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

Fujitsu Laboratories Ltd., Japan

Abstract

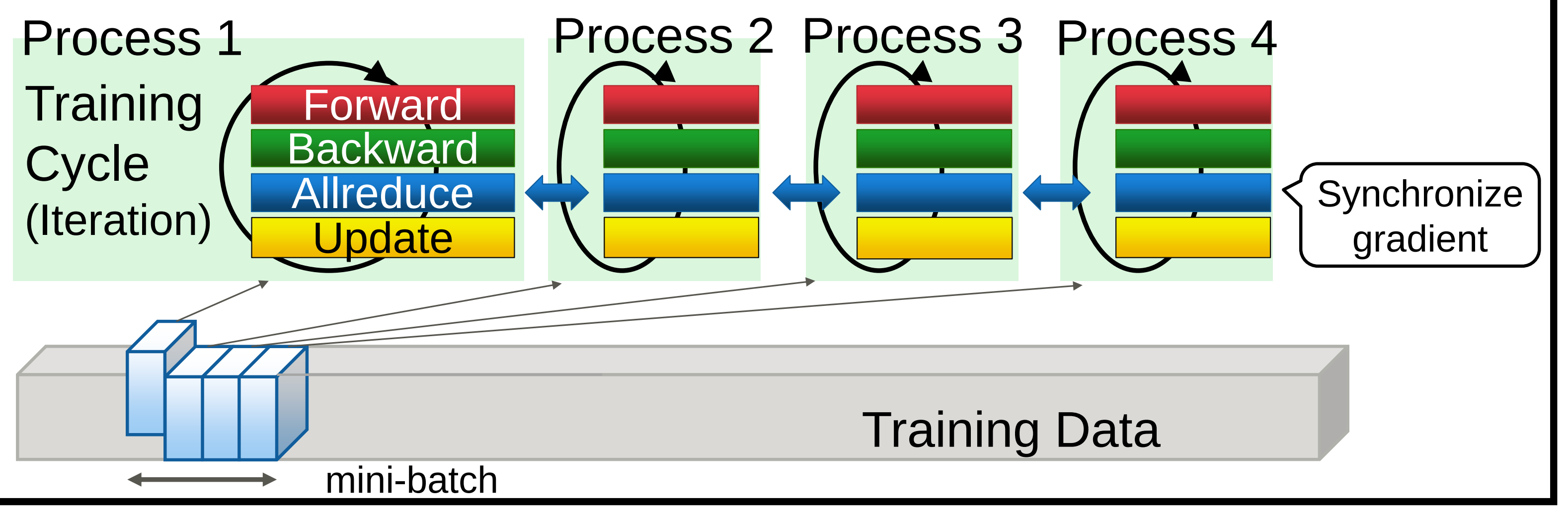
Distributed deep learning is a key technology to accelerate training in deep learning; however, achieving high scalability on large clusters is difficult. In this study, we optimize two points, reducing the additional computation costs and optimizing communication scheduling. By applying the techniques and using 2,048 GPUs, we achieved the world's fastest ResNet-50 training in MLPerf which is a *de facto* standard Deep neural network (DNN) benchmark (as of July 2019).

	Mini-batch size	Processor	Time	MLPerf version
Facebook	8,192	Tesla P100 × 256	1 hour	-
PFN	32,768	Tesla P100 × 1,024	15 min.	-
Tencent	65,536	Tesla P40 × 2,048	6.6 min.	-
Sony	55,296	Tesla V100 × 3,456	2 min.	-
NVIDIA	55,904	Tesla V100 × 1,536	1.33 min.	v0.6 closed
Google	32,768	TPUv3 × 1,024	1.28 min.	v0.6 closed
Our work	**86,016**	**Tesla V100 × 2,048**	**1.17 min.**	**v0.6 open (*)**

* Used closed Division rules, except for tuning six hyper parameters

Distributed deep learning

Distributed deep learning based on data parallelism is an effective method for accelerating DNN training. The approach is used to scale the training process on the cluster, and the mini-batch size becomes larger with an increase in the number of processes.

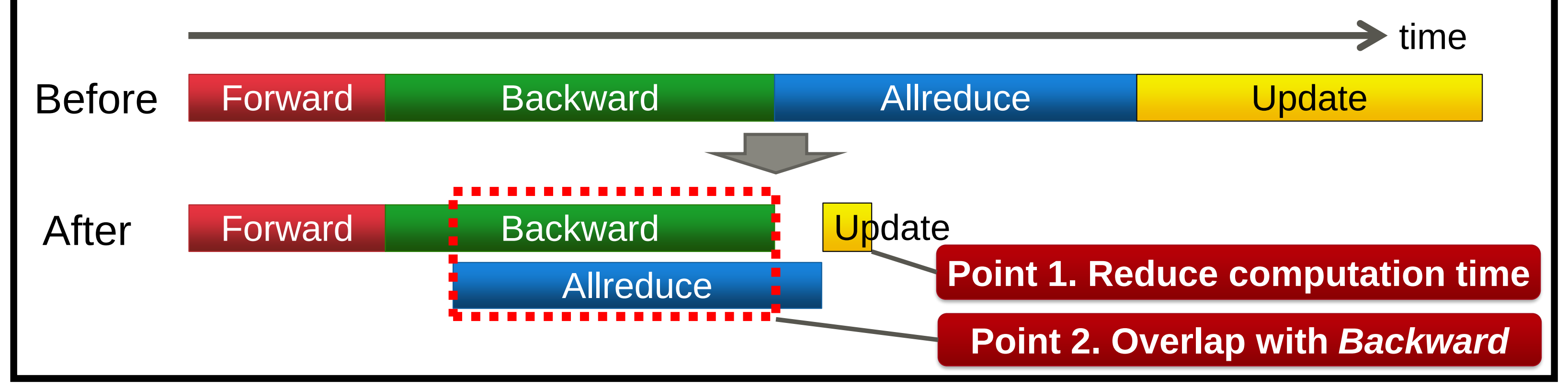


Optimization points

We used the MXNet framework, which is optimized for general training but not enough for large-scale training. Therefore, the following two points are further considered for optimization:

Point 1. **LARS acceleration**: To prevent the degradation of validation accuracy in a large mini-batch training, we use layer-wise adaptive rate scaling (LARS) on *Update* and reduce the computation time.

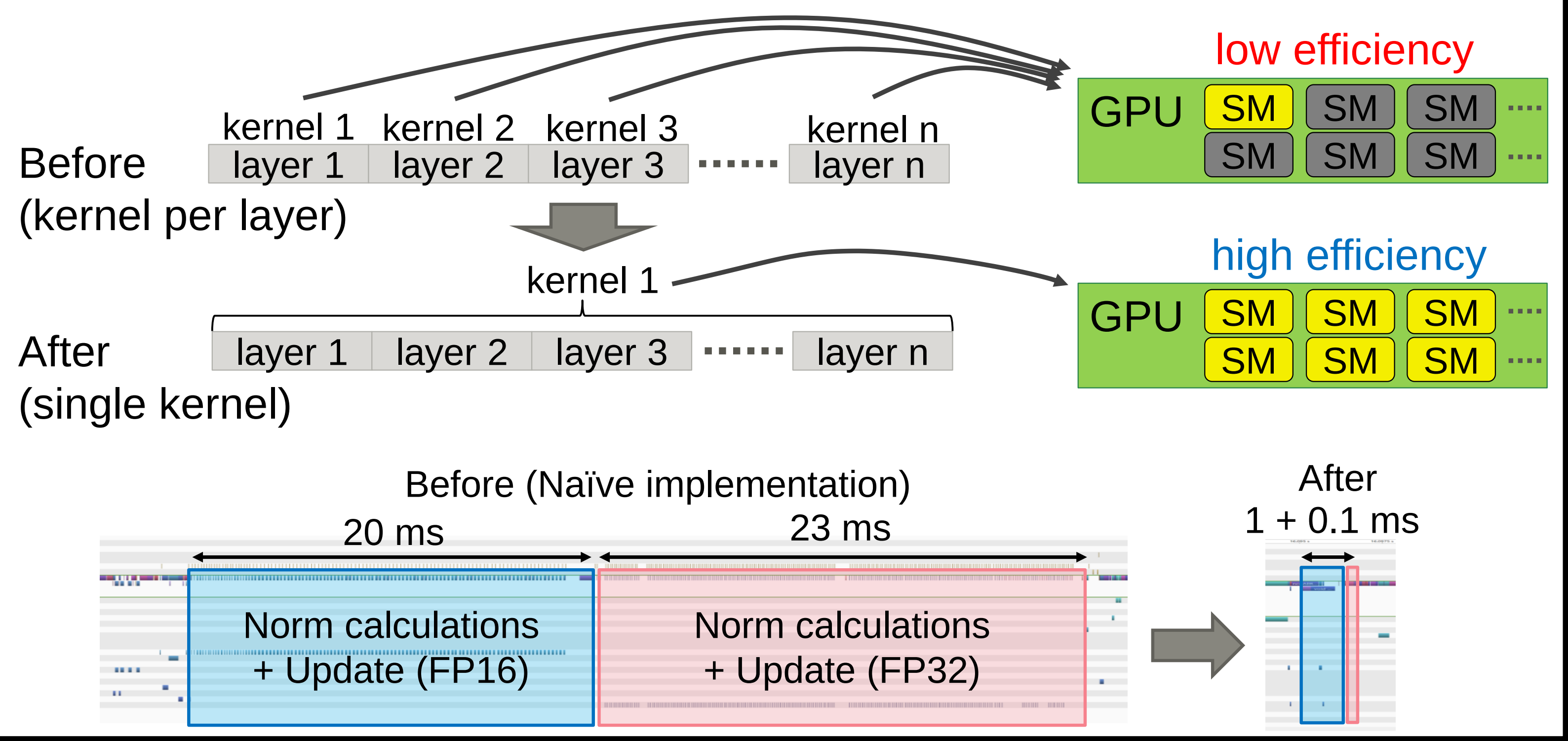
Point 2. **Variable group Allreduce**: The original implementation performs *Allreduce* after *Backward*. Hence, we try to overlap *Allreduce* with *Backward* as much as possible.



Point 1. LARS acceleration

LARS technique adjusts the learning rate of each layer based on the L2-norms ratio between the current weight and weight gradient.

The key idea is **aggregating the LARS calculations for several layers.** We reduce the kernel launch overhead and exploit GPU parallelism.

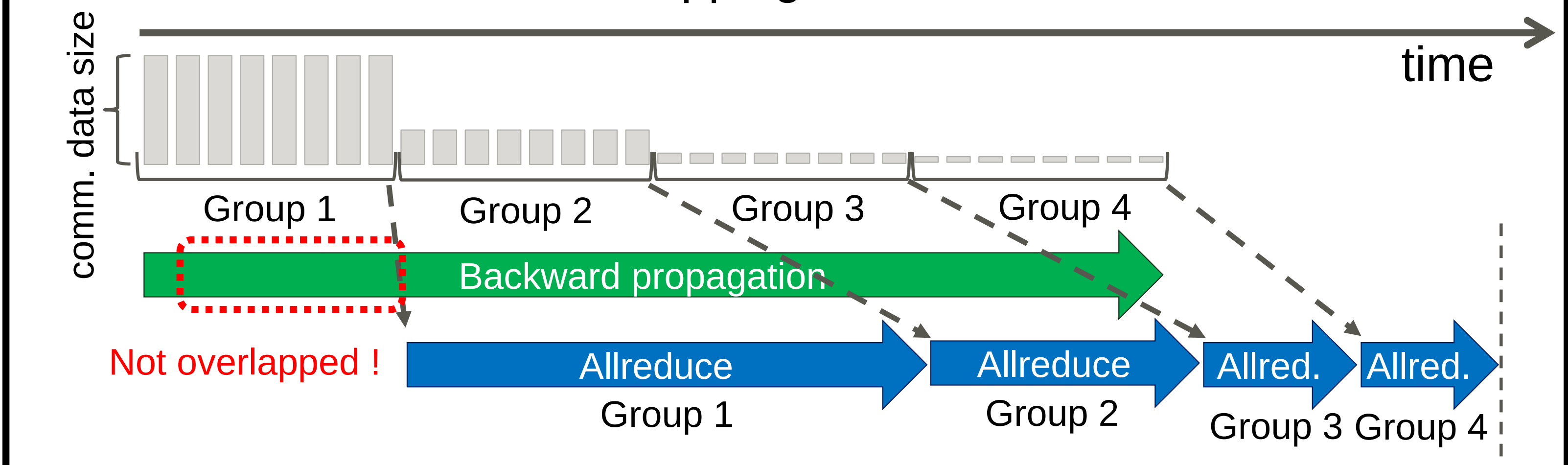


Point 2. Variable group Allreduce

Each layer's *Allreduce* can be performed just after each layer's *Backward*, but *Allreduce* per each layer leads to a large overhead. Hence, we improved the communication efficiency by grouping the communication for each layer. We investigated the following two grouping ideas and adopted a variable group *Allreduce* that **changes the number of layers in a group to efficiently overlap with the *Backward*.**

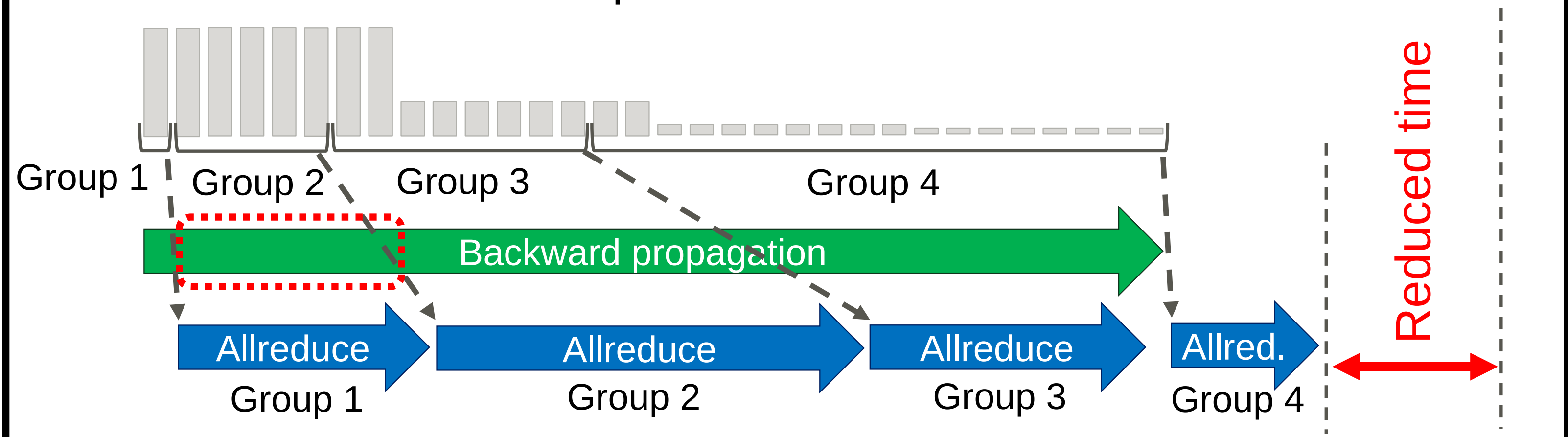
Idea 1: Groups with the same number of layers

Each group has the same number of layers. The first *Allreduce* starts late and it incurs a non-overlapping time with the *Backward*.



Idea 2: Groups with different numbers of layers

The first group has fewer layers and the subsequent groups have more layers so that the first *Allreduce* starts early and the communication time overlaps well with the *Backward*.



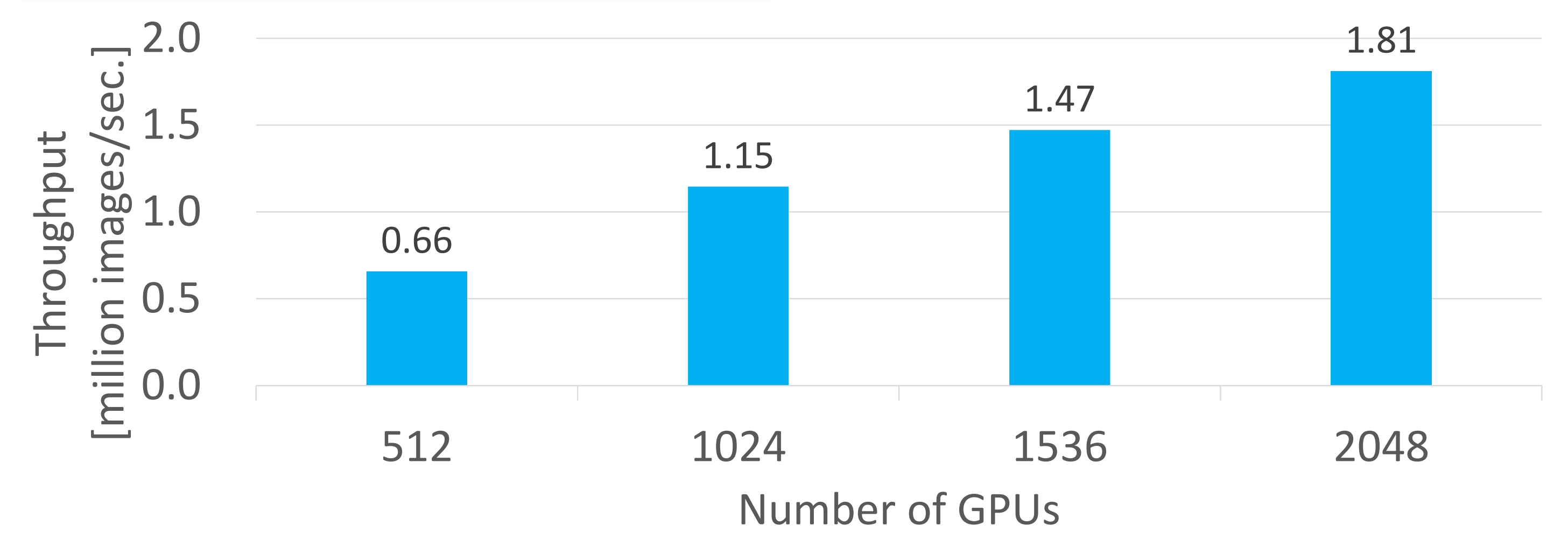
Evaluation environment

We used the AI Bridging Cloud Infrastructure (ABCI) cluster, which is the 8th in the Top 500 list of June 2019, to evaluate the performance of our optimized 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.

Results

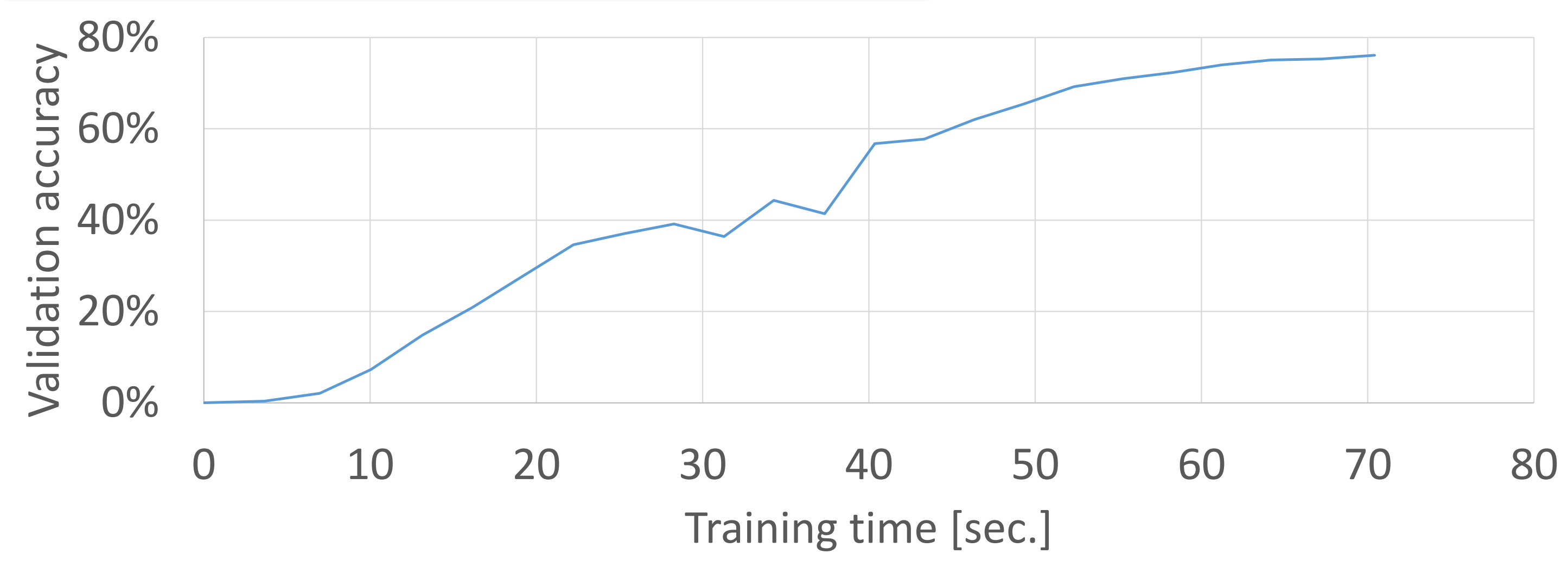
Our measurement of ResNet-50 training is according to the MLPerf v0.6 rule. The dataset is the ILSVRC2012, which contains 1.2 million training images and 50,000 validation images. The training time does not include initialization and is from the start of training to when the validation accuracy exceeds 75.9%.

Throughput scalability



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. The total communication time increases with the number of GPUs, while the computation time decreases as the number of GPUs increases. This scalability shows that our implementation can overlap the communication time with the computation time until 2,048 GPUs.

Training time and validation accuracy



We completed the training with a mini-batch size of 86,016 using **2,048 GPUs**, and the time was **70.4 seconds (1.17 minutes).**

Acknowledgment

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.