Introduction to GPUs on Compute Canada

Sabrina Berger and Christine Yue

WHAT EVEN IS A GRAPHICS PROCESSING UNIT (GPU)?

You have already probably used a graphics card on your computer already. Within the graphics card is a powerful processor for generating the images quickly on your screen–the Graphics Processing Unit or GPU. It was first popularized for video gamers as fast graphics make for an even better gameplay. Central Processing Units (CPUs) can also perform these computations to render images but they do so much more slowly. A single GPU has the power to break up a huge task into small bits and pieces that it can process more quickly than a CPU. GPUs, even by themselves, are the queens of parallel computing. Running code on GPUs has never been easier with software allowing simple flags to be changed to enable the GPU. Interacting with the GPU, however, presents many more programming challenges and can be done with applications such as Compute Unified Device Architectur (CUDA) and others. Try a quick Google search if you're curious about learning more about GPUs, but don't worry if they seem confusing. Regardless, their ability to accelerate machine learning computations is unparalleled.

1 GETTING AN ACCOUNT ON COMPUTE CANADA

You can make an account on Compute Canada at this link: https://ccdb.computecanada.ca/security/login after Adrian has given you access. You can choose to work with GPUs on Helios, Beluga (recommended as currently the best cluster available on Compute Canada), Cedar or Graham. We will cover using GPUs on Helios and Beluga. If you want to use Helios, you will also have to get an account with Calcul Quebec (See this link after you have your main Compute Canada account set up: https://www.calculquebec.ca/en/academic-research-services/procedures/) You can log onto Helios like this, remembering that your Calcul Quebec password may be different than your Compute Canada one:

```
# Helios
ssh <user_name>@helios.calculquebec.ca

# Beluga
ssh <user_name>@beluga.computecanada.ca
```

2 JOB SCRIPTS AND SETUP

As you do when submitting jobs using CPUs, you must also submit a job script to run on GPUs. Compute Canada uses the resource manager/job scheduler, Slurm Check out this link for more help getting started with Slurm: https://support.ceci-hpc.be/doc/_contents/QuickStart/SubmittingJobs/SlurmTutorial.html.

2.1 *Helios: Checking whether CUDA is available*

You will want to first find your RAP ID with:

```
helios-info
```

With that in hand, you will be able to request up to 8 GPUs/node for Helios K20 nodes and 16 GPUs/node on Helios K80 nodes.

2.2 *Helios: sample job script*

```
#!/bin/bash
#PBS -N
#PBS -A
#PBS -l
walltime=300
#PBS -l nodes=1:gpus=2
cd "${PBS_O_WORKDIR}"
python test.py
```

2.3 *Helios: sample script which checks to see whether CUDA is available*

```
import torch as t
CUDA = t.cuda.is_available() print(CUDA)
### This should error when you run it locally.


cd "${PBS_O_WORKDIR}"
```

makes sure to execute the file where the job is submitted. Hopefully, this script will return True after you submit it with:

```
sbatch test.sh
```

2.4 *Beluga: tensorflow-gpu / Keras*

Do you want to accelerate your machine learning training to less than half the amount of time it takes on one CPU (or more? I don't really know the scaling, but you get to find out!)? Well this is your section! After logging onto Beluga, you're going to want to load your favorite python version, e.g. Python 3.6:

```
module load python/3.6
```

Then, you're going to want to create a Python virtual environment that you can activate before you start your job. You'll probably want to create your environment in the PROJECT storage area (1TB of space/group) For example, my project folder path is like this: /project/def-acliu/sabrinab Create a virtual environment called test-env (do change this name when you make yours!):

```
virtualenv -no-download test-env
```

Activate your virtual environment:

```
source test-env/bin/activate
```

Install the GPU version of tensorflow: Be careful here! Are you using tensorflow 2.0 already? Wow, you are way ahead of the game if you do. I am not. So I have to specify the tensorflow version that I want. Beluga already has a bunch pre-configured and available so try installing the version you want.

```
pip install -no-index tensorflow_gpu==1.14.1
```

The no-index flag ensures that you are installing a package that was compiled by Compute Canada people and can potentially help with missing/conflicting dependencies. If you're using the Keras framework, you'll also have to install that with:

```
pip install -no-index keras == your version here
```

2.5 *Beluga: sample job script to run a GPU on Beluga*

```
#!/bin/bash
#SBATCH -gres=gpu:1 # request GPU "generic resource"
#SBATCH -cpus-per- task=6 # maximum CPU cores per GPU request: 6 on Cedar, 16 on Graham.
#SBATCH -mem=128G # memory per node (may change depending how much memory your script needs)
#SBATCH -time=0-1:00 # time (DD-HH:MM)

source /project/def-acliu/sabrinab/test-env/bin/activate
python /project/def- acliu/sabrinab/train.py
```

Try a sample job script submission like the one above to run train.py on a GPU. Surprisingly, easy! This saved us from having to learn CUDA and directly interact with the GPU. We will save that for next time... or maybe the next life.