HPML Lab 4 – Distributed Deep Learning

Overview

Distributed deep learning is the de facto approach to training neural networks at scale. Synchronous SGD (SSGD) has been the most widely used training algorithms among all the learning algorithms. In this lab, we are going to experiment with PyTorch's DataParallel Module [1], which is PyTorch's SSGD implementation across a number of GPUs on the same server. In particular, We re-use lab 2 code with **default SGD solver and its hyper-parameter setup (e.g., learning rate and weight decay) and 2 num_workers IO processes**, running up to 4 GPUs with DataParallel Module.

What to hand-in (1) A report that answers all the questions. The suggested answer format is given below for each question. (2) Question 1 – Question 4.1 requires coding. Please attach your python script and a readme file which lists all the necessary commands to run for each experiment. (3) Question section 4.2 – Question section 7 are essay questions. You only need to provide your answers. If you happen to have code that corroborate your answers, you are welcome to hand in!

Machines to use You are welcome to use any GPU cluster that can run 4-GPU job, but do keep the type of GPUs you run consistent.

Question 1 Computational Efficiency w.r.t Batch Size (5pt)

Problem Description Measure how long does it take to compete 1 epoch training using different batch size on **single** GPU. Start from batch size 32, increase by 4-fold for each measurement (i.e., 32, 128, 512 ...) until single GPU memory cannot hold the batch size. For each run, run **2** epochs, the first epoch is used to warmup CPU/GPU cache; and you should report the training time (**excluding** data I/O; but **including** data movement from CPU to GPU, gradients calculation and weights update) based on the 2nd epoch training.

Expected Answers On *Model-X* GPU, it takes *x1* seconds to train one epoch (w/o data-loading) using mini-batch size 32, *x2* seconds using batch-size 128 ... When batch size is larger, it takes *longer or shorter* time to train, because ...

Question 2 Speedup Measurement (5pt)

Problem Description Measure running time with batch size **per GPU** you used in Question 1(i.e., 32, 128, ...) on 2 GPUs and 4 GPUs and calculate speedup for each setup. Again, for each setup, run **2** epochs, and only measure the 2nd epoch. When measuring speedup, one should include **all the training components** (e.g., data loading, cpu-gpu time, compute time)

Expected Answers Table 1 records the training time and speedup for different batch size up to 4 GPUs. *Comment* on which type of scaling we are measuring: weak-scaling or strong-scaling? *Comment* on if the other type scaling speedup number will be better or worse than what you we are measuring and give a data-point to corroborate your argument. (*hint*: you don't have to do any extra experiment or measurement to find this data-point.)

Question 3 Computation vs Communication (15pt)

3.1 How much time spent in computation and communication (5pt)

Problem Description Report for each batch size per gpu (i.e., 32, 128, 512 ...), how much time spent in computation (**including CPU-GPU transferring and calculation**) and how much time spent in communication in 2-GPU and 4-GPU case for one epoch. (*hint* You could use the training time reported in Question 1 to facilitate your calculation)

Expected Answers First, describe how do you get the compute and communication time in each setup. Second, list compute and communication time in Table 2.

	Batch-size 32 per GPU		Batch-size 128 per GPU		Batch-size 512 per GPU	
	Time(sec)	Speedup	Time (sec)	Speedup	Time (sec)	Speedup
1-GPU		1		1		1
2-GPU						
4-GPU						

Table 1: Speedup Measurement for different batch size.

	Batch-size 32 per GPU		Batch-size 128 per GPU		Batch-size 512 per GPU	
	Compute(sec)	Comm(sec)	Compute(sec)	Comm(sec)	Compute(sec)	Comm(sec)
2-GPU						
4-GPU						

Table 2: Compute and Communication time for different batch size.

3.2 Communication bandwidth utilization (10pt)

Problem Description Assume PyTorch DP implements the all-reduce algorithm as described in [2], calculate communication bandwidth utilization for each multi-gpu/batch-size-per-gpu setup.
Expected Answers First, list the formula to calculate how long does it take to finish an allreduce. Second, list the formula to calculate the bandwidth utilization. Third, list the calculated results in Table 3.

Question 4 Large Batch Training (10 pt)

4.1 Accuracy when using large batch

Problem Description Report the average training loss and training accuracy for the 5th epoch using the largest batch size per gpu you found in Question 1 with 4 GPUs and compare it with the training loss and training accuracy from Lab 2 (trained with batch size 128).
Expected Answers The average training loss and training accuracy for the 5th epoch for batch size per gpu x on 4 GPUs is x and x.

4.2 How to improve training accuracy when batch size is large

Problem Description By reading [3], come up with two remedies that can improve training accuracy when batch size is large.
Expected Answers Remedy 1 ... Remedy 2 ...

	Batch-size-per-GPU 32	Batch-size-per-GPU 128	Batch-size-per-GPU 512
	Bandwidth Utilization(GB/s)	Bandwidth Utilization(GB/s)	Bandwidth Utilization(GB/s)
2-GPU			
4-GPU			

Table 3: Communication Bandwidth Utilization

Question 5 Distributed Data Parallel (5pt)

Problem Description In order to run across *different* servers, PyTorch provides Distributed Data Parallel (DDP) [4]. One major difference between DP and DDP is that one needs to set up the epoch ID for data loader at the beginning of each epoch training in DDP whereas one doesn't need to specify the epoch ID in DP. By reading DDP document, comment on why one must set the epoch ID in DDP case.
Expected Answers One needs to set up epoch ID because ...

Question 6 What are passed on network ? (5pt)

Problem Description Are gradients the only message communicated across learners ? (*hint* C7 in Lab 2)
Expected Answers Yes, because ... or No, xxx are also communicated.

Question 7 What if we only communicate gradients ? (5pt)

Problem Description For the {batch size per GPU 512, 4-GPU} case, would it be sufficient to communicate only gradients across 4 GPUs? (*hint* read [3])
Expected Answers Yes, it is okay because ... or No, because ...

References

[1] PyTorch, *PyTorch Data Parallel*, Available at `https://pytorch.org/docs/stable/_modules/torch/nn/parallel/data_parallel.html`.

[2] P Patarasuk and X Yuan, "Bandwidth optimal all-reduce algorithms for clusters of workstations," *J. Parallel Distrib. Comput.*, vol. 69, pp. 117–124, 2009.

[3] P Goyal, P Dollár, R. B Girshick, P Noordhuis, L Wesolowski, A Kyrola, A Tulloch, Y Jia, and K He, "Accurate, large minibatch SGD: training imagenet in 1 hour," *CoRR*, vol. abs/1706.02677, 2017.

[4] PyTorch, *PyTorch Distributed Data Parallel*, Available at `https://pytorch.org/tutorials/intermediate/ddp_tutorial.html`.