Distributed Deep Learning Based on AWS SageMaker

Xingyu Pan, Binfang Ye
{xp2009, by2034}@nyu.edu

May 8, 2022

1 Goal/Objective

In order to take advantage of distributed deep learning, we try to explore the performance Amazon SageMaker[Jos20] Distributed Training Libraries, one of distributed training libraries for data parallelism and model parallelism. Through optimized SageMaker training environment, we try to adapt our different deep learning jobs to SageMaker, and improve training speed and throughput. And we will make experiments on comparison between SageMaker and serverless computing machines to show the performance benefits of AWS SageMaker.

2 Challenges

The models (ResNet18[HZRS15], VGG[SZ14], and DenseNet[HLvdMW16]) will be deployed on New York University High-Performance Computer and Amazon SageMaker with the use of parallel techniques. The main challenges we encounter are not only the understanding of theoretical gradient updating and models training in distributed system, but the practical utilizing of multiple GPUs to speed up the model training. Hybrid parallel techniques, combining the data parallelism and model parallelism, will be adopted in our model training process. Some other challenges we are facing are that how to apply distributed parallel libraries to the Amazon SageMaker, how those libraries perform AllReduce.

3 Approach/Techniques

SageMaker provides distributed training libraries for data parallelism and model parallelism. The SageMaker distributed data parallel library extends SageMaker training capabilities on deep learning models with near-linear scaling efficiency, achieving fast time-to-train with minimal code changes. And for cloud-based SageMaker, distributed training is scaled to use a cluster of multiple nodes—not just multiple GPUs in a computing instance, but multiple instances with multiple GPUs. For model, parallelism, SageMaker makes model parallelism more accessible by providing automated model splitting and sophisticated pipeline execution scheduling. The model splitting algorithms can optimize for speed or memory consumption. While using the library, training is executed in a pipelined fashion over microbatches to maximize GPU usage. One example in Pytorch implementation, it takes advantage of Tensor parallelism, a type of model parallelism in which specific model weights, gradients, and optimizer states are split across devices.

4 Implementation details

We would like to choose cloud-based GPU servers deployed on AWS SageMaker and use AWS ES2 instances to run the model. In AWS, we can choose different instances, GPUs and cluster size, and we try to find the most optimal parameter. For NYU HPC, multiple GPUs will also be executed in our experiments. For the code, we would like to use Pytorch Framework and reuse the models(ResNet18, VGG, and DenseNet) in our experiments and compare the difference of performance and accuracy of different models. For the dataset, we will use CIFAR-10 and Fashion-MNIST to train.

5 Demo planned

We will show the performance comparison of three models (ResNet18, VGG, and DenseNet) under different hardware configurations. The performance comparison includes the use of time, bandwidth, speed up, training accuracy and so on. The parallel process of gradient updating and model training will be shown in the demo.

References

[HLvdMW16] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks, 2016.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[Jos20] Ameet V Joshi. Amazon's machine learning toolkit: Sagemaker. In *Machine Learning and Artificial Intelligence*, pages 233–243. Springer, 2020.

[SZ14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.