Benchmarking Horovod and Ray for Distributed DNN training on GPU cluster

Shreeparna Dey   Roopkatha Banerjee

April 28, 2022

Distributed Deep Learning

Need for Distributed Learning?

- Data set too big.
- Model size too big.
- Results need to produced in a reasonable amount of time.

Methods of Distributed Learning

- Model Parallelism - difficult to split model efficiently.
- Data Parallelism - limited by the all-reduce at the end of each iteration

Distributed Deep Learning Frameworks

Horovod

- Optimizes the inter-node communication$\rightarrow$ Data Parallelism
- Uses NCCL 2 $\rightarrow$ Provides semantics for inter-node ring-allreduce.
- No fault tolerance

Ray Train

- Facilitates inter-node communication with the use of a parameter server
- Parameter server collects and averages the gradients
- Allows fault tolerance

Experiments

Datasets

We have used three sets of data: MNIST, FMNIST and Data from the Sloan Digital Sky Survey.

MNIST Dataset

- image dataset of handwritten digits from 0 to 9, multiclass-classification problem with 10 class labels.
- 60000 training and 10000 testing data points
- one image which is in grayscale is 28x28

Fashion-MNIST Dataset

- image dataset of Zalando's article images, multiclass-classification problem with 10 class labels.
- 60000 training and 10000 testing data points
- one image which is in grayscale is 28x28

Sloan-Digital-Sky-Survey Dataset

- imaging and spectroscopic red-shift survey of the northern and southern hemispheres.

Models

- **Resnet50**: It is a DNN (Deep Neural Network) which is 50 layers deep where there is 48 Convolution layers and 1 Max- Pooling and 1 Average Pooling layer
- **MobileNetV2**: MobileNet-v2 by Google is a convolutional neural network that is 53 layers deep. It is a light-weight feature detector, which is suited to devices with low computational power

Hardware Setup

- Node 5 of the IOE cluster which houses two Titan RTXs with a total of 48GB GPU memory and two AMD EPYC processors with a total 32 threads and 256GB RAM
- Node 9 of the IOE cluster which house four V100 with a total of 128GB memory and an Intel Xeon processor with 32 threads and 192GB RAM. These cores have Non-volatile memory and have $2 \times$ 128GB Intel Optane memory installed.

Each of the nodes on the IOE cluster is running CentOS7, gcc version 9.3 and CUDA version 11.0. The cluster uses PBS job scheduler.

Results

Training Time

batch size = 32	RayTrain	Training time(Sec)		
	Node 5(2*TitanRTX)		Node 9(4*V100)	
	1	2	1	2
RN50+MNIST	4.0798	2.789	3.353	1.848
MN+F-MNIST	0.817	0.654	0.792	0.808
batch size = 32	Horovod	Training time(Sec)		
	Node 5(2*TitanRTX)		Node 9(4*V100)	
	1	2	1	2
RN50+MNIST	4.132	3.4474	3	3.7002
MN+F-MNIST	0.691	0.6409	0.614	0.936

Figure: Model Training Times of models on different datasets across RayTrain and Horovod frameworks with different number and variety of GPUs

Accuracy

batch size = 32	RayTrain	Testing Accuracy		
	Node 5(2*TitanRTX)		Node 9(4*V100)	
	1	2	1	2
RN50+MNIST	0.9883	0.9922	0.99	0.99
MN+F-MNIST	0.6951	0.47	0.5802	0.7498
batch size = 32	Horovod	Testing Accuracy		
	Node 5(2*TitanRTX)		Node 9(4*V100)	
	1	2	1	2
RN50+MNIST	90.2	90.4	90.3	90.3
MN+F-MNIST	72.9	72	72.2	71.8

Figure: Accuracies of models on different testing datasets across RayTrain and Horovod frameworks with different number and variety of GPUs

Images processed per Second

batch size = 32	RayTrain	Images/Sec		
	Node 5(2*TitanRTX)		Node 9(4*V100)	
	1	2	1	2
RN50+MNIST	14706	21513	17894	32467
MN+F-MNIST	73739	91743	75757	74257
batch size = 32	Horovod	Images/Sec		
	Node 5(2*TitanRTX)		Node 9(4*V100)	
	1	2	1	2
RN50+MNIST	14520	17404	17400	16215
MN+F-MNIST	86830	93618	93618	64102

Figure: Images processed per Second by models on different datasets across RayTrain and Horovod frameworks with different number and variety of GPUs

Conclusions, Challanges. and Future Work

- In out experiments, Ray Train show good speedups for multi-GPU training, especially when compared to Horovod.
- Caffe2 has been deprecated and merged into PyTorch
- We also tried a PyTorch distributed implementation, however, we could not debug it in time.
- We also tried to train big models like VGG16(~ 138 billion parameters) but it could not fit into the available GPU memory.

Thank you!