Love at First Byte GPU Acceleration on Alpine

GPU-Accelerated Matrix Multiplication

Login to CURC systems using your credentials:

```
ssh <username>@login.rc.colorado.edu
```

Load the Alpine Slurm module:

```
module load slurm/alpine
```

View Alpine GPU resources and configurations:

```
sinfo --Format NodeList:30,Partition,Gres |grep gpu |grep -v
"mi100|a100"
```

Navigate to your projects directory:

```
cd /projects/$USER/
```

Git clone the training repository:

```
git clone
https://github.com/ResearchComputing/Intro_GPU_Acceleration.git
```

Make the scripts executable:

```
cd Intro_GPU_Acceleration
chmod u+x tf*
```

Start an interactive node on Alpine with one Nvidia A100 GPU:

```
sinteractive --time=01:00:00 --ntasks=20 --gres=gpu:1 --partition=aa100
--reservation=gpulove
```

Look at the software and compilers available on Alpine:

```
module avail
```

Activate the conda environment:

```
module load anaconda
conda activate /curc/sw/conda_env/tf-gpu-cuda11.2
```

Run the CPU-only script:

```
python tf.matmul-cpu.py
```

Run the GPU-accelerated script:

```
python tf.matmul-gpu.py
```

Discussion Questions (if time allows)

1. How could we speed up the matrix multiplication even further?
2. How would matrix size affect the observed speed-up?

Steps to Load PyTorch on AMD GPUs

```
#get on an AMD GPU node with one GPU and 5 CPU cores
sinteractive --time=01:00:00 --ntasks=5 --gres=gpu:1 --partition=aa100
--reservation=amdgpulove

#load modules
module load rocm
module load pytorch

#start python
python3
```

```
#from the python shell
>>> import torch
>>> torch.cuda.is_available()
True
>>> print(torch.__version__)
1.13.0+rocm5.2
```