Distributed Deep Learning Training Using Silicon Photonic Switched Architectures

Cite as: APL Photonics (in press) (2021); https://doi.org/10.1063/5.0070711
Submitted: 08 September 2021 • Accepted: 06 February 2022 • Accepted Manuscript Online: 07 February 2022

Ziyi Zhu, Min Yee Teh, Zhenguo Wu, et al.

Distributed Deep Learning Training Using Silicon Photonic Switched Architectures

Ziyi Zhu,[1] Min Yee Teh,[1] Zhenguo Wu,[1] Madeleine Strom Glick,[1] Shijia Yan,[1] Maarten Hattink,[1] and Keren Bergman[1]
Department of Electrical Engineering, Columbia University, New York, New York 10027, USA

(*Electronic mail: zz2374@columbia.edu)

(Dated: 10 January 2022)

The scaling trends of deep learning models and distributed training workloads are challenging network capacities in today's datacenters and HPCs. We propose a system architecture that leverages silicon photonic (SiP) switch-enabled server regrouping using bandwidth steering to tackle the challenges and accelerate distributed deep learning training. In addition, our proposed system architecture utilizes a highly integrated OS-based SiP switch control scheme to reduce implementation complexity. To demonstrate the feasibility of our proposal, we built an experimental testbed with a SiP switch-enabled reconfigurable fat tree topology and evaluated the network performance of distributed ring all-reduce and parameter server workloads. The experimental results show up to 3.6× improvements over the static non-reconfigurable fat tree. Our large-scale simulation results show that server regrouping can deliver up to 2.3× flow throughput improvement for a 2× tapered fat tree and a further 11% improvement when higher-layer bandwidth steering is employed. The collective results show the potential of integrating SiP switches into datacenter and HPC systems to accelerate distributed deep learning training.

I. INTRODUCTION

Deep learning (DL) is a branch of machine learning that has become a major driving force behind the progress in artificial intelligence applications such as image classification,[1] natural language processing,[2] and recommendation systems.[3] The demand for better DL models has resulted in a rise of more complex models that support larger dataset sizes to improve these deep neural networks.[4,5] The typical approach to speed up the training process of these larger DL models is parallelization using many GPU-equipped nodes,[6–8] which requires a high-bandwidth interconnect to support the communication requirements between training devices.[9] DL workloads are taking a large proportion of the computation in today's high-performance computing (HPC) operations, and observation has shown that the demand is dramatically growing in datacenters.[10] These trends have shifted the performance bottleneck from the compute to the network interconnect due to system fragmentation (applications often receive an allocation on a set of distant and non-contiguous nodes). This places a tremendous challenge on interconnect designs to provide high bandwidth and low latency networking to sustain the continual growth of these hardware-driven deep learning applications.

These challenges present a unique opportunity for flexible photonic switched networks that have the capabilities to perform topology reconfiguration and have motivated much research to explore reconfigurable network architectures based on optical circuit switches (OCSs). These OCS-based architectures employ various different technologies, such as 3D MEMS,[11,12] silicon photonic switches,[13] wireless transceivers based on free space optics,[14,15] RotorSwitch,[16] and tunable lasers.[17] Early architectures of reconfigurable network such as Helios[12] used OCSs to build a hybrid optical/electrical architecture to serve bandwidth-bound large flows using the OCS network while serving the latency-bound small flows with static electrical packet switches (EPSs). Later works such as ProjecToR[14] and RotorNet[16] used customized switching prototypes to build flatter network topologies where the top-of-rack (ToR) switches are directly connected with a single layer of OCSs for higher energy efficiency. Meanwhile, silicon photonic (SiP) switches have also been proposed as another solution that could provide power-efficient high bandwidth scaling at low fabrication cost. Flexfly[13] and Flexspander[18] placed SiP switches in between clusters/groups of EPSs to achieve better scalability.

In addition to applying these reconfigurable network architectures to traditional HPC workloads, various works in the literature have explored employing them under distributed machine learning settings as well. Truong *et al.*[19] have proposed using a hybrid electrical/optical architecture, similar to Helios,[12] to serve long-lived DL training communications using the OCS network while using the EPS network for smaller messages. Evaluation with real DL workload shows significant communication speedup when employing the hybrid architecture. Lu *et al.*[20] have proposed to build a hierarchical network, similar to Flexfly,[13] for distributed machine learning applications. Results show that X-NEST outperforms RotorNet[16] across different DL workloads and performs similarly to fat trees with fewer hardware components.

While many reconfigurable network architectures have been explored in the past, prior work has typically proposed architectures with reconfigurability at a single network layer (e.g. between ToR and aggregation EPSs[21] or between dragonfly groups[13]). In this work, we propose our reconfigurable SiP architecture[22] that uses SiP switches between servers and ToR, and between ToR and aggregation EPSs in a fat tree topology. This architecture introduces two unique network functionalities: (1) server-regrouping between servers and ToR switches to recover job-level traffic locality, and (2) bandwidth-steering between ToR and aggregation layers to maximize traffic retention at the lower fat tree layers. We demonstrate an improvement in overall network

performance for distributed ring all-reduce and parameter server deep learning training algorithms. An optimized SiP switch control scheme is presented to simplify the control implementation complexity and to achieve better integration of the SiP switches into large-scale systems. In our experimental hardware testbed,[22] we present new results demonstrating that regrouping servers and steering network bandwidth can result in more efficient execution of the distributed deep learning workloads. We report a 1.9× to 3.6× performance improvements depending on different distributed training strategies and test cases. In this paper we also present new system-scale simulations. We perform server regrouping and bandwidth steering on a large-scale tapered fat tree with 1024 compute nodes. Our simulation results show that server regrouping can deliver up to 2.3× flow throughput improvement for a 2× tapered fat tree and a further 11% improvement when higher-layer bandwidth steering is applied.

II. SILICON PHTONICS FOR OPTICAL CIRCUIT SWITCHING

Optical circuit switching offers a promising approach to reconfigure the interconnect in order to (1) regroup a set of distant and non-contiguous nodes and (2) steer bandwidth at higher layers for efficiency. Depending upon underlying traffic patterns of the nodes at different times, optimized topology connections can be dynamically formed on demand.

Commercially available technologies, such as microelectromechanical systems (MEMS),[23] beam-steering,[24] and liquid crystal on silicon (LCOS),[25] can be used to implement the reconfigurable network. However, there are still challenges to achieve commercial adoption. The rigorous calibration and the installation of discrete components introduce significant complexity and result in high cost per port. Similarly, arrayed waveguide grating routers (AWGRs)[26] based interconnects usually require higher cost tunable wavelength transceivers that add complexity and additional power consumption in broadcast and select type architectures. For low-cost datacenter adoption, lithography-based photonic integration technologies hold great promise for large-scale optical integrated switch fabrics with smaller device footprint, and reduced assembly and calibration overheads.

The silicon photonics platform, in particular, leverages the mature and widespread CMOS manufacturing infrastructure, and SiP switches are promising for the dynamic topology reconfiguration with better power efficiency, lower cost-per-port, smaller footprint, and the potential for nanosecond range dynamic switching.[27-31] However, there are several technical challenges to address in this platform, specifically loss through the switch, polarization dependency, thermal stability, and switch radix scalability. Research works have been reported to address these challenges, and the primary switching cells that are being explored are Mach-Zehnder interferometers (MZIs), microring resonators (MRRs), and MEMS-actuated couplers.

MZI switching circuits of 32 × 32 connectivity have been realized using thermo-optic (T-O) phase shifters with 6.1 dB on-chip loss.[32] To overcome the polarization dependency, a polarization-diversity SiP MZI switch was further developed.[33] The current record for the T-O MZI switch is a 64 × 64 implementation in Bene topology.[34] For fast electro-optic (E-O) switching, carrier-injection based PIN junctions are employed. 16 × 16 and 32 × 32 E-O MZI-based switches were proposed by Lu *et al.*[35] and Qiao *et al.*[36] Performance, however, can be limited due to the high insertion loss. Gain-integrated switches for lossless operation can be applied to overcome this challenge.[37]

MRR based devices show ultra-compact and energy-efficient potentials for optical switching. Recent work has demonstrated 8 × 7 cross-bar,[38] 8 × 8 Omega,[39] 4 × 4 switch-and-select architectures.[40] Add-drop filters assembled in a 1-D bus structure can act as spatial (de)multiplexers.[41] Thermal stabilization[42,43] is necessary for MRR based switches to address wavelength drifts due to the thermal dependencies to the varying ambient temperature.

The largest-scale SiP switch fabric reported to date is the MEMS-actuated cross-bar switch with 240 × 240 connectivity, which consists of a 3 × 3 array of identical 80 × 80 switch blocks.[44] Maximum on-chip loss of 9.8 dB was reported. Multilayer bus waveguides can be used for eliminating waveguide crossings to reduce insertion loss and for addressing polarization sensitivity.[45] Recent work has shown successful fabrication of SiP MEMS using a commercial foundry with reduced driving voltage down to 9.45V.[46]

More detailed discussions on the photonic switching technologies in datacenter/HPC systems can be found in the reviews.[27-29] We note that SiP switches are promising for optical switching in datacenter/HPC rack-to-rack applications; however, the loss should be further reduced before being deployed in practice. Approaches, such as (1) integration with semiconductor optical amplifiers (SOAs), (2) improvement of coupling loss, and (3) progress on individual component to have a better loss performance, are being taken to further reduce the loss of silicon photonic switched architectures.

III. SYSTEM ARCHITECTURE AND SIP SWITCH CONTROL

A. System Architecture

Distributed deep learning training workflows, including data parallelism and model parallelism, show strong communication patterns with high-bandwidth requirement between server nodes. We demonstrate our proposed system architecture on synchronized ring all-reduce[6] and asynchronized parameter server[47] data-parallel techniques. Figure 1(a) illustrates our proposed system architecture. It consists of EPSs, SiP-based OCSs, and servers to demonstrate the capabilities of server regrouping and network bandwidth steering. By using SiP OCSs between servers and ToR EPSs, this architecture allows servers with intense communication requirements to be grouped locally within the same ToR-switch, thereby reintroducing traffic locality between physically distant servers. An example of regrouped servers is shown in Fig. 1(b). With the demand of traffic between servers (in orange), the SiP OCS is capable of dynamically changing the connectivity and connecting the regrouped servers (orange) under the same ToR EPS. Due to the limited port count of the SiP OCS, it is not feasible to realize all-to-all ToR connectivity for systems at scale. Therefore, SiP OCSs are also inserted between the ToR and the aggregation layers. When a partial server regrouping is performed, bandwidth steering

will be applied to reduce contentions at higher layers. An example is shown in Fig. 1(c). Bandwidth steering above the ToR is used to relocate connections from the ToR to desired aggregation EPSs. The overall system architecture essentially reconstructs locality of connection and optimizes topology to better fit network traffic demands.

B. SiP Switches and Control

Our proposed architecture and control scheme are agnostic to the choice of SiP switching devices. We optimized our control scheme of the SiP switches and fabricated custom DAC cards to provide a software-based network control interface, and to ease control implementation complexity and achieve better integration. Depending on the traffic patterns of the distributed deep learning training, the overall network controller reconfigures network topology on demand. Figure. 2(a) shows our overall network control plane. It consists of (1) a Ryu-based SDN controller that manages the flow tables on the EPSs; (2) a TCP/IP client program that sends new reconfiguration requests to the SiP OCS subsystem as shown in Fig. 2(b). In the subsystem, the SiP network controller is built upon a Xilinx ZCU 106 board. We leverage Xilinx PetaLinux to build a kernel image stored in a SD card and boot a Linux/Ubuntu operating system (OS) from a hard-drive. A TCP/IP server program running on the ARM processors responds to the reconfiguration requests from the overall network controller. Control algorithms such as calibration and thermal stabilization can be not only implemented in the software for simplicity, but also implemented as hardware logic in the field programmable gate array (FPGA) for control speed. A custom 80-channel DAC daughter card was fabricated to demonstrate a path toward large-scale system integration. Using the DAC daughter cards, the switch controller implemented on a Terasic TR4 board provides correct bias voltages to the switching elements in the packaged SiP switches. The SiP network and switch controllers are connected using GPIOs and the interface from SiP switch controller to the fan-out PCB uses SMA cables. Figure. 2(c) shows the physical devices.

IV. TESTBED

To run distributed deep learning workloads and demonstrate the network improvements of our proposed system architecture, we built a 16-node HPC/datacenter testbed[22] as shown in Fig. 3(a). We used 4 GPU servers (in orange) equipped with NVIDIA M40 GPU to run ring all-reduce and parameter server training algorithms across them. The other 12 CPU servers (in blue) are used for running other applications to generate background traffic across the network. The EPSs are virtually partitioned from an OpenFlow-enabled PICA8 packet switch with 10G SFP+ ports. We use a 1×2 and a 1×4 MRR-based OCSs to perform server regrouping and bandwidth steering above the ToR EPSs. For the fully server-regrouped case (defined as case #1), the SiP switches are connected to servers #9 and #10, and two separate ports on EPS #2 and #3, respectively. For the case (defined as case #2) where the server regrouping is partially performed and bandwidth is steered above the ToR, the SiP switches are connected to server #9 and a port on EPS #3, and individual ports on EPS #3, #6, #2, #5 respectively. In this case, server #10 is connected to EPS #3 without going through the SiP OCSs. 10G SFP+ optical transceivers are used for reconfigured links and static links are using 10G electrical transceivers. A detailed experimental setup is shown in Fig. 3(b). Two SFP+ transceivers with wavelengths at 1554.94 nm (λ_5) and 1556.55 nm (λ_6) are used for server #9 and #10 to transmit data to EPS #3 and EPS #2 (in case #1) or for server #9 and EPS #3 to transmit data to EPS #3, #6, and EPS #2, #5 (in case #2). Four SFP+ transceivers, with wavelengths at 1545.32 nm (λ_1), 1546.92 nm (λ_2), 1553.33 nm (λ_3) and 1554.94 nm (λ_4), are used for the opposite direction. The polarization controllers (PC) are used to maximize the optical power being coupled into and out of the SiP chips. An erbium doped fiber amplifier (EDFA) is necessary to compensate the loss due to the grating couplers of the SiP switch chips. We note that the approaches described in Section II can potentially reduce the loss and allow the system to work without EDFAs. Detailed SiP switching characteristics can be found in the previous work.[48] The SiP network controller FPGA board (ZCU 106) receives configuration requests from the overall network controller and triggers the SiP switch controller (TR4). The switch controller will then configure each MRR by tuning the resonance with bias voltage. A photograph of EPSs, CPU servers, GPU servers, SiP switches, SiP network controller, and SiP switch controller is shown in Fig. 3(c). We note that the reconfiguration speed limitation is the transceiver locking and the EPS polling time[49] at the millisecond scale. This is a negligible effect in the current architecture due to the fact that the topology reconfiguration only happens before an application starts. Thermal drift of the MRR-based switches could lead to system performance degradation, and thermal stabilization[42,43] should be applied to address this issue before the deployment of MRR-based architectures in future datacenter/HPC networks. The experiments described in this work take place in a thermally stable environment.

V. EXPERIMENTS AND RESULTS

We used the distributed communication package in PyTorch,[50] which enables the processing groups for each of the workers used in the synchronized training and the parameter server and workers in the asynchronized training. 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 – parameter server and #9, #10 - workers) for the parameter server algorithm. For the remaining 12 servers, we run skeletonized version of the Gyrokinetic Toroidal Code (GTC) benchmark applications[51] as the background traffic across the network. There are two test cases. (1) Assuming OCS port count is sufficient for server regrouping, we use baseline (no dynamic reconfigured links) to compare with server-regrouping (servers #9, #10 regrouped to EPS #2) as our test case #1. (2) For test case #2, a partial server-regrouping (only server #9 regrouped to EPS #2) compares server-regrouping with bandwidth steering above the ToR

(server #9 regrouped to EPS #2 and a steered link from EPS #3 to EPS #5). Simplified diagrams for the two test cases are shown in the Fig. 4(a) and (b), respectively.

Figure 5 plots the throughput of incoming traffic to servers #9 and #10 (blue and red), from EPS #5 to EPS #7 (green), and from EPS #1 to EPS #5 (yellow) for various training strategies and test cases. The plotted links are sufficient to show the network performance of deep learning workloads. The neural network is VGG[1] for image classification and the dataset is imagenette.[52] Figure 5(a) shows the results for the synchronized training. For test case #1, Fig. 5(a) left, the green curve in the baseline diagram (top left) 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 approximately 5341 s to train the VGG network for 1 epoch for the baseline. For the regrouped case, server #9 and server #10 are regrouped to EPS #2, and the training job's traffic is within EPS #2, such that the communication bandwidth for the ring all-reduce training processes is restored. The red curve in the server regrouping diagram (bottom left) in Fig. 5(a) shows a 5 Gb/s bandwidth on average for the ring all-reduce algorithm with a 72% difference in execution time which corresponds to a 3.6× network performance improvement. For test case #2, Fig. 5(a) right shows the results for server regrouping with limited OCS port count and when bandwidth steering above the ToR is applied. We observe a 5709 s execution time (in Fig. 5(a) top right) when only server #9 is regrouped to EPS #2 and no bandwidth is steered between ToR and aggregation EPSs. In comparison, server regrouping and bandwidth steering above the ToR (Fig. 5(a) bottom right) provides a 61% difference in execution time which corresponds to a 2.6× network performance improvement due to the fact that the deep learning training flows are not going through the core layer of the network. Figure 5(b) shows the performance improvements for the parameter server training algorithm. Similar performance improvements are observed for the server regrouping and the server regrouping with bandwidth steering above the ToR as 67% and 47% in execution time differences (3.0× and 1.9× improvements), 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. 5(b) right. The comparative experimental results can be found in Table I.

VI. SYSTEM-SCALE EVALUATION

We study the scalability and network performance of the proposed system architecture on two distributed deep learning training algorithms: (1) ring all-reduce and (2) parameter server. For each of the workloads, we analyze how server regrouping and bandwidth steering affect the performance of large-scale networks with various tapering ratios. In addition to using uniformly mapped jobs as a performance upper bound, we also simulate non-uniformly mapped jobs since past work[21] has shown that frequent system fragmentations in high performance systems could make the job mapping largely non-uniform. For the purpose of this work, which is to show the performance improvement of the proposed strategies, we assume that server regrouping and bandwidth steering strategies for non-uniform job placements happen before a workload starts in the simulation and therefore has no packet loss due to reconfiguration. We plan to add this reconfiguration functionality to the Netbench simulator as future work.

A. Simulation Setup

We use Netbench,[53] a discrete event-driven packet-level simulator, to evaluate network performance at scale.
The simulated network is a tapered 3-layer fat tree constructed using EPSs with 32 bidirectional ports. We assume the link bandwidth to be 100 Gb/s. The fat tree topology contains 1024 compute nodes distributed among 4 pods. Each pod consists of 16 ToR switches each connected to 16 servers, with a total of 256 servers per pod, as shown in Fig. 6. The tapering in our fat tree refers to the difference in bisection bandwidth between any two levels of the tree, as described by Michelogiannakis *et al*.[54] We taper the fat tree at the aggregation and core layers to emulate production networks. For our simulation, we taper the aggregation layer by 2×, 4×, 8×, and 16× while tapering the core layer with a constant 8× with respect to the aggregation layer. We use Equal-Cost Multi-Path (ECMP) routing on both the bandwidth-steered and static baseline fat trees with per-packet load-balancing.

We test server regrouping on both ring all-reduce and parameter server types of traffic workload in our simulation. The ring all-reduce traffic and parameter-server traffic each contains 32 and 16 compute nodes per job, respectively. Under uniform job placement, each process is mapped sequentially onto each server. However, job placement might not always be uniform in real high performance systems. Past work[21] has shown that applications are often placed on a set of distant and non-contiguous nodes, resulting in system fragmentation. In order to verify the effectiveness of server regrouping at large scale, we introduce a mixed job mapping strategy to generate a more adversarial traffic pattern to the static fat tree. This mapping hinges on the ratio of intrapod to interpod traffic. For our simulation, we set the ratio so that half of the nodes in each pod communicates with other nodes in the same pod, and the other half would communicate with nodes in the other pods. The mapping within each half is also shuffled to introduce randomness in the mapping.

B. Server Regrouping and Bandwidth Steering

Since the usage of small-radix OCSs could impose extra physical constraints on the topology-wiring problem,[55] we first make the assumption that given k ToRs with k downlinks each, the OCS layer in between the server and the ToR layer is comprised of a single large-radix OCS with k2 ports. The OCS can be viewed as a k2 by k2 fully non-blocking switch ca- pable of reaching and regrouping servers across all pods.

This assumption is necessary for the purpose of our work which is to evaluate the performance and scalability of server re- grouping strategy. Experiment with this initial assumption can serve as a performance upper bound for our experiments with smaller radix OCSs. To evaluate the network performance of smaller-radix OCSs, we split the single large-radix OCS into two OCSs with half the radix (each connecting two pods) and into four OCSs with a quarter of its original radix (each connecting one pod).

The server regrouping heuristic for the ring all-reduce traffic is similar to that of the parameter-server, and it can be split into two substeps:

1. Group jobs that only contain servers in the same pod. For these jobs, group as many servers under the same ToR switch as possible.
2. Group jobs that contain servers in different pods. If the OCS port count is big enough to reach all the servers in the same job, then group these servers under the first available ToR. If not, group as many servers in the same job as possible under a single ToR for each pod that contains these jobs.

When server regrouping falls short, bandwidth steering at the ToR to aggregation layer can be applied together with server regrouping to further improve the performance. We assume a single large OCS between the aggregation and core layers as a performance upper bound. Smaller radix OCS could also be used here similar to previous works[13,21] for bandwidth steering at higher layers depending on the reconfiguration requirements and tapering ratios. Bandwidth steering above the ToRs is configured so that the number of flows traversing the core layer is minimized. This is done by first regrouping the servers that have the same destination pod under the same ToR switch within each pod and then wire the OCS such that the two ToRs with the heaviest communication are connected by the same aggregation switch. For these simulations, we assume that the topology is reconfigured according to the described server regrouping or bandwidth steering strategy before a workload starts.

C. Results

In this section we evaluate the performance of server re-grouping and higher-layer bandwidth steering in large-scale systems. Fig. 7 shows the simulation results for average flow throughput (for both intrapod and interpod flows) as the job-mapping and topology design vary for all ToR-to-aggregation tapering ratios. Uniform job placement corresponds to the case where jobs are mapped sequentially onto the servers in the topology without any shuffling. It achieves the highest average throughput since the communicating nodes are placed close to each other, so the tapered core layer links are not congested by interpod flows. It serves therefore as the performance upper bound for all the server regrouping schemes in our experiments. Indeed, when servers are regrouped with one large-radix OCS (RG (#OCS = 1)), results for both ring all-reduce workload in Fig.7(a) and parameter server workload in Fig.7(b) show matching behavior with the uniform case. Note here that the mean flow throughput for parameter server is much lower than that of ring all-reduce because parameter server jobs contain many more flows in each iteration, resulting in much higher link congestion. On the other end, baseline corresponds to the case where jobs are randomly mapped onto servers across different pods without server regrouping.

RG (#OCS = 2) and RG (#OCS = 4) correspond to the cases where the servers are regrouped with two and four OCSs respectively. Since we have four pods in total, the former case with two OCSs would mean that each OCS connects to servers and ToRs in two pods. Similarly, the latter case with four OCSs means that each OCS is responsible for connecting servers and ToRs in only one pod. For the cases where servers can only be partially regrouped, we still observe improvement from the baseline case, especially under higher tapering. Although not all servers can be regrouped, other properly regrouped servers have already reduced the amount of traffic traversing the tapered layer by an appreciable amount.

On top of server regrouping, we can further improve the performance of the network by employing bandwidth steering in the ToR-to-aggregation layer to alleviate congestion at the top layers. We see that for the cases with both server regrouping and higher-layer bandwidth steering (BS), the performance is consistently higher than the purely regrouped cases, especially at lower tapering ratio. For higher tapering ratios, the number of available reconfigurable links be- tween each ToR switch and the aggregation switches is limited, which limits the benefits of higher-layer bandwidth steering.

Simulation results show that our approach improves the network performance for the ring all-reduce and parameter server workloads at scale. We only consider results with two or more OCSs as the RG (#OCS = 1) case is the same as our performance upper bound. We found that for 2× tapering ratio, server regrouping alone can improve the throughput performance from the baseline by 2.3×, and higher-layer bandwidth steering can provide up to 11% further improvement (2.5× improvement in total from the grey bar to the pink bar). For higher tapering ratios, the total improvements can reach up to 8.6× for ring all-reduce and 2× for parameter server, respectively. Table I shows the performance improvements of different architectures including both experiments and simulations.

VII. CONCLUSIONS

In this work, we have shown a reconfigurable datacenter/HPC system architecture using SiP switches to accelerate distributed deep learning training workloads. We used VGG as the primary workload for our experiment, but our proposed architecture could work on a wide range of distributed machine learning applications that employ ring all-reduce or parameter-server types of collective operations. Using silicon photonic switches, we introduce topological reconfigurability at two network levels to achieve two optimization goals: (1) server-regrouping by introducing SiP OCSs between the ToR switches and servers, and (2) bandwidth-steering by introducing SiP OCSs between the ToR and aggregation layers. We demonstrate our proposed architecture using a physical testbed with 16 nodes arranged in a fat tree topology and show up to 3.6× network improvement. At system scale, server regrouping delivers a

2.3× flow throughput improvement and higher-layer bandwidth steering provides a further 11% improvement in a 2× tapered fat tree. These results show the proof-of-concept functionalities of our proposed system architecture and the potential of integrating SiP switches in datacenters and HPCs to improve DL training performance.

TABLE I
EXPERIMENT AND SIMULATION PERFORMANCE MEASUREMENTS

Configurations	Improvements
Server regrouping compared to baseline for ring all-reduce training (experiment)	3.6×
Server regrouping with bandwidth steering compared to server regrouping with limited point count for ring all-reduce training (experiment)	2.6×
Server regrouping compared to baseline for parameter server training (experiment)	3.0×
Server regrouping with bandwidth steering compared to server regrouping with limited point count for parameter server training (experiment)	1.9×
Server regrouping using two OCSs on 2× tapered fat tree compared to baseline for ring all-reduce training (simulation)	2.3×
Server regrouping using two OCSs with bandwidth steering on 2× tapered fat tree compared to baseline for ring all-reduce training (simulation)	2.5×
Server regrouping using two OCSs on 2× tapered fat tree compared to baseline for parameter server training (simulation)	1.2×
Server regrouping using two OCSs with bandwidth steering on 2× tapered fat tree compared to baseline for parameter server training (simulation)	1.4×

ACKNOWLEDGMENTS

This work was supported in part by ARPA-E ENLITENED Program (project award DE-AR00000843), in part by the U.S. Department of Energy (DOE) SBIR/STTR Program under Photonic-Storage Subsystem Input/Output (P-SSIO) Interface Project under Grant FPHOTO S7146-01, and in part by the National Security Agency (NSA) Laboratory for Physical Sciences (LPS) Research Initiative (R3/NSA) (Contract FA8075- 14-D-0002-0007, TAT 15-1158).

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

REFERENCES

1 K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556 (2014).

2 J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv:1810.04805 (2018).

3 P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in *Proceedings of the 10th ACM conference on recommender systems* (2016) pp. 191–198.

4 J. Hestness, S. Narang, N. Ardalani, G. Diamos, H. Jun, H. Kianinejad, M. Patwary, M. Ali, Y. Yang, and Y. Zhou, "Deep learning scaling is predictable, empirically," arXiv preprint arXiv:1712.00409 (2017).

5 J. Hestness, N. Ardalani, and G. Diamos, "Beyond human-level accuracy: Computational challenges in deep learning," in *Proceedings of the 24th Symposium on Principles and Practice of Parallel Programming* (2019) pp.1–14.

6 A. Gibiansky, "Bringing hpc techniques to deep learning," Baidu Research,Tech. Rep. (2017).

7 P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola,A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch sgd: Training imagenet in 1 hour," arXiv preprint arXiv:1706.02677 (2017).

8 A. Sergeev and M. Del Balso, "Horovod: fast and easy distributed deep learning in tensorflow," arXiv preprint arXiv:1802.05799 (2018).

9 M. Khani, M. Ghobadi, M. Alizadeh, Z. Zhu, M. Glick, K. Bergman, A. Vahdat, B. Klenk, and E. Ebrahimi, "Sip-ml: high-bandwidth optical network interconnects for machine learning training," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference* (2021) pp. 657–675.

10 M. Naumov, J. Kim, D. Mudigere, S. Sridharan, X. Wang, W. Zhao, S. Yilmaz, C. Kim, H. Yuen, M. Ozdal, *et al.*, "Deep learning training in facebook data centers: Design of scale-up and scale-out systems," arXiv preprint arXiv:2003.09518 (2020).

11 S. Kamil, A. Pinar, D. Gunter, M. Lijewski, L. Oliker, and J. Shalf, "Reconfigurable hybrid interconnection for static and dynamic scientific applications," in *Proceedings of the 4th international conference on Computingfrontiers* (2007) pp. 183–194.

12 N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat, "Helios: a hybrid electrical/optical switch architecture for modular data centers," in *Proceedings of the ACM SIGCOMM 2010 Conference* (2010) pp. 339–350.

13 K. Wen, P. Samadi, S. Rumley, C. P. Chen, Y. Shen, M. Bahadori, K. Bergman, and J. Wilke, "Flexfly: Enabling a reconfigurable dragon- fly through silicon photonics," in *SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (IEEE, 2016) pp. 166–177.

14 M. Ghobadi, R. Mahajan, A. Phanishayee, N. Devanur, J. Kulkarni, G. Ranade, P.-A. Blanche, H. Rastegarfar, M. Glick, and D. Kilper, "Projector: Agile reconfigurable data center interconnect," in *Proceedings of the2016 ACM SIGCOMM Conference* (2016) pp. 216–229.

15 N. Hamedazimi, Z. Qazi, H. Gupta, V. Sekar, S. R. Das, J. P. Longtin, H. Shah, and A. Tanwer, "Firefly: A reconfigurable wireless data center fabric using free-space optics," in *Proceedings of the 2014 ACM conference on SIGCOMM* (2014) pp. 319–330.

16 W. M. Mellette, R. McGuinness, A. Roy, A. Forencich, G. Papen, A. C. Snoeren, and G. Porter, "Rotornet: A scalable, low-complexity, optical

datacenter network," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication* (2017) pp. 267–280.

17 H. Ballani, P. Costa, R. Behrendt, D. Cletheroe, I. Haller, K. Jozwik, F. Karinou, S. Lange, K. Shi, B. Thomsen, *et al.*, "Sirius: A flat datacenter network with nanosecond optical switching," in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication onthe applications, technologies, architectures, and protocols for computer communication* (2020) pp. 782–797.

18 M. Y. Teh, Z. Wu, and K. Bergman, "Flexspander: augmenting expander networks in high-performance systems with optical bandwidth steering," Journal of Optical Communications and Networking **12**, B44–B54 (2020).

19 T.-N. Truong and R. Takano, "Hybrid electrical/optical switch architectures for training distributed deep learning in large-scale," IEICE TRANSACTIONS on Information and Systems **104**, 1332–1339 (2021).

20 Y. Lu, H. Gu, X. Yu, and P. Li, "X-nest: A scalable, flexible, and high performance network architecture for distributed machine learning," Journal of Lightwave Technology (2021).

21 G. Michelogiannakis, Y. Shen, M. Y. Teh, X. Meng, B. Aivazi, T. Groves, J. Shalf, M. Glick, M. Ghobadi, L. Dennison, *et al.*, "Bandwidth steering in hpc using silicon nanophotonics," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (2019) pp. 1–25.

22 Z. Zhu, S. Yan, M. S. Glick, M. Y. Teh, and K. Bergman, "Silicon photonic switch-enabled server regrouping using bandwidth steering for distributed deep learning training," in *Optical Fiber Communication Conference* (Optical Society of America, 2021) pp. Th5H–3.

23 J. Kim, C. Nuzman, B. Kumar, D. Lieuwen, J. Kraus, A. Weiss, C. Lichtenwalner, A. Papazian, R. Frahm, N. Basavanhally, *et al.*, "1100 x 1100 port mems-based optical crossconnect with 4-db maximum loss," IEEE Photonics Technology Letters **15**, 1537–1539 (2003).

24 A. N. Dames, "Beam steering optical switch," (2008), uS Patent 7,389,016.

25 B. Robertson, H. Yang, M. M. Redmond, N. Collings, J. R. Moore, J. Liu, A. M. Jeziorska-Chapman, M. Pivnenko, S. Lee, A. Wonfor, *et al.*, "Demonstration of multi-casting in a 1 × 9 LCOS wavelength selective switch," Journal of Lightwave Technology **32**, 402–410 (2013).

26 C.-T. Lea, "A scalable awgr-based optical switch," Journal of Lightwave Technology **33**, 4612–4621 (2015).

27 Q. Cheng, S. Rumley, M. Bahadori, and K. Bergman, "Photonic switching in high performance datacenters," Optics express **26**, 16022–16043 (2018).

28 Q. Cheng, M. Bahadori, M. Glick, S. Rumley, and K. Bergman, "Recent advances in optical technologies for data centers: a review," Optica **5**, 1354–1370 (2018).

29 K. Suzuki, R. Konoike, S. Suda, H. Matsuura, S. Namiki, H. Kawashima, and K. Ikeda, "Low-loss, low-crosstalk, and large-scale optical switch based on silicon photonics," Journal of Lightwave Technology **38**, 233–239(2020).

30 H. Subbaraman, X. Xu, A. Hosseini, X. Zhang, Y. Zhang, D. Kwong, and R. T. Chen, "Recent advances in silicon-based passive and active optical interconnects," Optics express **23**, 2487–2511 (2015).

31 W. Bogaerts and L. Chrostowski, "Silicon photonics circuit design: methods, tools and challenges," Laser & Photonics Reviews **12**, 1700237 (2018).

32 K. Suzuki, R. Konoike, J. Hasegawa, S. Suda, H. Matsuura, K. Ikeda, S. Namiki, and H. Kawashima, "Low-insertion-loss and power-efficient 32 × 32 silicon photonics switch with extremely high-δ silica plc connector," Journal of Lightwave Technology **37**, 116–122 (2018).

33 K. Suzuki, R. Konoike, N. Yokoyama, M. Seki, M. Ohtsuka, S. Saitoh, S. Suda, H. Matsuura, K. Yamada, S. Namiki, *et al.*, "Nonduplicate polarization-diversity 32 × 32 silicon photonics switch based on a sin/si double-layer platform," Journal of Lightwave Technology **38**, 226–232 (2020).

34 T. Chu, L. Qiao, W. Tang, D. Guo, and W. Wu, "Fast, high-radix silicon photonic switches," in *2018 Optical Fiber Communications Conference and Exposition (OFC)* (IEEE, 2018) pp. 1–3.

35 L. Lu, S. Zhao, L. Zhou, D. Li, Z. Li, M. Wang, X. Li, and J. Chen, "16 × 16 non-blocking silicon optical switch based on electro-optic mach-zehnder interferometers," Optics express **24**, 9295–9307 (2016).

36 L. Qiao, W. Tang, and T. Chu, "32×32 silicon electro-optic switch with built-in monitors and balanced-status units," Scientific Reports **7**, 1–7 (2017).

37 Q. Cheng, A. Wonfor, J. Wei, R. Penty, and I. White, "Demonstration of the feasibility of large-port-count optical switching using a hybrid mach–zehnder interferometer–semiconductor optical amplifier switch module in a recirculating loop," Optics letters **39**, 5244–5247 (2014).

38 P. DasMahapatra, R. Stabile, A. Rohit, and K. A. Williams, "Optical crosspoint matrix using broadband resonant switches," IEEE Journal of SelectedTopics in Quantum Electronics **20**, 1–10 (2014).

39 Y. Huang, Q. Cheng, Y.-H. Hung, H. Guan, X. Meng, A. Novack, M. Streshinsky, M. Hochberg, and K. Bergman, "Multi-stage 8 × 8 silicon photonic switch based on dual-microring switching elements," Journalof Lightwave Technology **38**, 194–201 (2019).

40 Q. Cheng, L. Y. Dai, N. C. Abrams, Y.-H. Hung, P. E. Morrissey, M. Glick, P. O'Brien, and K. Bergman, "Ultralow-crosstalk, strictly non-blocking microring-based optical switch," Photonics Research **7**, 155–161 (2019).

41 A. Gazman, C. Browning, M. Bahadori, Z. Zhu, P. Samadi, S. Rumley, V. Vujicic, L. P. Barry, and K. Bergman, "Software-defined control-plane for wavelength selective unicast and multicast of optical data in a silicon photonic platform," Optics express **25**, 232–242 (2017).

42 K. Padmaraju, D. F. Logan, T. Shiraishi, J. J. Ackert, A. P. Knights, and K. Bergman, "Wavelength locking and thermally stabilizing microring resonators using dithering signals," Journal of Lightwave Technology **32**, 505–512 (2013).

43 J. A. Cox, A. L. Lentine, D. C. Trotter, and A. L. Starbuck, "Control of integrated integrated micro-resonator wavelength via balanced homodyne locking" Optics express **22**, 11279–11289 (2014).

44 T. J. Seok, K. Kwon, J. Henriksson, J. Luo, and M. C. Wu, "Wafer-scale silicon photonic switches beyond die size limit," Optica **6**, 490–494 (2019).

45 S. Han, T. J. Seok, K. Yu, N. Quack, R. S. Muller, and M. C. Wu, "Large-scale polarization-insensitive silicon photonic mems switches," Journal of Lightwave Technology **36**, 1824–1830 (2018).

46 S. Han, J. Beguelin, L. Ochikubo, J. Jacobs, T. J. Seok, K. Yu, N. Quack, C.-K. Kim, R. S. Muller, and M. C. Wu, "32 x 32 silicon photonic mems switch with gap-adjustable directional couplers fabricated in commercial cmos foundry," Journal of Optical Microsystems **1**, 024003 (2021).

47 M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, "Scaling distributed machine learning with the parameter server," in *11th {USENIX} Symposium on Operating Systems Design and Implementation ({ OSDI}14)* (2014) pp. 583–598.

48 Z. Zhu, G. Di Guglielmo, Q. Cheng, M. Glick, J. Kwon, H. Guan, L. P. Carloni, and K. Bergman, "Photonic switched optically connected memory: An approach to address memory challenges in deep learning," Journalof Lightwave Technology **38**, 2815–2825 (2020).

49 Y. Shen, M. H. Hattink, P. Samadi, Q. Cheng, Z. Hu, A. Gazman, and K. Bergman, "Software-defined networking control plane for seamless integration of multiple silicon photonic switches in datacom networks," Optics express **26**, 10914–10929 (2018).

50 Torch.Distributed, https://pytorch.org/docs/stable/distributed.html.

51 Y. Shen, S. Rumley, K. Wen, Z. Zhu, A. Gazman, and K. Bergman, "Accelerating of high performance data centers using silicon photonic switch-enabled bandwidth steering," in *2018 European Conference on Optical Communication (ECOC)* (IEEE, 2018) pp. 1–3.

52 Imagenette, https://github.com/fastai/imagenette.html.

53 Netbench, https://github.com/ndal-eth/netbench.

54 G. Michelogiannakis, K. Z. Ibrahim, J. Shalf, J. J. Wilke, S. Knight, and J. P. Kenny, "Aphid: Hierarchical task placement to enable a tapered fat tree topology for lower power and cost in hpc networks," in *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)* (IEEE, 2017) pp. 228–237.

55 K.-T. Foerster, M. Ghobadi, and S. Schmid, "Characterizing the algorithmic complexity of reconfigurable data center architectures," in *Proceedingsof the 2018 Symposium on Architectures for Networking and Communications Systems* (2018) pp. 89–96.

FIGURE CAPTIONS

FIG. 1. (a) System architecture demonstration with server nodes arranged in the fat tree topology to show SiP switch-based server regrouping and higher-layer bandwidth steering. (b) An example of before (left) and after (right) server regrouping. (c) An example of before (left) and after (right) bandwidth steering above the ToR.

FIG. 2. (a) Overall network control plane. (b) SiP OCS subsystem including the SiP network controller, SiP switch controller, and SiP switches. (c) The SiP network controller board (ZCU106), SiP switch controller board (TR4), and PCB holding a packaged SiP switch. Revised from Zhu *et al*.[22]

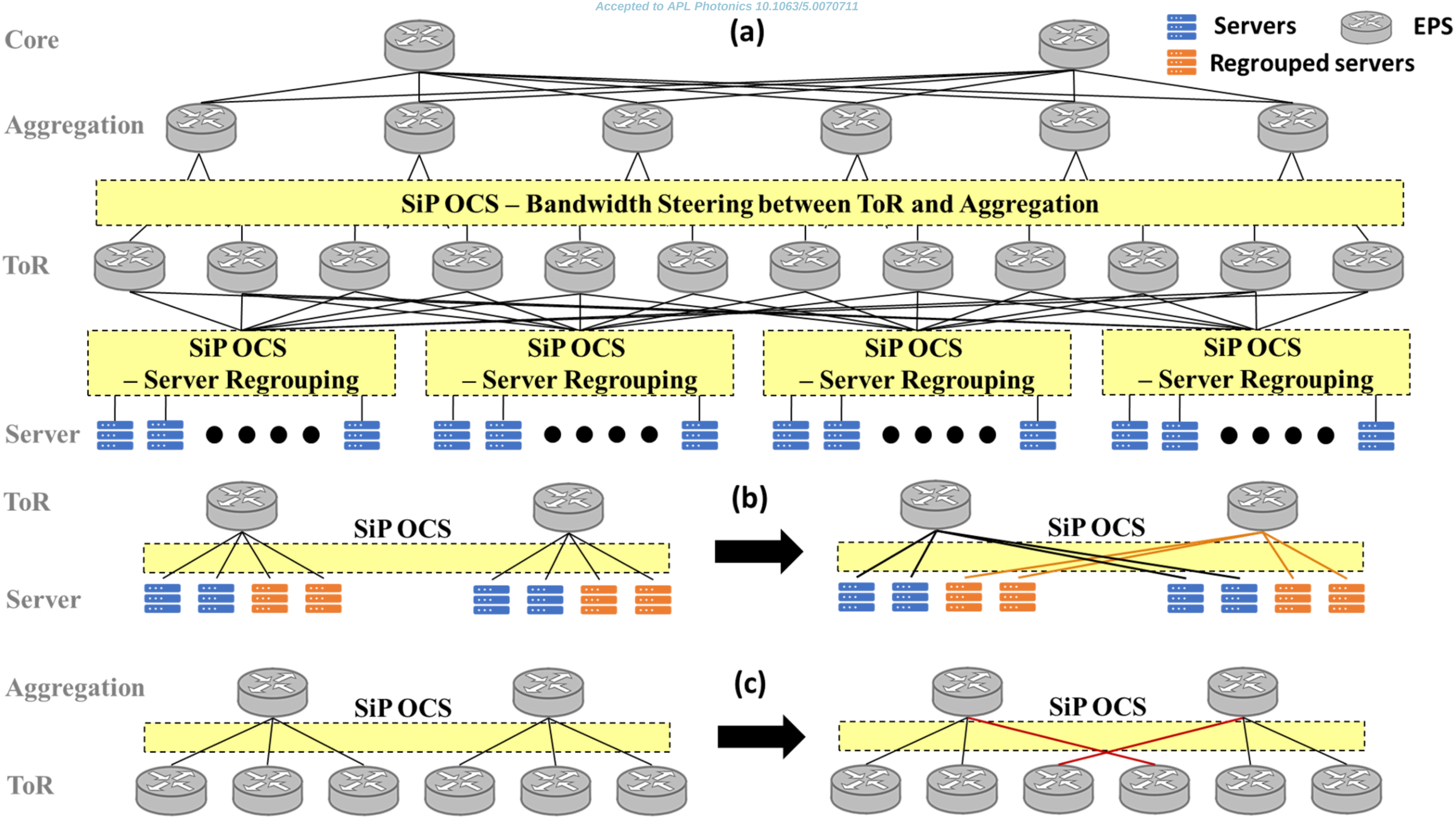
FIG. 3. (a) A 16-node experimental testbed with SiP OCSs and EPSs in a reconfigurable fat tree topology. (b) Experimental setup demonstrating the cases of server regrouping and bandwidth steering above the ToR. (c) A photograph of EPSs, CPU servers, GPU servers, SiP switches, SiP network controller, and SiP switch controller.

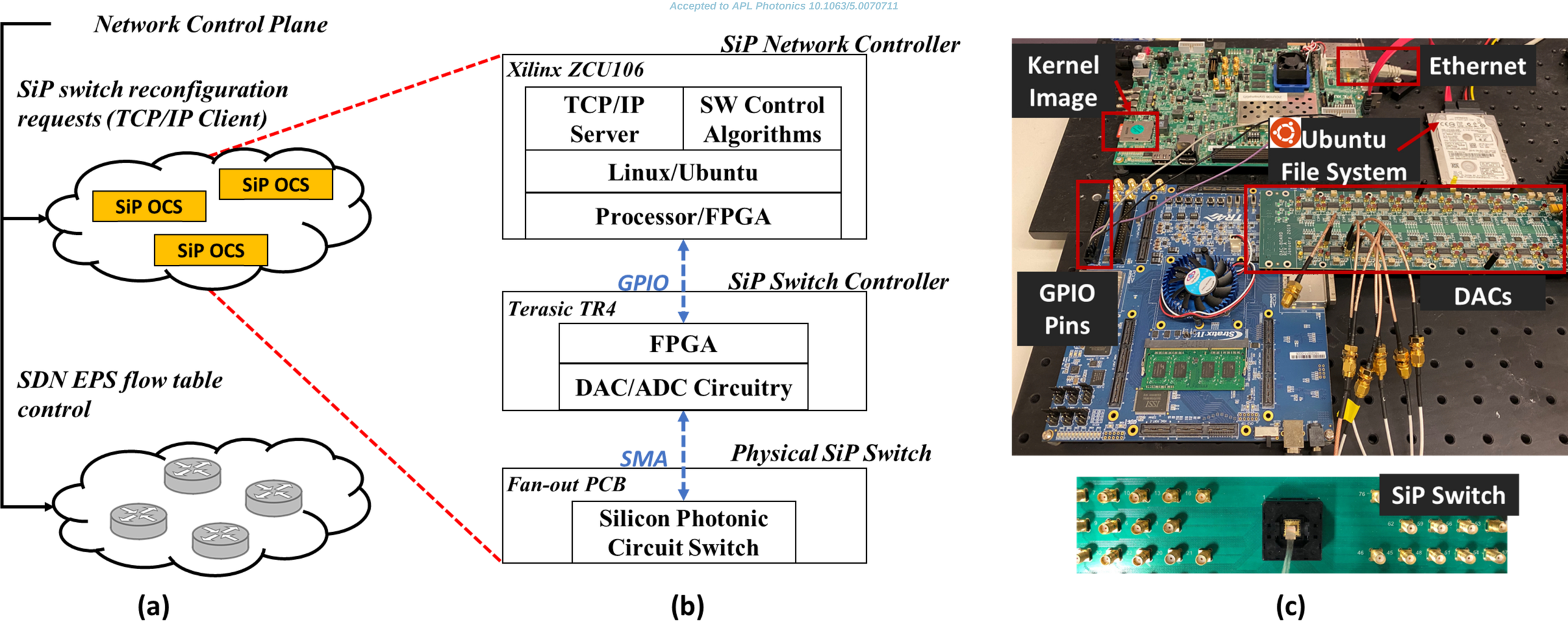
FIG. 4. (a) Test case #1 for demonstrating the network performance improvement by server regrouping. (b) Test case #2 for demonstrating the network performance improvement by partial server regrouping and bandwidth steering above the ToR when SiP OCS port count is limited.

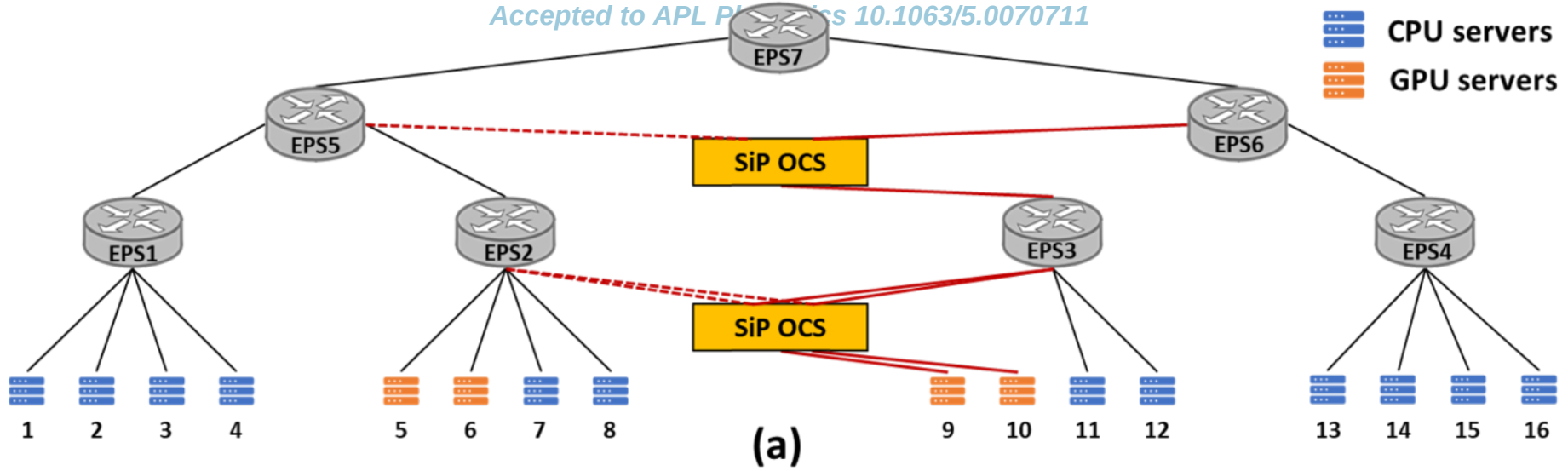
FIG. 5. (a) Throughput of the links to server #9 and #10, from EPS #1 to EPS #5, and from EPS #5 to EPS #7 for the two test cases in the synchronized training of the VGG neural network. (b) Throughput of the links to server #9 and #10, from EPS #1 to EPS #5, and from EPS #5 to EPS #7 for the two test cases in the asynchronized training of the VGG neural network. Revised from Zhu *et al*.[22]

FIG. 6. A 1024-node untapered fat tree topology with SiP OCSs in between the server-ToR and ToR-aggregation layers.

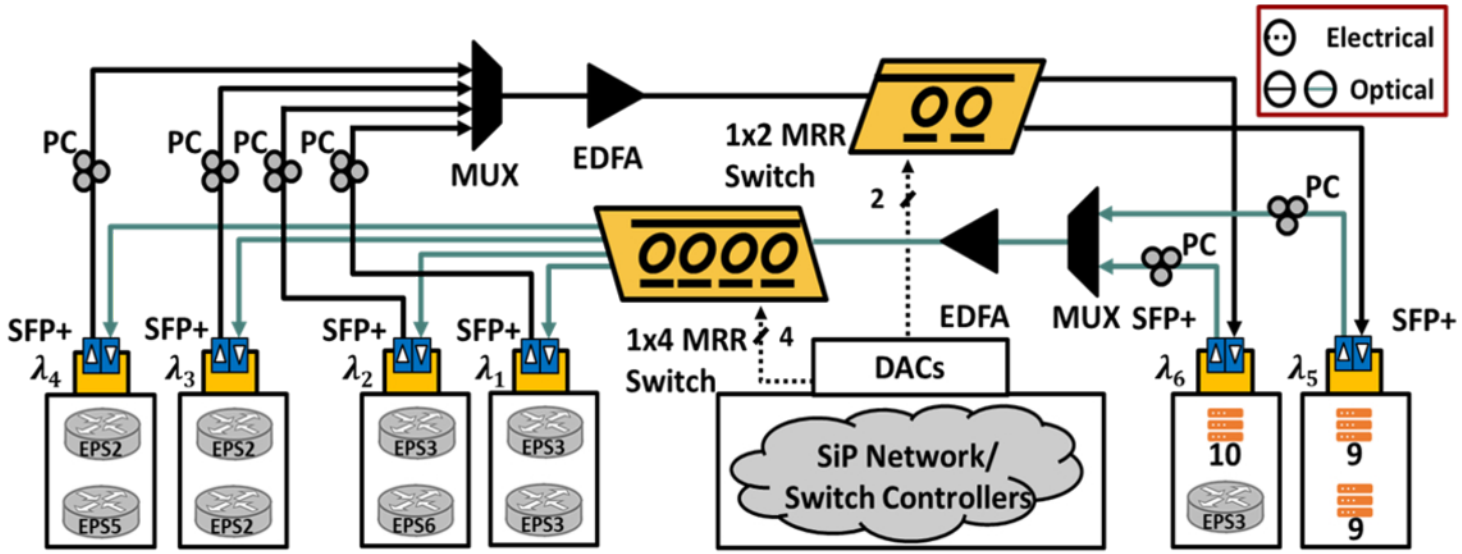
FIG. 7. Average flow throughput of all the flows as a function of the tapering ratio for all the traffic and job mapping scenarios. RG denotes regrouping and BS denotes higher-layer bandwidth steering. With the exception of Uniform (uniform job-mapping), all other cases assume an adversarial interpod job mapping as described in the Simulation Setup Section. (a) Results for ring all-reduce flows. (b) Results for parameter server flows.





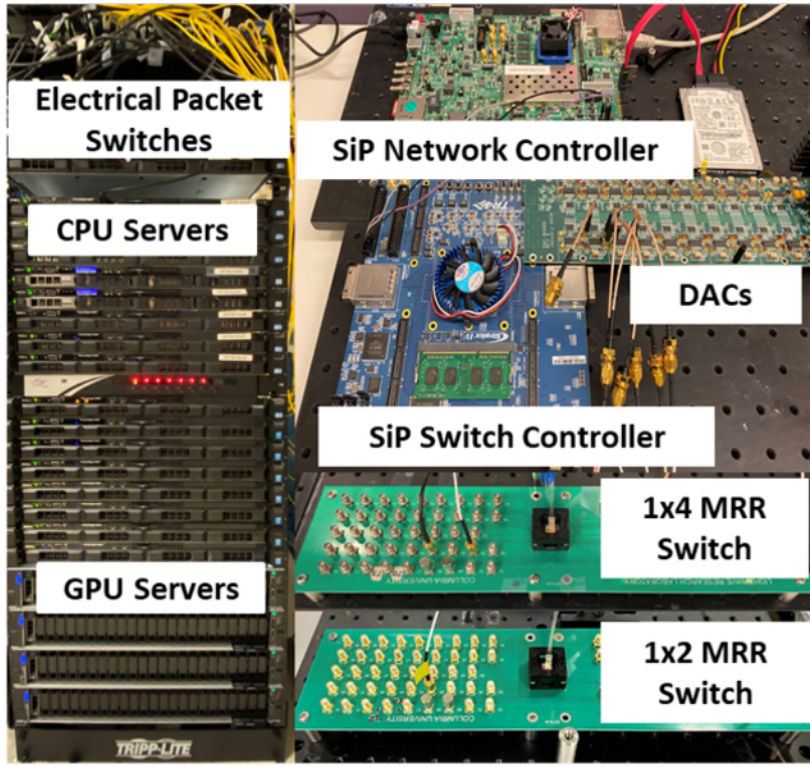


(a)



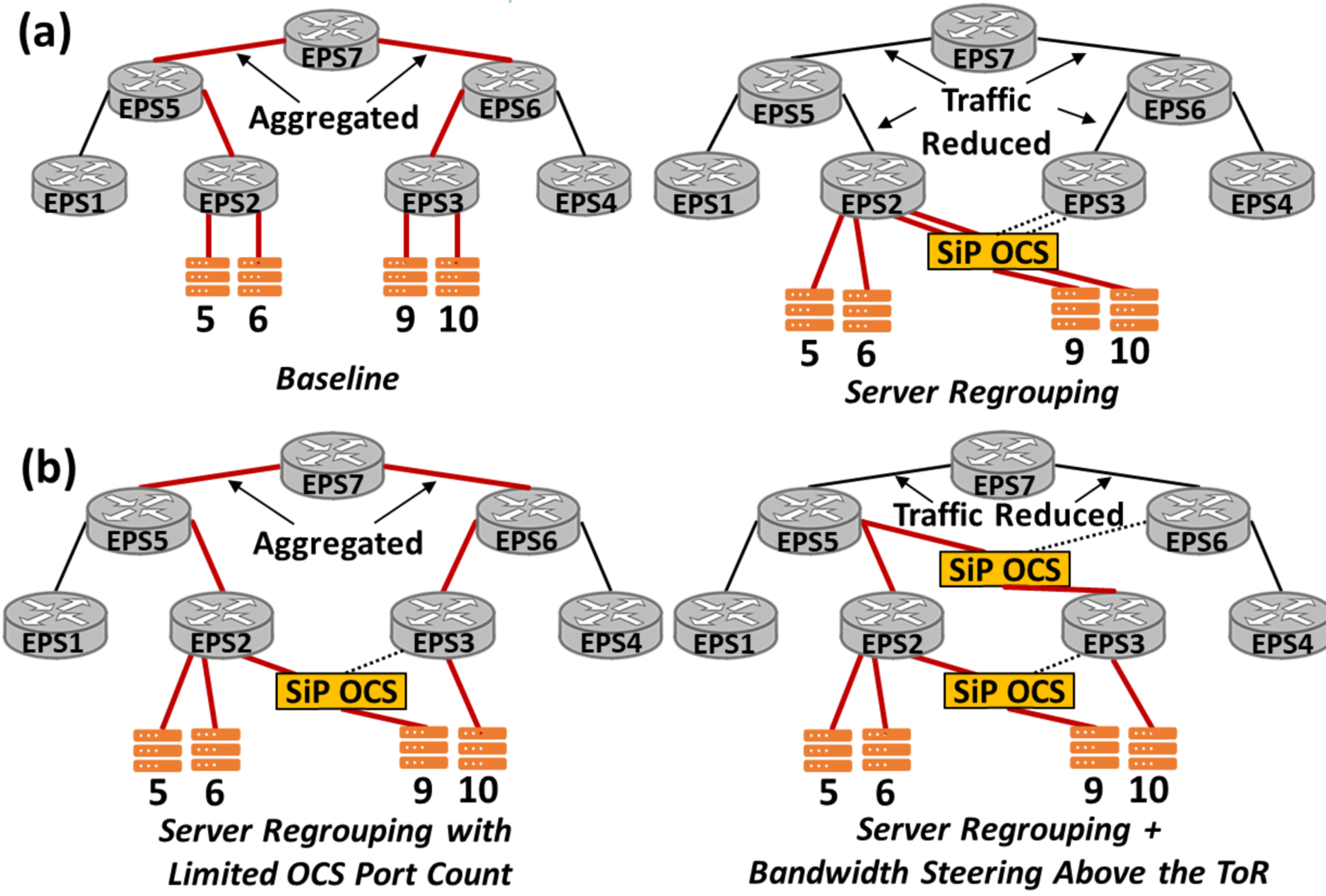
PC – Polarization Controller
DUX – Optical Multiplexer
EDFA – Erbium-Doped Fiber Amplifier
DAC – Digital to Analog Converter
λ_1 = 1545.32nm λ_2 = 1546.92nm λ_3 = 1553.33nm
λ_4 = 1554.94nm λ_5 = 1554.94nm λ_6 = 1556.55nm

(b)



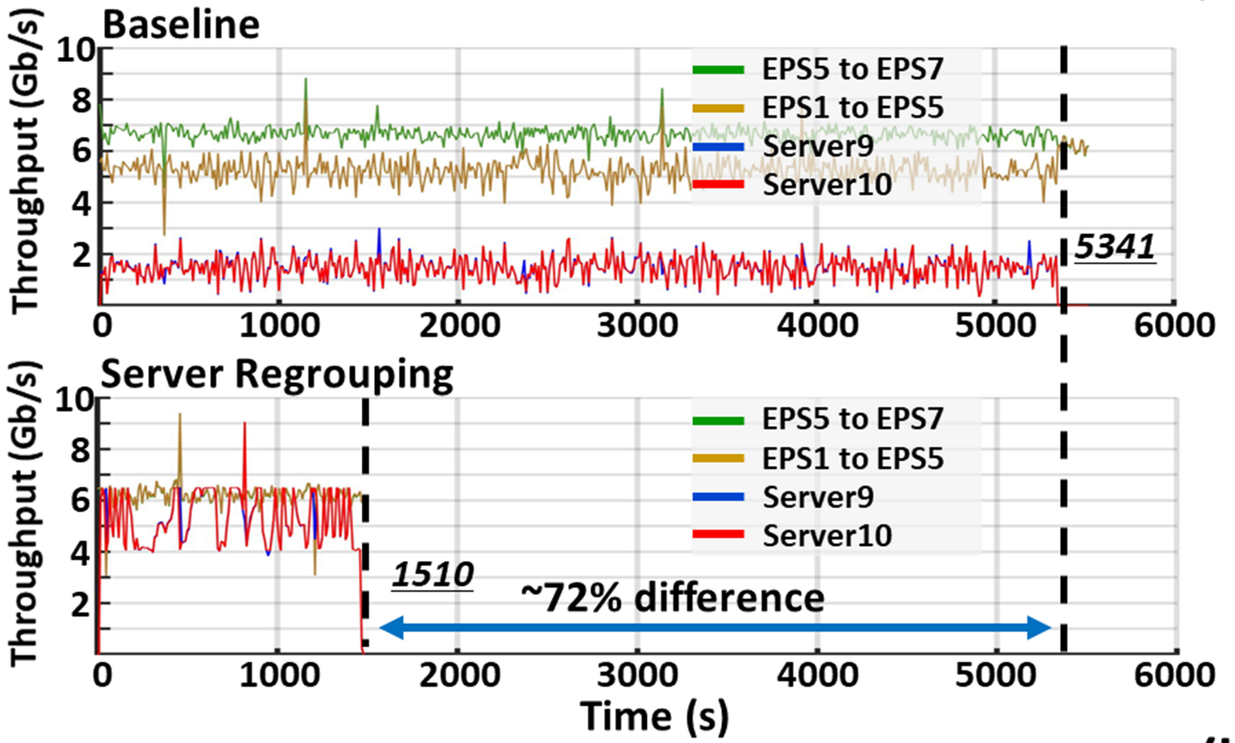
(c)

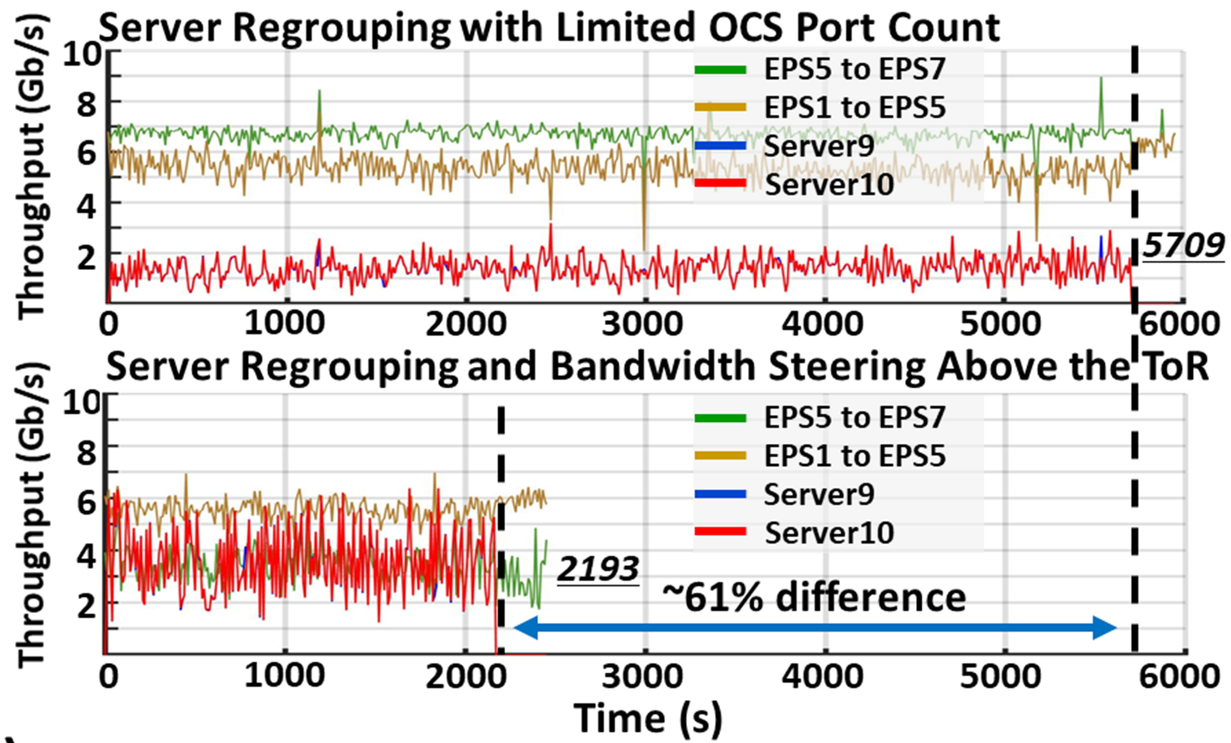
Accepted to APL Photonics 10.1063/5.0070711



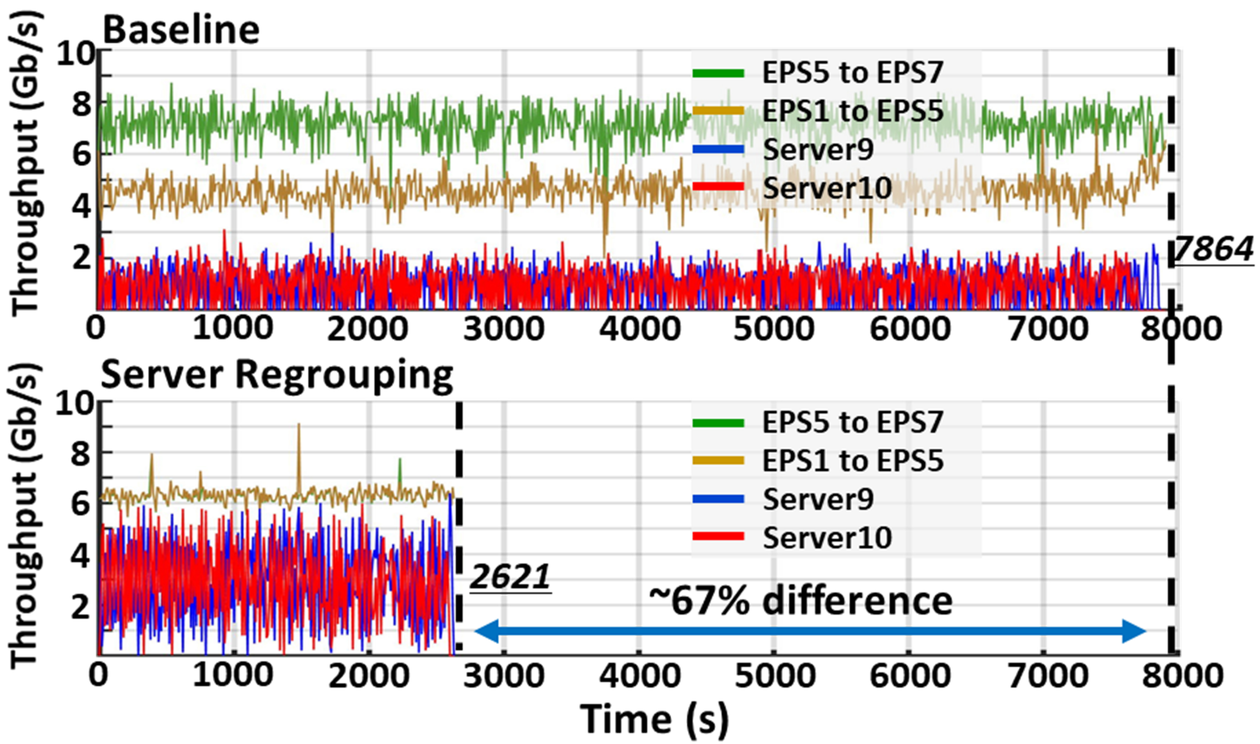
Accepted to APL Photonics 10.1063/5.0070711

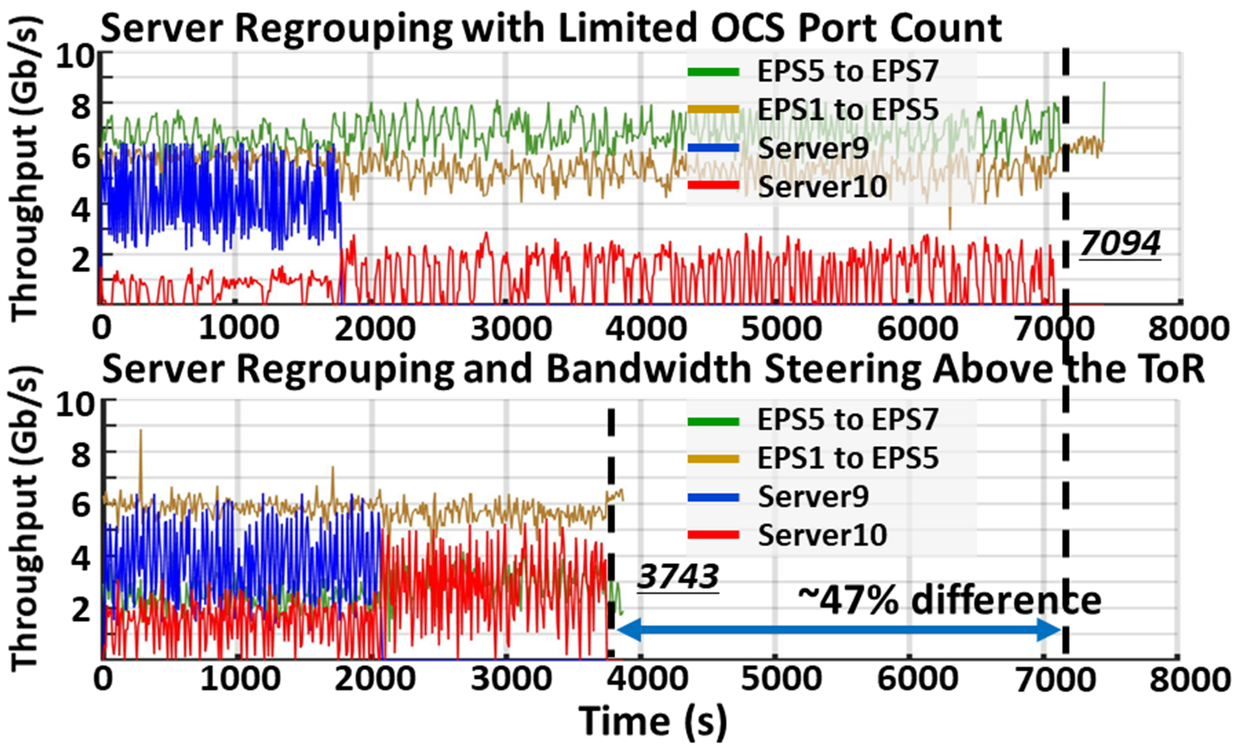
(a)

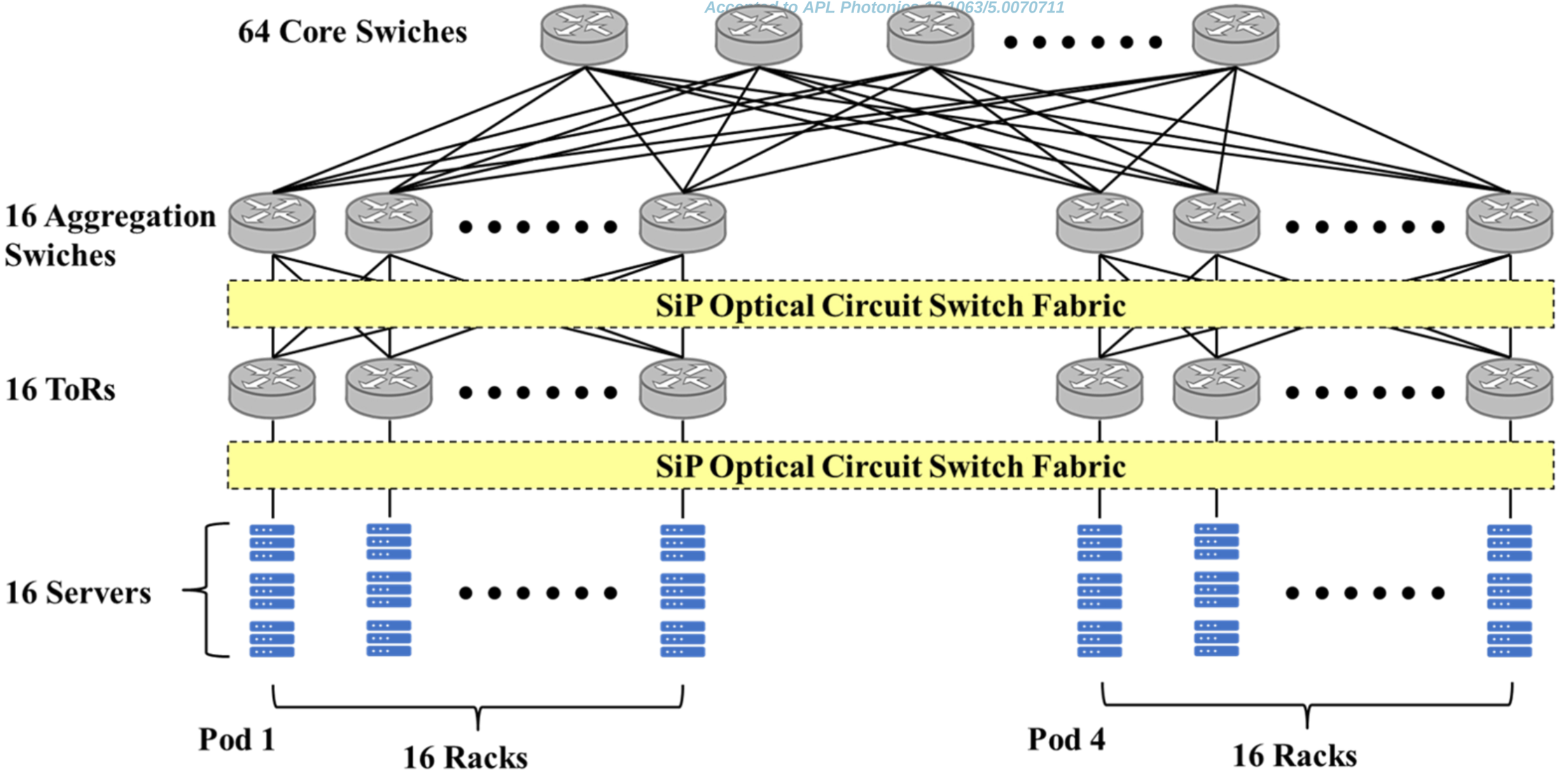




(b)







Accepted to APL Photonics 10.1063/5.0070711