ARL-TR-8580 ● Nov 2018

US Army Research Laboratory

Scaling to Multiple Graphics Processing Units (GPUs) in TensorFlow

by Song J Park

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-TR-8580 ● Nov 2018

US Army Research Laboratory

Scaling to Multiple Graphics Processing Units (GPUs) in TensorFlow

by Song J Park
Computational and Information Sciences Directorate, ARL

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
November 2018	Technical Report	1 February–12 October 2018

4. TITLE AND SUBTITLE
Scaling to Multiple Graphics Processing Units (GPUs) in TensorFlow

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Song J Park

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
US Army Research Laboratory
ATTN: RDRL-CIH-S
Adelphi, MD 20783-1138

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-8580

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Although accuracies of neural networks are surpassing human performance, training a deep neural network is a time-consuming task due to its increasing high-dimensional parameters. It is not uncommon for the training of deep neural networks to run for a week. Accordingly, the size of neural networks has doubled every 2.4 years, exhibiting an exponential growth from 1958 to 2014. The increasing size of neural network architectures will likely lead to higher computational complexity that will need scalable solutions. To mitigate the computational requirement and maximize throughput, this work focuses on multi-graphics-processing-unit scalability.

15. SUBJECT TERMS
deep learning, iterative training, TensorFlow, GPU scalability, matrix multiply

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Song J Park
Unclassified	Unclassified	Unclassified	UU	16	19b. TELEPHONE NUMBER (Include area code) 410-278-5444

Contents

List of Figures

List of Tables

1. Introduction

The interest in artificial neural networks has peaked again in recent years. The popularity of neural networks has fluctuated since the introduction of perceptron by Rosenblatt.[1] Neural networks have undergone a transformation from the early days of perceptron to modern deep neural networks (DNNs). Success, however, was not achievable until the rise of big data and access to highly parallel computing power. In terms of input data, it is recommended that the size of training examples be at least 10 million[2] for DNNs. Hence, availability of labeled data can be a challenge. As for the neural network model complexity, the trend shows that starting from perceptron to GoogLeNet, the size of neural networks has doubled every 2.4 years, exhibiting an exponential growth from 1958 to 2014.[2] The increasing size of neural network architectures will likely lead to higher computational complexity that will need scalable solutions.

DNNs have demonstrated remarkable advances in the following fields: image recognition,[3–6] natural language processing,[7–9] object detection,[10,11] and reinforcement learning.[12,13] For instance, DNNs now outperform humans in the object recognition ImageNet competition, as human experts mislabel objects due to multiple objects in an image or difficulty in fine-grained species recognition.[14] Although accuracies of neural networks are surpassing human performance, training a DNN is a time-consuming task due to its increasing high-dimensional parameters. It is not uncommon for the training of DNNs to run for a week. Exploring deeper neural networks with higher complexity will require parallel execution and distributed scaling of DNNs' training algorithms.

Given the compute-intensive nature of DNNs, a powerful computing platform is ideal to mitigate the training time. The weighted sum of a neural network can be concisely represented as a matrix multiplication operation, which means that DNNs' core computations comprise multiply and add operations. Due to design choices behind graphics processing unit (GPU) architectures, a GPU processor will have a high number of raw processing units for executing multiply and add operations. Therefore, to maximize throughput, this work focuses on multi-GPU scalability.

TensorFlow[15] provided the software framework to perform a scalability analysis of neural networks that can target a multi-GPU system. The objective of this study was to identify the tradeoffs between portability and programmability in TensorFlow. The aim was to study the impact of the flexibility and portability of TensorFlow on programming complexity for multi-GPU implementations.

2. Software Programming Framework

TensorFlow is an open-source library for numerical computations using dataflow graphs originally developed by Google Brain.[15] The TensorFlow tool is widely used for machine learning and deep learning research. TensorFlow features include front-end Python application programming interface (API), auto-differentiation, and visualization. It is highly portable, supporting central processing units (CPUs), GPUs, tensor processing units (TPUs), and ARM architectures running Linux, Windows, iOS, and Android. A notable difference with TensorFlow is its declarative style of programming via dataflow graphs. In essence, TensorFlow separates the definition of computations from actual execution. As such, a source code will be composed of a description portion for a computational graph and a session portion for the execution of operations.

Parallel strategies for training a DNN include model parallelism and data parallelism. In model parallelism, a DNN model is divided and assigned to computational resources. This can free up memory requirements on each computing resource since only a fraction of a DNN model is assigned for each computing device. On the other hand, data parallelism replicates the whole DNN model on all available computing resources, and different batches of training data are assigned to each computing device, referred to as a "tower". In data parallelism, DNN weights need to be shared across devices, requiring a synchronization protocol. This report focuses on the data parallelism approach on multiple GPUs.

3. Hardware Computing Platforms

DNNs can be targeted for CPUs, GPUs, field-programmable gate arrays (FPGAs), and application-specific integrated circuits (ASICs). Commercial examples of the use cases of GPUs, FPGAs, and ASICs can be found at Amazon,[16] Microsoft,[17] and Google.[18] Multiple tradeoffs exist in this spectrum of silicon hardware alternatives. For a comparison and tradeoff, attributes of interest are limited to throughput, power, flexibility, and development time. Table 1 illustrates the attribute values for a TPU, FPGA, and GPU.

Table 1 Characteristics for TPU, FPGA, and GPU

Processor	Throughput	Power (W)	Flexibility	Development time
TPU	92 TOPs (INT8) [19]	75	Low	15 months
FPGA (VU9P)	21 TOPs (INT8) [20]	225	Medium	Several months
GPU (P100)	18.7 TFLOPs (FP16) [21]	250	High	<1 month

Note: TOP = tera operations per second; TFLOP = tera floating-point operations per second.

For throughput, TPU and FPGA values reflect the precision of 8-bit integer operations, whereas the GPU throughput conveys 16-bit floating-point operations. As expected, the ASIC solution is the winner for throughput performance and low power. However, ASICs require longer development time and, once designed, the circuitry is fixed for a specific purpose. FPGAs and GPUs have similar characteristics for throughput and power, but GPUs are simpler and faster to program, develop, and modify. To be adaptable as the landscape of deep learning evolves, the flexible option of GPUs was selected for a scalability study.

The latest development in the shift of Nvidia's compute element architecture is described here for background. Architectural changes to Nvidia's GPUs in the current product lineup are specifically designed to target mixed precision matrix multiplication, which is at the core of neural networks. Introduced in Volta architecture, Tensor Cores are added to Nvidia's multiprocessor to enhance artificial intelligence performance. With Tensor Cores, the theoretical throughput of mixed precision (FP16 and FP32) operations are boosted to 125 TFLOPs.[22] In terms of performance, programmability, and flexibility, GPU's position on this tradeoff space seems balanced and competitive.

4. Experimental Evaluation

Matrix multiplication is at the core of DNN's training computations. Therefore, matrix multiplications can be a simplified representation of a DNN's training operations to test multi-GPU scalability. TensorFlow's multi-GPU scalability was tested on the DGX-1 system consisting of eight Tesla P100 GPUs. TensorFlow will automatically attempt to execute on a GPU if one is available. Moreover, TensorFlow reserves all visible GPU memories at start even though only one GPU is used by default. This behavior was observed for a simple single GPU matrix multiply code. To be explicit in device allocation, TensorFlow allows specification of device with the "tf.device" function. This function illustrates the flexibility of targeting different processing architectures as well as low-level programming in TensorFlow as evidenced by verbose source code for the multi-GPU execution. In the multi-GPU implementation, matrix multiplication was assigned to each GPU then summed using a CPU. Matrix dimensions were on the order of 10,000 by 10,000. Execution times for the various number of GPUs are provided in Table 2, where batch sizes represent the workload assigned to each GPU.

Table 2 Execution time for matrix multiplication on multiple GPUs

Batch size	2	8	32	128	512
GPUs	(s)	(s)	(s)	(s)	(s)
1	4.2	14.1	53.9	213.7	851.8
2	2.8	7.9	28.0	107.9	427.4
4	2.6	5.2	15.1	55.2	215.8
8	2.5	3.0	8.8	30.3	112.1

As depicted in Fig. 1, larger batch sizes led to improved performance toward the linear speed-up. In reference to the factors against parallelism, the major contributing factor seemed to be the startup costs of transferring data to a GPU device, whereas interference and skew were minimal. At the batch size of 2, the low-compute intensity does not warrant the data transfer communication costs to the GPUs.

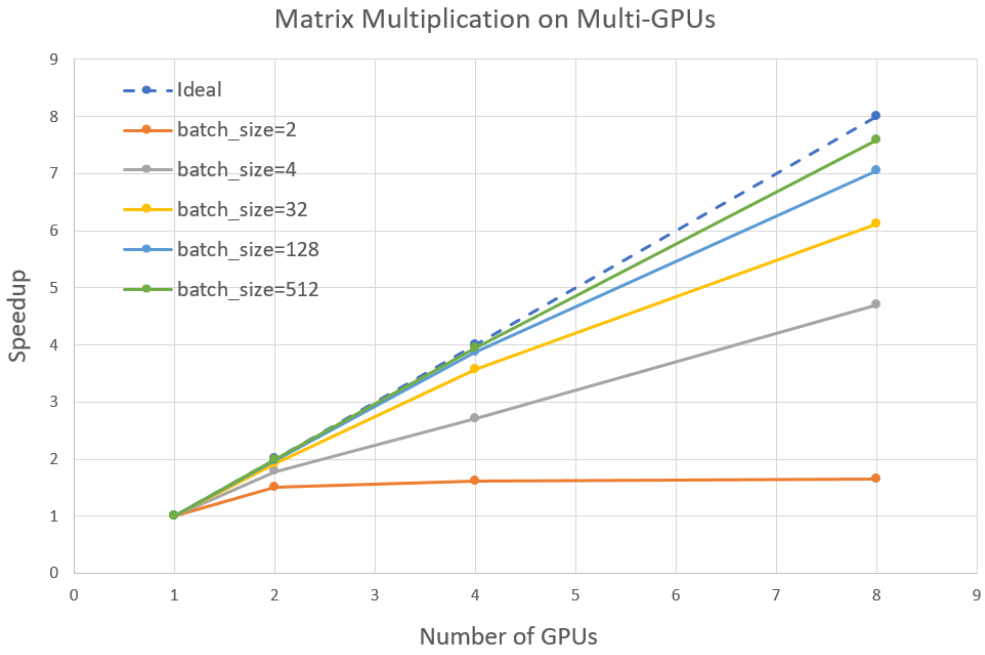

Fig. 1 Speed-up plot for matrix multiplication targeting multiple GPUs at different batch sizes

CIFAR10 source code was provided by the TensorFlow tutorial package.[23] The number of GPUs can be provided as an argument for running the CIFAR10 code. Execution time was measured to be 888, 561, 533, and 523 s for one, two, four, and eight GPUs, respectively. As indicated in Fig. 2, the speed-up curve flattens after two GPUs. Monitoring the Tesla GPUs via Nvidia tool revealed that the use of the GPU processors was approximately 80%, 65%, 33%, and 17% for one, two, four, and eight GPUs, respectively. These observations indicate that as the number of

GPUs increased, each GPU's utilization decreased, which resulted in a suboptimal speed-up. Similar suboptimal multi-GPU results for DNNs are reported in Gawande et al.[24] and Shi et al.[25] Underutilization of GPU cores can be the result of an insufficient input data pipeline. Future work is to explore efficient input pipeline support within TensorFlow.

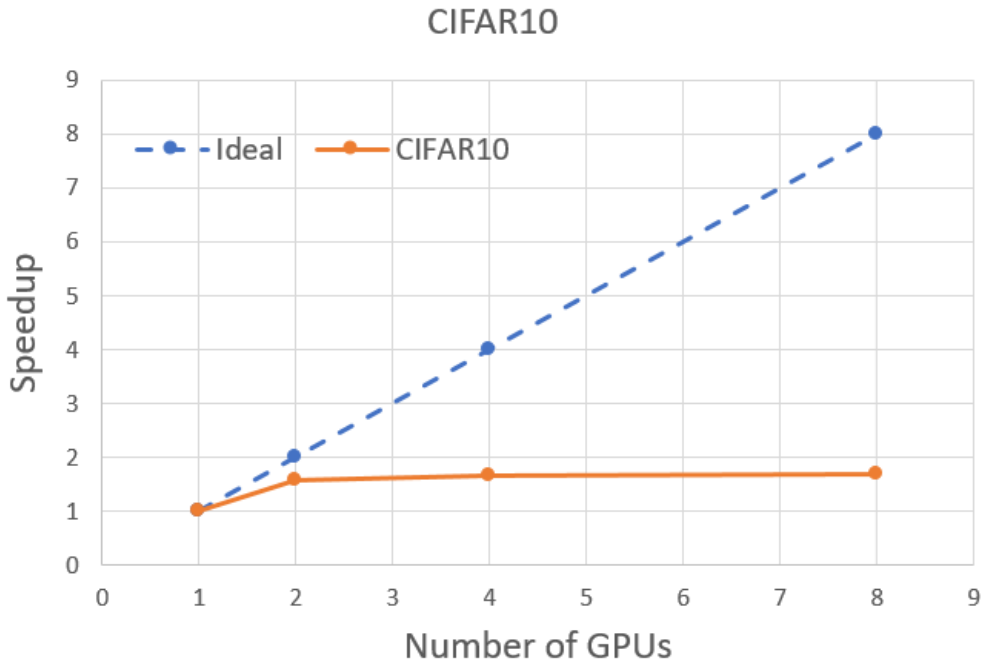

Fig. 2 Speed-up plot for CIFAR10 using multiple GPUs

5. Conclusion

The multi-GPU matrix multiplication exhibited almost linear speed-up, whereas the multi-GPU CIFAR10 showed barely any performance improvement after two GPUs. TensorFlow provides flexible, verbose, and low-level programming interface for scaling to multiple GPUs. However, multi-GPU programming in TensorFlow is an involved process, which can be abstracted by a high-level programming language. For instance, Keras is a high-level API that runs on top of TensorFlow and can implement a multi-GPU version with minimal code modification. The tradeoff between hardware portability and verbose programming model was experienced in TensorFlow. To target the broadest audience and applications, TensorFlow seems to sacrifice high-level programming approach for portability.

6. References

1. Rosenblatt F. The perceptron: a probabilistic model for information storage and organization in the brain. Psychological Review. 1958;65(6):386–408.

2. Goodfellow I, Bengio Y, Courville A. Deep learning. Cambridge (MA): MIT Press; 2016. p. 800.

3. Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. Communications of the ACM. 2012;60 (6):84–90.

4. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A. Going deeper with convolutions. Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2015 June 7–12; Boston, MA.

5. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Rrecognition (CVPR); 2016 June 26–July 1; Las Vegas, NV.

6. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556; 2014 Sep 4.

7. Graves A, Mohamed A-R, Hinton GE. Speech recognition with deep recurrent neural networks. Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing; 2013 May 26–31; Vancouver, Canada.

8. Seide F, Li G, Yu D. Conversational speech transcription using context-dependent deep neural networks. Redmond (WA): Microsoft Research; 2011 Aug [accessed 2018 Oct 11]. https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/CD-DNN-HMM-SWB-Interspeech2011-Pub.pdf.

9. Hinton G, Deng L, Yu D, Dahl G, Mohamed A-r, Jaitly N, Senior A, Vahhoucke V, Nguyen P, Kingsbury B, Sainath T. Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. IEEE Signal Processing Magazine. 2012;29(6):82–97.

10. Redmon J, Divvala S, Girshick R, Farhadi A. You only look once: unified, real-time object detection. Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2016 June 26–July 1; Las Vegas, NV.

11. Ren S, He Kaiming, Girshick R, Sun J. Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2017;39(6):1137–1149.

12. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller, Kidjeland AK, Ostrovski G, et al. Human-level control through deep reinforcement learning. Nature. 2015;518:529.

13. Finn C, Tan XY, Duan Y, Darrell T, Levine S, Abbeel P. Deep spatial autoencoders for visuomotor learning. Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA); 2016 May 16–21; Stockholm, Sweden.

14. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, et al. ImageNet large scale visual recognition challenge. Int J Comput Vision. 2015;115(3):211–252.

15. Abadi M, Baarham P, Chen J, Chen Z, Davis A, Dean J, Devin M, Ghemawat S, Irving G, Isard M, et al. TensorFlow: a system for large-scale machine learning. Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation; 2016 Nov 2–4; Savannah, GA. p. 265–283.

16. Amr Ragab CK, Huilgol R, Lee J, Mullenbach T, Wu Y. Scalable multi-node deep learning training using GPUs in the AWS Cloud. Seattle (WA): Amazon; 2018 July 20 [accessed 2018 Oct 11]. https://aws.amazon.com/blogs/machine-learning/scalable-multi-node-deep-learning-training-using-gpus-in-the-aws-cloud/.

17. Chung E, Fowers J, Ovtcharov K, Papamichael M, Caulfield A, Massengill T, Liu Ming, Lo D, Alkalay S, Haselman M, et al. Serving DNNs in real time at datacenter scale with project brainwave. IEEE Micro. 2018;38(2):8–20.

18. Jouppi NP, Young C, Patil N, Patterson D. A domain-specific architecture for deep neural networks. Communication of the ACM. 2018;61(9):50–59.

19. Jouppi NP, Young C, Patil N, Patterson D, Agrawal G, Bajwa R, Bates S, Bhatia S, Boden N, Borchers A, et al. In-datacenter performance analysis of a tensor processing unit. Proceedings of the 44th Annual International Symposium on Computer Architecture; 2017 June 24–28; Toronto, Canada. p. 1–12.

20. Xilinx Virtex UltraScale+ FPGA VCU1525 acceleration development kit. San Jose (CA): Xilinx; 2018 [accessed 2018 Sep 9]. https://www.xilinx.com/products/boards-and-kits/vcu1525-a.html#overview.

21. Nvidia Tesla P100. San Jose (CA): NVIDIA; 2018 [accessed 2018 Sep 22]. https://www.nvidia.com/en-us/data-center/tesla-p100/.

22. Durant L, Giroux O, Harris M, Stam N. Inside Volta: the world's most advanced data center GPU; 2017 [accessed 2018 Oct 11]. https://devblogs.nvidia.com/inside-volta/ 23.

24. Gawande NA, Landwehr JB, Daily JA, Tallent NR, Vishnu A, Kerbyson DJ. Scaling deep learning workloads: Nvidia DGX-1/Pascal and Intel Knights Landing. Proceedings of the 2017 IEEE International Parallel and Distributed Processing Symposium Workshops; 2017 May 29–June 2; Lake Buena Vista, FL. p. 399–408.

25. Shi S, Wang Q, Xu P, Chu X. Benchmarking state-of-the-art deep learning software tools. Proceedings of the 2016 7th International Conference on Cloud Computing and Big Data (CCBD); 2016 Nov 16–18; Macau, China.

List of Symbols, Abbreviations, and Acronyms

API	application programming interface
ASIC	application-specific integrated circuit
CPU	central processing unit
DNN	deep neural network
FLOP	floating-point operations per second
FPGA	field-programmable gate array
GPU	graphics processing unit
iOS	Apple mobile operating system
OPS	operations per second
TFLOP	teraflop
TOP	tera operations per second
TPU	tensor processing unit

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
2 (PDF)	DIR ARL IMAL HRA RECORDS MGMT RDRL DCL TECH LIB
1 (PDF)	GOVT PRINTG OFC A MALHOTRA
1 (PDF)	ARL RDRL CIH S S PARK