Using TensorFlow on a GPU on the UCalgary Clusters

UCalgary clusters overview

The UCalgary has several computational clusters. The procedure to access and use them are the same in most cases. For our ENEL 645 class, there are two clusters that are of interest:

- Advanced Research Computing (ARC) cluster: it is meant for research purposes. If you are aligning your final project with your research, then it is probably fine to use this cluster. You need to request IT to create an account for you and copy your supervisor when contacting them. Your supervisor needs to be onboard with you creating an account at ARC. The ARC cluster is currently equipped with >20 Tesla V100 graphics processing units (GPUs)
- Teaching and Learning Cluster (TALC): it is meant to support courses and workshops. It was recently upgraded (February 2021) with new nodes and it is now equipped with 15 T4 GPUs. I am currently in contact with IT to potentially get accounts for all students in the class.

Before using the UCalgary clusters, it is good to look into their documentation: https://rcs.ucalgary.ca/RCS_Home_Page. The clusters are shared resources across students and researchers in the university. What you do there can affect others, so you need to be mindful of that.

Running TensorFlow GPU Jobs – step-by-step

Step 1. Making sure that you are at the UCalgary network

In order to connect to the clusters, you need to be in the UCalgary network. If you are working on campus, then you should be fine, you can go to the next step. If you are not on campus, then you need to connect to the UCalgary general virtual private network (VPN). Instructions can be found here: https://ucalgary.service-now.com/kb_view.do?sysparm_article=KB0033044. You will need to install the s FortiClient SSL VPN client software.

Step 2. Connecting to the cluster

Now that you are at the UCalgary network, we will connect to the cluster. Open a terminal window or command prompt (this works for Linux and Mac computers, but should be similar for Windows computers). To connect to the ARC cluster

ssh <user_name>@arc.ucalgary.ca (If connecting to the ARC cluster)

ssh <user_name>@talc.ucalgary.ca (If connecting to the TALC cluster)

The user name is the same as your UCalgary email address. You will be asked to type your password, which is also the same as your UCalgary password. If successful, you should get a message in the terminal saying that you are connected to the cluster. The cluster runs on a Linux system, so it is good to learn how to use the command line. This is an awesome and freely available book to help you with that: http://linuxcommand.org/tlcl.php

(Optional)

***

Step 3. Installing tensorflow-gpu in a virtual environment and activating it

You only need to install tensorflow-gpu once. You activate the environment every time that you want to use it.

There are three major steps to this:

(1) install miniconda

(2) create a virtual environment

(3) install tensorflow-gpu with a specified version

Taking them a step at a time:

(1) You can download miniconda by copying the link from https://docs.conda.io/en/latest/miniconda.html and using it in a curl or wget statement. I would do something like in the command line:

cd ~ # Goes to your home directory

mkdir software

cd software

curl -O https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh

bash Miniconda3-latest-Linux-x86_64.sh

Follow the instructions (choosing ~/software/miniconda3 as the directory to create), agree to the license, decline the offer to initialize.

This will take some time to unpack but probably not an egregious amount of time. Once it is done, you will have a working anaconda3 installation (without all of the baggage of the full distribution).

In order to use this Python you will need to run

export PATH=~/software/miniconda3/bin:$PATH

every time that you start a new terminal and want to use this python.

(2) You can create a virtual environment to perform your tensorflow installation in. The environment name should be descriptive. First you need to add the bin for miniconda to your path as I mentioned above and then which conda should refer to this conda. Once this is done, creating the environment can be done with a statement like:

conda create -n tf2gpu python=3

which will produce a new conda environment with a python 3 as its basis.

(3) Then you can build on this by installing tensorflow in this environment

conda install -c anaconda --name=tf2gpu tensorflow-gpu

this will trigger a lot of dependencies that need to be installed. Agree to them and it will install in this environment. You can always add more packages to this environment later but it is better to do this now if possible. Tensorflow has a vulgar number of dependencies and triggering an update of one of them later *could* break it. Anyway, this is done in precisely the same way as you installed tensorflow (although you can drop the channel option -c anaconda if this causes a problem). You can see the article I wrote and shared above for more details.

For you to use this package, there are two steps that must be done at the start of every interactive job or batch script:

(a) export PATH=~/software/miniconda3/bin:$PATH

(b) source activate tf2gpu

when you are done (in an interactive mode) you can use conda deactivate to exit the environment.

***

Step 4. Define the Python script that you want to run in the cluster

In the cluster, it is easier to rung Python scripts than running Jupyter Notebooks that require interactivity with the user. Define the Python script that you want to run in the cluster. Make sure that you copy any data that you may need to the cluster.

Step 5. Define the bash file **(Optional – only if using docker)**

SLURM is as job scheduling system (https://slurm.schedmd.com/documentation.html). You need to define it to specify what kind of resources your Python code requires from the cluster.

Step 6. Define the SLURM file

SLURM is as job scheduling system (https://slurm.schedmd.com/documentation.html). You need to define it to specify what kind of resources your Python code requires from the cluster.

Step 7. Submit and monitor your job

After creating your Python and SLURM files, it is time to submit the job to the cluster.

sbatch <file_name>.slurm

You can monitor your jobs using the squeue command:

squeue # prints all jobs submitted to the cluster

squeue –u <user name> #prints all your jobs submitted the cluster

(Optional) Transferring data to the cluster

https://rcs.ucalgary.ca/index.php/How_to_transfer_data