Silicon Photonic Switch-Enabled Server Regrouping Using Bandwidth Steering for Distributed Deep Learning Training

Ziyi Zhu, Shijia Yan, Madeleine Strom Glick, Min Yee Teh, and Keren Bergman
Department of Electrical Engineering, Columbia University, 500 W 120th St., New York, New York 10027
zz2374@columbia.edu

Abstract: We demonstrate SiP switch-enabled server regrouping using bandwidth steering for performance improvement in distributed deep learning training in a Fat-tree testbed. Our proposed SiP switch control scheme enables scaling to large-scale datacenter and HPC systems.

1. Introduction

Machine Learning and especially deep learning workloads are taking an increasingly larger proportion of computation in datacenter and high-performance computing (HPC) operations. Distributed training has been applied utilizing many server nodes and training models with parallelism to speed up the training process at scale. However, this trend has shifted the performance bottleneck from compute to the network interconnect and an inefficient network can slow down the distributed training process [1]. Past silicon photonic (SiP) based bandwidth steering work [2-3] has shown the capability of mitigating the bottleneck to some degree at the core network level, but prior work lacks a capability of grouping servers to desired top-of-rack switches, and does not improve the job locality.

In this work, we demonstrate the use of SiP switches to perform both server regrouping in the lowest layer and bandwidth steering in higher layers of a hierarchical topology. This architectural approach increases the job locality by regrouping server nodes expecting significant communication with each other and improves network locality by steering bandwidth to higher layers when server regrouping alone is insufficient. In addition, an optimized SiP switch control scheme is presented, which eases the control implementation complexity of the SiP switches for the integration into large scale systems. We built a testbed that consists of 16 servers in a Fat-tree topology and is able to regroup servers and steer network bandwidth to more efficiently execute distributed deep learning workloads. We report 45% to 59% performance improvements depending on different distributed training strategies and test cases.

2. System Architecture and Control Plane

System Architecture Enabling Server Regrouping Using Bandwidth Steering: Figure 1 shows our proposed system architecture that consists of electrical packet switches (EPS), SiP-based optical circuit switches (OCS), and 16 servers arranged in a Fat-tree topology to demonstrate the capabilities of server regrouping and network bandwidth steering. In addition to the SiP OCSs between the top-of-rack (ToR) and aggregation EPSs [2], the SiP OCSs are also inserted between the ToR EPSs and servers. This deployment enables regrouping servers that communicate intensively with each other, and increases job locality by localizing the traffic within a ToR EPS. Figure 1 shows an example. The servers (#5, 6, 9, 10) in orange can be grouped to the same ToR EPS (#2) with the connections (red lines) by reconfiguring the SiP OCSs based on the traffic patterns exposed by the workflows. Distributed deep learning training workflows can show strong communication patterns between certain server nodes. Observations and experimental results are presented in Section 3. In addition, higher layer bandwidth steering between the ToR and aggregation switches can still enhance overall system optimization. Due to the limited port count of the OCS, it is not feasible to realize all-to-all ToR connectivity for systems at scale. For those cases where the traffic cannot be localized and has to travel through aggregation and core EPSs, SiP-enabled bandwidth steering above the ToR switches can then be applied to relieve network congestions at higher layers and optimize network topology to better fit traffic demands.

SiP Switches and Optimized Control Scheme: SiP switches with CMOS compatible manufacturing processes are promising for power-efficient, low fabrication cost interconnects. Our proposed control scheme can apply to any primary switching cells (Mach-Zehnder interferometers, MEMS-actuated couplers, and microring resonators (MRRs)), which have been shown high scale integration [4]. We optimize our control scheme of the SiP switches [3] to provide a clean network control interface, and ease the control implementation complexity for future deployment at scale. Figure 2(a) shows the overall network control plane including a Ryu-based SDN controller that manages the flow tables of the EPSs and a TCP/IP client that sends reconfiguration requests to the SiP OCS subsystem. In the subsystem, shown in Fig. 2(b), a network controller is built upon Xilinx ZCU 106 board and leverages a Linux/Ubuntu operating system running on the ARM processors to hold a TCP/IP server that responds to the reconfiguration requests. Along with other control algorithms such as calibration and thermal stabilization, all the control logic can be implemented in the software and this dramatically simplifies the control implementation

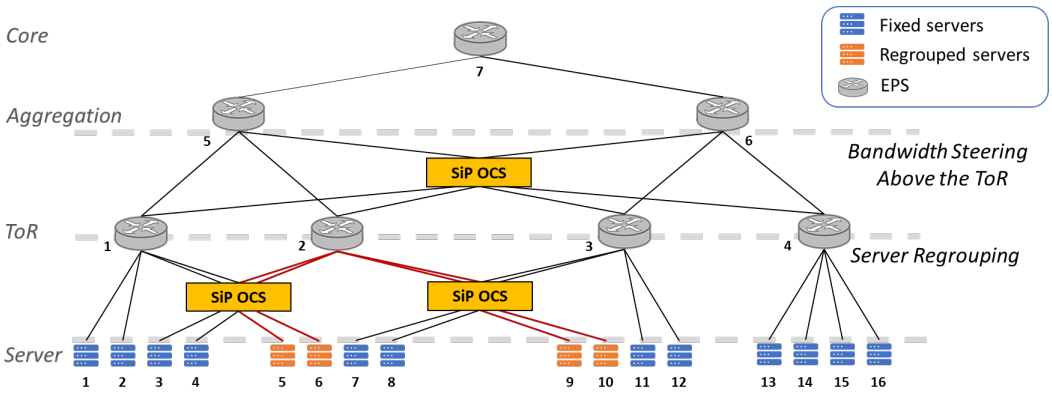

Fig. 1. System architecture demonstration with 16 server nodes arranged in the Fat-tree topology to show SiP switch-based server regrouping and bandwidth steering above the ToR for a small-scale demonstration

complexity. Using digital to analog converters (DACs) and analog to digital converters (ADCs), the switch controller implemented on a Terasic TR4 FPGA board provides correct bias voltages to the switching cells in the packaged SiP switches sitting on the fan-out PCBs. The network and switch controller are connected using GPIOs and the interface from switch controller to the fan-out PCB is using SMA cables. Figure 2(c) shows the network controller and switch controller FPGA evaluation boards and the SiP switch fan-out PCB.

3. Experimental Testbed and Results

We study the proposed SiP-enabled server regrouping using bandwidth steering for both synchronized ring all-reduce and asynchronized parameter server training algorithms. We use distributed communication package in PyTorch and implement both synchronized and asynchronized training algorithms for the VGG neural network [5]. The training jobs run across 4 server nodes (#5, 6, 9, 10) for the ring all-reduce algorithm and run across 3 server nodes (#5, 9, 10) for the parameter server algorithm with a NVIDIA M40 GPU in each server. For the remaining 12 servers, we run skeletonized version of the Gyrokinetic Toroidal Code (GTC) benchmark applications [3] as the background traffic across the network. There are three network configuration scenarios: baseline (no dynamic reconfigured links), server-regrouping (#9, 10 regrouped to EPS2), and server-regrouping with bandwidth steering above the ToR (#9 regrouped to EPS2 and a steered link between EPS3 to EPS5), which are shown in the Fig. 3(a), (b) and (c), respectively. 10G SFP+ optical transceivers are used for reconfigured links connected to SiP OCSs, and static links are using 10G SFP+ electrical transceivers. Bidirectional 4x2 SiP MRR-based OCSs are adequate for the experimental setups in Fig. 3(b) and (c). Detailed physical implementation can be found in Ref. [6].

Figure 3(d) shows the throughput of EPS1 to EPS5, EPS5 to EPS7, incoming traffic to server 9 and to server 10 when the ring all-reduce algorithm is being processed across the network. These throughput curves are sufficient to demonstrate the results. The green curve in the baseline diagram indicates the traffic at the core level is aggregated by the background GTC traffic (yellow) and the ring all-reduce training traffic (red or blue). The training process is suppressed by the background GTC traffic and it takes around 4387 s to train the VGG network for 1 epoch for the baseline. For the regrouped server nodes, the training job's traffic is constrained within the EPS2, such that the

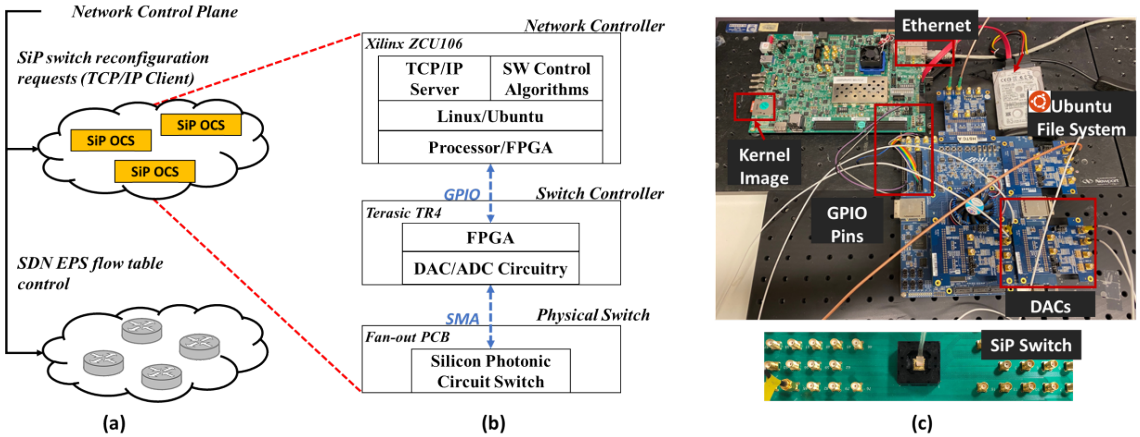

Fig. 2. (a) Optimized network control scheme, (b) SiP OCS subsystem and its controllers, (c) Implementation of the network and switch controllers, and the fanout PCB holding a packaged SiP switch.

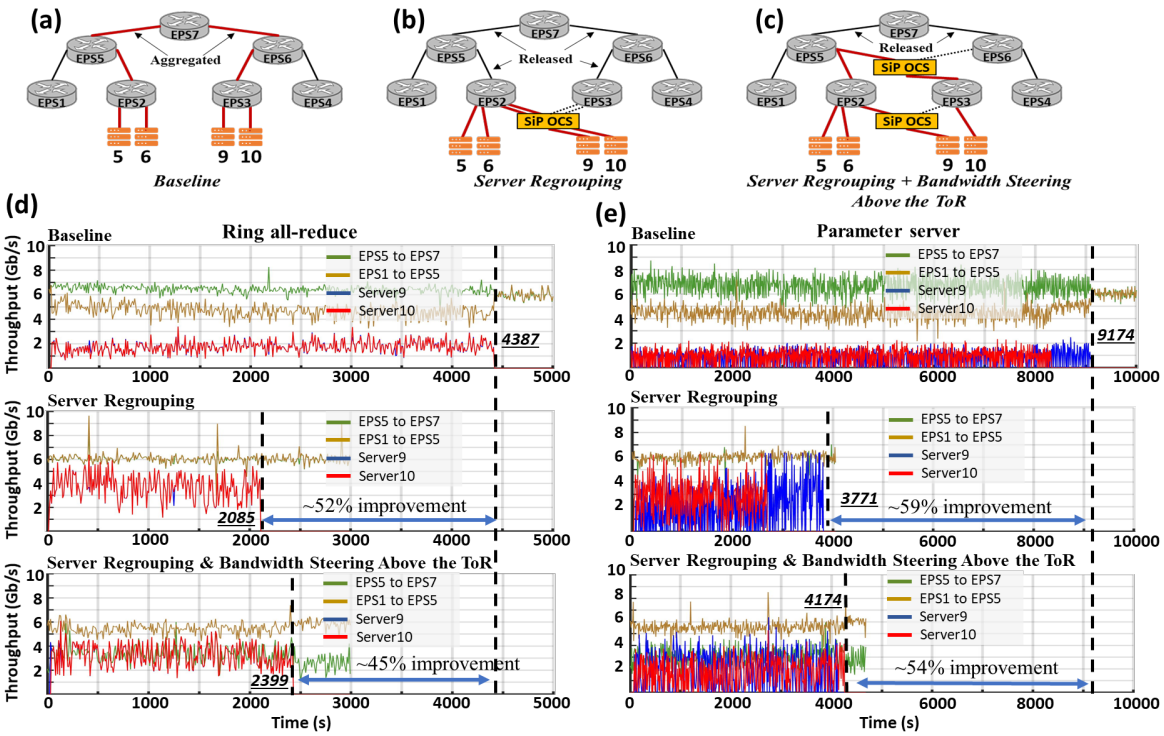

Fig. 3. Network configurations for (a) baseline, (b) server-regrouped and (c) server-regrouped and above ToR bandwidth-steered Fat-tree topologies. Only the 4 servers running deep learning applications are shown in the figure. (d) Throughput of the links of interests for the three cases (a), (b), and (c) for the ring all-reduce algorithm of the training process. (e) Throughput of the links of interests for three cases for the parameter server algorithm.

communication bandwidth for the ring all-reduce training processes is restored. The red curve in server regrouping diagram in Fig. 3(d) shows a 4 Gb/s bandwidth on average. This enhancement results in a 2085 s execution time with a 52% performance improvement. For the case where both server regrouping and bandwidth steering above the ToR are applied, we observe a 2399 s execution time with a 45% performance improvement. It is expected to see the servers #5, 6, 9, and 10 experience the same communication bandwidth due to the synchronized training's nature.

Figure 3(e) shows the performance improvements by applying server regrouping and bandwidth steering above the ToR for parameter server training algorithm. Similar performance improvements are observed for the server regrouping only and the server regrouping with bandwidth steering above the ToR as 59% and 54%, respectively. We should note that parameter server training is an asynchronized training, and it is reasonable that the two worker nodes finish their individual training job at different time stamps as indicated by the red and blue curves in Fig. 3(e).

4. Conclusion

We demonstrate SiP switch-enabled server regrouping using bandwidth steering are applicable to distributed deep learning training and show 45% to 59% performance improvement when operating on our testbed. The proposed SiP switch control scheme makes the SiP switches feasible to large-scale systems. In future work, we will further integrate our SiP switches and apply server regrouping using bandwidth steering to other systems such as Spark.

Acknowledgments. This work was supported by ARPA-E ENLITENED Program (project award DE-AR00000843) and the National Security Agency (NSA) Laboratory for Physical Sciences (LPS) Research Initiative (R3/NSA) (Contract FA8075-14-D-0002-0007, TAT 15-1158).

5. References

[1] Peng, Yanghua, et al. "A generic communication scheduler for distributed dnn training acceleration." Proceedings of the 27th ACM Symposium on Operating Systems Principles. 2019.
[2] Michelogiannakis, George, et al. "Bandwidth steering in HPC using silicon nanophotonics." Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. 2019.
[3] Shen, Yiwen, et al. "Accelerating of high performance data centers using silicon photonic switch-enabled bandwidth steering." ECOC, 2018.
[4] Cheng, Qixiang, et al. "Photonic switching in high performance datacenters." Optics express 26.12 (2018)
[5] Simonyan, Karen, et al. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).
[6] Zhu, Ziyi, et al. "Photonic Switched Optically Connected Memory: An Approach to Address Memory Challenges in Deep Learning." Journal of Lightwave Technology 38.10 (2020): 2815-2825.