Distributed Deep Learning with FPGA Ring Allreduce

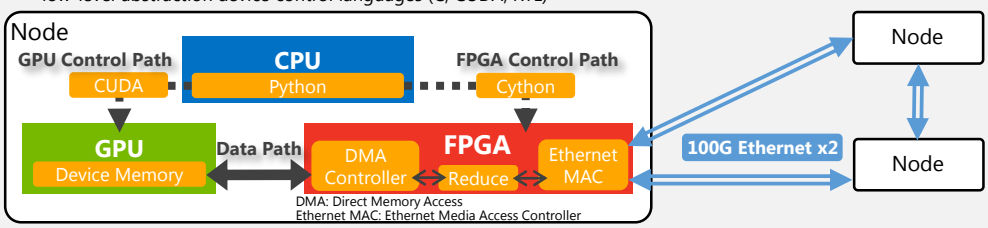
Kenji Tanaka[1], Yuki Arikawa[1], Kenji Kawai[1], Junichi Kato[1], Tsuyoshi Ito[1], Huy Cu Ngo[1], Kazutaka Morita[2], Fumiaki Miura[2], Takeshi Sakamoto[1], Satoshi Shigematsu[1]. 1. NTT Device Technology Laboratories, NTT Corporation. 2. NTT Software Innovation Center, NTT Corporation.

Introduction

- **Deep Learning (DL) becomes supercomputing** when trying to solve advanced challenges such as Climate Analytics [1].
- Among various methods for efficient distributed DL [2], the top three state-of-the-art ImageNet/ResNet-50 training were achieved by utilizing a distributed data-parallel DL with **Ring Allreduce [3, 4] or 2D-Torus Allreduce [5, 6]**.
- However, it is difficult to apply them at large scale because **latency is accumulated at each node due to data moving** to GPU or CPU for Reduce processes.
- Our solution is to use **In-Network Computing** to handle data reduction while it is being transferred in the network, and not to move data to the GPU or CPU during Allreduce [7, 8, 9].

Proposed System Overview

- Since the conventional In-Network Computing hardware can apply to only hierarchical Allreduce [7, 8], **in this work, we propose a new In-Network Computing hardware that can support Ring Allreduce.** Moreover we apply an **Allreduce specific buffering** [9] to process Allreduce with lower latency than conventional buffering.
- In order to minimize communication overhead**, we apply layer-based computing/communication overlap [10, 11] and optimize it for our proposed In-Network Computing system**.
- We also propose a **highly productive software stack** consisting of
  - ✓ a high-level abstraction DL framework (Python)
  - ✓ low-level abstraction device control languages (C, CUDA, RTL)

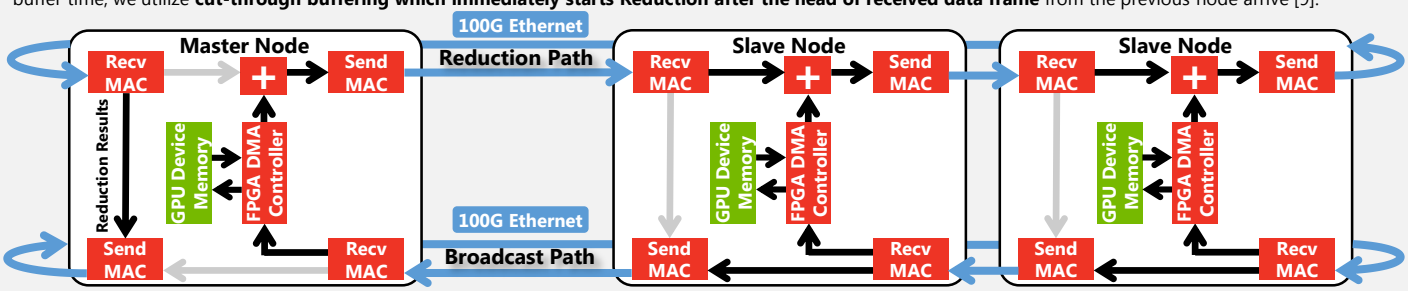


Proposed FPGA Ring-Allreduce System

By utilizing following two techniques to the proposed system, we achieve lower latency than conventional Ring-Allreduce system.

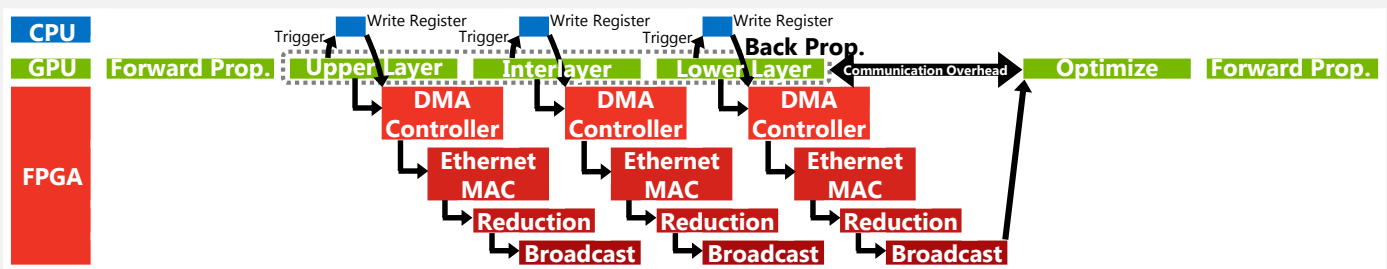
Technique1: Separation of Reduction Path and Broadcast Path.
In conventional system, Reduction and Broadcast in Ring Allreduce are sequentially executed. **In Proposed FPGA Ring-Allreduce system, execution of Reduction and Broadcast are pipelined by designing each Path separately**. Reduction results are send to the Broadcast Path in the master node.

Technique2: Cut-Through Buffering
In slave nodes, model parameters sent from the previous node is added to the model parameters read from a GPU Device memory.
In conventional In-Network Computing system, latency occurs because summation is executed after buffering entire received data frame [8]. To reduce latency due to buffer time, we utilize **cut-through buffering which immediately starts Reduction after the head of received data frame** from the previous node arrive [9].



Parameter-based Computing/Communication Overlap (PCCO)

Distributed DL training systems can overlap Allreduce operations of upper layers with computation of lower layers, reducing dedicated communication overhead [10].
This strategy is called layer-based computation/communication overlap [11].
We apply this strategy and **extended it to operate on each parameter to take advantage of the performance of the proposed FPGA Ring-Allreduce system.**
We call this new strategy parameter-based computation/communication overlap (PCCO).

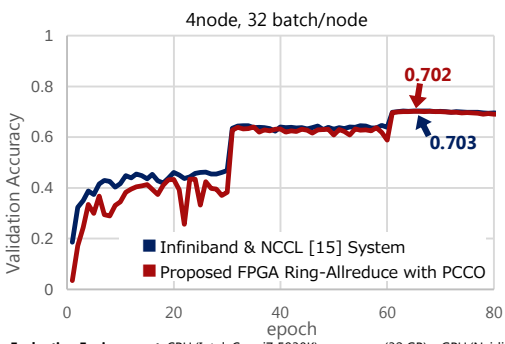


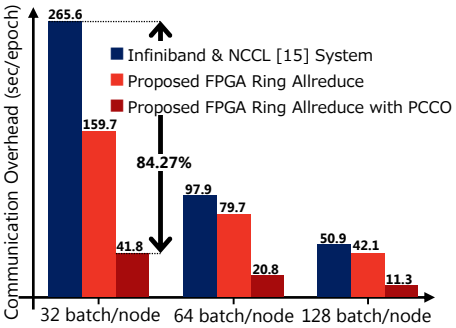
Evaluation Setting

We used Chainer [12] and ChainerMN [13] framework, and trained default ResNet-50 model. However our proposed method can also be used with other Python based DL frame works. Learning rate: exponential shift 0.1 every 30 epochs. Data augmentation: random crop, horizontal flip. Optimization: Momentum SGD (momentum = 0.9). Weight decay: 0.0001 [14].

Evaluation Results

- The results show that we can **reduce the communication overhead by 84.27% at 32 batch /node** without any accuracy degradation.
- In best case, one communication overhead can be reduced to about **2.3 msec at 32 batch /node**.
- Moreover, the total **learning time can be reduced by 7% when using 4 nodes** learning system.





Evaluation Environment: CPU (Intel, Core i7 5930K)、memory (32 GB)、GPU (Nvidia, Tesla P100)、HCA (Mellanox, ConnectX-4 HCA)、IB Switch (Mellanox, Switch IB-2)、Infiniband EDR (MCP1600)、FPGA (Xilinx: VCU118)、DMA Controller (Xilinx, XDMA)、Ehternet MAC (Xilinx, CMAC)、100G Ethernet (100GBASE-SR4)

Conclusion

- It is confirmed that **our system can significantly reduce the communication overhead without deteriorating accuracy** when applying to following cases:
  - ✓Large-scale distributed DL with a large communication load.
  - ✓Distributed DL model with small batch size training.
- Although the current top data is 2-D Torus Allreduce using ASIC in domain specific architecture [5], the result shows that the communication overhead is shorter by applying our proposed method, which **indicates the possibility of In-Network Computing.**
- Our proposed system can be **easily extended to 2-D Torus Allreduce** which improves node scaling efficiency.

Reference: [1] T. Kurth, et. al., 2018. SC'18. Article No. 51., [2] T. Ben-Num, and T. Hoefler. 2018. arXiv:cs.LG/1802.09941., [3] A. Sergeev, and M. Balso, 2018. arXiv:cs.LG/1802.05799., [4] X. Jia, et. al., 2018. arXiv:cs.LG/1807.11205., [5] C. Ying, et. al., 2018. arXiv:cs.LG/1811.06992., [6] H. Mikami, et. al., 2018. arXiv:cs.LG/1811.05233., [7] G. Bloch, et. al., 2017. Supercomput. Front. Innov., [8] G. Bloch, et. al. 2015. Patent NO. US20170063613A1., [9] K. Tanaka, et. al., 2018. SC18. Poster No. 140., [10] H. Zhang, et. al., 2017. arXiv:cs.LG/1706.03292., [11] P. Sun, et. al., 2019. arXiv:cs.DC/1902.06855., [12] S. Tokui, et. al., 2015. Proceedings of Workshop on LearningSys in NIPS., [13] T. Akiba, et. al., 2017. Proceedings of Workshop on ML Systems in NIPS., [14] P. Goyal, et. al., 2017. arXiv:cs.CV/1706.02677., [15] S. Jeaugey. 2017. NCCL https://github.com/NVIDIA/nccl