

HIGH PERFORMANCE RESEARCH COMPUTING

Introduction to HPRC Computing Resources

January 23, 2026
Josh Winchell



High Performance
Research Computing
DIVISION OF RESEARCH

Outline

- **Introduction**
 - Setup
 - Usage Policies
 - HPRC Cluster Overview
- **Getting Started**
 - Accessing HPRC Clusters
 - HPRC Portal
- **Software Infrastructure**
 - Module System
 - Python Virtual Environments
- **Cluster Computing with Slurm**
 - Consumable Resources
 - Slurm
 - Job Management Tools

Introduction



Setup For This Course

To access the resources:

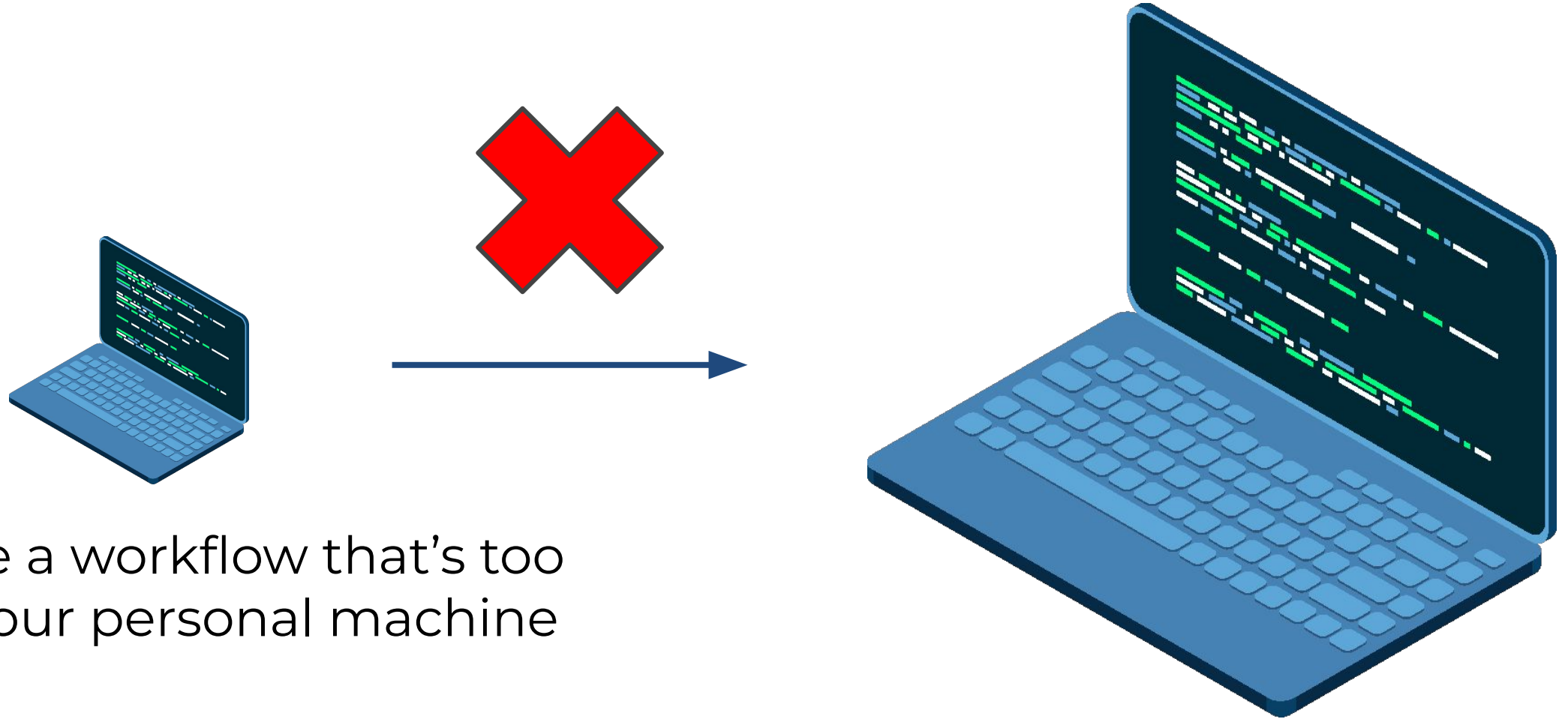
- HPRC Account: https://hprc.tamu.edu/user_services/#accounts
- TFA set up: <https://it.tamu.edu/duo>

To do the exercises:

- Basic Linux/Unix skills
 - 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:
<https://www.youtube.com/texasamhprc>

Answers to frequently asked questions can be found on our Knowledge Base (KB):
<https://hprc.tamu.edu/kb/FAQ/Accounts>

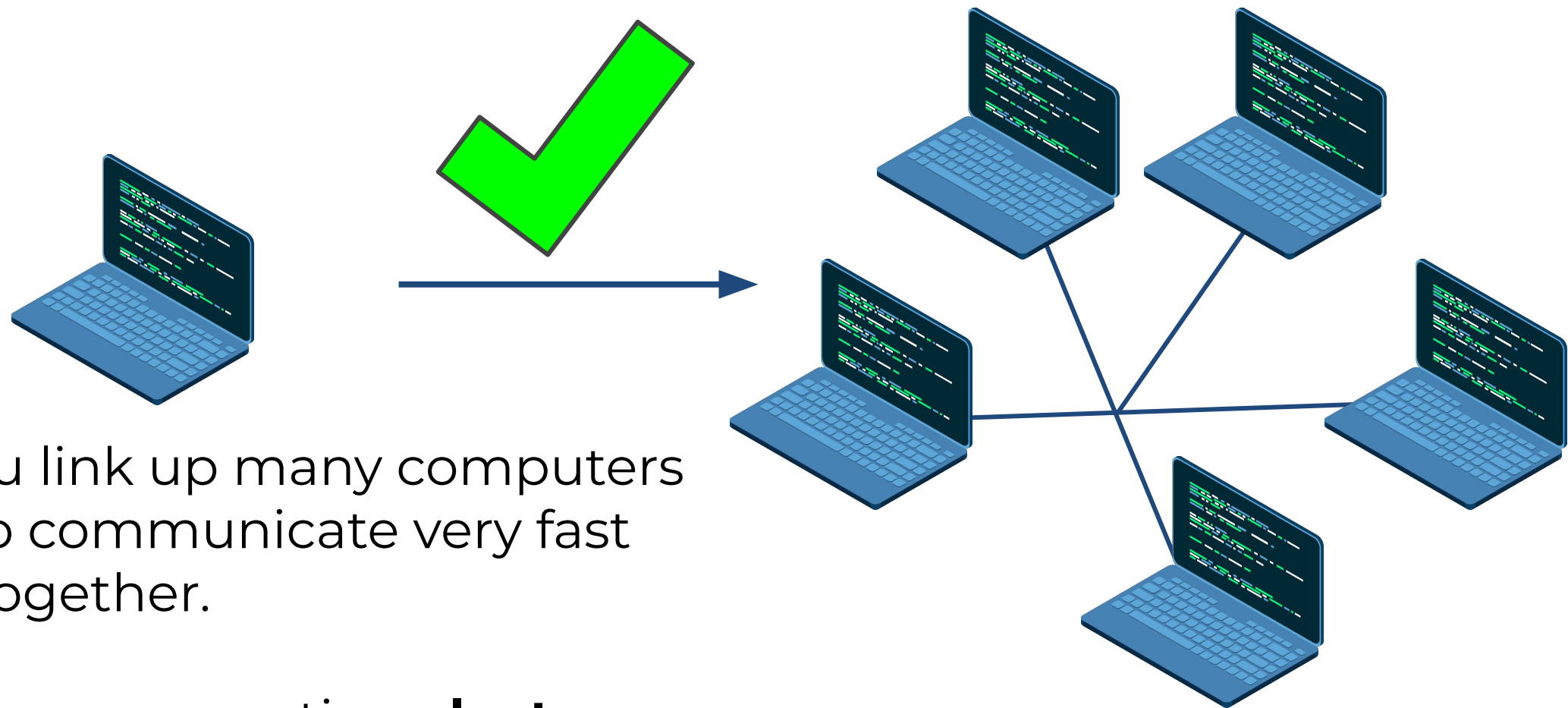
What is Supercomputing?



You have a workflow that's too big for your personal machine

...But you don't just get a bigger computer...

What is ~~Super~~Cluster computing?



Instead, you link up many computers (“nodes”) to communicate very fast and work together.

I.e. you make a computing **cluster**.

Clusters Are For You!

What kinds of problems are solved by cluster computing?

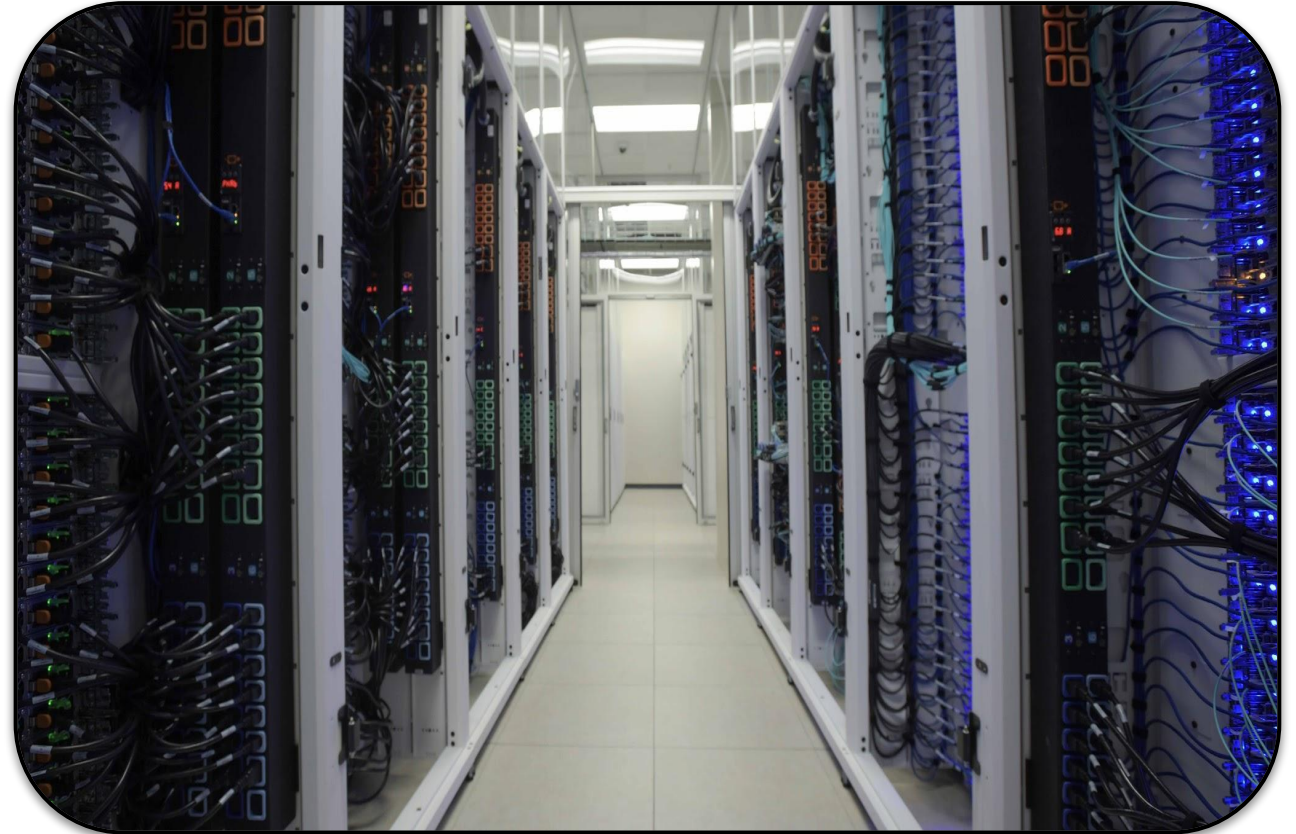
- Problems that are too big to fit in a laptop or workstation, due to limitation on memory, core count, gpu count, etc.
- Problems that scale well with more CPU cores or memory
- Single-threaded problems with many permutations
- Problems that require large high speed storage and/or interconnect

HPRC Clusters

- We operate five clusters
- Today we'll focus on just one: **Grace**
 - It is our largest cluster
 - It is dedicated for TAMU usage

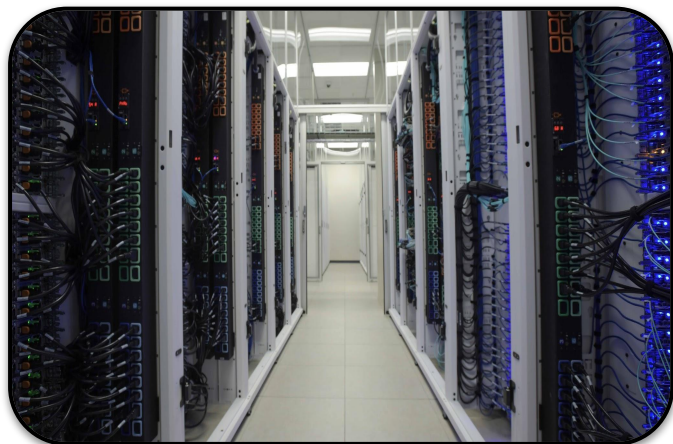
Others:

- Launch
- FASTER
- ViDaL
- ACES



See hprc.tamu.edu/resources
for more info on all our clusters

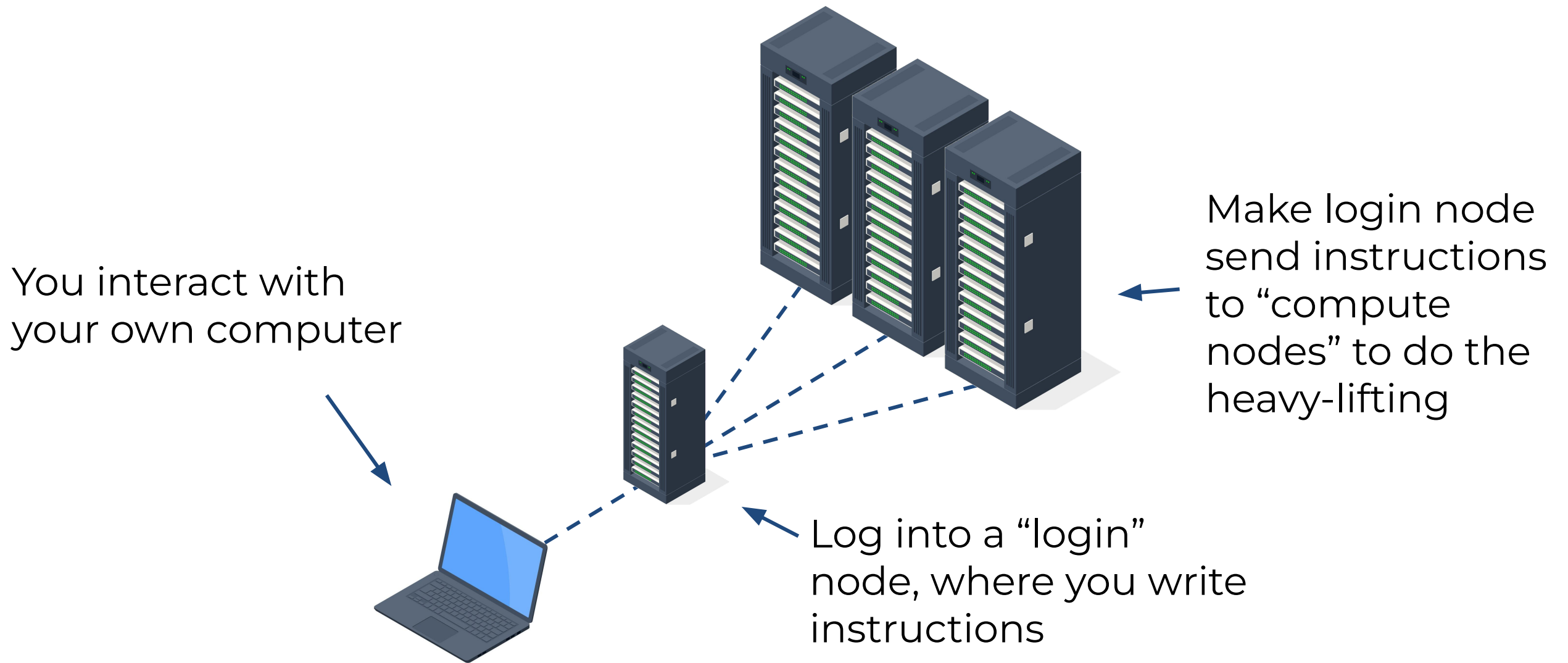
Grace Overview



Host Name:	grace.hprc.tamu.edu
Operating System:	Linux (RHEL 8)
Total Compute Cores/Nodes:	45,376 cores 940 nodes
Compute Nodes:	800 48-core compute nodes 100 48-core GPU nodes, each with two A100s 9 48-core GPU nodes, each with two RTX 6000s 8 48-core GPU nodes, each with 4 T4s 15 48-core GPU nodes, each with two A40s 8 80-core large memory nodes, each with 3TB RAM
Interconnect:	Mellanox HDR 100 InfiniBand
Peak Performance:	6.3 PFLOPS

More details at: <https://hprc.tamu.edu/kb/User-Guides/Grace>

Computing on HPRC Clusters



(This image is not to-scale: we have way more machines than this!)

HPRC Knowledge Base

See our Knowledge Base for announcements, more hardware details, and more about the subjects we cover today.

hprc.tamu.edu/kb
hprc.tamu.edu/kb/User-Guides/Grace

The image shows two overlapping screenshots of the Texas A&M HPRC Knowledge Base website. The top screenshot displays the 'Home' page, which includes a navigation bar with links to Home, Quick Start, User Guides, Software, Helpful Pages, and FAQ. The main content area features a 'High Performance Research Computing Home Page' header, an 'Announcements' section with a bullet point about the 'FASTER Cluster Status', and a 'Getting an Account' section with links to 'Understanding HPRC' and 'New to HPRC's resources?'. A 'Table of contents' sidebar on the right lists various topics. The bottom screenshot shows the 'Hardware' page, specifically the 'Grace: A Dell x86 HPC Cluster' section. It includes a sidebar with a 'User Guides' menu, a central image of server racks, and a 'Table of contents' sidebar on the right. Below the image, there are input fields for 'System Name' (Grace) and 'Host Name' (grace.hprc.tamu.edu).

Texas A&M HPRC

Home Quick Start User Guides Software Helpful Pages FAQ

Home

High Performance Research Computing Home Page

Announcements

- **FASTER Cluster Status:** Service unit accounting active as of August 1, 2023.

Getting an Account

- **Understanding HPRC:** For a brief overview of HPRC and what resources we offer, check out [this page](#) and watch [this video](#) in our getting started series on YouTube.
- **New to HPRC's resources?** [This page](#) explains the HPRC resources available to the TAMU community. Also

Table of contents

- Announcements
- Getting an Account
- Using The Clusters
 - QuickStart Guides
 - Batch Jobs
- HPRC's YouTube Channel

Home Quick Start User Guides Software Helpful Pages FAQ

Hardware

Grace: A Dell x86 HPC Cluster



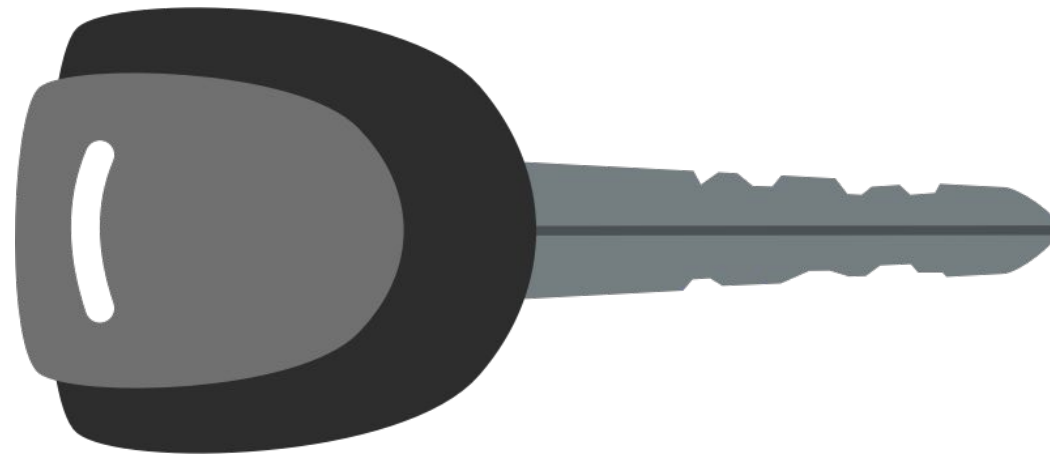
System Name: Grace

Host Name: grace.hprc.tamu.edu

Table of contents

- Grace: A Dell x86 HPC Cluster
- Compute Nodes
- Usable Memory for Batch Jobs
- Login Nodes
- Mass Storage
- Interconnect
- Namesake

Getting Started



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 .

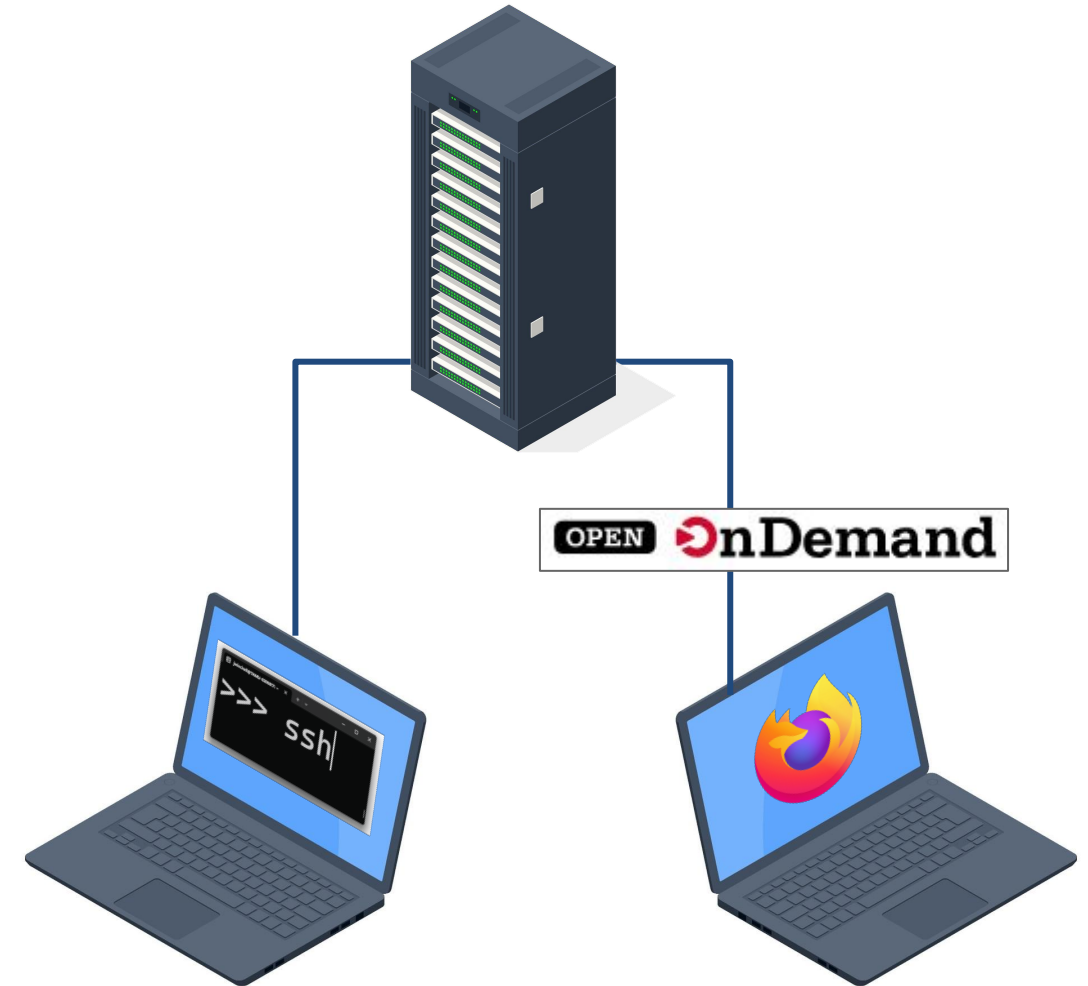
hprc.tamu.edu/policies

Getting on a Cluster

There are two ways you can access the Grace (and FASTER) clusters:

1. Using ssh (or related tools) on your local command line
2. Through the Open OnDemand portal in a web browser

If you like Linux, you can use option 1.
You can also use Linux in option 2
–but you don't have to!



Accessing Clusters via SSH

- SSH (Secure Shell) allows users to establish a connection between their local machine and the TAMU HPRC clusters.
- SSH Programs:

Operating System	Windows	MacOS	Linux
Programs	<u>MobaXTerm*</u> <u>PowerShell</u> <u>Windows Command Prompt</u> <u>PuTTY SSH</u>	<u>Terminal*</u>	<u>Terminal*</u>
* Recommended			

- If off-campus, connect with VPN first: u.tamu.edu/VPnetwork
- If on-campus or connected to VPN, log in with:

```
ssh NetID@grace.hprc.tamu.edu
```

← Green box/highlights =
text typed into a shell

Accessing Clusters via Portal

- (First, if you're off-campus, connect to TAMU VPN: u.tamu.edu/VPnetwork)
- HPRC webpage: hprc.tamu.edu: Portal dropdown menu



HPRC Portal

TAMU HPRC OnDemand (Grace)

Files ▾

Jobs ▾

Clusters ▾

Interactive Apps ▾

Dashboard ▾

Utilities ▾

 My Interactive Sessions

 Help ▾

 Logged in as username

 Log Out



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

Message of the Day

Shared Storage Status for Grace and FASTER Clusters, December 3

We are continuing to experience performance issues with the shared storage for Grace and FASTER.

Our only possible solution remains to reclaim more free space so that the storage consistently has at least 800 TB free at any given time. Unfortunately, new data from users is outpacing the inactive/old data that we have purged since September.

Where possible, please delete, compress or move any data that will not be used for active work on Grace and FASTER to other TAMU storage resources.

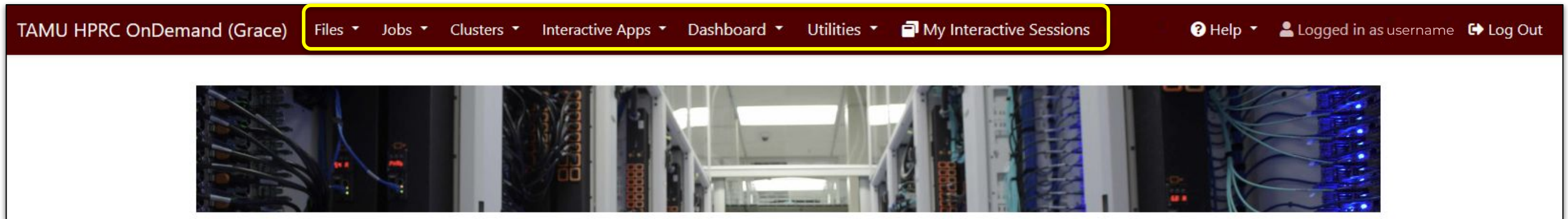
We apologize if the storage issues are disrupting your work. As of now, we do not have an estimated time of resolution.

Thank you for your patience.

IMPORTANT POLICY INFORMATION

- **Unauthorized use of HPRC resources is prohibited and subject to criminal prosecution.**
- **Use of HPRC resources in violation of United States export control laws and regulations is prohibited. Current HPRC staff members are US citizens and legal residents.**

HPRC Portal Overview



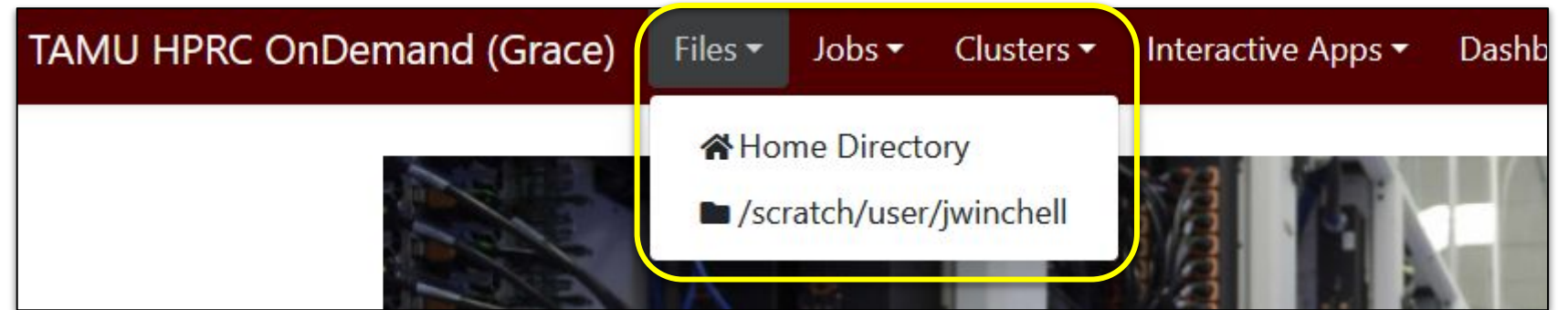
[TAMU HPRC OnDemand \(Grace\)](#): (click to return to portal homescreen)

Files :	file browser on the cluster
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
Dashboard :	view file quotas and computing account allocations
Utilities :	check cluster status
My Interactive Sessions *	check interactive app status

**(We'll talk about these three later)*

Portal File Browser

The Portal includes a file browser in which you can do many of the things a terminal would do... or just launch a terminal.



TAMU HPRC OnDemand (Grace) Files Jobs Clusters Interactive Apps Dashboard My Interactive Sessions Help Logged in as jwinchell Log Out

Open in Terminal New File New Directory Upload Download Copy/Move Delete

Home Directory
/scratch/user/jwinchell

Change directory Copy path

Show Owner/Mode Show Dotfiles Filter: Showing 10 of 43 rows - 0 rows selected

Type	Name	Size	Modified at
Folder	fortran_tutorial	-	4/8/2022 10:06:50 AM
Folder	geant4_workdir	-	3/21/2022 10:11:25 AM

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. Backed-up.
/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; NOT backed up.

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

View file usage and quota limits on the command line with:

showquota

Do NOT share your home or scratch directories.

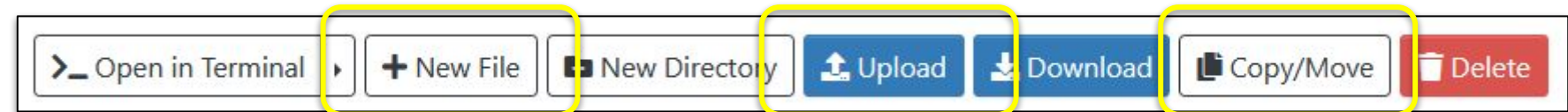
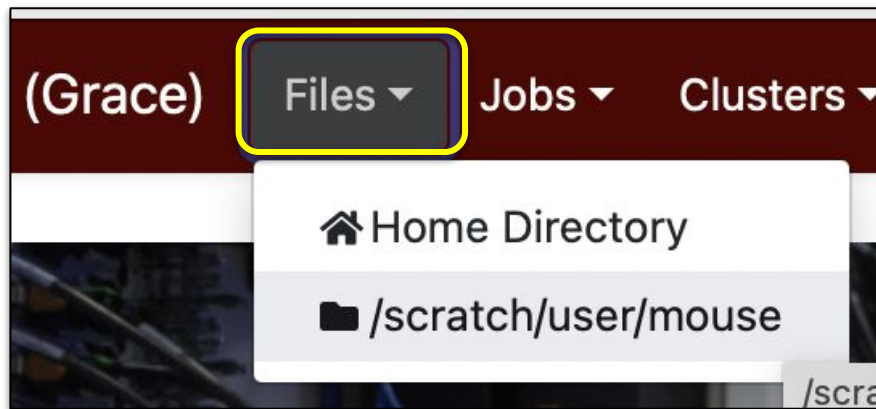
Request a group directory for sharing files using the portal Dashboard.

Note also: we are providing **work** space, not **storage** space.

Your data will be there as long as your account is active, but after that–no promises!

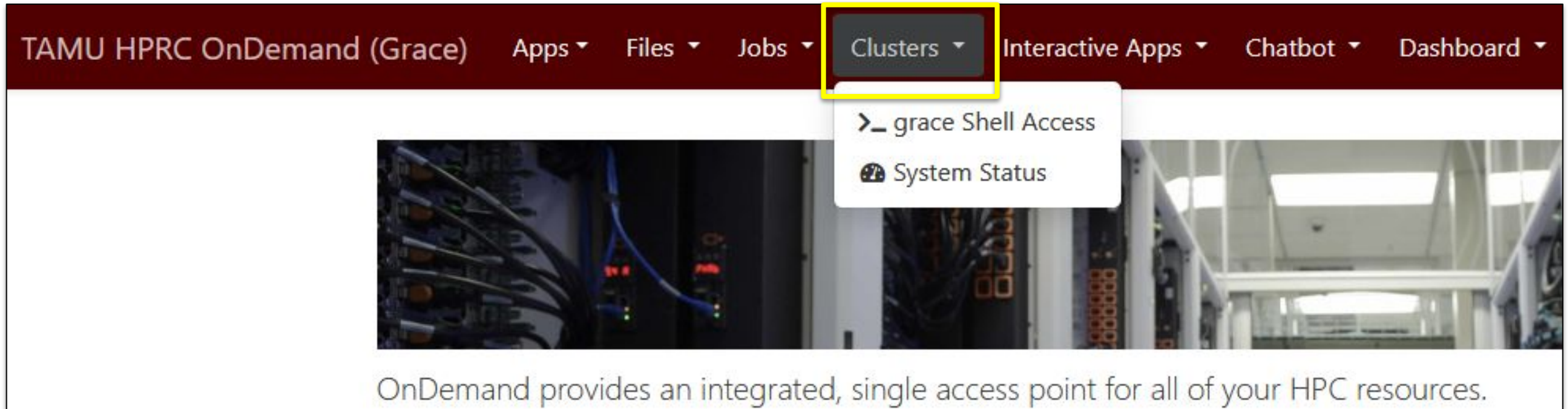
Hands-on Exercise

1. Enter your Scratch directory from the Portal:
Files → /scratch/user/<NetID>
2. Create or upload a file.
3. Move the file to your home directory.



Linux Command Line

- In Portal: “Clusters” dropdown → “Shell Access”



Linux Command Line

```
=====
|      Texas A&M University High Performance Research Computing      |
|                                                                    |
| Website:      https://hprc.tamu.edu                               |
| Consulting:   help@hprc.tamu.edu (preferred) or (979) 845-0219 |
| ACES Documentation: https://hprc.tamu.edu/kb/User-Guides/ACES |
| FASTER Documentation: https://hprc.tamu.edu/kb/User-Guides/FASTER |
| Grace Documentation: https://hprc.tamu.edu/kb/User-Guides/Grace |
| Terra Documentation: https://hprc.tamu.edu/kb/User-Guides/Terra |
| YouTube Channel: https://www.youtube.com/texasamhprc           |
|                                                                    |
|=====

*****
*      === IMPORTANT POLICY INFORMATION ===      *
* - Unauthorized use of HPRC resources is prohibited and subject to *
* criminal prosecution. *
* - Use of HPRC resources in violation of United States export control *
* laws and regulations is prohibited. Current HPRC staff members are *
* US citizens and legal residents. *
* - Sharing HPRC account and password information is in violation of *
* Texas State Law. Any shared accounts will be DISABLED. *
* - Authorized users must also adhere to ALL policies at: *
*      https://hprc.tamu.edu/policies/ *
*****

!! WARNING: THERE ARE ONLY NIGHTLY BACKUPS OF USER HOME DIRECTORIES. !!

Please restrict usage to 8 CORES across ALL login nodes.
Users found in violation of this policy will be SUSPENDED.

To see these messages again, run the motd command.

Your current disk quotas are:
Disk      Disk Usage  Limit  File Usage  Limit
/home/username      899M   10.0G    7259   10000
/scratch/user/username  477.7G  1.0T   165425  250000
/scratch/group/group1  32.9T   60.0T   5322917 15000000
* Quota increase for /scratch/group/group1 will expire on Aug 26, 2026
Type 'showquota' to view these quotas again.
username@login3:~$
```

Prints out a lot of text on
successful login
--same as ssh login

Message of the
day & policy info

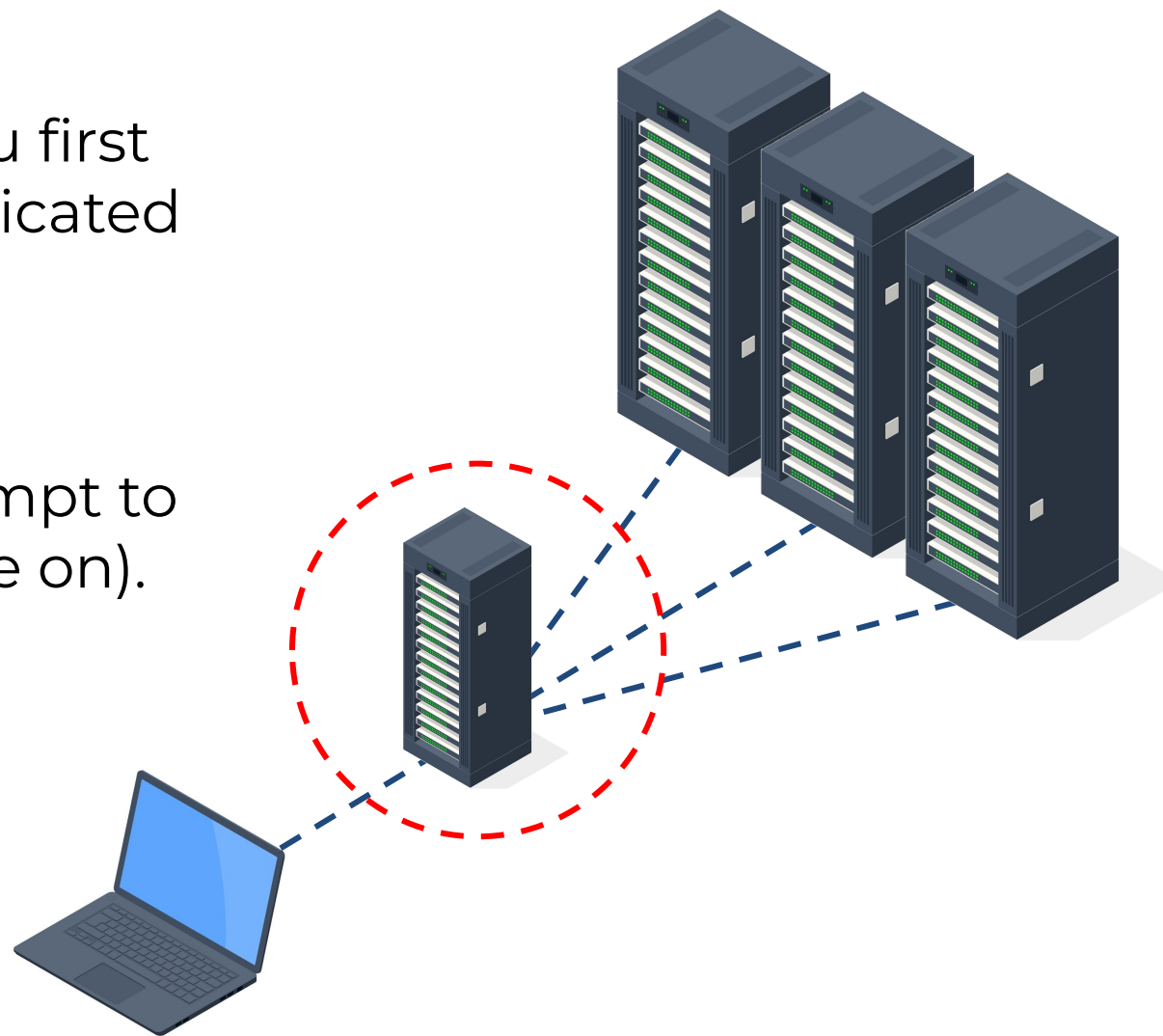
Your quota info

Command Prompt

Logged-In

Remember, when you first log in, you are on dedicated *login nodes*.

Grace has 5 of them (check your shell prompt to see which one you are on).

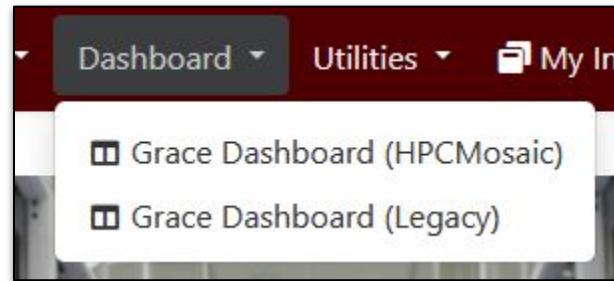


These are not for running big processes!

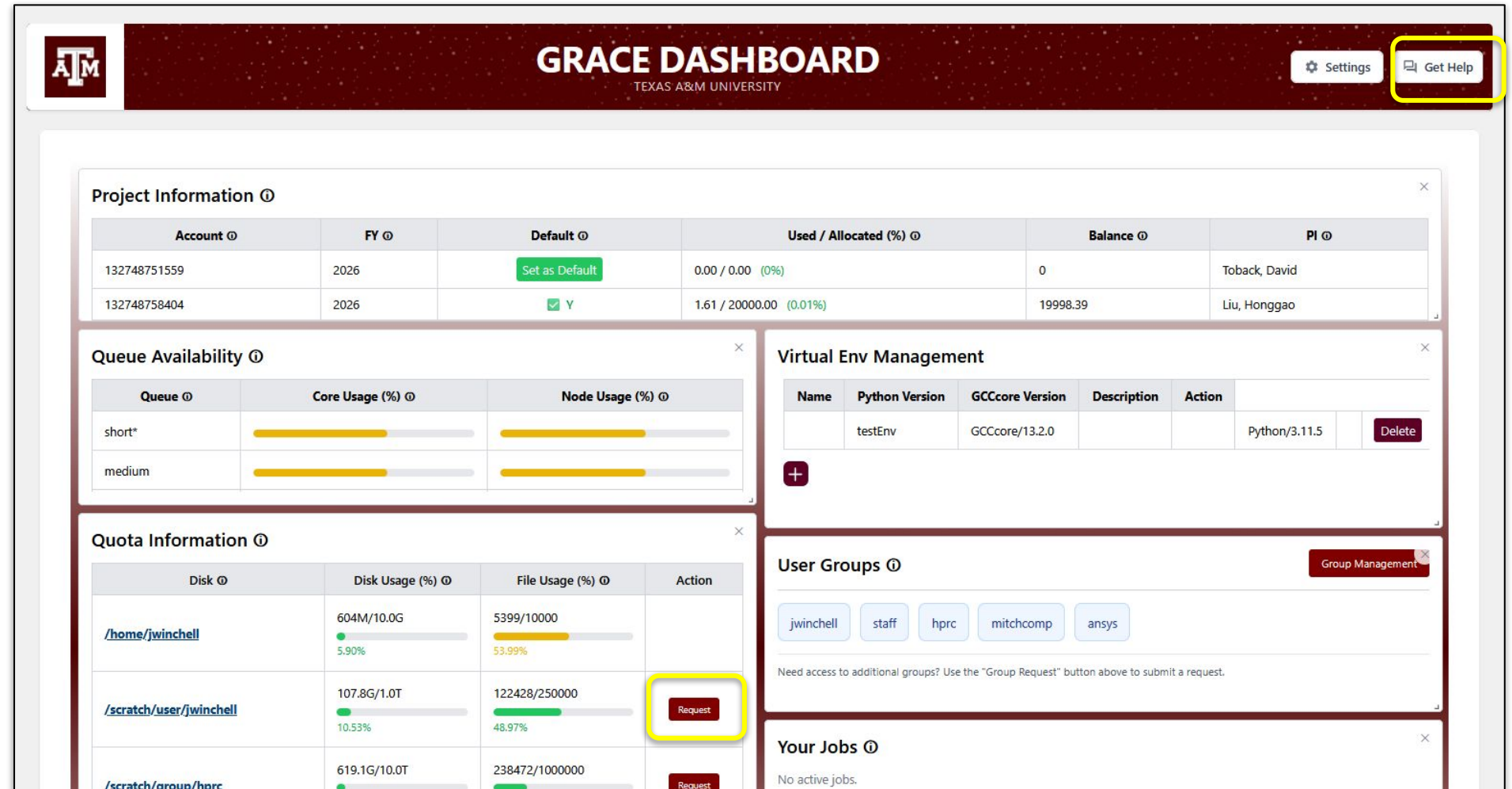
There are rules:

- No processes longer than 1 hr
- Sessions idle for 1 hr will be killed.
- Do not use more than 8 cores.
- Do not use `sudo`.

HPRC Portal Dashboard



- Request help or new software
- Request quota increases
- Manage user groups
- Check job and cluster status



GRACE DASHBOARD
TEXAS A&M UNIVERSITY

Settings Get Help

Project Information

Account	FY	Default	Used / Allocated (%)	Balance	PI
132748751559	2026	Set as Default	0.00 / 0.00 (0%)	0	Toback, David
132748758404	2026	☒ Y	1.61 / 20000.00 (0.01%)	19998.39	Liu, Honggao

Queue Availability

Queue	Core Usage (%)	Node Usage (%)
short*		
medium		

Virtual Env Management

Name	Python Version	GCCcore Version	Description	Action
testEnv		GCCcore/13.2.0		Delete

Quota Information

Disk	Disk Usage (%)	File Usage (%)	Action
/home/jwinchell	604M/10.0G 5.90%	5399/10000 53.99%	Request
/scratch/user/jwinchell	107.8G/1.0T 10.53%	122428/250000 48.97%	Request
/scratch/group/hprc	619.1G/10.0T	238472/1000000	Request

User Groups

[Group Management](#)

jwinchell staff hprc mitchcomp ansys

Need access to additional groups? Use the "Group Request" button above to submit a request.

Your Jobs

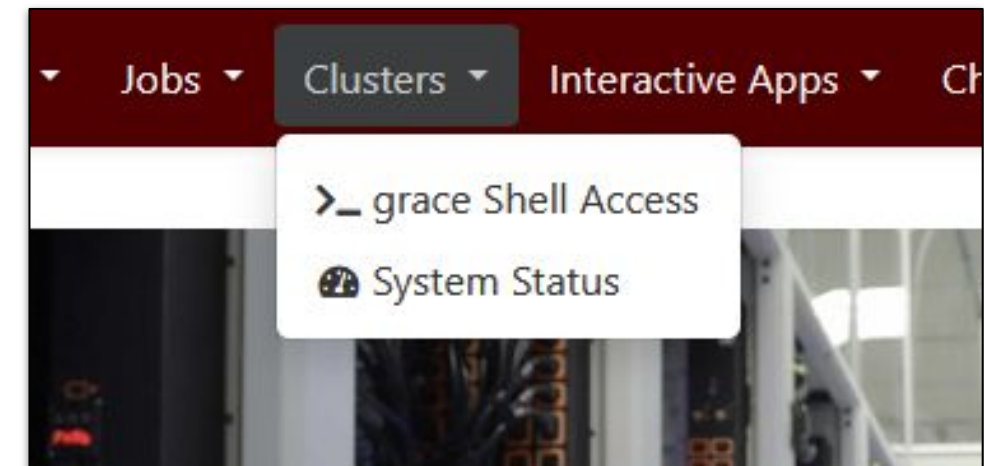
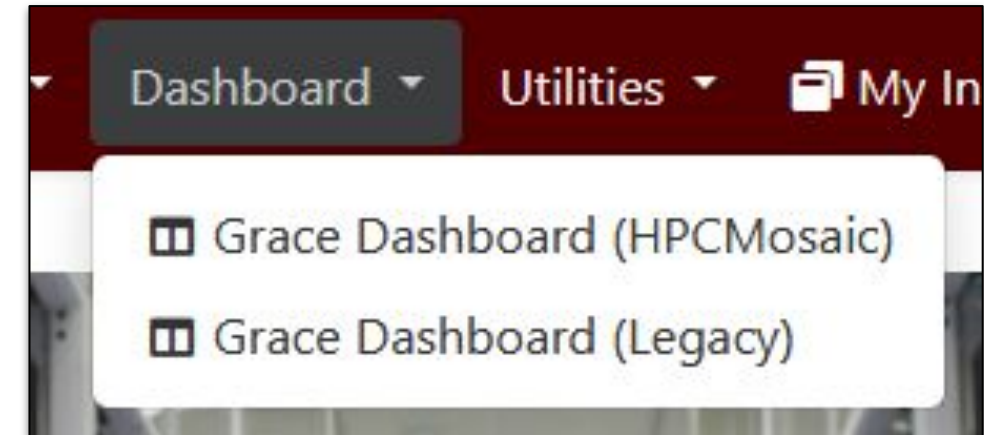
No active jobs.

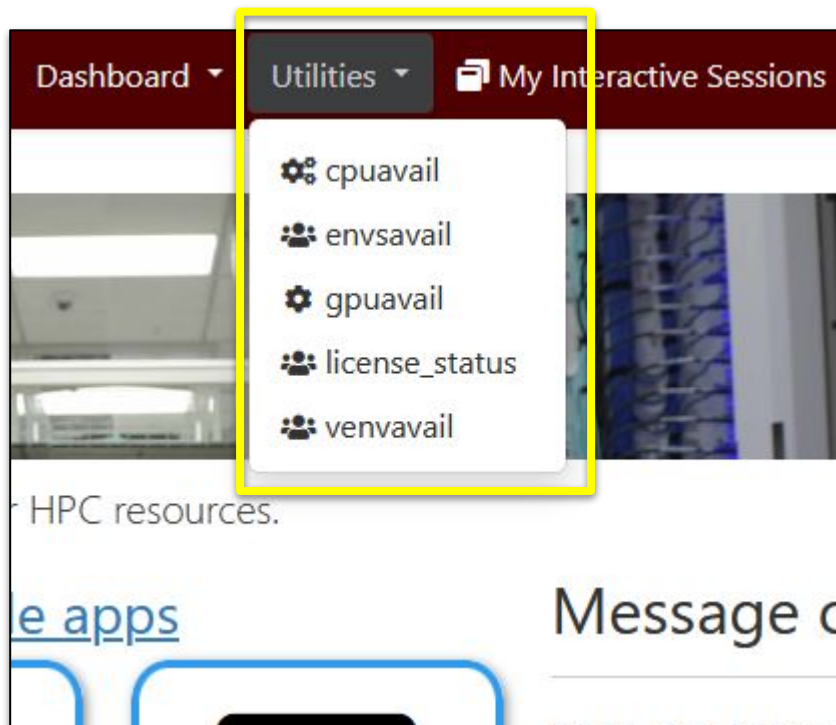
Hands-on Exercise

Check your file Quota:

1. In the Portal:
Dashboard → Grace Dashboard
(either version)
2. In the shell/command-line:
(shown when you log in
- else type `showquota`)

Verify that they match!





Utilities

See availability of hardware, licenses, and environments.

(You can also call these on the command line)

Grace OnDemand Portal / Grace GPU Availability

Grace GPU Availability

This displays the CONFIGURATION of installed GPUs and the status of GPUs readily available for jobs. GPUs and GPU nodes not in the AVAILABILITY table are busy with other jobs and will be available after jobs have completed.

Output generated: 2025-09-03 16:04:39

\$ gpuavail

CONFIGURATION	
NODE TYPE	NODE COUNT
gpu:a100:2	100
gpu:a40:3	15
gpu:rtx:2	9
gpu:t4:4	8

Grace OnDemand Portal / Grace License Status

Grace License Status

Output generated: 2025-09-03 16:05:21

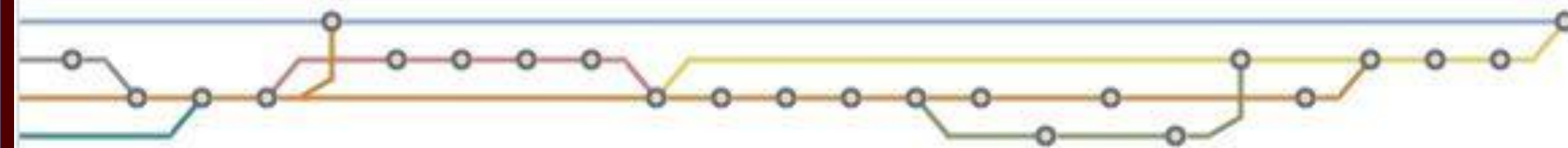
\$ license_status -a

Abaqus | ANSYS | COMSOL | ConvergeCFD | ECLIPSE | FDTD | FEMZIP | FLOW3D | Hyperworks | Intel | INTERSECT | LS-DYNA | Madymo | Matlab | Matlab_MDCS | Metashape | MOE | Oasys | PGI_compilers | StarCCM | Schrodinger |

License status for Abaqus:

License Name	# Issued	# In Use	# Available
abacus_teaching	320	0	320
abacus_extended_teaching	40	0	40
cse_teaching	1	0	1

Software Infrastructure

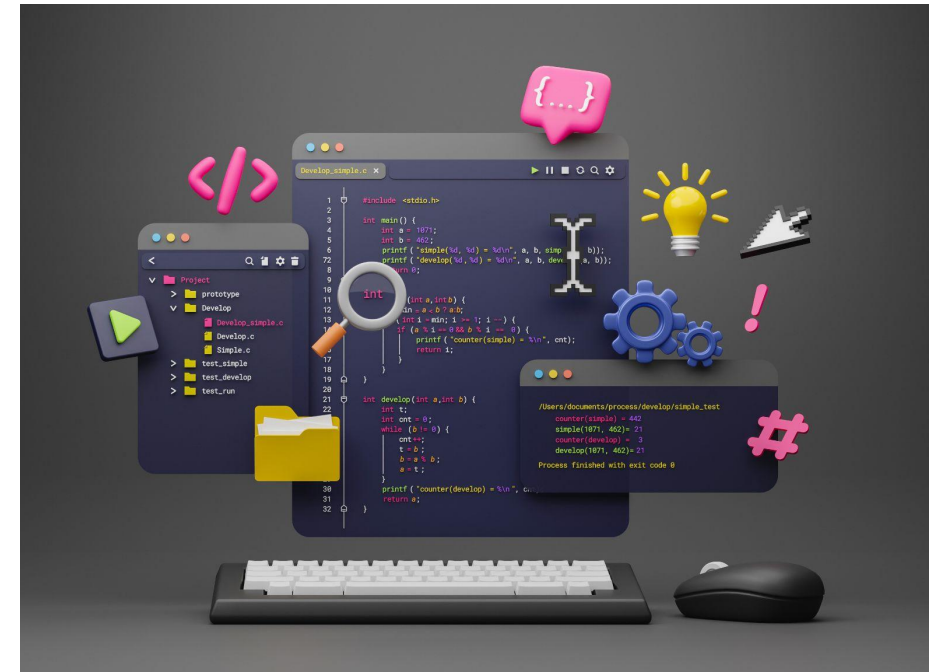


(Warning: Linux)

Getting Software

You need software! How to get it?

1. Check to see if we have it installed as a 'module'* already.
2. If not, see about getting it installed:
 - a. Install for yourself in your \$SCRATCH.
 - b. Ask us to install system-wide as a module.



**or possibly an “Interactive App”*

Software Modules

- Lots of software is on the cluster already, but your programs don't know how to find them.
- The [module](#) system ('lmod') is how we manage software--and how you'll tell your programs where to find software

SOFTWARE MODULES ON THE GRACE CLUSTER			
ACES Software Modules FASTER Software Modules Grace Software Modules			
(Last Updated: Dec 04, 2025)			
Show 10 entries Search: Numpy			
Name	Versions available	Description	Notes
EINOPS	einops/0.8.0 einops/0.7.0 einops/0.4.1	Flexible and powerful tensor operations for readable and reliable code. Supports numpy, pytorch, tensorflow, jax, and others.	
ML_DTYPES	ml_dtypes/0.5.0 ml_dtypes/0.3.2	ml_dtypes is a stand-alone implementation of several NumPy dtype extensions used in machine learning libraries, including: bfloat16: an alternative to the standard float16 format float8_*: several experimental 8-bit floating point representations including: float8_e4m3b11fnuz float8_e4m3fn float8_e4m3fnuz float8_e5m2 float8_e5m2fnuz	
MRCFILE	mrcfile/1.5.0	mrcfile is a Python implementation of the MRC2014 file format, which is used in structural biology to store image and volume data. It allows MRC files to be created and opened easily using a very simple API, which exposes the file's header and data as numpy arrays. The code runs in Python 2 and 3 and is fully unit-tested. This library aims to allow users and developers to read and write standard- compliant MRC files in Python as easily as possible, and with no dependencies on any compiled libraries except numpy. You can use it interactively to inspect files, correct headers and so on, or in scripts and larger software packages to provide basic MRC file I/O functions.	
NUMBA	numba/0.62.0 numba/0.61.2 numba/0.60.0 numba/0.58.1 numba/0.56.4	Numba is an Open Source NumPy-aware optimizing compiler for Python sponsored by Continuum Analytics, Inc. It uses the remarkable LLVM compiler infrastructure to compile Python syntax to machine code.	

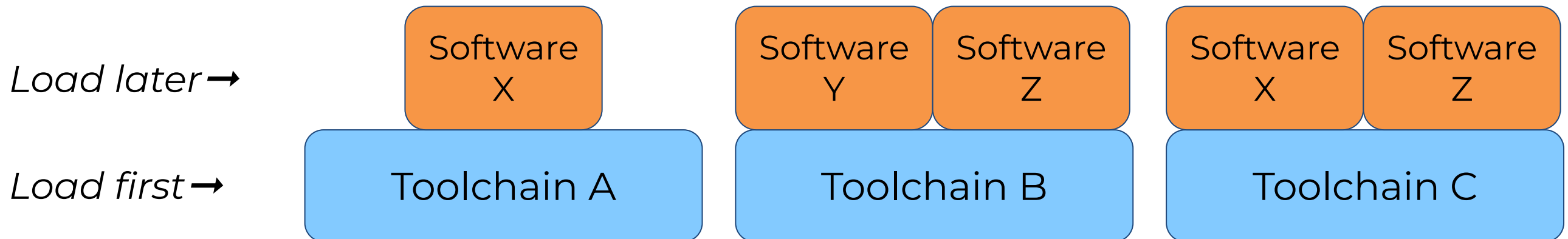
See modules at: <https://hprc.tamu.edu/software/grace/>
More info: <https://hprc.tamu.edu/kb/Software/useful-tools/Modules>

Module Structure

- Module names have the format:

`software-name/version[-dependency-version (optional)]`
TensorFlow/2.11.0-CUDA-11.7.0

- Modules are organized hierarchically:
 - Underlying 'toolchain' modules are used to build various softwares
 - Load the toolchain dependency first, then the software you want
 - Don't mix toolchains!



Modules: Search for Software

module avail

- Lists all available modules (may be slow).
- Navigation:
 - spacebar, arrows, or j and k
 - quit with q
- Case-sensitive search: /
- Use **mla** instead to save results to a file as well (will be named `module.avail.grace` or similar)

module spider <word>

- Case-sensitive search for modules with “word” in name.
- Provide an *exact name with version* to see dependencies.

```
jwinchell@login5:~$ module spider Python

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

Versions:
  Python/2.7.18-bare
  Python/2.7.18
  Python/3.8.6
  Python/3.9.5-bare
  Python/3.9.5
  Python/3.9.6-bare
  Python/3.9.6
  Python/3.10.4-bare
  Python/3.10.4
  Python/3.10.8-bare
  Python/3.10.8
  Python/3.11.3
  Python/3.11.5
  Python/3.12.3
  Python/3.13.1
Other possible modules matches:
  Biopython Boost.Python Brotli-python GitPython IPython Python-bundle-PyPI bx-python cuda_python flatbuffers-python
...
```


Modules: Manage Software

`module list`

- See what modules are loaded in your current session

`module load <module>`

- add <module> paths to the current environment variables

`module purge`

- Unload all modules

```
jwinchell@login5:~$ module list
No modules loaded
jwinchell@login5:~$ module load GCCcore/14.2.0
jwinchell@login5:~$ module list

Currently Loaded Modules:
  1) GCCcore/14.2.0

jwinchell@login5:~$ module purge
jwinchell@login5:~$ module list
No modules loaded
jwinchell@login5:~$
```

These commands also have shorthands:

```
ml
ml <module>
ml purge
```

Example: Load Module with Dependencies

Search for the software “snakemake”;
it saves a file of available modules.

```
mla snakemake
```

```
snakemake/5.26.1-Python-3.8.2  
snakemake/6.1.0  
snakemake/6.10.0  
snakemake/7.22.0  
snakemake/7.32.3  
snakemake/8.4.2
```

Choose a specific version and use it
to search for its dependencies.

```
module spider snakemake/8.4.2
```

```
-----  
snakemake: snakemake/8.4.2  
-----
```

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/8.4.2" module is available to load.

```
GCC/12.3.0  OpenMPI/4.1.5
```

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

```
module load GCC/12.3.0  OpenMPI/4.1.5  snakemake/8.4.2
```

Module Commands Summary

Commonly-used module commands and their shorthands:

Command	Shorthand	Description
<code>module avail</code>	<code>m1a</code> <i>(saves a searchable file as well)</i>	See what modules can be loaded now.
<code>module spider</code>	<code>m1 spider</code>	Search for modules and find dependencies.
<code>module list</code>	<code>m1</code>	See what modules have been loaded already.
<code>module load foo bar</code>	<code>m1 foo bar</code>	Load specified modules.
<code>module unload foo bar</code> <code>module load baz goo</code>	<code>m1 -foo -bar baz goo</code>	Load and unload specified modules.
<code>module purge</code>	<code>m1 purge</code>	Unload all modules.

Hands-on Exercise: Module Loading

1.

```
mla blast+
```

-See which versions of BLAST+ are available.

2.

```
ml BLAST+/2.16.0
```

-error! You cannot do that yet.

3.

```
module spider BLAST+/2.16.0
```

-Learn how to load this module.

4.

```
ml            BLAST+/2.16.0
```

-Fill in the blank (with the correct toolchains) to load this module.

5.

```
ml
```

-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 all loaded modules.

8.

```
ml purge
```

-Remove all loaded modules.

Module Usage Practices

- Most software installed as modules are available to all users.
- 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: `m1 GCC/13.2.0` instead of `m1 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/kb/Software/useful-tools/Modules>

Installing Software

- Researchers can install software in their own directories.
 - Exact steps depend on the software.
 - You **CANNOT** run the "sudo" command when installing software.
 - Watch your file quotas! Install in \$SCRATCH!
- Contact us if you need help
 - We can install software *cluster-wide*.
 - Requests can be sent from the Dashboard.
- License-restricted software
 - Check on command line with `license_status`
 - Contact help@hprc.tamu.edu

Python and Virtual Environments

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

Once you have loaded the appropriate Python module, you can install these additional packages yourself using the *Virtual Environment* feature. Instructions are on our KB:

<https://hprc.tamu.edu/kb/Software/Python/#hprc-venv-management-tools>

Our HPRC student workers have developed a “create_venv” tool to help!

Hands-on Exercise: Python Software Install

1. `m1 purge`
`m1 GCCcore/13.2.0 Python/3.11.5`

2. `create_venv myEnv`

3. `source activate_venv myEnv`

4. `python -c "import pytime"`

5. `pip install python-time`

6. `python -c "import pytime; print(pytime)"`

7. `deactivate`

-Set up the underlying Python module

-Use HPRC's *create_venv* tool;
Create a virtual environment in
\$SCRATCH/virtual_envs

-Activate virtual environment.

-Check if python-time is installed (it is not).

-Install python-time.

-Where is python-time installed?

-Close virtual environment.

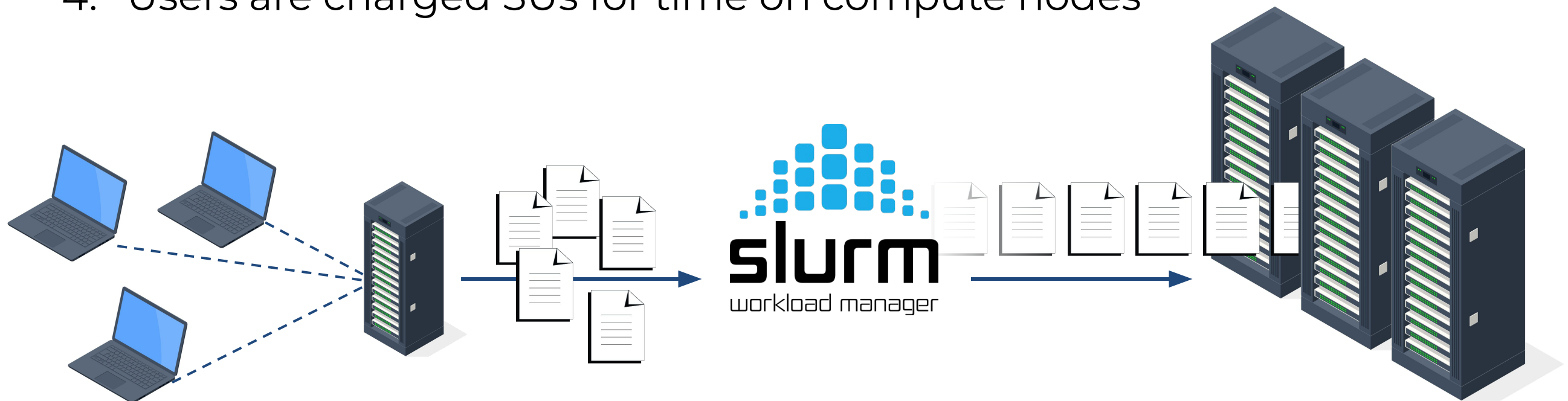
Cluster Computing

- Consumable Resources
- Slurm
- Job Management Tools

Batch Computing on HPRC Clusters: Overview

Since the cluster is a shared resource, jobs have to be moderated somehow so everyone has a turn. In short:

1. Write instructions on the login node
2. Give the instructions to Slurm
3. Slurm distributes everybody's instructions to compute nodes
4. Users are charged SUs for time on compute nodes



Consumable Computing Resources

Computing resources are shared!

You will not be able to use an infinite amount!

- Resources external to jobs themselves:
 - Service Unit (SU)
 - Software license/token
- Compute resources used by the job:
 - Processor cores
 - Memory
 - Wall time
 - GPU

Software Licenses

To check the availability of licensed software on the clusters, use the command **license_status**.

- List software the tool knows about:
- Help for this command:
- Find available license for "ansys":

```
license_status -l
```

```
license_status -h
```

```
license_status -s ansys
```

Exact availability varies by license policy.
Contact us if you have questions.

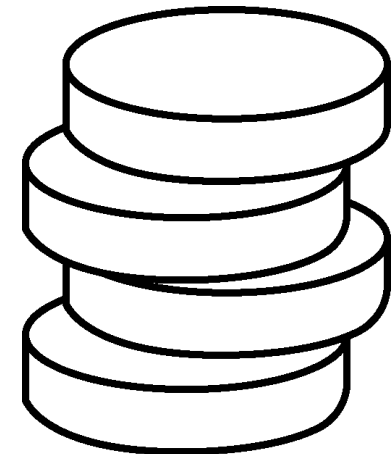
License status for ANSYS:

License Name	# Issued	# In Use	# Available
aa_mcad	50	0	50
aa_r	50	32	18
aim_mp1	50	0	50
.....			

https://hprc.tamu.edu/kb/Software/useful-tools/License_Checker

Service Units

- Service Units (SUs) are part of our account management system (AMS).
- With a default (Basic) allocation, you start with 20,000 SUs.
- You are charged 1 SU per core-hour on the **compute** nodes.
 - Work on login nodes is not charged (but will be limited)
 - GPUs and extra memory will cost more SUs
- You can check your SU balance on both:
 - The command line
 - The HPRC Portal



<https://hprc.tamu.edu/kb/User-Guides/AMS>

<https://hprc.tamu.edu/kb/User-Guides/AMS/#service-unit>

Check your (SU) Balance: Command Line

- List the SU Balance of your Account(s) with:

myproject

=====							
List of YourNetID's Project Accounts							

Account	FY	Default	Allocation	Used & Pending SUs	Balance	PI	

1228000223136	2024	N	10000.00	0.00	10000.00	Doe, John	

1428000243716	2024	Y	5000.00	-71.06	4928.94	Doe, Jane	

1258000247058	2024	N	5000.00	-0.91	4999.09	Doe, Jane	

- Run **myproject -d <Account#>** to change default project account.
(Replace <Account#> with your number!)
- Run **myproject -h** to see more options.

Check your (SU) Balance: Portal

SUs can also be checked in the Dashboard.

(The same place we checked file quotas previously)

The screenshot displays the GRACE DASHBOARD for Texas A&M University. The dashboard includes several sections: Project Information, Queue Availability, Virtual Env Management, Quota Information, User Groups, and Your Jobs. A yellow arrow points to the 'Balance' column in the Project Information table.

Project Information

Account	FY	Default	Used / Allocated (%)	Balance	PI
132748751559	2026	Set as Default	0.00 / 0.00 (0%)	0	Toback, David
132748758404	2026	☒ Y	1.61 / 20000.00 (0.01%)	19998.39	Liu, Honggao

Queue Availability

Queue	Core Usage (%)	Node Usage (%)
short*		
medium		

Virtual Env Management

Name	Python Version	GCCcore Version	Description	Action
testEnv		GCCcore/13.2.0		Delete

Quota Information

Disk	Disk Usage (%)	File Usage (%)	Action
/home/jwinchell	604M/10.0G 5.90%	5399/10000 53.99%	
/scratch/user/jwinchell	107.8G/1.0T 10.53%	122428/250000 48.97%	Request
/scratch/group/hprc	619.1G/10.0T	238472/1000000	Request

User Groups

[Group Management](#)

jwinchell staff hprc mitchcomp ansys

Need access to additional groups? Use the "Group Request" button above to submit a request.

Your Jobs

No active jobs.

Hands-on Exercise

1. Check your SUs in both the command line and the Portal.
-Verify that they match
2. Check that you have a default account set.

myproject

Project Information ⓘ					
Account ⓘ	FY ⓘ	Default ⓘ	Used / Allocated (%) ⓘ	Balance ⓘ	PI ⓘ
?	?	Set as Default	?	?	?
?	?	✓ Y	?	?	?

Cluster Computing

- Consumable Resources
- Slurm
- Job Management Tools

Example Job Script

```
#!/bin/bash

##NECESSARY JOB SPECIFICATIONS
#SBATCH --job-name=hello_world
#SBATCH --time=00:15:00
#SBATCH --ntasks=2
#SBATCH --ntasks-per-node=2
#SBATCH --nodes=1
#SBATCH --mem=3G
#SBATCH --output=hello_world_log.%j

# load required module(s)
module purge
module load GCCcore/13.2.0
module load Python/3.11.5
python hello_world.py

# Job Environment variables
echo $SLURM_JOBID
echo $SLURM_SUBMIT_DIR
echo $TMPDIR
echo $SCRATCH
```

These #SBATCH lines specify what resources you want to use, which in turn determine how many SUs you will be charged.

← (Start with module purge; best practice to start with a clean environment)

Whatever commands or scripts you want to run. Here, we set up the modules we need for our environment, run a python program, and print out some environment variables.

Example Job Script: Slurm Parameters

```
#!/bin/bash

##NECESSARY JOB SPECIFICATIONS
#SBATCH --job-name=hello_world
#SBATCH --time=00:15:00
#SBATCH --ntasks=2
#SBATCH --ntasks-per-node=2
#SBATCH --nodes=1
#SBATCH --mem=3G
#SBATCH --output=hello_world_log.%j

# load required module(s)
module purge
module load GCCcore/13.2.0
module load Python/3.11.5
python

# Job
echo
echo
echo
echo
```

--job-name	Arbitrary name we can use to track the job
--time	Maximum time limit for the script itself to run
--ntasks	Number of processes (minimum number of cores)
--ntasks-per-node	Minimum number of cores per node
--nodes	Number of nodes
--mem	Maximum memory (Might push you beyond the requested number of 'cores')
--output	File where STDOUT and STDERR will be saved

Generally the above information is **required**, but there are other slurm commands (and alternate ways to specify the above).

More job script and parameter info:

<https://hprc.tamu.edu/kb/User-Guides/Grace/Batch/#building-job-files>

Example Job Script: Variables

```
#!/bin/bash
```

<pre>##NBZ #SBA #SBA #SBA #SBA #SBA #SBA #SBA</pre> \$SLURM_JOBID	Unique number assigned by Slurm to track this job
<pre>#SBA</pre> \$SLURM_SUBMIT_DIR	Directory from which the job was submitted
<pre>#SBA #SBA #SBA #SBA #SBA #SBA #SBA</pre> \$TMPDIR	File space on the compute node itself: /work/job.\$SLURM_JOBID • Faster to read and write here than to read/write to your normal \$HOME or \$SCRATCH • However--this space is ephemeral. If you use it, be sure the job saves your end results in \$HOME or \$SCRATCH

```
module load Python/3.11.5
python hello_world.py
```

```
# Job Environment variables
echo $SLURM_JOBID
echo $SLURM_SUBMIT_DIR
echo $TMPDIR
echo $SCRATCH
```

Normal non-Slurm variables like \$SCRATCH are also available in jobs.

<https://hprc.tamu.edu/kb/User-Guides/Grace/Batch/#environment-variables>

Submit a Job and Check Job Status

Submit job

```
sbatch example01.job
```

```
Submitted batch job 6853258
(from job_submit) your job is charged as below
      Project Account: 122792016265
      Account Balance: 1687.066160
      Requested SUs:   3
```

```
./example01.job
```

This does not submit the job!
This runs it on the login node!
If the job is big, it might fail or
slow down the login node
for everyone!

Check status

```
squeue -u $USER
```

or

```
squeue --me
```

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	2024-01-23T17:27:3	None	c[180,202]
6853257	jobname	someuser	xlong	2	96	RUNNING	3-07:36:56	16:23:04	2024-01-23T17:27:2	None	c[523-524]

Job Submission and Tracking

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

<https://hprc.tamu.edu/kb/Helpful-Pages/Batch-Translation/#job-specifications>

Hands-on Exercise

Submit a simple job file:

1. Copy example directory to your \$SCRATCH space:

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

(can also do this in the Portal's file browser)

2. Submit this job file: `sbatch hello_world.slurm`

3. Check the job's status with: `squeue -u $USER` or `squeue --me`

4. Check the output file (will be named like `hello_world_log.#`).

Memory for Jobs 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.

Note: if you request 1 core, but all the memory on the node, you get charged for all the cores (since you're using their memory)

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

An 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. (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 <i>ask for more</i>	24	1	24
1	360	360	1	48
48	7.5	360	1	48

<https://hprc.tamu.edu/kb/User-Guides/AMS/#service-unit>

Slurm: SU Charges for GPU jobs

GPUs will use more SUs per hour than CPUs.

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

gpuinfo

GPU	COMPUTE	CUDA	GB MEM	ARCH	SUs per GPU
a100	8.0	11.1+	40	Ampere	72
a40	8.6	11.1+	48	Ampere	48
rtx	7.5	10.0+	24	Ada	48
t4	7.5	10.0+	16	Turing	24

Cluster Computing

- Consumable Resources
- Slurm
- Job Management Tools

Batch Queues

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

<https://hprc.tamu.edu/kb/User-Guides/Grace/Batch/#batch-queues>

Checking Resources for Jobs

`sinfo`

`pestat`

`maxconfig`

There are several tools available to check what you can use in your jobs...

Checking Resources for Jobs: Queues

sinfo

pestat

maxconfig

```
username@login1:~$ sinfo
```

PARTITION	AVAIL	TIMELIMIT	JOB	SIZE	NODES (A/I/O/T)	CPUS (A/I/O/T)
short*	up	2:00:00	1-32		711/0/89/800	33689/439/4272/38400
medium	up	1-00:00:00	1-128		711/0/89/800	33689/439/4272/38400
long	up	7-00:00:00	1-64		711/0/89/800	33689/439/4272/38400
xlong	up	21-00:00:00	1-32		711/0/89/800	33689/439/4272/38400
vnc	up	12:00:00	1-32		100/5/12/117	985/4055/576/5616
gpu	up	4-00:00:00	1-32		100/5/12/117	985/4055/576/5616
bigmem	up	2-00:00:00	1-4		7/0/1/8	560/0/80/640
staff	up	infinite	1-infinite		817/14/101/932	34689/5199/4848/4473
special	up	7-00:00:00	1-infinite		811/5/101/917	34674/4494/4848/4401
gpu-a40	up	4-00:00:00	1-15		6/9/0/15	15/705/0/720

Shows job queue status

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

Checking Resources for Jobs: Nodes

sinfo

pestat

maxconfig

```
username@login1:~$ pestat -p gpu -G
Print only nodes in partition gpu
GPU GRES (Generic Resource) is printed after each jobid
Hostname      Partition  Node Num_CPU  CPUload  Memsize  Freemem  GRES/node  Joblist
              Partition  State Use/Tot  (15min)  (MB)      (MB)      JobID(JobArrayID) User GRES/job ...
g001          gpu      mix  2  48    1.94    368640    271338    gpu:a100:2 <jobID> <username> <gpus>
g002          gpu      drain* 0  48    0.00*   368640      0    gpu:a100:2
g003          gpu      drain* 0  48    0.01    368640    370427    gpu:a100:2
.....
```

Shows status of nodes

Notable options:

- -p <partition name> show only a specific partition
- -u \$USER show only specified user's jobs
- -G show "generic resources" (e.g. the gpus used)

Checking Resources for Jobs: SUs

sinfo

pestat

maxconfig

If you ask for too many resources, your job will not run. You can check beforehand:

```
username@login1:~$ maxconfig
```

```
Grace partitions:  short medium long xlong vnc gpu bigmem special gpu-a40
```

```
Grace GPUs in gpu partition:  a100:2  a40:3  rtx:2  t4:4
```

```
Showing max parameters (cores, mem, time) for partition long
```

```
CPU-billing * hours * nodes =  SUs
          48 *   168 *     1 = 8,064
```

```
#!/bin/bash
```

```
#SBATCH --job-name=my_job
```

```
#SBATCH --time=7-00:00:00
```

```
#SBATCH --nodes=1
```

```
#SBATCH --ntasks-per-node=1
```

```
#SBATCH --cpus-per-task=48
```

```
#SBATCH --mem=360G
```

```
#SBATCH --output=stdout.%x.%j
```

```
#SBATCH --error=stderr.%x.%j
```

Check specific partitions with:

```
maxconfig -p <partitionName>
```

Estimate the SUs a job will require with:

```
maxconfig -f <jobScriptName>
```

Job Summary: myjob

Command-line tool to show you details of a specific job:

myjob

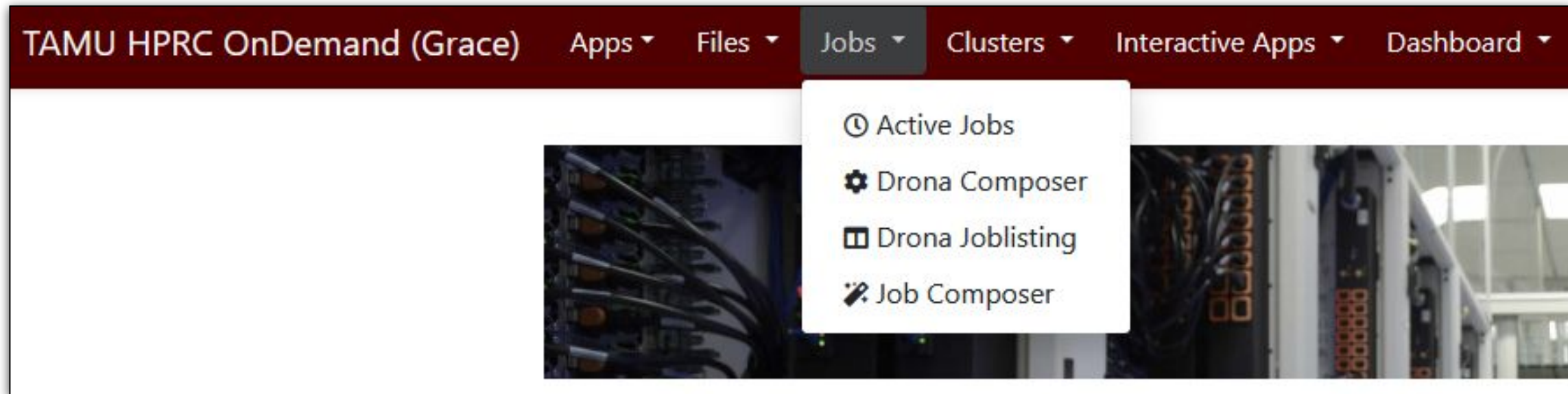
Shows, e.g.:

- Submission details
- Resource usage of finished jobs
- Status of pending job

```
[u.ab12345@aces-login2 ~]$ myjob 9814093
```

```
Job ID: 9814093
Cluster: grace
User/Group: userid/userid
Account: 123456789101
SUs charged: 283.82
State: COMPLETED (exit code 0)
Partition: long
Node Count: 1
NodeList: c121
Cores per node: 48
CPU Utilized: 01:07:56
CPU Efficiency: 0.42% of 11-08:28:00 core-walltime
Submit time: 2025-02-24 17:32:41
Start time: 2025-02-24 17:32:50
End time: 2025-02-24 23:13:25
Job Wall-clock time: 05:40:35
Memory Utilized: 40.95 GB
Memory Efficiency: 11.38% of 360.00 GB
Job Name: parafold
Job Submit Directory: /scratch/user/netid/myproject
Submit Line: sbatch run_parafold_grace.sh
```

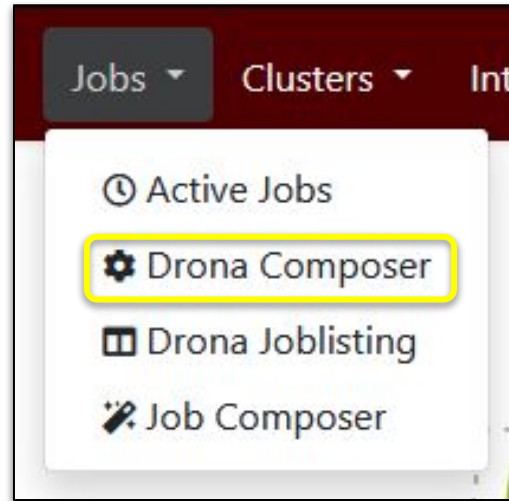
Portal Job Composers



Job management can also be done from the Portal.

- “Active Jobs” and “Job Composer” come with Open OnDemand
- The “Drona” options are developed by HPRC and have additional functionality

Drona Composer



Import additional environments

Select customizable environments or workflows

The main interface of the Drona Composer (GRACE) web application. It features a dark red header with the ATM logo and 'High Performance Research Computing DIVISION OF RESEARCH' on the left, and 'DRONA COMPOSER (GRACE)' on the right. The main content area is titled 'Job Composer' and contains three input fields: 'Job Name' (a text box), 'Location' (a text box with a 'Change' button next to it, containing the path '/scratch/user/jwinchell/drona_composer/runs'), and 'Environments' (a dropdown menu with the text '-- Choose an option --' and a '+' button to its right). A 'Preview' button is located below the 'Environments' field. At the bottom, a yellow warning icon is followed by the text: 'Cautions: Job files will overwrite existing files with the same name. The same principle applies for your executable scripts.'

Drona Composer

Generic environment to
create custom Slurm
batch job

Add modules for job

Fill out info that
Slurm would need

ATM High Performance Research Computing DIVISION OF RESEARCH

DRONA COMPOSER (GRACE)

Job Composer

Job Name

Location [Change](#) /scratch/user/jwinchell/drona_composer/runs

Environments [+](#)

Upload files [?](#) [Add](#)

Add modules [?](#) [+](#)

Number of tasks [?](#) [Advanced task options](#) [?](#) ☐

Use Accelerator [?](#)

TOTAL Memory [?](#) Expected run time [?](#) Project Account [?](#) Additional Slurm parameters [?](#)

[Preview](#)

Drona Composer

Job Preview

Files: **template.txt** driver.sh

```
1 #!/bin/bash
2 #SBATCH --job-name=testJob
3 #SBATCH --time=2:0:00 --mem=30G
4 #SBATCH --ntasks=1 --nodes=1 --cpus-per-task=1
5 #SBATCH --output=out.%j --error=error.%j
6
7
8
9 module purge
10 module load WebProxy foss/2024a Python/3.12.3
11 cd /scratch/user/jwinchell/drona_composer/runs/testJob
12 # ADD YOUR COMMANDS BELOW
13
14
```

Live Output **READY**

Live Output Stream

Job execution output will appear here in real-time once you submit your configuration.

Submit Job **Close**

Configure your job on the left, then submit to see live output on the right. Drag the center divider to resize panes.

Generated
Slurm
directives and
module loads

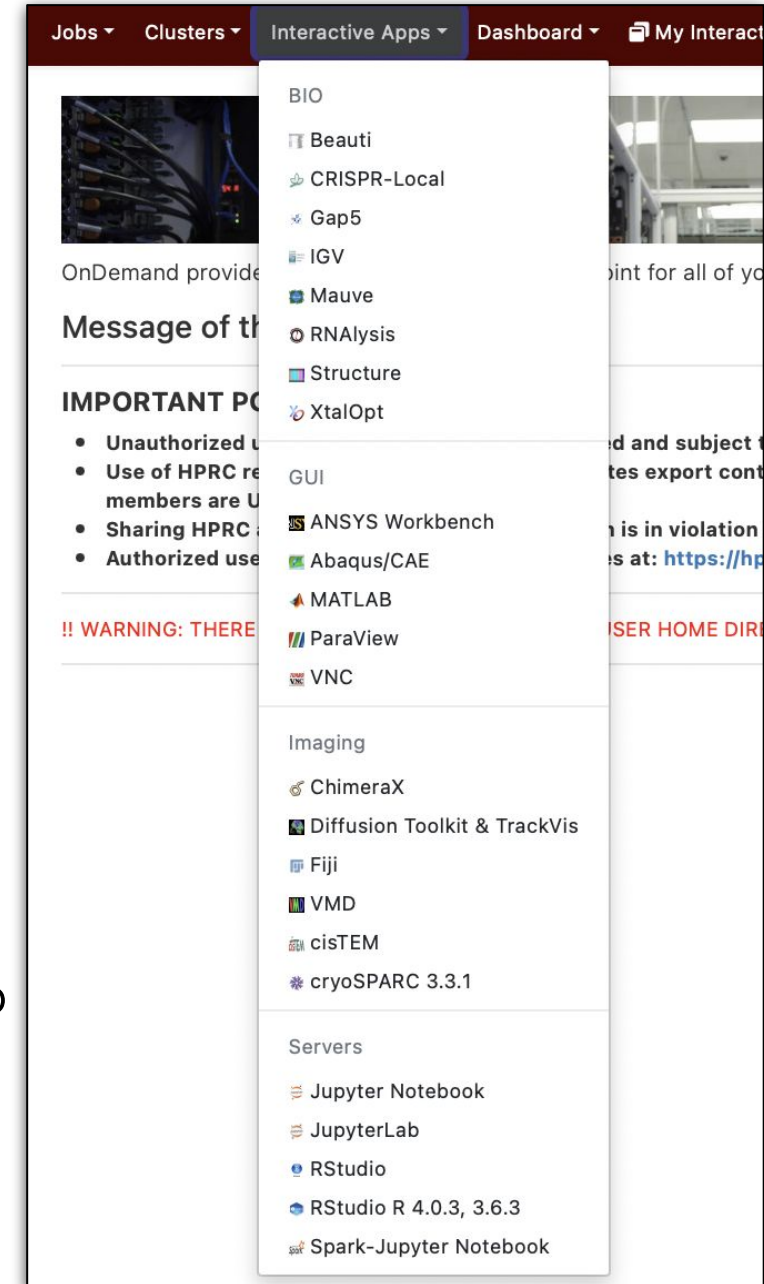
Add commands
directly into
editable window

Output of job
submission

Hit "Submit"

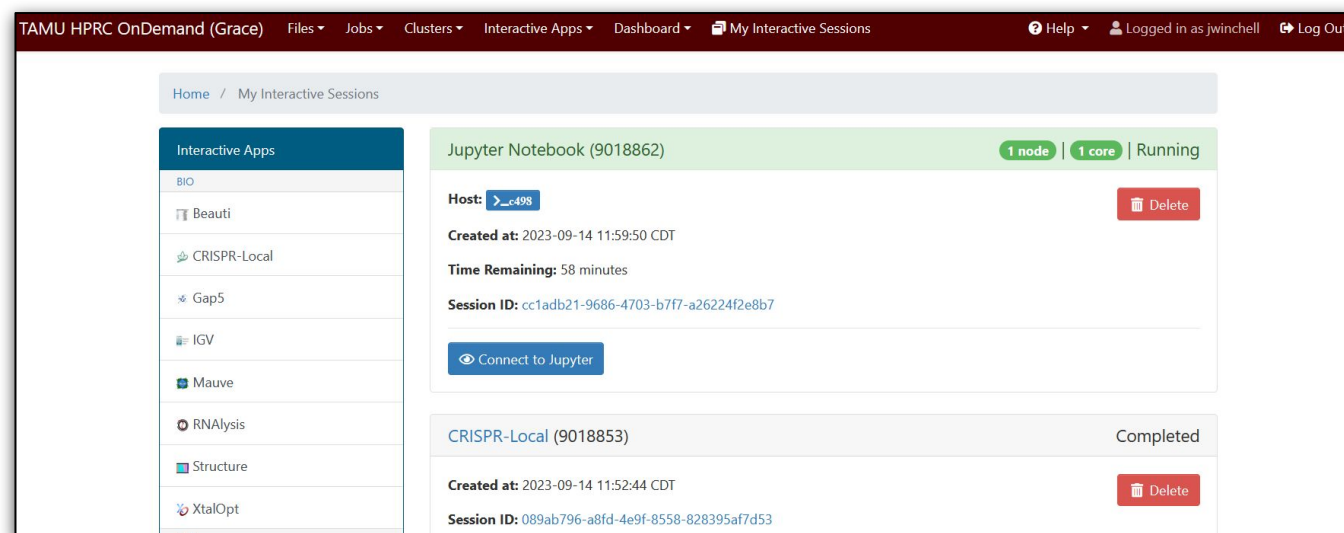
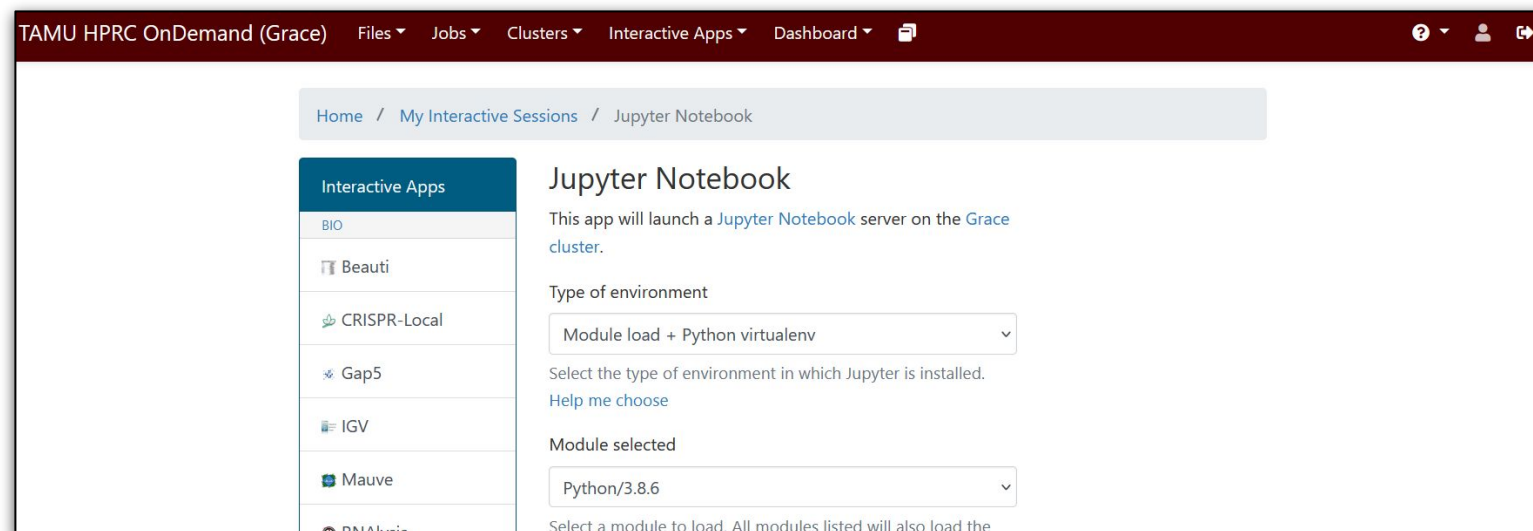
Other Type of Jobs

- Visualization:
 - portal.hprc.tamu.edu
 - Interactive Apps > choose 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
 - If a tamulauncher job gets killed, you can resubmit the same job to complete the unfinished commands in the input file.
 - <https://hprc.tamu.edu/kb/Software/tamulauncher>



Interactive Apps

1. Select an app from the dropdown menu.
2. Specify environment and resources.
3. Hit “Launch” at the bottom.
4. On the “My Interactive Sessions” screen:
 - a. wait for box to turn green
 - b. connect to app
 - c. quit from the app when you are done
 - d. box should turn gray



Need Help?

First check the FAQ <https://hprc.tamu.edu/kb/FAQ/Accounts>

- Grace User Guide <https://hprc.tamu.edu/kb/User-Guides/Grace>
- 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

Remember you can
start help requests
from the Dashboard!



High Performance
Research Computing
DIVISION OF RESEARCH

Thank you
Questions?

Give us feedback on the class with this survey:

https://u.tamu.edu/hprc_shortcourse_survey



Batch Job Examples

Slurm Job File (Serial Example)

```
#!/bin/bash
```

```
##NECESSARY JOB SPECIFICATIONS
```

```
#SBATCH --job-name=JobExample1
```

```
#SBATCH --time=01:30:00
```

```
#SBATCH --ntasks=1
```

```
#SBATCH --mem=6G
```

```
#SBATCH --output=Example1Out.%j
```

```
#Set the job name to "JobExample1"
```

```
#Set the wall clock limit to 1hr and 30min
```

```
#Request 1 task
```

```
#Request 6GB per node
```

```
#Send stdout/err to "Example1Out.[jobID]"
```

SUs = 1.5

```
##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
```

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
```

```
#SBATCH --time=1-12:00:00
```

```
#SBATCH --ntasks=8
```

```
#SBATCH --ntasks-per-node=2
```

```
#SBATCH --mem=5G
```

```
#SBATCH --output=Example3Out.%j
```

```
# Set the job name to "JobExample3"
```

```
# Set the wall clock limit to 1 Day and 12hr
```

```
# Request 8 tasks (cores)
```

```
# Request 2 tasks(cores) per node
```

```
# Request 5GB per node
```

```
# Send stdout and stderr to "stdout.[jobID]"
```

SUs = 288

```
##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
```

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=5G                         # Request 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 = 72

Slurm Job File (multi GPU)

```
#!/bin/bash
```

```
##NECESSARY JOB SPECIFICATIONS
```

```
#SBATCH --job-name=JobExample5
```

```
#SBATCH --time=01:00:00
```

```
#SBATCH --ntasks=2
```

```
#SBATCH --nodes=1
```

```
#SBATCH --mem=250G
```

```
#SBATCH --output=Example4Out.%j
```

```
#SBATCH --gres=gpu:a100:2
```

```
#SBATCH --partition=gpu
```

```
# Set the job name to "JobExample4"
```

```
# Set the wall clock limit to 1hr
```

```
# Request 1 task (core)
```

```
# Request 250GB per node
```

```
# Send stdout and stderr to "Example4Out.[jobID]"
```

```
# Request 2 A100 GPUs
```

```
# Request the GPU partition/queue
```

SUs = 179

```
##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
```

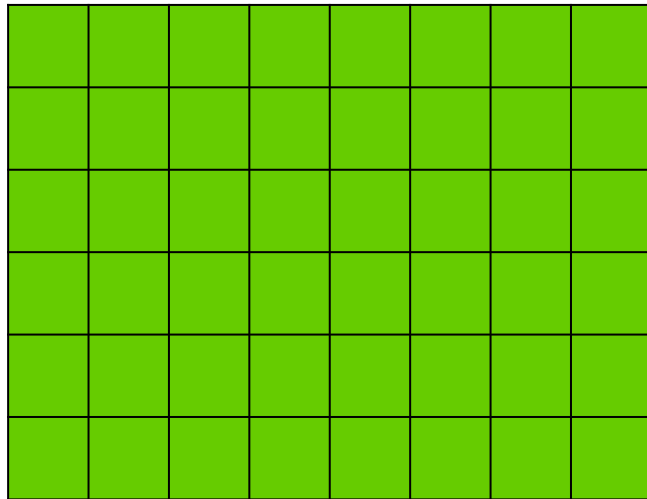
```
module load CUDA/11.8.0
```

```
# run your program
```

```
my_cuda_enabled_program
```

Mapping Jobs to Cores per Node on Grace

A.

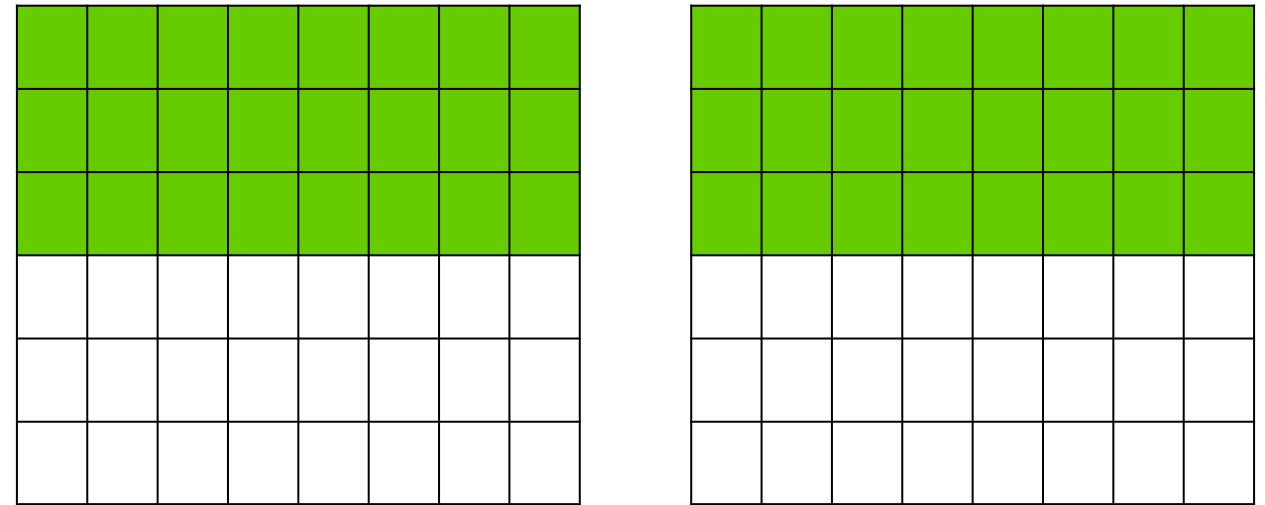


48 cores on
1 compute node

#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

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