Evaluation of Volta-based DGX-1 System Using DNN Workloads

Saiful A. Mojumder[1], Marcia S Louis[1], Yifan Sun[2], Amir Kavyan Ziabari[3],
José L. Abellán[4], John Kim[5], David Kaeli[2], Ajay Joshi[1]
[1]ECE Department, Boston University; [2]ECE Department, Northeastern University; [3]Advanced Micro Devices;
[4]CS Department, UCAM; [5]School of EE, KAIST;
{msam, marcia93, joshi}@bu.edu, {yifansun, kaeli}@ece.neu.edu,
amirkavyan.ziabari@amd.com, jlabellan@ucam.edu, jjk12@kaist.edu

***Abstract*—Multi-GPU systems are being widely used to train deep neural networks (DNNs) as GPUs can significantly reduce the training time. Data parallelism is a popular choice to train DNNs on a multi-GPU system. GPUs in a multi-GPU system repeatedly perform Forward Propagation (FP), Backward Propagation (BP) and, Weight Update (WU) to train a DNN. During the WU stage, GPUs communicate with each other. To improve communication time, different data transfer mechanisms and libraries have been introduced by NVIDIA, and adopted by high-level frameworks to train DNNs. We evaluate the use of two of the most popular communication methods (peer-to-peer (P2P) data transfer and NCCL library-based communication) for training DNNs on a DGX-1 multi-GPU system. We profile and analyze the training of five DNNs using 1, 2, 4 and 8 GPUs. Our analyses provide insights into the software– and hardware–level limiting factors for training DNNs on a multi-GPU system.**

I. Introduction

From virtual personal assistants to video surveillance, we observe the dominant presence of machine learning in our day-to-day life. One of the popular categories of machine learning is deep learning (DL). DL is very popular because it can produce very high accuracy while solving complex problems such as image classification, object detection, natural language processing and speech recognition [1], [2], [5], [6].

Training a DNN on a multi-GPU system introduces new challenges including data distribution across GPUs, communication and synchronization among GPUs. The programmer can use data parallelism or model parallelism [7]. Both approaches require data to be transferred and synchronized across GPUs. We can parallelize the computation for training DNNs. However, the GPUs still need to communicate with each other during the different phases of training. Recent multi-GPU systems support different methods and libraries for GPU-to-GPU communication. We evaluate the effectiveness of the hardware– and software– level solutions to reduce the communication time.

We train five DNNs: GoogLeNet, AlexNet, Inception-v3, ResNet and LeNet, on NVIDIA's Volta-based DGX-1 system to identify the performance bottlenecks. These workloads have a wide variety in terms of computation and communication. We analyze the speedup in the training of various DNNs with respect to GPU count on the DGX-1 multi-GPU system. We identify hardware-level and software-level bottlenecks that exist in training of DNNs using multi-GPU systems. The hardware-level bottlenecks include computation, communication and memory capacity while the software-level bottlenecks include programming framework for DNN and NCCL library.

We analyze Forward Propagation (FP), Backward Propagation (BP), and Weight Update (WU) stages that repeatedly occur during the training of a DNN. We measure the time spent in computation– and communication–intensive stages. We also measure the time and memory requirement for training the DNNs for both the peer-to-peer (P2P) data transfer method and NCCL library based communication method. Based on our analysis, we identify the bottlenecks and provide guidance on how to improve the performance of each stage.

II. Multi-GPU DNN Training

Multi-GPU systems can accelerate the training of a DNN by parallelizing the training process. Figure 1 shows the timeline for training DNNs with a multi-GPU system consisting of 4 GPUs using data parallel synchronous stochastic gradient descent (SGD) algorithm. The training process starts with the CPU generating the network model parameters and transferring the parameters along with a unique mini-batch of data to all the GPUs. After receiving the data, GPUs perform FP and BP to calculate the local gradients. These local gradients are averaged and synchronized across multiple GPUs, typically using a tree reduction topology. Only one GPU (GPU0 in this example) uses the averaged gradients to update the weights (network parameters) and transfers the updated weights to all other GPUs, typically using a broadcast operation. This process continues for a specific number of epochs. [1]

III. DGX-1 Volta Multi-GPU System

We perform our evaluations on a NVIDIA's Volta-based DGX-1 system [4]. Figure 2 shows the network topology of the Volta-based DGX-1 system. This system has two 20-Core Intel Xeon E5-2698 v4 CPUs and each CPU is directly connected to four Tesla V100 GPUs using PCIe interconnect. It can be noticed that the GPUs are connected using a hybrid mesh cube topology. There is asymmetry in the NVLink distribution as each GPU can support a total of six NVLinks. This asymmetry can lead to a challenge to the programmer as the programmer has to be aware of the best routing path for GPU-to-GPU

[1]An epoch represents the processing of the entire dataset.

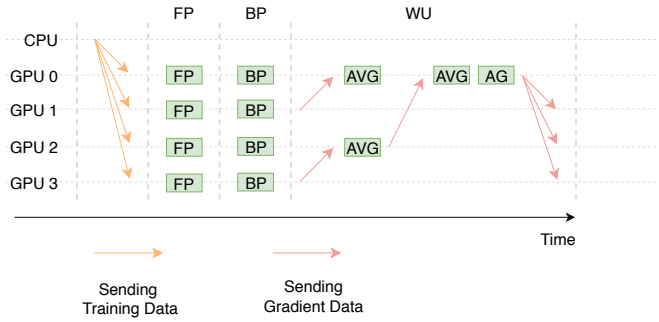

Fig. 1: The timeline of an epoch during multi-GPU DNN training using the data-parallelism approach with synchronous SGD. FP, BP, AVG, and AG represent forward propagation, backward propagation, averaging, and add gradients, respectively. (This figure is not drawn to scale.) [3]

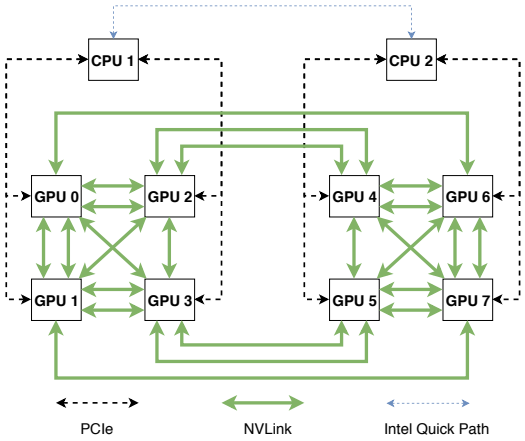

Fig. 2: Network Topology in a DGX-1 System. [3]

communication. This is because some paths have two NVLinks (50 GB/s total bandwidth) while some path have one NVLink (25 GB/s bandwidth) for GPU-to-GPU communication.

IV. Evaluation

A detailed discussion of various analyses that we performed when evaluating the DGX-1 system using DNN workloads is provided in our IISWC 2018 paper [3]. Here, we summarize those analyses. We compare the training time for 5 DNNs using P2P memcpy and NCCL communication method. Our evaluation based on training time provides the following insights:

- Increase in batch size results in linear decrease in per epoch training time for all the five workloads we have evaluated in this work. However, more efficient hardware and software support is needed to facilitate training with larger batch sizes.
- Increasing GPU count does not result in proportionate reduction in training time. Speedup in the training with an increase in GPU count depends on the computation-intensity of the workload and the communication method.
- Having more computation-intensive layers in a DNN workload leads to increased computation time. But we can reduce the training time by increasing the number of GPUs.
- For 4 and 8 GPUs, training time decreases significantly with NCCL if the DNN workload has a sufficiently large number of computation-intensive layers.
- Additional overhead associated with NCCL implementation (compared to P2P implementation) leads to increased training time for smaller networks and GPU count of 1 and 2. Hence, NCCL should be used only if the GPU count is 4 or more and the DNN has sufficient number of compute-intensive layers.

We determine the breakdown of training time in compute-intensive (FP+BP) and communication-intensive (WU) portions for training DNNs. Based on the breakdown, we obtain the following insights:

- As we increase the GPU count, time for FP+BP dominates the training time for all the 5 workloads.
- To reduce the training time, GPUs need to perform significantly more computation during FP+BP stages compared to the number of data transfers during WU stage. This can be achieved by increasing batch size and correspondingly, reducing the number of data transfers for a fixed dataset.

We also measure the GPU memory required to train the 5 different DNNs. Our memory analysis provide the following insights:

- Although increasing the batch size reduces the training time of DNNs for each epoch, the GPU memory capacity limits the maximum batch size that can be used for training DNN workloads.
- For large DNNs, the GPU memory required for intermediate outputs or feature maps is much more compared to the GPU memory required for the DNN model.

V. Conclusion

We performed a thorough analysis to understand the computation and communication pattern of training different DNN workloads on a Volta-based DGX-1 multi-GPU system. We evaluated two of the mostly used GPU-to-GPU communication methods (P2P memcpy and NCCL) in the context of deep learning. Our evaluation shows that multi-GPU scalability heavily depends on the neural network architecture, batch size, and the GPU-to-GPU communication method.

References

[1] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[2] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[3] S. A. Mojumder, M. S. Louis, Y. Sun, A. K. Ziabari, J. L. Abellán, J. Kim, D. Kaeli, and A. Joshi. Profiling dnn workloads on a volta-based dgx-1 system. In *Workload Characterization (IISWC), 2018 IEEE International Symposium on*, 2018.

[4] NVidia. NVIDIA DGX-1 With Tesla V100 System Architecture.

[5] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[6] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[7] O. Yadan, K. Adams, Y. Taigman, and M. Ranzato. Multi-gpu training of convnets. *arXiv preprint arXiv:1312.5853*, 2013.