Indian Institute of Science, Bengaluru

CSA

E0-294 Systems for ML

Spring 2023

Project #1 Machine Learning Case Studies,

Due: Feb 15th 2023

Overview:

This project asks you to use Pytorch and Pytorch profiling tools to practice model partitions and learn the characteristics of machine learning models in a real system.

GPU System:

In this project, you will need two GPUs in a single machine to perform certain tasks. When using Pytorch, you will use the "to" function to send data to different GPUs. The argument to the "to" function is "cuda:#", where "#" is the GPU number associated with a GPU (Please see link1, link2 and search to ".to" for detailed usage).

Starter Code:

Get your starter code here (proj1).

Task 1: Partitioning A Deep Model

In task1.py, there exists the implementation of a dummy model with many layers of Convolution and Fullly-Connected layers.

1. Use the "to" function such that they can run on a single GPU
2. Adjust num_of_fc and num_of_conv such that the dummy model will not fit inside a single GPU.
3. Partition the model such that some of layers are inside your first GPU, and some of layers are inside your second dedicated GPU.
4. Find a partition point where the execution time for each GPU is roughly equal.

Task 2:

Using the Linux command "watch -n 0.01 nvidia-smi" allows you to monitor GPU memory usage and power utilization in the real-time. Write your own python script to figure out GPU utilization for ReLU, MaxPooling2D, Convolution, and Linear layers. Configurations of those

layers should be the same as those inside VGG16 (you guys determine which layer to test, one for each is enough). VGG16 code is provided in vgg.py.

Task 3:

For VGG16 inference, Maxpooling2D and ReLU can be swapped (ReLU(Maxpool) -> Maxpool(ReLU)). Adjust the code in vgg.py and report if they have performance differences. Run the model **both on GPU and CPU**. If not, can you think of a scenario where this swapping can be beneficial?

Task 4:

Read the tutorial https://pytorch.org/tutorials/beginner/dist_overview.html and its related posts and learn how to run distributed training on your two GPUs. Of course, training data can be random tensors. Record runtime for each GPU and the communication overhead. Also, record speed-ups of distributed training w.r.t. a single GPU.

Communications and computation decomposition can be found using Pytorch Profiler.
https://pytorch.org/docs/stable/profiler.html
https://github.com/pytorch/kineto/tree/main/tb_plugin

Final Report:

Write a report about all the tasks, and submit the report alongside with your zipped source code.

Task 1:

The partition point, and the execution time of each partition.

Task 2:

Record GPU utilization for ReLU, MaxPooling2D, Convolution, and Linear layers.

Task 3:

Report runtime for the original VGG16 and VGG16 with Maxpooling & ReLU swapped

Task 4:

Record runtime for each GPU and the communication overhead