Extremely Accelerated Deep Learning: ResNet-50 Training in 70.4 Seconds

Akihiro Tabuchi, Akihiko Kasagi, Masafumi Yamazaki, Takumi Honda, Masahiro Miwa
Takashi Shiraishi, Motohiro Kosaki, Naoto Fukumoto, Tsuguchika Tabaru, Atsushi Ike
Kohta Nakashima
(tabuchi.akihiro,kasagi.akihiko,m.yamazaki,honda.takumi,masahiro.miwa
shiraishi-ten,kosaki.motohiro,fukumoto.naoto,tabaru,ike,nakashima.kouta)@fujitsu.com
Fujitsu laboratories ltd.
kanagawa, Japan

ABSTRACT

Distributed deep learning using a large mini-batch is a key technology to accelerate training in deep learning. However, it is difficult to achieve a high scalability and maintain validation accuracy in distributed learning on large clusters. We introduce two optimizations, reducing the computation time and overlapping the communication with the computation. By applying the techniques and using 2,048 GPUs, we achieved the world's fastest ResNet-50 training in MLPerf, which is a *de facto* standard DNN benchmark (as of July 2019).

1 INTRODUCTION

Deep neural network (DNN) models trained on large datasets are delivering impressive results in various fields, such as object detection and language translation. However, the computation cost becomes large with an increase in the sizes of DNN models and datasets.

Distributed deep learning based on data parallelism is known to be an effective approach to accelerate the training on clusters. In this approach, all processes launched on the cluster have the same DNN model. Each process repeats the training cycle of *Forward*, *Backward*, *Allreduce*, and *Update* with different mini-batches. Each process obtains own weight gradients after *Forward* and *Backward* and shares the gradients in *Allreduce*. The shared weight gradients are used in *Update* to improve the DNN model. In this paper, we focus on two optimization points, saving the additional computation cost and optimizing communication scheduling.

Deep neural network training generally uses the stochastic gradient descent (SGD) approach, and this approach requires many updates to train a DNN model. However, for a large number of processes, the number of updates for the SGD approach is not large enough because the mini-batch size, which is the total data size for each training cycle, increases. Thus, the validation accuracy tends to decrease compared with training by few processes.

To prevent poor validation accuracy, many researchers have proposed different techniques. Goyal et al. [3] proposed the warmup technique to keep the validation accuracy with a mini-batch size of 8,192. Because the difference between the weight gradient norm and the weight norm of each layer causes a relatively poor accuracy, the layer-wise adaptive rate scaling (LARS) of [8] normalizes the difference of each layer, and the DNN model can be trained with 32,768 without loss of validation accuracy. These techniques improve the validation accuracy, but some techniques require additional computations. We implemented these techniques on the GPU without significant additional cost.

We also optimized communication scheduling because the overhead becomes a significant problem for large clusters. The basic approach is to overlap the communication time with the computation time and improve the efficiency of communications. The communication efficiency is improved when the communication data size is large. Hence, communicating the data of several layers at once can improve the efficiency. However, in this approach, the communication time may overlap less with the computation time because of the computation dependency. We optimized the communication scheduling in order to obtain a better performance.

2 LARS ACCELERATION

The LARS technique adjusts the learning rate of each layer based on the L2-norms ratio between the current weight and weight gradient. This enables us to use a larger learning rate to sufficiently train the model even if the number of training cycles is small. However, compared with major computations of DNN, LARS calculation on the GPU is not optimized. The key idea of acceleration is to aggregate the LARS calculations for several layers. Since the element size of weights is small in most of the layers in convolutional neural networks such as ResNet-50, the computation of the L2-norm of each layer is not efficient for GPUs. Therefore, launching such small kernels many times causes a significant overhead. By aggregating the LARS calculations, we reduce the kernel launch overhead and exploit GPU parallelism.

3 VARIABLE GROUP ALLREDUCE

Allreduce operation per each layer leads to a large overhead due to frequent callings of communication operation. Hence, we distribute all layers of the model into several groups and perform *Allreduce* group by group. The straightforward approach is to distribute the layers so that the number of layers is the same for each group. However, this approach incurs significant non-overlapping time because each group does not perform *Allreduce* until every layer in the group completes the backward computation. The key idea of our approach is to change the number of layers in a group to efficiently overlap the communication with the backward computation. Our approach distributes few layers into the earlier group in order to reduce the non-overlapping time. Especially, the first group should perform the *Allreduce* early in order to overlap the

Table 1: Training time and Top-1 validation accuracy with ResNet-50 on ImageNet

	Batch Size	Processor	DL Library	Time	Accuracy	MLPerf Version
Facebook [3]	8,192	Tesla P100 × 256	Caffe2	1 hour	76.3 %	-
Preferred Networks [2]	32,768	Tesla P100 × 1,024	Chainer	15 mins	74.9 %	-
Tencent [4]	65,536	Tesla P40 × 2,048	TensorFlow	6.6 mins	75.8 %	-
Google [7]	65,536	TPU v3 × 1,024	TensorFlow	1.8 mins	75.2 %	0.5
Sony [5]	55,296	Tesla V100 × 3,456	NNL	2.0 mins	75.29 %	-
NVIDIA [1]	55,904	Tesla V100 × 1,536	MXNet	1.33 mins	76.97 %	0.6
Google [1]	32,768	TPU v3 × 1,024	TensorFlow	1.28 mins	76.28 %	0.6
This work	**86,016**	**Tesla V100 × 2,048**	**MXNet**	**1.17 mins**	**76.10%**	**0.6**

communication time with the backward computation time. In addition, we maintain a requirement that each group has more than a few megabytes of data in order to obtain a better communication throughput of *Allreduce*.

4 ENVIRONMENT AND RESULT

We used the AI Bridging Cloud Infrastructure (ABCI) to evaluate the performance of our optimized MXNet framework. Each node of the ABCI cluster consists of two CPUs of Xeon Gold 6148 and four GPUs of NVIDIA Tesla V100 SXM2. In addition, GPUs in one node are connected by NVLink, and nodes are connected by two InfiniBand connections. We used up to 512 nodes, or 2,048 GPUs.

We used a mixed precision method, where we computed and communicated using half precision floating point numbers and updated own weights using single precision floating point numbers. We used polynomial learning rate scheduling and also used warmup [3] and LARS [8] techniques to stabilize the training using large mini-batch. In addition, we used label smoothing [6] to improve the validation accuracy.

Our measurement of ResNet-50 training is according to the MLPerf 0.6 rule. This means that the training time does not include both the initialization time and memory allocation time. As shown in Table 1, in our optimized deep learning framework, the ResNet-50 training on ImageNet was completed in 70.4 seconds with 76.10% validation accuracy. We also measured the scalability of ResNet-50. Figure 1 shows the computational throughput of 86,016 mini-batch size according to the number of GPUs. The results of 256 or fewer GPUs could not be measured because GPUs do not have enough memory to allocate the data. The red one is our fastest configuration, which completes training in 70.4 seconds. Since the data size that must be communicated does not depend on the mini-batch size, the total communication times of the results increase with the number of GPUs. On the other hand, the computation times decrease as the number of GPUs increases because the batch size per GPU decreases. This scalability shows that our implementation can overlap the communication with the computation until 2,048 GPUs.

5 CONCLUSION

We optimized two points of the DNN training for large-scale GPU clusters. The first one is the LARS calculation, and we aggregate the calculations for several layers to improve the computation efficiency and reduce the kernel call overhead. The second one is

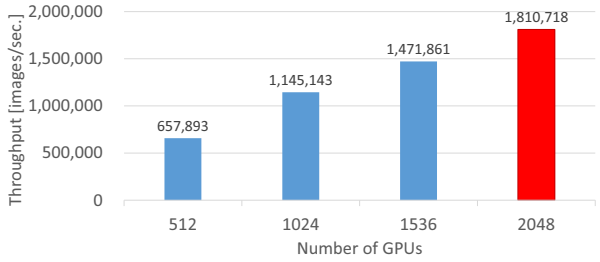

Figure 1: The throughput of our optimized framework is in images per second. The whole mini-batch size is fixed at 86,016, and the number of GPUs is changed from 512 to 2,048.

Allreduce scheduling, and we distribute layers into several groups so that each group efficiently overlaps with the backward computation. We measured the training time of ResNet-50 using 2,048 GPUs on ABCI cluster. The result of our DNN training achieves 76.10% validation accuracy in 70.4 seconds according to MLPerf 0.6 rule.

6 ACKNOWLEDGMENTS

We appreciate the National Institute of Advanced Industrial Science and Technology (AIST) and their support team for the stable use of ABCI on a large cluster.

REFERENCES

[1] [n. d.]. MLPerf. https://mlperf.org/.
[2] T. Akiba, S. Suzuki, and K. Fukuda. 2017. Extremely Large Minibatch SGD: Training ResNet-50 on ImageNet in 15 Minutes. *arXiv:1711.04325* (2017).
[3] P. Goyal, P. Dollar, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. 2017. Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour. *ArXiv:1706.02677* (2017).
[4] X. Jia, S. Song, W. He, Y. Wang, H. Rong, F. Zhou, L. Xie, Z. Guo, Y. Yang, L. Yu, T. Chen, G. Hu, S. Shi, and X. Chu. 2018. Highly Scalable Deep Learning Training System with Mixed–Precision: Training ImageNet in Four Minutes. *arXiv:1807.11205* (2018).
[5] H. Mikami, H. Suganuma, P. U-chupala, Y. Tanaka, and Y. Kageyama. 2019. Massively Distributed SGD: ImageNet/ResNet-50 Training in a Flash. *arXiv:1811.05233v2* (2019).
[6] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. 2015. Rethinking the Inception Architecture for Computer Vision. *arXiv:1512.00567v3* (2015).
[7] C. Ying, S. Kumar, D. Chen, T. Wang, and Y. Cheng. 2018. Image Classification at Supercomputer Scale. *arXiv:1811.06992v2* (2018).
[8] Y. You, I. Gitman, and B. Ginsburg. 2017. Large Batch Training Of Convolutional Networks. *arXiv:1708.03888* (2017).