HIGH PERFORMANCE RESEARCH COMPUTING

Introduction to Composable Resources: FASTER and ACES

HPRC Training

February 7, 2023

High Performance
Research Computing
DIVISION OF RESEARCH

Outline

- Composability
- FASTER Overview
- ACES Overview
- Usage Policies
- Accessing FASTER
- HPRC Computing Environment
- Cluster Computing Basics
- Accessing ACES

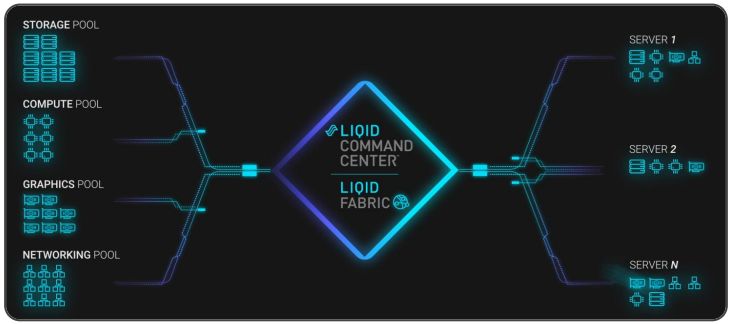
High Performance Computing (HPC) Architecture Comparison

Legacy HPC

- Built on Converged HW
- Static Hardware Design
- Fixed GPUs/Accelerators
- Fixed Memory
- Legacy Storage: SATA and SAS

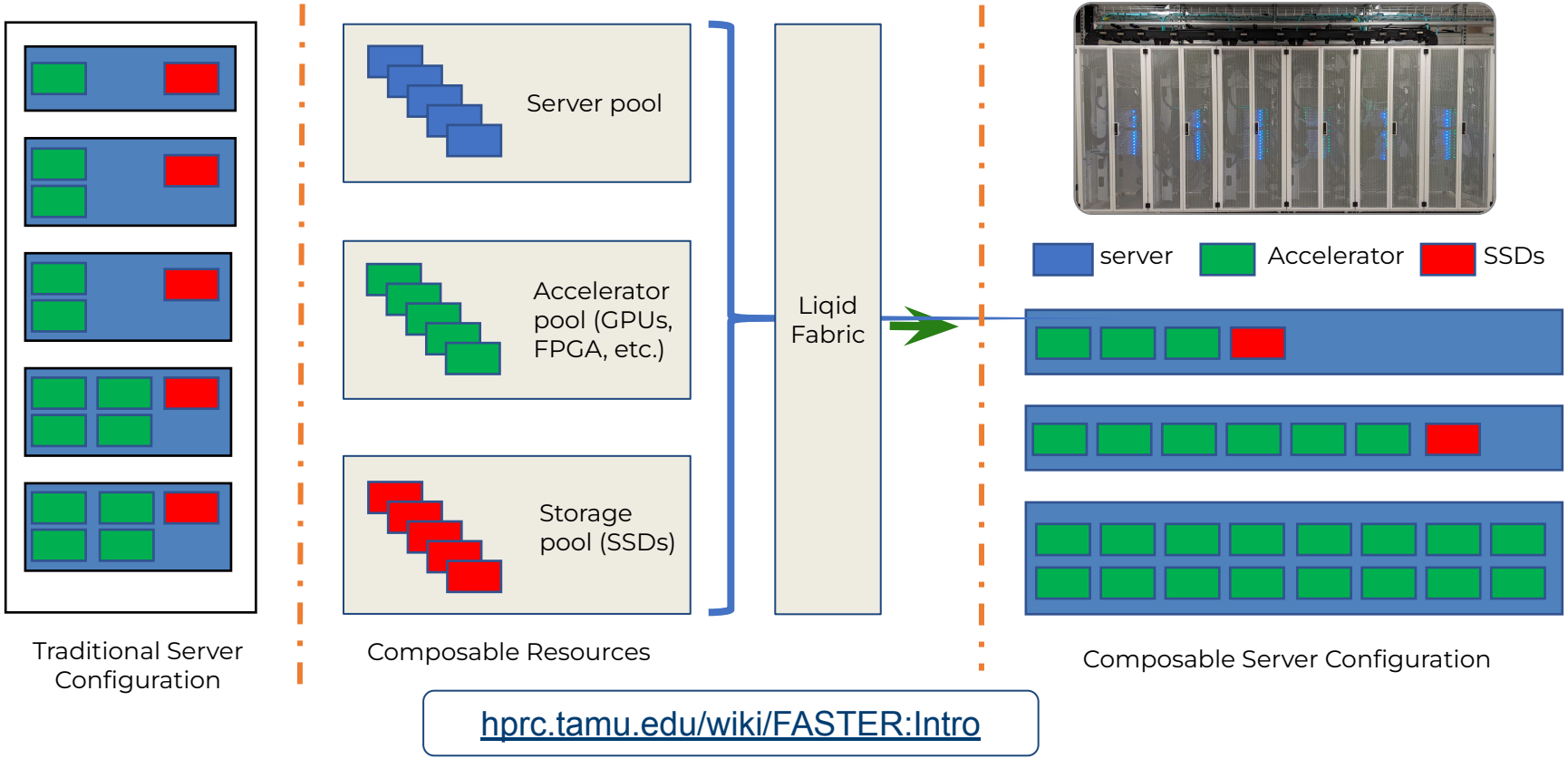
Modern HPC

- Built on Disaggregated HW
- Composable Hardware Platform
- Composable GPUs/Accelerators
- Composable Memory - Optane
- Modern Storage: NVMe-oF



A Modern HPC Platform Supporting Composable GPUs/Accelerators and Storage

Composability at the Hardware Level



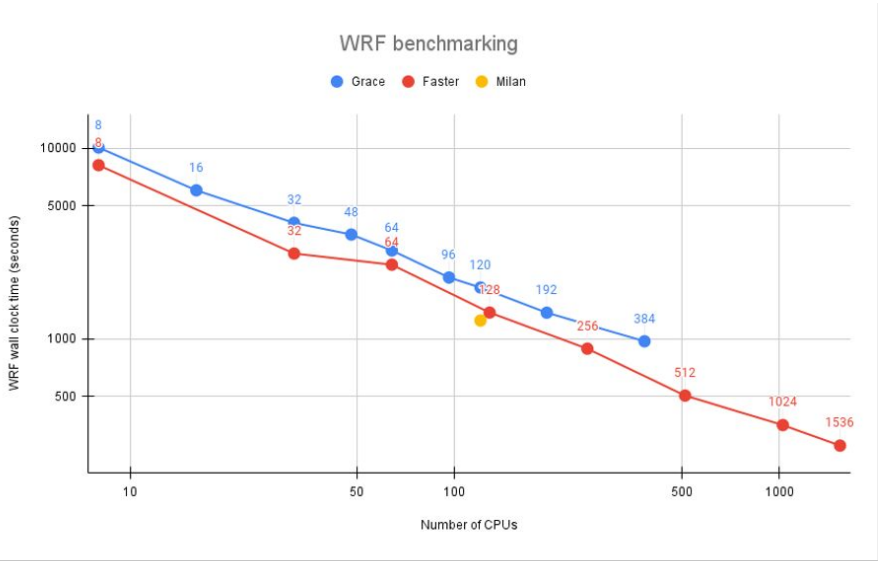
hprc.tamu.edu/wiki/FASTER:Intro

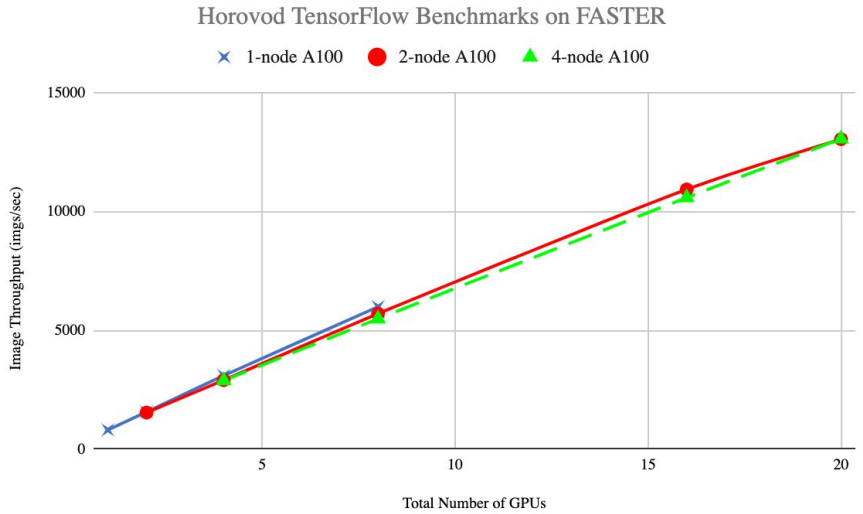
FASTER (Fostering Accelerated Scientific Transformations, Education, and Research)

hprc.tamu.edu/wiki/FASTER:Intro

Node Type	Quantity
64-core login nodes	3 (2 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) Liqid PCIe Gen4 (GPU composability)
Global Disk	5PB DDN Lustre appliances

Scaling on Composable Fabrics





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

ACES - Accelerating Computing for Emerging Sciences (Phase I)

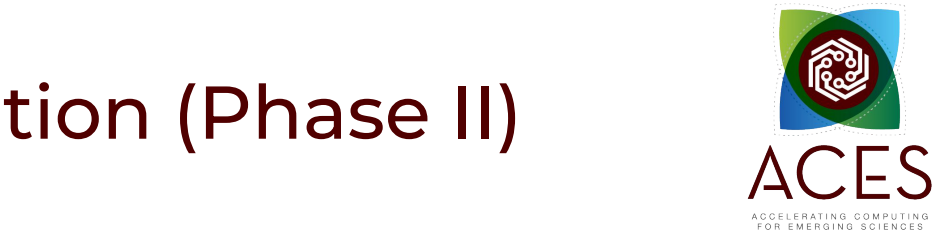
Component	Quantity	Description
Graphcore IPU	16	16 Colossus GC200 IPUs and dual AMD Rome CPU server on a 100 GbE RoCE fabric
Intel FPGA PAC D5005	2	FPGA SOC with Intel Stratix 10 SX FPGAs, 64 bit quad-core Arm Cortex-A53 processors, and 32GB DDR4
Intel GPU (under NDA)	12	Intel GPUs for HPC, DL Training, AI Inference
Intel Optane SSDs	8	3 TB of Intel Optane SSDs addressable as memory using MemVerge Memory Machine.

Available through FASTER (NSF Award #2019129)

ACES - Accelerating Computing for Emerging Sciences (Phase II)

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

ACES System Description (Phase II)

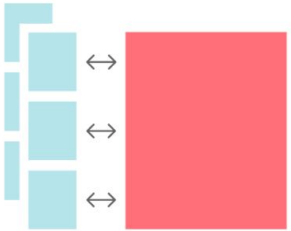
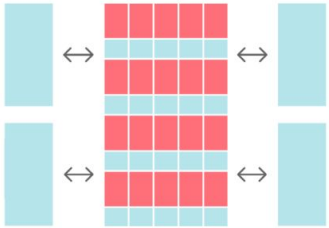
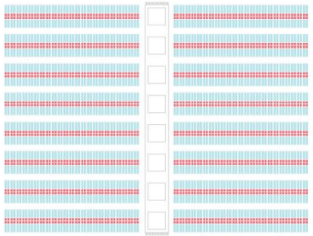


Component	Description
CPU-centric computing with variable memory requirements	Dual Intel Sapphire Rapids 2.1 GHz 48 core processors **96** cores per node, 512 GB memory, 1.6 TB NVMe storage (PCIe 5.0), NVIDIA Mellanox NDR 200 Gbps InfiniBand
Composable infrastructure	Reconfigurable infrastructure that allows up to 20 PCIe cards (GPU, FPGA, VE, etc.) per compute node
Data transfer nodes	Same as compute nodes, 100 Gbps network adapter

Research Workflows - Accelerators (Phases I and II)

Hardware Profile	Applications Supported	
NEC Vector Engines	• AI/ML (Statistical Machine Learning, Data Frame) • Chemistry (VASP, Quantum ESPRESSO) • Earth Sciences • NumPy Acceleration	• Oil & Gas (Seismic Imaging, Reservoir Simulation) • Plasma Simulation • Weather/Climate Simulation
Graphcore IPUs	• Graph Data • LSTM Neural Networks	• Markov Chain Monte Carlo • Natural Language Processing (Deep Learning)
Intel/Bittware FPGA	• AI Models for Embedded Use Cases • Big Data • CXL Memory Interface • Deep Learning Inference • Genomics	• MD Codes • Microcontroller Emulation for Autonomy Simulations • Streaming Data Analysis
Intel Optane SSDs	• Bioinformatics • Computational Fluid Dynamics (OpenFOAM)	• MD Codes • R • WRF
NextSilicon	• Biosciences (BLAST) • Computational Fluid Dynamics (OpenFOAM) • Cosmology (HACC) • Graph Search (Pathfinder)	• Molecular Dynamics (NAMD, AMBER, LAMMPS) • Quantum ChromoDynamics (MILC) • Weather/Environment modeling (WRF)

GraphCore IPU

	CPU	GPU	IPU
Parallelism	Designed for scalar processes	SIMD/SIMT architecture. Designed for large blocks of dense contiguous data	Massively parallel MIMD. Designed for fine-grained, high-performance computing
			
Memory Access	Off-chip memory	Model and data spread across off-chip and small on-chip cache, and shared memory	Model and data tightly coupled, and large locally distributed SRAM

www.graphcore.ai/bow-processors

GRAPHCORE

Porting TensorFlow 2 Models Quick Start

Version: Latest

Search this document

Version: latest

Download PDF

1. IMPORT THE TENSORFLOW IPU MODULE

First, we import the TensorFlow IPU module.

Add the import statement in Listing 1.1 to the beginning of your script.

Listing 1.1 Importing ipu Python module

```
from tensorflow.python import ipu
```

For the `ipu` module to function properly, we must import it directly rather than accessing it through the top-level TensorFlow module.

2. IPU CONFIG

To use the IPU, you must create an IPU session configuration in the main process. A minimum configuration is in Listing 2.1.

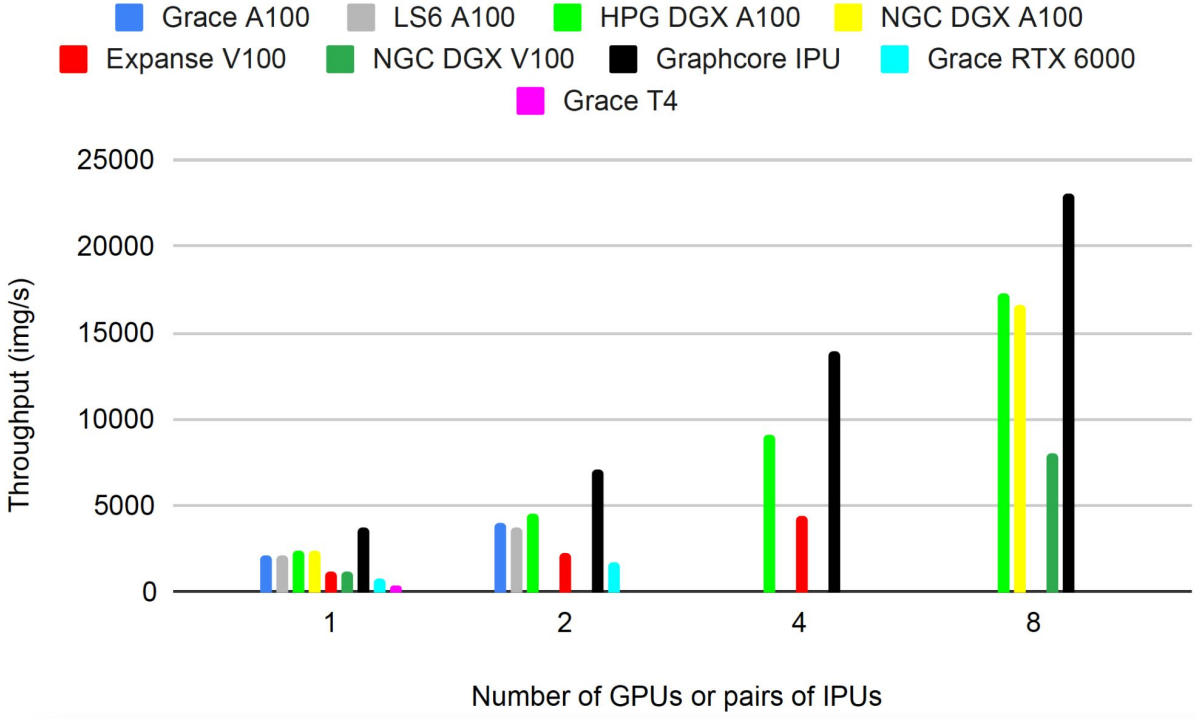
Listing 2.1 Example of a minimum configuration

```
ipu_config = ipu.config.IPUConfig()
ipu_config.auto_select_ipus = 1  # Select 1 IPU for the model
ipu_config.configure_ipu_system()
```

This is all we need to get a small model up and running. A full list of configuration options is available in the Python API documentation.

docs.graphcore.ai/en/latest/

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

Graphcore IPU Training

IPU Labs

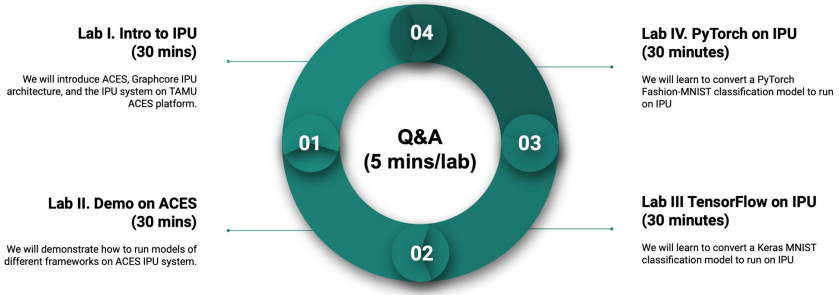
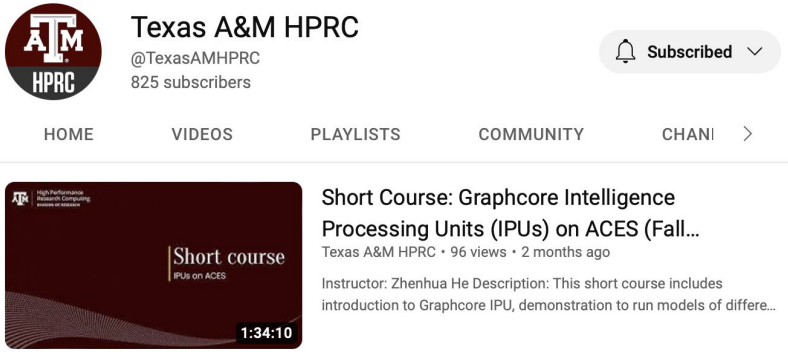


Figure 1. Structure of the IPU Training Laboratories.



https://youtu.be/E_tQA-VuNEU

ACES - GRAPHCORE IPU: PRACTICAL ADVICE FOR USERS

https://hprc.tamu.edu/training/aces_ipus.html

Overview

Instructor: Alex Tsyplikhin of Graphcore

Time: Tuesday, February 21, 2023 1:30PM-4:00PM CST

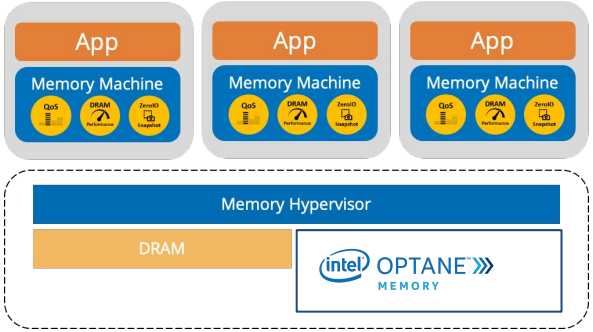
Location: Online using Zoom

Agenda

There are a total of four lab sessions:

1. **Intro to IPU** (30 mins)
   We will introduce Graphcore, IPU architecture, and the IPU system on TAMU FASTER platform.

MemVerge Memory Machine



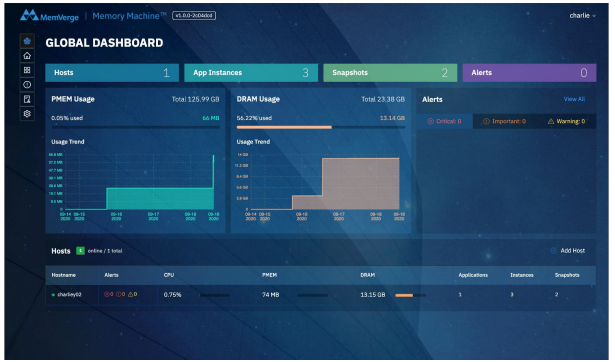
HPL Benchmark

HPL baseline without MemVerge Memory Machine:

T/V	N	NB	P	Q	Time	Gflops
WR12C2R4	140000	384	1	1	569.21	**3.21387e+03**

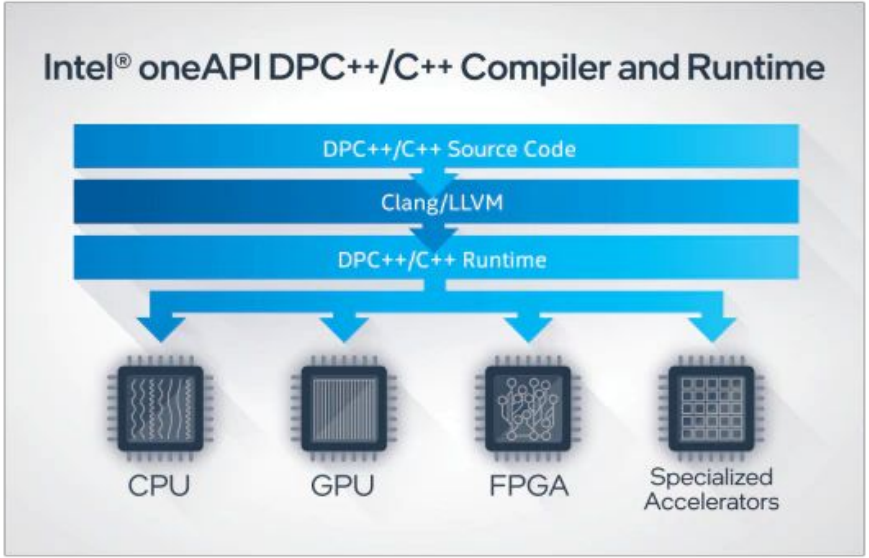
HPL with same input and memory usage above 140G going to SSDs with MemVerge Memory Machine:

T/V	N	NB	P	Q	Time	Gflops
WR12C2R4	140000	384	1	1	600.93	**3.04423e+03**



Intel FPGA PAC D5005

- Intel Stratix 10 SX FPGA family
  - Inline interfaces - 100 Gbps
- Fast code prototyping - OneAPI DPC++
  - Compile on CPU, validate design, then compile via FPGA
- Software
  - Intel Quartus Prime Pro
  - Intel FPGA SDK for OpenCL
  - Intel FPGA Add-On for the oneAPI Base Toolkit



Intel FPGA PAC D5005

FFTFPGA

FFTFPGA is an OpenCL based library for Fast Fourier Transformations for FPGAs. This repository provides OpenCL host code in the form of FFTW like APIs, which can be used to offload existing FFT routines to FPGAs with minimal effort. It also provides OpenCL kernels that can be synthesized to bitstreams, which the APIs can utilize.

Supported FPGAs

This library has been tested using the following FPGAs present in the Noctua cluster of the Paderborn Center for Parallel Computing (PC2) at Paderborn University:

- Bittware 520N card with Intel Stratix 10 GX 2800 FPGA
- Intel FPGA PAC D5005 card with Intel Stratix 10 SX 2800 FPGA

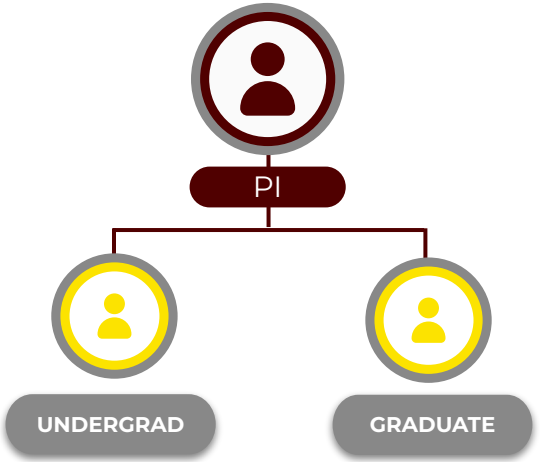
Who is using FFTFPGA?

- CP2K: the quantum chemistry software package has an interface to offload 3d FFTs to Intel FPGAs that uses the OpenCL kernel designs of FFTFPGA.

github.com/pc2/fft3d-fpga

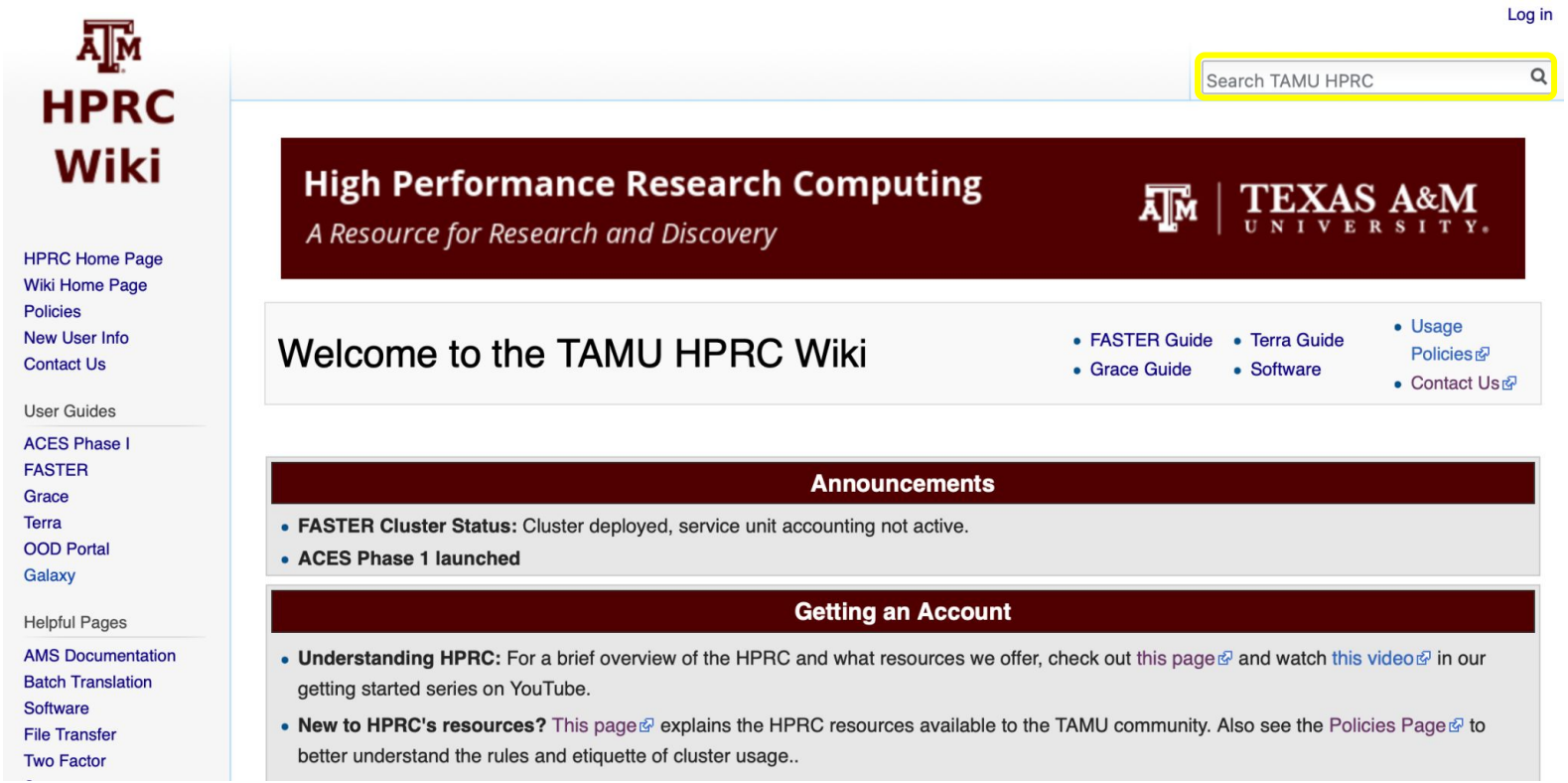
Getting on ACES

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

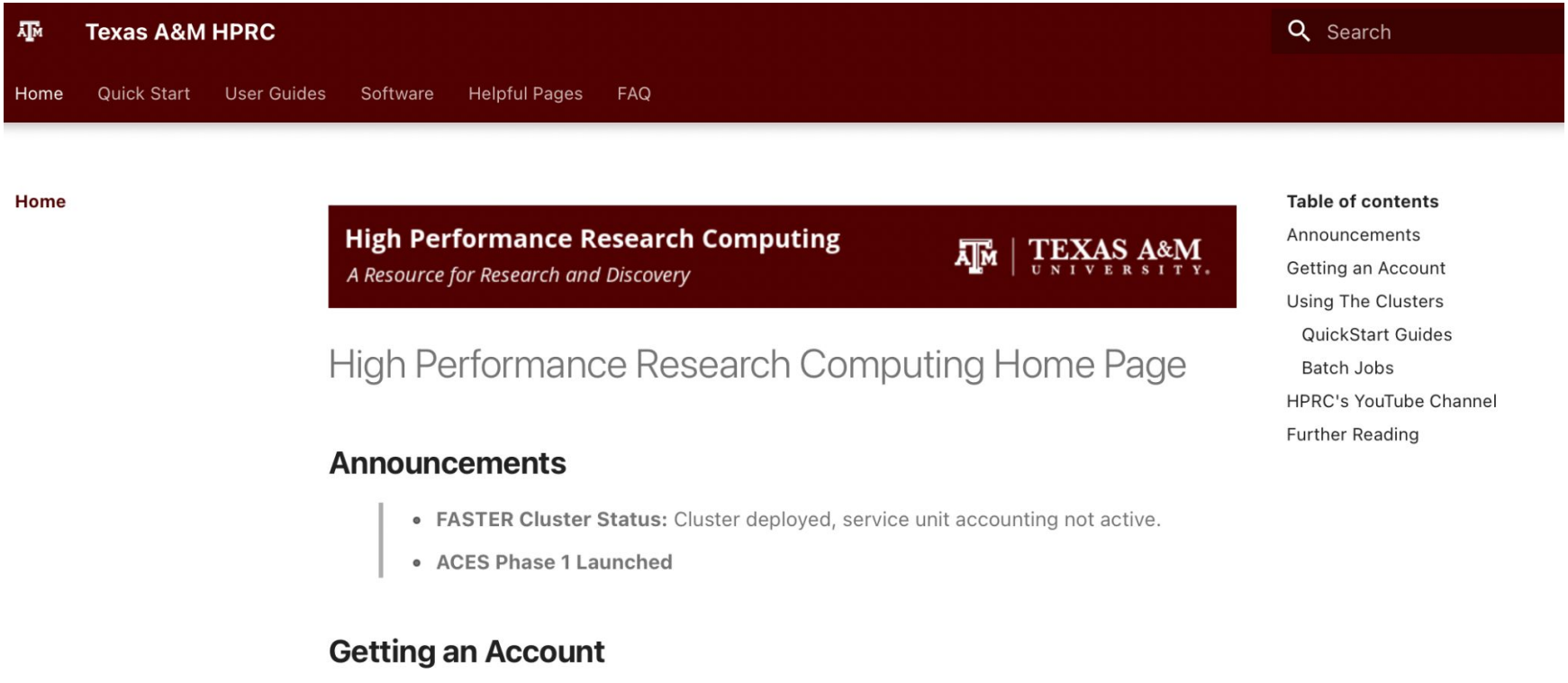


PIs can apply for an account and sponsor accounts for their researchers.

HPRC Documentation Wiki



HPRC Documentation - Sneak Peak

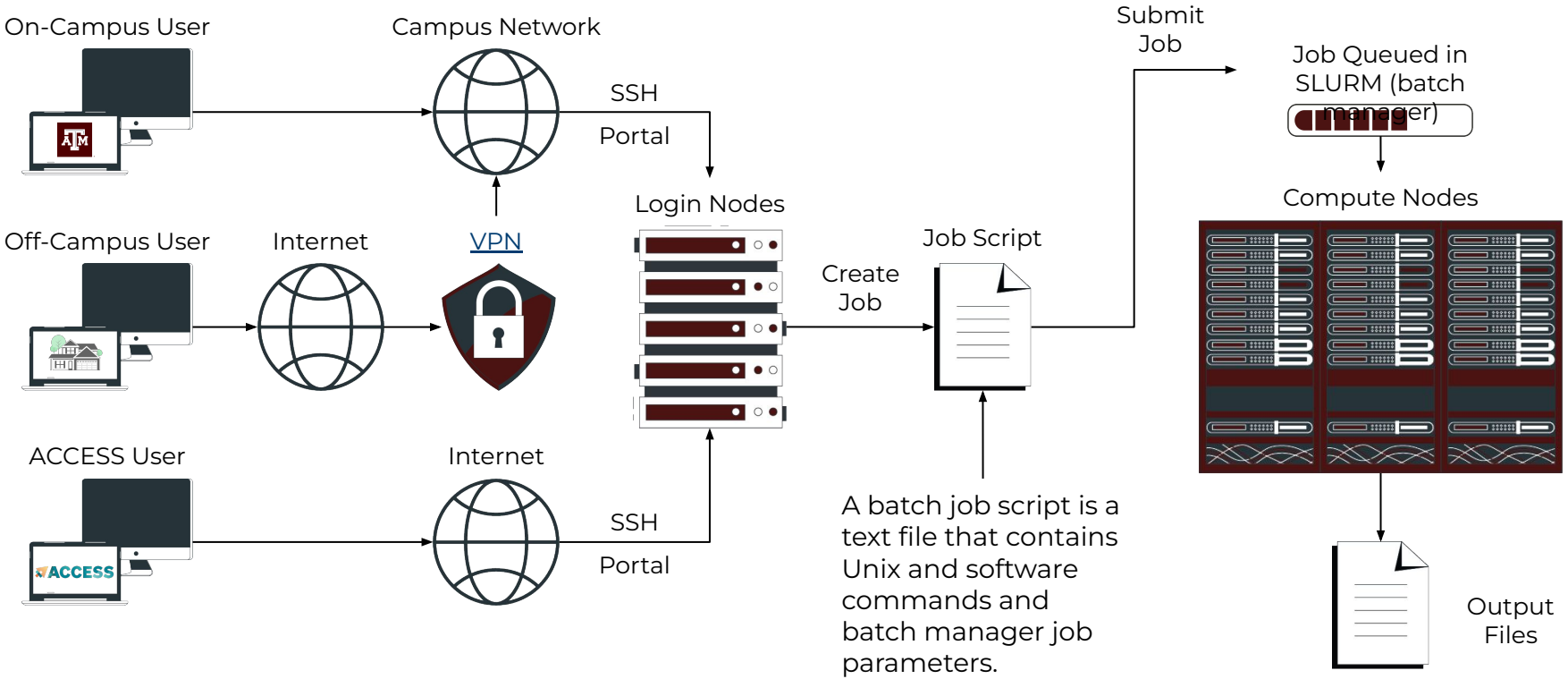


Usage Policies
(Be a good compute citizen)

- 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

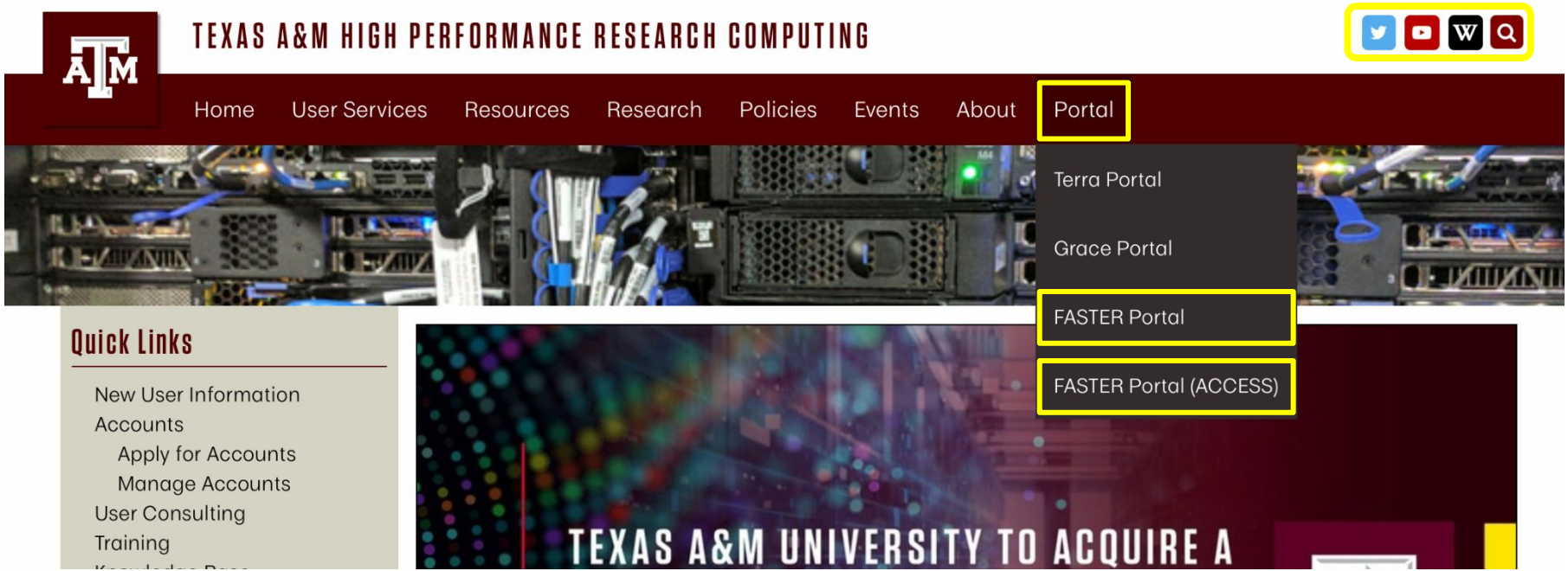
hprc.tamu.edu/policies

Batch Computing on HPRC Clusters



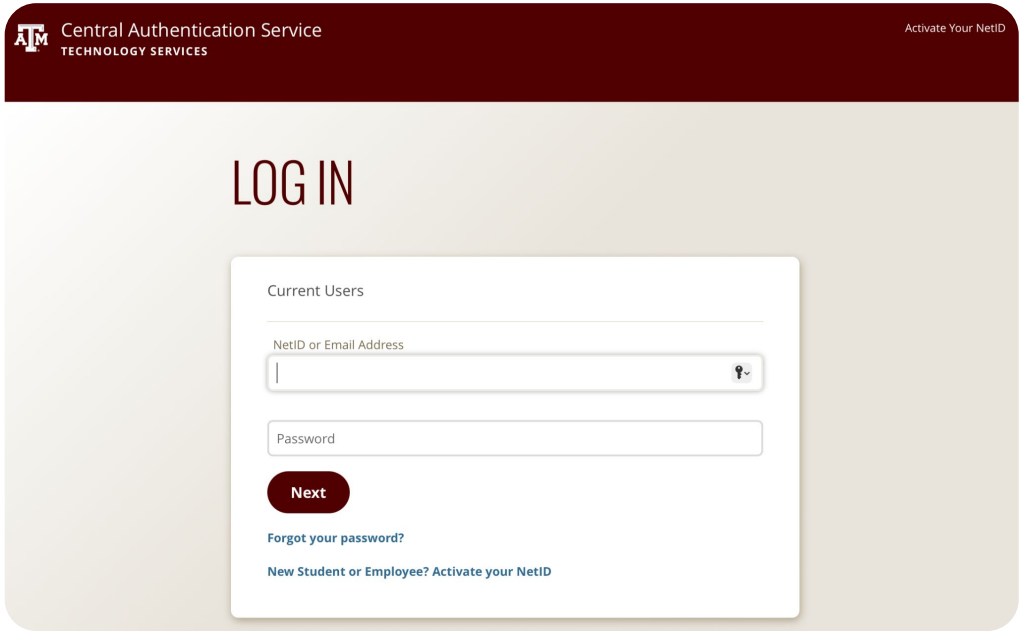
Accessing the HPRC Portal

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



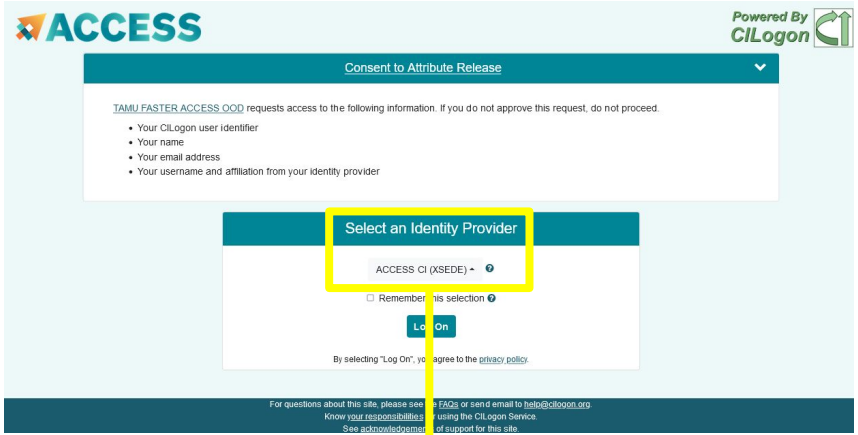
Accessing FASTER via the HPRC Portal (TAMU)

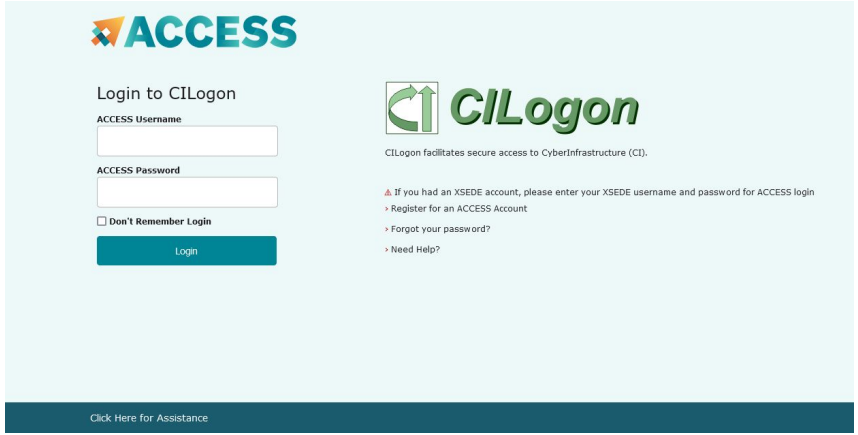
Log-in using your TAMU NetID credentials.

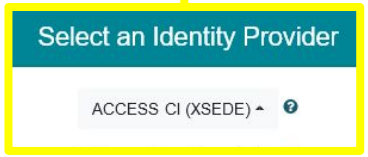


Accessing FASTER via the HPRC Portal (ACCESS)

Log-in using your ACCESS credentials.





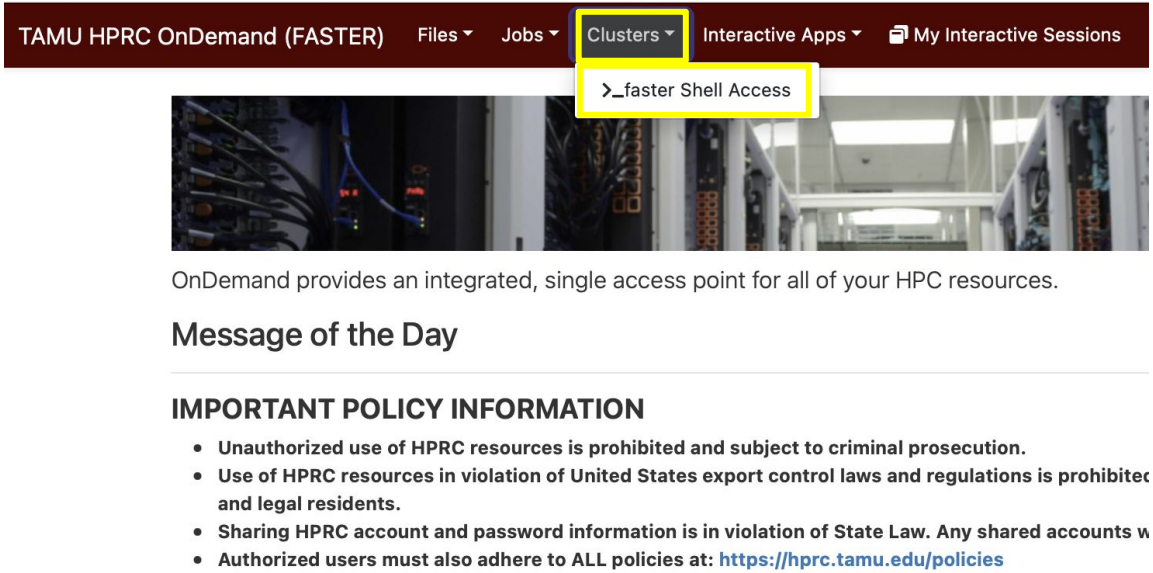


Select the Identity Provider appropriate for your account.

Shell access via the HPRC Portal

Access through (most) web browsers

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



File Systems and User Directories

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

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

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

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

hprc.tamu.edu/wiki/FASTER:Filesystems_and_Files

1,500+ Software Modules!

SOFTWARE MODULES ON THE FASTER CLUSTER

Faster Software Modules | Grace Software Modules | Terra Software Modules

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 leve... optimizations and transforms.'

hprc.tamu.edu/software/faster/

Software

- HPRC provides both pre-installed Software and installation assistance
- See the Software pages for instructions and examples
  - hprc.tamu.edu/wiki/SW
  - hprc.tamu.edu/software/
- License-restricted software
  - Contact help@hprc.tamu.edu
- Contact us for software installation help/request
  - User can install software in their home/scratch directory
  - Do NOT run the "sudo" command when installing software

Computing Environment

Managing software versions using lmod

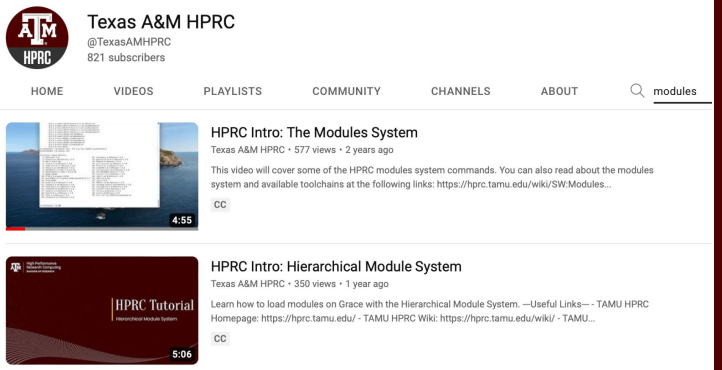
- Uses the command: **module**
- Each version of a software, application, library, etc. is available as a module.
  - Module names have the format:

toolchain-name / version toolchain-name / version

GCC/10.3.0 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

Module Usage Practices

- Software installed as modules are available to all users
  - (except for restricted modules)
- It's a good habit to unload modules before loading new modules.

```
module purge
```

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

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

Software: 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 purge                    # unload all modules
module list                     # shows which modules are loaded
module load GCC/10.3.0          # load GCC compiler version 10.3.0
module list                     # show which modules are loaded
```

Module Usage Example

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

```
mla Pytorch
```

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

```
PyTorch/1.7.1
PyTorch/1.8.1
PyTorch/1.9.0-imkl
PyTorch/1.9.0
PyTorch/1.10.0-CUDA-11.3.1
PyTorch/1.10.0
```

```
module spider PyTorch/1.10.0
```

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

```
----------------------------------------------------------------------------------------------------------------
  PyTorch: PyTorch/1.10.0
----------------------------------------------------------------------------------------------------------------
    Description:
      Tensors and Dynamic neural networks in Python with strong GPU acceleration. PyTorch is a deep learning
      framework that puts Python first.

    You will need to load all module(s) on any one of the lines below before the "PyTorch/1.10.0" module is available to load.

      GCC/10.3.0  OpenMPI/4.1.1
```

```
module load GCC/10.3.0  OpenMPI/4.1.1 PyTorch/1.10.0
```

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

Software Install Example: Virtual Env

Python is a language which supports many external libraries in the form of extensions. (called Python Packages).

Some commonly used packages:

- SciPy & NumPy
- Jupyter notebook
- Scikit-learn

You can install these yourself using the Virtual Environment feature. Instructions are on the wiki:

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

Exercise: Python Software Install

1.
```
cd $SCRATCH
mkdir python_envs
cd python_envs
```
setup workspace

2.
```
module purge
module load GCC/10.2.0 Python/3.8.6
```
setup Python module

3.
```
virtualenv my_example_venv
source my_example_venv/bin/activate
```
setup your virtual environment

4.
```
python -c "import pytime"
```
check if python-time is installed (it's not)

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

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

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

Consumable Computing Resources

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

```
myproject
```

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

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

```
license_status -a
```

Find available license for "Matlab":

```
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 **3.5** 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

sinfo : Current Queues on FASTER

PARTITION	AVAIL	TIMELIMIT	JOB_SIZE	NODES(A/I/O/T)	CPUS(A/I/O/T)
cpu*	up	7-00:00:00	1-32	99/0/13/112	1426/4460/1282/7168
gpu	up	7-00:00:00	1-32	38/0/14/52	1190/1242/896/3328
memverge	up	2-00:00:00	1-2	0/0/2/2	0/0/128/128
fpga	up	2-00:00:00	1-2	0/0/2/2	0/0/128/128
spr	up	2-00:00:00	1	0/1/0/1	0/112/0/112
atsp	up	2-00:00:00	1	0/10/0/10	0/640/0/640
staff	down	7-00:00:00	1-infinite	137/10/31/178	2616/6342/2434/11392

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 Num_CPU  CPUload  Memsize  Freemem  GRES/node
                           State Use/Tot  (15min)     (MB)     (MB)
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
```

Batch Job Scripts

Sample Job Script Structure (FASTER)

```
#!/bin/bash
#NECESSARY JOB SPECIFICATIONS
#SBATCH --job-name=JobExample1
#SBATCH --time=01:30:00
#SBATCH --ntasks=8
#SBATCH --mem=3G
#SBATCH --gres=gpu:T4:8
#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 the resources needed for your program/simulation to the job scheduler

Account number to be charged

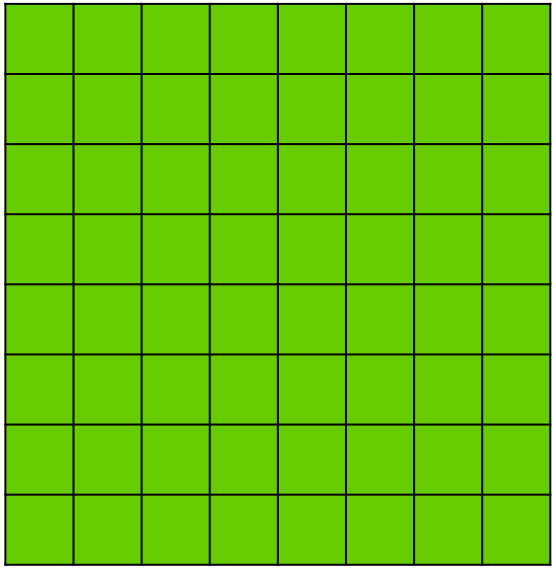
This is single line comment and not run as part of the script

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.

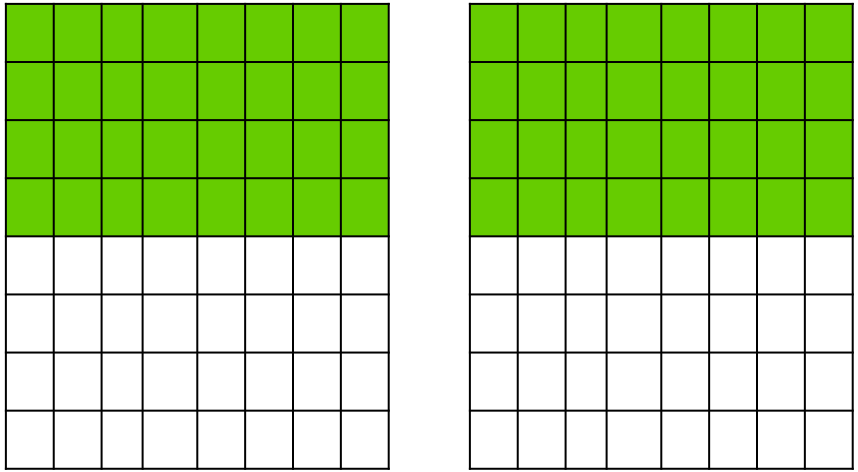


64 cores on
1 compute node

#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

FASTER 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

##OPTIONAL JOB SPECIFICATIONS
#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
# load required module(s)
module purge
module load intel/2022a
module load CUDA/11.7.0
# run your program
./my_gpu_program
```

SUs = 1344

Job Submission and Tracking

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

hprc.tamu.edu/wiki/HPRC:Batch_Translation

Knowledge Base

Graphcore IPUs [edit]

From the login node, ssh into the poplar1 system.

```
[username@login ~]$ ssh poplar1
```

Set up the Poplar SDK environment [edit]

In this step, set up several environment variables to use the Graphcore tools and Poplar graph programming framework.

```
[username@poplar1 ~]$ source /opt/gc/poplar/poplar_sdk-ubuntu_20_04-[ver]/poplar-ubuntu_20_04-[ver]/enable.sh
[username@poplar1 ~]$ source /opt/gc/poplar/poplar_sdk-ubuntu_20_04-[ver]/popart-ubuntu_20_04-[ver]/enable.sh
```

[ver] indicates the version number of the package.

Example commands with an existing version on ACES:

```
source  /opt/gc/poplar/poplar_sdk-ubuntu_20_04-3.1.0+1205-58b501c780/poplar-ubuntu_20_04-3.1.0+6824-9c103dc348/enable.sh
```

```
mkdir -p /localdata/$USER/tmp
export TF_POPLAR_FLAGS=--executable_cache_path=/localdata/$USER/tmp
export POPTORCH_CACHE_DIR=/localdata/$USER/tmp
# export POPLAR_LOG_LEVEL=INFO
```

hprc.tamu.edu/wiki/ACES

Accessing ACES - Graphcore IPUs

- SSH into poplar
  - **[username@login ~]$ ssh poplar1**
- Enable the SDK environment
  - **source /opt/gc/poplar/poplar_sdk-ubuntu_20_04-3.1.0+1205-58b501c780/poplar-ubuntu_20_04-3.1.0+6824-9c103dc348/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 IPUs

```
mouse@poplar1:~$ gc-monitor
+-------------+-----------------------------------------------------------------------------------+
| gc-monitor  |              Partition: p16 [active] has 16 reconfigurable IPUs                   |
+-------------+-----------------+--------+--------------+----------+---------+----+------+-------+
|    IPU-M    |      Serial     |IPU-M SW|Server version|  ICU FW  |  Type   | ID | IPU# |Routing|
+-------------+-----------------+--------+--------------+----------+---------+----+------+-------+
| 10.1.5.1    | 0010.0002.8213921 |      |    1.9.0     |  2.4.4   | M2000   | 0  |  3   |  DNC  |
| 10.1.5.1    | 0010.0002.8213921 |      |    1.9.0     |  2.4.4   | M2000   | 1  |  2   |  DNC  |
| 10.1.5.1    | 0010.0001.8213921 |      |    1.9.0     |  2.4.4   | M2000   | 2  |  1   |  DNC  |
| 10.1.5.1    | 0010.0001.8213921 |      |    1.9.0     |  2.4.4   | M2000   | 3  |  0   |  DNC  |
+-------------+-----------------+--------+--------------+----------+---------+----+------+-------+
| 10.1.5.2    | 0030.0002.8213921 |      |    1.9.0     |  2.4.4   | M2000   | 4  |  3   |  DNC  |
| 10.1.5.2    | 0030.0002.8213921 |      |    1.9.0     |  2.4.4   | M2000   | 5  |  2   |  DNC  |
| 10.1.5.2    | 0030.0001.8213921 |      |    1.9.0     |  2.4.4   | M2000   | 6  |  1   |  DNC  |
```

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

Accessing ACES FPGAs

Enter **`srun --partition=fpga --time=24:00:00 --pty bash`**

Enter **`fpgainfo`** `[errors,power,temp,fme,port,bmc]` to view fgpa telemetry

```
cp -R /scratch/training/oneapi-aces $SCRATCH
cd $SCRATCH/oneapi-aces
singularity shell --env-file env-vars oneapi-2022.1.0.sif
source /opt/intel/oneapi/setvars.sh
aocl diagnose
aocl initialize acl0 pac_s10
aocl list-devices
```

hprc.tamu.edu/wiki/ACES

Accessing Intel ATSP

```
##NECESSARY JOB SPECIFICATIONS
#SBATCH --job-name=Example          #Set the job name to Example
#SBATCH --time=24:00:00             #Set the wall clock limit to 24 hrs
#SBATCH --nodes=1                   #Request 1 nodes
#SBATCH --ntasks-per-node=64        #Request 64 tasks/cores per node
#SBATCH --mem=248G                  #Request 248G (248GB) per node
#SBATCH --output=Example.%j         #Redirect stdout/err to file
#SBATCH --partition=atsp            #Specify the Intel GPU

source /sw/restricted/oneapi_nda/setvars.sh

source /sw/restricted/oneapi_nda/setvars.sh --force source activate intelpython
```

https://hprc.tamu.edu/wiki/ACES#Intel_GPU

Accessing MemVerge

```
srun --partition=memverge --time=24:00:00 --pty bash

Sample job file:
#!/bin/bash

##NECESSARY JOB SPECIFICATIONS
#SBATCH --job-name=Example           #Set the job name to Example
#SBATCH --time=24:00:00              #Set the wall clock limit to 24 hrs
#SBATCH --nodes=1                    #Request 1 nodes
#SBATCH --ntasks-per-node=64         #Request 64 tasks/cores per node
#SBATCH --mem=248G                   #Request 248G (248GB) per node
#SBATCH --output=Example.%j          #Redirect stdout/err to file
#SBATCH --partition=memverge         #Specify the MemVerge partition

#lines required to setup the environment for your code
# add the mm command in front of your executable to run with memory machine
mm executable
```

https://hprc.tamu.edu/wiki/ACES#Liqid_PCIe_Card_with_Intel_Optane_SSDs

Need Help?

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

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

Help us, help you -- we need more info

- Which Cluster
- Username
- Job id(s) if any
- Location of your jobfile, input/output files
- Application used if any
- Module(s) loaded if any
- Error messages
- Steps you have taken, so we can reproduce the problem

High Performance
Research Computing
DIVISION OF RESEARCH

Thank you

Questions?