Intro to GPU Acceleration on CURC Alpine

Self-Guided Hands-on Tutorial

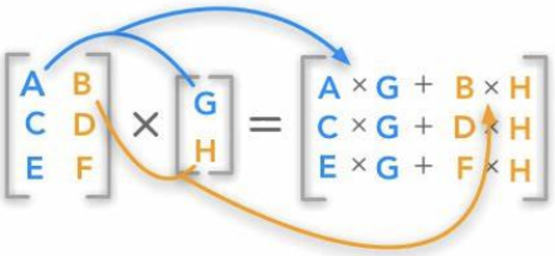
GPU-Accelerated Matrix Multiplication

- A matrix is a rectangular array of numbers arranged in rows and columns.
- The order of a matrix is the number of rows and columns.

$$\begin{bmatrix} 6 & 2 & 4 \\ 2 & 1 & 6 \end{bmatrix}$$

order = 2x3

Matrix multiplication is used in many ML algorithms.



We will use Tensorflow's `tf.matmul` function.

Login to CURC systems using your credentials. CU Boulder and CSU users can login directly from the terminal. RMACC and most AMC users will need to go through Open OnDemand.
https://curc.readthedocs.io/en/latest/access/rmacc.html#logging-in-to-open-ondemand

CU Boulder and CSU:

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

Navigate to your projects directory:

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

If you haven't already grabbed the repository, do a `git clone` :

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

Otherwise, run `git pull` from the GPU Acceleration repo.

Make the scripts executable:

```
cd /projects/$USER/Intro_GPU_Acceleration
chmod u+x tf*
```

Start an interactive job on Alpine with one NVIDIA A100 GPU and 20 CPU cores:

```
sinteractive --time=60:00 --ntasks=20 --gres=gpu:1 --
partition=aa100 --reservation=intro_gpu_nvidia
```

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
```

Look at the CPU-only script:

```
cat tf.matmul-cpu.py
```

Run the CPU-only script:

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

Look at the GPU-only script:

```
cat tf.matmul-gpu.py
```

Run the GPU-accelerated script:

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

Discussion Questions

Try running each of these scripts a few times. Why might the first GPU script run take longer than subsequent runs?

Try adjusting the size of the matrices. What do you notice about GPU speedups when the matrices are very small? What could be limiting the size of the matrices you are able to try right now?

How could we speed up the script even further?

Start 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=ami100 --reservation=intro_gpu_amd

#load modules
module load rocm/5.2.3
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
```