
ImageNet/ResNet-50 Training in 224 Seconds

**Hiroaki Mikami, Hisahiro Suganuma, Pongsakorn U-chupala,
Yoshiki Tanaka and Yuichi Kageyama**

Sony Corporation

{Hiroaki.Mikami, Hisahiro.Suganuma, Pongsakorn.Uchupala,
Yoshiki.Tanaka, Yuichi.Kageyama}@sony.com

Abstract

Scaling the distributed deep learning to a massive GPU cluster level is challenging due to the instability of the large mini-batch training and the overhead of the gradient synchronization. We address the instability of the large mini-batch training with batch size control. We address the overhead of the gradient synchronization with 2D-Torus all-reduce. Specifically, 2D-Torus all-reduce arranges GPUs in a logical 2D grid and performs a series of collective operation in different orientations. These two techniques are implemented with Neural Network Libraries (NNL)¹. We have successfully trained ImageNet/ResNet-50 in 224 seconds without significant accuracy loss on ABCI² cluster.

1 Introduction

As the size of datasets and deep neural network (DNN) model for deep learning increase, the time required to train a model is also increasing. Large-scale distributed deep learning with data parallelism is an obvious course to effectively reduce the training time. However, there are two technical issues with large-scale distributed deep learning with a massive GPU cluster. The first issue is convergence accuracy degradation with large mini-batch training [1] [2]. The second issue is communication overhead of gradient synchronization among GPUs. A new approach to distributed processing is required to address these two issues.

In the past few years, many techniques have been proposed [1] [3] [4] [5] [6] to address these two issues. These works utilize ImageNet/ResNet-50 training to benchmark the training performance. ImageNet/ResNet-50 is one of the most popular datasets and DNN models for benchmarking large-scale distributed deep learning. Table 1 compares the training time and top-1 validation accuracy of the recent works. Among these works, 1-hour training with 256 Tesla P100 GPUs [1] is a well-known research to accelerate this task.

The instability of a large mini-batch training and the gradient synchronization overhead are the primary issues that we addressed. Our best effort reduces the training time to 224 seconds with the validation accuracy of 75.03% using 2176 Tesla V100 GPUs. We also attempt to improve GPU scaling efficiency without significant accuracy loss. We achieved the GPU scaling efficiency 91.62% with 1088 Tesla V100 GPUs (Table 2).

¹ An open source deep learning library, developed by Sony. <https://nnabla.org/>

² AI Bridging Cloud Infrastructure (ABCI) is the world's first large-scale Open AI Computing Infrastructure, constructed and operated by National Institute of Advanced Industrial Science and Technology (AIST). <https://abci.ai/>

Table 1 : Training time and top-1 1-crop validation accuracy with ImageNet/ResNet-50

	Batch Size	Processor	DL Library	Time	Accuracy
He et al. [7]	256	Tesla P100 x8	Caffe	29 hours	75.3%
Goyal et al. [1]	8K	Tesla P100 x256	Caffe2	1 hour	76.3%
Smith et al. [4]	8K→16K	full TPU Pod	TensorFlow	30 mins	76.1%
Akiba et al. [5]	32K	Tesla P100 x1024	Chainer	15 mins	74.9%
Jia et al. [6]	64K	Tesla P40 x2048	TensorFlow	6.6 mins	75.8%
This work	34K→68K	Tesla V100 x2176	NNL	224 secs	75.03%

Table 2 : GPU scaling efficiency with ImageNet/ResNet-50 training

	Processor	Interconnect	GPU scaling efficiency
Goyal et al. [1]	Tesla P100 x256	50Gbit Ethernet	~90%
Akiba et al. [5]	Tesla P100 x1024	Infiniband FDR	80%
Jia et al. [6]	Tesla P40 x2048	100Gbit Ethernet	87.9%
This work	Tesla V100 x1088	Infiniband EDR x2	91.62%

2 Approach

There are two primary issues with large-scale distributed training: instability of large mini-batch training and the synchronization communication overhead.

It is well-known that training with large mini-batch is unstable and creates generalization gap [1] [2] [8]. In up to 32K mini-batch training on ImageNet/ResNet-50, this instability was alleviated by several groups [1] [5] [9]. Besides this, [6] has achieved training with 64K mini-batch.

A data parallel distributed training requires an extra step between every training iteration to synchronize and average gradients across participating GPUs. This step is implemented using an all-reduce collective operation. On a large-scale GPU cluster, the overhead of the all-reduce collective operation makes it extremely challenging to achieve linear scaling [5] [6].

These two issues are addressed in this work. We adopt batch size control technique introduced in [4], [10], and [11] to address large mini-batch instability. We develop 2D-Torus all-reducing scheme to efficiently exchange gradients across GPUs.

2.1 Batch Size Control

According to the previous efforts, gradually increasing total mini-batch size during the training reduces the instability of the large mini-batch training. Intuitively, increasing the batch size as the loss landscape of the training becomes "flatter" helps evading the local minima [4] [10] [11]. In this work, batch-size control is adopted to reduce accuracy degradation with mini-batch size exceeding 32K. A predetermined batch-size change scheduling is employed during the training.

2.2 2D-Torus All-reduce

An efficient communication topology is vital for reducing communication overhead of a collective operation. Several communication topologies including Ring all-reduce [12] and hierarchical Ring all-reduce [6] are proposed to improve the efficiency of the all-reduce operation in the previous efforts.

Ring all-reduce algorithm cannot fully utilize the bandwidth of an extremely large-scale cluster with over thousand GPUs. This is because the communication overhead of the

algorithm increases in proportion to the number of GPUs due to network latency as illustrated in [12].

We develop 2D-Torus all-reduce to address this problem. The 2D-Torus topology is described in Figure 1. The GPUs in the cluster are arranged in a 2D grid. In the 2D-torus topology, all-reduce consists of three steps: reduce-scatter, all-reduce, and all-gather. An example case of 2D-Torus all-reduce is shown in Figure 2. Firstly, reduce-scatter is performed horizontally. Then, all-reduce is performed vertically. Finally, all-gather is performed horizontally. Communication overhead of the 2D-Torus all-reduce is smaller than that of Ring all-reduce. Let N be the number of GPUs in the cluster, X be the number of GPUs in the horizontal direction, Y be the number of GPUs in the vertical direction. 2D-Torus all-reduce executes $2(X-1)$ GPU-to-GPU operations. Comparatively, Ring all-reduce scheme executes $2(N-1)$ GPU-to-GPU operations [12]. While the hierarchical all-reduce also does the same amount of GPU-to-GPU operation as the 2D-Torus all-reduce, the data size of the second step (vertical all-reduce) of the 2D-Torus all-reduce scheme is X times smaller than that of the hierarchical all-reduce.

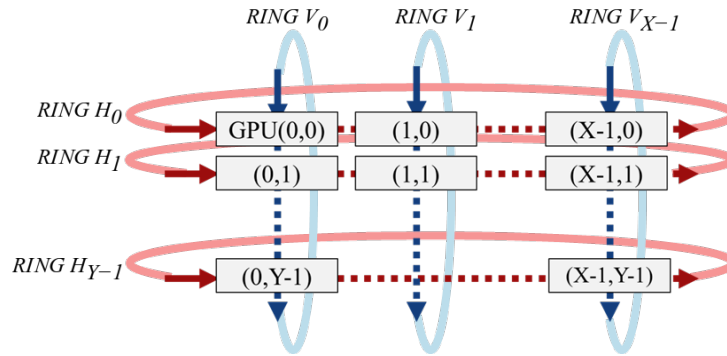


Figure 1 : The 2D-Torus topology comprises of multiple rings in horizontal and vertical orientations.

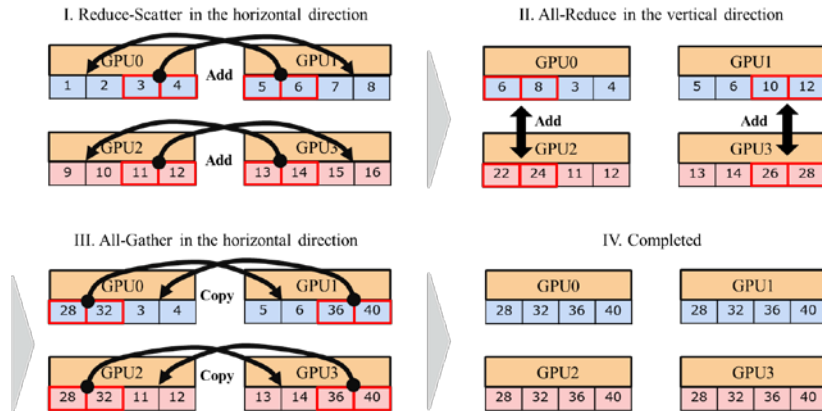


Figure 2 : The 2D-Torus all-reduce steps of a 4-GPU cluster, arranged in 2x2 grid

3 Evaluation

3.1 Experimental Environments

Software: We used Neural Network Libraries (NNL) and its CUDA extension as a DNN training framework. We used development branches based on NNL version 1.0.0. CUDA version 9.0 with cuDNN version 7.3.1 is employed to train DNN in GPUs. We used NCCL version 2.3.5 and OpenMPI version 2.1.3 as communication libraries. The 2D-Torus all-reduce is implemented with NCCL2. The above software is packaged in a Singularity container. We used Singularity version 2.5.2 to run distributed DNN training.

Hardware: We used AI Bridging Cloud Infrastructure (ABCI) as a GPU cluster. ABCI is a GPU cluster operated by National Institute of Advanced Industrial Science and Technology (AIST). It includes 1088 nodes. Each node has 4 NVIDIA Tesla V100 GPUs, 2 Xeon Gold 6148 processors, and 376 GB of memory. GPUs in the same node are connected with NVLink2 interconnect, whereas nodes are connected with 2 InfiniBand EDR interconnects.

3.2 Experimental Settings

Dataset and Model: We used ImageNet [13] dataset. This is a dataset for 1,000 classes image classification. ImageNet consists of 1.28 million training images and 50,000 validation images. We used NNL’s implementation of image augmentation operations including padding, scaling, rotations, resizing, distortion, flipping, brightness adjustment, contrast adjustment, and noising in all our experiments. We used ResNet-50 [7] as a DNN model. All layers in the model are initialized by the values described in [9].

Training Settings: We used LARS [9] with coefficient of 0.01 and eps of 1e-6 to update the weights. The learning-rate (LR) is calculated by the following formula. We used 5-epochs LR warmup. The base LR of 29 and 50 are the exact value used in [9] and the maximum value suggested in [3] respectively.

$$epoch = \frac{ProcessedSamples}{DataSize}$$

$$LearningRate(epoch) = \begin{cases} 0.2 + (29 - 0.2) \frac{epoch}{5} & \text{if } epoch < 5 \\ 29 \left(1 - \frac{epoch}{90}\right)^2 & \text{if } epoch < 30 \\ 50 \left(1 - \frac{epoch}{90}\right)^2 & \text{otherwise} \end{cases}$$

The momentum is calculated from total mini-batch size and learning-rate according to the relation proposed by [14] with the following formula.

$$NoiseScale(epoch) = \frac{LearningRate(epoch) \cdot 32 \cdot 1024}{1 - 0.9^3}$$

$$Momentum(epoch) = 1 - \frac{LearningRate(epoch) \cdot B}{NoiseScale(epoch)}$$

We also employed mixed-precision training introduced in [15]. The forward/backward computations and the communication to synchronize gradients are conducted in half precision float (FP16). The computation in LARS was conducted in single precision float (FP32) because LARS required a wider dynamic range than the FP16 format [6]. We also employed “Batch Normalization without Moving Average” [5] to get the accurate sample mean and variance. The communication to synchronize batch mean and batch squared mean was also conducted in FP32 due to the wider dynamic range. We used 1024 times loss-scaling to decrease numeric errors. We tuned per-worker and total mini-batch size as shown in Table 3 to increase the maximum total mini-batch size. We experimented with increasing the maximum total mini-batch size by increasing the number of GPUs (Exp.1 to Exp.4). However, the training became inefficient when we used over 2176 GPUs (described in the next section). Because of this issue, Exp. 5 and Exp. 6 used only 2176 GPUs.

Gradient Synchronization: Backward computation and communication to synchronize gradients are conducted in parallel by using NNL’s feature. Gradients are synchronized every 14MB. This number is the result of the preliminary experiments not discussed in this paper.

³ B is a total mini-batch size

Table 3: The list of per-worker/total mini-batch size used in our experiments

	#GPUs (Max)	Epoch 1-30	Epoch 31-45	Epoch 46-75	Epoch 76-90
Reference ⁴	64	Per-worker 32 Total 32K			
Exp. 1	1088	Per-worker 32 Total 34K	Per-worker 32 Total 68K		
Exp. 2	2176	Per-worker 16 Total 34K	Per-worker 32 Total 68K		
Exp. 3	2720	Per-worker 16 Total 34K ⁵	Per-worker 16 Total 42.5K	Per-worker 32 Total 85K	
Exp. 4	3264		Per-worker 16 Total 51K	Per-worker 20 Total 63.75K	Per-worker 32 Total 102K
Exp. 5	2176	Per-worker 16 Total 34K	Per-worker 32 Total 68K	Per-worker 40 Total 85K	Per-worker 56 Total 119K
Exp. 6	2176		Per-worker 32 Total 68K	Per-worker 64 Total 136K	

Table 4: The grid dimensions of the 2D-Torus topology used in our experiments.

#GPUs	Vertical	Horizontal
1088	32	34
2176	64	34
2720	40	68
3264	48	68

3.3 Results

We finished the ResNet-50 training in 224 seconds with no significant accuracy loss as shown in Table 5. The training error curves closely resemble the reference curve (Figure 3). While the maximum mini-batch size can be increased to 119K with no significant accuracy loss, further increasing the maximum mini-batch size to 136K decreases the accuracy by about 0.5% (Exp. 6 in Table 5).

We describe training speed and GPU scaling efficiency compared to a single node (4 GPUs) of our method. Table 6 shows the number of GPUs and training throughput when per-worker mini-batch size is set to 32. Although the GPU scaling efficiency decreased from 50 to 70% when we used over 2176 GPUs, it is over 90% when we used 1088GPUs. Previous research [6] reported that the GPU scaling efficiency is 87.9% when they used 1024 Tesla P40s with per-worker mini-batch size set to 32. Compared to the previous research, our communication scheme achieved higher GPU scaling efficiency with faster GPUs (Tesla V100) and more GPUs.

⁴ Training settings (e.g., hyper parameters) reported in [9] are used.

⁵ We used only 1088GPUs in these durations.

Table 5: Top-1 1-crop validation accuracy and training time

	#GPUs (Max)	Batch Size (Min/Max)	Validation Accuracy	Time
Reference ⁶	64	32K	74.74%	124.89 mins
Exp. 1	1088	32K/68K	74.68%	291 secs
Exp. 2	2176	32K/68K	75.03%	224 secs
Exp. 3	2720	32K/85K	74.86%	296 secs
Exp. 4	3264	32K/102K	74.86%	247 secs
Exp. 5	2176	32K/119K	74.79%	232 secs
Exp. 6	2176	32K/136K	74.21%	247 secs

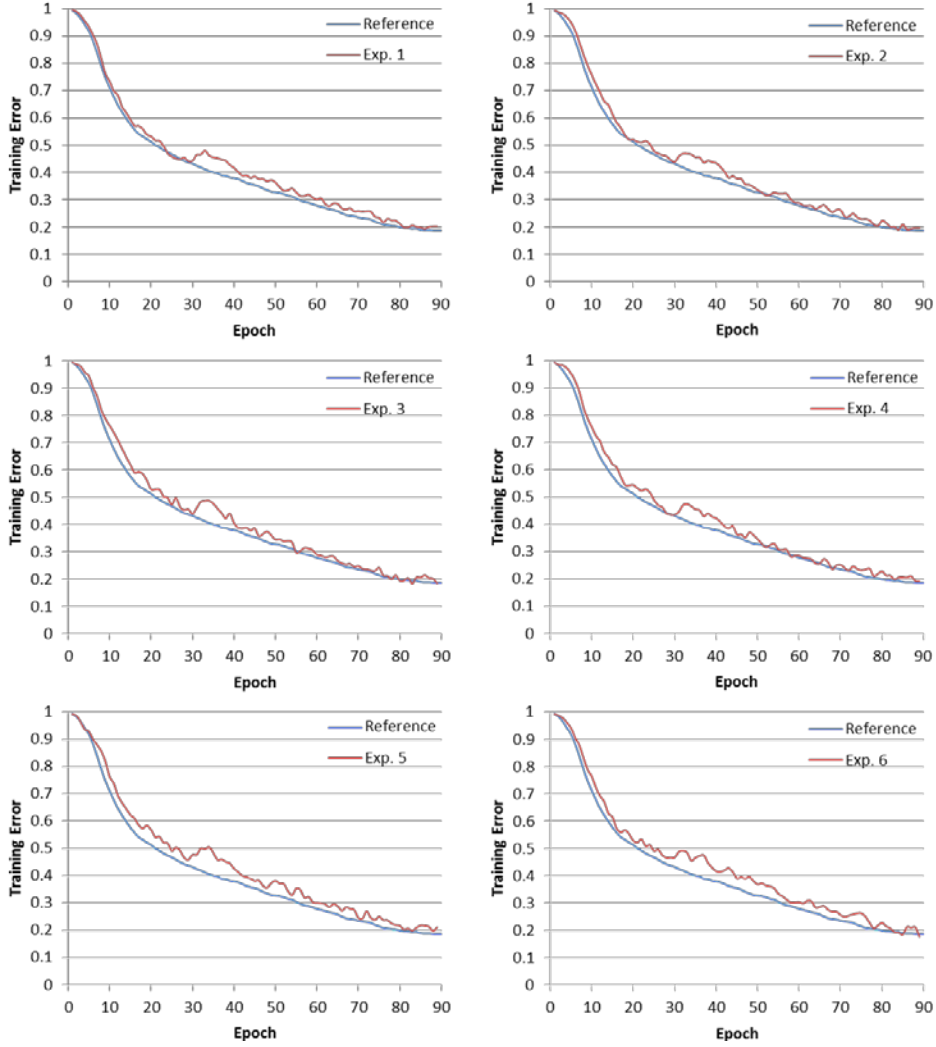


Figure 3: Training error curves

⁶ FP32 is used for computation and communication.

Table 6: Training throughput and scaling efficiency of the 2D-Torus all-reduce

#GPUs	Images per Second	GPU Scaling Efficiency
4	1608	-
1088	400778	91.62%
2176	579040	66.19%
2720	729051	66.67%
3264	688504	52.47%

4 Conclusion

Large-scale distributed deep learning is an effective approach to reduce a DNN training time. We employ several techniques to reduce accuracy degradation while maintaining high GPU scaling efficiency when training with an enormous GPU cluster. The techniques are implemented using Neural Network Libraries. We achieved the training time of 224 seconds and the validation accuracy of 75.03% using 2176 Tesla V100 GPUs. We also achieved over 90% GPU scaling efficiency with 1088 Tesla V100 GPUs.

Acknowledgments

Computational resource of AI Bridging Cloud Infrastructure (ABCI) was awarded by "ABCI Grand Challenge" Program, National Institute of Advanced Industrial Science and Technology (AIST).

The authors would like to thank K. Yoshiyama, T. Narihira, Y. Kobayashi and A. Nakamura for the technical advice as well as A. Shin for the help regarding the manuscript.

References

- [1] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia and K. He, "Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour," arXiv:1706.02677, 2017.
- [2] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy and P. T. P. Tang, "On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima," in *ICLR*, 2017.
- [3] Y. You, Z. Zhang, C.-J. Hsieh, J. Demmel and K. Keutzer, "ImageNet Training in 24 Minutes," arXiv:1709.05011v1, 2017.
- [4] S. L. Smith, P.-J. Kindermans, C. Ying and Q. V. Le, "Don't Decay the Learning Rate, Increase the Batch Size," in *NIPS*, 2017.
- [5] T. Akiba, S. Suzuki and K. Fukuda, "Extremely Large Minibatch SGD: Training ResNet-50 on ImageNet in 15 Minutes," arXiv:1711.04325, 2017.
- [6] 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, "Highly Scalable Deep Learning Training System with Mixed-Precision: Training ImageNet in Four Minutes," arXiv:1807.11205, 2018.
- [7] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," in *CVPR*, 2016.
- [8] E. Hoffer, I. Hubara and D. Soudry, "Train longer, generalize better: closing the generalization gap in large batch training of neural networks," in *NIPS*, 2017.
- [9] Y. You, I. Gitman and B. Ginsburg, "Large Batch Training of Convolutional Networks," arXiv:1708.03888, 2017.
- [10] A. Devarakonda, M. Naumov and M. Garland, "AdaBatch: Adaptive Batch Sizes for Training Deep Neural Networks," arXiv:1712.02029, 2017.
- [11] Z. Yao, A. Gholami, K. Keutzer and M. Mahoney, "Large batch size training of neural networks with adversarial training and second-order information," arXiv:1810.01021, 2018.

- [12] Baidu Research, "baidu-allreduce," 2017. [Online]. Available: <https://github.com/baidu-research/baidu-allreduce>.
- [13] J. Deng, W. Dong, R. Socher, L. jia Li, K. Li and L. F. fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009.
- [14] S. L. Smith and Q. V. Le, "A Bayesian Perspective on Generalization and Stochastic Gradient Descent," in *ICLR*, 2018.
- [15] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh and H. Wu, "Mixed Precision Training," in *ICLR*, 2018.