Proposal: Exploring Distributed Deep Learning Training through Horovod

Andrew Tu

March 21, 2019

1 Applications: Horovod, TensorFlow and ResNet-50

TensorFlow: TensorFlow is one of the most widely used frameworks for building, training, and deploying deep learning networks. Open sourced by Google, the library offers multiple levels of abstractions for tackling a variety of deep learning methods. The TensorFlow community has tons of learning material to introduce beginners in machine learning and deep learning to the field. Additionally, many open-sourced state of the art algorithms are built on and discussed within the Tensorflow communities. For this project, I am interested in using TensorFlow to train ResNet-50 on the ImageNet data set to be used to classify images. I hope this project will serve as an introduction to deep learning and an exploration to scaling the training process.

Horovod: As the deep learning problems we are trying to solve continue to grow and grow, training these deep learning models on becomes incredibly expensive in terms of computation and time. While TensorFlow offers some built in support for multi-node/multi-gpu, benchmarks run by Uber show that the marginal speedup gained from adding more GPUs plateaus for larger numbers of GPUs. In order to improve performance of TensorFlow and be able to train their deep neural networks at scale, Uber built and open-sourced *Hodorvod*, a framework for distributed training of deep learning models.

2 Platform: multi-GPU nodes on Discovery

I will be using the multi-gpu nodes on Discovery to complete my experiments. Horovod should abstract away most of the hardware layer implementation details for scaling up the TensorFlow training modules. Updating the configurations should primarily be an update to the "Horovod"/MPI configuration files for the program.

3 Experiments

For experimentation, I will compare the amount of time it takes to train a ResNet-50 ImageNet Model through Tensorflow on various compute configurations using Horovod to achieve scalability in the training process. In particular, I plan to collect timing data for 8, 16, 32 and potentially 256 GPUs. In these experiments, all configurations are trained through to the end so we'd expect to see similar performance from the resulting weights. The major difference between the configurations is how quickly (wall time), did the model take to train.

If the training times for the neural nets take too long to run a reasonable number of experiments, I will instead change models to a fixed wall time run, stopping the training process epoch after the wall time elapses. Under different hardware configurations, I would expect more powerful configurations to reach further epochs and as a result, perform better in training.

4 Expected Results

1. **A**: Show an increase in training speed as the number of GPUs increase between configurations.
2. **A-**: Benchmark training on multi-node, multi-GPU system using Horovod. The expected training time for 32 GPUs is on the order of two hours.
3. **B+**: Benchmark training on single node, multi-GPU system system using Horovod. The expected training time for training on a single node, four GPUs is on the order of hours.
4. **B**: Get a ”hello world” Horovod program running to demonstrate successful installation of all dependencies

5 Expected Lessons

- Intro to TensorFlow and Deep Learning
- Working with Docker (for installation of Horovod)
- Breaking up Training of Deep Learning across a distributed system
- Working on a SLURM system

6 Useful references

- Amazon Horovod Tutorial
- Distributed TensorFlow using Horovod — Towards Data Science
- Horovod Github