Department: Head
Editor: Name, xxxx@email

Distributed Deep Learning with GPU-FPGA Heterogeneous Computing

Kenji Tanaka[1], **Yuki Arikawa**[1], **Tsuyoshi Ito**[1], **Kazutaka Morita**[2], **Naru Nemoto**[1], **Kazuhiko Terada**[1], **Junji Teramoto**[2], **Takeshi Sakamoto**[1]

1. NTT Device Technology Labs., NTT Corporation.
2. NTT Software Innovation Center, NTT Corporation.

***Abstract*—**
In distributed deep learning (DL), collective communication algorithms, such as Allreduce, used to share training results between graphical processing units (GPUs) are an inevitable bottleneck. We hypothesize that a cache access latency occurred at every Allreduce is a significant bottleneck in the current computational systems with high-bandwidth interconnects for distributed DL. To reduce this frequency of latency, it is important to aggregate data at the network interfaces. We implement a data aggregation circuit in a field-programmable gate array (FPGA). Using this FPGA, we proposed novel Allreduce architecture and training strategy without accuracy degradation. Results of the measurement show Allreduce latency reduction to 1/4. Our system can also conceal about 90% of the communication overhead and improve scalability by 20%. The end-to-end time consumed for training in distributed DL with ResNet-50 and ImageNet is reduced to 87.3% without any degradation in validation accuracy.

DEEP NEURAL NETWORKS (DNNs) are receiving increased attention from different industries and academia [1]. They have been specialized in various domains, such as visual recognition, speech recognition, and natural language processing. Several academic breakthroughs have also been achieved by using DNNs to mimic complex, large-scale physical phenomena.

However, to improve the validation accuracy, the amount of data and computation used by DNNs is increased, requiring the use of high-performance computing systems [2]. Increasing the number of processors and accelerators can reduce overall deep learning (DL) training time, but the cost of communication between processors generally limits system scalability. In particular, Allreduce, a collective communication operation, is known to be the main bottleneck in data-parallel distributed DL. Current computing systems for distributed DL are heterogeneous

ones comprising a CPU and multiple graphical processing units (GPUs) [2]. With NVIDIA's GPUDirect RDMA (remote direct memory access) [3] to push data directly to GPUs, bypassing the CPU, such heterogeneous computing systems can achieve high performance in distributed DL.

According to Torsten et al., CPUs access the L3 cache with more latency than the data delivery with 100-Gbps network interface cards (NICs) [4]. This means that the processor's cache access latency is relatively large because the interconnect becomes high-bandwidth, and the performance of RDMA with processing, e.g., Allreduce, is difficult to improve. We hypothesize that a similar phenomenon occurs in GPUDirect RDMA with Allreduce. Zhe et al. showed that GPU L2 cache latency (V100 with 193 cycles, P100 with 234 cycles) is more than 20 times higher than the latency for 100-Gbps message delivery [5]. Therefore, in the current computational systems with high-bandwidth interconnects for distributed DL, the cache access latency occurred at every Allreduce is a significant bottleneck, and to reduce this frequency of latency, it is important to aggregate data at the network interfaces.

In this work, we aim to improve the training speed with a heterogeneous computing system that combines GPUs and FPGA NICs. To achieve this goal, we address the following issues.

- We configure FPGA NICs that aggregate data and execute GPU-FPGA RDMA.
- For distributed DL, this paper proposes a better Allreduce architecture based on network topology and algorithms when data can be aggregated at network interfaces.
- We propose a distributed DL strategy that will be improved by novel hardware and Allreduce architecture.

PROPOSED SYSTEM

First, we introduce a DMA driver and memory controller for FPGA NICs that directly or more efficiently push data into GPU memory than regular NICs. Next, we propose FPGA Ring-Allreduce, which is an Allreduce architecture suitable for proposed FPGA NICs with a data aggregation function and ring networks (Fig. 1) [6]. We also propose a strategy for distributed deep learning with FPGA Ring-Allreduce.

GPU-FPGA DMA driver Since the data transfer via the CPU requires a large latency, we developed a DMA driver that bypasses the CPU and transfers data from/to the GPU device memory directly to/from the FPGA's static random access memory (SRAM). We used NVIDIA's GPU DirectRDMA interface to implement a DMA between the GPU and FPGA. This driver uses CUDA to pin the GPU memory region for data transfer and zero-copy send to/receive from the SRAM on the FPGA. With this driver, the DMA controller is responsible for moving the data until the CPU notifies the completion, thus enabling us to obtain a silicon-level performance with the DMA controller.

Memory controller As shown in our system (Fig. 1), the DMA controller in the FPGA receives the trigger from the DMA driver and moves the training result from GPU memory to the SRAM in the FPGA. However, FPGA vendors' DMA controller has a memory access bandwidth of up to 90 Gbps due to redundant interactions with memory.To maximize the memory access bandwidth, we optimized a memory controller for the device driver. In our devices, an address incrementing function and a burst length adjustment function are not needed for memory access because the device driver pins the memory. Since these functions are redundant for transferring large data sizes, such as DNN weights, to SRAM, our memory controller works to prevent these from wasting bandwidth.

FPGA Ring-Allreduce First, we adopted the ring-network topology to connect each server. The ring-based Allreduce is bandwidth optimal so that some existing distributed DL systems also applied a ring-Allreduce [7]. Therefore, we connected the FPGAs with a ring topology (Fig. 1).

Next, we adopted Ring-Allreduce as the most suitable Allreduce algorithm for the ring topology. For details of the algorithm, we referred to a previous study [6]. Implementing the Ring-Allreduce algorithm in a physical ring topology makes it possible to restrict the physical data flow in one direction and prevent congestion. Two directional paths are used for data aggregation and broadcast, respectively.

In the following, we explain the operation of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MM.2020.3039835, IEEE Micro

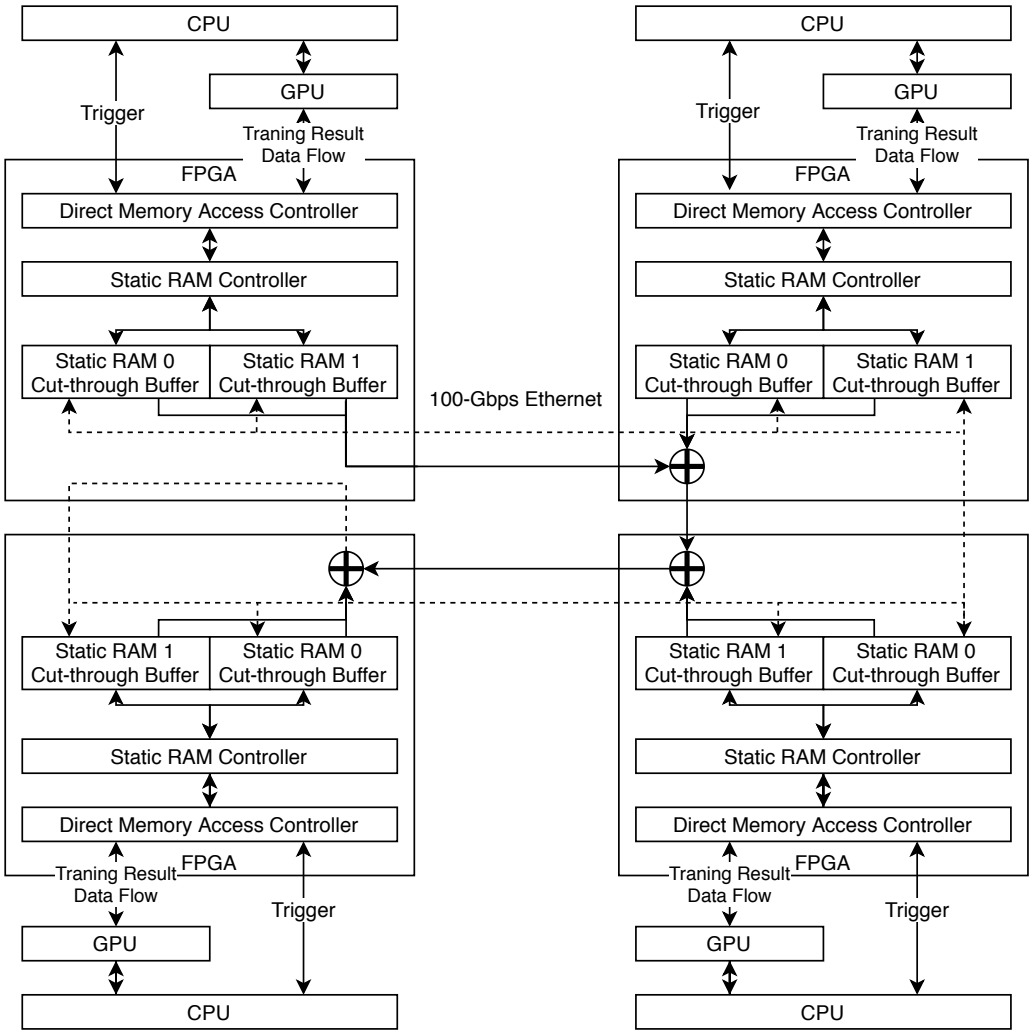

Figure 1. Schematic of proposed heterogeneous computing system and operation mechanism of FPGA Ring-Allreduce. The inter-node connections depicted by the solid and dashed lines are the data aggregation and data broadcast paths, respectively.

FPGA Ring-Allreduce (Fig. 1). The master node transmits the buffered data to the next node, and the other nodes wait for a data packet from the previous node. When a data packet flows into the next node, the packet and the buffered packet are input to the aggregation circuit (denoted as $\oplus$ in Fig. 1). Then, outputs from the aggregation circuit are sent to the next node further along the ring. After one round of data flow through the ring network, the aggregation is complete at all nodes, and the result arrives at the master node. The proposed FPGA Ring-Allreduce can pipeline aggregation and broadcast.

Parameter-based computing/communication overlap

We modified the training strategy so that it is suitable for our computing system and does not degrade validation accuracy. A previous study proposed a training strategy called layer-based

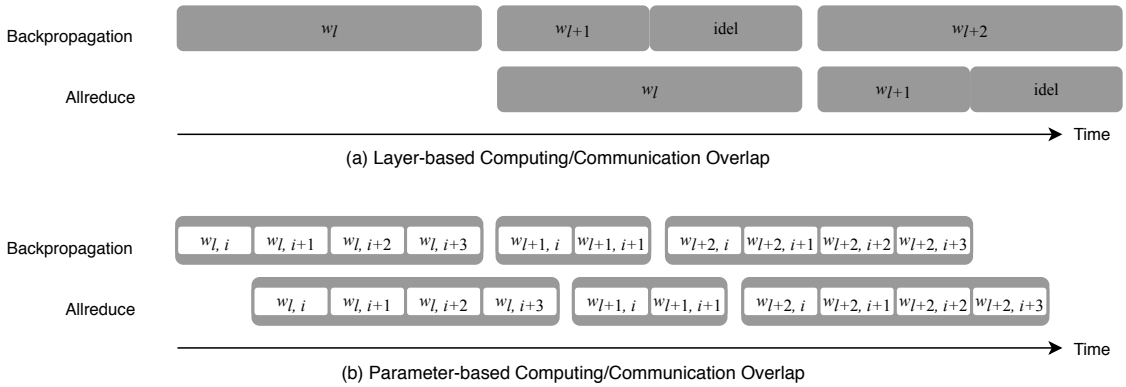

Figure 2. Comparison of the behaviors of the (a) conventional and (b) proposed strategies when layer size has an imbalance. Here, l and i are the indexes of the layer and the parameter, respectively.

computation/communication overlap [8]. However, it is not suitable for our pipeline system because the data size in each layer is different. System efficiency is improved by pipelining processing fine and uniform data sizes. Hence, our modified strategy extends the previous strategy by overlapping the computation and communication for each parameter (Fig. 2). Parameter-based computing/communication overlap (PCCO) can be run on conventional computer systems, but not efficiently because frequent DMAs increase CPU load. We implemented PCCO as a module to Horovod [9], an existing communication library for DL that can be applied to various DL frameworks such as TensorFlow, Keras, and Pytorch.

PERFORMANCE EVALUATION

Evaluation environment We used the Alveo U250 FPGA cluster system with 100-Gbps Ethernet for the performance evaluation. For comparison, we prepared an InfiniBand EDR-based system with a Mellanox ConnectX-4 HCA and Switch IB-2 with GPUDirect RDMA and the NVIDIA Collective Communication Library (NCCL) version 2.4 [10]. NCCL 2.4 adopts a binary tree Allreduce algorithm. Each node has two Intel Xeon CPUs (E5-2660 v4 @2.00 GHz) and two NVIDIA Volta V100 GPUs. We built a system consisting of eight of these servers. We used TensorFlow and Horovod for training and communication, respectively. We implemented PCCO so that we can use Horovod.

Hardware performance First, we check the bandwidth of data movement via PCIe Gen3 x16. Fig. 3 shows the bandwidth of data movement via PCIe with the DMA controller and the proposed memory controller. The results show that the DMA controller writes to SRAM with a 95.0-Gbps throughput and reads from SRAM with a 110.5-Gbps throughput. This throughput is almost the limit of PCIe Gen3 x16, which is equivalent to 100-Gbps Ethernet bandwidth between nodes.

Next, we check the latency of FPGA Ring-Allreduce of the 64-Byte parameters across two nodes. We measured the number of clock cycles for inter-node FPGA Ring-Allreduce. The total latency was 1888 nsec at 250 MHz clock speed. Among this latency, the inter-node data transfer latency was 1740 nsec, and the total latency of reading from SRAM buffer, aggregation, broadcast, and writing back to SRAM buffer was 148 nsec. This total latency is almost equivalent to the latency of the GPU accessing the L2 cache once.

Effectiveness of hardware offloading We examined the performance improvement achieved by offloading Allreduce to the FPGA NICs. We compared the Allreduce latency of the proposed system with that of the InfiniBand-based system using NCCL and GPU DirectRDMA. We set the message size of Allreduce to 64 MB and ran it on a different number of nodes. The average values of the measurement are plotted in Fig. 4. This result shows that the latency of FPGA Ring-Allreduce is about 25% lower than that of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MM.2020.3039835, IEEE Micro

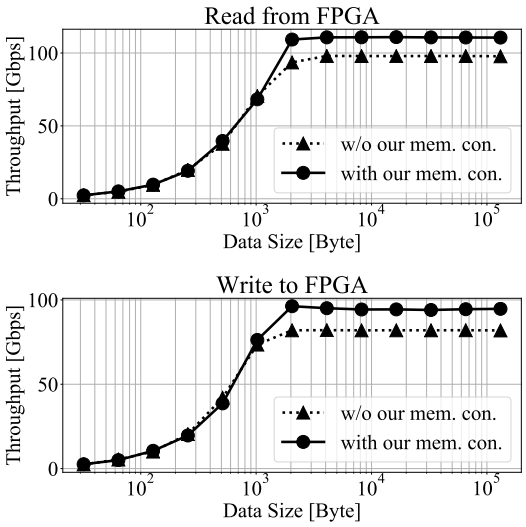

Figure 3. DMA performance evaluation

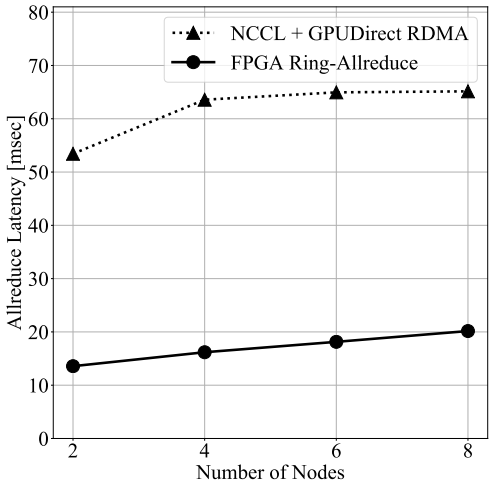

Figure 4. Allreduce latency evaluation

NCCL Allreduce. We fitted these results to the latency model of the Ring-Allreduce algorithm ($2\alpha(n-1)$) and the binary tree Allreduce algorithm ($2\beta \log n$) for a n-node cluster and got latency parameters α and β, respectively. The prediction indicated that FPGA Ring-Allreduce had lower latency even more than 32 nodes. This prediction indicated that FPGA Ring-Allreduce has lower latency than InfiniBand-based system even in a 32-node or larger cluster.

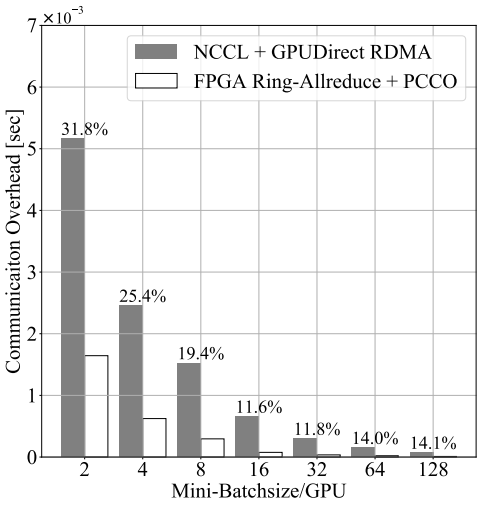

Figure 5. Communication overhead of our system and conventional system.

Effectiveness of hardware/software co-operation We examined the reduction in communication overhead when combining FPGA Ring-Allreduce and PCCO. We used the TensorFlow Synthetic Benchmark, which is equivalent to the training of ResNet-50 without storage overhead. We looked at the communication time trends at runtime with different mini-batch sizes per GPU and measured the reduction between two nodes at two nodes. The results are shown in Fig. 5. It can be seen that the overhead can be reduced by using the proposed system for all mini-batch sizes. In particular, when the mini-batch size per GPU is 16 or more, the communication overhead can be reduced by 85% or more.

Training efficiency Figure 6 shows the improvement in the number of images that can be trained in one second (training efficiency). We ran the TensorFlow Synthetic Benchmark with different mini-batch sizes per GPU with two GPUs installed on each of the eight nodes. Our system improved training efficiency by up to 155.6%. A 130.7% improvement was achieved when the batch size per GPU was set to 32.

Training efficiency-variation Variations in training efficiency is a major undesired factor in distributed DL with synchronous communication.

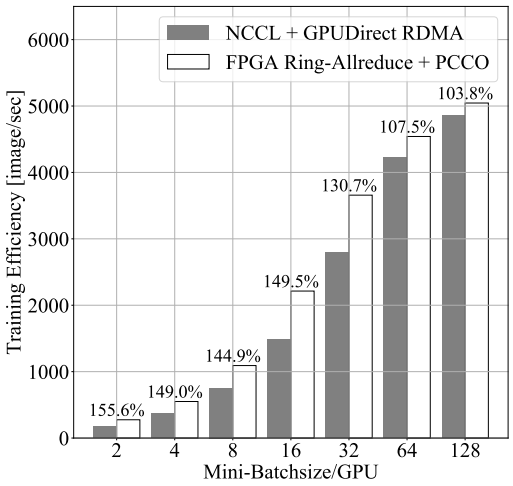

Figure 6. Improvement of training efficiency.

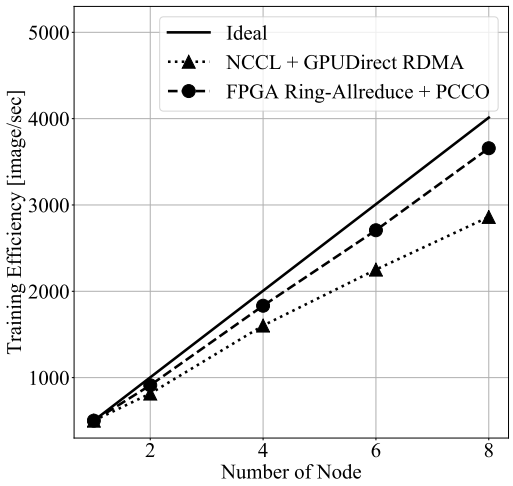

Figure 8. Scalability of proposed system in comparison with ideal scalability.

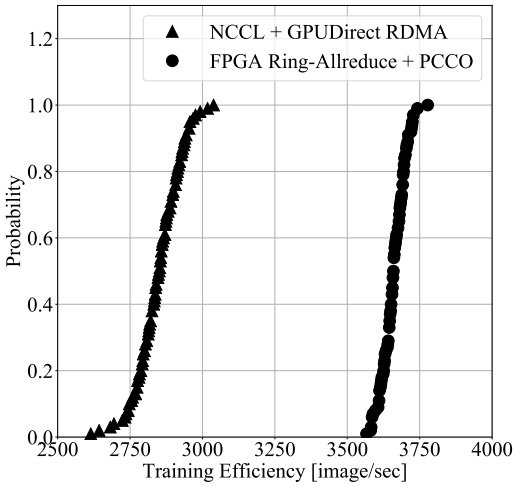

Figure 7. Variation of training efficiency plotted as CDF.

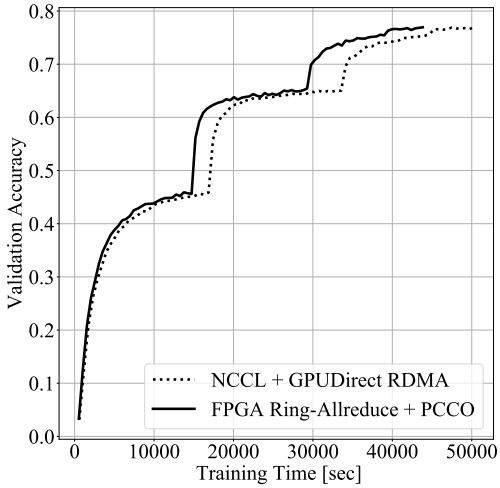

Figure 9. Comparison of time consumption for end-to-end training.

In Fig. 7, the variation in the TensorFlow Synthetic Benchmark training efficiency is presented as an empirical cumulative distribution function (CDF). The evaluation shows that the difference between the median and 99.9th percentile latency of FPGA Ring-Allreduce is 94.4 images/sec, 2.5 times less than that of NCCL Allreduce.

Training efficiency scalability Figure 8 shows that our FPGA Ring-Allreduce has a higher scaling efficiency than NCCL Allreduce. When the mini-batch size per GPU was set to 32, the scaling efficiency compared with a single node could reach 91.2%. Compared with NCCL Allreduce, the scaling efficiency improved from 71.3% to 91.2%.

End-to-end training time We evaluated the performance gains for end-to-end training. We

used the most popular DNN model (ResNet-50 [1]) and trained it on an ImageNet-1K (ILSVRC12) dataset [1]. We set the mini-batch size to 32 per GPU. As shown in Fig. 9, switching the computing system to the proposed system can reduce the time spent in end-to-end DNN training up to 87%. Furthermore, the verification accuracy of the proposed system has no degradation.

RELATED WORK

In-network computing There is a recent focus on moving various application tasks inside the network, known as in-network computing. The improvement of ResNet-50's training efficiency using the in-network computing switch [11] was 111%, but our performance improvement was 130.7%. Therefore, though more research in needed, our FPGA NIC is expected to achieve better training efficiency than the in-network computing switch. Tokusashi et al. found that in-network computing improves not only latency performance but also power efficiency [12]. Also, for distributed DL, Jia proposed a large-scale Allreduce algorithm that combines Ring-Allreduce and Hierarchical-Allreduce [2]. This large-scale Allreduce algorithm shows that combining in-network computing switches with our FPGA NIC is preferable.

CONCLUSION

In this study, we propose a GPU-FPGA heterogeneous computing system to relax the Allreduce bottleneck of distributed DL. In recent years, RDMA technology has become popular, enabling data to be pushed directly to the computing device memory, but the frequent cache access of computing devices has become a bottleneck. In this work, we offload Allreduce into FPGAs and propose GPU-FPGA heterogeneous computing as a distributed DL infrastructure with the following features.

- We configure FPGA NICs that aggregate data and execute GPU-FPGA RDMA.
- For distributed DL, this paper proposes a better Allreduce architecture based on network topology and algorithms when data can be aggregated at network interfaces.
- We propose a distributed DL strategy that will be improved by novel hardware and Allreduce architecture.

With these proposals, we designed a communication library module to adapt an FPGA NIC to a distributed DL framework. The hardware and software can be run in TensorFlow, Keras, and Pytorch. We compared the system's performance with conventional systems using InfiniBand and GPUDirect RDMA. The proposed FPGA NIC can move data with a throughput equivalent to 100-Gbps Ethernet and perform inter-node Allreduce with latency comparable to intra-node Allreduce. Offloading Allreduce to FPGA NICs reduces the inter-node Allreduce latency by a quarter. We also found that our system can conceal about 90% of the communication overhead and improve scalability by 20%. End-to-end distributed DL with ResNet-50 and ImageNet improved by 12.7% with no degradation in validation accuracy. In conclusion, heterogeneous computing, which incorporates special devices that aggregate data at a network interface, enables us to select an acceleration strategy previously unavailable, and can provide significant performance improvements in accelerating distributed DL.

Finally, we describe our future research agenda. First, our FPGA Ring-Allreduce architecture is estimated to have a scalability advantage over the current GPU DirectRDMA-based Allreduce architecture for less than 64 nodes. However, for further scale-out, it is needed to expand to a two-dimensional torus topology. FPGA Ring-Allreduce is an architecture suitable for one-dimensional torus topologies, but it can easily support two-dimensional torus topologies by adding two Ethernet ports to the FPGA. Since the processing latency in the FPGA is very small, tens of nanoseconds, the processing latency does not become a major bottleneck as the number of ports increases. We need to consider which Allreduce algorithm to apply for two-dimensional torus topology. Next, we will reduce the latency caused by storage. To access the storage server over the network, we will implement NVMe over Fabrics in our FPGAs. In addition, it is known that special attention to data shuffling is required when training large models, such as language models, with large data for distributed DL. In distributed DL, data shuffling is needed for data loaders, thus offloading data shuffling capabilities into our FPGA NICs also reduce the storage

overhead.

REFERENCES

1. I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, MIT press, 2016.
2. T. Ben-Nun and T. Hoefler, "Demystifying Parallel and Distributed Deep Learning: An In-depth Concurrency Analysis," ACM Comp. Surveys, vol. 52, no. 4, pp. 1–43, 2019.
3. NVIDIA, "DEVELOPING A LINUX KERNEL MODULE USING RDMA FOR GPUDIRECT," 2020. [Online]. Available: https://docs.nvidia.com/pdf/GPUDirect_RDMA.pdf
4. T. Hoefler, S. D. Girolamo, K. Taranov et al., "sPIN: High-performance streaming Processing In the Network," *Proc. SC17*, pp. 1–16, 2017.
5. Z. Jia, M. Maggioni, B. Staiger et al., "Dissecting the NVIDIA Volta GPU Architecture via Microbenchmarking," preprint, 2020, arXiv:1804.06826.
6. P. Patarasuk and X. Yuan, "Bandwidth optimal all-reduce algorithms for clusters of workstations," *J. Parallel Distrib. Comp.*, vol. 69, no. 2. pp. 117–124, 2009.
7. C. Yingm S. Kumar, D. Chen et al., "Image Classification at Supercomputer Scale," preprint, 2018, arXiv:1811.06992.
8. P. Sun, W. Feng, R. Han et al., "Optimizing Network Performance for Distributed DNN Training on GPU Clusters: ImageNet/AlexNet Training in 1.5 Minutes," preprint, 2019, arXiv:1902.06855.
9. A. Sergeev and M. D. Balso, "Horovod: fast and easy distributed deep learning in TensorFlow," preprint, 2018, arXiv:1802.05799.
10. NVIDIA, "NVIDIA Collective Communications Library (NCCL)," 2020. [Online]. Available: https://docs.nvidia.com/deeplearning/nccl/
11. G. Shainer and D. Gruner, "InfiniBand In-Network Computing Technology and Roadmap," *BoF SC18*, 2018.
12. Y. Tokusashi, T. D. Huynh, P. Fernando et al., "The case for in-network computing on demand," In: The Fourteenth EuroSys Conference 2019, pp. 1–16. (2019)

Kenji Tanaka is currently working as a Research Engineer with the Device Architecture Research Group, NTT Device Technology Laboratories. He received his M.S. degree in Complexity Science and Engineering from the University of Tokyo, Tokyo, Japan, in 2015. In 2016, he joined NTT Device Technology Laboratories, where he researched ultrahigh-speed ICs and systems for optical transmission using FPGA. His current research is adding programmability to network switches and NICs and finding new use cases for newly available programmable hardware. He is a member of the Institute of Electronics, Information and Communication Engineers (IEICE) and Information Processing Society of Japan (IPSJ), Japan

Yuki Arikawa is currently working as a Research Engineer with the Device Architecture Research Group, NTT Device Technology Laboratories. He received his B.E. and M.E. degrees from Waseda University, Tokyo, Japan, in 2008 and 2010, respectively. Since joining NTT in 2010, he has been researching device architectures for 10G-EPON and 5G mobile communications systems. He is currently engaged in research and development of computer architectures towards the post-Moore era. He is a member of IEICE, Japan.

Tsuyoshi Ito received his B.E. and M.E. degrees in Physical Electronics from Nagoya Institute of Technology, Aichi, Japan, in 1996 and 1998, respectively. Since joining NTT System Electronics Laboratories in 1998, he has been engaged in research and development on high-speed opto-electronic devices and optical access systems. He is currently working as a Research Engineer with NTT Device Technology Laboratories. His current interests are high-speed optical interconnects and systems for future computing. He is a member of IEEE, IEEE Photonics Society, IEEE Circuits and Systems Society, IEEE Communications Society, and IEEE Computer Society.

Kazutaka Morita received the B.E. and M.E. degrees in mathematical informatics from the University of Tokyo, in 2005 and 2007, respectively. He joined NTT Corporation, a Japan-based telecommunication company, as a research engineer in 2007. He has been involved in various open-source projects such as QEMU/KVM and OpenStack. His recent focus is on optimization of deep learning and he is a committer of TVM, an open-source compiler stack for deep learning. He is a recipient of the IPSJ Yamashita SIG Research Award in 2012.

Naru Nemot is with NTT Device Technology Laboratories, Kanagawa, Japan. He received the B. E. degrees in inorganic materials from Tokyo Institute o Technology in 2003 and 2005, respectively. He has been a Research Engineer with the Photonic Node Module Development Project, Photonic Network Device Project, Device Innovation Center, NTT. He received the JSPE Best Presentation Award in 2009 and JSPE Young Engineer Award in 2010. He is a member of the Japan Society of Mechanical Engineers and the Japan Society for Precision Engi-

neering (JSPE).

Kazuhiko Terada received the B.S. and M.S. degrees in electrical and electronic engineering from Kyoto University, Kyoto, Japan, in 1998, 1999, respectively. In 1999, he joined NTT Network Innovation Laboratories, where he was engaged in research and development of WAN interfaces based on Gbit/s Ethernet and 10 Gbit/s Ethernet technologies. He is currently working at the Photonics-Electronics Convergence Laboratory in NTT Device Technology Laboratories, Kanagawa, Japan.

Junji Teramoto received the B.E. and M.E. degrees in Electronics, Information, and Communication Engineering from WASEDA University in 1996, 1998, respectively. Since joining NTT Information Laboratories, NTT Corporation, Kanagawa, Japan, in 1998, he has been engaged in research and development on multimedia database management systems. He is currently a senior research engineer with the NTT Software Innovation Center. His current interest is a distributed computing system.

Takeshi Sakamoto received his B.E. and M.E. degrees in Electronics Engineering from Kyoto University, Kyoto, Japan, in 1994 and 1996, respectively. In 1996, he joined NTT Opto-electrical Laboratories. From 2007 to 2016, he worked with NTT Access Network Service System Laboratories to develop optical access systems. He is currently working as a Senior Manager with NTT Device Technology Laboratories. His current research area is optoelectronic devices and systems for data processing. He is a member of IEICE and IEEE.