Parallelized Object Detection

Sneha Mhaske

Abstract

In this Project, we will be working with Object Detection - given a scene, we classify the objects in the scene and provide their bounding box co-ordinates as well. We are identifying and locating the objects in the scene. Object detection is a vital technology for ADAS (Advanced Driving Assistance Systems) and are used in video surveillance and other applications too. Since neural networks take a huge amount of time to train, we will be using the parallelized approach for object detection to speed up the entire training process making use of the GPUs made available by HPCC. Several state-of-the-art techniques are available for this task – Single Shot Multi-Box Detector (SSD), Region Based Convolutional Neural Networks (RCNN), You Only Look Once (YOLO). SSD will be used in this project.

Software

Python
PyTorch
Cuda – Pytorch natively supports Cuda with torch.cuda
MPI

Parallelization Strategies

Distributed Data Parallelism (DDP)–
It is a single-program multiple-data training paradigm in Pytorch where the input is split into batches and distributed to the allocated GPUs and the model is replicated across the GPUs. The gradients will be aggregated across GPUs from the backpropagation step. Under the hood, PyTorch also performs synchronization at critical points so that no GPU is left behind. As a result, at the end of each iteration all model replicas have the same updated parameters (same weights and biases).

DDP uses multi-processing i.e., creates a process for each GPU and is thus advantageous to use as compared to Data Parallelism which uses multi-threading instead and incurs additional overhead of Python's GIL (Global Interpreter Lock).

To start with, we will begin with a single GPU to train the model and then scale it to 2 and 4 GPUs on HPCC. Performance metrics will then be used to compare the serial vs parallel versions.

Approach

SSD is the algorithm that will be implemented for Object Detection. There are some existing SSD implementations available in PyTorch, but in this project, SSD will be implemented from scratch as a learning experience and made parallel using PyTorch's Distributed Data Parallelism (based on torch.distributed package)

Existing Standards

For SSD, the official GitHub repo provides benchmarking results on COCO dataset for 1,4 and 8 GPUs. These can be used as a reference to compare the SSD implemented in this project.

Verification

Verification will be done using common performance metrics like speedup, efficiency, execution time, weak and strong scaling, memory footprint and load balancing. Results will be presented in the form of graphs and tables.

References

1. Wei Liu, Dragomir Anguelov, et al. SSD: Single Shot MultiBox Detector
2. Shaoqing Ren, Kaiming He, et al. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks
3. Ross Girshick. Fast R-CNN
4. Ross Girshick, Jeff Donahue, et al. Rich feature hierarchies for accurate object detection and semantic segmentation