

Large-Message Size Allreduce at Wire Speed for Distributed Deep Learning

Kenji Tanaka[†], Yuki Arikawa[†], Kenji Kawai[†], Junichi Kato[†], Tsuyoshi Ito[†], Huy Cu Ngo[†],

Kazutaka Morita^{*}, Fumiaki Miura^{*}, Takeshi Sakamoto[†], Satoshi Shigematsu[†]

[†]NTT Device Technology Labs, NTT Corporation

^{*}NTT Software Innovation Center, NTT Corporation

{kenji.tanaka.hr, yuki.arikawa.zs, kenji.kawai.dk, junichi.kato.ug, tsuyoshi.ito.pw, huycu.ngo.ge,
kazutak.morita.fp, fumiaki.miura.rf, takeshi.sakamoto.wa, satoshi.shigematsu.cy}@hco.ntt.co.jp

ABSTRACT

In large-scale distributed deep learning, the Allreduce operation for large messages (100 KB or more) is critical for gathering gradients from multiple worker nodes and broadcasting the sum of the gradients to them. When the message is large, the latency in Allreduce operation would make it difficult to take advantage of large-scale distributed deep learning. To reduce the latency, we devised a dataflow architecture with an Allreduce-specific hardware accelerator that performs data aggregation and reduction while data is being transferred. The accelerator is designed to immediately start Allreduce operation before an entire message is received. Furthermore, Allreduce can be operated at wire speed by vectorizing the gradients and summing them in parallel. Experimental results reveal that the proposed architecture performs Allreduce at 96% of wire speed for a large message. Moreover, the latency of Allreduce is reduced by 65% compared with a state-of-the-art Allreduce method when applied for ResNet-50.

KEYWORDS

Distributed Deep Learning, Allreduce, Hardware Acceleration, FPGA, GPU, Aggregation Node, Wire Speed

1 INTRODUCTION

Deep learning (DL) is promising for various application fields. Training DL models, which use huge amounts of data, requires enormous computational resources. Recently, large-scale distributed DL has been studied as a solution, and it has shown good performance with a synchronous data-parallelism approach [1, 2]. In this approach, collective communication of Allreduce is used to share the gradients calculated at each worker node. Since the message size of gradients reaches hundreds of kilobytes to several hundred megabytes (e.g. the message size of gradients in ResNet-50 is 97.5 MB), increasing the throughput and reducing the latency of Allreduce for large-messages are key issues [3].

To achieve high throughput in collective communications of large messages using multiple GPUs, NVIDIA provides the NVIDIA Collective Communications Library (NCCL) using the Ring-Allreduce algorithm [4]. An advantage of the Ring-Allreduce algorithm is that bandwidth congestion does not occur like it does in the Butterfly-Allreduce algorithm, so it is suitable for intra-node

Allreduce. In contrast, its operation time increases in proportion to the number of transfers, which limits the inter-node Allreduce performance [5]. Therefore, the NCCL is better suited for intra-node and alternative collective-communication methods for inter-node Allreduce [6, 7, 8].

For inter-node Allreduce, Mellanox proposed the Scalable Hierarchical Aggregation Protocol (SHArP) which is a network switch with various collective-communication algorithms. To solve the bandwidth congestion problem, SHArP performs data aggregation and reduction while data is being transferred within the network [9]. In this way, SHArP significantly improves scalability and reduces the latency of Allreduce for message sizes below 5 KB [10, 11]. Ideally, it can perform collective communication at wire speed. However, as message size increases, operating Allreduce at wire speed becomes difficult due to data-store latency generated by temporarily storing all data in the buffer before the reduce operation [9, 12].

In this paper, we propose a dataflow architecture for the aggregation node and worker nodes to achieve low-latency Allreduce at wire speed for large-messages. For the aggregation node, we designed an Allreduce-specific hardware accelerator implemented in a field programmable gate array (FPGA) with cut-through buffering for each vectorized gradient. For worker nodes, we implemented gradient casting to the aggregation node in a streaming way. We carried out experimental measurements to compare the hardware latency with SHArP Allreduce. The results show that the proposed architecture achieves low-latency Allreduce operation at wire speed for large messages.

2 ARCHITECTURE AND MECHANISM

The proposed dataflow architecture consists of an aggregation node using an Allreduce-specific dataflow hardware accelerator and multiple worker nodes to achieve large-message Allreduce at wire speed. In the proposed accelerator, cut-through processing implemented with vectorized gradients (frames) performs Allreduce and transmission as Ethernet frames are received. Moreover, gradient casting at each worker node is performed asynchronously with various numbers of buffers with different size to optimize Allreduce latency.

Allreduce-specific dataflow hardware accelerator: Fig. 1(a) shows the buffering method with conventional store and forward

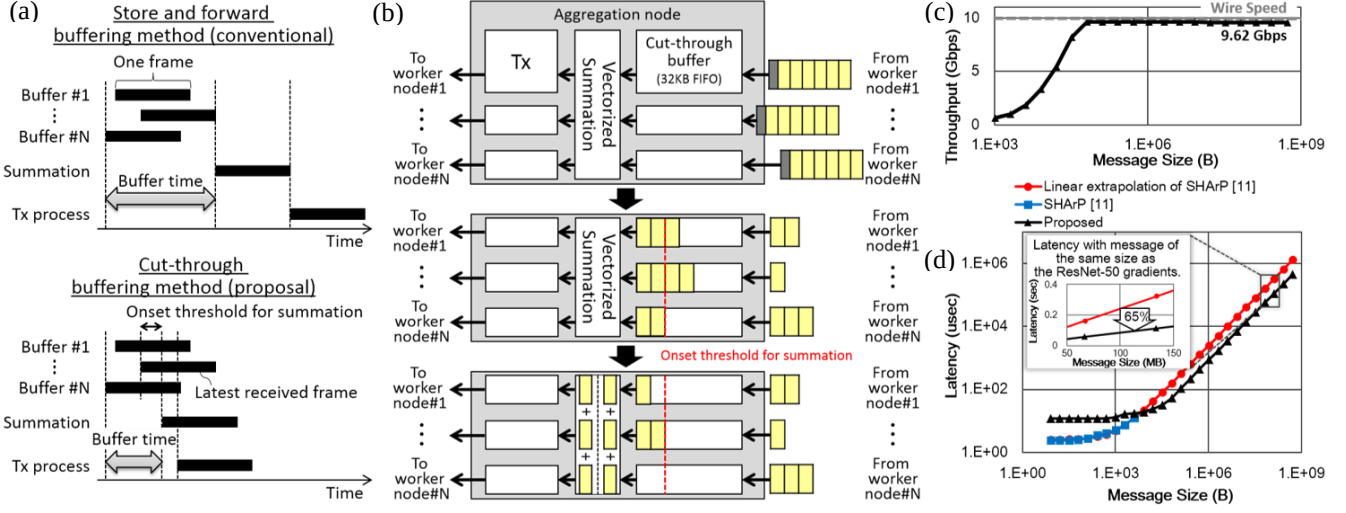


Figure 1: (a) Conventional and proposal buffering methods. (b) Allreduce-specific dataflow hardware accelerator. (c) Allreduce throughput in proposed architecture. (d) Comparison of Allreduce latency with SHARp and proposed architecture.

processing and the proposed cut-through processing. In the conventional processing, the buffer time becomes hardware latency, since the vectorized summation starts after an entire frame is stored. In contrast, our proposed cut-through processing can reduce buffer time it starts the vectorized summation immediately after the heads of input frames from all worker nodes arrive. As shown in Fig.1 (b), the vectorized summation starts when the number of buffered frames exceeds a threshold, which is set in the cut-through buffer. Furthermore, the gradients are vectorized and are then summed in parallel to achieve wire-speed Allreduce. Therefore, the proposed Allreduce-specific dataflow hardware accelerator can achieve lower latency compared with conventional store and forward processing.

Gradient casting technique: Reducing gradient-casting latency between GPUs and the NIC at worker nodes is important for achieving low-latency Allreduce. Conventionally, the Eager method has been used for low-latency data transfer because messages are sent as soon as possible without checking receiving buffers. However, since the number of buffers is fixed, it is difficult to perform pipeline data transfer. Moreover, for large messages, it is hard to achieve low latency using Eager due to a lack of buffers. In order to solve this issue, we provide multiple send and receive buffers so that pipeline data transfer can be asynchronously performed. In this way, the gradient-casting latency between GPUs and the NIC is optimized by choosing the appropriate number and size of buffers.

3 EVALUATION

To measure the latency of the proposed architecture, we constructed a system with four worker nodes using Xeon CPU E5-2603 with 64 GB RAM, a Tesla K20 GPU, and a 10G Ethernet Controller and one aggregation node using a VC709 FPGA.

First, the Allreduce latency of the aggregation node for one Ethernet frame (1 KB) was measured with data-quality analyzer.

The measured Allreduce latency was 0.60 μ s, which corresponds to processing speed of 13.6-Gbps in the aggregation node, which is higher than wire speed (10Gbps). Second, the throughput of the proposed system was measured by changing the message size. The throughput was 9.62 Gbps at message sizes of 64 KB or more, which indicated that Allreduce was operated at almost wire speed (Fig. 1(c)). From the above results, we expect that large-message Allreduce at wire speed can be achieved even if the number of nodes increases.

Finally, to compare our architecture with SHARp based on a linear scale of the latency, we linearly extrapolated the data taken from [11]. In [11], SHARp's performance was measured with 128 worker nodes and eight leaf switches. Each node had two 14-core Broadwell CPUs with 256 GB of RAM. ConnectX-4 HCAs were used at 100 Gbps. Fig. 1(d) shows the latency comparison between the proposed architecture and SHARp. For message sizes less than 8 KB, the latency is higher than SHARp because our architecture transfers data from the GPU memory using 10-Gbps Ethernet, whereas [11] considered that data is transferred from the main memory using 100-Gbps InfiniBand. For message sizes over 8 KB, which are common in DL, the latency can be reduced by 65% at the same message size as ResNet-50's gradients (97.5 MB) [3]. The present evaluation is a proof of concept of large-message Allreduce at wire speed for distributed DL.

In the future, we will investigate our proposed architecture using the same measurement conditions as in [11] to evaluate the scalability and Allreduce speed at 100 Gbps. Moreover, we will expand the width of vectorized gradients and optimize the threshold timing in FPGA aiming for over 100 Gbps. Since the data transfer rate in a worker node is limited, we will use new technology such as RDMA over Converged Ethernet. Last but not least, we will build a distributed DL infrastructure using the proposed low-latency Allreduce-specific dataflow architecture.

REFERENCES

- [1] Tal Ben-Num, and Torsten Hoefler. 2018. Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis. arXiv:cs.LG/1802.09941.
- [2] Takuya Akiba, Shuji Suzuki, and Keisuke Fukuda. 2017. Extremely Large Minibatch SGD: Training ResNet-50 on ImageNet in 15 Minutes. *Deep Learning at Supercomputer Scale (NIPS'17 Workshop)*, California, USA. arXiv:cs.DC/1711.04325.
- [3] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J. Dally. 2018. Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training. *6th International Conference on Learning Representations (ICLR'18)*, Vancouver, Canada. arXiv:cs.CV/1712.01887.
- [4] Sylvain Jeaugey. 2017. Nccl 2.0. <http://on-demand.gputechconf.com/gtc/2017/presentation/s7155-jeaugey-nccl.pdf>
- [5] Pitch Patarasuk, and Xin Yuan. 2009. Bandwidth Optimal All-reduce Algorithms for Clusters of Workstations. *Journal of Parallel and Distributed Computing*, 69 (2), 117-124.
- [6] Thomas B. Rolinger, Tyler A. Simon, and Christopher D. Krieger. 2018. An Empirical Evaluation of Allgather on Multi-GPU Systems. *18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid'18)*, Washington, USA.
- [7] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. 2017. Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour. arXiv:cs.CV/1706.02677.
- [8] Xianyan Jia, Shutao Song, Wei He, Yangzihao Wang, Haidong Rong, Feihu Zhou, Liqiang Xie, Zhenyu Guo, Yuanzhou Yang, Liwei Yu, Tiegang Chen, Guangxiao Hu, Shaohuai Shi, and Xiaowen Chu. 2018. Highly Scalable Deep Learning Training System with Mixed-Precision: Training ImageNet in Four Minutes. arXiv:cs.LG/1807.11205.
- [9] Gil Bloch, Diego Crupnicoff, Benny Koren, Oded Wertheim, Lion Levi, Richard Graham, and Michael Kagan. 2015. Aggregation protocol. (Aug. 2015). Patent No. US20170063613A1, Filed Aug. 31th, 2015, Issued Mar. 2nd., 2017.
- [10] Richard L. Graham, Devendar Bureddy, Pak Lui, Hal Rosenstock, Gilad Shainer, Gil Bloch, Dror Goldenberg, Mike Dubman, Sasha Kotchubievsky, Vladimir Koushnir, Lion Levi, Alex Margolin, Tamir Ronen, Alexander Shpiner, Oded Wertheim, and Eitan Zahavi. 2016. Scalable Hierarchical Aggregation Protocol (SHARP): A Hardware Architecture for Efficient Data Reduction. In *Proceedings of the First Workshop on Optimization of Communication in HPC (SC'16 Workshop)*, 1-10, Utah, USA.
- [11] Gil Bloch, Devendar Bureddy, Richard L. Graham, Gilad Shainer, and Brian Smith. 2017. Towards A Data Centric System Architecture: SHARP. *Supercomputing Frontiers and Innovations: an International Journal*, 4 (4), 4-16.
- [12] Mohammadreza Bayatpour, Sourav Chakraborty, Hari Subramoni, Xiaoyi Lu, Dhabaleswar K. (DK) Panda. 2017. Scalable Reduction Collectives with Data Partitioning-based Multi-Leader Design. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC'17)*, Denver, USA.