

HIGH PERFORMANCE RESEARCH COMPUTING

Introduction to FASTER and ACES Clusters

Short Course
September 20, 2022



High Performance
Research Computing
DIVISION OF RESEARCH

Outline

- Usage Policies
- Resources
- FASTER Overview
- ACES Overview
- Break
- Accessing HPRC
- HPRC Computing Environment
- Break
- Cluster Computing Basics
- Cluster Computing Exercises
- Accessing ACES

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"](https://www.youtube.com/channel/UC...)
 - Answers to frequently asked questions can be found on our Wiki
https://hprc.tamu.edu/wiki/Main_Page
<https://access-ci.atlassian.net/wiki/spaces/ACCESSdocumentation/overview>

Follow Along

Cluster computing exercises:

- FASTER: Example files located in the directory:
`/scratch/training/Intro-to-Faster`

Interface for hands-on exercises:

- ssh
- HPRC Portal
- MobaXterm

FASTER Cluster

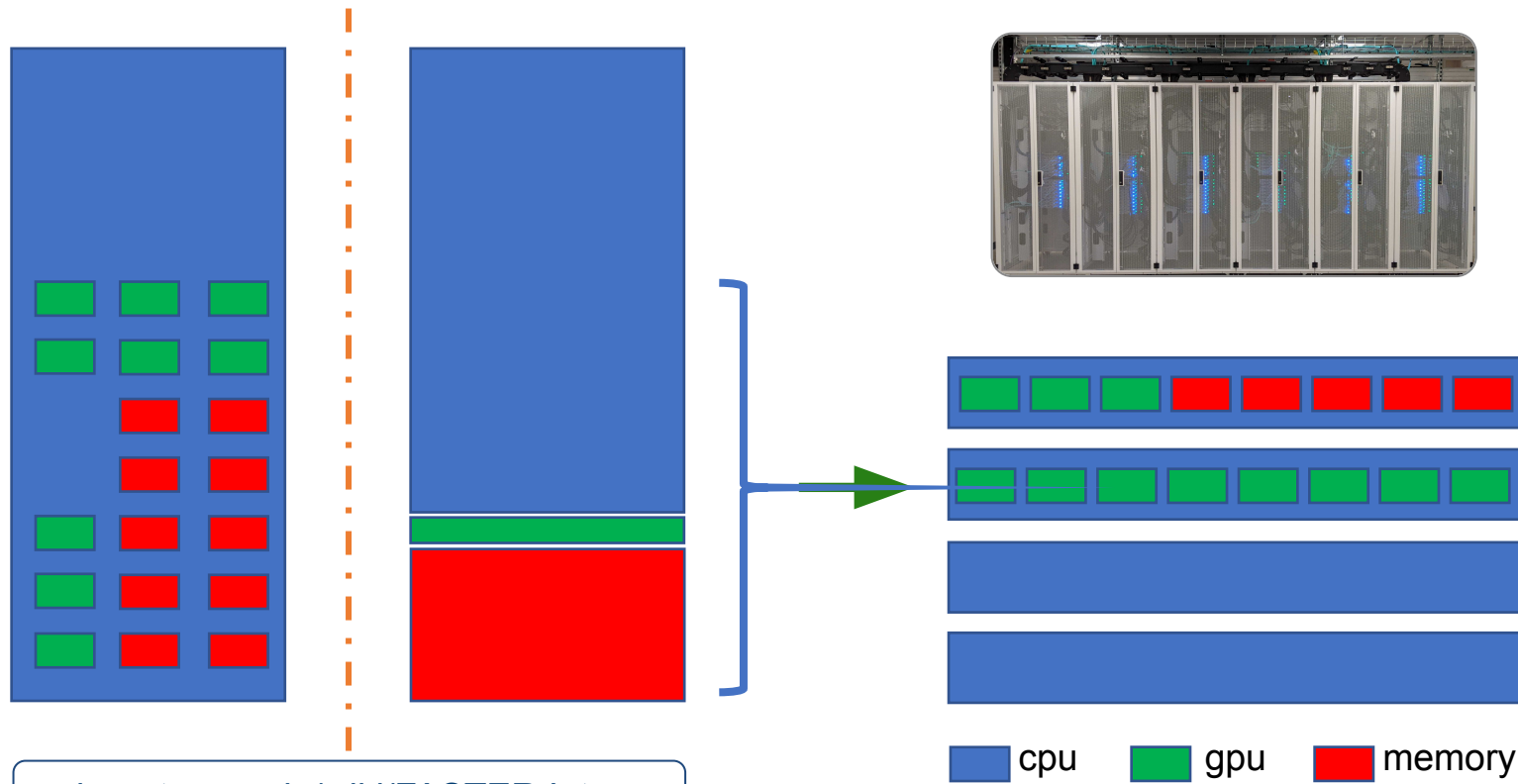
hprc.tamu.edu/wiki/FASTER:Intro

Node Type	Quantity
64-core login nodes	4 (3 for TAMU, 1 for ACCESS)
64-core compute nodes (256GB RAM each)	180 (11,520 cores)
Composable GPUs	200 T4 16GB 40 A100 40GB 10 A10 24GB 4 A30 24GB 8 A40 48GB
Interconnect	Mellanox HDR100 InfiniBand (MPI and storage) Liquid PCIe Gen4 (GPU composability)
Global Disk	5PB DDN Lustre appliances



FASTER (Fostering Accelerated Sciences Transformation Education and Research) is a 180-node Intel cluster from Dell with an InfiniBand HDR-100 interconnect and Liquid PCIe Gen4 for composing the GPUs. Nvidia A100, A10, A30, A40 and T4 GPUs are available. The 180 nodes are based on the Intel Ice Lake processor.

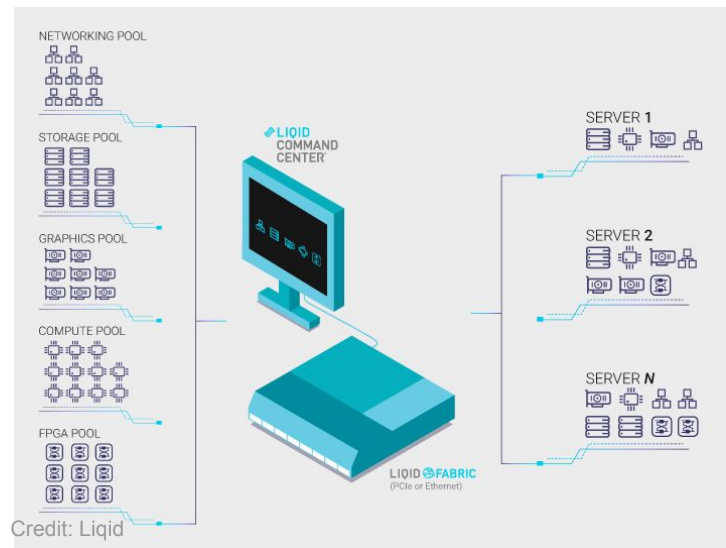
Composability at the Hardware Level



hprc.tamu.edu/wiki/FASTER:Intro

Composability on FASTER

- GPUs can be added to compute nodes by using the “gres” option in a Slurm script.
 - 200 T4 16GB GPUs
 - 40 A100 40GB GPUs
 - 10 A10 24GB GPUs
 - 4 A30 24GB GPUs
 - 8 A40 48GB GPUs



1,100+ Software Modules!

SOFTWARE MODULES ON THE FASTER CLUSTER

Last Updated: Jul 7 17:13:36 CDT

The available software for the **Faster cluster** is listed in the table. Click on any software package name to get more information such as the available versions, additional documentation if available, etc.

Show 10 entries Search: tensorflow

Name	Description
einops	'Flexible and powerful tensor operations for readable and reliable code. Supports numpy, pytorch, tensorflow, jax, and others.'
Horovod	'Horovod is a distributed training framework for TensorFlow.'
Keras	'Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow.'
ONNX-Runtime	'ONNX Runtime inference can enable faster customer experiences and lower costs, supporting models from deep learning frameworks such as PyTorch and TensorFlow/Keras as well as classical machine learning libraries such as scikit-learn, LightGBM, XGBoost, etc. ONNX Runtime is compatible with different hardware, drivers, and operating systems, and provides optimal performance by leveraging hardware accelerators where applicable alongside graph optimizations and transforms.'

hprc.tamu.edu/software/faster/

ACES - Accelerating Computing for Emerging Sciences



ACES
ACCELERATING COMPUTING
FOR EMERGING SCIENCES

HPC | **WIRE**

Since 1987 - Covering the Fastest Computers
in the World and the People Who Run Them

- Home
- Technologies
- Sectors
- COVID-19
- AI/ML/DL
- Exascale
- Specials
- Resource Library
- Podcast
- Events
- Job Bank
- About
- Our Authors
- Solution Channels
- Subscribe



September 23, 2021

As Moore's law slows, HPC developers are increasingly looking for speed gains in specialized code and specialized hardware – but this specialization, in turn, can make testing and deploying code trickier than ever. Now, researchers from Texas A&M University, the University of Illinois at Urbana-Champaign and the University of Texas at Austin have teamed, with NSF funding, to build a \$5 million prototype supercomputer ("ACES") with a dynamically configurable smörgåsbord of hardware, aiming to support developers as hardware needs grow ever more diverse.

ACES (short for "Accelerating Computing for Emerging Sciences") is presented as an "innovative composable hardware platform." ACES will leverage a PCIe-based composable framework from Liqid to offer access to Intel's high-bandwidth memory Sapphire Rapids processors and more than 20 accelerators: Intel FPGAs; NEC Vector Engines; NextSilicon co-processors; Graphcore IPUs (Intelligence Processing Units); and Intel's forthcoming Ponte Vecchio GPUs. All this hardware will be coupled with Intel Optane memory and DDN Lustre Storage and connected with Mellanox NDR 400Gbps networking.

"ACES will enable applications and workflows to dynamically integrate the different accelerators, memory, and in-network computing protocols to glean new insights by rapidly processing large volumes of data," the [NSF grant](#) reads, "and provide researchers with a unique platform to produce complex hybrid programming models that effectively supports calculations that were not feasible before."



<https://www.hpcwire.com/2021/09/23/three-universities-team-for-nsf-funded-aces-reconfigurable-supercomputer-prototype/>

ACES

Accelerating Computing for Emerging Sciences

Our Mission:

- Offer an accelerator testbed for numerical simulations and AI/ML workloads.
- Provide consulting, technical guidance, and training to researchers.
- Collaborate on computational and data-enabled research.



ACES - Accelerating Computing for Emerging Sciences (Phase I)



Component	Quantity	Description
Graphcore IPU	16	16 Colossus GC200 IPUs and dual AMD Rome CPU server on a 100 GbE RoCE fabric
Intel FPGA PAC D5005	2	FPGA SOC with Intel Stratix 10 SX FPGAs, 64 bit quad-core Arm Cortex-A53 processors, and 32GB DDR4
Intel Optane SSDs	8	3 TB of Intel Optane SSDs addressable as memory using MemVerge Memory Machine.

ACES Phase I components are available through [FASTER](#)

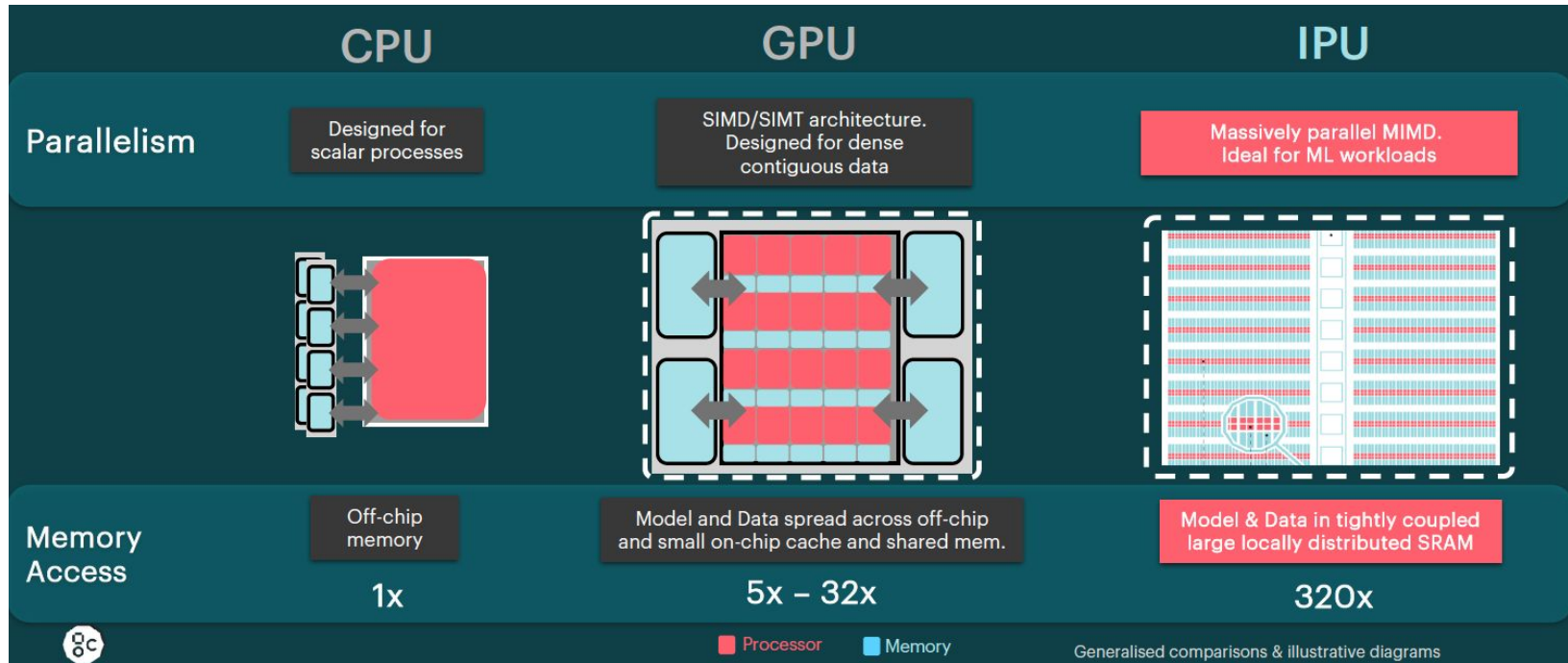
ACES - Accelerating Computing for Emerging Sciences (Phase II)



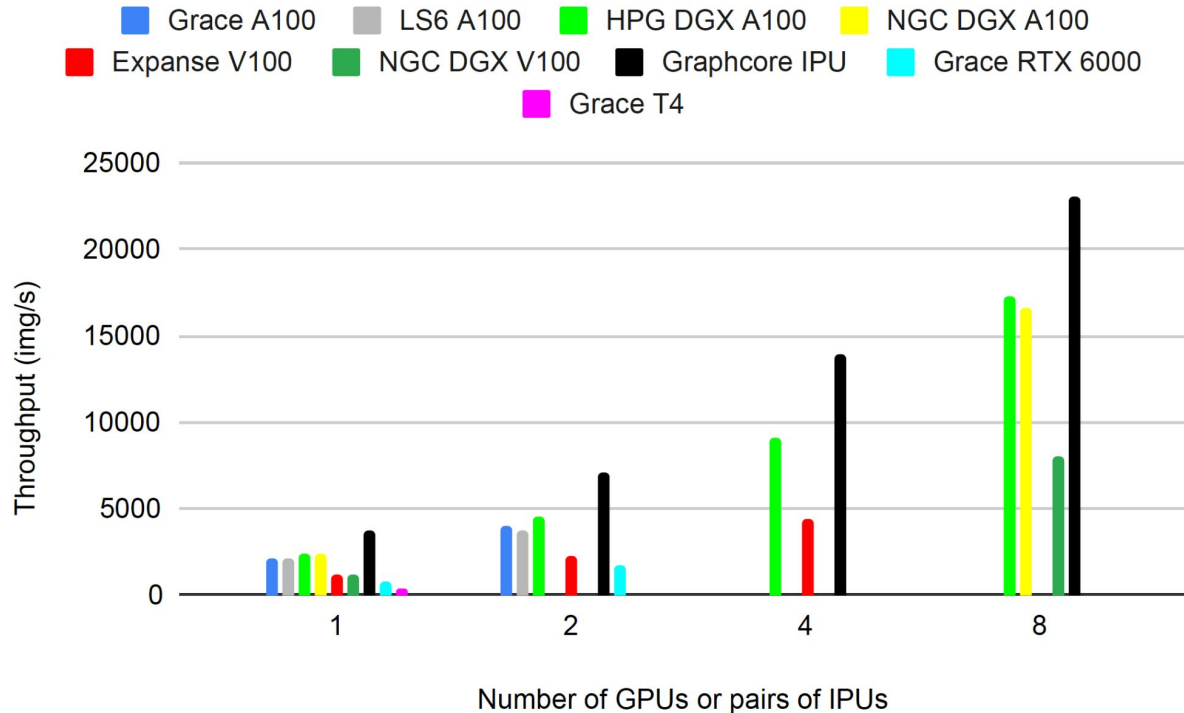
Component	Quantity*	Description
Graphcore IPU	32	16 Colossus GC200 IPU, 16 Bow IPU, and a dual AMD Rome CPU server on a 100 GbE RoCE fabric
Intel FPGA PAC D5005	2	FPGA SOC with Intel Stratix 10 SX FPGAs, 64 bit quad core Arm Cortex-A53 processors, and 32GB DDR4
Bittware IA-840F FPGA	2	Accelerator based on Intel Agilex FPGA
NextSilicon coprocessor	20	Reconfigurable accelerator with an optimizer continuously evaluating application behavior.
NEC Vector Engine	24	Vector computing card (8 cores and HBM2 memory)
Intel Ponte Vecchio GPU	100	Intel GPUs for HPC, DL Training, AI Inference
Intel Optane SSDs	48	18 TB of Intel Optane SSDs addressable as memory w/ MemVerge Memory Machine.

**Estimated quantities*

GraphCore IPU



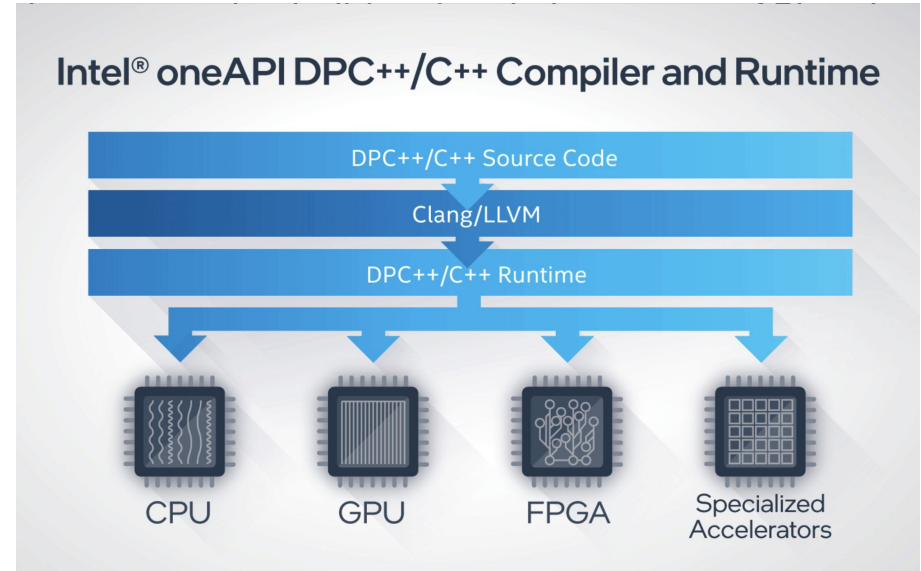
PyTorch ResNet50 - GPU vs IPU



Abhinand S. Nasari, Hieu T. Le, Richard Lawrence, Zhenhua He, Xin Yang, Mario M. Krell, Alex Tsyplikhin, Mahidhar Tatineni, Tim Cockerill, Lisa M. Perez, Dhruva K. Chakravorty and Honggao Liu. 2022. Benchmarking the Performance of Accelerators on National Cyberinfrastructure Resources for Artificial Intelligence / Machine Learning Workloads. In Practice and Experience in Advanced Research Computing (PEARC '22), July 10-14, 2022, Boston, MA, USA. ACM, New York, NY, USA, 13 Pages. <https://doi.org/10.1145/3491418.3530772>

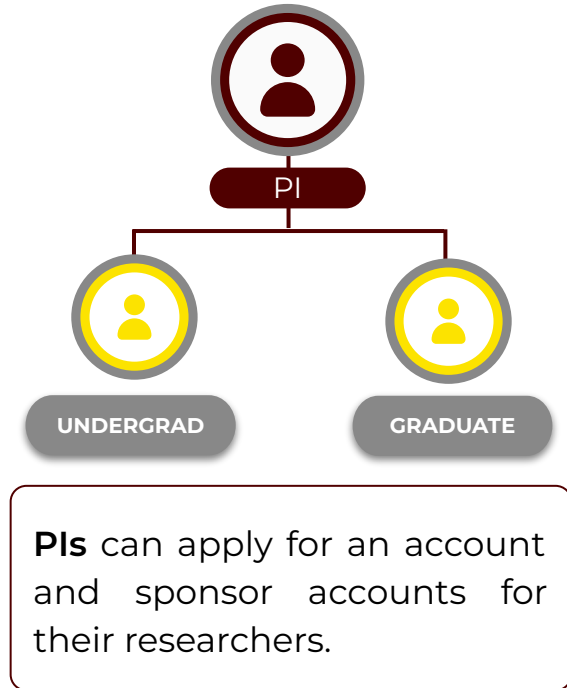
Intel FPGA PAC D5005

- FPGA is an integrated circuit (IC) where a large majority of the electrical functionality inside the device can be changed after its manufacture.
- Fast code prototyping - OneAPI DPC++
 - Compile on CPU, validate design, then compile via FPGA
- Support for legacy OpenCL workflow with Intel OpenCL FPGA SDK



Getting on ACES Phase I

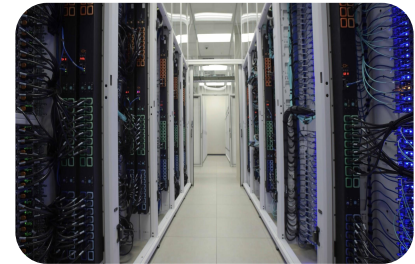
- Allocation is upon special request during this phase of deployment.
- You must have an [ACCESS](#) account!
- Applications are available at hprc.tamu.edu/aces/
- Email us at help@hprc.tamu.edu for questions, comments, and concerns.



Clusters Are For You!

What kinds of problems are solved by cluster computing?

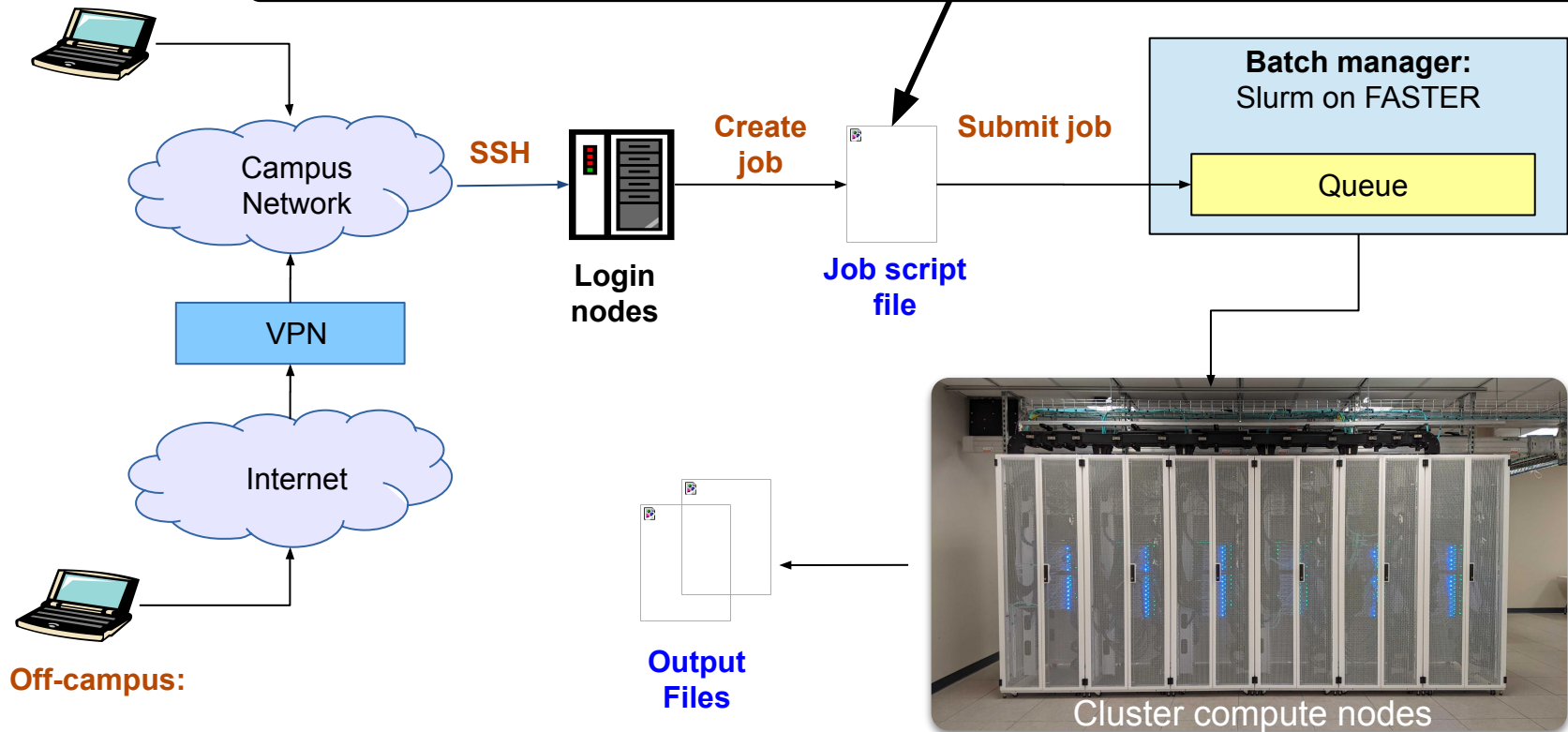
- Problems that are too big to fit in one laptop or workstation, due to limitation on memory, gpu count, core count, or node count
- Problems that scale well with more CPU cores or memory
- Single-threaded problems with millions of permutations
- Problems that require large high speed storage and/or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



Batch Computing on HPRC Clusters

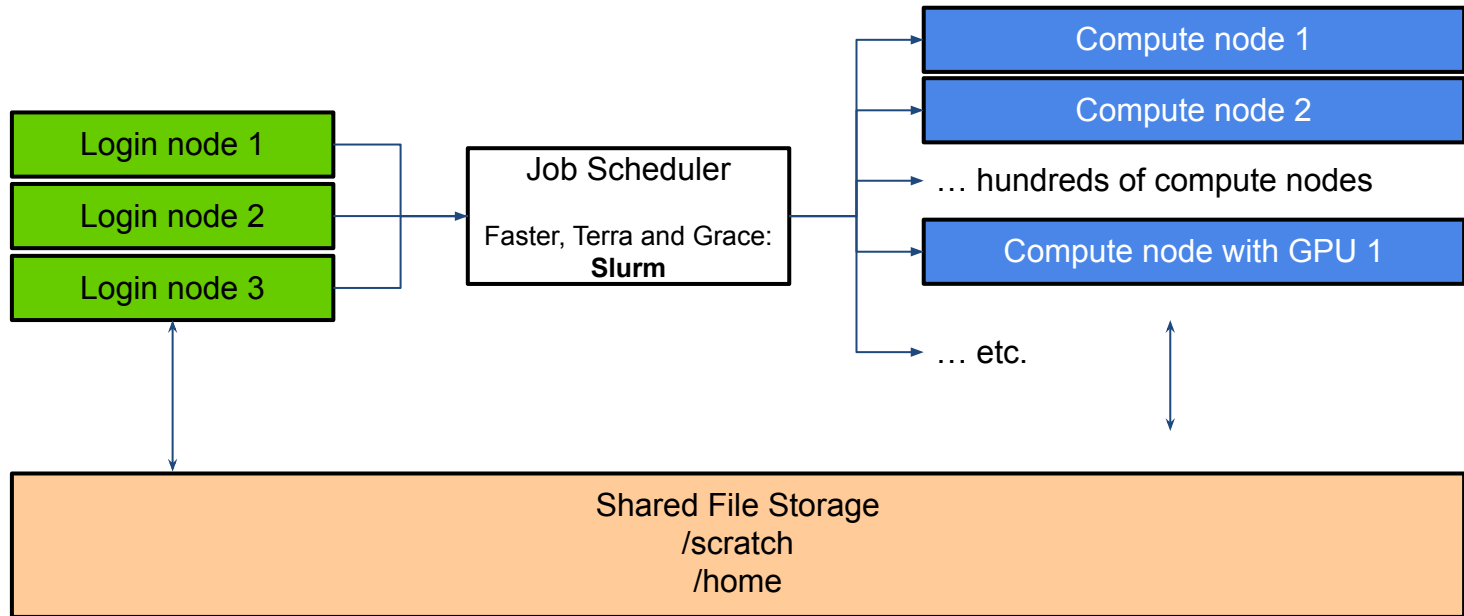
On-campus:

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



Off-campus:

Cluster Diagram



HPRC Wiki - Hardware

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


Information about FASTER hardware is on page <https://hprc.tamu.edu/wiki/FASTER>

Log in

Search TAMU HPRC

High Performance Research Computing

A Resource for Research and Discovery



Welcome to the TAMU HPRC Wiki


- [FASTER Guide](#)
- [Terra Guide](#)
- [Usage Policies](#)
- [Grace Guide](#)
- [Software](#)
- [Contact Us](#)

Announcements

- **Grace Cluster Status:** Cluster deployed, with service unit accounting.

Getting an Account

- **Understanding HPRC:** For a brief overview of what services HPRC offers, see [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 see the [Policies Page](#) to better understand the rules and etiquette of cluster usage..
- **Accessing the clusters:** All computer systems managed by the HPRC are available for use to TAMU faculty, staff, and students who require large-scale computing capabilities. The HPRC hosts the [Terra](#), and [Grace](#) clusters at TAMU. To apply for or renew an HPRC account, please visit the [Account Applications](#) page. For information on how to obtain an allocation to run jobs on one of our clusters, please visit the [Allocations Policy](#) page. *All accounts expire and must be renewed in September of each year.*



- [HPRC Home Page](#)
- [Wiki Home Page](#)
- [Policies](#)
- [New User Info](#)
- [Contact Us](#)
- [User Guides](#)
 - [ACES Phase I](#)
 - [FASTER](#)
 - [Grace](#)
 - [Terra](#)
 - [OOD Portal](#)
 - [Galaxy](#)
- [Helpful Pages](#)
 - [AMS Documentation](#)

High Performance Research Computing

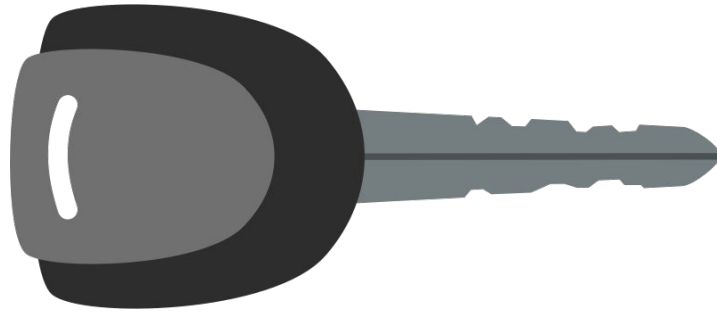
A Resource for Research and Discovery

FASTER User's Guide

- [Quick Start Guide](#)
- [Key Policies](#)
- [Hardware Summary](#)
- [Access](#)
- [Computing Environment](#)
- [File Systems, Quotas and File Transfers](#)
- [Compiling and Running Programs](#)
- [Batch Job Translation Guide](#)
- [Batch Processing \(Slurm\)](#)
 - [Introduction](#)

BREAK

Getting Started



Accessing FASTER via SSH

- SSH command is required for accessing FASTER:
 - On campus: `ssh userNetID@faster.hprc.tamu.edu`
 - Off campus:
 - Set up and start VPN (Virtual Private Network): u.tamu.edu/VPnetwork
 - Then: `ssh userNetID@faster.hprc.tamu.edu`
 - *Two-Factor Authentication* enabled for CAS, VPN, SSH
- SSH programs for Windows:
 - MobaXTerm (preferred, includes SSH and X11)
 - PuTTY SSH
 - Windows Subsystem for Linux
- <https://portal-faster.hprc.tamu.edu/>
 - Select the “Clusters” tab and then “_faster Shell Access”
- FASTER has 2 login nodes for TAMU users. Check the bash prompt.
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 **FASTER** for ACCESS users

- The Advanced Cyberinfrastructure Coordination Ecosystem: Services and Support ([ACCESS](#)) is a virtual collaboration funded by the National Science Foundation that facilitates free, customized access to advanced digital resources, consulting, training, and mentorship.
- View the [getting started documentation](#) to create an ACCESS account.
- FASTER has 1 login node for ACCESS users
- SSH via Jump Host:
 - `ssh -J fasterusername@faster-jump.hprc.tamu.edu:8822 fasterusername@login.faster.hprc.tamu.edu`

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
 - SSH/SFTP to HPRC clusters
- 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
- Not all software supports SSH+Duo: SFTP in MATLAB

Example: SSH login with Duo

```
$ ssh userNetID@faster.hprc.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

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

Hands-On Activity - 2 Minutes

1. Please try to login to FASTER now.
2. What message do you see when you log on?

Pop Quiz



How many T4 GPUs are available on FASTER?

- A. 10
- B. 20
- C. 50
- D. 200

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` directory is currently shared between Grace and FASTER clusters.
- View usage and quota limits using the command: `showquota`
- Request a group directory for sharing files.
- **Do not share your home or scratch directories.**

hprc.tamu.edu/wiki/FASTER:Filesystems_and_Files

Hands-on exercise:

Upload a File to FASTER using the Portal

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

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

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?

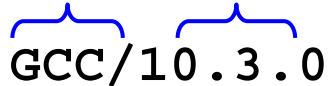
hprc.tamu.edu/wiki/SW:Modules

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:

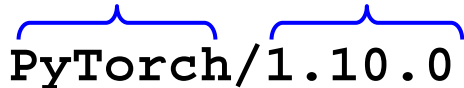
toolchain-name / version


GCC/10.3.0

toolchain-name / version


OpenMPI/4.1.1

software-name / version


PyTorch/1.10.0

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

hprc.tamu.edu/wiki/SW:Modules

Software: Application Modules

- Installed applications are made available with the module system
FASTER 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 list
```

```
# shows which software is available
```

```
module load GCCcore/11.3.0
```

```
# load GCC compiler version 11.3.0
```

```
module list
```

```
# show which software is now available
```

Software: Modules and Toolchains

- Toolchains are what we call groups of compilers & libraries
- There's a variety of toolchains on the clusters:
 - intel/2022a
 - iomkl/2021a
 - foss/2022a
 - GCCcore/11.3.0

```
module purge
```

```
# removes/unloads all loaded modules
```

Software: Application Modules

- Finding software

```
module spider keyword
```

search for a specific piece of software

```
module -t avail
```

show software available for the current toolchain

Hands on Activity - Module Loading Exercise

1. `ml list` # list all loaded modules
2. `module spider blast+` # see which versions of BLAST+ are available
3. `ml GCC/11.2.0 OpenMPI/4.1.1` # load the listed dependencies
4. `ml BLAST+/2.12.0` # load the version of BLAST+ available
5. `ml list` # list all loaded modules
6. `ml OpenMPI/4.1.2` # change version of a loaded module
notice the message about reloaded modules
7. `ml list` # list all loaded modules
notice the list on inactive modules
8. `ml purge` # unload 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	gomp	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`

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 important to unload unused modules before loading new modules (ml purge).
- It is recommended to load a specific software version instead of the defaults
- **Do NOT load modules in your ~/.bashrc file**
- Avoid mixing toolchains when loading multiple modules at the same time. This usually leads to one of them not working. The hierarchical module system helps prevent accidentally mixing toolchains.

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](#):

hprc.tamu.edu/wiki/SW:Python#Create_a_virtual_environment

Software Install Exercise

- ```
cd $SCRATCH
mkdir python_example
cd python_example
```

 # setup workspace
- ```
module purge  
ml GCCcore/11.3.0  
ml Python/3.10.4
```

 # load Python module
- ```
virtualenv my_example_venv
source my_example_venv/bin/activate
```

 # setup your virtual environment
- ```
python -c "import pytime"
```

 # check if python-time is installed (it's not)
- ```
pip install python-time
```

 # install a python package
- ```
python -c "import pytime"
```

 # check if python-time is installed (it is)
- ```
deactivate
```

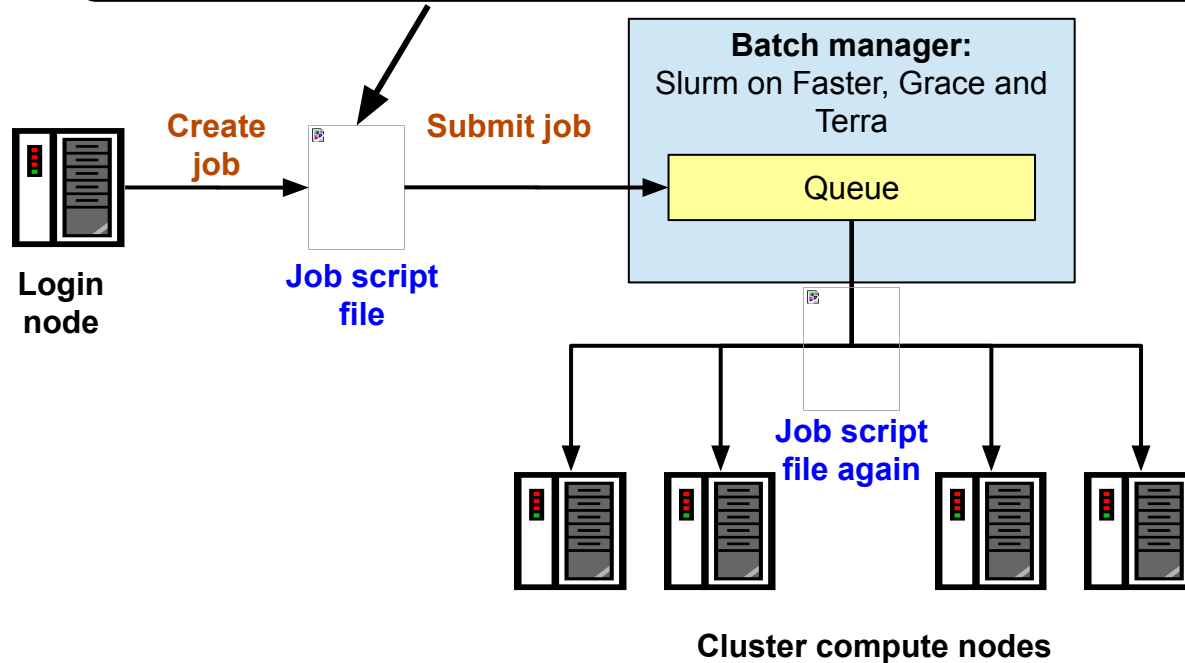
 # all done

BREAK

# 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)

- Use "myproject" to query

[hprc.tamu.edu/wiki/HPRC:AMS:Service\\_Unit](http://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 |

- Other resources:

- Software license/token

- Use "license\_status" to query
- [hprc.tamu.edu/wiki/SW:License\\_Checker](http://hprc.tamu.edu/wiki/SW:License_Checker)

```
license_status -a
```

Find available license for "ansys":

```
license_status -s Matlab
```

```
License status for Matlab:
```

```

License Name	# Issued	# In Use	# Available
Matlab	50	0	50
-----	-----	-----	-----
```

Find detail options:

```
license status -h
```

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

A **Service Unit (SU)** on **FASTER** is equivalent to one core or 3 GB memory usage for one hour + SUs for the requested GPUs.

| Number of Cores | GB of memory per core | Total Memory (GB) | Hours | SUs charged |
|-----------------|-----------------------|-------------------|-------|-------------|
| 1               | 3                     | 3                 | 1     | 1           |
| 1               | 4                     | 4                 | 1     | 2           |
| 1               | 250                   | 250               | 1     | 64          |
| 64              | 250                   | 250               | 1     | 64          |

On FASTER each T4 GPU would be an additional 64 SUs for one hour and each A100, A10, A30, or A40 would be an addition 128 SUs for one hour.

[hprc.tamu.edu/wiki/HPRC:AMS:Service\\_Unit](https://hprc.tamu.edu/wiki/HPRC:AMS:Service_Unit)

# 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 | 2022 | N       | 10000.00   | 0.00               | 10000.00 | Doe, John |
| 1428000243716 | 2022 | Y       | 5000.00    | -71.06             | 4928.94  | Doe, Jane |

- Run `"myproject -d Account#"` to change default project account
- Run `"myproject -h"` to see more options

[hprc.tamu.edu/wiki/HPRC:AMS:Service\\_Unit](https://hprc.tamu.edu/wiki/HPRC:AMS:Service_Unit)

[hprc.tamu.edu/wiki/HPRC:AMS:UI](https://hprc.tamu.edu/wiki/HPRC:AMS:UI)

## Hands-On Activity - 2 Minutes

1. Use myproject to check the SU balance of your accounts.
2. Check the license status of Schrodinger. How many licenses are in the first row of the # Issued column?



# 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/FASTER:Batch](https://hprc.tamu.edu/wiki/FASTER:Batch)

# sinfo : Current Queues on **FASTER**

| PARTITION | AVAIL | TIMELIMIT  | JOB_SIZE | NODES(A/I/O/T) | CPUS(A/I/O/T)     |
|-----------|-------|------------|----------|----------------|-------------------|
| faster    | down  | infinite   | 1-16     | 0/40/0/40      | 0/2560/0/2560     |
| faster-10 | down  | infinite   | 1-16     | 0/40/0/40      | 0/2560/0/2560     |
| cpu*      | up    | 7-00:00:00 | 1-32     | 4/73/13/90     | 256/4672/832/5760 |
| gpu       | up    | 7-00:00:00 | 1-32     | 6/24/8/38      | 324/1596/512/2432 |
| memverge  | up    | 2-00:00:00 | 1-2      | 0/0/2/2        | 0/0/128/128       |
| fpga      | up    | 2-00:00:00 | 1-2      | 0/2/0/2        | 0/128/0/128       |

**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**

# pestat - Processor Status

- **pestat** allows you to check the status of the nodes on FASTER
- Type **pestat -G** to show the status of all nodes
- -p allows you to show a specific partition
- Example:
  - **pestat -G -p gpu** shows the gpu node status

[GPU GRES (Generic Resource) is printed after each jobid

Print only nodes in partition gpu

| Hostname | Partition | Node State | Num_CPU Use/Tot | CPUload (15min) | Memsize (MB) | Freemem (MB) | GRES/node   |
|----------|-----------|------------|-----------------|-----------------|--------------|--------------|-------------|
| fc001    | gpu       | down*      | 0 64            | 31.40*          | 1030000      | 803818       | gpu:a100:8  |
| fc002    | gpu       | alloc      | 64 64           | 3.69*           | 256000       | 233455       | gpu:a100:4  |
| fc009    | gpu       | idle       | 0 64            | 0.00            | 256000       | 253607       | gpu:t4:4    |
| fc010    | gpu       | idle       | 0 64            | 0.00            | 256000       | 253675       | gpu:t4:4    |
| fc011    | gpu       | drain*     | 0 64            | 0.00            | 256000       | 254151       | gpu:t4:4    |
| fc012    | gpu       | idle       | 0 64            | 0.00            | 256000       | 253897       | gpu:t4:8    |
| fc013    | gpu       | idle       | 0 64            | 0.00            | 256000       | 253862       | gpu:t4:8    |
| fc024    | gpu       | alloc      | 64 64           | 3.13*           | 1030000      | 1006063      | gpu:a100:12 |
| fc025    | gpu       | drng*      | 64 64           | 2.00*           | 256000       | 80779        | gpu:a100:4  |
| fc026    | gpu       | alloc      | 64 64           | 3.30*           | 256000       | 233584       | gpu:a100:4  |

## Hands-On Activity - 2 Minutes

1. Use `sinfo` to see partition information.
2. Use `pestat` to check the status of GPU and CPU nodes.

# Batch Job Scripts

# Sample Job Script Structure (**FASTER**)

```
#!/bin/bash
##NECESSARY JOB SPECIFICATIONS
#SBATCH --job-name=JobExample1
#SBATCH --time=01:30:00
#SBATCH --ntasks=1
#SBATCH --mem=3G
#SBATCH --output=stdout.%j
```

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

```
load required module(s)
module purge
module load GCC/12.1.0
```

```
./my_program.py
```

These parameters describe your job to the job scheduler

Account number to be charged

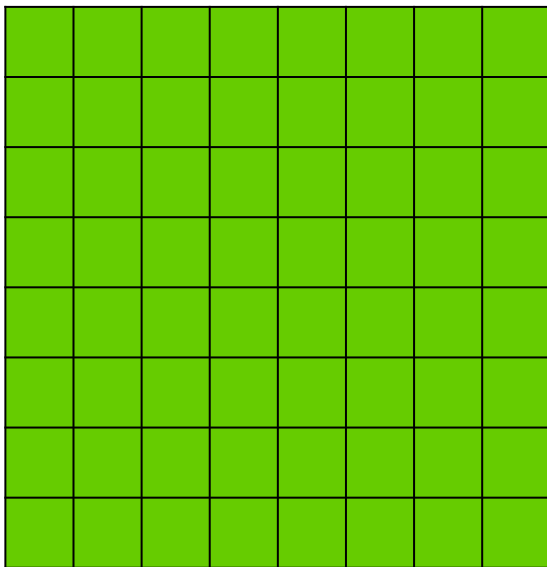
This is single line comment and not run as part of the script

Load the required module(s) first

This is a command that is executed by the job

# Mapping Jobs to Cores per Node on **FASTER**

A.

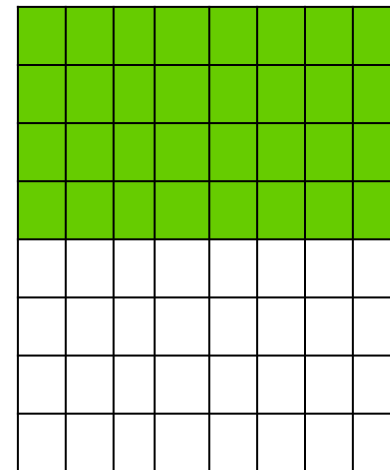
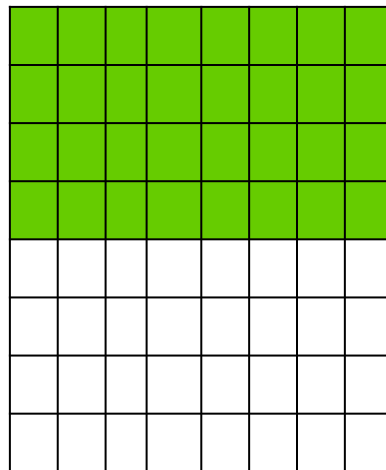


64 cores on  
1 compute node

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

Preferred Mapping  
(if applicable)

B.



64 cores on  
2 compute nodes

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

# Important Batch Job Parameters

| <b>FASTER</b>                                                                                                 | <b>Comment</b>                                                                 |
|---------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------|
| <code>#SBATCH --time=HH:MM:SS</code>                                                                          | Specifies the time limit for the job.                                          |
| <code>#SBATCH --ntasks=NNN</code>                                                                             | Total number of tasks (cores) for the job.                                     |
| <code>#SBATCH --ntasks-per-node=XX</code>                                                                     | Specifies the maximum number of tasks (cores) to allocate per node             |
| <code>#SBATCH --mem=nnnnM</code><br>or<br><code>#SBATCH --mem=nG</code><br><br><code>(memory per NODE)</code> | Sets the maximum amount of memory (MB).<br><br>G for GB is supported on FASTER |



# Job Memory Requests on **FASTER**

- Specify memory request based on memory per node:

**#SBATCH --mem=xxxxM**  
or

**# memory per node in MB**

**#SBATCH --mem=xG**

**# memory per node in GB**

- On 256GB nodes, usable memory is at most 250 GB.  
The per-process memory limit should not exceed ~3900 MB for a 64-core job.

# Pop Quiz

```
#SBATCH --job-name=stacks_S2
#SBATCH --ntasks=128
#SBATCH --ntasks-per-node=32
#SBATCH --mem=40G
#SBATCH --time=48:00:00
```

How many nodes is this job requesting?

- A. 1600
- B. 80
- C. 20
- D. 4

# FASTER Job File (Serial Example)

```
#!/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
```

```
load required module(s)
```

```
module purge
```

```
module load intel/2022a
```

```
run your program
```

```
./myprogram
```

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

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

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

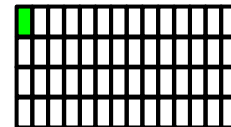
```
Request 2GB per node
```

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

```
Set billing account to 123456
```

```
Send email on all job events
```

```
Send all emails to email_address
```



SUs = 1.5

# FASTER 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=32 # Request 14 tasks (cores) per node
#SBATCH --mem=64G # Request 64GB per node
#SBATCH --output=stdout.%j # Send stdout to "stdout.[jobID]"
#SBATCH --error=stderr.%j # Send stderr to "stderr.[jobID]"

##OPTIONAL JOB SPECIFICATIONS
#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
./my_multicore_program
```

SUs =  
32

# FASTSER Job File (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=10 # Request 10 task (core)
#SBATCH --mem=250G # Request 250GB per node
#SBATCH --output=stdout.%j # Send stdout and stderr to "stdout.[jobID]"
#SBATCH --gres=gpu:a100:10 # Request a node with 10 A100 GPU's
#SBATCH --partition=gpu # Request the GPU partition/queue
```

SUs = 1344

```
##OPTIONAL JOB SPECIFICATIONS
```

```
#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/2022a
module load CUDA/11.7.0
run your program
./my_gpu_program
```

# Submitting Your Job and Check Job Status

Submit job

```
sbatch example01.job
```

```
161997
```

Check status

```
squeue -u $USER
```

| JOBID | NAME     | USER     | PARTITION | NODES | CPUS | STATE   | TIME    | TIME_LEFT  | START_TIME         | REASON | NODELIST    |
|-------|----------|----------|-----------|-------|------|---------|---------|------------|--------------------|--------|-------------|
| 31745 | AP_PVP_8 | someuser | cpu       | 2     | 128  | RUNNING | 9:49:37 | 1-16:10:23 | 2022-09-19T16:20:4 | None   | fc[014-015] |
| 31744 | AP_PVP_8 | someuser | cpu       | 2     | 128  | RUNNING | 9:50:48 | 1-16:09:12 | 2022-09-19T16:19:3 | None   | fc[007-008] |

# Hands-on exercises:

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

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

Inspect the contents.

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

# Job Exercise: Check Output

Submit job

```
cd batch_examples
```

```
FASTER: sbatch example01.job
```

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

Check status

```
FASTER: squeue -u $USER
```

| JOBID | NAME    | USER     | PARTITION | NODES | CPUS | STATE   | TIME | TIME_LEFT | START_TIME         | REASON    | NODELIST |
|-------|---------|----------|-----------|-------|------|---------|------|-----------|--------------------|-----------|----------|
| 64039 | somejob | someuser | medium    | 1     | 1    | PENDING | 0:00 | 2:00      | 2022-01-30T21:00:4 | Resources |          |
| 64038 | somejob | someuser | medium    | 4     | 256  | RUNNING | 2:49 | 17:11     | 2022-01-30T20:40:4 | None      |          |

txt=[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

```
FASTER: #SBATCH -o output.ex01.env_variables.%j
```



# Job Exercise: Check Status

Submit job

```
cd batch_examples
```

**FASTER:** `sbatch example02.job`

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

Check status

**FASTER:** `squeue -u $USER`

| JOBID | NAME    | USER     | PARTITION | NODES | CPUS | STATE   | TIME | TIME_LEFT | START_TIME         | REASON    | NODELIST |
|-------|---------|----------|-----------|-------|------|---------|------|-----------|--------------------|-----------|----------|
| 64039 | somejob | someuser | medium    | 1     | 1    | PENDING | 0:00 | 2:00      | 2022-01-30T21:00:4 | Resources |          |
| 64038 | somejob | someuser | medium    | 4     | 112  | RUNNING | 2:49 | 17:11     | 2022-01-30T20:40:4 | None      |          |

txt= [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

```
FASTER: #SBATCH -o output.ex02.echo_numbers.%j
```

# Job Exercise: Debug job failures

Submit job

```
cd batch_examples
```

**FASTER:** `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

```
FASTER: #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
```

**FASTER:** `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?

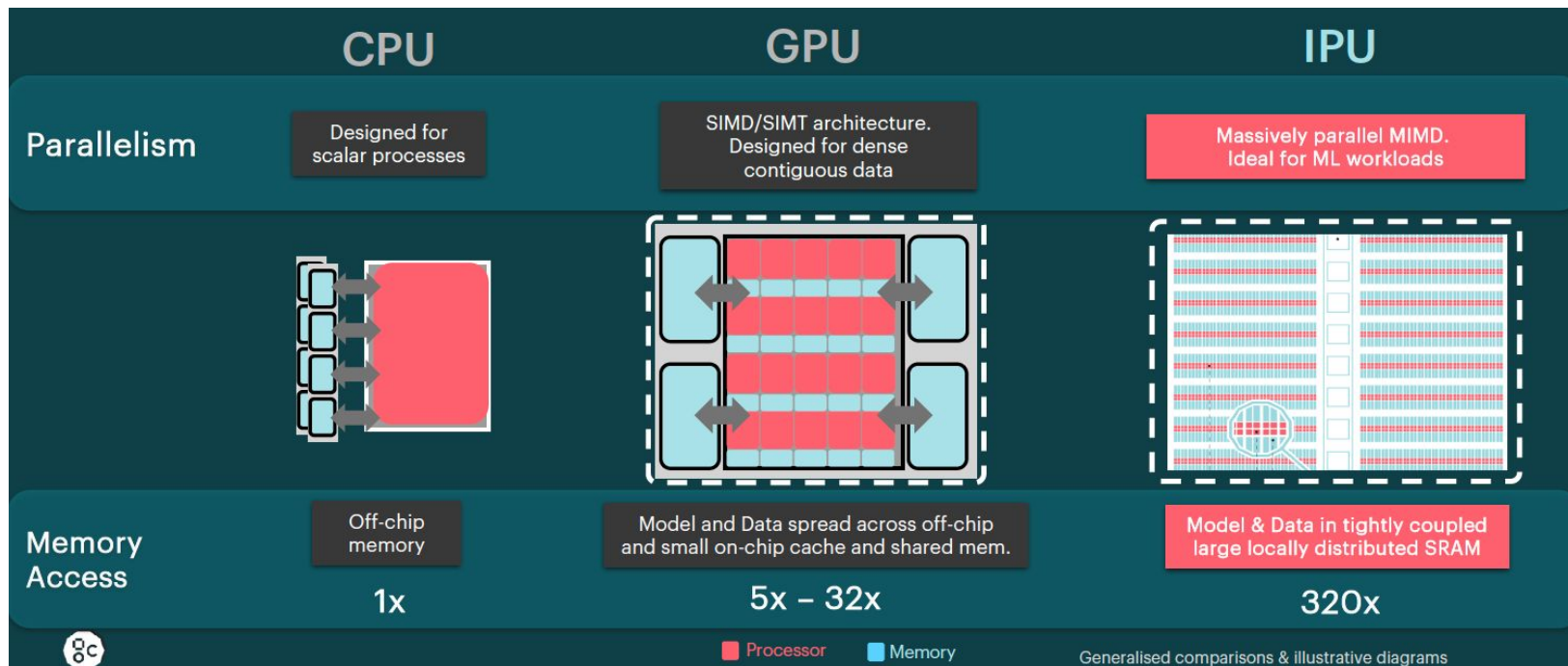
# Hands-On Activity - 3 Minutes

- 1. Create a submission file for a serial job.
  - Write the following below the line '#run your program':
    - echo 'Hello World' > ExOutput.txt
- 2. Submit this job file using sbatch.
- 3. Check which queue the job is in.
- 4. When your job completes, check the contents of the ExOutput.txt file.

# Job Submission and Tracking

| <b>FASTER</b>                                  | <b>Description</b>                                                              |
|------------------------------------------------|---------------------------------------------------------------------------------|
| <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<br>(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>lnu job_id</code>                        | Show resource usage for a job                                                   |
| <code>pestat -u \$USER</code>                  | Show resource usage for a running job                                           |
| <code>seff job_id</code>                       | Check CPU/memory efficiency for a job                                           |

# GraphCore IPU



# Accessing ACES - Graphcore IPU

- SSH into poplar
  - `[username@faster ~]$ ssh poplar1`
- Enable the SDK environment
  - `source /opt/gc/poplar/poplar_sdk-ubuntu_18_04-2.5.1+1001-64add8f33d/poplar-ubuntu_18_04-2.5.0+4748-e94d646535/enable.sh`
  - `source /opt/gc/poplar/poplar_sdk-ubuntu_18_04-2.5.1+1001-64add8f33d/popart-ubuntu_18_04-2.5.1+4748-e94d646535/enable.sh`
  - `mkdir -p /localdata/$USER/tmp`
  - `export TF_POPLAR_FLAGS=--executable_cache_path=/localdata/$USER/tmp`
  - `export POPTORCH_CACHE_DIR=/localdata/$USER/tmp`
- Type `gc-monitor` to view the status of the IPU

## Interested in IPUs?

- ACES IPU Training Short Course
  - Tuesday, September 27 1:30PM-4:00PM
  - Limited availability, Register Now!
  - [https://hprc.tamu.edu/training/aces\\_ipus.html](https://hprc.tamu.edu/training/aces_ipus.html)



# Intel FGAs



[https://hprc.tamu.edu/wiki/ACES#Intel\\_FPGA\\_PAC\\_D5005](https://hprc.tamu.edu/wiki/ACES#Intel_FPGA_PAC_D5005)

# Accessing FPGAs

<https://hprc.tamu.edu/wiki/ACES>

- Enter `srun --partition=fpga --time=24:00:00 --pty bash`
- Enter `source /opt/intel/oneapi/setvars.sh`
- Enter `fpgainfo` to view fgpa telemetry
- Enter `aocl diagnose` to run a status check on the fpga

# Need Help?

- First check the FAQ [hprc.tamu.edu/wiki/HPRC:CommonProblems](http://hprc.tamu.edu/wiki/HPRC:CommonProblems)
  - FASTER User Guide [hprc.tamu.edu/wiki/FASTER](http://hprc.tamu.edu/wiki/FASTER)
    - Exercises [hprc.tamu.edu/wiki/FASTER:Exercises](http://hprc.tamu.edu/wiki/FASTER:Exercises)
  - Email your questions to [help@hprc.tamu.edu](mailto:help@hprc.tamu.edu). (Managed by a ticketing system)
  
- Help us, help you -- we need more info
  - Which Cluster
  - UserID/NetID (*UIN is not needed!*)
  - Job id(s) if any
  - Location of your jobfile, input/output files
  - Application used if any
  - Module(s) loaded if any
  - Error messages
  - Steps you have taken, so we can reproduce the problem



**HIGH PERFORMANCE  
RESEARCH COMPUTING**  
TEXAS A&M UNIVERSITY

**Thank you.**

*Any questions?*