HIGH PERFORMANCE RESEARCH COMPUTING

HPRC Training

Introduction to HPRC Computing Resources

February 3, 2023

High Performance
Research Computing
DIVISION OF RESEARCH

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

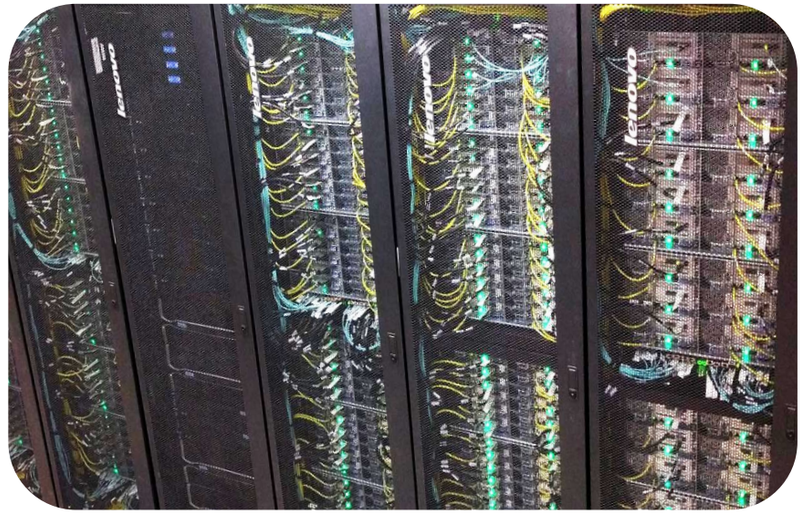
- Knowledge Foundation:
  - Basic knowledge of LINUX commands
  - Slides from our LINUX short course are at:

    hprc.tamu.edu/training/intro_linux.html
  - Watch the relevant Introduction and Primer videos on our YouTube Channel

    Texas A&M HPRC YouTube channel
  - Answers to frequently asked questions can be found on our Wiki

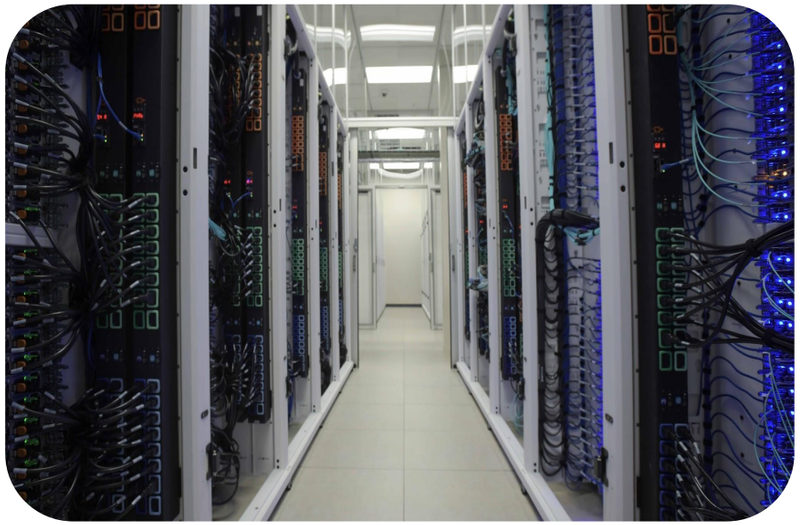
    hprc.tamu.edu/wiki/

HPRC Clusters



Terra

Deployed in 2017



Grace

Deployed in 2021
Flagship cluster

FASTER

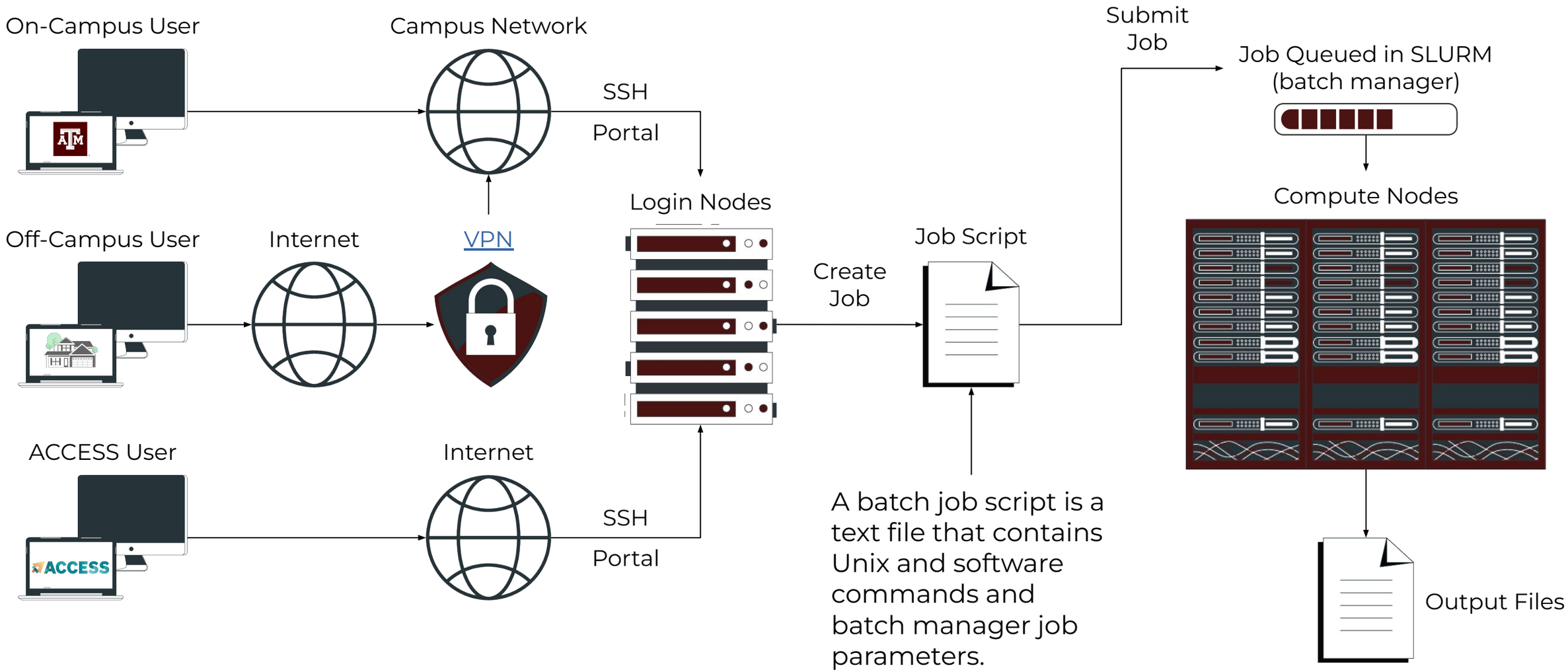
Deployed in 2022

Clusters Are For You!

What kinds of problems are solved by cluster computing?

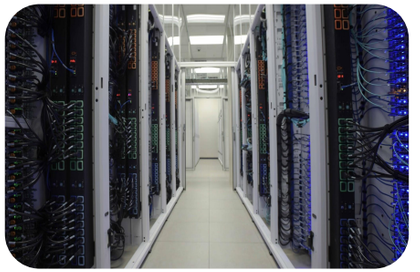
- Problems that are too big to fit in a laptop or workstation, due to limitation on memory, core count, or node count
- Problems that scale well with more CPU cores or memory
- Single-threaded problems with many permutations
- Problems that require large high speed storage and/or interconnect

Batch Computing on HPRC Clusters



HPRC Clusters

Terra



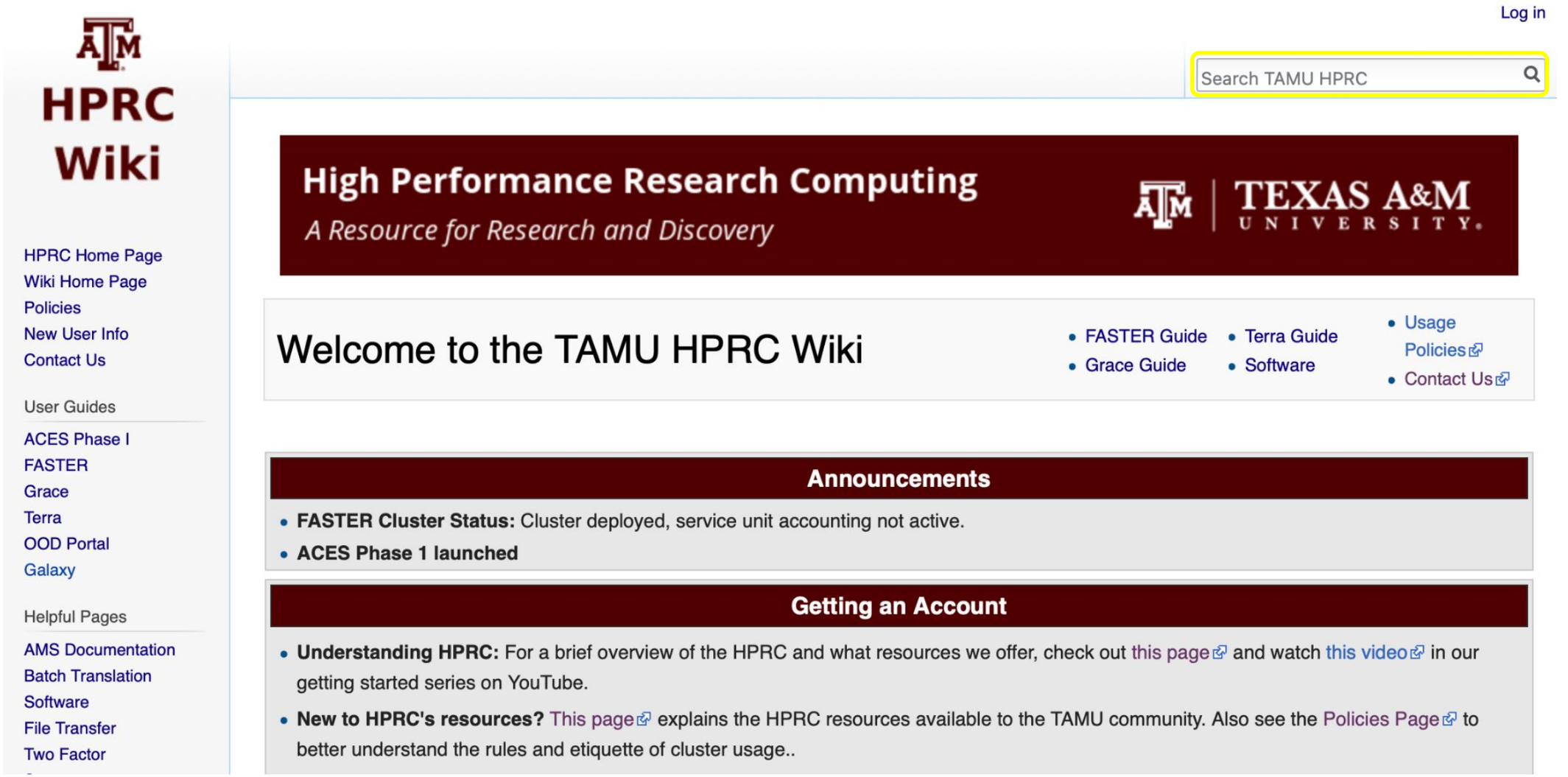
Grace

FASTER

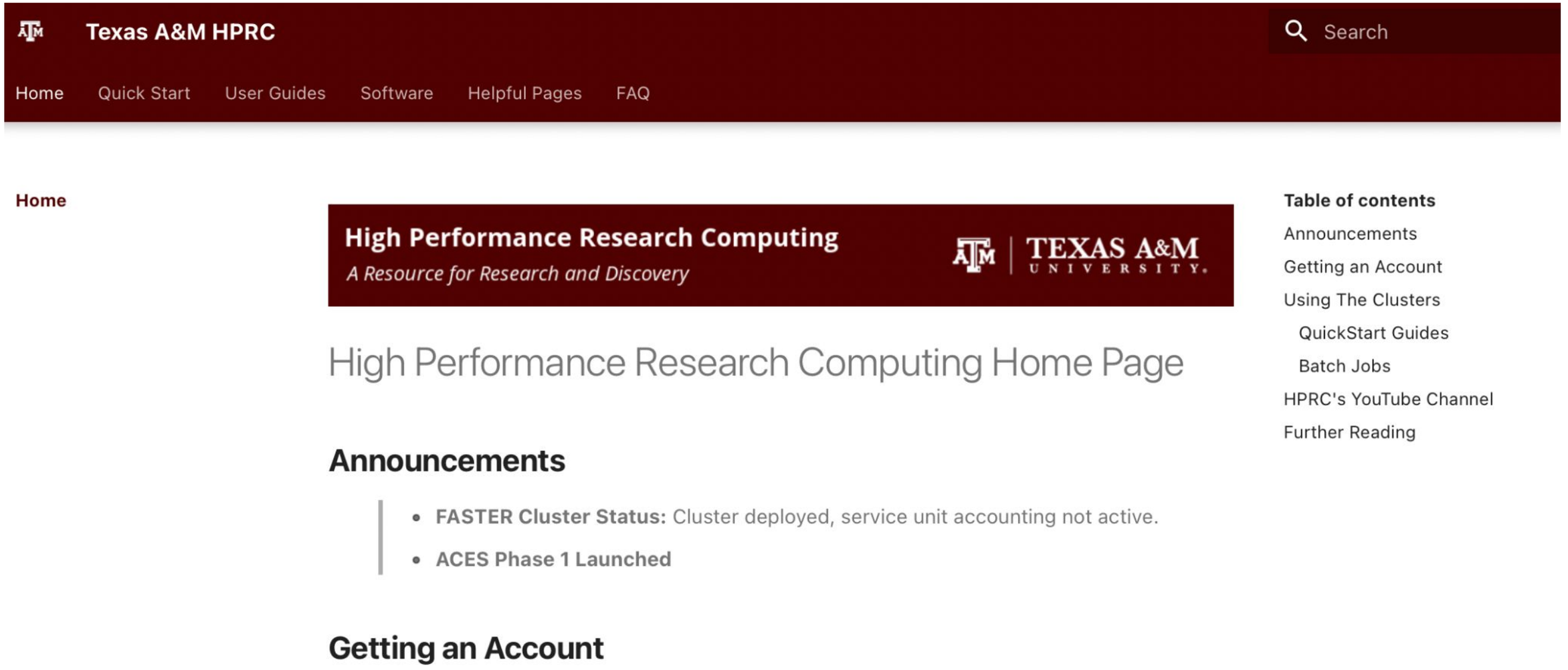
Total Nodes (Cores)	320 (9,632)	925 (44,656)	180 (11,520)
General Nodes	28 cores 64GB	48 cores 384GB	64 cores 256GB
Features	GPUs	GPUs - multiprecision Big Memory Nodes	Composable GPUs Big Memory Nodes
Job Scheduler	Slurm	Slurm	Slurm
Online Since	2017	2021	2022

hprc.tamu.edu/resources

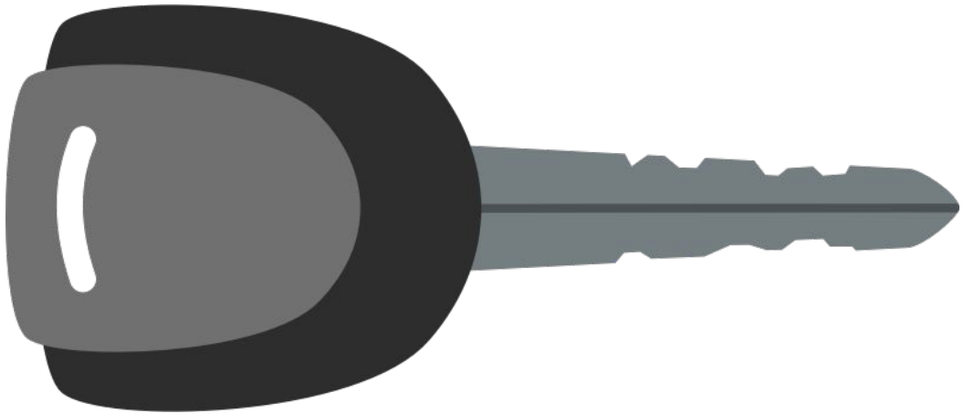
HPRC Wiki - Hardware



HPRC Documentation - Sneak Peak

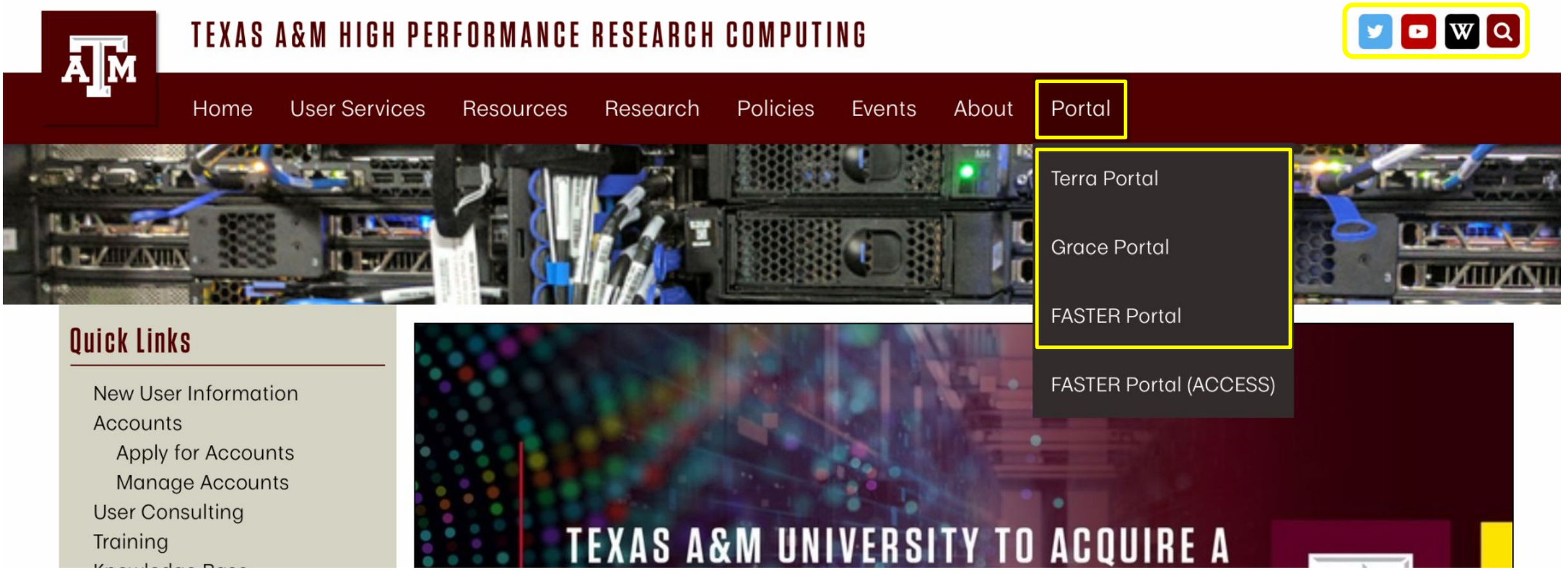


Getting Started



Accessing the HPRC Portal

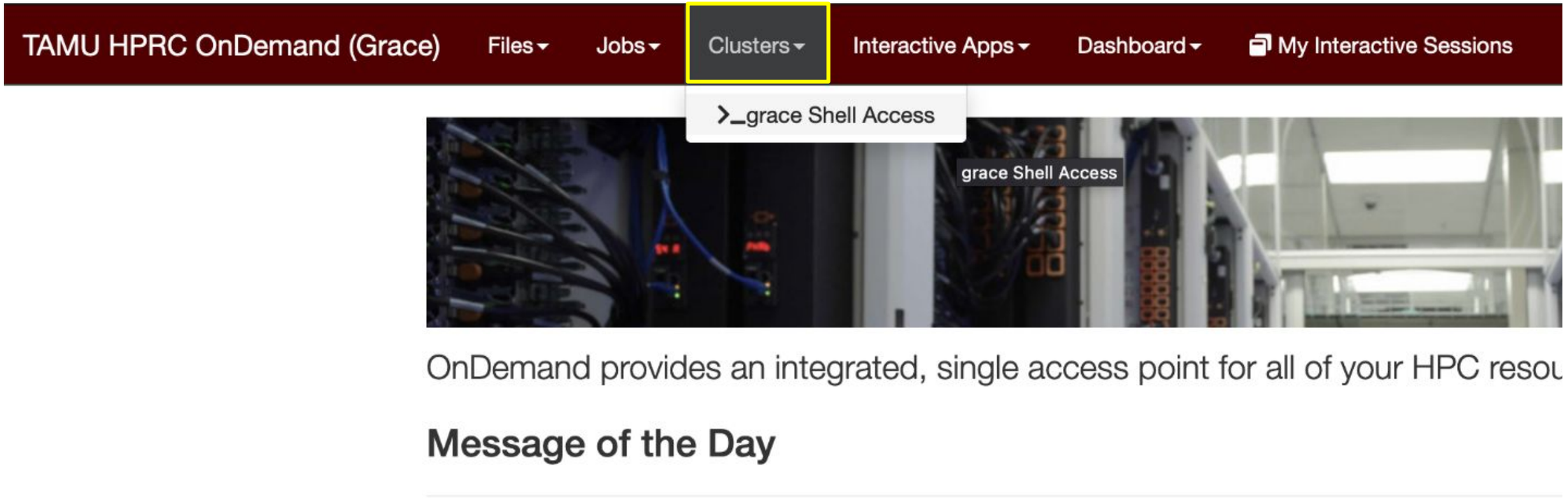
- HPRC webpage: hprc.tamu.edu, Portal dropdown menu



Shell access via the HPRC Portal

Access through (most) web browsers

–Top Banner Menu “Clusters” -> “Shell Access”



Accessing the HPRC clusters via SSH

- Accessing faster, grace, or terra via ssh:
  - On campus:
  - Off campus:
    - You will need to run VPN before you can ssh.
  - Login requires Two-Factor Authentication

```
ssh [NetID]@[cluster].hprc.tamu.edu
```

- SSH programs for Windows:
  - MobaXTerm (preferred, includes SSH and X11), PuTTY SSH, Windows Subsystem for Linux
- Login nodes: Grace (5), Terra (3), Faster (2)

  Check the bash prompt to determine which login node you are logged into.

  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.

hprc.tamu.edu/wiki/HPRC:Access

Accessing the HPRC clusters via SSH

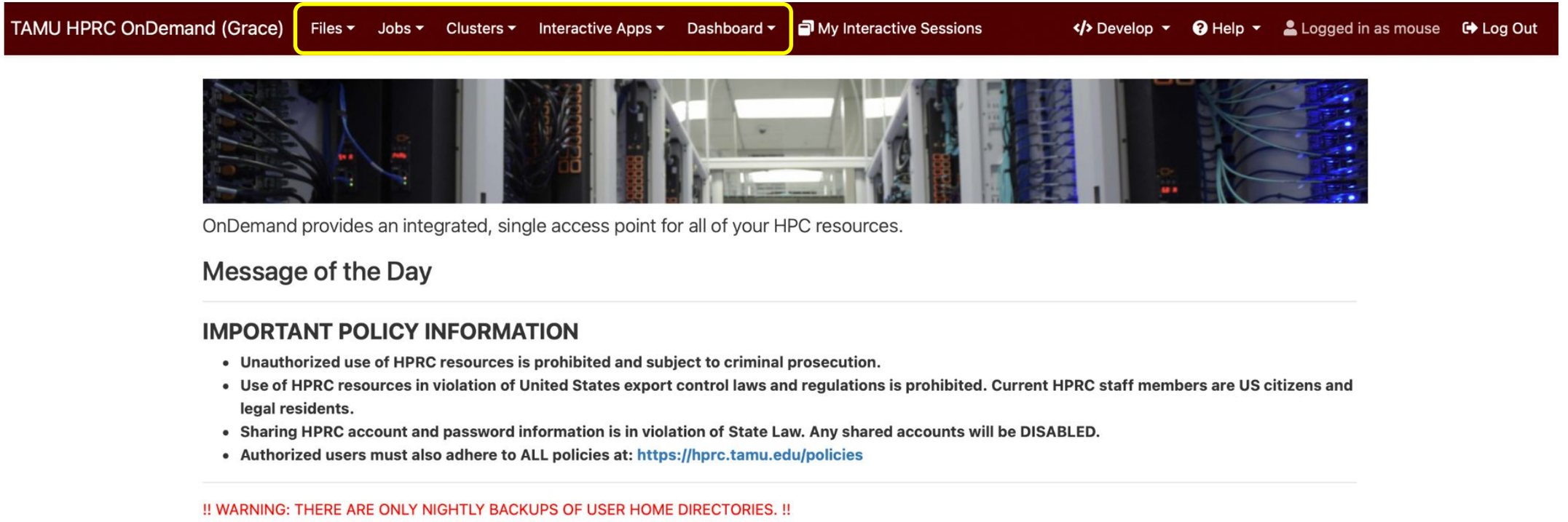
```
$ ssh grace.hprc.tamu.edu
**************************************************************************
This computer system and the data herein are available only for authorized
purposes by authorized users. Use for any other purpose is prohibited and may
result in disciplinary actions or criminal prosecution against the user. Usage
may be subject to security testing and monitoring. There is no expectation of
privacy on this system except as otherwise provided by applicable privacy laws.
Refer to University SAP 29.01.03.M0.02 Acceptable Use for more information.
**********************************************************

Password: (Type password, not no characters will show up)
Duo two-factor login for [NetID]

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…

[NetID@grace5 ~]$
```

Hands-on exercise:

Login to the Grace Portal

hprc.tamu.edu

HPRC Portal



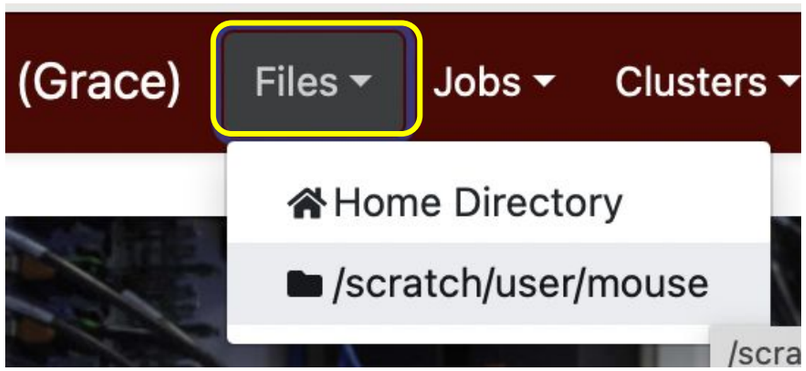
Files: copy and edit files on the cluster's filesystems
Jobs: submit and monitor cluster jobs
Clusters: open a shell terminal (command line) on a login node
Interactive Apps: start graphical software on a compute node and connect to it
Dashboard: view file quotas and computing account allocations

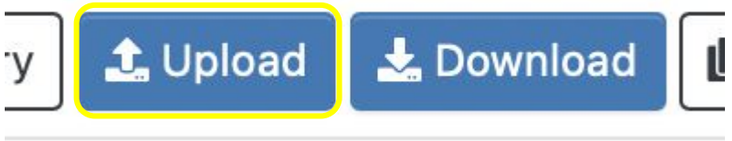
Hands-on exercise:

Upload a File using the Grace Portal

Files > /scratch/user/<NetID>

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





File Systems and User Directories

Directory	Environment Variable	Space Limit	File Limit	Intended Use
/home/$USER	$HOME	10 GB	10,000	Small to modest amounts of processing.
/scratch/user/$USER	$SCRATCH	1 TB	250,000	Temporary storage of large files for on-going computations. Not intended to be a long-term storage area.

`$SCRATCH` is shared between FASTER (TAMU/ACCESS) and Grace (TAMU) clusters.

View file usage and quota limits using the command: **`showquota`**

Do **NOT** share your home or scratch directories. Request a group directory for sharing files.

File System Wiki pages: FASTER Grace Terra

Data on Our Clusters: Grace and Terra

What's the difference between these filesystems?

`/home`

high performance
global storage
will not be expanded
backed up

`/scratch`

high performance
global storage
can be expanded
not backed up

`/tmp`

local storage

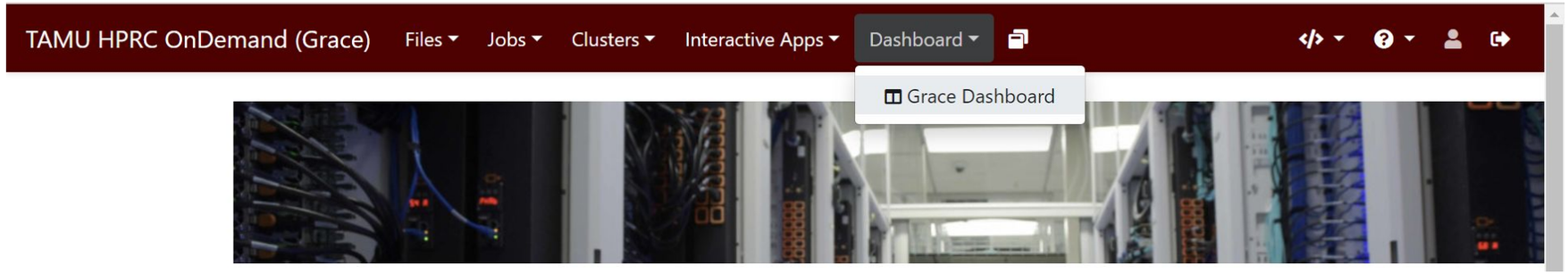
not backed up

Need more space?
Submit a ***Quota Increase Request***

Contact help@hprc.tamu.edu

HPRC Portal Quota Increase Request

Grace Portal Homepage → Grace Dashboard

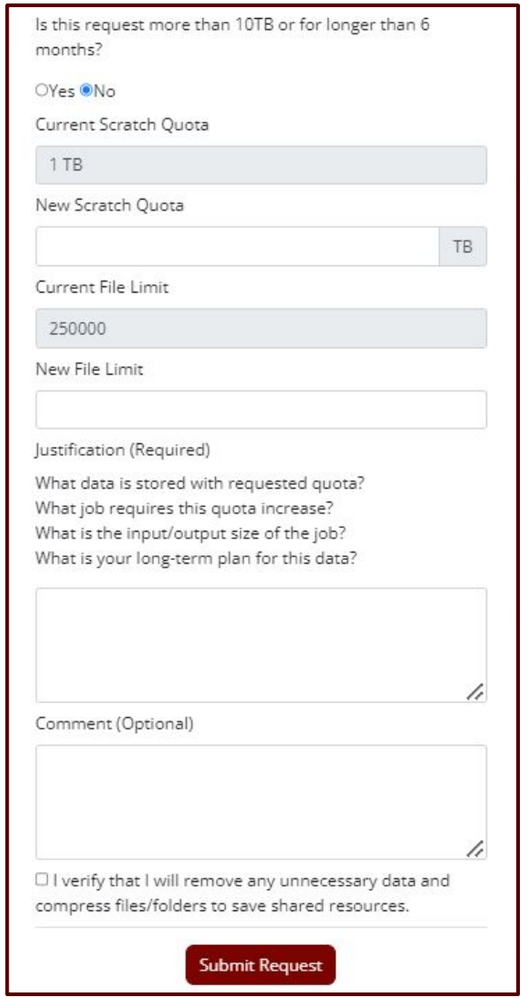


Request quota increases directly from the dashboard with a guided form

Disk Quotas

Disk	Disk Usage	Limit	File Usage	Limit
/home	160 KB (0.00 %)	10 GB	35 (0.35 %)	10000
/scratch	837.84 MB (0.08 %)	1 TB	11795 (4.72 %)	250000

Request Quota Increase

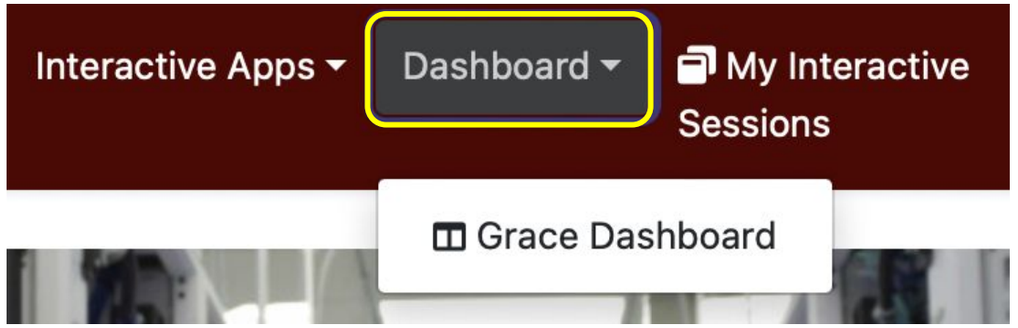


Hands-on exercise:

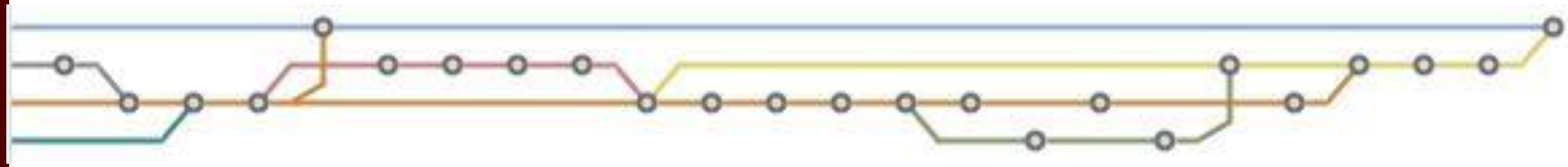
Check your file Quota in the Portal

Dashboard > Grace Dashboard

Locate your /scratch disk usage stats



Software Infrastructure



Software

- HPRC provides both pre-installed Software and installation assistance
- See the Software pages for instructions and examples
  - hprc.tamu.edu/wiki/SW
  - hprc.tamu.edu/software/
- License-restricted software
  - Contact help@hprc.tamu.edu
- Contact us for software installation help/request
  - User can install software in their home/scratch directory
  - 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)]
TensorFlow/2.5.0-Python-3.7.4
```

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

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)
```

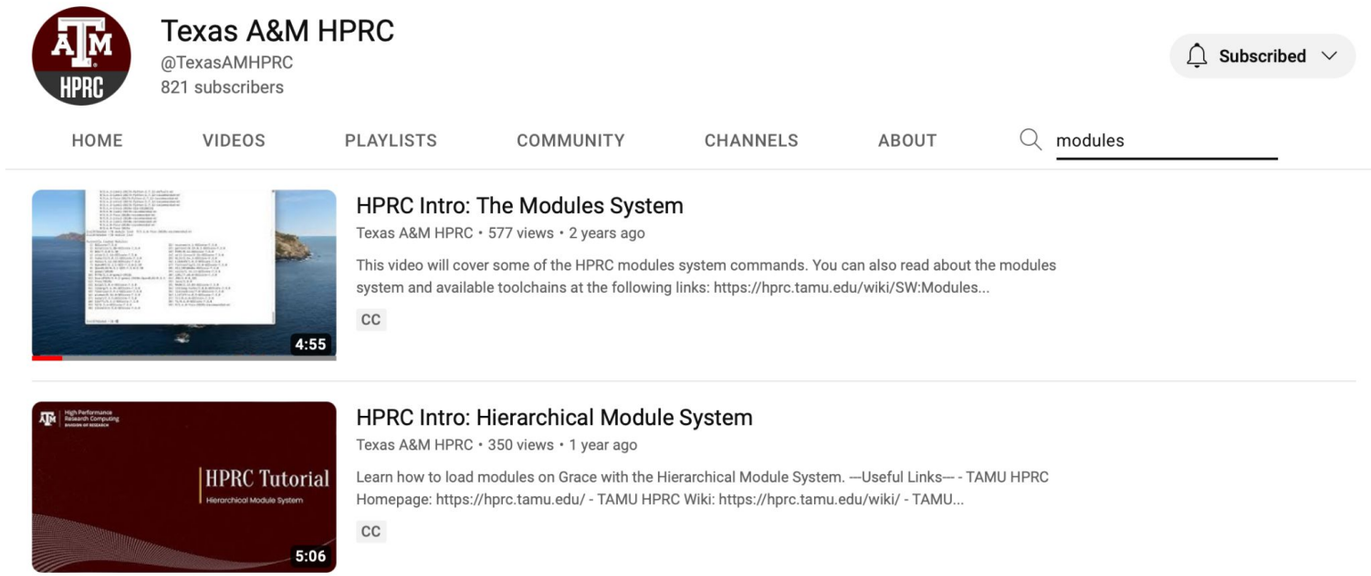
module spider <word>

```
# case insensitive search for modules with 'word' in name
```

module load <module>

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

module commands



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.

module commands

Module Usage Example

Grace and FASTER use a *software hierarchy* inside the module system

```
mla snakemake
```

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

```
snakemake/5.2.4-Python-3.6.6
snakemake/5.7.1-Python-3.7.2
snakemake/5.7.1-Python-3.7.4
snakemake/5.26.1-Python-3.8.2
snakemake/6.1.0
snakemake/6.10.0
```

```
module spider snakemake/6.10.0
```

```
# find how to load a particular module using the
# full module name based on the above results
```

```
----------------------------------------------------------------------------------------
  snakemake: snakemake/6.10.0
----------------------------------------------------------------------------------------
    Description:
      The Snakemake workflow management system is a tool to create reproducible and scalable  data analyses.

    You will need to load all module(s) on any one of the lines below before the "snakemake/6.10.0" module is available to load.

      GCC/11.2.0  OpenMPI/4.1.1
```

```
module load GCC/11.2.0  OpenMPI/4.1.1
```

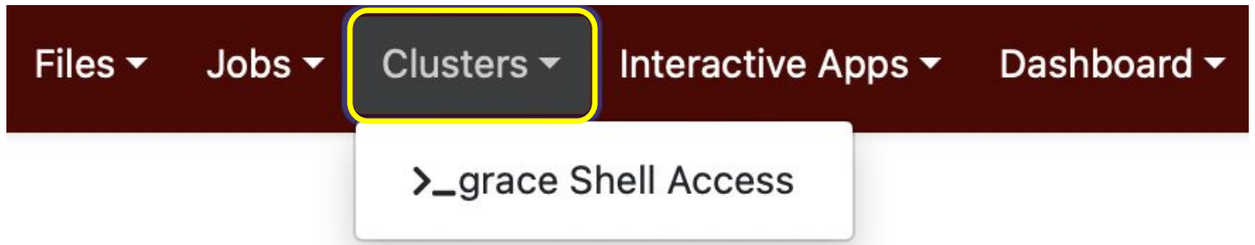
```
# Load the base dependency
# module(s) first then the full
# module name
```

Hands-on exercises:

Open a terminal on Grace in Portal

Clusters > _Grace Shell Access

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



Hands on Exercise: Module Loading

1. `mla blast+` # see which versions of BLAST+ are available
2. `ml BLAST+/2.13.0` # error; you can't do that yet
3. `module spider BLAST+/2.13.0` # learn how to load this module
4. `ml _____ BLAST+/2.13.0` # 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/2022a
```

hprc.tamu.edu/wiki/SW:Toolchains

Module Usage Practices

- Software 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. `module purge`
- It is recommended to load a specific software version instead of the defaults: `ml GCC/12.2.0` instead of `ml Gcc`
- Avoid loading modules in your $HOME/.bashrc
- Avoid mixing toolchains when loading multiple modules at the same time. This usually leads to one of them not working.

https://hprc.tamu.edu/wiki/SW: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

Python Software Install Exercise - Grace

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
Terra(ml Python/3.8.6-GCCcore-10.2.0)

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)

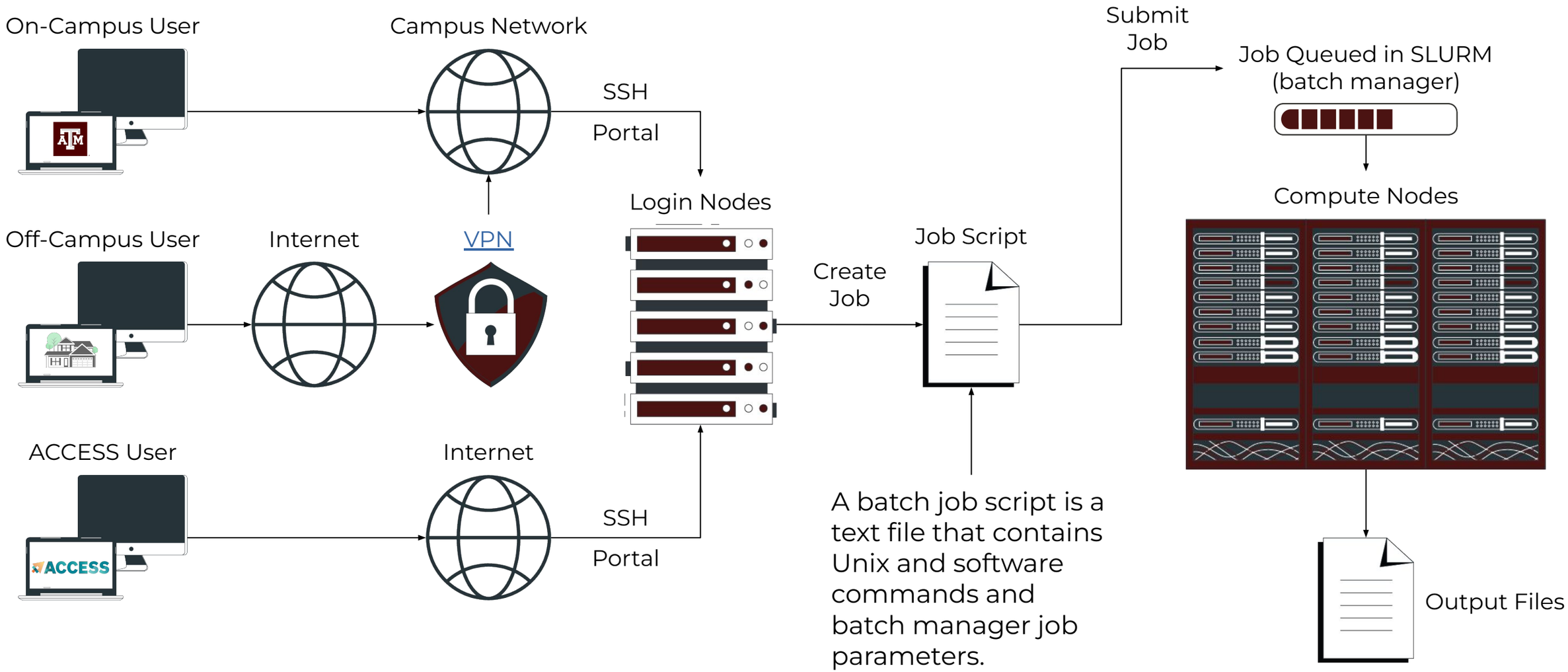
5.
```
pip install python-time
```
install python-time

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



Consumable Computing Resources

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

```
myproject
```

```
================================================================================================
              List of YourNetID's Project Accounts
------------------------------------------------------------------------------------------------
|   Account    |   FY   | Default | Allocation |Used & Pending SUs|    Balance   |     PI      |
------------------------------------------------------------------------------------------------
|1228000223136|  2023  |       N|    10000.00|              0.00|    10000.00|Doe, John   |
------------------------------------------------------------------------------------------------
|1428000243716|  2023  |       Y|     5000.00|            -71.06|     4928.94|Doe, Jane   |
------------------------------------------------------------------------------------------------
```

- Other resources:
  - Software license/token
    - Use "**license_status**" to query

```
license_status -a
```

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:

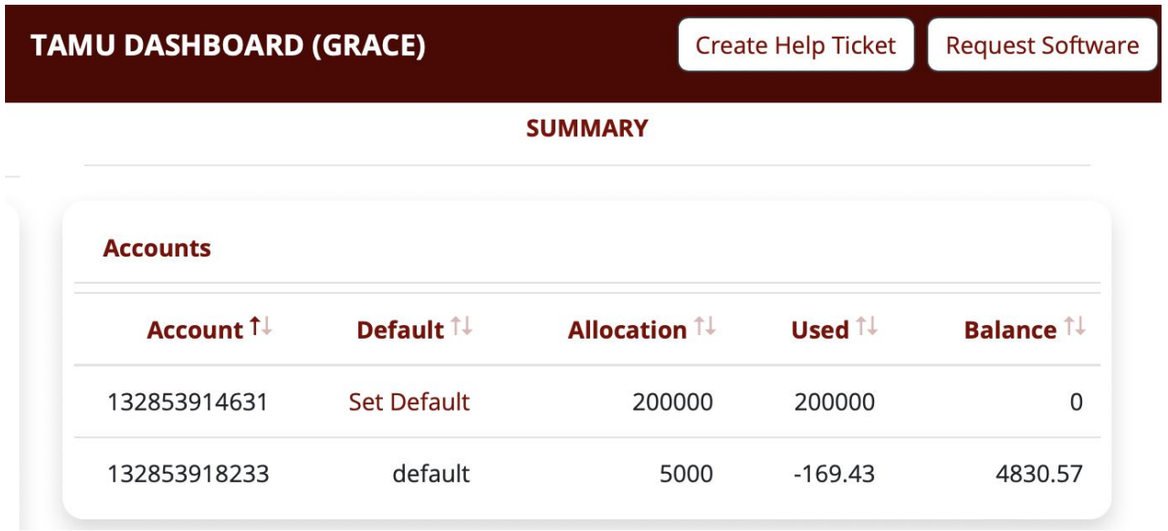
```
license_status -h
```

Hands-on exercise:

Check your SU Accounts on Grace in the Portal

Dashboard > Grace Dashboard

Locate your default account



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

A **Service Unit (SU)** is equivalent to one core hour *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

Slurm: SU charges for GPU jobs

Grace

GPU jobs on grace are charged additional SUs per effective GPU requested:

Effective GPU	SU charge per one hour (wall_time)
A100	72
RTX 6000	48
T4	24

Batch Queues

- Job submissions are auto-assigned to batch queues based on the resources requested (number of cores/nodes, gpus 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 on Grace

dylan — dylan@login4:~ — ssh ◂ c

```
[dylan@grace4 ~]$ sinfo
PARTITION        AVAIL  TIMELIMIT    JOB_SIZE    NODES(A/I/O/T)   CPUS(A/I/O/T)
short*           up     2:00:00      1-32        768/0/32/800     34768/1836/1796/3840
medium           up     1-00:00:00   1-128       768/0/32/800     34768/1836/1796/3840
long             up     7-00:00:00   1-64        768/0/32/800     34768/1836/1796/3840
xlong            up     21-00:00:00  1-32        768/0/32/800     34768/1836/1796/3840
vnc              up     12:00:00     1-32        49/63/5/117      2192/3184/240/5616
gpu              up     4-00:00:00   1-32        49/63/5/117      2192/3184/240/5616
bigmem           up     2-00:00:00   1-4         2/5/1/8          160/400/80/640
staff            up     infinite     1-infinite  817/63/37/917    36960/5020/2036/4401
special          up     7-00:00:00   1-infinite  817/63/37/917    36960/5020/2036/4401
[dylan@grace4 ~]$
```

```
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 GCC/10.2.0 Python/3.8.6
source $SCRATCH/my_example_venv/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)

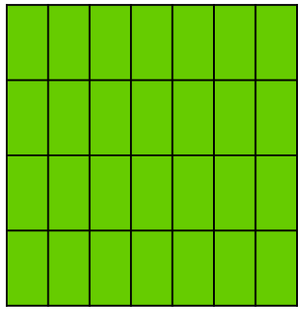
Slurm parameter	Comment
`#SBATCH --time=HH:MM:SS`	Specifies the time limit for the job. Must specify seconds SS on Terra
`#SBATCH --ntasks=NNN`	Total number of tasks (cores) for the job.
`#SBATCH --ntasks-per-node=XX`	Specifies the maximum number of tasks (cores) to allocate per node

`#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.

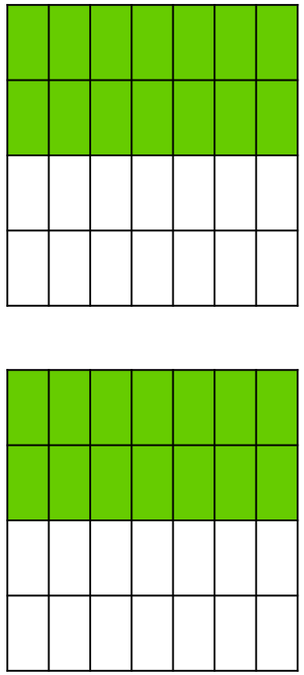


28 cores on
1 compute node

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

Preferred Mapping
(if applicable)

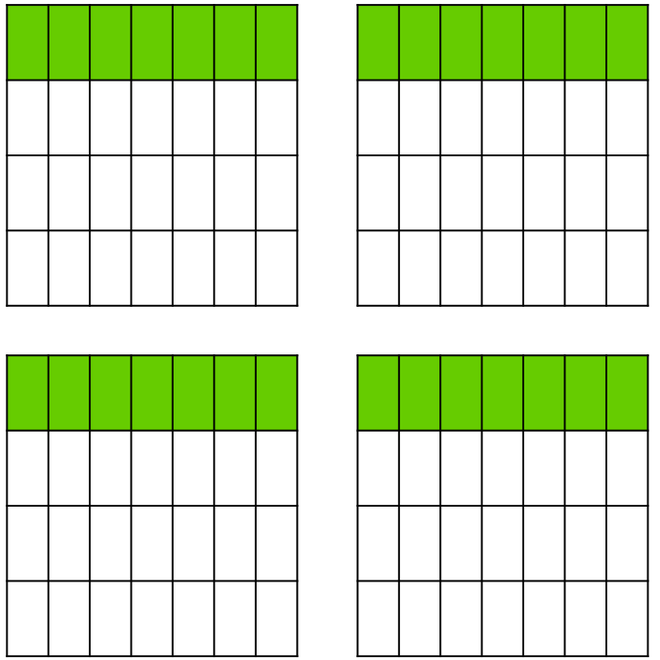
B.



28 cores on
2 compute nodes

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

C.

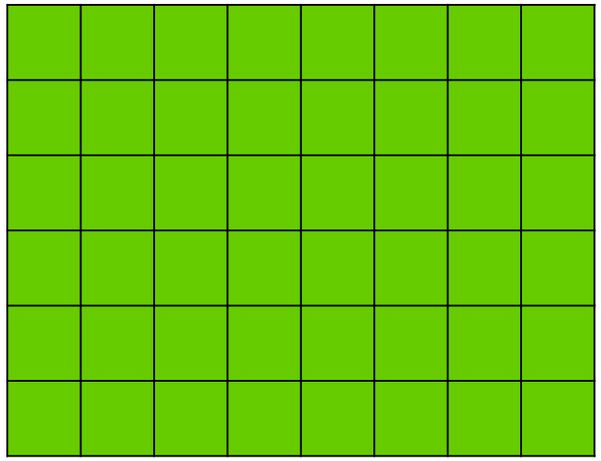


28 cores on
4 compute nodes

#SBATCH --ntasks 28
#SBATCH --tasks-per-node=7

Mapping Jobs to Cores per Node on Grace

A.

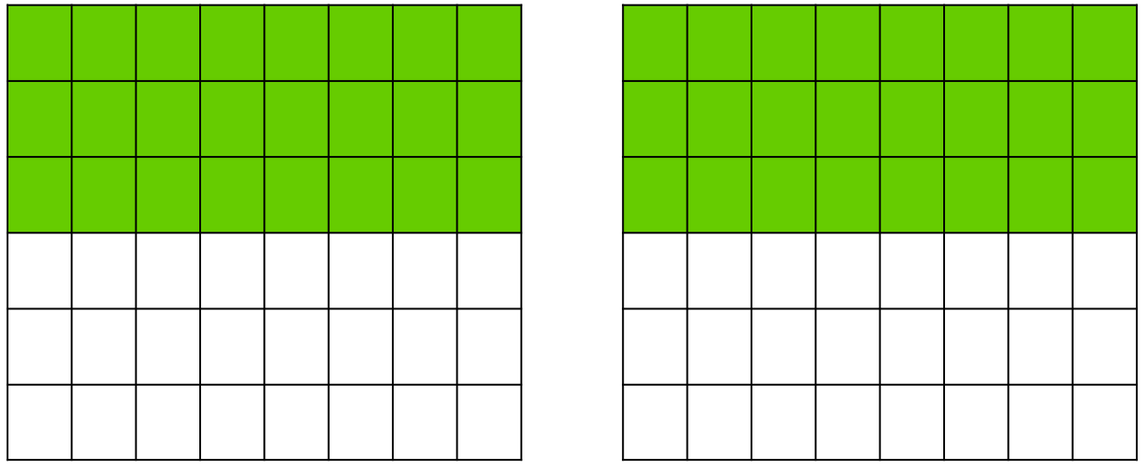


48 cores on
1 compute node

#SBATCH --ntasks=48
#SBATCH --tasks-per-node=48

Preferred Mapping
(if applicable)

B.

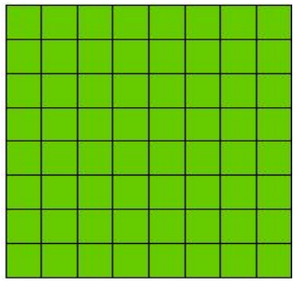


48 cores on
2 compute nodes

#SBATCH --ntasks=48
#SBATCH --tasks-per-node=24

Mapping Jobs to Cores per Node on FASTER

A.

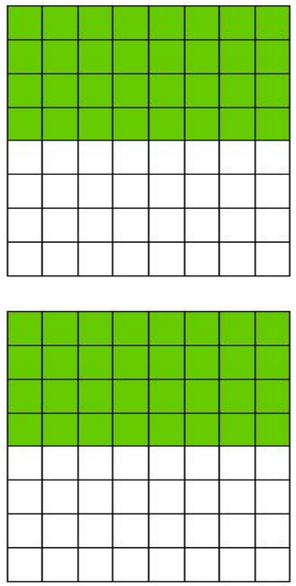


64 cores on
1 compute node

```
#SBATCH --ntasks=64
#SBATCH --ntasks-per-node=64
```

Preferred Mapping
(if applicable)

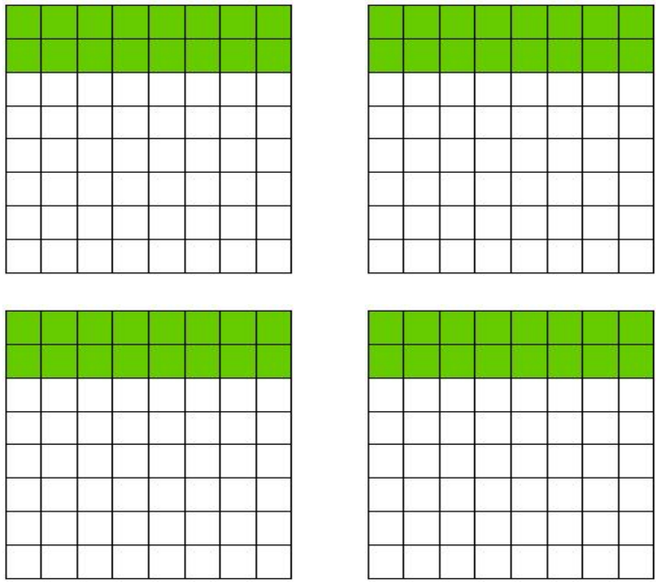
B.



64 cores on
2 compute nodes

```
#SBATCH --ntasks=64
#SBATCH --ntasks-per-node=32
```

C.



64 cores on
4 compute nodes

```
#SBATCH --ntasks=64
#SBATCH --ntasks-per-node=16
```

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.

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

Job Submission and Tracking

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

Include env in your jobfile to list env variables

Batch Job Exercises

Hands-on exercises:

On grace, 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 *
tree
```

Job Exercise: Check Output

Submit job

```
cd $SCRATCH/Intro-to-Grace
```

```
sbatch example01.job
```

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

Check status

```
squeue -u netID
```

```
JOBID    NAME      USER      PARTITION  NODES  CPUS  STATE    TIME         TIME_LEFT  START_TIME          REASON     NODELIST
6853258  jobname   someuser   xlong       2     96   RUNNING  3-07:36:50   16:23:10   2023-01-23T17:27:3   None      c[180,202]
6853257  jobname   someuser   xlong       2     96   RUNNING  3-07:36:56   16:23:04   2023-01-23T17:27:2   None      c[523-524]
```

Check output

```
cat output.example01.<tab autocomplete>
```

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

#SBATCH -o output.example01.%j

Job Exercise: Bad job script

Submit job

```
cd $SCRATCH/Intro-to-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?

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.

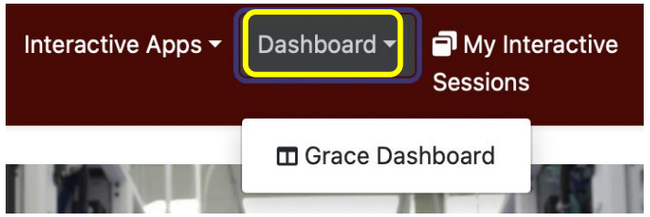
```
$ sbatch myjob
sbatch: error: (from job_submit) your account's balance is not sufficient to submit your job
               Project Account: 123940134739
               Account Balance: 382.803877
               Requested SUs:   18218.666666667
```

- 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



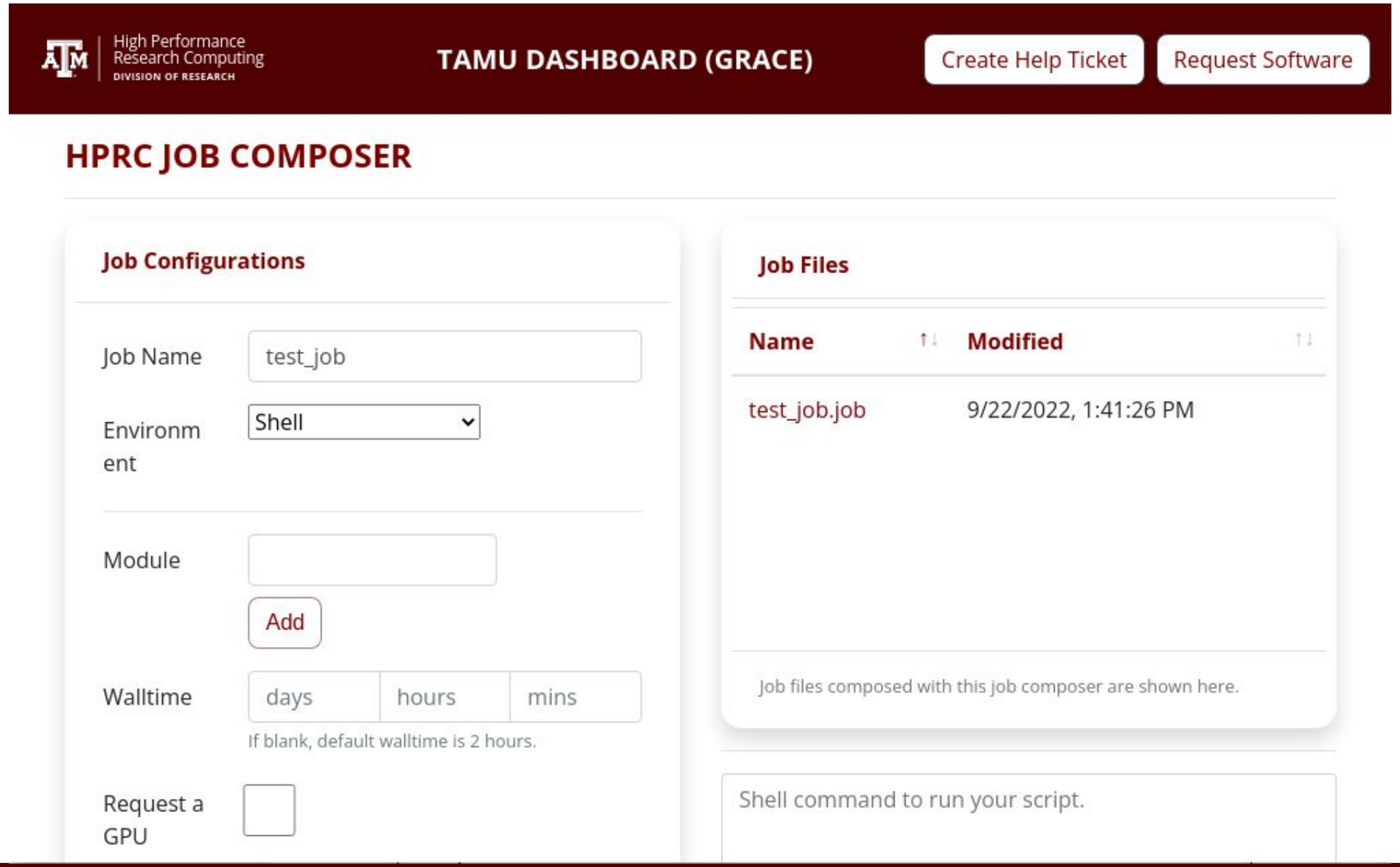
Dashboard

Grace Dashboard

Button

“Job Composer”

(at the bottom)



Other Batch Job Examples

Slurm Job File (Serial Example)

```
#!/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 (2.56GB) 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/2022a

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

SUs = 1.5

Slurm Job File (multi core, single node)

```
#!/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/2022a

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

SUs = 52

Slurm Job File (multi core, multi node)

```
#!/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/2022a

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

SUs = 288

Slurm Job File (serial GPU)

```
#!/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:a100:1               # Request 1 A100 GPUs
#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 CUDA/11.8.0

# run your program
my_cuda_enabled_program
```

SUs = 73

Slurm Job File (multi GPU)

```
#!/bin/bash

##NECESSARY JOB SPECIFICATIONS
#SBATCH --job-name=JobExample5          # Set the job name to "JobExample4"
#SBATCH --time=01:00:00                 # Set the wall clock limit to 1hr
#SBATCH --ntasks=2                      # Request 1 task (core)
#SBATCH --mem=250G                      # Request 250GB per node
#SBATCH --output=Example4Out.%j         # Send stdout and stderr to "Example4Out.[jobID]"
#SBATCH --gres=gpu:a100:2               # Request 1 A100 GPUs
#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 CUDA/11.8.0

# run your program
my_cuda_enabled_program
```

SUs = 192

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)

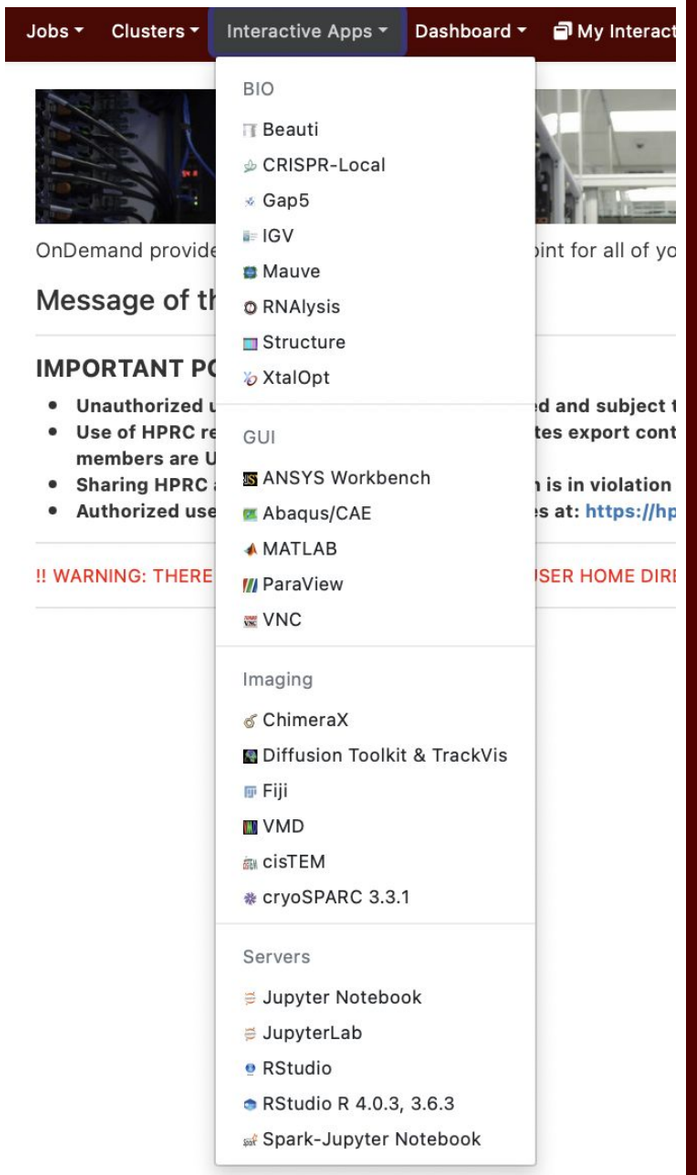
```
pestat -u $USER
```

```
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

- Visualization:
  - portal.hprc.tamu.edu Interactive Apps > choose a visual application
- Large number of concurrent single core jobs
  - Check out *tamulauncher*
    - Useful for running many single core commands concurrently across multiple nodes within a job
    - Can be used with serial or multi-threaded programs
    - 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?

First check the FAQ hprc.tamu.edu/wiki/HPRC:CommonProblems

- FASTER User Guide hprc.tamu.edu/wiki/FASTER
- Email your questions to help@hprc.tamu.edu

Help us, help you -- we need more info

- Which Cluster
- Username
- 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

High Performance
Research Computing
DIVISION OF RESEARCH

Thank you

Questions?