World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:13, No:8, 2019

Performance Evaluation of Distributed Deep Learning Frameworks in Cloud Environment

Shuen-Tai Wang, Fang-An Kuo, Chau-Yi Chou, Yu-Bin Fang

***Abstract*—**2016 has become the year of the Artificial Intelligence explosion. AI technologies are getting more and more matured that most world well-known tech giants are making large investment to increase the capabilities in AI. Machine learning is the science of getting computers to act without being explicitly programmed, and deep learning is a subset of machine learning that uses deep neural network to train a machine to learn features directly from data. Deep learning realizes many machine learning applications which expand the field of AI. At the present time, deep learning frameworks have been widely deployed on servers for deep learning applications in both academia and industry. In training deep neural networks, there are many standard processes or algorithms, but the performance of different frameworks might be different. In this paper we evaluate the running performance of two state-of-the-art distributed deep learning frameworks that are running training calculation in parallel over multi GPU and multi nodes in our cloud environment. We evaluate the training performance of the frameworks with ResNet-50 convolutional neural network, and we analyze what factors that result in the performance among both distributed frameworks as well. Through the experimental analysis, we identify the overheads which could be further optimized. The main contribution is that the evaluation results provide further optimization directions in both performance tuning and algorithmic design.

***Keywords*—**Artificial Intelligence, machine learning, deep learning, convolutional neural networks

I. Introduction

With the development of machine learning, many machine learning techniques [1], [2] have been developed and are being updated to adapt to new software libraries, which bring a big challenge. Machine learning is a type of artificial intelligence that provides machines with the ability to learn without being explicitly programmed. The purpose of machine learning is to learn from variable data and create a suitable model by repeatedly optimizing, evaluating, and tuning parameters of the model. And with a large amount of input data, deep learning techniques [3]-[5] can learn the feature presentation very well. Deep learning is a recently developed field which is a new area in machine learning methods. Deep learning emulate to imitate the human brain by learning, analyzing and solving different kinds of complex problems. It is derived from the concept of artificial neural networks [6] and utilizes learning algorithms that are inspired by our understanding of how the brain learns. In addition, deep learning has a layered-based architecture that has been motivated by artificial intelligence. The output layers give the output that determine which input has been read.

Currently, one of the important challenges of deep learning is it is a very time-consuming process. Large scale of input data will result in a high requirement of computation resources. Reference [7] indicated the insight in deep learning which is possible to replace in modeling by using of large-scale datasets for training. With the advances in deep learning frameworks, this insight led to large scale deep learning deployments. Moreover, designing a deep learning model requires data space exploration of a large number of hyper-parameters and processing the data. Thus, accelerating the training process is critical for deep learning developments. Distributed deep learning is one of the necessary technologies in reducing training time. A single accelerator has limited computational resource to process a large scale neural network, so distributed training algorithms are proposed to solve this problem such as model parallelization and data parallelization [8].

For complex deep learning tasks, and especially for training deep neural networks, large scale deep learning platform is inevitably built as distributed processing. Several popular distributed deep learning frameworks including TensorFlow [9] and Horovod [10] have achieved not only high throughput in a single GPU, but also have well scalability across multiple GPUs and multiple nodes. In this paper, we compare these distributed frameworks for deep learning and benchmark their performance against the centralized frameworks. They are evaluated using the running performance on ResNet-50 [11] convolutional neural network with the ILSVRC2012 ImageNet dataset [12] over multi GPU and multi nodes in our cloud environment.

The rest of this paper is organized as follows. Section II introduces the distributed training frameworks. Section III gives descriptions of our experimental environment. We show our experimental results in Section IV, and Section V discusses future work and concludes.

II. Distributed Deep Learning Framework

Training a deep learning model can take a significant amount of time. Typically, millions of parameters need to be trained and a huge amount of data need to be stored. While there are many possible ways to solve this issue, one can either extract a smaller portion as a sample of the whole dataset or upgrade the computing device. However, these approaches may lead to accuracy loss, and upgrading the device may be costly. So, a new way to fix this issue is to use distributed deep learning techniques to reduce the training time.

S. T. Wang, F. A. Kao, C. Y. Chou and Y. B. Fang are with the National Center for High-Performance Computing, Taiwan, R.O.C. (e-mail: stwang@nchc.org.tw).

A. Distributed TensorFlow

TensorFlow is an open source deep learning library for research and production use. Distributed TensorFlow is one of the TensorFlow's API which is designed to be easy to use and provide good performances. There are two ways to update the gradient in Distributed TensorFlow, i.e., synchronous, and asynchronous. Synchronous uses Ring-AllReduce algorithm [13] and asynchronous uses parameter server architecture to send models to all of the workers. The usage of these two methods is different. For asynchronous, it is often used when there are many instances with low computing power. For synchronous, it was used by strong machines such as GPUs, high speed network between machines would be an important issue.

For evaluating the performance of Distributed TensorFlow, we use Mirrored Strategy in the benchmark. Reference [14] is the description of this strategy. It has to contain one or more parameter servers and workers when running Distributed TensorFlow benchmarks. The parameter server only needs to broadcast the updates. The worker coordinates model training, initializes the model, counts the number of training steps, monitors the session and saves and restores model checkpoints to recover from failures. One disadvantage of Distributed TensorFlow is that we have to manage the starting and stopping of servers, keep track of the ports of all servers, and starting and stopping those servers manually.

B. Horovod

Horovod is another approach to distributed deep learning framework by Uber. At first, they use the standard distributed TensorFlow technique, but soon after they realized that some problems were found:

- The documentation of Distributed TensorFlow did not state what code modification is needed, which could slow trainings.
- Uber lost half of their resources due to scaling problems.

In order to solve these issues, Uber developed a new component called Horovod, in their Michelangelo machine learning platform. Horovod adopted works from TensorFlow Ring-AllReduce algorithm and made few changes:

1. Horovod without having to configure TensorFlow versions. It was developed as a package which can help to cut down the time needed for installing.
2. Using the Ring-AllReduce of NVIDIA NCCL (Collective Communications Library) [15] to communicate between different machines. This model can support multiple GPUs.
3. Implemented a broadcast operation that enforces consistent initialization of the model on all workers.

We examined Horovod, as it has a simpler API and good performance on Nvidia GPUs. Horovod requires the installation of Open MPI and NCCL-2 libraries, and it only requires a few of changes to Distributed TensorFlow programs. It introduces the hvd objects that have to be initialized. The hvd object averages the gradients using allreduce or allgather. A specific GPU will be bound to the process using its local rank.

III. Experimental Environment

In this section, we will introduce Taiwania 2 - the testbed of experiments to compare the performance between different distributed deep learning frameworks at National Center for High Performance Computing (NCHC) [16]. As shown in Fig. 1, Taiwania 2 has achieved a ranking of 20th in the world in the November 2018 edition of the TOP500 List, the highest rating for a Taiwan-made supercomputer ever. It consists of 252 nodes, each of which contains two CPUs and eight of the most advanced GPUs. It takes 2016 NVIDIA Tesla V100 GPUs that deliver 9 petaFLOPS of superior performance. Its host architecture design is in line with international trends.

Fig. 1 Taiwania 2 Supercomputer

In addition to utilizing its cloud computing platform to offer fast computing capabilities, large storage space, and a secure network, it now provides more immediate and convenience services to industry and academia. The most important cloud platform service is TWCC (Taiwan Computing Cloud). TWCC runs on the strength of Taiwania 2; it was designed that this platform could expedite the developments of AI-related technologies and services. In the meanwhile, TWCC not only supports large numbers of nodes for high-speed parallel computing across nodes, but also employs the latest container virtualization techniques [17] for GPGPU (General Purpose Computing on GPU) service. Table I lists the detailed information of the hardware and software configurations.

TABLE I
Taiwania 2 Specification

CPU	Dual Intel Xeon Gold 6154(18C) 3.0GHz
Local Storage	240GB SATA3 SSD ; 4TB NVMe
Memory	768GB DD4-2666 RDIMM
GPU	8x NVIDIA Tesla SXM2 V100 w/ 32GB HBM2
Network	4x Mellanox IB EDR 100Gb single port HCA
Operating System	CentOS 7.5
Parallel File System	IBM Spectrum Scale (GPFS)

IV. Experimental Results and Analysis

This section presents the performance results of distributed training. The effective way to evaluate the running performance is to measure the time duration of an iteration that processes a batch of input data. We choose the ResNet-50 model running on the ILSVRC2012 ImageNet dataset. This deep model has its own characteristics to test the performance of frameworks. In order to avoid the file I/O overheads from hard disks, we run

three epochs, and the first epoch is not to calculate the average time of the iteration. Since the total number of images is up to one million, it is very time consuming to run all the tests in one epoch. So, we change the epoch size to make each experiment run about 64-128 batches in one epoch.

Beside the running time measured, we also quantify the impact of network speed for data communication during distributed training computation. To evaluate the network performance and detect possible bottlenecks, a number of comparisons could be done. The idea was to evaluate the performance of various protocols of the system as independently as possible. Fig. 2 focuses on evaluating the node-to-node network performance. We compared performance of Infiniband and 10 GbE (10 Gigabit Ethernet). Performance can be checked using utilities such as NetPIPE (Network Protocol Independent Performance Evaluator) [18]. NetPIPE benchmark is a protocol independent utility, designed to probe the full throughput characteristics between networked computers. For the evaluation process, two nodes are configured as a sender and a receiver. We present the performance of node-to-node communication in Fig. 2, the InfiniBand network is slightly better than 10 GbE network, which shows an average of 14 times faster than 10 GbE while the message size is 1000 Kbytes. This is as expected, the 10 GbE network performance is much lower compared to the InfiniBand network while running distributed training works on our platform.

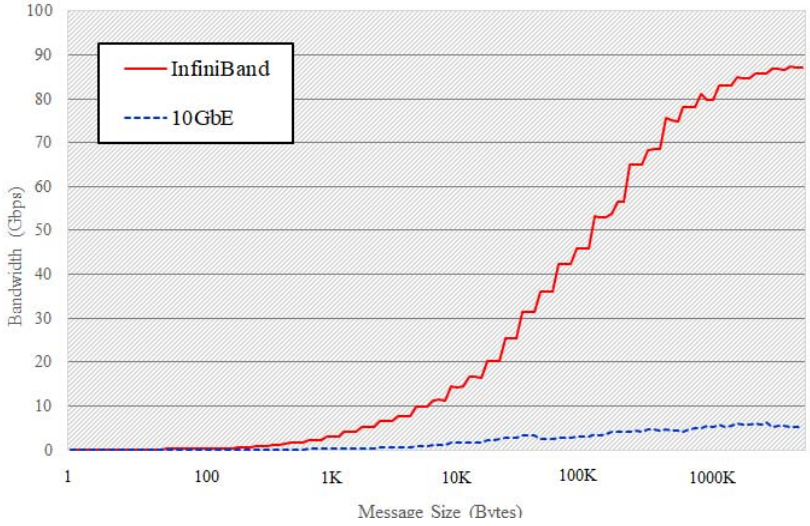

Fig. 2 Node-to-Node Network Bandwidth Comparison

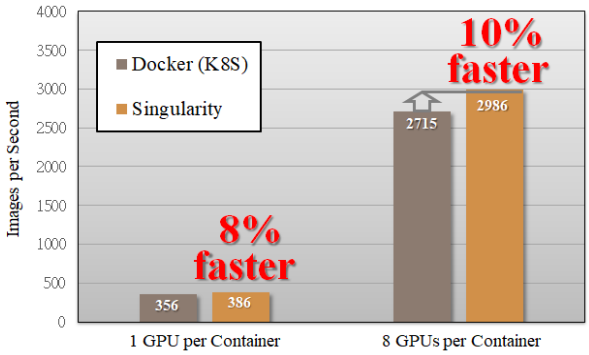

Fig. 3 ResNet-50 Training – Container Performance Comparison

We now employ Docker [19] and Singularity [20] container virtualization techniques for GPGPU and HPC (High-performance Computing) service. Docker basically extends LXC with a kernel level API and mainly focusing on network service virtualization. Singularity is another container-based approach which was created with the idea of compute mobility. Comparison is performed on TensorFlow CNN Benchmark [21], because there are reference implementations of a few of most popular models, picked models are used in a variety of tasks. All models are trained on the exact same data, where the same method of data loading and preprocessing is applied. Fig. 3 is the result times present mean epoch time between all runs and epochs for ResNet-50 model in both frameworks. The Singularity container runs roughly 10% faster than Docker either training with 1 GPU or 8 GPUs per container. This is because Singularity container seamlessly integrates with diverse HPC environments and tools, it does not support context change then I/O operations flow directly between environments where those operations are happening reducing the operation overhead and execution times.

We benchmarked Distributed TensorFlow and Horovod on 2 nodes against the convolutional neural network benchmark for the ResNet-50 model to see if the different batch size in one epoch for each network would lead to different results. In Fig. 4, we used an initial batch size of 128 as default. We then changed the batch size 256 to determine its effect on the performance. We ran the same experiment several times to ensure the consistency of the results. It shows the training performance of Horovod is 17% faster than Distributed TensorFlow. The main reason is the implementation overhead of communication protocol between workers. This can be explained by ResNet's high number of model parameters, caused by the use of fully

connected layers combined with its small number of layers. These characteristics shifted the critical path from GPU computation to communication and created a networking bottleneck.

The main premise of benchmarking distributed training is to scale out computation across many nodes in order to speed up I/O and to lower execution time and to see if the additional communication overhead would lead to different results.

Fig. 5 plots the results obtained using a ResNet-50 using up to 64 GPUs. With 64 GPUs, we achieved 89% scaling efficiency due to overheads over the InfiniBand network, but 65% scaling efficiency over 10b GbE. The efficiency of InfiniBand has achieved almost linear scaling from 1 to 64 GPUs, while 10 GbE has only a slight speedup. This is as depicted in Fig. 2 above, due to the performance of the InfiniBand network that is slightly better than 10GbE network. On other hand, the training speed over InfiniBand was about 37% as fast as over 10 GbE. This benchmark demonstrates that training over InfiniBand network scales well and experience a significant efficiency gain when using RDMA (Remote Direct Memory Access) protocol.

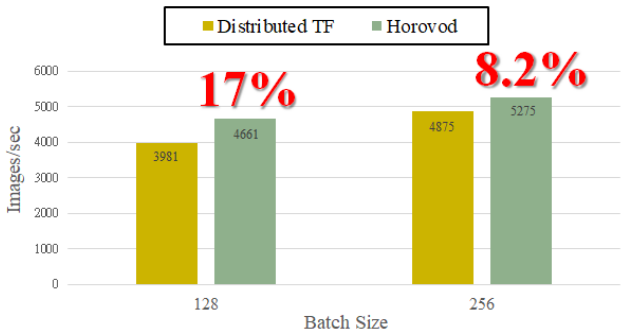

Fig. 4 Distributed DL Frameworks Comparison

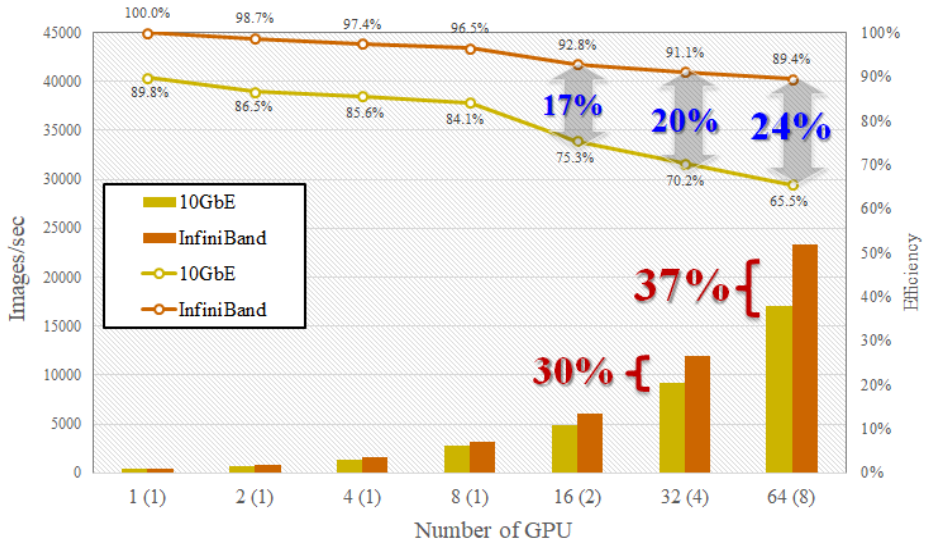

Fig. 5 ResNet-50 Training – InfiniBand vs. 10GbE

V. Conclusion

In this paper, we have presented works on such different benchmarks of distributed training for deep learning workloads over many GPUs using TensorFlow and Horovod in our cloud platform. According to the experimental results, we identify overheads and bottlenecks which could be further optimized. This also shows some performance gaps among the different training frameworks, and it exists many optimum methods that could be further optimized to improve the speed of data communication across intra-node and inter-node.

There is still a lack of benchmarks to adequately assess the performance of scaling out deep learning workloads. In the future, this evaluation can be further extended on heterogeneous environment for large neural networks to get the best acceleration. We plan to evaluate the scalability of deep learning frameworks across low-bandwidth or high-latency networks. We also plan to utilize AMD GPUs to further evaluate the performance of parallel training and inference.

References

[1] Witten, Ian H., et al. Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann, 2016.

[2] Kotsiantis, Sotiris B., I. Zaharakis, and P. Pintelas. "Supervised machine learning: A review of classification techniques." Emerging artificial intelligence applications in computer engineering 160 (2007): 3-24.

[3] LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." nature 521.7553 (2015): 436.

[4] Schmidhuber, Jürgen. "Deep learning in neural networks: An overview." Neural networks 61 (2015): 85-117.

[5] Abadi, Martin, et al. "Deep learning with differential privacy." Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2016.

[6] Yao, Xin. "Evolving artificial neural networks." Proceedings of the IEEE 87.9 (1999): 1423-1447.

[7] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." arXiv preprint arXiv:1502.03167 (2015).

[8] Lee, Seunghak, et al. "On model parallelization and scheduling strategies for distributed machine learning." Advances in neural information processing systems. 2014.

[9] Abadi, Martín, et al. "Tensorflow: A system for large-scale machine learning." 12th Symposium on Operating Systems Design and

Implementation, 2016.
[10] Sergeev, Alexander, and Mike Del Balso. "Horovod: fast and easy distributed deep learning in TensorFlow." arXiv preprint arXiv:1802.05799 (2018).
[11] Chilimbi, Trishul, et al. "Project adam: Building an efficient and scalable deep learning training system." 11th Symposium on Operating Systems Design and Implementation, 2014.
[12] Large Scale Visual Recognition Challenge 2012 (ILSVRC2012), http://www.image-net.org/challenges/LSVRC/2012/
[13] Hasanov, Khalid, and Alexey Lastovetsky. "Hierarchical redesign of classic MPI reduction algorithms." The Journal of Supercomputing 73.2 (2017): 713-725.
[14] Li, Shengbo Eben, Shaobing Xu, and Dongsuk Kum. "Efficient and accurate computation of model predictive control using pseudospectral discretization." Neurocomputing 177 (2016): 363-372.
[15] Potluri, Sreeram, et al. "Efficient inter-node MPI communication using GPUDirect RDMA for InfiniBand clusters with NVIDIA GPUs." 2013 42nd International Conference on Parallel Processing. IEEE, 2013.
[16] NCHC, National Center for High-performance Computing. http://www.nchc.org.tw/
[17] Soltesz, Stephen, et al. "Container-based Operating System Virtualization: a Scalable, High-performance Alternative to Hypervisors," ACM SIGOPS Operating Systems Review. Vol. 41. No. 3. ACM, 2007.
[18] Snell, Quinn O., Armin R. Mikler, and John L. Gustafson. "Netpipe: A network protocol independent performance evaluator." IASTED international conference on intelligent information management and systems. Vol. 6. 1996.
[19] Bernstein, David. "Containers and cloud: From lxc to docker to kubernetes." IEEE Cloud Computing 1.3 (2014): 81-84.
[20] Kurtzer, Gregory M., Vanessa Sochat, and Michael W. Bauer. "Singularity: Scientific containers for mobility of compute." PloS one 12.5 (2017): e0177459.
[21] Pena, Dexmont, et al. "Benchmarking of CNNs for low-cost, low-power robotics applications." RSS 2017 Workshop: New Frontier for Deep Learning in Robotics. 2017.