Motivating In-Network Aggregation for Distributed Deep Neural Network Training

Liang Luo*, Ming Liu*, Jacob Nelson†
Luis Ceze*, Amar Phanishayee†, Arvind Krishnamurthy*
*University of Washington, †Microsoft Research

ABSTRACT

DNN training is moving from single node to cluster level with the aid of current DNN training frameworks[1, 2]. However, we find distributed DNN training time scales only sublinearly with the number of participating nodes in the system. Our experiments show this is due to frequent large-size parameter updates during parameter synchronization which is capped by the network capacity. This is especially relevant as model complexity is growing.

We present a comprehensive characterization and pinpoint the network as the bottleneck of DNN training today. We examine In-Network Aggregation (INA) in the context of distributed DNN training which targets directly at eliminating the bottleneck, and show the effectiveness of INA by simulating with existing hardware and DNN Framework, which promises a strong linear scaling of DNN training throughput.

1. INTRODUCTION

Modern neural networks are deeper and thus demand more computational power. Table 1 lists recent DNNs and their corresponding training time on a single machine.

Network	Layers	Training Time	Machine Config
AlexNet[3] 2012	9	5-6 days	2x GTX 580
ZFNet[4] 2013	8	12 days	GTX 580
VGGNet[5] 2014	19	2-3 weeks	4x Titan Black
RESNET[6] 2015	up to 269	2-3 weeks	8x GPUs

Figure 1: Recent DNNs and their sizes in layers, their reported training time and the machines they are trained on.

The ever increasing training time makes it no longer feasible to run large scale DNN training on a single machine. Parameter Server[7] enables distributed machine learning and recent deep machine learning frameworks such as TensorFlow and MXNET pushed deep learning to cluster and even datacenter levels.

One prevailing strategy of distributing DNN training popular DNN tasks such as image classification is data parallelism, where different samples in the training data are distributed among all workers, and each worker maintains a view of the state of DNN training. The unit of work queued to a worker is called a batch, and a batch consists of multiple samples. A batch generally undergoes forward, backward pass and parameter synchronization. Forward and backward passe performs inference and calculates gradients for neural network weight updates. Parameter synchronization generally happens at the end of each batch, where the different views of workers on the state of DNN training are synchronized. In this phase, workers send out the local gradients for aggregation at an abstraction known as a parameter server. The parameter server then aggregates and stores the updated weights that are used in subsequent batches. The organization of parameter servers can be central or sharded by keys.

2. NETWORK AS THE BOTTLENECK: A CASE FOR INA

We performed a system-level profiling study of the training process. We observe low CPU, memory and disk utilization, and constant near peak utilization for GPU and physical network. This hints that the bottleneck of distributed DNN training lies in GPU and physical network.

We now show the motivation and necessity for enabling In-network Aggregation for large scale distributed deep learning, by pointing out the bottleneck of distributed DNN training - the network, with 3 observations[1].

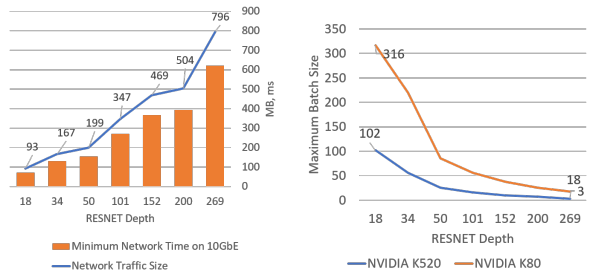
I. Deeper Neural Networks Shifts Training Bottleneck to Physical Network. Deeper neural networks contains more weights that need to be synchronized in a batch (Figure 2a) and potentially longer processing time for a fixed batch. On the other hand, given a fixed GPU, deeper neural networks result in a smaller maximum batch size which reduces the total workload per batch (Figure 2b). The reduction in batch size outplays the effect of longer processing time per fixed batch on the actual batch time, resulting in a shorter batch time with deeper neural networks, using maximum possible batch (Figure 2c). These results indicate the network needs to transmit more data within shorter amount of time, increasing the communication over computation ratio (Figure 2d). Consequently *for **any fixed** hardware, increasing the depth of the network (inevitable) shifts bottleneck from GPU to network.*

II. Network is a Bottleneck Today. We compare performance of distributed training of RESNET-269 with MXNET on varying number of machines with the performance of launching same number of training processes locally on a machine with 16 GPUs, and according to results reported in Figure 3, we observed (1) drastically decreasing GPU activity and increasing batch time (up to 2x) as more workers participate in training regardless of parameter server setups and (2) increasing gap (up to 100%) between batch time of training distributively over the network and on the same machine, which is the overhead due to network[2]. These results show that with today's hardware and neural network models, the network is already a bottleneck. Combining this with Observation I, we know *for **any fixed** reasonable **current** hardware and neural network configuration, network is a bottleneck.*

III. Network to Continue Bottleneck Distributed DNN Training. We now project into the future by observing the trend of GPU

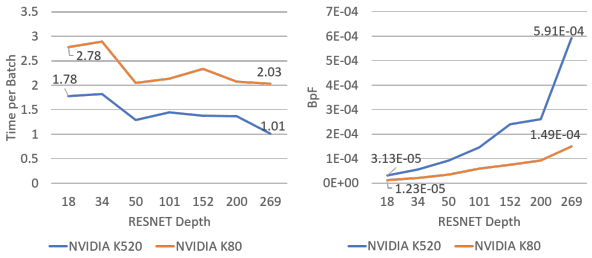
[1]While our observations are made in the context of MXNET and RESNET, they are generally applicable to other frameworks and neural network architectures.

[2]The surge from 8-16 workers in Figure 3b is due to the exhaustion of TCP processing capacity on the single machine.



(a) Size of communication and network time per batch increases as depth of neural network grows.

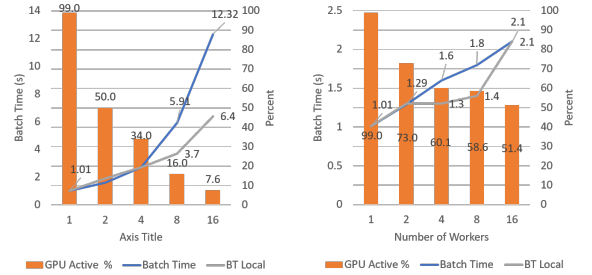
(b) Maximum batch size decreases as depth of neural network grows.



(c) Batch time with maximum batch size decreases as network depth grows.

(d) Bytes of network data per flop of computation increases as the depth of neural network depth.

Figure 2: Effect of neural network depth on communication size per batch, batch size, batch time and communication per unit of computation (Bytes per Flop). Observations made on two GPUS: NVIDIA K520 and NVIDIA K80.

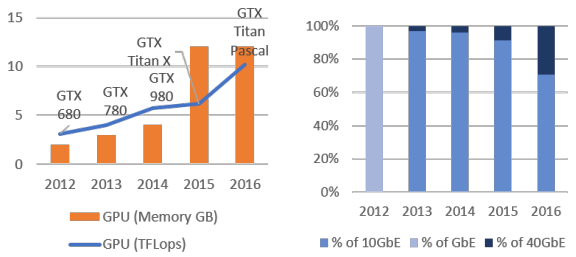


(a) With central parameter server, batch time increases as more workers are added in while GPU active time decreases. Workload: RESNET-269.

(b) With sharded parameter server, a better scalability is achieved, but the network overhead is not eliminated. Workload: RESNET-269.

Figure 3: Network bottlenecks DNN training in both central and sharded parameter setups. The gap between blue line (distributed training across network) and gray line (training with processes within one machine) shows the overhead due to network.

and physical network development (Figure 4) and show the observation we made holds true. On GPU side, both its computation power and memory capacity grew hand in hand. This implies Observation I continues to be true tomorrow, as the batch time, i.e., the frequency of parameter synchronization will not drastically increase. On the other hand, the adoption of faster 40GbE in datacenter (began in 2013) which is projected to be less than 30% by the end of 2016, is slow when compared to GPU development. The discrepancy between GPU and network development is only going



(a) GPU growth trend[8].

(b) Datacenter network adoption[9].

Figure 4: GPU growth outpaces the growth of network bandwidth.

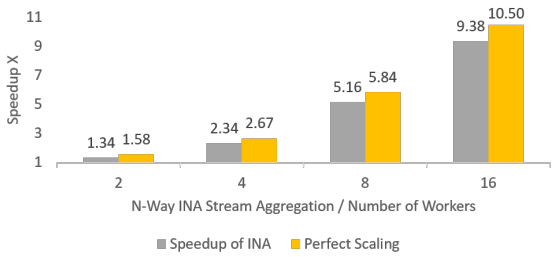


Figure 5: Simulated speedup of INA when compared to a central parameter server baseline: INA provides a near-optimal scaling.

to make the network bottleneck problem worse. Combining this with Observation I and II, we can strengthen our conclusion: *for **any** hardware today or in the future, with **existing or future** deep neural networks, the physical network is a bottleneck in distributed DNN training*.

3. IN-NETWORK AGGREGATION

INA is proved to be effective for long in many fields such as large sensor networks[10], graph processing and key value store[11], big data processing[12], as well as high performance computing[13]. Thus it is promising to re-examine it in the context of distributed DNN training. One form of INA for DNN is to leverage programmable network devices[14] such as switches: as update streams from different workers arrive, instead of simply forwarding them to the parameter server, the device can perform immediate aggregation of the streams from different ports in parallel and only the single aggregated copy of the stream is sent to the parameter server. INA targets directly at the network bottleneck by cutting total data movement by a factor of up to 2, reducing parameter server bandwidth requirement by a factor of number of workers. INA can also be deployed in many levels: by organizing INA-enabled devices into a reduction tree, as shown in FireCaffe[15], INA can provide further benefit of cross-device parallel parameter reduction.

Figure 5 provides a simulated effect of INA with N-Way stream aggregation within a rack of machines connected to a network device (switch), by launching 1 worker running RESNET-269, and varying the number of sharded parameter servers over the network, as only one copy of the stream is sent to the network and each parameter server can process one stream at a time. INA provides strong linear scaling when compared to the baseline of a central parameter server.

Conclusion. It is promising to apply INA in the context of distributed DNN training for network bottleneck elimination and linear performance scaling.

4. REFERENCES

[1] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, "Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems," *arXiv preprint arXiv:1512.01274*, 2015.

[2] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[4] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*, pp. 818–833, Springer, 2014.

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

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[7] M. Li, L. Zhou, Z. Yang, A. Li, F. Xia, D. G. Andersen, and A. Smola, "Parameter server for distributed machine learning," in *Big Learning NIPS Workshop*, vol. 6, p. 2, 2013.

[8] "Techpowerup gpu database." `https://www.techpowerup.com/gpudb`.

[9] "The market need for 40 gigabit ethernet." `http://www.cisco.com/c/en/us/products/collateral/switches/catalyst-6500-series-switches/white_paper_c11-696667.html`.

[10] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi, "In-network aggregation techniques for wireless sensor networks: a survey," *IEEE Wireless Communications*, vol. 14, no. 2, 2007.

[11] V. T. Lee, J. Nelson, M. Oskin, and L. Ceze, "A 10g netfpga prototype for in-network aggregation," in *Workshop on Architectural Research Prototyping (WARP w/ISCA)*, 2015.

[12] P. Costa, A. Donnelly, A. Rowstron, and G. O'Shea, "Camdoop: Exploiting in-network aggregation for big data applications," in *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pp. 3–3, USENIX Association, 2012.

[13] T. Eicken, D. E. Culler, S. C. Goldstein, and K. E. Schauser, "Active messages: a mechanism for integrated communication and computation," in *Computer Architecture, 1992. Proceedings., The 19th Annual International Symposium on*, pp. 256–266, IEEE, 1992.

[14] "Mellanox hpc-x framework extends smart in-network computing." `http://ir.mellanox.com/releasedetail.cfm?ReleaseID=976238`.

[15] F. N. Iandola, K. Ashraf, M. W. Moskewicz, and K. Keutzer, "Firecaffe: near-linear acceleration of deep neural network training on compute clusters," *arXiv preprint arXiv:1511.00175*, 2015.