ACES: Introducing Intel PVC GPUs

Zhenhua He, Akhil Chilumuru, Druva Gunda, and Richard Lawrence

11/07/2023

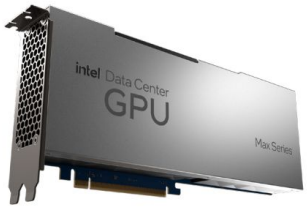
High Performance
Research Computing
DIVISION OF RESEARCH

Outline

Intro to ACES and Intel PVC	Demos on ACES	PyTorch on PVC	TensorFlow on PVC	LAMMPS on PVC
We will introduce Intel's PVC, its architecture, and the PVC GPUs on the TAMU ACES platform.	We will demonstrate how to run models of different frameworks with PVC GPUs on the ACES system.	Students will learn how to convert a PyTorch image classification model to run on a PVC GPU.	Students will learn how to convert a TensorFlow image classification model to run on a PVC GPU.	We will demonstrate how to run molecular dynamics simulations in the LAMMPS framework with PVC GPUs on the ACES system.

Lab I. Introducing Intel PVC GPUs on ACES

ACES
ACCELERATING COMPUTING FOR EMERGING SCIENCES



Intel Data Center GPU Max Series PCIE Card

intel

NSF ACES

Accelerating Computing for Emerging Sciences

Our Mission:

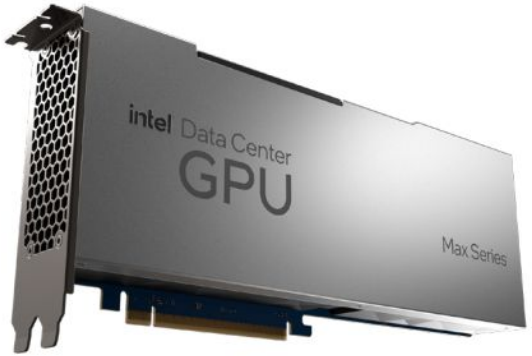
- Offer an accelerator testbed for numerical simulations and AI/ML workloads
- Provide consulting, technical guidance, and training to researchers
- Collaborate on computational and data-enabled research.

ACES Accelerators

Component	Quantity	Description
Graphcore IPU	32	16 Colossus GC200 IPUs, 16 Bow IPUs. Each IPU group hosted with a CPU server as a POD16 on a 100 GbE RoCE fabric
Intel PAC D5005 FPGA	2	Accelerator with Intel Stratix 10 GX FPGA and 32 GB DDR4
BittWare IA-840F FPGA	2	Accelerator with Agilex AGF027 FPGA and 64 GB of DDR4
NextSilicon Coprocessor	2	Reconfigurable accelerator with an optimizer continuously evaluating application behavior.
NEC Vector Engine	8	Vector computing card (8 cores and HBM2 memory)
Intel Optane SSD	48	18 TB of Intel Optane SSDs addressable as memory w/ MemVerge Memory Machine.
NVIDIA H100 GPU	30	NVIDIA GPUs for HPC, DL Training, AI Inference
NVIDIA A30 GPU	4	NVIDIA GPUs for AI Inference and Mainstream Compute
Intel Arctic Sound (ATS-P)	22	Software Development Platform for PVC

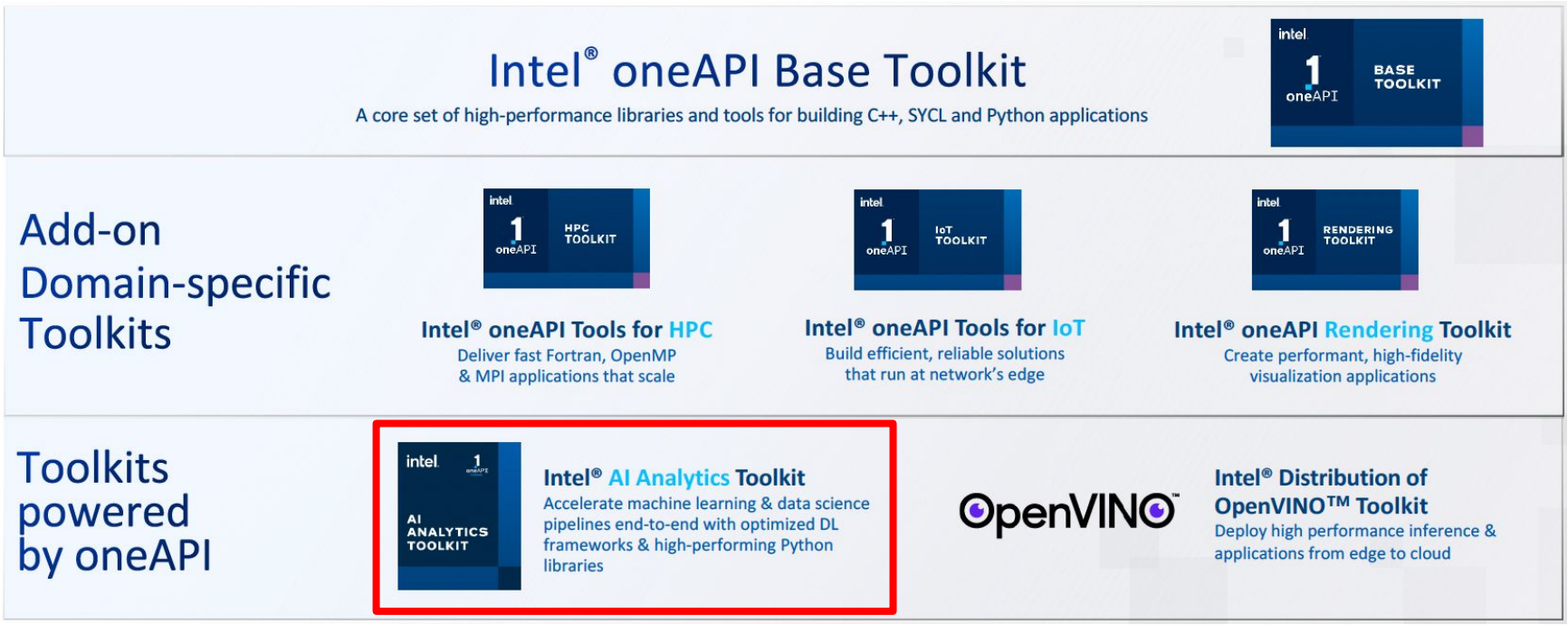
Intel Max GPU 1100

- 1 tile/stack per card
- 56 X^e cores, 448 execution units (8 per core)
- 300W PCIe Gen5 x16 card
- 48GB HBM2e memory
- 1.2 TB/s memory bandwidth
- 22 TF FP64 peak performance



Intel Data Center GPU Max Series PCIE Card

Intel® oneAPI Toolkits

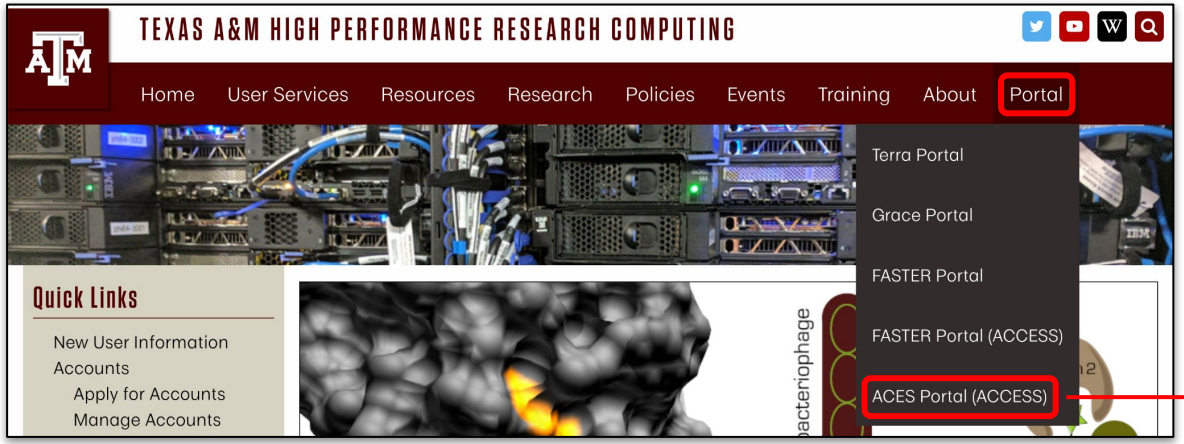


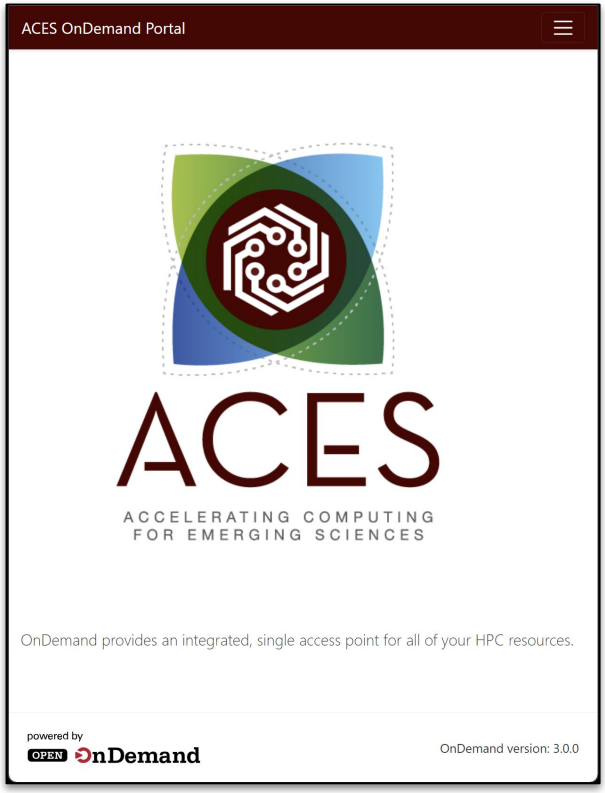
(Source: Intel)

Shared Data Directory on ACES

- Datasets: ImageNet datasets for PyTorch and TensorFlow
  /scratch/data/pytorch-computer-vision-datasets
  /scratch/data/tensorflow-computer-vision-datasets
- Models: Intel AI models
  /scratch/data/intel-ai-models
- Containers:
  /scratch/data/containers/intel-deep-learning-2023.2-py3.10-perms.sif

ACES Portal



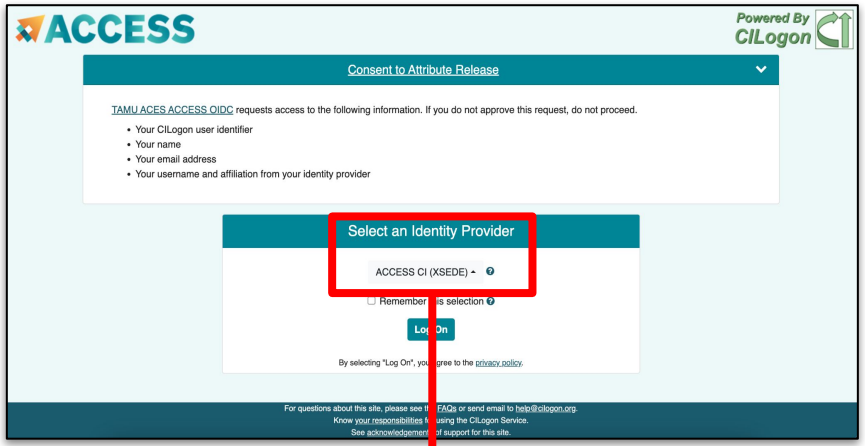


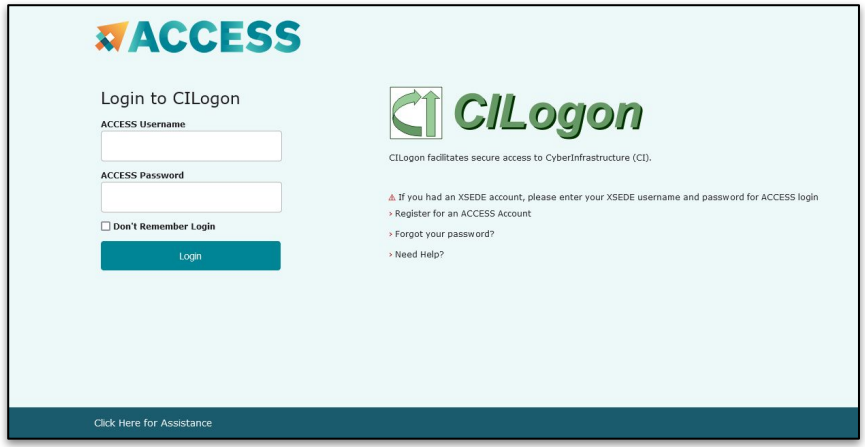
The ACES Portal portal-aces.hprc.tamu.edu
is the web-based user interface for the ACES cluster

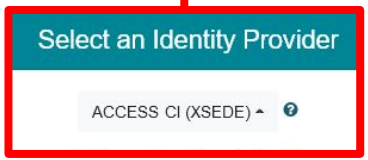
Open OnDemand (OOD) is an advanced web-based graphical interface framework for HPC users

Authentication via CILogon

Log-in using your ACCESS CI credentials.



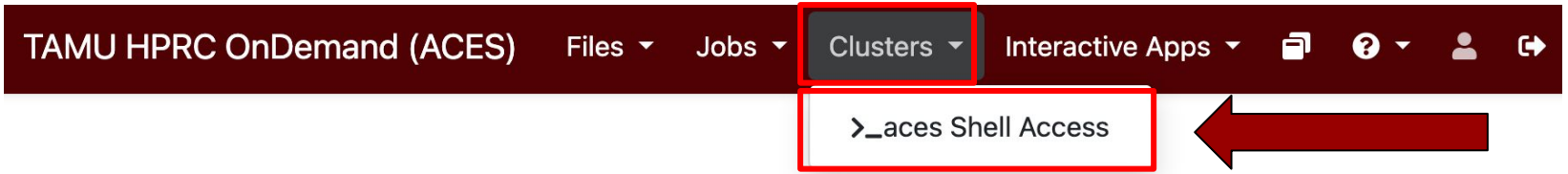




Select the Identity Provider appropriate for your account.

Get a Shell on ACES

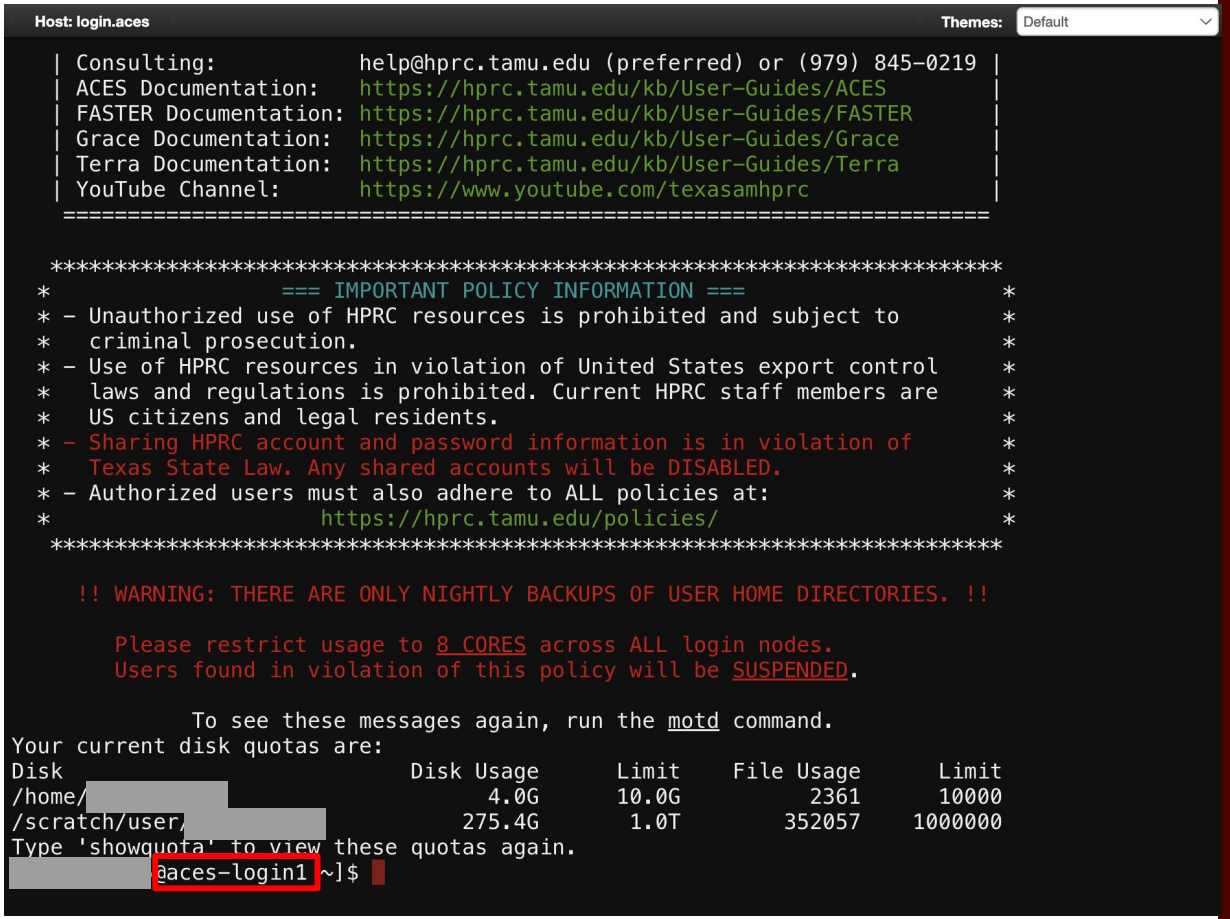
Click on “Clusters” menu → _aces Shell Access



Success!

Welcome to the ACES login node.

Check which login node you are on.



PVC Slurm Nodes Status Check

- View the pvc nodes and number of GPUs

```
$ pestat -p pvc -G
```

Copy the Materials to Personal Directory

- Navigate to your personal scratch directory

  ```
  $ cd $SCRATCH
  ```

- Files for this course are located at

  ```
  /scratch/training/aces_pvc_course
  ```

  Make a copy in your personal scratch directory

  ```
  $ cp -r /scratch/training/aces_pvc_course $SCRATCH
  ```

- Enter this directory (your local copy)

  ```
  $ cd $SCRATCH/aces_pvc_course
  ```

Lab II. Using PVCs on ACES

Environment Setup for PyTorch Models

Use Intel AI Analytics Toolkit

```
# load all the necessary modules
module purge
module load intel/AIKit/2023.2.0
module load intel/2023.07

ENV_NAME=aikit-pt-gpu-clone

# If it doesn't exist, create the environment
if ! conda env list | grep -q "$ENV_NAME"; then

    conda create -n $ENV_NAME --clone aikit-pt-gpu
fi

# activate the conda environment
source activate $ENV_NAME
```

in pt_demo.slurm

Environment Setup for PyTorch Models

Use Python Virtual Environment (Alternative for reference)

```
# Change to pytorch directory
cd $SCRATCH/aces_pvc_course/pytorch

# Load modules
module load WebProxy
module load intel/2023.03
module load Python/3.10.8

# Create and activate a Python virtual environment
python -m venv pt-pvc-labs
source pt-pvc-labs/bin/activate
```

Please do not type

Environment Setup for PyTorch Models

Use Python Virtual Environment (Alternative for reference)

```
# Install torch, torchvision and oneccl_bindings_for_pytorch
python -m pip install torch==1.13.0a0+git6c9b55e
torchvision==0.14.1a0 intel_extension_for_pytorch==1.13.120+xpu -f
https://developer.intel.com/ipex-whl-stable-xpu

python -m pip install oneccl_bind_pt==1.13.200+gpu -f
https://developer.intel.com/ipex-whl-stable-xpu

# Install tensorboard
python -m pip install tensorboard
```

Run PyTorch ResNet50 model

- We have prepared a Slurm job file (*pt_demo.slurm*) to run the PyTorch ResNet50 model. Submit the job using the command
- $ cd pytorch/
- $ sbatch pt_demo.slurm

Environment Setup for TensorFlow Models

Using the Intel AI Analytics Toolkit

```
# load all the necessary modules
module purge
module load intel/2023.07
module load intel/AIKit/2023.2.0

ENV_NAME=aikit-tf-gpu-clone

# If it doesn't exist, create the environment
if ! conda env list | grep -q "$ENV_NAME"; then

    conda create -n $ENV_NAME --clone aikit-tf-gpu
fi

# activate the conda environment
source activate $ENV_NAME
```

in tf_demo.slurm

Run Tensorflow ResNet50 Model

- We have prepared a Slurm job file (*tf_demo.slurm*) to run the Tensorflow ResNet50 model. Submit the job using the command

```
$ cd ..
$ cd tensorflow/
$ sbatch tf_demo.slurm
```

Lab III. PyTorch on PVC

1. Import Intel Extension for PyTorch

Intel Extension for PyTorch is a Python package for extending PyTorch models to run on an Intel platform.

Add the following import statement to the beginning of your script:

```
import intel_extension_for_pytorch as ipex
```

2. Move the Model and Criterion to “xpu”

```
model = model.to("xpu")
criterion = criterion.to("xpu")
```

3. Apply the “ipex optimize” Function

Apply the ipex optimize function against the model and optimizer objects.

```
model, optimizer = ipex.optimize(model, optimizer=optimizer, dtype=torch.bfloat16)
```

4. Move the Data and Target to “xpu”

In the training loop,

```
data = data.to("xpu")
target = target.to("xpu")
```

5. Use Auto Mixed Precision (AMP)

Use automatic mixed-precision (AMP) with BFloat16 data type with the *torch.xpu.amp.autocast* context manager

```
with torch.xpu.amp.autocast(enabled=True, dtype=torch.bfloat16):
```

Hands-on Session

- Navigate to the pytorch exercises directory

  ```
  $ cd $SCRATCH/aces_pvc_course/pytorch/exercises
  ```

- Open the exercise file (*cifar10_pvc_todo.py*) with your preferred editor (e.g. vim) or the file editor of the OnDemand portal.
- Complete the **#Todos** in the *cifar10_pvc_todo.py* file.
- Modify the Slurm job file (*pt_cifar10_pvc.slurm*) and submit your job.

  ```
  $ sbatch pt_cifar10_pvc.slurm
  ```

Lab IV. TensorFlow on PVC

Install Intel Extension for Tensorflow

The Intel Extension for Tensorflow is based the on Tensorflow PluggableDevice interface to bring Intel XPU (GPU, CPU, etc) devices into Tensorflow.

To check the version, add import statement to the beginning of your script:

```
import intel_extension_for_tensorflow as itex
print(itex.__version__)
```

The default device will be Intel GPU after installing `intel-extension-for-tensorflow`

Source: Intel presentation at ACES Workshop

No Code Changes are Needed!

Credit: Bing Chat Enterprise

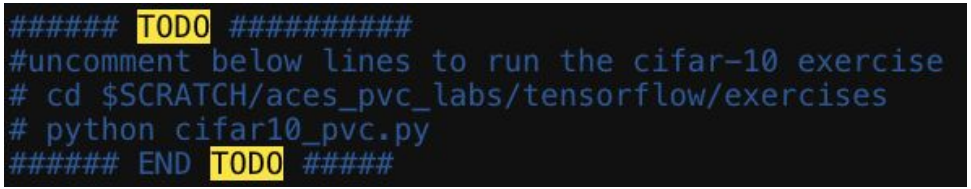
Hands-on Session

- Navigate to the pytorch exercises directory

  ```
  $ cd $SCRATCH/aces_pvc_course/tensorflow/exercises
  ```

- Open the exercise file (*cifar10_pvc.py*) with your preferred editor (e.g. vim) or the file editor of the OnDemand portal.
- Read through the code to verify that there are no code changes
- Modify the Slurm job file (*tf_cifar10_pvc.slurm*) and submit your job.

  ```
  $ sbatch tf_cifar10_pvc.slurm
  ```



PVC Monitoring Tools

- View the pvc nodes and number of GPUs

```
$ pestat -p pvc -G
```

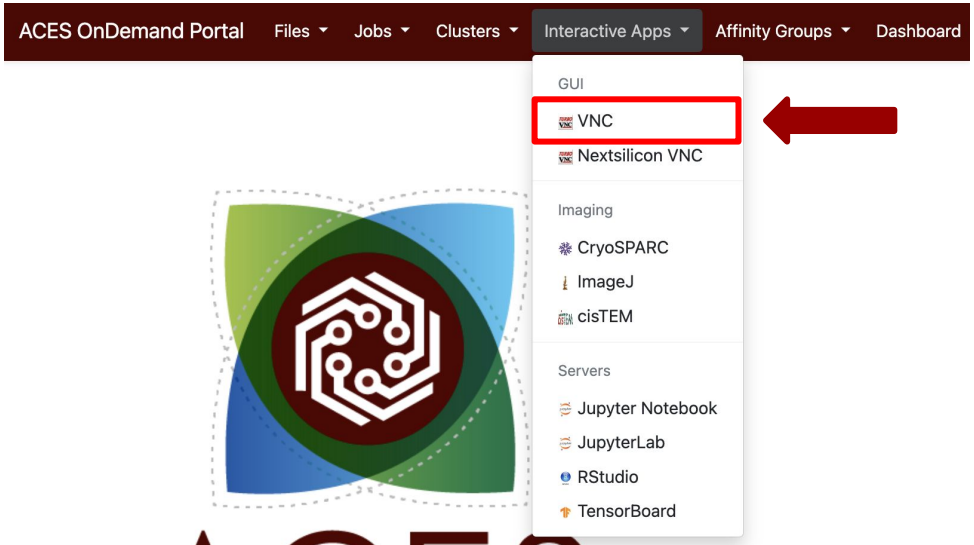
- Monitor the system activity

```
$ watch -n 5 sysmon
```

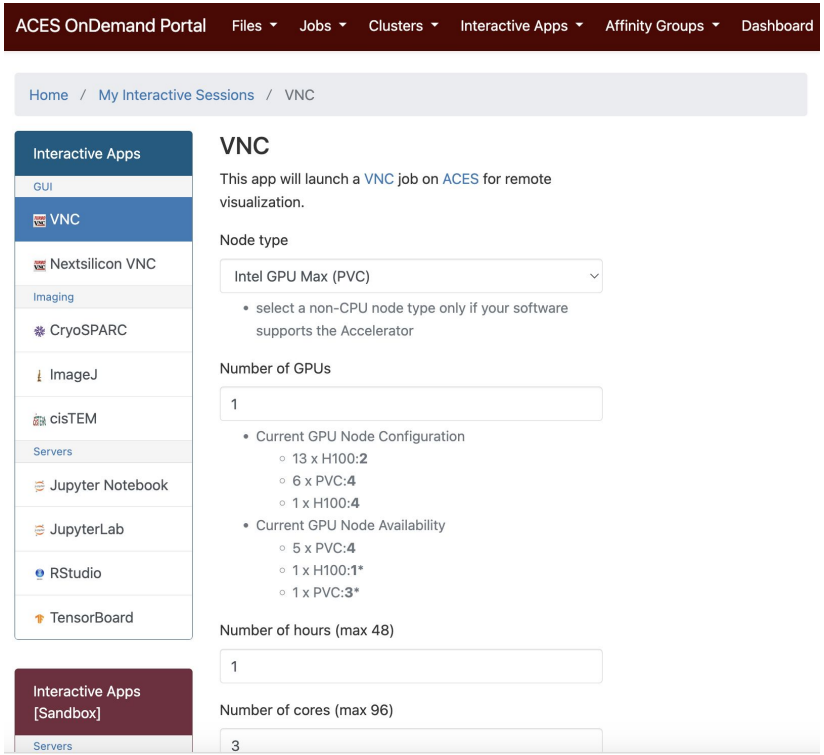
- Intel XPU manager

```
$ watch -n 5 xpumcli stats -d <device index>
```

Start a VNC job



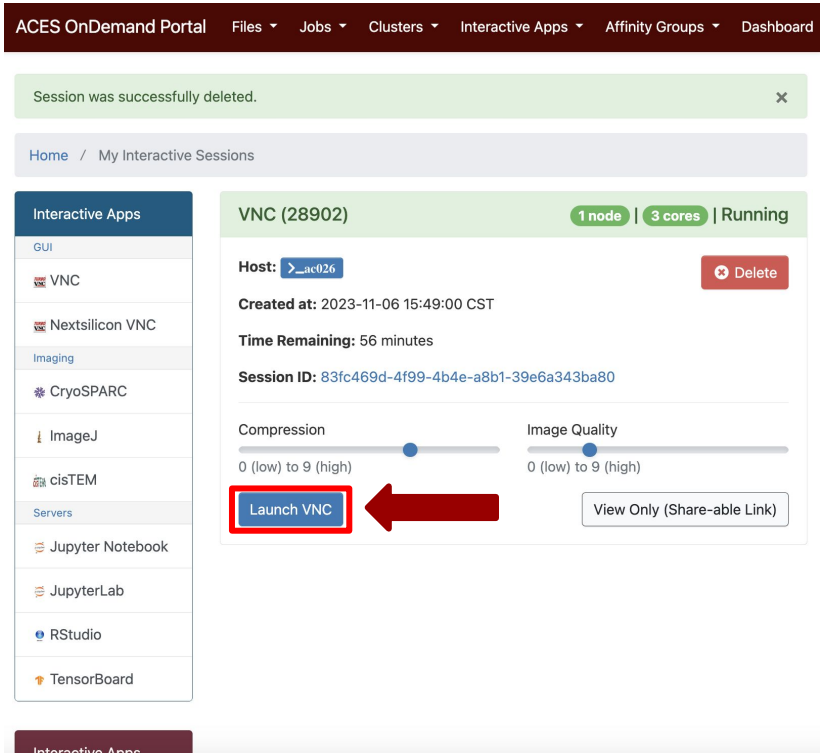
VNC Form

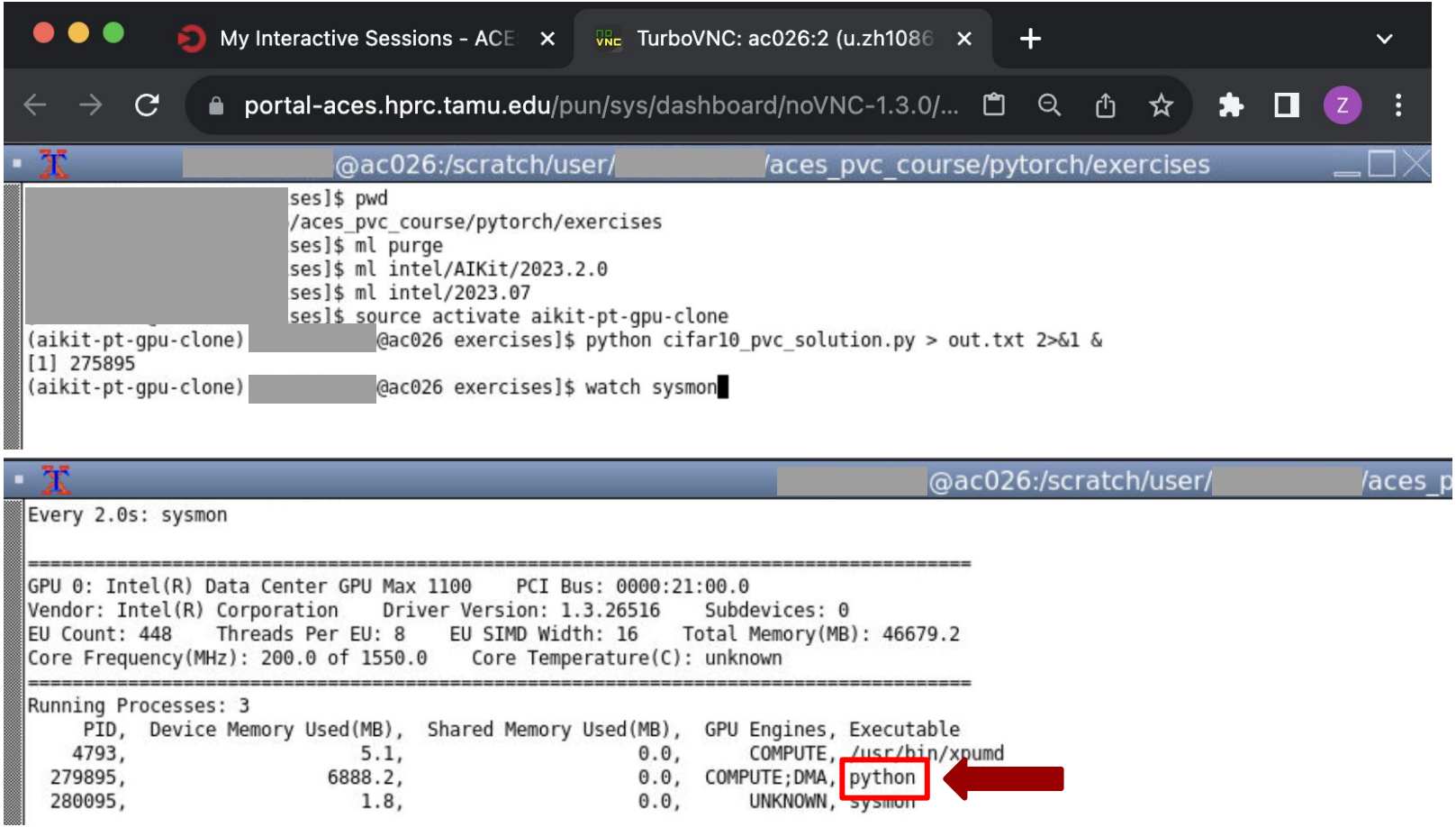


Fields:

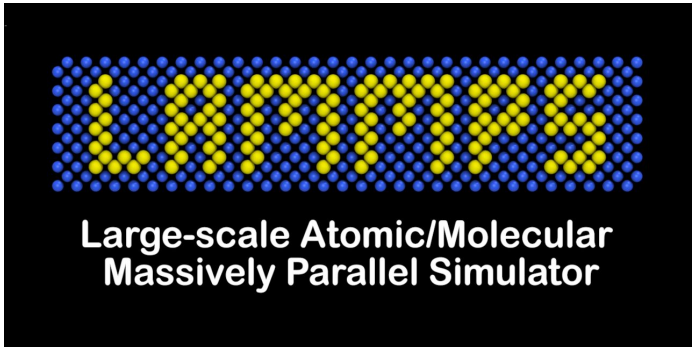
Node Type: Intel GPU Max (PVC)
Number of GPUs: 1
Number of hours: 1
Number of cores: 3
Total memory (GB): 5

Launch VNC


ACES OnDemand Portal
Files
Jobs
Clusters
Interactive Apps
Affinity Groups
Dashboard
Session was successfully deleted.
Home / My Interactive Sessions
Interactive Apps
GUI
VNC
Nextsilicon VNC
Imaging
CryoSPARC
ImageJ
cisTEM
Servers
Jupyter Notebook
JupyterLab
RStudio
TensorBoard
VNC (28902)
1 node | 3 cores | Running
Host: >_ac026
Delete
Created at: 2023-11-06 15:49:00 CST
Time Remaining: 56 minutes
Session ID: 83fc469d-4f99-4b4e-a8b1-39e6a343ba80
Compression
0 (low) to 9 (high)
Image Quality
0 (low) to 9 (high)
Launch VNC
View Only (Share-able Link)


My Interactive Sessions - ACE
TurboVNC: ac026:2 (u.zh1086
portal-aces.hprc.tamu.edu/pun/sys/dashboard/noVNC-1.3.0/...
@ac026:/scratch/user/ /aces_pvc_course/pytorch/exercises
ses]$ pwd
/aces_pvc_course/pytorch/exercises
ses]$ ml purge
ses]$ ml intel/AIKit/2023.2.0
ses]$ ml intel/2023.07
ses]$ source activate aikit-pt-gpu-clone
(aikit-pt-gpu-clone) @ac026 exercises]$ python cifar10_pvc_solution.py > out.txt 2>&1 &
[1] 275895
(aikit-pt-gpu-clone) @ac026 exercises]$ watch sysmon
@ac026:/scratch/user/ /aces_p
Every 2.0s: sysmon
GPU 0: Intel(R) Data Center GPU Max 1100 PCI Bus: 0000:21:00.0
Vendor: Intel(R) Corporation Driver Version: 1.3.26516 Subdevices: 0
EU Count: 448 Threads Per EU: 8 EU SIMD Width: 16 Total Memory(MB): 46679.2
Core Frequency(MHz): 200.0 of 1550.0 Core Temperature(C): unknown
Running Processes: 3
PID, Device Memory Used(MB), Shared Memory Used(MB), GPU Engines, Executable
4793, 5.1, 0.0, COMPUTE, /usr/bin/xpumd
279895, 6888.2, 0.0, COMPUTE;DMA, python
280095, 1.8, 0.0, UNKNOWN, sysmon

Lab V. Running LAMMPS on PVC GPUs



LAMMPS on GPUs

- LAMMPS has a *modular* back-end for GPU acceleration
  - GPU package
    - OpenCL and CUDA GPUs are supported
    - divides work among the GPU and the CPUs
  - Kokkos package
    - many more kinds of accelerators are supported
    - default strategy is “everything on the GPU”

This one today

LAMMPS on ACES

- Module `LAMMPS/3Aug2023-intel-2023.07`
- Requires toolchain module `intel/2023.07`
- uses GPU package
- Main executable is named `lmp_oneapi`
- Software located at `$HPRCROOTLAMMPS`

```
module load intel/2023.07 LAMMPS
echo $HPRCROOTLAMMPS
which lmp_oneapi
```

MPI on ACES

- Tip: don't mix `srun` and `mpirun` (or else performance very poor). Use `sbatch` with `mpirun` instead.
  - `sbatch filename`
  - `mpirun -np N`
- Suggested MPI thread configuration
  - `I_MPI_PIN_DOMAIN=auto:compact`
  - `I_MPI_FABRICS=shm`
  - `KMP_AFFINITY="granularity=core,scatter"`
  - `KMP_BLOCKTIME=1000`

Compute Aggregation Layer

- A tool for oneAPI Level Zero and OpenCL Applications.
- Helps to orchestrate efficient communication between MPI processes and GPU devices.
- `cal` provided with LAMMPS module.
- prepend an mpi launch with `calrun`

```
calrun mpirun -np N lmp_mpi
```

LAMMPS Testing

LAMMPS input files for common benchmarks

- in.intel.airebo
- in.intel.dpd
- in.intel.eam
- in.intel.lc
- in.intel.lj
- in.intel.rhodo
- in.intel.snap
- in.intel.sw
- in.intel.tersoff
- in.intel.water

```
cd $HPRCROOTLAMMPS/apps/TEST
```

LAMMPS Demo

- Exercise: try different input files.
- Exercise: try different resource combinations.
- Output files at `lammps_demo.<jobid>` and `demo-logs/<input>.log`

```
cd $SCRATCH/aces_pvc_course/lammps
cat lammps_demo.slurm
sbatch --reservation=training lammps_demo.slurm
```

Acknowledgement

This work was supported by the National Science Foundation (NSF)

- NSF award number 2112356 ACES - Accelerating Computing for Emerging Sciences,
- NSF award number 1925764 SWEETER - SouthWest Expertise in Expanding, Training, Education and Research,
- NSF award number 2019129 FASTER - Fostering Accelerated Scientific Transformations, Education, and Research,
- Dunni Aribuki from Intel,
- staff and students at Texas A&M High Performance Research Computing.

High Performance Research Computing

DIVISION OF RESEARCH

https://hprc.tamu.edu

HPRC Helpdesk:

help@hprc.tamu.edu
Phone: 979-845-0219

Help us help you. Please include details in your request for support, such as, Cluster (Faster, Grace, Terra, ViDaL), NetID (UserID), Job information (Job id(s), Location of your jobfile, input/output files, Application, Module(s) loaded, Error messages, etc), and Steps you have taken, so we can reproduce the problem.

Environment Setup for PyTorch models

Use Intel AI Analytics Toolkit

- Navigate to the pytorch directory

  ```
  $ cd pytorch
  ```

- Clean up the environment and load the intel and intel/AIKit/2023.2.0 modules

  ```
  $ module purge
  $ module load intel/2023.07
  $ module load intel/AIKit/2023.2.0
  ```

- List the pre-installed environments

  ```
  $ conda env list
  ```

Environment Setup for PyTorch models

Use Intel AI Analytics Toolkit (Cont'd)

- Create a clone of the environment needed (aikit-pt-gpu for pytorch)

  ```
  $ conda create -n aikit-pt-gpu-clone --clone aikit-pt-gpu
  ```

- Activate the environment

  ```
  $ source activate aikit-pt-gpu-clone
  ```

- Install tqdm in the environment for the progress

  ```
  $ conda install tqdm
  ```

- Deactivate the environment

  ```
  $ conda deactivate
  ```