Lab_3 09/23

Agenda

- Task 1: Slurm Tutorial using command line tools or the job composer
- Task 2: Benchmark and Basic Evaluation Metrics
    - Use of timer
    - Script for Rivanna Slurm queues
    - Training vs. Inference Time
    - Cumulative Execution Time at Each Epoch
- Task 3: Benchmark the CNN model with MNIST using PyTorch or Tensorflow

Task 1: Slurm Tutorial

- What is Slurm?
    - Cluster and Job management ecosystem
    - Provides a series of interactive and batch tools based on a reservation request
    - The system ensures that users have confidence that they will have reserved access to system hardware, and to enforce limits(e.g. time length) on users to prevent abuse
- Why Slurm? Why not just use the interactive notebook session?
    - Both do the same jobs
    - Interactive sessions assume that you to be "interactive and active" all the time
- How to Use Slurm?
    - Job Composer
        - Step 1: Log in to Rivanna
        - Step 2: Upload scripts, notebooks, and data for your experiments to Home Directory
        - Step 3: Go to **Jobs -> Job Composer**
        - Step 4: Create a Slurm Script according to how you'd like to design your experiments (e.g. specify the parameters for GPUs desired, set up the experiments to be running in parallel)
        - Step 5: Upload the .slurm file to the **Job Composer** and Click **Submit** to push your experiments in the queue. Your experiments will automatically start and complete without you having to sit in front of your computer. Yayy!
        - Slurm scripts given below:

```
### Example_1

#!/usr/bin/env bash
#SBATCH --job-name=mydemojob
#SBATCH --output=%u-%j.out
#SBATCH --error=%u-%j.err
#SBATCH --partition=gpu
#SBATCH -c 1
#SBATCH --gres="gpu:v100:1"
#SBATCH --mem=4GB
#SBATCH --time=3:00
```

```
#SBATCH --account=ds7003-fall22

module load cuda cudnn # This line is critical for CUDA and Deep Learning
frameworks on rivanna. If you do not load these modules, it's very likely
you'll be using a CPU workloads unless you install your own version of GPU
libraries.

module load anaconda
conda create -y -n demo python=3.9
conda activate demo
# your automation code here...
```

```
### Example_2

#!/bin/bash
# This slurm script file runs


#SBATCH -p standard
#SBATCH --mem=40000
#SBATCH --time=5-00:00:00
#SBATCH --mail-type=begin
#SBATCH --mail-type=end
#SBATCH --mail-user=xxxx@virginia.edu
#SBATCH --ntasks=1

docn="xxx/xxx/xxx/xxx/xxx" #path to your directory

for k in $(cat $docn)
do
  # your automation and configuration code here...
done
```

- Rivanna Shell Access
  - Step 1: Log in to Rivanna_Shell
  - Step 1(alternative): Log in to Rivanna_Portal_Shell -> Click on "Clusters" then "Rivanna Shell Access" -> Try to run the below
  - Step 2: A Terminal/Command Line window will pop up
  - Step 3: Start writing commands in the command prompt window
  - Command Line Commans given below:

```
nvidia-smi # this command should fail
# you may need to conda init bash if you haven't run anaconda on rivanna
before
# Request a very small resource allocation (optionally you can use k80,
p100, or a100)
ijob -A ds7003-fall22 --gres=gpu:v100:1 --partition=gpu -c 1 --mem 8GB --
time=10:00
```

```
# Wait for allocation; you will see several salloc messages
nvidia-smi # this command should work
module load cuda cudnn anaconda
conda create -y -n rivanna-demo python=3.9 tensorflow-gpu
conda activate rivanna-demo
python
>>> import tensorflow as tf
>>> tf.config.list_physical_devices('GPU')
# you should see a lot of output and an array of the Physical Devices
namedtuple
>>> exit()
conda deactivate && conda env remove -y -n rivanna-demo
exit # this command will end your allocation request (automatically
happens at --time)
```

Task 2: Benchmark and Basic Evaluation Metrics

Few weeks back, we asked you to implement the following goals by adding in a timer to measure the execution cost, for simplicity, the execution cost here just refers the time used to run the experiments:

Record the execution time that each epoch uses under TF and PyTorch, as well as the total time used for the entire experiment. Perform Part a. on different GPUs(e.g. A100, V100), and record the execution time under different settings, Write down your thoughts on how to speed up the experiments, or the related strategies to optimize the experiment execution.

Rivanna uses a queue to manage all submitted jobs, and each GPU type (e.g. A100, V100) has an independent queue. This implies that to have resources allocated to your jobs as soon as possible, the best practice is to request other GPUs when the requested GPU type is not available. To request a specific type of GPU, use '--gres=' parameter in your Slurm script. For example, '--gres=gpu:v100:4' requests 4 V100 GPUs.

Benchmarking provides measurements of execution time, accuracy and efficiency of your system. However, in the last lab, many of you recorded the execution time in an excel sheet, which is not a standard benchmark in real practice. Thus, for this lab, **we are going to ask you to generate the measurement/metric plots within your Script/Notebook.** You may use either PyTorch or TensorFlow (based on your own preference) to run a the same CNN model with MNIST dataset, then generate the training history plot like the following within the Notebook/Script. Please refer to the following resource for calculating the elapsed time, source here.

Since `time.clock()` is deprecated as of Python 3.3, you will want to use `time.perf_counter()` for system-wide timing, or `time.process_time()` for process-wide timing, just the way you used to use `time.clock()`:

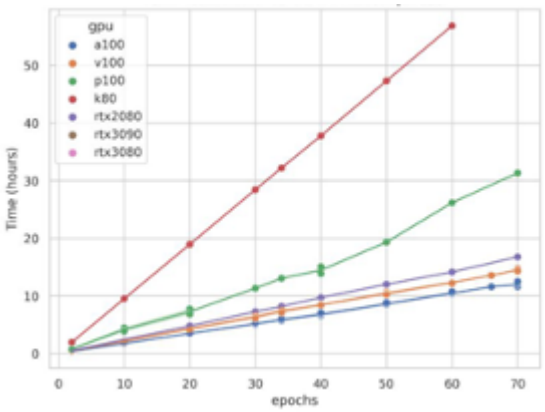
```
import time

t = time.process_time()
#do some stuff
elapsed_time = time.process_time() - t
```

Task 3(You may use either PyTorch or Tensorflow): Benchmark the CNN model with MNIST

Let's still use the CNN model from the previous labs. Now, you need to implement the following benchmarking method and generate a similar plot as shown below:

- Training-Time-Comparison Among Different GPUs (an example plot shown below)



- Pick a GPU of your choice, then generate breakdown measurements for I/O(Data Loading) Time VS Training Time VS. Inference Time.

Final Deliverables

- Make sure you submit your scripts/notebooks and plots via the Slack channel #labs.