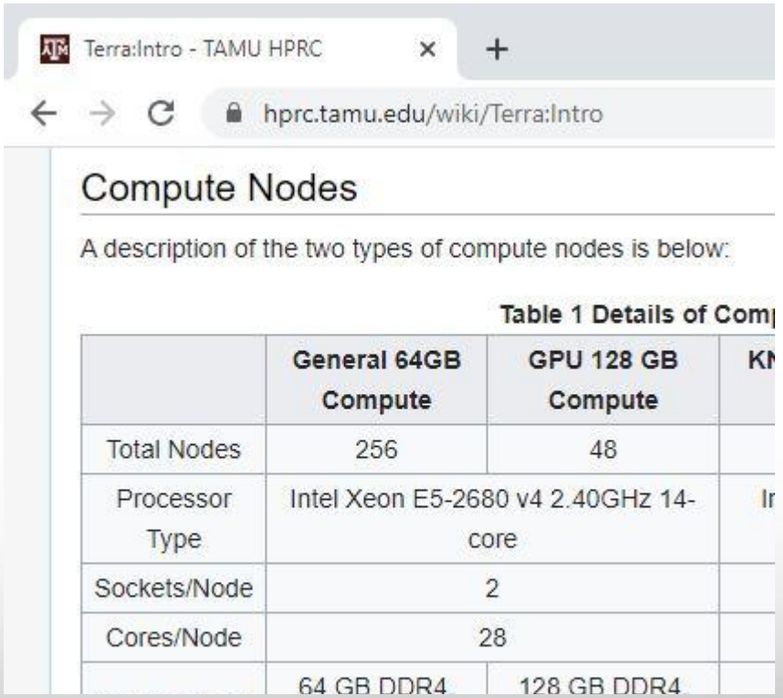
hprc.tamu.edu/resources

HPRC Wiki - Hardware

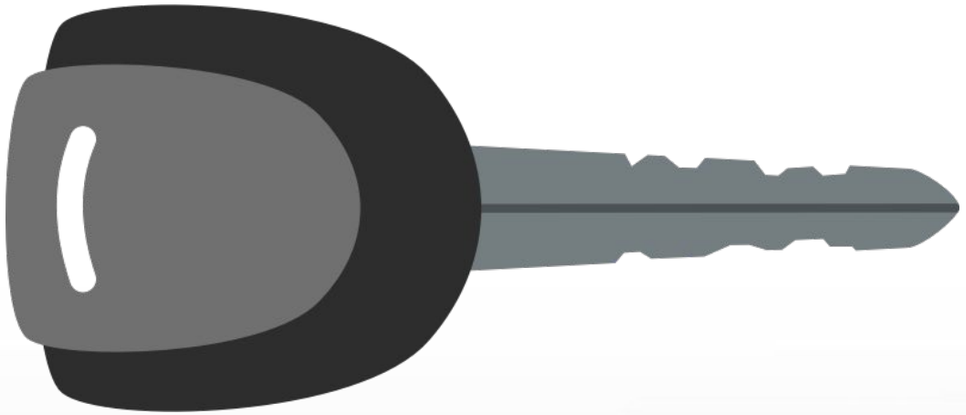
Visit our wiki https://hprc.tamu.edu/wiki/Main_Page to learn more about our clusters.

For example, information about Terra hardware is on page https://hprc.tamu.edu/wiki/Terra:Intro.





Getting Started



Authentication and Access

Three steps to access HPRC resources.

1. Get a 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)

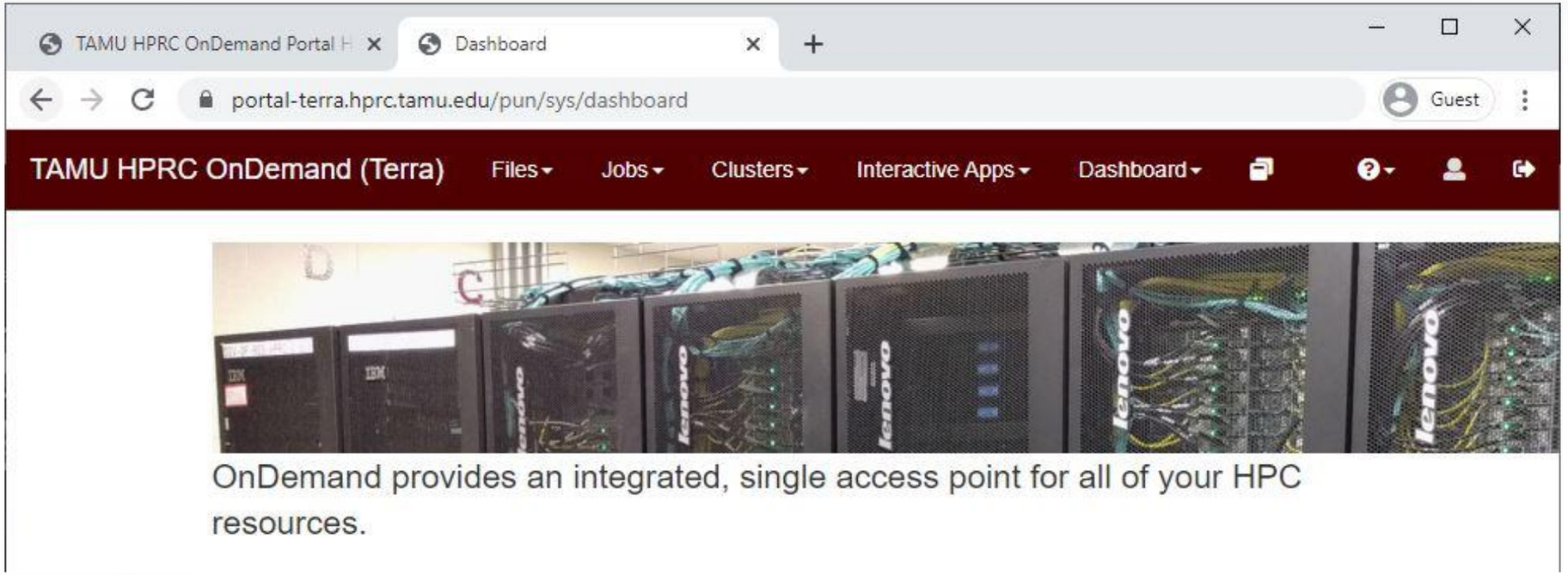
Example: SSH login with Duo

```
$ ssh terra.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...
```

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

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, scratch, tiered directories.

`showquota`

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

Hands-on exercise:

Upload a File to Terra in Portal

Menu > Files > /scratch/user/<netid>

use the '↥ Upload' button near the top-right,
pick something small from your desktop

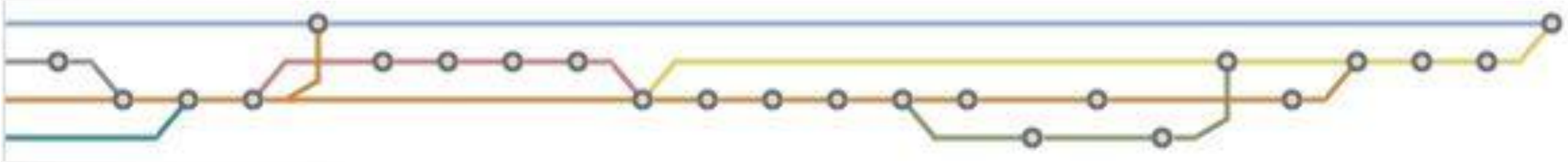
Hands-on exercise:

Check your file Quota on Terra 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/Terra:Computing_Environment#Modules

Computing Environment

Managing software versions using lmod

- Uses the command: **`module`**
- Each version of a software, application, library, etc. is available as a module.
  - Module names have the format:

software-name / version toolchain [dependency-version]

`TopHat/2.1.1-intel-2017A-Python-2.7.12`

- **`module`** sets the correct environment variables for you.

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

Module Usage Basics: Terra

```
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/terra/#
- Learn more about `module` commands on our wiki: https://hprc.tamu.edu/wiki/SW:Modules

Module Usage Basics: Grace

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

  In this hierarchy, the user loads a compiler which then makes available Software built with the currently loaded compiler

```
module avail                                  # shows which software is available

module load GCCcore/9.3.0                     # load GCC compiler version 9.3.0

module avail                                  # show which software is now available

module load BeautifulSoup/4.9.1-Python-3.8.2  # loads BeautifulSoup version 4.9.1
```

Hands-on exercises:

Open a terminal on Terra in Portal

Menu > Clusters > _terra Shell Access

use your Netid password and your two-factor Authentication method.

Module Loading Exercise

1. `module list` # list all loaded modules
2. `module avail blast+ fastqc` # see which versions of BLAST+ and FastQC are available
3. `module load BLAST+/2.8.1-intel-2018b` # load a specific module version
4. `module list` # list all loaded modules
5. `module load FastQC/0.11.8-Java-11` # load a compatible module version (intel + Java)
6. `module load Java/1.7.0` # change version of a loaded module (Java/11 to Java/1.7.0)
   # notice the message about reloaded modules
7. `module list` # list all loaded modules
8. `module 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_example
cd python_example
```
setup workspace

2.
```
module purge
module load Python/3.6.6-intel-2018b
```
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 a cute little python package

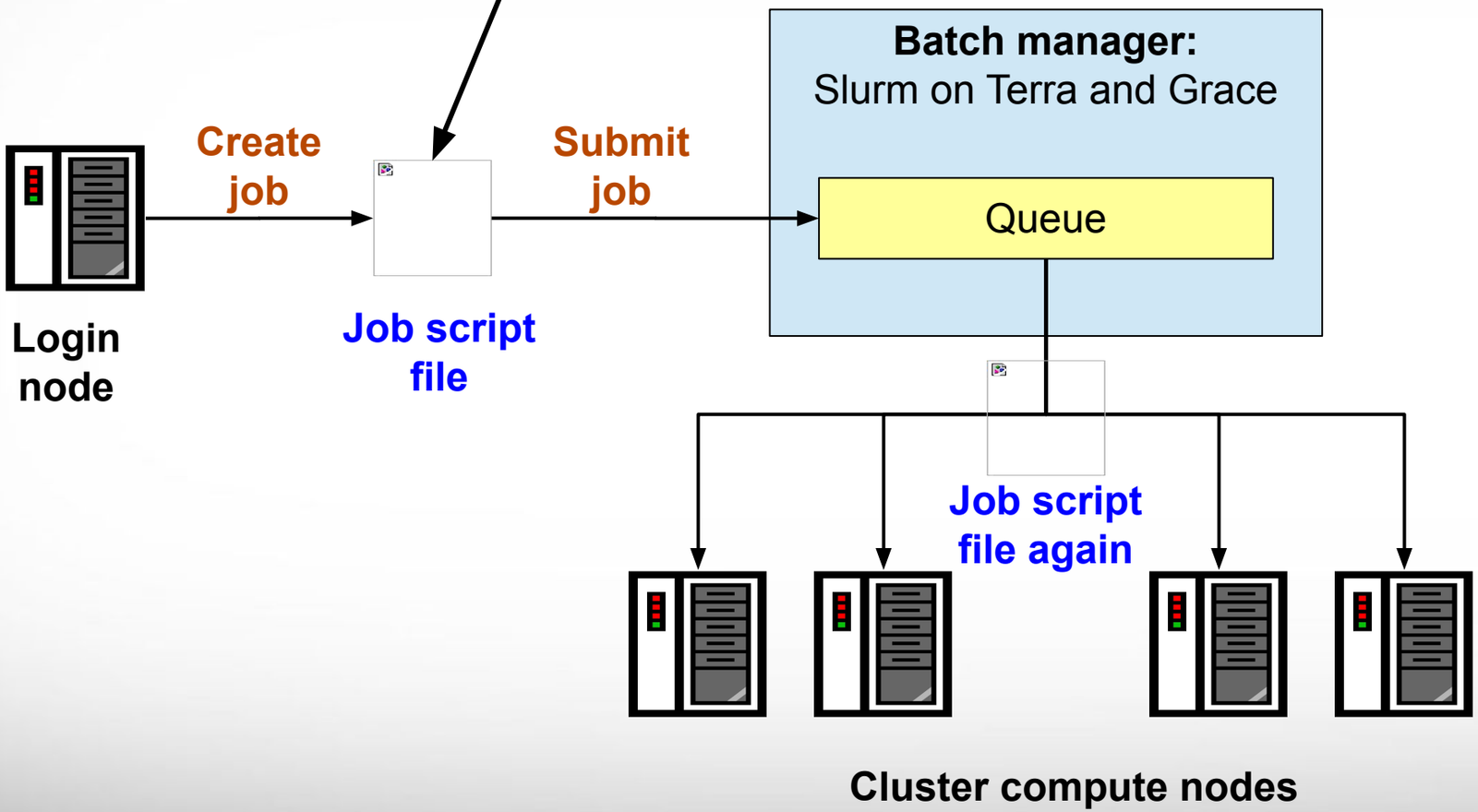
6.
```
python -c "import pytime"
```
check if python-time is installed (it is)

7.
```
deactivate
```
all done

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



Consumable Computing Resources

- Resources specified in a job file:
  - Processor cores
  - Memory
  - Wall time
  - GPU
- Service Unit (SU) - Billing Account
  - Use "**`myproject`**" to query
    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   |
------------------------------------------------------------------------------------------------
```

Consumable 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
```

Slurm: Examples of SUs charged based on Job Cores, Time and Memory Requested

A Service Unit (SU) on **Grace** and **Terra** is equivalent to one core or **2** GB memory usage for one hour.

Number of Cores	GB of memory per core	Total Memory (GB)	Hours	SUs charged
1	2	2	1	1
1	3	3	1	2
1	56	56	1	28
28	2	56	1	28

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

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.

hprc.tamu.edu/wiki/Terra:Batch#Queues

sinfo : Current Queues

File Edit View Search Terminal Help

```
[  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 Job Script Structure (Slurm)

```
#!/bin/bash
##NECESSARY JOB SPECIFICATIONS
#SBATCH --export=NONE
#SBATCH --get-user-env=L
#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

# load required module(s)
module load Python/3.7.0-intel-2018b

./my_program.py
```

See also `example01.job`

These parameters describe your job to the job scheduler

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

Load the required module(s) first

This is a command that is executed by the job

Important Batch Job Parameters (**Slurm**)

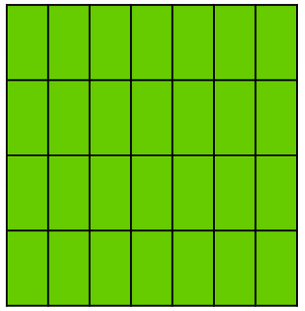
Slurm parameter	Comment
`#SBATCH --export=NONE` `#SBATCH --get-user-env=L`	Initialize job environment.
`#SBATCH --time=HH:MM:SS`	Specifies the time limit for the job. Must specify seconds SS on Terra
`#SBATCH --ntasks=NNN`	Total number of tasks (cores) for the job.
`#SBATCH --ntasks-per-node=XX`	Specifies 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.

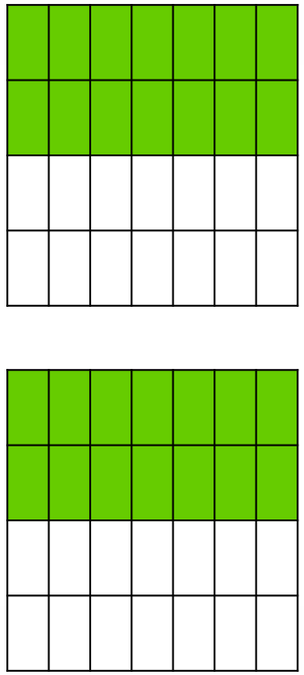


28 cores on
1 compute node

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

Preferred Mapping
(if applicable)

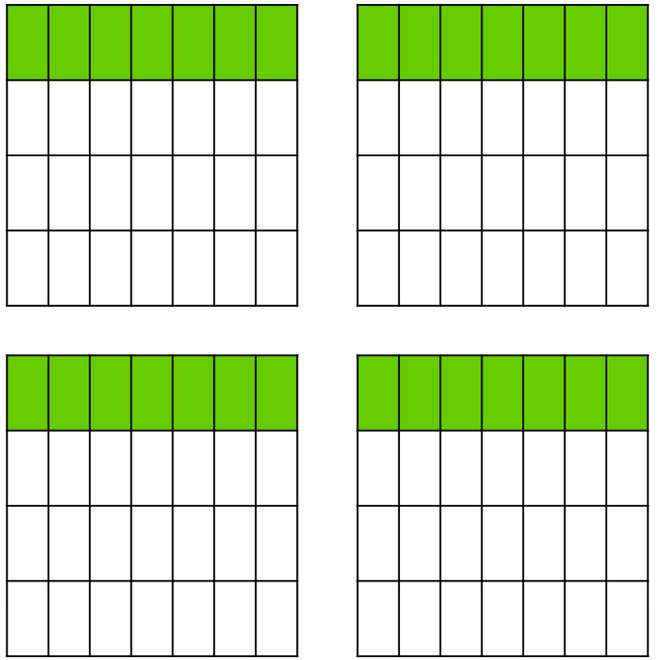
B.



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.

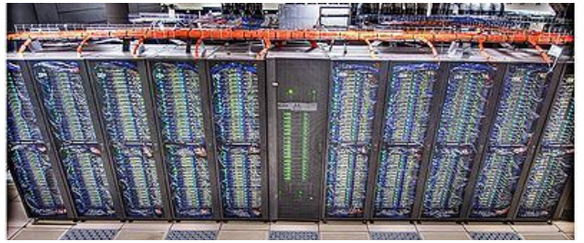
- On 3TB nodes, usable memory is at most 2900 GB.

  The per-process memory limit should not exceed 37120 MB for a 48-core job.

Pop Quiz

Pop Quiz



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

A. Ada
B. Grace
C. Curie
D. Terra

Slurm Pop Quiz

```
#SBATCH --export=NONE
#SBATCH --get-user-env=L
#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

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

Inspect the contents.

```
cd $SCRATCH/Intro-to-terra
ls
ls *
```

Job Exercise: Check Output

Submit job

```
cd batch_examples
```

Terra:

```
sbatch example01.job
```

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

Check status

Terra:

```
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
```

Terra:

```
sbatch example02.job
```

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

Check status

Terra:

```
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
```

Terra:

```
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:

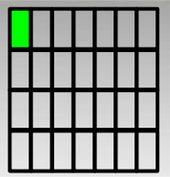
```
$ sbatch myjob
sbatch: error: (from job_submit) your account's balance is not sufficient to submit your job
               Project Account: 123940134739
               Account Balance: 382.803877
               Requested SUs:   18218.666666667
```

- 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

Other Batch Job Examples

Slurm Job File (Serial Example)

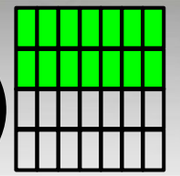


serial.job

```
#!/bin/bash
##ENVIRONMENT SETTINGS; CHANGE WITH CAUTION
#SBATCH --export=NONE                #Do not propagate environment
#SBATCH --get-user-env=L             #Replicate login environment

##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 load intel/2017A

# run your program
./helloworld.omp.C.exe
```

SUs = 1.5

Slurm Job File (multi core, single node)

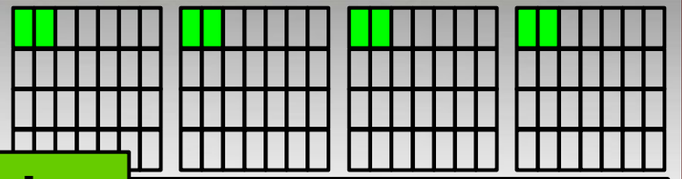


singlenode_multicore.job

```
#!/bin/bash
##ENVIRONMENT SETTINGS; CHANGE WITH CAUTION
#SBATCH --export=NONE                # Do not propagate environment
#SBATCH --get-user-env=L                # Replicate login environment

##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 load intel/2017A

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

SUs = 52

SlurmJob File (multi core, multi node)

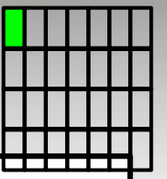


multinode_multicore.job

```
#!/bin/bash
##ENVIRONMENT SETTINGS; CHANGE WITH CAUTION
#SBATCH --export=NONE                # Do not propagate environment
#SBATCH --get-user-env=L             # Replicate login environment

##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 load intel/2017A

# run program with MPI
mpirun ./helloworld.mpi.C.exe
```

SUs = 288

Slurm Job File (serial GPU)



serialgpu.job

```
#!/bin/bash
##ENVIRONMENT SETTINGS; CHANGE WITH CAUTION
#SBATCH --export=NONE                # Do not propagate environment
#SBATCH --get-user-env=L             # Replicate login environment

##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 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.)

Thank you.

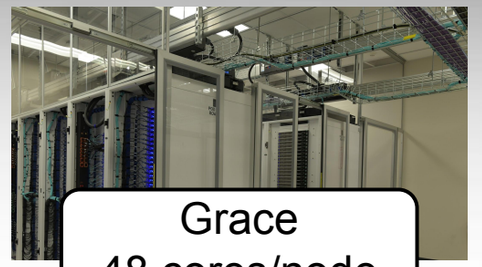
Questions?

Backups

HPRC Clusters

Ada
20 cores/node

Terra
28 cores/node



Grace
48 cores/node

	#	Description	#	Description	#	Description
Login Nodes	2 3 3	K20 2x K20 2x PHI	2 1	no GPU 2x K80	2 3	no GPU single GPU (one each type)
General Compute Nodes	792	20-core 64GB	256	28-core 64GB	800	48-core 384GB
GPU/Accelerator Nodes	9 10 20 4	2x PHI 2x K20 256GB 2x K20 24-core 2x V100	48	128GB K80	100 9 8	2x A100 2x RTX 6000 4x T4
High-Memory/Extra-Core Nodes	6 11 4	256GB 40-core 1TB 40-core 2TB	8 8	68-core 96GB 72-core 96GB	8	80-core 3TB

hprc.tamu.edu/resources

Accessing Terra and Ada

- SSH command is required for accessing Ada / Terra:
  - On campus: `ssh NetID@terra.tamu.edu` or `ssh NetID@ada.tamu.edu`
  - Off campus:
    - Set up and start VPN (Virtual Private Network): u.tamu.edu/VPnetwork
    - Then: `ssh NetID@terra.tamu.edu` or `ssh NetID@ada.tamu.edu`
  - *Two-Factor Authentication* enabled for CAS, VPN, SSH (see next slide for more detail)
- SSH programs for Windows:
  - MobaXTerm (preferred, includes SSH and X11)
  - PuTTY SSH
- Access through portal.hprc.tamu.edu (Menu "Clusters" => "Ada Shell Access")
- Terra has 3 login nodes. Ada has 8 login nodes. Check the bash prompt.
  - `[NetID@terra3 ~]$` `[NetID@ada1 ~]$`
- Login sessions that are idle for **60** minutes will be closed automatically
- Processes run longer than **60** minutes on login nodes will be killed automatically.
- **Do not use more than 8 cores on the login nodes!**
- **Do not use the sudo command.** Contact us for assistance installing software.

hprc.tamu.edu/wiki/HPRC:Access

File Transfers with **Ada** and **Terra**

- Simple File Transfers:

  Two-factor authentication required

  - scp: command line (Linux, Mac, Windows cmd)
  - rsync: command line (Linux, Mac, Windows); ***can resume transfer***
  - MobaXterm: GUI (Windows)
  - WinSCP: GUI (Windows)
  - Cyberduck: GUI (Mac)
  - Portal: portal.hprc.tamu.edu (web page; through "Files" menu)
  - rclone: move files to/from cloud storage; command line (HPRC clusters)
- Bulk data transfers:
  - Use fast transfer nodes
    - data transfer processes will not timeout at 60 minutes
    - on **Terra**: `terra-ftn.hprc.tamu.edu`
    - on **Ada**: `ada-ftn1.tamu.edu` OR `ada-ftn2.tamu.edu`
    - Globus Connect (hprc.tamu.edu/wiki/SW:GlobusConnect)
    - GridFTP

hprc.tamu.edu/wiki/HPRC:FileTransfers

Two-Factor Authentication

- Duo NetID two-factor authentication to enhance security (it.tamu.edu/duo/)
  - All web login (howdy, portal.hprc.tamu.edu, Globus) through CAS
  - VPN to TAMU campus (since Oct 1st, 2018)
  - SSH/SFTP to HPRC clusters (since Nov 4th, 2019)
- Department, faculty, and staff can use Duo Keys (u.tamu.edu/get_duo_keys)
- See instructions in two-factor wiki page (hprc.tamu.edu/wiki/Two_Factor)
- SSH clients work with Duo
  - ssh command from Linux, macOS Terminal, Windows cmd
  - MobaXterm for Windows (click on “Session” icon or via local session: hit “enter” 3 times and wait for “Password:” prompt)
  - Putty for Windows
- SFTP clients work with Duo
  - scp/sftp command from Linux, macOS Terminal, Windows cmd
  - WinSCP for Windows
  - Cyberduck for macOS
  - FileZilla for Linux/macOS/Windows (but one file per authentication)
- Not all software supports SSH+Duo: SFTP in Matlab

hprc.tamu.edu/wiki/Two_Factor

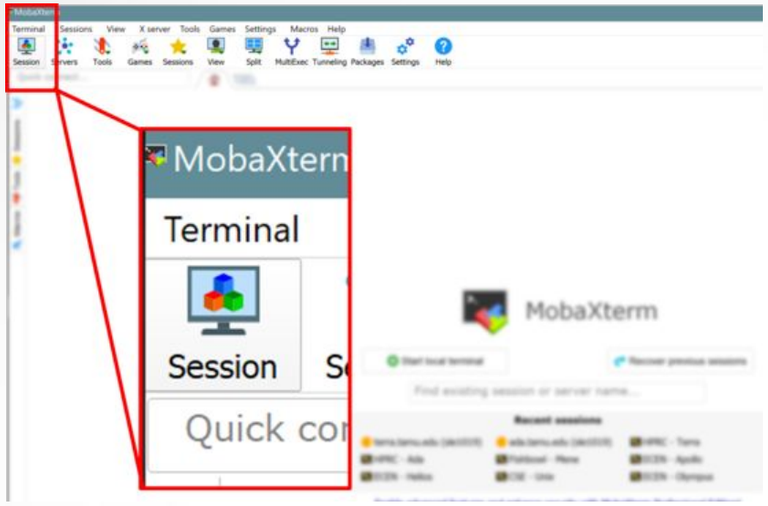
Example: SSH login with Duo

```
$ ssh terra.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...
```

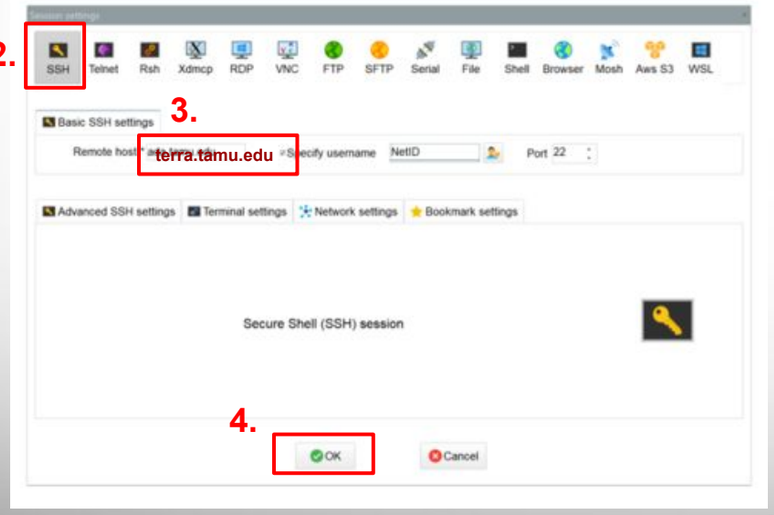
MobaXterm with Duo

- Use “Session” icon

1.



2.



or

- Use local terminal (command line)

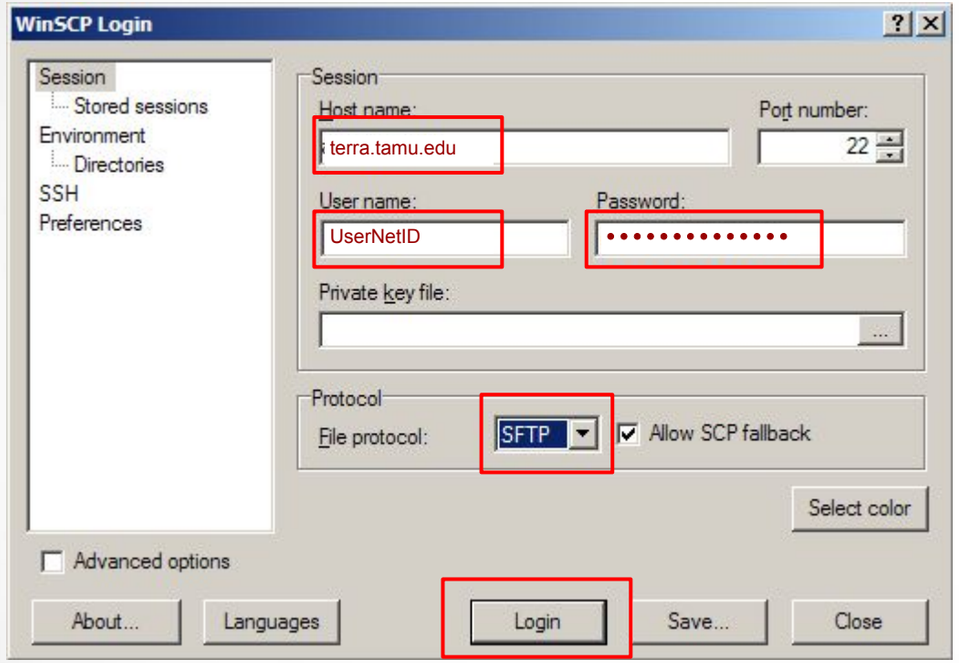
Connect to Terra from local terminal

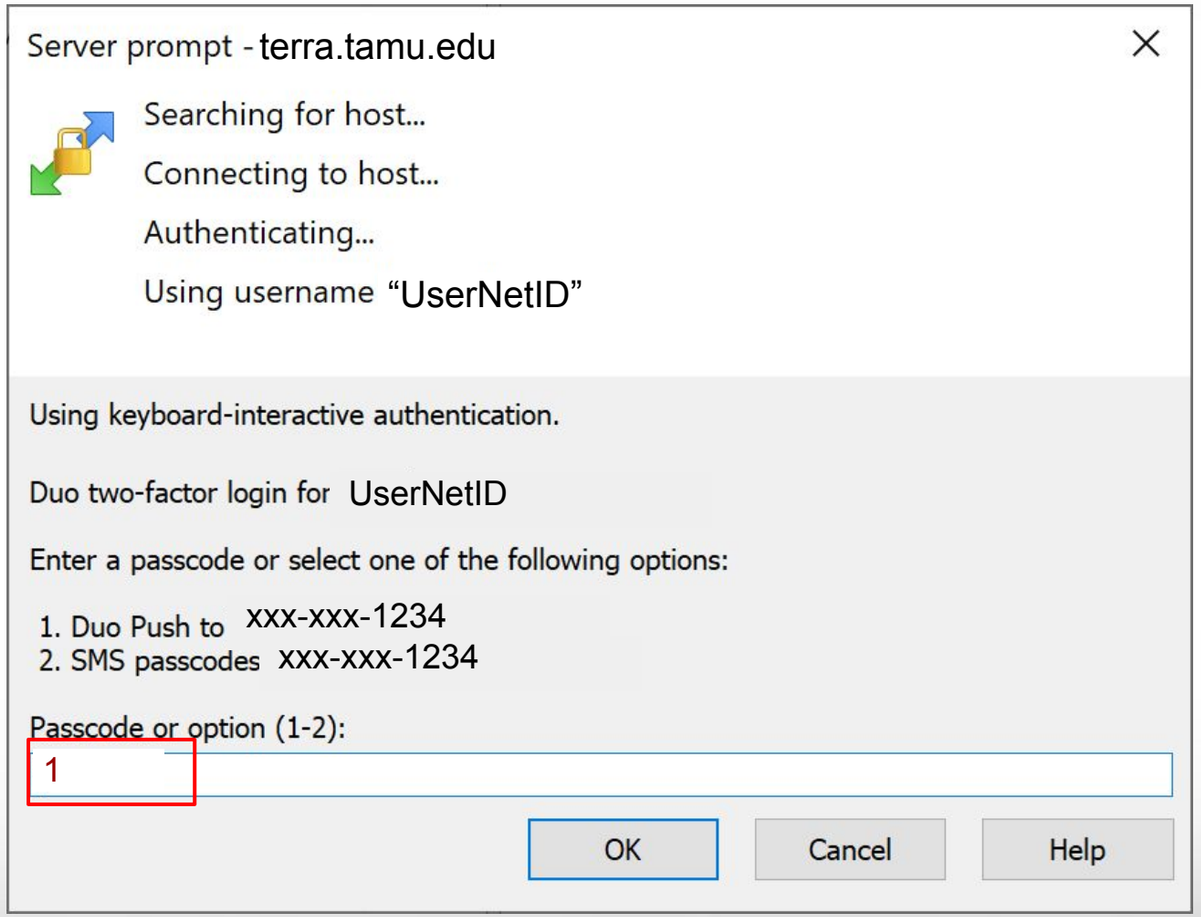
```
$ ssh UserNetID@terra.tamu.edu
************************************************************
…. warning message (snipped) …….
************************************************************

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





hprc.tamu.edu/wiki/Two_Factor#WinSCP_.28Windows_only.29

The Case (in)sensitive spider command

The following commands will give you different results on **Ada** because the `module spider` command is not case sensitive unless it finds an exact match for the search term

```
module spider python
```

```
module spider Python
```

use **Python** for Terra

```
module spider python

----------------------------------------------------------------------------------------------------------------------------
  python:
----------------------------------------------------------------------------------------------------------------------------
     Description:
       Python is a programming language that lets you work more quickly and integrate your systems more effectively. - Homepage:
       http://python.org/

     Versions:
       python/2.7.6-generic
       python/2.7.6-ictce-7.1.2
       python/2.7.10-intel-2015B.badSSL
       python/2.7.10-intel-2015B
       python/2.7.13-generic
     Other possible modules matches:
       Biopython  IPython  MySQL-python  NGS-Python  Python  ScientificPython  bx-python  findpython  myPython  netcdf4-python  ...
```

look for other possible modules

hprc.tamu.edu/wiki/Ada:Computing_Environment#Modules
hprc.tamu.edu/wiki/Terra:Computing_Environment#Modules

Modules and Toolchains

- Load modules with the same toolchains in your job scripts
- The **`2018b`** and **`GCCcore-7.3.0`** toolchain versions are recommended
    - `intel/2018b`
    - `iomkl/2018b`
    - `foss/2018b`
    - `GCCcore/7.3.0`
- Avoid loading modules in your .bashrc and .bash_profile files
- Avoid mixing toolchains if loading multiple modules in the same job script

```
module load HISAT2/2.0.4-foss-2016b
module load TopHat/2.1.1-intel-2017A-Python-2.7.12
module load Cufflinks/2.2.1-intel-2015B
```

- Same rule applies to compilers and libraries.

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

Ada: Examples of SUs charged based on Job Cores, Time and Memory Requested

A Service Unit (SU) on **Ada** is equivalent to one core or **2500** MB memory usage for one hour.

	Number of Cores	MB of memory per core	Total Memory (GB)	Hours	SUs charged
1.	1	2500	2.5	1	1
2.	1	2600	2.6	1	2
3.	1	50000	50	1	20
4.	20	2500	50	1	20

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

Historical HPRC Cluster Usage

hprc.tamu.edu

Cluster Status

Terra

Nodes	228/315 (72%)
Cores	5889/9408 (63%)
Jobs	311R-1Q

Ada

Nodes	692/823 (84%)
Cores	12070/16740 (72%)
Jobs	318R-77Q

Curie

Nodes	41/57 (72%)
Cores	656/911 (72%)
Jobs	41R-5Q

Historical Status

Queue Limits on Terra

Queue	Job Max Cores / Nodes	Job Max Walltime	Compute Node Types	Per-User Limits Across Queues	Notes
short	448 cores / 16 nodes	2 hrs	64 GB nodes (256) 128 GB nodes with GPUs (36)	1800 cores per user	
medium	1792 cores / 64 nodes	1 day			
long	896 cores / 32 nodes	7 days			
xlong	448 cores / 16 nodes	21 days	64 GB nodes (256)	448 cores per user	--partition xlong
gpu	1344 cores / 48 nodes	2 days	128 GB nodes with GPUs (48)		For jobs requiring GPUs.
vnc	28 cores / 1 node	12 hours	128 GB nodes with GPUs (48)		For remote visualization jobs
knl	68 cores / 8 nodes 72 cores / 8 nodes	7 days	96 GB nodes with KNL processors (16)		For jobs requiring a KNL processor

bqueues : Current Queues on **Ada**

```
[ user_netid@ada1 ~]$ bqueues
QUEUE_NAME      PRIO STATUS          MAX JL/U JL/P JL/H NJOBS  PEND   RUN  SUSP
staff            450  Open:Active       -    -    -    -     0     0     0     0
special          400  Open:Active       -    -    -    -  2080     0  2080     0
xlarge           100  Open:Active       -    -    -    -    80     0    80     0
vnc               90  Open:Active       -    -    -    -     2     0     2     0
sn_short          80  Open:Active       -    -    -    -     0     0     0     0
mn_short          80  Open:Active    2000    -    -    -     0     0     0     0
mn_large          80  Open:Active    8000    -    -    -  2500   720  1780     0
v100              80  Open:Active       -    -    -    -     0     0     0     0
general           50  Closed:Inact      0    -    -    -     0     0     0     0
sn_regular        50  Open:Active       -    -    -    -   846    31   815     0
sn_long           50  Open:Active       -    -    -    -   897     0   897     0
sn_xlong          50  Open:Active       -    -    -    -  2466    60  2406     0
mn_small          50  Open:Active    7000    -    -    -  3550    30  3520     0
mn_medium         50  Open:Active    6000    -    -    -  3784     0  3784     0
curie_devel       40  Open:Active       -    -    -    -     0     0     0     0
curie_medium      35  Open:Active       -    -    -    -     0     0     0     0
curie_long        30  Open:Active       -    -    -    -   765   301   464     0
curie_general     25  Closed:Inact      0    -    -    -     0     0     0     0
low_priority       1  Open:Active    2500  500    -    -     0     0     0     0
```

hprc.tamu.edu/wiki/Ada:Batch_Queues

Queue Limits on **Ada**

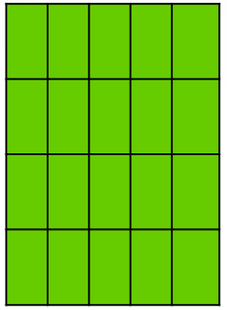
Queue	Min/Default/ Max Cores	Default/Max Walltime	Compute Node Types	Pre-Queue Limits	Aggregate Limits Across Queues	Per-User Limits Across Queues	Notes
sn_short	1 / 1 / 20	10 min / **1 hr**	64 GB nodes (811) 256 GB nodes (26)		Maximum of **7000** cores for all running jobs in the single-node (sn_*) queues.	Maximum of **1000 cores and 100 jobs per user** for all running jobs in the single node (sn_*) queues.	For jobs needing **only one compute node**.
sn_regular		1 hr / **1 day**					
sn_long		24 hr / **4 days**					
sn_xlong		4 days / **30 days**					
mn_short	2 / 2 / 200	10 min / **1 hr**		Maximum of **2000** cores for all running jobs in this queue.	Maximum of **12000** cores for all running jobs in the multi-node (mn_*) queues.	Maximum of **3000 cores and 150 jobs per user** for all running jobs in the multi-node (mn_*) queues.	For jobs needing **more than one compute node**.
mn_small	2 / 2 / 120	1 hr / **10 days**		Maximum of **7000** cores for all running jobs in this queue.			
mn_medium	121 / 121 / 600	1 hr / **7 days**		Maximum of **6000** cores for all running jobs in this queue.			
mn_large	600 / 601 / 2000	1 hr / **5 days**		Maximum of **8000** cores for all running jobs in this queue.			
xlarge	1 / 1 / 280	1 hr / **10 days**	1 TB nodes (11) 2 TB nodes (4)				For jobs needing **more than 256GB of memory per compute node**.
vnc	1 / 1 / 20	1 hr / **6 hr**	GPU nodes (30)				For remote visualization jobs.
special	None	1 hr / **7 days**	64 GB nodes (811) 256 GB nodes (26)				Requires permission to access this queue.
v100	1 / 1 / 96	1 hr / **2 days**	192 GB nodes dual 32 GB v100 (4)				

Run ***"blimits -w"*** to show how policies are applied to users and queues.

hprc.tamu.edu/wiki/Ada:Batch_Queues

Mapping Jobs to Cores per Node on **Ada**

A.

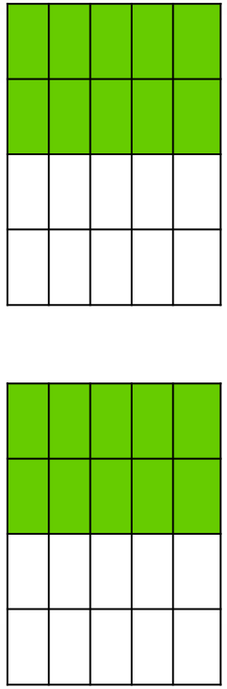


20 cores on
1 compute node

```
#BSUB -n 20
#BSUB -R “span[ptile=20]
```

Preferred Mapping
(if applicable)

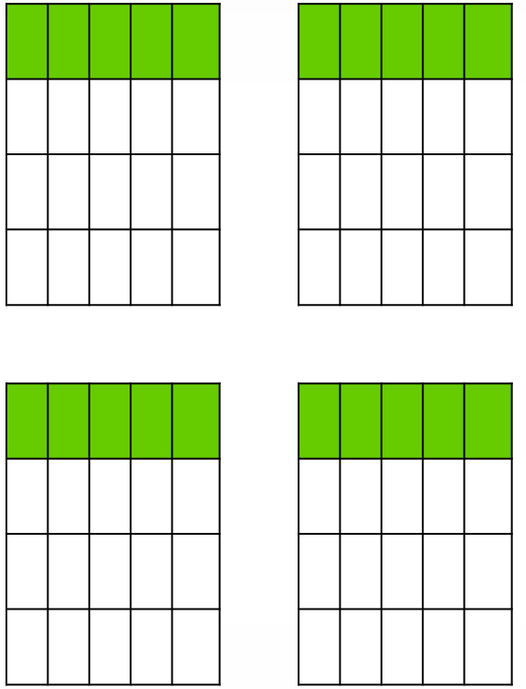
B.



20 cores on
2 compute nodes

```
#BSUB -n 20
#BSUB -R “span[ptile=10]
```

C.



20 cores on
4 compute nodes

```
#BSUB -n 20
#BSUB -R “span[ptile=5]
```

Job Memory Requests on **Ada**

- Must specify both parameters for requesting memory:

  #BSUB -R "rusage[mem=process_alloc_size]"

  #BSUB -M process_size_limit
- Default value of 2500 MB (2.5 GB) per job slot if -R/-M not specified, but it might cause memory contention when sharing a node with other jobs.
- On 64GB nodes, usable memory is at most **54 GB** (where 10 GB is used by the system).
  - The per-process memory limit should not exceed **2700 MB** for a 20-core job.
- If more memory is needed, request the large memory nodes:
  - If under 256 GB and up to 20 cores per node use:
    - **#BSUB -R "select[mem256gb]"**
  - If you need up to 1 or 2 TB of memory and up to 40 cores:
    - use the **-q xlarge** option with either **-R "select[mem1tb]"** or **-R "select[mem2tb]"**
    - The mem1tb and mem2tb nodes are accessible only via the *xlarge* queue.

Education Resources

For further learning on other topics, attend one of our upcoming short courses: https://hprc.tamu.edu/training/

SLURM Job Scheduling	Friday, February 12
Introduction to Next Generation Sequencing Analysis	Friday, February 19
Introduction to Perl	Friday, February 19
Introduction to Python	Friday, February 26
Introduction to Scientific Python	Friday, February 26
Introduction to Quantum Chemistry Simulations with ORCA	Friday, March 12
Drug Docking with Schrodinger	Friday, March 26
Scientific Machine Learning	Friday, March 26

Submitting Your Job and Check Job Status

Submit job

```
cd $SCRATCH/batch_examples
```

Terra:

```
sbatch example01.job
```

```
Submitted batch job 161997
(from job_submit) your job is charged as below
               Project Account: 122792016265
               Account Balance: 1687.066160
               Requested SUs:   3
```

Ada:

```
bsub < example01.job
```

```
Verifying job submission parameters...
Verifying project account...
     Account to charge:   082792010838
          Balance (SUs):      4871.5983
          SUs to charge:         0.0333
Job <2470599> is submitted to default queue <sn_short>.
```

Check status

Terra:

```
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]
```

Ada:

```
bjobs
```

Ada (more detailed):

```
bjobs -l 2470599
```

(dash lower case `l` as in *list*)

```
JOBID    STAT USER        QUEUE     JOB_NAME  NEXEC_HOST SLOTS RUN_TIME     TIME_LEFT
2470599  RUN  tmarkhuang sn_short sample01  1            1     0 second(s)  0:5 L
```

Ada: Software for Large Memory Nodes

- The large memory nodes (1TB and 2TB) on Ada have their own separate modules
- To see the list of available large memory node modules use the following commands:

```
module load Westmere
module avail
```

- Look in the Westmere section:

```
--------------------------- /software/easybuild/Westmere/modules/all ------------------------------------
   ABINIT/8.0.8-intel-2016a                                      Tcl/8.6.3-foss-2015a
   ABySS/1.9.0-foss-2016a                                        Tcl/8.6.3-intel-2015B
   ABySS/1.9.0-intel-2015B-Python-2.7.10                  (D)    Tcl/8.6.4-foss-2016a
```

- Or type **module spider tool_name/version** to see if Westmere needs to be loaded

```
module spider Canu/1.7-intel-2017A-Perl-5.24.0
```

```
    You will need to load all module(s) on any one of the lines below before the
    "Canu/1.7-intel-2017A-Perl-5.24.0" module is available to load.

    Westmere/0.devel
    Westmere/1
```

- You can just load Westmere along with the other module(s)

```
module load Westmere
module load Canu/1.7-intel-2017A-Perl-5.24.0
```

Sample Job Script for 1TB Node on Ada

```
##NECESSARY JOB SPECIFICATIONS
#BSUB -L /bin/bash                      # Uses bash to initialize the job's execution environment.
#BSUB -J my_job_script                  # Set the job name to "my_job_script"
#BSUB -W 24:00                          # Set the wall clock limit to 24hr
#BSUB -q xlarge                         # Request xlarge queue
#BSUB -R "select[mem1tb]"               # Request 1TB memory node
#BSUB -n 40                             # Request 40 core
#BSUB -R "span[ptile=40]"               # Request 40 core per node.
#BSUB -R "rusage[mem=24500]"            # Request 24500MB (24.5GB) per process (CPU) for the job
#BSUB -M 24500                          # Set the per process enforceable memory limit to 24500MB.
#BSUB -o stdout.%J                      # Send stdout to "stdout.[jobID]"
#BSUB -e stderr.%J                      # Send stderr to "stderr.[jobID]"
#BSUB -u email_address                  # (optional) Send all emails to email_address
#BSUB -B -N                             # (optional) Send email on job begin (-B) and end (-N)

# load required module(s) for use on the large memory nodes
module load Westmere
module load Canu/1.7-intel-2017A-Perl-5.24.0

# run your commands
canu -assemble *fastq
```

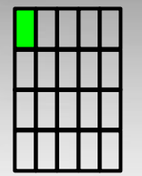
Ada Pop Quiz

```
#BSUB -L /bin/bash
#BSUB -J stacks_S2
#BSUB -n 10
#BSUB -R "span[ptile=10]"
#BSUB -R "rusage[mem=2500]"
#BSUB -M 2500
#BSUB -W 36:00
#BSUB -o stdout.%J
#BSUB -e stderr.%J
```

How much total memory is requested for this job?

A. 2.5 GB
B. 2500 MB
C. 25 GB
D. 250 GB

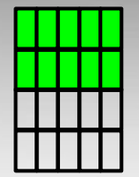
Ada Job File (Serial Example)



```
##NECESSARY JOB SPECIFICATIONS
#BSUB -L /bin/bash                    # Uses bash to initialize the job's execution environment.
#BSUB -J ExampleJob1                  # Set the job name to "ExampleJob1"
#BSUB -W 2:00                         # Set the wall clock limit to 2hr
#BSUB -n 1                            # Request 1 core
#BSUB -R "span[ptile=1]"              # Request 1 core per node.
#BSUB -R "rusage[mem=2500]"           # Request 2500MB per process (CPU) for the job
#BSUB -M 2500                         # Set the per process enforceable memory limit to 2500MB.
#BSUB -o stdout.%J                    # Send stdout to "stdout.[jobID]"
#BSUB -e stderr.%J                    # Send stderr to "stderr.[jobID]"

##OPTIONAL JOB SPECIFICATIONS
#BSUB -P 123456                       # Set billing account to 123456
#BSUB -u email_address                # Send all emails to email_address
#BSUB -B -N                           # Send email on job begin (-B) and end (-N)

# load required module(s)
module load Python/3.5.2-intel-2017A

# run your program
./my_program.py
```

SUs = 2

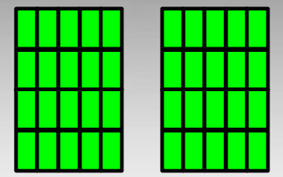
Ada Job File (multi core, single node)



```
##NECESSARY JOB SPECIFICATIONS
#BSUB -L /bin/bash                  # Use bashto initialize the job's execution environment.
#BSUB -J ExampleJob2                # Set the job name to "ExampleJob2"
#BSUB -W 6:30                       # Set the wall clock limit to 6hr and 30min
#BSUB -n 10                         # Request 10 cores total for the job
#BSUB -R "span[ptile=10]"           # Request 10 cores per node.
#BSUB -R "rusage[mem=2500]"         # Request 2500MB per process (CPU) for the job
#BSUB -M 2500                       # Set the per process enforceable memory limit to 2500MB.
#BSUB -o stdout.%J                  # Send stdout to "stdout.[jobID]"
#BSUB -e stderr.%J                  # Send stderr to "stderr.[jobID]"

##OPTIONAL JOB SPECIFICATIONS
#BSUB -P 123456           # Set billing account to 123456  #find your account with "myproject"
#BSUB -u email_address              # Send all emails to email_address
#BSUB -B -N                         # Send email on job begin (-B) and end (-N)

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

# run your program
./my_multicore_program
```

SUs = 65

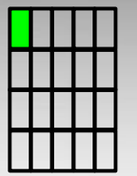
Ada Job File (multi core, multi node)



```
##NECESSARY JOB SPECIFICATIONS
#BSUB -J ExampleJob3                  # Set the job name to "ExampleJob3"
#BSUB -L /bin/bash                    # Use bash to initialize the job's execution environment.
#BSUB -W 24:00                        # Set the wall clock limit to 24hr
#BSUB -n 40                           # Request 40 cores total for the job
#BSUB -R "span[ptile=20]"             # Request 20 cores per node.
#BSUB -R "rusage[mem=2500]"           # Request 2500MB per process (CPU) for the job
#BSUB -M 2500                         # Set the per process enforceable memory limit to 2500MB.
#BSUB -o stdout.%J                    # Send stdout to "stdout.[jobID]"
#BSUB -e stderr.%J                    # Send stderr to "stderr.[jobID]"

##OPTIONAL JOB SPECIFICATIONS
#BSUB -P 123456                   # Set billing account to 123456 #find your account with " myproject"
#BSUB -u email_address            # Send all emails to email_address
#BSUB -B -N                       # Send email on job begin (-B) and end (-N)

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

# run your program
./my_multicore_multinode_program
```

SUs = 960

Ada Job File (serial GPU)



```
##NECESSARY JOB SPECIFICATIONS
#BSUB -J ExampleJob4           # Set the job name to "ExampleJob4"
#BSUB -L /bin/bash             # Use bash login shell to initialize the job's execution environment.
#BSUB -W 2:00                  # Set the wall clock limit to 2hr
#BSUB -n 1                     # Request 1 core total for the job
#BSUB -R "span[ptile=1]"       # Request 1 core per node.
#BSUB -R "rusage[mem=2500]"    # Request 2500MB per process (CPU) for the job
#BSUB -M 2500                  # Set the per process enforceable memory limit to 2500MB.
#BSUB -o stdout.%J             # Send stdout to "stdout.[jobID]"
#BSUB -e stderr.%J             # Send stderr to "stderr.[jobID]"

#BSUB -R "select[gpu]"         # Request a node with a GPU
##OPTIONAL JOB SPECIFICATIONS
#BSUB -P 123456                # Set billing account to 123456 #find your account with "myproject"
#BSUB -u email_address         # Send all emails to email_address
#BSUB -B -N                    # Send email on job begin (-B) and end (-N)

# load required module(s)
module load CUDA/9.2.148.1

# run your program
./my_gpu_program
```

SUs = 1

List Node Utilization on **Ada**: *lnu*

```
lnu [-h] [-l] -j jobid
```

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

Example with a multi-node job (4 nodes):

```
lnu -l -j 795375
```

```
Job           User                    Queue         Status Node  Cpus
795375        jomber23                medium                R    4    80
   HOST_NAME    status  r15s   r1m  r15m   ut    pg  ls    it   tmp   swp   mem  Assigned Cores
   nxt1417          ok  20.0  21.0  21.0  97%   0.0   0 94976  366M  3.7G 41.6G     20
   nxt1764 (L)      ok  19.7  20.0  20.0  95%   0.0   0 95040  366M  3.7G 41.5G     20
   nxt2111          ok  20.0  20.0  20.0  98%   0.0   0 91712  370M  4.2G 41.5G     20
   nxt2112          ok  20.0  21.1  21.0  97%   0.0   0 91712  370M  4.2G 41.6G     20
================================================================================
```

The % utilization (**ut**) in conjunction with Assigned Cores is the most useful.
Note that the **tmp**, **swp**, and **mem** refer to available amounts respectively and not usage.
See "*man lsload*" for explanations on labels.

hprc.tamu.edu/wiki/Ada:Batch_Processing_LSF#Job_Tracking

Common Job Problems

- Control characters (`^M`) in job files or data files edited with Windows editor
  - remove the `^M` characters with: `dos2unix my_job_file`
- Did not load the required module(s)
- Insufficient walltime specified in `#SBATCH --time` or `#BSUB -W` parameter
- Insufficient memory specified in , `#SBATCH --mem` or `--mem-per-cpu` or `#BSUB -M` and `-R "rusage[mem=xxx]"` parameters
- No matching resource (`--mem` or `-R rusage[mem]` too large)
- Running OpenMP jobs across nodes
- Insufficient SU: See your SU balance: `myproject`
- Insufficient disk or file quotas: check quota with `showquota`
- Using GUI-based software without setting up X11 forwarding
  - Enable X11 forwarding at login `ssh -X user@ada.tamu.edu`
  - Or use Portal (portal.hprc.tamu.edu)
- Software license availability

`license_status -a`

FAQ: hprc.tamu.edu/wiki/HPRC:CommonProblems

CRLF Line Terminators

Windows editors such as Notepad will add hidden Carriage Return Line Feed (CRLF) characters that will cause problems with many applications

```
cd $SCRATCH/batch_examples
file dos_text.txt
```

use `file` command to check

```
dos_text.txt: ASCII English text, with CRLF line terminators
```

```
cat -v dos_text.txt
```

use `cat` command to see CRLF characters

```
dos2unix dos_text.txt
file dos_text.txt
```

use `dos2unix` command to correct

```
dos_text.txt: ASCII English text
```