Bandwidth Steering in HPC Using Silicon Nanophotonics

George Michelogiannakis
Lawrence Berkeley National Lab
Stanford University
mihelog@lbl.gov

Yiwen Shen
Columbia University
ys2799@columbia.edu

Min Yee Teh
Columbia University
mt3126@columbia.edu

Xiang Meng
Columbia University
meng@ee.columbia.edu

Benjamin Aivazi
Columbia University
bja2133@columbia.edu

Taylor Groves
Lawrence Berkeley National Lab
tgroves@lbl.gov

John Shalf
Lawrence Berkeley National Lab
jshalf@lbl.gov

Madeleine Glick
Columbia University
msg144@columbia.edu

Manya Ghobadi
MIT
ghobadi@mit.edu

Larry Dennison
NVIDIA
ldennison@nvidia.com

Keren Bergman
Columbia University
bergman@ee.columbia.edu

ABSTRACT

As bytes-per-FLOP ratios continue to decline, communication is becoming a bottleneck for performance scaling. This paper describes *bandwidth steering* in HPC using emerging reconfigurable silicon photonic switches. We demonstrate that placing photonics in the lower layers of a hierarchical topology efficiently changes the connectivity and consequently allows operators to recover from system fragmentation that is otherwise hard to mitigate using common task placement strategies. Bandwidth steering enables efficient utilization of the higher layers of the topology and reduces cost with no performance penalties. In our simulations with a few thousand network endpoints, bandwidth steering reduces static power consumption per unit throughput by 36% and dynamic power consumption by 14% compared to a reference fat tree topology. Such improvements magnify as we taper the bandwidth of the upper network layer. In our hardware testbed, bandwidth steering improves total application execution time by 69%, unaffected by bandwidth tapering.

CCS CONCEPTS

• **Networks** → *Network resources allocation.*

1 INTRODUCTION

Interconnection networks are a crucial feature of today's large-scale computing systems. Highly-efficient networks are built to enable distributing computing elements of massively parallel systems to work in concert to solve challenging large-scale problems. The successful development of powerful node architectures on the path to Exascale combined with necessary energy efficiency improvements is rapidly shifting the bottleneck in today's high performance computing (HPC) and datacenter systems from computation to communication. From 2010 through 2018, there has been a 65× increase in computational throughput with only a 4.8× increase in off-node communication bandwidth in the top 500 HPC systems [17], resulting in an overall 0.08× bytes-per-FLOP ratio [6]. Notably, there has been an 8× decrease in bytes-per-FLOP just from the Sunway TaihuLight (November 2017) to the Summit system (June 2018). If we are to achieve Exascale within 20 MWatt [78] while preserving the computation to communication ratios of Summit, an exascale system will require 50 GFLOPs per Watt and 0.1 pJ per bit for its total communication budget (assuming 200 bits of data communication per FLOP) [17]. Based on these observations, we expect the bytes-per-FLOP ratio in future systems to continue its steady decline and over-provisioning HPC system networks will be increasingly impractical.

Given these trends, communication becomes a significant challenge towards preserving computation improvements, especially those expected by specialization, an increasing trend in HPC [6]. Successful specialization improves compute performance, but also proportionally increases the demand on the interconnection network. As an example, Google's Tensor Processing Unit (TPU) can reach 90 TOPS/s using 8-bit precision, but has a peak off-board communication bandwidth of only 34 GB/s [50]. Therefore, applications need to reuse a byte fetched on the board *2706* times to saturate computational throughput. Only two applications in [50] show such high operational intensity.

Over the past few decades, silicon nanophotonics, leveraging silicon electronics manufacturing infrastructure, has emerged as a promising technology to both increase bandwidth density [92] and reduce energy per bit in all layers of the system [61]. The ability to share the same silicon photolithography processes used for conventional complementary metal oxide semiconductor (CMOS) logic enables a path towards scalable manufacturing and co-integration of optical with electrical components. However, reducing system power solely by network advancements cannot meet future targets because

SC '19, November 17–22, 2019, Denver, CO, USA

ACM ISBN 978-1-4503-6229-0/19/11...$15.00
https://doi.org/10.1145/3295500.3356145

the network consumes just 4% to 12% of the power in large-scale systems [71]. Even if these numbers grow to 25%, an ideal zero-energy network cannot reduce system power by 2×. Therefore, achieving future goals with photonics cannot be a simple matter of drop-in replacements of current network components. Instead, it requires architectural changes that take advantage of key silicon photonic properties to achieve significant network-wide improvements.

In this paper, we propose to use silicon photonic (SiP) switches as circuit-switched components between electronic packet switches (EPSs) in order to dynamically adjust the connectivity (steer bandwidth) between the ToR (first-level) and aggregation (second-level) EPSs of a three-level fat tree topology. Fat trees and their variants such as HyperX are extensively used in datacenters [21, 22] and HPC [10, 45, 49, 77]. Our goal is to optimally adjust the connectivity of the lower layers of the network to minimize the number of packets that traverse the higher layers, while avoiding to divide the network into disjoint electrical and optical parts [11]. This also helps to recover from system fragmentation that is otherwise hard to mitigate using common task placement strategies. We refer to this process as bandwidth steering and combine it with a scalable maximum-weight matching algorithm to achieve our goal for large networks. A key advantage of bandwidth steering is enabling operators to aggressively oversubscribe top-layer bandwidth, something that is suited to a hierarchical topology and is already common practice due to the cost of top-layer bandwidth [23], but without performance penalties because normally tapering the top layer would increase congestion. Efficiently utilizing top-layer bandwidth is imperative due to the cost of high-bandwidth equipment placed at the top (e.g., core) level [39, 83].

In our experimental hardware testbed, bandwidth steering provides an application execution time improvement of 69%. With bandwidth steering, application execution time does not get penalized if we aggressively taper top-layer bandwidth. In contrast, in a vanilla fat tree execution time degrades when tapering top-layer bandwidth. In our system-scale simulations with a few thousand network endpoints, we show that bandwidth steering reduces the ratio of upper-layer link utilization to lower-layer link utilization by 59%. Furthermore, bandwidth steering reduces power per unit throughput (dissipated power per byte transferred) by 36% static and 14% dynamic, on average across different applications. Bandwidth steering also reduces average network latency by up to 20%, increases average throughput by an average of 1.7×, and reduces the effects of fragmentation [38, 72, 98] by reconstructing locality between application tasks that were meant to be neighbors. These improvements further increase by up to 20% for throughput and 25% for network latency as we aggressively taper top-layer bandwidth. Bandwidth tapering is immediately applicable to HPC and datacenter networks using today's EPSs and SiP switches. Our SiP switches reconfigure in 20 μs using software-defined networking (SDN).

2 BACKGROUND AND MOTIVATION

Communication in modern datacenters and HPC systems faces two important challenges: cost and operational efficiency. Cost includes both capital expenditures of buying the system and operating expenditures for increasing available network bandwidth. Cost is the primary motivator to eliminate the negative consequences such as congestion of reducing bandwidth at the higher-levels of multi-tiered interconnect topologies (typically referred to as "bandwidth tapering") because that bandwidth covers more physical distance and therefore tends to be more costly [58]. This may lead the top layer to be oversubscribed, possibly sacrificing system efficiency [23]. For example, to scale to 20k nodes, Google's Jupiter topology was intentionally designed with a 2:1 oversubscription at the ToR uplink layer [83]. Similarly, Microsoft's data center networks have a 5:1 oversubscription at the ToR layer [39]. These cost-cutting measures result in network congestion that is well known to create "long tail" communication latencies, which severely diminish the performance of interactive and bulk-synchronous parallel applications [28].

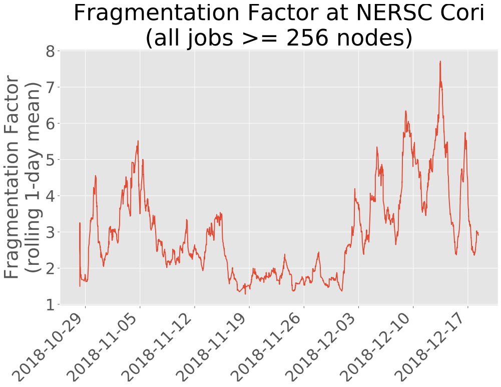

Figure 1: Measured data from NERSC's Cori shows that fragmentation is both persistent and dynamic. In this graph, fragmentation is defined as the observed number of Aries (Dragonfly) groups that an application spans, divided by the smallest possible number of groups that the application would fit in.

Related work attempts to mitigate congestion to enable more aggressive bandwidth tapering with no performance penalty by exploiting locality in task placement [46, 64]. While these approaches have shown significant benefits, there is little they can do in the case of applications with imperfect locality or for non-optimal placement of applications across compute resources [64]. For instance, a 4D stencil application where MPI tasks communicate persistently with a handful of other tasks is unlikely to avoid bandwidth contention at the top level of the topology.

Furthermore, system fragmentation is a well-documented challenge [18, 38, 72, 98]. System fragmentation is caused by the dynamic nature of modern systems where applications of various sizes initiate and terminate continuously. As a result, applications often receive an allocation on a set of distant and non-contiguous nodes. Hence, even communication between consecutive MPI ranks is not physically neighboring. Figure 1 illustrates that in NERSC's Cori fragmentation is both persistent and dynamic. Consequently, improved task placement would be ineffective in optimizing bandwidth tapering because it is applied only to the tasks of an application, wherever they physically reside. One would need to either idle nodes to await for a contiguous range of endpoints (resulting in system underutilization), or migrate tasks to find contiguous nodes on the lower tiers of the topology. However, task migration takes on the order of seconds to complete [93]. Schedulers alone cannot resolve the fragmentation issue. Therefore, we must find a solution that involves both the hardware and the scheduler.

3 THE PROMISE OF SILICON PHOTONICS FOR RECONFIGURABILITY

Optical circuit switching (OCS) offers a promising approach to reconfiguring the interconnect on demand in order to maximize efficiency by enabling more effective "right sizing" of bandwidth tapering. While EPSs impose multiple cycles of latency and likewise an energy penalty to traverse a switch due to buffering and control logic, silicon photonics (SiP) in-package switches impose negligible dynamic energy and latency penalties that make them essentially transparent at an architectural level [26, 80]. To reap these benefits, architectures using SiP switches must avoid conversions to the electrical domain (OEO conversions), which also means there can be no computation on traversing packets. This motivates the use of photonic switches as circuit-switched elements [56, 76, 81].

A common approach for implementing OCS for telecommunication applications is to use microelectromechanical switches (MEMS) [70], such as Glimmerglass and Calient. However, these high radix mechanical switches can be difficult to insert into a tightly integrated HPC fabric and can also impose large reconfiguration delays. Similarly, trade-offs arise when considering the use of arrayed waveguide grating routers (AWGRs). Although the AWGR itself is passive, the interconnect based on the AWGR usually requires tunable wavelength transceivers and possibly tunable wavelength converters, adding complexity and additional power consumption [55]. Although lower radix than a typical MEMS OCS, SiP switches and the resulting photonic multi-chip-modules (photonic MCMs) are a promising path for tight integration into the interconnect fabric, enabling extremely high off-package escape bandwidths.

The silicon photonics platform leverages the mature and widespread CMOS electronics manufacturing infrastructure, reducing the cost of manufacture. Silicon photonic microring-resonator (MRR)-based links offer high bandwidth density, energy efficient performance, and small footprint. The basic switching cell in the SiP MRR switch is a microring coupled to a bus waveguide. The path the light takes is determined by the coupling between the ring and the waveguide, which is controlled by tuning the resonance between the on and off states. In our device we use silicon's strong thermo-optic (T-O) coefficient (1.8×10^{-4} /K), to tune the resonance at microsecond time scales. The thermo-optic control could be replaced by an electro-optic phase shifter, based on carrier injection or depletion into the microring, which would operate at nanosecond time scales [20, 24]. The recent development of microresonator based comb laser sources with greater than 10% wall plug efficiency greatly improves the potential for low cost, energy efficient, ultra-high bandwidth communication. Laser combs with more than 100+ channels realized using conventional silicon lithography have been demonstrated. Interconnects constructed from these comb lasers in conjunction with arrays of ring resonators operating at 25 gigabits/second per channel should achieve bandwidth densities exceeding 2.5 terabits/second [57].

We propose to take advantage of the inherent reconfigurability of SiP switches [26, 56, 81] to design a reconfigurable network that can migrate connections between EPSs – an approach referred to as bandwidth steering. By steering bandwidth to minimize the use of higher layers of a hierarchical topology, we can taper those higher layers more aggressively. In essence, we use bandwidth steering to reconstruct locality that was lost due to system fragmentation and was hard to recover with task placement. Therefore, our network will both have lower cost and be less affected by fragmentation than a baseline network with no bandwidth steering. In addition, bandwidth steering increases the maximum throughput of the network and reduces power and latency.

Steering bandwidth with EPSs instead of SiP switches would impose a large performance and energy penalty, mitigating some of the benefits [25]. Due to the distances between ToR and aggregation EPSs, modern systems use optical cables. Inserting additional EPSs in between would require another OEO conversion, which would significantly raise procurement cost, power, and latency.

Because our motivation to reduce the cost of the network via bandwidth tapering is only possible in a hierarchical network, we choose a fat tree, originally proposed as a folded Clos [27], to represent a popular hierarchical topology. Fat trees and their variants are extensively used in datacenters [21, 22, 83] and HPC [10, 45, 49, 77] due to their favorable wiring properties and their ability to provide full throughput for any traffic pattern, assuming perfect load balancing and full bisection bandwidth (i.e., the higher layers of the fat tree have the same total bandwidth as the lower layers) [43]. A recent example in HPC is the HyperX topology that is similar to a fat tree [10]. Our approach applies to other hierarchical topologies as well. Non-hierarchical topologies do not provide opportunities for bandwidth tapering in the same way and thus we leave them for future work.

Network reconfigurability aligns with the observation that homogeneous networks do not perform as well as heterogeneous networks with the same building blocks [84], or networks with a combination of heterogeneous and structured connections [62].

4 BANDWIDTH STEERING IN A FAT TREE

This section presents our approach to bandwidth steering in a hierarchical topology without resorting to disjoint optical and electrical network paths. We use a fat tree throughout our paper though our proposal applies to similar hierarchical topologies where bandwidth can be steered in the lower levels, such as a HyperX [10].

4.1 Notation

The notations used in this section are tabulated in Table 1. For the sake of convenience, we shall abuse the vector notation $\mathbf{x^o}$ to denote the vector $[x^o_{ij}]\ \forall\ i, j \in \{1, 2, ..., n\}$ associated with OCS index o.

4.2 Conceptual Design

Figure 2 illustrates the concept of bandwidth steering applied to a three-level fat tree topology. We only apply bandwidth steering to uplinks because that is sufficient to ensure that traffic using steered connections does not use the top layer of the fat tree. Steering downlinks is also possible and would provide extra degrees of freedom at the cost of algorithm complexity and extra SiPs OCSs.

Bandwidth steering in a hierarchical topology such as a fat tree reduces hop count and places less stress on the top layer of the topology. This not only reduces latency and contention, but also allows us to reduce the available bandwidth at the top layer, commonly referred to as bandwidth tapering, thereby reducing power and procurement costs. In addition, bandwidth steering can increase vertical bisection bandwidth (between the left and right parts) by relocating

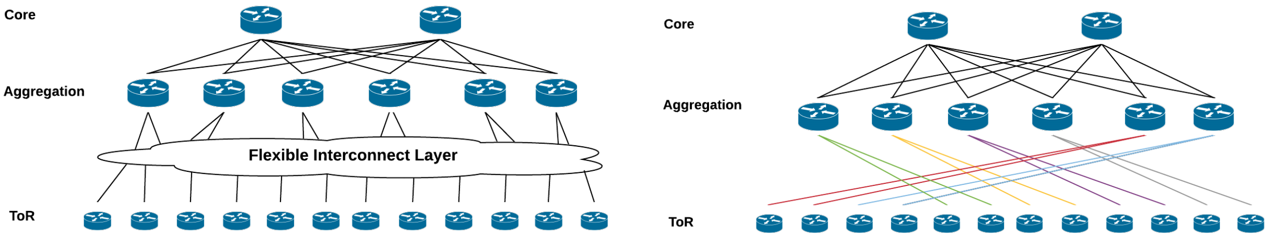

Figure 2: We apply bandwidth steering between the ToR and aggregation EPSs of a three-level "vanilla" fat tree topology (left). On the right is one example of a steered topology where ToR EPS i connects to aggregation EPS $\frac{i-2}{2} - 2$ mod 6.

n	Total number of pods (groups between first-level ToR and second-level aggregation switches) in the fat tree topology
k	Total number of SiP OCSs in the fat tree topology
$T = [t_{ij}] \in \mathbb{R}^{n\times n}$	Traffic matrix, where t_{ij} denotes the traffic (Gbps) sent from s_i to s_j
r	Radix of SiP optical circuit switches
G_o	Bipartite graph representation of a SiP OCS with index o
$X = [x^o_{ij}] \in \mathbb{Z}^{n\times n\times k}$	Matching of all SiP OCSs, where x^o_{ij} denotes the number of ToR upward-facing links from pod i connected to aggregation switches in pod j via OCS o
$D = [d_{ij}] \in \mathbb{Z}^{n\times n}$	Interpod logical topology; d_{ij} denotes the number of upward-facing links connecting a ToR from pod i to an aggregation switch from pod j. Note that $d_{ij} = \sum_{o\in\{1,2,\ldots,k\}} x^o_{ij}\ \forall\ i,j \in \{1,2,\ldots,n\}$

Table 1: Notations

connections that would not otherwise cross that bisection. Essentially, bandwidth steering tries to capture persistent communication that spans groups of ToR (first-level) and aggregation (second-level) EPSs. These groups are called pods. Communication that crosses pods would normally use top-level switches. By steering, we can essentially reconstruct locality of communication between distant nodes.

We do not apply bandwidth steering at the top layer of the fat tree because none of the aforementioned benefits of bandwidth steering would apply. Traffic destined to a top-level switch has the same amount of bandwidth to any one of the top-level switches. However, in a fat tree with more than three levels we can apply bandwidth steering to each layer individually. We choose three levels because that is typical for large-scale systems with fat trees [9, 11].

4.3 Implementation

Figure 3 shows how we arrange SiP switches between electrical routers to implement bandwidth steering at the first layer of our three-level fat tree. Once configured, SiP switches are "pass-through" in that they perform no OEO (Optical-Electrical-Optical) conversions or computation on passing traffic. Consequently, they incur virtually no latency or dynamic power penalties. The static power for our SiP switches is orders of magnitude less than the EPSs as we discuss in Section 7.1. These qualities make SiPs attractive for seamless network reconfiguration. Using circuit-switched EPSs instead would incur higher dynamic and static power, as well as increased latency [4].

We place only one SiP switch between EPSs to minimize optical loss. Therefore, traffic does not take more than one consecutive hop in the optical domain. Otherwise, the optical signal may require re-amplification, which in turn would diminish our cost and energy motivation.

Furthermore, the placement of SiPs determines our strategy for bandwidth steering. Our strategy, as shown in Figure 3(bottom), allows for each ToR switch to directly connect to $r - 1$ other aggregation switches, where r is SiP OCS radix. However, only one optical cable is devoted to each connection. Therefore, connections can be steered farther but with only one cable's worth of bandwidth. Alternatively, we can place SiPs such that they steer connections between neighboring aggregation switches only. This limits the distance of steering, but it increases the available bandwidth to steer from any particular ToR switch to an aggregation switch.

Our choice of Figure 3(bottom) was motivated by several factors. Firstly, numerous HPC applications create persistent communication with a small number of distant nodes [1] that are well beyond neighboring, as in the case of a multi-dimensional stencil code. Because application performance is often constrained by that communication, increasing the distance of bandwidth steering increases the probability of capturing that critical communication. Secondly, system fragmentation increases the average physical distance between tasks of the same application. Therefore, bandwidth steering can essentially "undo" some of the adverse effects of fragmentation. Finally, the default placement in many applications contains locality created by programming optimizations [30]. Therefore, a significant fraction of application traffic stays within the same pod anyway without steering, and therefore high-bandwidth steering to only neighboring pods is not needed.

The number of input and output ports of SiP switches also represents a trade-off. More inputs and outputs allow bandwidth steering across farther pods or with higher bandwidth. However, this is offset by the increased reconfiguration delay, increased static power of higher-radix SiP switches, reconfiguration algorithm complexity, and the number of applications that can take advantage of the additional ports.

Our topology steered with SiPs does not have more optical fibers compared to a vanilla fat tree, as the total number of EPS ports remains the same. We use optical half cables through SiP switches

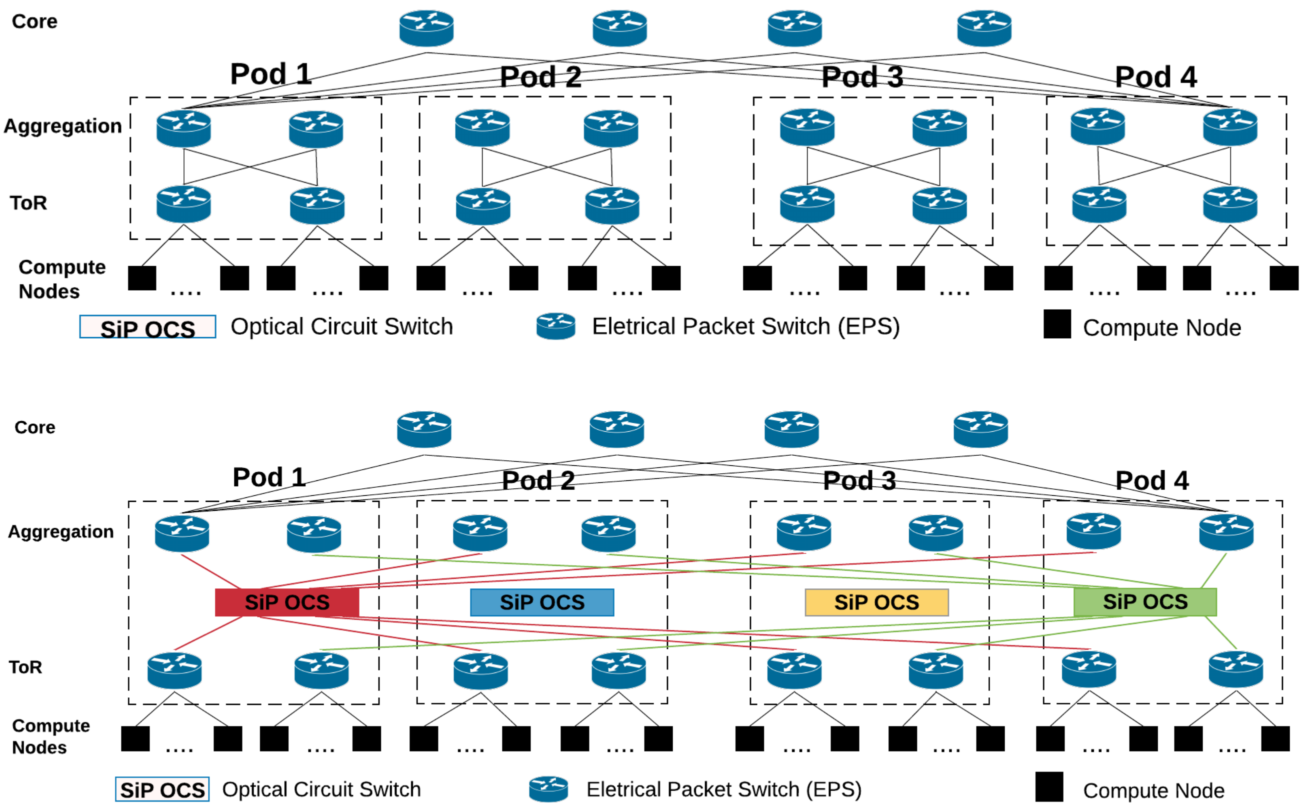

Figure 3: Top: A regular ("vanilla") 3-level fat tree topology without SiP OCSs for bandwidth steering. Bottom: A reconfigurable fat tree topology enabled by the placement of SiP OCSs between the top-of-rack (ToR) switches at the first level of the topology and aggregation switches (second level) of different pods. Link colors correspond to which SiP OCSs they connect to. Only links to and from the leftmost and rightmost SiP OCSs are shown for clarity. Pods are groups of ToR and aggregation switches. The SiP switch ingress ports are connected to the upward-facing links from ToRs, while the egress ports are connected to aggregation switches, which are at the second level of the fat tree. Note that downward-facing links from aggregation switches are not connected to optical switches, and hence are not steerable. As shown, bandwidth steering moves bandwidth where it is most needed, including increasing the vertical bisection bandwidth if the traffic pattern warrants it.

to avoid doubling the number of optical transceivers in the first layer [75].

In addition to reduced latency, dynamic power, and increased throughput, a major motivation for bandwidth steering is that by keeping more traffic in the lower layers of the fat tree, we can aggressively reduce (taper) the bandwidth of the higher layers. This reduces procurement cost as well as static power [39, 83].

4.4 Routing in a Steered Network

Even though in many cases deterministically routing packets through the shortest available path is best, this strategy risks overloading the steered paths. Instead, networks can route packets either obliviously (at random) or dynamically (based on observed conditions such as congestion) between the shortest steered and original unsteered paths [48, 99]. We leave this exploration as future work because the optimal strategy is co-dependent on the bandwidth steering algorithm as well as the traffic pattern, including transient congestion. For our work, we consider two strategies: (i) a deterministic shortest-path routing algorithm that always chooses the steered path if there is one because that is shortest, and (ii) a load-balancing algorithm that always chooses a steered path if there are no packets queued for it, otherwise it chooses randomly between the steered path and any of the unsteered paths. This choice is made on a per-packet basis at every router, individually for each steered path.

4.5 Bandwidth Steering Algorithm

Bandwidth steering requires knowledge of application traffic patterns in order to reconfigure the logical connectivity of the fat tree. There are multiple methods for applications to communicate their expected traffic matrix or simply which flows are expected to be dominant. In the case of MPI, this can be done in graph form [42] or through text files [7, 12]. Alternatively, application traffic can be predicted using neural networks [14, 59], deep learning [68], or other techniques. Any of these techniques is appropriate to our approach for bandwidth steering. In fact, as our algorithm relies on ranking flows instead of precise knowledge of the volume of traffic, and because steering bandwidth to serve a source–destination pair can also serve neighboring source–destination pairs in the same pods depending on the network's routing algorithm, our bandwidth steering approach can tolerate traffic matrix inaccuracies.

We also note that HPC benchmarks typically generate persistent traffic patterns that change slowly or do not change at all [16]. This is an observation we confirm in our experiments. Still, we note that some traces are in fact more dynamic. For those, we can apply our algorithm and reconfigure SiP OCSs more often. However, the expected benefit of reconfiguration based on the new traffic pattern must be weighed against the energy and delay cost of reconfiguration. This has already been noted in related work with algorithms that minimize reconfigurations [19], model hybrid optical/electrical networks [60], and model the expected energy cost [96].

Given a traffic matrix, our bandwidth-steering algorithm primarily aims to reduce the amount of traffic that traverses core layer links. Reducing traffic at the core layer has the added benefit of reducing packet latency in addition to reducing congestion due to over-subscription. The high-level idea of our implementation is to iteratively solve for the configuration of each OCS as a maximum-weight matching problem, with the matching weights being proportional to the amount of unsatisfied traffic flows.

Each SiP OCS can be modeled as a bipartite graph with weighted edges, G_o. We can generate G_o as follows:

(1) Generate a dummy source and sink node, and two sets of nodes namely the left-sided and right-sided nodes, L and R respectively.
(2) For each OCS ingress port, create a node labelled with the source's pod ID and place it in L.
(3) For each OCS egress port, create a node labelled with the destination's pod ID and place it in R.
(4) Connect the dummy source node to all nodes in L with unit capacity; then connect all nodes in R to the dummy sink node with unit capacity.
(5) Connect all the nodes in L and R with unit capacity.

Note that the preceding pseudocode does not explicitly set the weights associated with the edges in each bipartite graph. This is because the weights will be set at runtime of Algorithm 1. After the bipartite graphs of all SiP switches are formed, the bandwidth steering algorithm can be triggered to reconfigure the logical topology of the fat tree to better support an expected traffic matrix.

Algorithm 1: Bandwidth steering

```
input  : $T = [t_{ij}] \in \mathbb{R}^{n \times n}$ - inter-pod traffic matrix
input  : $\{G_1, G_2, ..., G_k\}$ - bipartite graphs
output : $X$ - SiP matchings
Initialize $d_{ij} := 0 \ \forall \ i, j \in \{1, 2, ..., n\}$
for $o \in \{1, 2, ..., k\}$ do
    $\omega_{ij} \leftarrow \frac{t_{ij}}{d_{ij}+1}, \quad \forall \ i, j \in \{1, 2, ...., n\}$;
    $\mathbf{x^o} \leftarrow$ solve max. weight matching ($G_o$, $\boldsymbol{\omega}$)
    for $(i, j) \in \{1, 2, ..., n\} \times \{1, 2, ..., n\}$ do
        if $x^o_{ij} > 0$ then
            $d_{ij} \leftarrow d_{ij} + x^o_{ij}$;
        end
    end
end
```

Given that the overall matching problem is NP-hard [35, 36, 60], our proposed greedy matching algorithm does not guarantee solution optimality. That said, we update the matching weights between source-destination pod pairs while taking into account already established links between pod pairs. This ensures that unformed OCS matchings will prioritize formation of links between pod-pairs with fewer links established.

In our implementation, we employ Edmond-Karp's algorithm [33, 44] to solve each maximum-weight matching subproblem. Each bipartite matching can be solved in $O(r^4)$. Thus, the overall algorithm has a runtime complexity of $O(kr^4)$. Though faster algorithms for solving weighted bipartite matching problems exist, the fact that r tends to be small for SiP switches makes our implementation very computationally efficient in practice.

By default, the bandwidth steering algorithm is run every time an application initiates or terminates in the system. This is based on our observation in NERSC's Cori that one multi-node application initiates approximately every 17 seconds, which is multiple orders of magnitude longer than the reconfiguration time of modern photonic switches that can be as low as nanoseconds [56]. In our experimental testbed, reconfiguration time is 10 to 20 μs (see Section 6.1). Networks with longer reconfiguration delays or large problems that make our matching algorithm have noticeable runtime may motivate us to apply our algorithm in longer and fixed time intervals, based on observed traffic since the last time bandwidth steering was applied.

5 EXPERIMENTAL DEMONSTRATION

In this section we describe our experimental hardware testbed for demonstrating and evaluating our concept.

5.1 Testbed Description and Evaluation

We built a 32-node HPC testbed. The topology diagram is shown in Figure 5 while a photograph of the components is shown in Figure 4. The EPSs shown are virtually partitioned from two OpenFlow-enabled PICA8 packet switches, each with 48 10G SFP+ ports. We use four SiP switches using microring-resonators (MRRs), each of which acts as a 4×4 OCS. Two of the SiP switches shown are emulated by manually connecting different optical fibers in between experiments, which maintains the properties of a SiP switch in terms of propagation latency. The SiP switches we use were manufactured at the OpSIS foundry. We have also performed experiments with similar results using SiP switches fabricated by AIM Photonics. The compute nodes are virtual machines, with each server housing two virtual machines. We use 16 servers; each uses Intel Xeon Processors E5-2430 with 6 cores and 24 GB of RAM. NICs are NetXtreme II BCM57810 10 Gigabit Ethernet from Broadcom Inc.

The network is arranged in a standard three-layer fat tree topology with $k = 2$ (two lower-level switches connect to a higher-level switch). It is also divided into two pods with 16 nodes per pod. Four nodes are connected to each top-of-rack (ToR) packet switch. Each ToR switch has two uplinks to a SiP switch. Therefore, each ToR EPS can have both of its uplinks connect to the EPSs of its own pod, or directly to pod EPSs in the other pod. As mentioned previously in Section 4.3, only uplink fibers are connected to SiP OCSs, not downlink fibers.

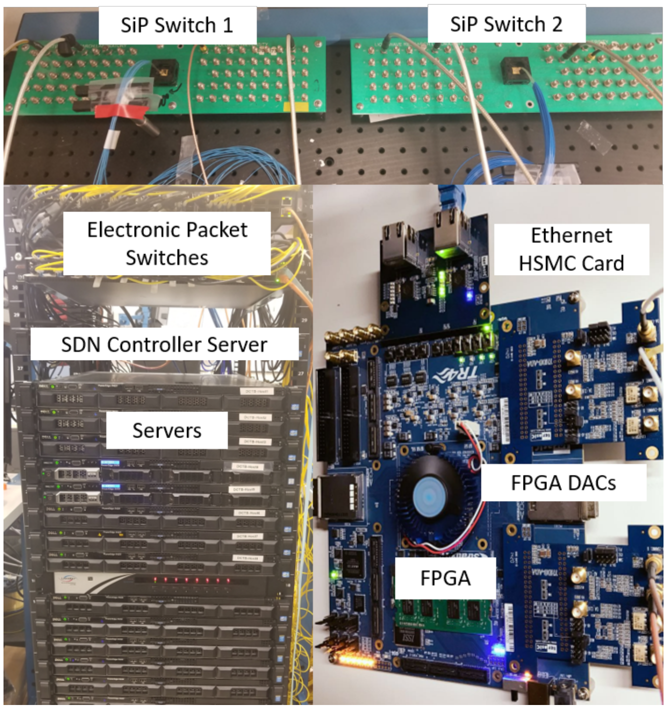

Figure 4: A photograph of our experimental testbed.

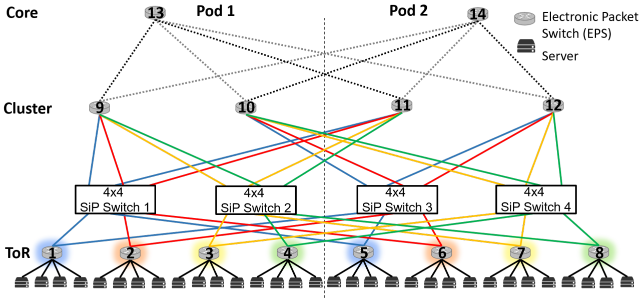

Figure 5: A 32-node experimental testbed with four SiP switches. Each electronic packet switch and silicon photonic switch is managed by a central SDN control plane.

The uplinks from ToR EPSs each transmit optical signals at a different wavelength, which are multiplexed together and enter the bus waveguide of the SiP device. Each SiP switch uses MRRs whose resonance frequencies are tuned to receive a specific wavelength from the wavelength-division multiplexed (WDM) signal. Depending on how the MRRs are tuned (using the bias electric signals from the FPGA in our control plane), different signals can be received on the output port of each MRR which allows the SiP device to be used as a switch. In our testbed, the wavelengths used (from the commercial SFP+ transceivers) are C38, C34, C30, and C26 on the ITU channel grid for each SiP switch.

5.2 Control Plane

A software-defined networking (SDN) control application acts as the central point of management for electronic packet and SiP OCSs. It contains the bandwidth steering algorithm. It is also used to add/remove flows from the flow table on the EPSs. Lastly, it can monitor the utilization of each link by processing byte count statistics from each EPS in the network.

Communication between the SDN controller and the EPSs uses the OpenFlow protocol. To manage the SiP switches, the SDN controller communicates with individual FPGAs connected by an Ethernet network, where each FPGA is responsible to apply the electric bias signals necessary for controlling the state of a certain number of SiP switches. During a switching operation, there is a certain amount of time in which the link is down and packets attempting to be sent during this time must be re-transmitted or delayed. In our testbed this is handled by standard TCP and Ethernet protocols.

6 EXPERIMENTAL TESTBED RESULTS

We ran the Gyrokinetic Toroidal Code (GTC) [2] HPC benchmark on our testbed described in Section 5.1. GTC is a scientific application that simulates the behavior of plasma microturbulence in the development of nuclear fusion power plants. We skeletonized the open-source code of this application in order to intensify generated traffic on our small-scale HPC testbed. The skeletonized version of GTC has its computational routines removed and therefore operates faster, but reproduces the same traffic behavior of the original application. We use message passing interface (MPI) [37] for synchronized communication between each computation node. We also use MPICH [3] to operate the GTC application over our testbed servers.

Figure 6 plots the throughput of the links in the upper fat tree layer (between the aggregation and core EPSs) over the entire runtime of the GTC application. Each colored line shows the throughput inside one link, identified in the legend through its source and destination EPSs (Figure 5). Traffic intensity in the upper-layer links illustrates the effect of configuring SiP switches to keep traffic in the lower-layer links (between ToR and aggregation EPSs). For the purposes of demonstrating our bandwidth steering concept, we place MPI ranks in a way that maximizes traffic at the top layer of the fat tree. Therefore as can be seen in the top plot of Figure 6, when we run the GTC application, each of the upper layer links of the standard fat tree topology are heavily utilized.

In the bandwidth-steered topology, SiP switches are configured in a way that allows both uplinks from each ToR switch to be directly connected to the destination's aggregation EPS, and therefore bypass the core layer, reducing packet switch hops by two. However, this topology causes any traffic that flows within the same pod to use upper layer links. However, as stated previously, the traffic generated is heavily inter-pod, which means that the bandwidth-steered topology is a better fit. This is reflected in the bottom plot of Figure 6, where we observe much lower bandwidth demand in upper-layer links. Only four of the upper layer links have traffic which ranges between 2 to 5 Gbps. The other links are virtually unused, because the bandwidth-steered topology has isolated the traffic to lower-layer links. We also notice that the application executes about 25% faster with the bandwidth-steered topology.

Following this observation, we can taper the bandwidth of the top-level links by removing links. Thus, we remove the links between EPS 10 and 13, EPS 12 and 13, EPS 9 and 14, and EPS 11 and 14 and perform a similar experiment to the previous paragraph. The

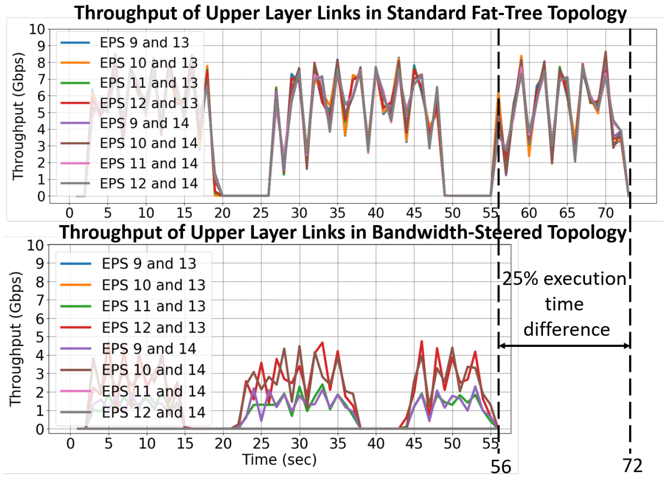

Figure 6: Throughput of upper-layer links (between aggregation and core EPSs) over the entire runtime of the GTC application for the standard fat tree topology (top) and the bandwidth-steered topology (bottom).

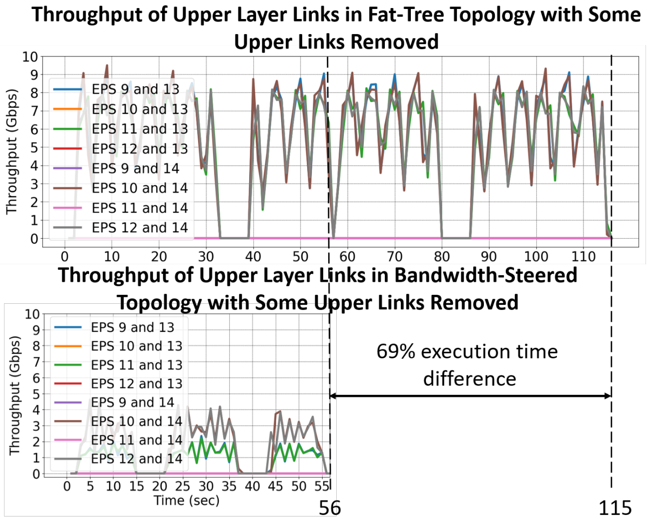

Figure 7: Throughput of upper-layer links (between aggregation and core packet switches) over the runtime of the GTC application for the standard fat tree topology (top) and the bandwidth-steered topology (bottom) *with some upper layer links removed* for reducing power consumption.

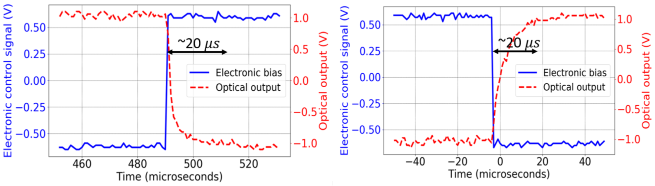

Figure 8: ON and OFF state switching times of a microring-resonator SiP switch.

results are shown in Figure 7. We note that with half its upper links removed, the standard fat tree topology now takes 115 seconds as opposed to 72 seconds to finish running the same application (69% difference). Also we observe that the links that are left are congested, with bandwidth demand reaching over 9 Gbps. Comparing this to the bandwidth-steered topology of Figure 6, we observe that it still takes the same amount of time of 56 seconds to finish, which matches expectations as we observed that the bandwidth steered topology did not use some of the upper layer links in the previous experiment. Therefore, the bandwidth-steered topology can tolerate more tapering with no performance penalty. Additionally, while real deployed HPC systems use much higher bandwidth links (e.g. 100G InfiniBand), the benefits of the bandwidth steering still apply, regardless of link data rate.

6.1 Silicon Photonic Switch Reconfiguration

We show the physical switching time of our SiP switches for ON and OFF states in Figure 8. This was done by sending a switching configuration command from the SDN controller, and measuring both the electrical bias signal sent from the DAC and the electrical signal from a photodiode connected to the SiP switch at the same time. Because we expect to reconfigure between application runs or within large phases of an application, a high speed switching time is not a priority. Therefore, we use thermo-optic switching of our MRR elements and observe approximately 20 μs for ON and OFF state switching.

After the SiP switch has changed to a new state, the optical signal is now traveling to a different receive port on the EPS. This EPS needs to contain the appropriate flow rule to allow for the incoming signal to be forwarded to the intended destination. The time required for the SDN controller to remove and add a new flow on each EPS is 78 μs per flow [82]. However, the factor that dominates the time in which the link is unusable is the polling time that current commercial EPSs apply to their ports. On our commercial EPSs, the polling time was measured to be 204 ms [82]. Therefore, the overall time that the link cannot be used during network reconfiguration is dominated by the EPS polling time. Even so, this reconfiguration time is orders of magnitude faster than the 17-second average job launch interval in Cori given that HPC benchmarks typically generate persistent traffic patterns that change slowly or never [16].

The large polling time is due to the limitation of current commercial packet switches that were not designed for handling changing endpoints. With our current hardware, the switch polling time is hard-coded into the operating system of the EPS and cannot be altered. To mitigate this, the EPS needs to be reprogrammed so that packets are aware of the status of the ports to know when they can be sent. A possible way to realize this is to have checks on the outport link status in addition to reading the Ethernet MAC address and IP addresses for each packet before sending. If the outgoing port is currently not available, the packet would need to be buffered. By designing packet switches with a much lower polling time, we can achieve much lower reconfiguration delays on the order of hundreds of microseconds [101], 10^9 less that the 17-second average job launch interval in Cori. This would make a practical application more straightforward but is outside the scope of this paper.

7 SYSTEM-SCALE EVALUATION

7.1 Method

We use a version of Booksim [47] modified to cycle-accurately simulate a fat tree topology with bandwidth steering and Infiniband-style lossless credit-based flow control [15]. We compare our "steered" fat tree against a vanilla fat tree. When traversing up the tree, we use per-packet oblivious routing among all shortest paths. For the steered networks, we evaluate the two routing strategies described in Section 4.4 and report the best results for each experiment. The number of lower-layer EPSs that connect to a higher-layer EPS (i.e., fat tree radix) is chosen such that the network best matches the number of tasks of each application. ToR EPSs have an equal number of "up" links to aggregation switches and "down" links to compute nodes. Similar to past work, we rely on endpoints to restore flow affinity [34, 67]. We pre-configure the network at the start of each simulation. Reconfiguration time was shown in Section 6.1.

EPSs are modeled in terms of performance and energy after the 36-port (36 inputs and 36 outputs) Mellanox 1U EDR 100Gb/s InfiniBand router with active optical cables [4]. SiP switches used for bandwidth steering have 16 input and 16 output ports and are modeled after [26]. For our SIP switches, the measured shift of resonance for a ring, as used in the receiver filter and optical switch, shows a thermal efficiency of 1nm/mW while for a silicon micro disk, as used in the transceiver modulator, the thermal efficiency is 0.54 nm/mW. For the transceiver power consumption, we assume expected improvements in fabrication and design that will give an average required shift of 1nm [13]. We assume tuning the input and output ring switching elements to half FSR consumes approximately 3 mW/ring. In our switch architecture the light beam traverses 2 rings per pass. We pessimistically assume all the paths in the switch are utilized. Therefore, our 16×16 SiPs have 96 mW total static power. Dynamic power is negligible. We note that as most papers do not include the stabilization due to environment and temperature variations during operation, we are also not including it here. Optimizing the power consumption of stabilization is an active area of research [31, 69, 86].

All links are optical and are modeled after Mellanox 100Gb/s QSFP28 MMF active optical cables [5]. For the distances in our network, all links have a single cycle of latency.

We use a collection of HPC traces that were identified by the US exascale computing program (ECP) as characteristic of future exascale applications [1] captured at NERSC's Cray XE06 (Hopper). To that collection we add a lattice quantum chromodynamics code ("MILC"). The primary communication in MILC is a 4D stencil with nearest neighbor and 3rd (Naik) neighbor traffic, followed by small message allreduce operations after each iteration. This is also a popular communication pattern in other HPC applications. MILC was captured at NERSC's Cray XC40 (Cori). All HPC traces were captured using DUMPI [8]. To study datacenter traffic, we use publicly-available traces from a production-level database pod that Facebook published to show their workload requirements [74]. Because in Facebook traces not all pods capture packets, we only retain traffic that is sourced and destined to pods that do capture packets. Otherwise we risk including sparse traffic such as requests without replies. Each trace name includes a number representing the number of tasks it is composed of. We map one MPI task or one source/destination ID in the case of Facebook traces to each network endpoint. We adjust the bytes-per-FLOP ratio in our simulations to not only normalize across traces since our traces were collected in different machines, but also to model lower bytes-per-FLOP ratios in future HPC systems following current trends and future predictions [6, 17].

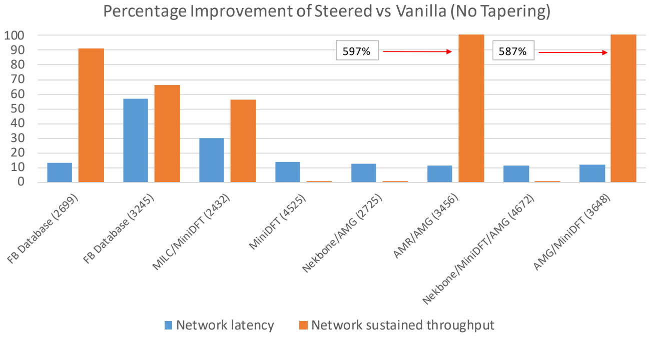

Figure 9: Average network percentage throughput improvement of the steered fat tree compared to a vanilla fat tree.

For all traces except Facebook, we randomize the placement of tasks to network endpoints. In addition, for experiments labelled with more than one benchmark name, we combine traffic from more than one application into a a single trace. This simulates the effects of fragmentation and multiple applications sharing the network in a production-level system, where a perfect linear placement is unrealistic. In fact, our random placement is pessimistic for bandwidth steering because it places many persistent flows beyond reach of our SiP switches for steering given their limited radix.

7.2 System-Scale Results

7.2.1 Throughput and Latency. Figure 9 summarizes throughput and network latency improvements of the steered fat tree over a standard vanilla fat tree with no bandwidth tapering (oversubscription) of the top layer. By average, bandwidth steering improves sustained network throughput by 1.7× and average network latency by 20%. Traces that stress the network more combined with providing the most opportunities for bandwidth steering to capture heavy communication are the ones demonstrating a higher throughput benefit. On the other hand, traces that do not provide a noticeable benefit do not stress the network adequately, generate intense communication between communication pairs beyond the reach of our SiP switches thus not allowing us to steer, or generate a more uniformly-loaded traffic pattern that the vanilla fat tree is well suited for.

We notice an almost 6× increase in sustained network throughput for two applications because they contain MiniDFT and the algebraic multigrid solver (AMG) trace, both of which generate heavy loads and communication of the appropriate distance to leverage bandwidth steering. However, even other traces demonstrate a solid improvement: MILC/MiniDFT 56%, Facebook (2699) 91%, and Facebook (3245) 66%. On the other hand, MiniDFT, Nekbone/AMG, and Nekbone/MiniDFT/AMG demonstrate insignificant throughput benefits due to a combination of more favorable random placement (that therefore does not benefit from bandwidth steering) and low communication load.

Table 2: Reduction by bandwidth steering compared to a vanilla fat tree with no tapering in the ratio of top-layer link utilization divided by lower-layer link utilization. The demonstrated reductions indicate how much less traffic uses the top-layer links.

Benchmark	Utilization reduction (%)
FB Database (2699)	74
FB Database (3245)	42
MILC/MiniDFT (2432)	15
MiniDFT (4525)	43
Nekbone/AMG (2725)	81
AMR/AMG (3456)	80
Nekbone/MiniDFT/AMG (4672)	69
AMG/MiniDFT (3648)	68
Average	59

Even though a vanilla fat tree with no bandwidth tapering can provide full sustained throughput for any traffic pattern that does not oversubscribe a destination, this is only feasible with perfect load balancing. In our experiments for the vanilla fat tree, we use oblivious routing that–in an untapered fat tree–can outperform adaptive routing due to imperfections of dynamic decisions with non-global data [73]. However, to avoid having to re-order flits within a packet, we apply routing decisions on a per-packet basis. Even though our maximum packet size is small (256 bytes), combined with transient imbalance from our dynamic traces, the end result is imperfect load balance. This explains why we notice a throughput improvement with steering compared to even an untapered vanilla fat tree. With perfect load balancing, throughput benefits would diminish but the relative gains of steering compared to a tapered vanilla fat tree, explained later, would magnify.

Network latency improves in all experiments due to a 5% to 20% reduced hop count from bandwidth steering because more flows reach their destination without having to use the top layer. The impact of steering to average hop count and latency depends on the fraction of traffic that utilizes steered paths. In the case of Facebook (3245), because more packets use steered paths proportionally to the other traces, average network latency is reduced by 57%. On the other hand, for most other benchmarks network latency is reduced by an average of 15%.

7.2.2 Link Utilization. To more directly demonstrate the impact of bandwidth steering, Table 2 shows the reduction with bandwidth steering compared to a vanilla fat tree of the ratio of utilization of upper-layer links divided by the utilization of lower-layer links. A larger improvement indicates that more communication does not use the higher-layer links. This utilization ratio directly correlates with latency reduction due to a decreased hop count even for benchmarks with no throughput benefit.

7.2.3 Bandwidth Tapering. Figure 10 illustrates the relative improvements of the steered topology compared to the vanilla fat tree by average across benchmarks as we reduce the available bandwidth at the top layer of the network. In other words, we show how much better steering performs compared to the vanilla fat tree as a function of tapering. We adjust the tapering factor by removing optical fibers at the top layer and then any core and aggregation switches to maintain high port utilization.

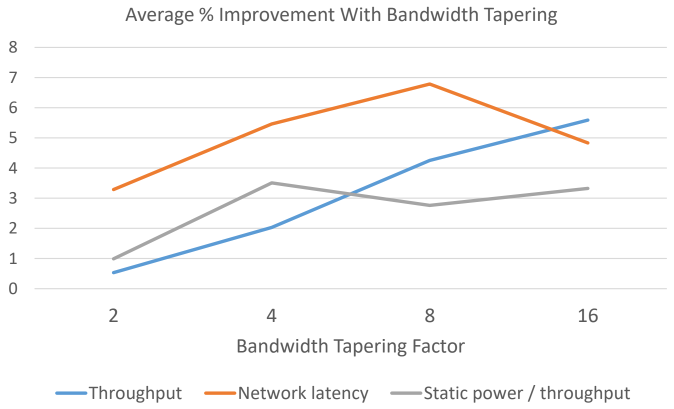

Figure 10: The percentage by which the gap between steered and unsteered topologies widens when tapering top-layer bandwidth. Tapering factor refers to how much less bandwidth is available at the top (aggregation-to-core) layer of the fat tree compared to the bottom (ToR-to-aggregation) layer. For instance, a bandwidth tapering factor of two means that the top layer has half the bandwidth than the bottom layer.

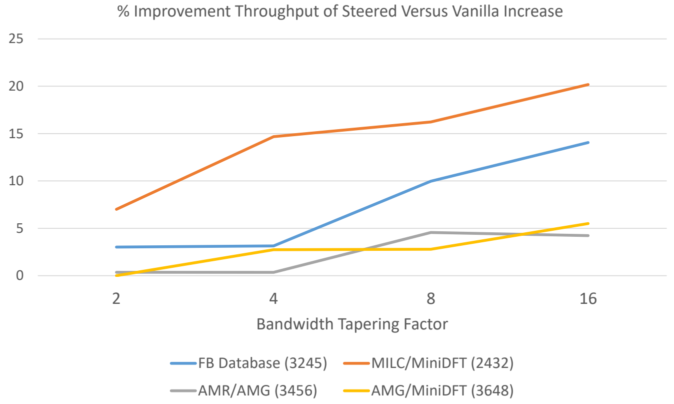

Figure 11: The percentage of throughput by which the gap between steered and unsteered topologies widens when tapering top-layer bandwidth for four benchmarks. The rest do not form adequate top-layer congestion to show an impact.

The vanilla fat tree experiences a decrease in throughput and increase in network latency as tapering increases. However, for a tapering factor of 16 we notice that in the steered network latency worsens as well, because even packets that do not benefit from steered paths are enough to form congestion at the top layer. The relative improvements of steered versus vanilla as tapering increases are driven by Facebook (3245), MILC/MiniDFT, AMR/AMG, and AMG/MiniDFT because these form adequate and sustained congestion in the upper layer to show the impact of tapering. For those benchmarks, throughput improvement reaches up to 20% (shown in Figure 11) and network latency 25%.

7.2.4 Power. When comparing power, we need to differentiate between dynamic and static power. Since dynamic power through our SiP switches is negligible, reducing hop count (EPS traversals) provides a proportional reduction in dynamic power. In addition, bandwidth steering reduces static power by preventing throughput

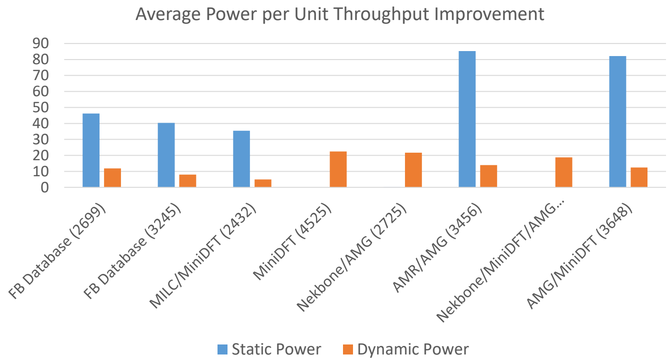

Figure 12: Power per unit throughput (dissipated power per byte transferred) reduction for bandwidth steering versus the vanilla fat tree.

loss with more aggressive bandwidth tapering. For example, tapering from a factor of 1 to 16 in the AMG/AMR application reduces the number of EPSs by 16%. Given that our commercial Mellanox EPSs have about three orders of magnitude higher static power due to their electrical components and SERDES compared to our SiPs, reducing the number of EPSs has a large impact to network-wide energy. Therefore, that 16% reduction in the number of EPSs combined with the reduced number of optical fibers translates to a 27% network static power reduction. Bandwidth steering makes obtaining these power reductions possible and not only prevents performance loss but also increases throughput in many of our applications.

To provide a more fundamental view of power, we define "power per unit throughput". Power per unit throughput is essentially power consumption divided by average throughput. This metric essentially shows how much power we need to pay per unit of throughput (byte transferred). In other words, for the same power consumption how much more throughput a topology with steering provides, or vice versa: how much less power for the same throughput.

Figure 12 shows that by average bandwidth steering consumes 14% less dynamic power and 36% less static power per unit throughput. This figure does not taper bandwidth, to show that the insignificant increases in dynamic and static power by our SiP switches are dwarfed by throughput and latency improvements. Traces with higher throughput improvements in Figure 9 tend to do better in static power. All traces show dynamic power improvement due to the reduced hop count by bandwidth steering even when communication load is low. The largest improvement comes from MiniDFT/AMG (82% static, 12% dynamic), AMG/AMR (85% static, 14% dynamic), and Facebook (2699) (46% static, 12% dynamic).

7.2.5 Latency Distribution. Bandwidth steering does not change the network diameter and therefore the maximum hop count since the property remains that once packets start traversing in the "down" direction, they do not switch back to the "up" direction. This means that the zero-load maximum latency remains unchanged. However, we notice a decrease in maximum network latency of up to 25% depending on the application, due to decreased congestion at the upper layer of the fat tree with bandwidth steering.

In addition, while bandwidth steering reduces average and mean network latency, it increases standard deviation. That is because the number of packets that do not need to use the top layer is increased with bandwidth steering compared to the vanilla fat tree. This creates a wider spread between short- and many-hop packets. This is intensified by randomizing the placement of tasks to network endpoints. In our experiments, this increase is from 5% to 54% depending on the application, with AMG/MiniDFT having the maximum and Facebook (3245) the minimum. All except two applications, AMG/MiniDFT with 54% and AMR/AMG with 37%, are in the 8% to 18% range. The gap narrows as we taper the top-layer bandwidth, because the steered network's distribution remains largely unchanged while the vanilla fat tree forms more congestion.

7.2.6 Fragmentation. Furthermore, we conduct experiments between traces with randomized placement (as has been the case thus far) and a "linear" placement where task i is placed on network endpoint i. This places neighboring tasks in neighboring network endpoints with no fragmentation and preserves all placement optimizations and communication locality that the application contains [30, 65]. The purpose is to clearly show how bandwidth steering can reconstruct locality in an execution where locality has been lost due to unfavorable placement and fragmentation on the system. For these experiments we run individual application traces.

In Nekbone, throughput drops about 45% from linear to randomized placements in the vanilla fat tree, illustrating that fragmentation ruins good locality in this application. However, bandwidth steering recovers locality and provides the same throughput in both linear and randomized placements, in particular 15% higher than vanilla linear and 1.3× higher than vanilla randomized.

The effect of randomization on bandwidth steering effectiveness varies. For MiniDFT with a linear placement, bandwidth steering provides a 24% throughput benefit. However, when steering on a randomized trace bandwidth steering provides marginal benefits because the lost locality is out of reach of our SiP switches. In contrast, we observe that AMG has a 50% higher throughput when steering on a randomized placement rather than a linear one. That is because AMG has some distant communication that is beyond the reach of our SiP switches in the linear case. Some of this locality can be recovered in the randomized case because some persistent communication pairs moved to be in range of our SiP switches.

7.2.7 SiP Switch Radix. Finally, we evaluate the impact of SiP switch radix (number of input and output ports). Larger-radix SiP switches provide more options to bandwidth steering because a single uplink from a ToR EPS can be connected to different output ports and thus more paths. Therefore, in our physical layout configuration of Figure 3, a larger SiP switch radix provides the opportunity to steer bandwidth between pods farther away. This has a performance impact for application traces with persistent communication between pods that can be captured by high-radix SiP switches but not lower radix ones, such as applications with a diverse set of communication endpoints. In our traces, if we reduce the radix of our SiP switches to 8×8 from the default 16×16, we notice a throughput drop of 0% to 5% for most traces with steering. However, other traces show a larger impact because radix affects a considerable number of their persistent flows. For example, AMG/MiniDFT experiences a 36% throughput reduction and a 21% increase in average latency, Facebook 3245 experiences a 35% throughput reduction and a 8% increase in average latency, while Nekbone/MiniDFT/AMG experiences only a 10% increase in average network latency.

We choose 16×16 SiP switches as default because they still have insignificant static power compared to our commercial EPSs and retain fast reconfiguration times. Optical switches with an even higher radix either by composing multiple SiP switches, using MEMS switches [70], or using AWGRs [55] switches would face reconfiguration time, signal loss, or other drawbacks which in turn would affect the system-wide efficiency of bandwidth steering. Our SiP switches do not require additional circuitry to boost the signal and assure signal integrity, and have negligible dynamic power and traversal latency. As our results show, bandwidth steering extracts significant benefit even with modest-radix SiP switches.

8 RELATED WORK

Bandwidth steering in system-scale networks appears in different contexts and makes use of different enabling technologies. Optiputer [29] 2002 started to bring telecom-grade optical circuit switches (Glimmerglass) into the datacenter, but was still largely focused on transcontinentally distributed datacenters. In a datacenter context, past work uses electrical four- or six-port converter switches to configure the network as either a fat tree or a random graph [97]. The algorithm that makes this decision is, however, not discussed and therefore experimental comparisons are infeasible. Other work performs steering of ToR switch uplinks with a greedy algorithm, but uses electrical circuit-switched switches that increase the performance and cost overhead of steering, and does not use steering to maximize bandwidth tapering among performance and cost [23]. Further work has shown the promise of random graphs in the datacenter, though practical implementation becomes more challenging [85]. More recent approaches offer reconfigurability in a two-level topology by cycling through pre-configured connections [63]. Other work uses free-space optics with ceiling mirrors to offer reconfigurability between ToR switches using a centralized controller [40].

Bandwidth steering is less studied in HPC despite its demonstrated potential [52, 100]. Early work, such as CLINT [32], focuses on dual-plane approaches where packet switches carry ephemeral traffic flows and configured dedicated circuit-switched paths carry persistent “elephant” traffic flows. However, the overhead of reconfiguration and the extra cost of operating both passive (circuit) and active (packet) commodity switch components have caused this approach to fall out of favor. Other similar work uses the same context but uses optical paths for persistent flows [16, 89, 90, 95]. HFAST 2005 [79] identified persistent patterns in HPC application communication patterns that could be exploited by using optical circuit switching to dynamically reconfigure interconnects to suit scientific applications. However, the “ fit-tree” algorithm [51] to guide the topological reconfiguration required very large-radix optical circuit switches. Recent work applies bandwidth steering using more practical-to-implement lower-radix circuit switching to a Dragonfly topology [53, 81] by using SiP switches to avoid group-to-group bottlenecks [96]. This is inherently a different goal than our objectives in a hierarchical topology. Other work uses wavelengths [54] or routing algorithms [91] to provide reconfigurability.

Further work examines the scheduling problem of reconfiguration by taking into account quality of service constraints [87], the reconfiguration delay that determines if reconfiguring the topology is prudent [94], considers the ratio of traffic that should use the electrical versus the optical resources [60], and minimizes the number of reconfigurations to achieve a certain effect [19]. Our work is synergistic with these techniques. Our work is also synergistic with task placement to increase locality [46, 64], performance and cost predictors [88], and global power optimization methods [88] because these approaches aim for a similar result as bandwidth steering but in a different way. Process migration can also reconstruct locality, but that takes up to tens of seconds which is orders of magnitude longer than our bandwidth steering approach [93].

Our work offers a complete solution to reduce procurement and energy costs of networks by minimizing idle bandwidth (maximizing bandwidth tapering) combined with latency gains. This motivates us to choose a hierarchical topology that allows us to pursue our goal, and in particular a fat tree as a representative hierarchical topology. Fat trees and their variants are extensively used in datacenters [21, 22] and HPC [10, 45, 49, 77] due to the favorable wiring properties and their ability to handle any traffic pattern with full bisection bandwidth assuming perfect load balance [43]. Furthermore. we describe a practical scalable algorithm, we offer full configuration flexibility to match traffic demands instead of pre-determined random graphs, we do not separate the network into disjoint optical and electrical parts which complicates scheduling, we demonstrate the impact of bandwidth steering to fragmentation, and we focus on HPC where bandwidth tapering is less studied. We co-design bandwidth steering with optical components that we design with minimal added cost and no performance penalty. We compare our approach to a canonical fat tree because the comparison between a fat tree and random graphs and other topologies is well studied [41, 66].

9 CONCLUSION

We describe bandwidth steering, a synergistic hardware–software approach that takes advantage of the inherent reconfigurability of emerging SiP switches to change (steer) the connectivity of the lower layers of a hierarchical topology in order to reduce top-layer utilization, without dividing the network into disjoint electrical and optical paths. This allows us to more aggressively taper the expensive long-distance bandwidth of higher layers. Bandwidth steering reconstructs locality that was lost due to system fragmentation and was hard to recover with task placement. We demonstrate bandwidth steering with a scalable maximum-weight matching algorithm in a hardware testbed and at system scale using simulations with a few thousand network endpoints. In our experimental testbed, bandwidth tapering reduces execution time by 69%, and is unaffected from tapering upper-layer bandwidth. At system scale, bandwidth steering reduces power per unit throughput by 36% static and 14% dynamic, and average network latency by up to 20%. These improvements magnify by up to 20% for throughput and 25% for network latency as we aggressively taper top-layer bandwidth. Bandwidth tapering is immediately applicable to HPC and datacenter networks using today’s network technology.

ACKNOWLEDGMENTS

This work was supported by ARPA-E ENLITENED Program (project award DE-AR00000843) and the Director, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

REFERENCES

[1] [n.d.]. Characterization of the DOE Mini-apps. https://portal.nersc.gov/project/CAL/doe-miniapps.htm. Accessed: 2019-02-16.

[2] [n.d.]. GTC. https://www.nersc.gov/users/computational-systems/cori/nersc-8-procurement/trinity-nersc-8-rfp/nersc-8-trinity-benchmarks/gtc/. (Accessed on 04/02/2019).

[3] [n.d.]. MPICH | High-Performance Portable MPI. https://www.mpich.org/. (Accessed on 04/02/2019).

[4] 2015. *Mellanox 1U EDR 100Gb/s InfiniBand Switch Systems Hardware User Manual Models: SB7700/SB7790*. Technical Report. http://www.mellanox.com/related-docs/user_manuals/1U_HW_UM_SB77X0.pdf

[5] 2018. *100Gb/s QSFP28 MMF Active Optical Cables*. Technical Report. https://www.mellanox.com/related-docs/prod_cables/PB_MFA1A00-Cxxx_100GbE_QSFP28_MMF_AOC.pdf

[6] 2018. *The Top500 HPC list*. https://www.top500.org/green500/lists/2018/11/

[7] A. H. Abdel-Gawad, M. Thottethodi, and A. Bhatele. 2014. RAHTM: Routing Algorithm Aware Hierarchical Task Mapping. In *SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 325–335. https://doi.org/10.1109/SC.2014.32

[8] Helgi Adalsteinsson, Scott Cranford, David A. Evensky, Joseph P. Kenny, Jackson Mayo, Ali Pinar, and Curtis L. Janssen. 2010. A Simulator for Large-Scale Parallel Computer Architectures. *Int. J. Distrib. Syst. Technol.* 1, 2 (April 2010), 57–73. https://doi.org/10.4018/jdst.2010040104

[9] M. Adda and A. Peratikou. 2017. Routing and Fault Tolerance in Z-Fat Tree. *IEEE Transactions on Parallel and Distributed Systems* 28, 8 (Aug 2017), 2373–2386. https://doi.org/10.1109/TPDS.2017.2666807

[10] Jung Ho Ahn, Nathan Binkert, Al Davis, Moray McLaren, and Robert S. Schreiber. 2009. HyperX: Topology, Routing, and Packaging of Efficient Large-scale Networks. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis (SC '09)*. Article 41, 11 pages. https://doi.org/10.1145/1654059.1654101

[11] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. 2008. A Scalable, Commodity Data Center Network Architecture. In *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication (SIGCOMM '08)*. ACM, 63–74. https://doi.org/10.1145/1402958.1402967

[12] George Almási, Charles Archer, José G. Castaños, C. Chris Erway, Philip Heidelberger, Xavier Martorell, José E. Moreira, Kurt Pinnow, Joe Ratterman, Nils Smeds, Burkhard Steinmacher-burow, William Gropp, and Brian Toonen. 2004. Implementing MPI on the BlueGene/L Supercomputer. In *Euro-Par 2004 Parallel Processing*, Marco Danelutto, Marco Vanneschi, and Domenico Laforenza (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 833–845.

[13] Xiang Meng Yanir London Jigesh Patel Madeleine Glick Keren Bergman Anthony Rizzo, Liang Yuan Dai. 2019. Ultra-low power consumption silicon photonic link design analysis in the AIM PDK. https://doi.org/10.1117/12.2508514

[14] A. Azzouni and G. Pujolle. 2018. NeuTM: A neural network-based framework for traffic matrix prediction in SDN. In *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*. 1–5. https://doi.org/10.1109/NOMS.2018.8406199

[15] P. Balaji, S. Bhagvat, D. K. Panda, R. Thakur, and W. Gropp. 2007. Advanced Flow-control Mechanisms for the Sockets Direct Protocol over InfiniBand. In *2007 International Conference on Parallel Processing (ICPP 2007)*. 73–73. https://doi.org/10.1109/ICPP.2007.14

[16] K. J. Barker, A. Benner, R. Hoare, A. Hoisie, A. K. Jones, D. K. Kerbyson, D. Li, R. Melhem, R. Rajamony, E. Schenfeld, S. Shao, C. Stunkel, and P. Walker. 2005. On the Feasibility of Optical Circuit Switching for High Performance Computing Systems. In *SC '05: Proceedings of the 2005 ACM/IEEE Conference on Supercomputing*. 16–16. https://doi.org/10.1109/SC.2005.48

[17] K. Bergman. 2018. Empowering Flexible and Scalable High Performance Architectures with Embedded Photonics. In *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, Vol. 00. 378. https://doi.org/10.1109/IPDPS.2018.00047

[18] G. Bhanot, A. Gara, P. Heidelberger, E. Lawless, J.C. Sexton, and R. Walkup. 2005. Optimizing task layout on the Blue Gene/L supercomputer. *IBM Journal on Research and Development* 49 (March/May 2005).

[19] Andrea Bianco, Paolo Giaccone, and Marco Ricca. 2016. Scheduling Traffic for Maximum Switch Lifetime in Optical Data Center Fabrics. *Comput. Netw.* 105, C (Aug. 2016), 75–88. https://doi.org/10.1016/j.comnet.2016.05.002

[20] Wim Bogaerts, Peter De Heyn, Thomas Van Vaerenbergh, Katrien De Vos, Shankar Kumar Selvaraja, Tom Claes, Pieter Dumon, Peter Bienstman, Dries Van Thourhout, and Roel Baets. 2012. Silicon microring resonators. *Laser & Photonics Reviews* 6, 1 (2012), 47–73.

[21] C. Camarero, C. Martinez, and R. Beivide. 2017. Random Folded Clos Topologies for Datacenter Networks. In *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 193–204. https://doi.org/10.1109/HPCA.2017.26

[22] C. Camarero, C. Martinez, and R. Beivide. 2018. On Random Wiring in Practicable Folded Clos Networks for Modern Datacenters. *IEEE Transactions on Parallel and Distributed Systems* 29, 8 (Aug 2018), 1780–1793. https://doi.org/10.1109/TPDS.2018.2805344

[23] Andromachi Chatzieleftheriou, Sergey Legtchenko, Hugh Williams, and Antony I. T. Rowstron. 2018. Larry: Practical Network Reconfigurability in the Data Center. In *NSDI*.

[24] Qixiang Cheng, Meisam Bahadori, Madeleine Glick, Sébastien Rumley, and Keren Bergman. 2018. Recent advances in optical technologies for data centers: a review. *Optica* 5, 11 (2018), 1354–1370.

[25] Qixiang Cheng, Meisam Bahadori, Madeleine Glick, Sébastien Rumley, and Keren Bergman. 2018. Recent advances in optical technologies for data centers: a review. *Optica* 5, 11 (Nov 2018), 1354–1370. https://doi.org/10.1364/OPTICA.5.001354

[26] Qixiang Cheng, Liang Yuan Dai, Nathan C. Abrams, Yu-Han Hung, Padraic E. Morrissey, Madeleine Glick, Peter O'Brien, and Keren Bergman. 2019. Ultralow-crosstalk, strictly non-blocking microring-based optical switch. *Photon. Res.* 7, 2 (Feb 2019), 155–161. https://doi.org/10.1364/PRJ.7.000155

[27] C. Clos. 1953. A study of non-blocking switching networks. *The Bell System Technical Journal* 32, 2 (March 1953), 406–424. https://doi.org/10.1002/j.1538-7305.1953.tb01433.x

[28] Jeffrey Dean and Luiz André Barroso. 2013. The tail at scale. *Commun. ACM* 56, 2 (2013), 74–80. https://doi.org/10.1145/2408776.2408794

[29] T. DeFanti, M. Brown, J. Leigh, O. Yu, E. He, J. Mambretti, D. Lillethun, and J. Weinberger. 2003. Optical Switching Middleware for the OptIPuter. *IEICE Trans. FUNDAMENTALS/COMMUN./ELECTRON./INF. & SYST* (Feb. 2003).

[30] Wolfgang Denzel, Jian Li, Peter Walker, and Yuho Jin. 2010. A Framework for End-to-End Simulation of High-performance Computing Systems. *Simulation* 86 (05 2010), 331–350. https://doi.org/10.1177/0037549709340840

[31] Po Dong, Robert Gatdula, Kwangwoong Kim, Jeffrey H. Sinsky, Argishti Melikyan, Young-Kai Chen, Guilhem de Valicourt, and Jeffrey Lee. 2017. Simultaneous wavelength locking of microring modulator array with a single monitoring signal. *Opt. Express* 25, 14 (Jul 2017), 16040–16046. https://doi.org/10.1364/OE.25.016040

[32] Hans Eberle and Nils Gura. 2002. Separated High-bandwidth and Low-latency Communication in the Cluster Interconnect Clint. In *Proceedings of the IEEE Conference on Supercomputing*.

[33] Jack Edmonds. 1965. Paths, Trees and Flowers. *Canad. J. Math* 17 (1965), 449–467.

[34] Greg Faanes, Abdulla Bataineh, Duncan Roweth, Tom Court, Edwin Froese, Bob Alverson, Tim Johnson, Joe Kopnick, Mike Higgins, and James Reinhard. 2012. Cray Cascade: A Scalable HPC System Based on a Dragonfly Network. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC '12)*. IEEE Computer Society Press, Article 103, 9 pages.

[35] P. Festa. 2014. A brief introduction to exact, approximation, and heuristic algorithms for solving hard combinatorial optimization problems. In *2014 16th International Conference on Transparent Optical Networks (ICTON)*. 1–20. https://doi.org/10.1109/ICTON.2014.6876285

[36] Klaus-Tycho Foerster, Manya Ghobadi, and Stefan Schmid. 2018. Characterizing the Algorithmic Complexity of Reconfigurable Data Center Architectures. In *Proceedings of the 2018 Symposium on Architectures for Networking and Communications Systems (ANCS '18)*. ACM, New York, NY, USA, 89–96. https://doi.org/10.1145/3230718.3230722

[37] Message P Forum. 1994. *MPI: A Message-Passing Interface Standard*. Technical Report. Knoxville, TN, USA.

[38] Yiannis Georgiou and Matthieu Hautreux. 2013. Evaluating Scalability and Efficiency of the Resource and Job Management System on Large HPC Clusters. In *Job Scheduling Strategies for Parallel Processing*, Walfredo Cirne, Narayan Desai, Eitan Frachtenberg, and Uwe Schwiegelshohn (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 134–156.

[39] Albert Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Patel, and Sudipta Sengupta. 2009. VL2: A Scalable and Flexible Data Center Network. *SIGCOMM Comput. Commun. Rev.* 39, 4 (Aug. 2009), 51–62. https://doi.org/10.1145/1594977.1592576

[40] Navid Hamedazimi, Zafar Qazi, Himanshu Gupta, Vyas Sekar, Samir R. Das, Jon P. Longtin, Himanshu Shah, and Ashish Tanwer. 2014. FireFly: A Reconfigurable Wireless Data Center Fabric Using Free-space Optics. In *Proceedings of the 2014 ACM Conference on SIGCOMM (SIGCOMM '14)*. 319–330. https://doi.org/10.1145/2619239.2626328

[41] Vipul Harsh, Sangeetha Abdu Jyothi, Inderdeep Singh, and Philip Brighten Godfrey. 2018. Expander Datacenters: From Theory to Practice. *CoRR* abs/1811.00212 (2018). arXiv:1811.00212

[42] Torsten Hoefler, Rolf Rabenseifner, Hubert Ritzdorf, Bronis R. de Supinski, Rajeev Thakur, and Jesper Larsson Tr&#x00e4;ff. 2011. The Scalable Process Topology Interface of MPI 2.2. *Concurr. Comput. : Pract. Exper.* 23, 4 (March 2011), 293–310. https://doi.org/10.1002/cpe.1643

[43] T. Hoefler, T. Schneider, and A. Lumsdaine. 2008. Multistage switches are not crossbars: Effects of static routing in high-performance networks. In *2008 IEEE International Conference on Cluster Computing*. 116–125. https://doi.org/10.1109/CLUSTR.2008.4663762

[44] Chintan Jain and Deepak Garg. 2012. Improved Edmond Karps Algorithm for Network Flow Problem. *International Journal of Computer Applications* 37 (01 2012). https://doi.org/10.5120/4576-6624

[45] Nikhil Jain, Abhinav Bhatele, Louis H. Howell, David Böhme, Ian Karlin, Edgar A. León, Misbah Mubarak, Noah Wolfe, Todd Gamblin, and Matthew L. Leininger. 2017. Predicting the Performance Impact of Different Fat-tree Configurations. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '17)*. Article 50, 13 pages. https://doi.org/10.1145/3126908.3126967

[46] E. Jeannot, G. Mercier, and F. Tessier. 2014. Process Placement in Multicore Clusters:Algorithmic Issues and Practical Techniques. *IEEE Transactions on Parallel and Distributed Systems* 25, 4 (April 2014), 993–1002. https://doi.org/10.1109/TPDS.2013.104

[47] Nan Jiang, D. U. Becker, G. Michelogiannakis, J. Balfour, B. Towles, D. E. Shaw, J. Kim, and W. J. Dally. 2013. A detailed and flexible cycle-accurate Network-on-Chip simulator. In *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 86–96. https://doi.org/10.1109/ISPASS.2013.6557149

[48] Nan Jiang, John Kim, and William J. Dally. 2009. Indirect Adaptive Routing on Large Scale Interconnection Networks. In *Proceedings of the 36th Annual International Symposium on Computer Architecture (ISCA '09)*. ACM, New York, NY, USA, 220–231. https://doi.org/10.1145/1555754.1555783

[49] W. Jiang, J. Qi, J. X. Yu, J. Huang, and R. Zhang. 2018. HyperX: A Scalable Hypergraph Framework. *IEEE Transactions on Knowledge and Data Engineering* (2018), 1–1. https://doi.org/10.1109/TKDE.2018.2848257

[50] Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Daniel Killebrew, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon. 2017. In-Datacenter Performance Analysis of a Tensor Processing Unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA '17)*. ACM, New York, NY, USA, 1–12. https://doi.org/10.1145/3079856.3080246

[51] Shoaib Kamil, Leonid Oliker, Ali Pinar, and John Shalf. 2010. Communication Requirements and Interconnect Optimization for High-End Scientific Applications. *IEEE Trans. Parallel Distrib. Syst.* 21, 2 (2010), 188–202. https://doi.org/10.1109/TPDS.2009.61

[52] S. Kamil, A. Pinar, D. Gunter, M. Lijewski, L. Oliker, and J. Shalf. 2007. Reconfigurable Hybrid Interconnection for Static and Dynamic Scientific Applications. In *Proceedings of the ACM International Conference on Computing Frontiers*.

[53] J. Kim, W. J. Dally, S. Scott, and D. Abts. 2008. Technology-Driven, Highly-Scalable Dragonfly Topology. In *2008 International Symposium on Computer Architecture*. 77–88.

[54] A. K. Kodi and A. Louri. 2011. Energy-Efficient and Bandwidth-Reconfigurable Photonic Networks for High-Performance Computing (HPC) Systems. *IEEE Journal of Selected Topics in Quantum Electronics* 17, 2 (March 2011), 384–395. https://doi.org/10.1109/JSTQE.2010.2051419

[55] C. Lea. 2015. A Scalable AWGR-Based Optical Switch. *Journal of Lightwave Technology* 33, 22 (Nov 2015), 4612–4621. https://doi.org/10.1109/JLT.2015.2479296

[56] Benjamin G. Lee. 2018. Photonic switching platform for datacenters enabling rapid network reconfiguration. , 10560 - 10560 - 5 pages. https://doi.org/10.1117/12.2292149

[57] Jacob S. Levy, Alexander Gondarenko, Mark A. Foster, Amy C. Turner-Foster, Alexander L. Gaeta, and Michal Lipson. 2009. CMOS-compatible multiple-wavelength oscillator for on-chip optical interconnects. *Nature Photonics* 4 (20 Dec 2009), 37 EP –.

[58] E. A. LeÃşn, I. Karlin, A. Bhatele, S. H. Langer, C. Chambreau, L. H. Howell, T. D'Hooge, and M. L. Leininger. 2016. Characterizing Parallel Scientific Applications on Commodity Clusters: An Empirical Study of a Tapered Fat-Tree. In *SC '16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 909–920.

[59] Y. Li, H. Liu, W. Yang, D. Hu, and W. Xu. 2016. Inter-data-center network traffic prediction with elephant flows. In *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*. 206–213. https://doi.org/10.1109/NOMS.2016.7502814

[60] He Liu, Matthew K. Mukerjee, Conglong Li, Nicolas Feltman, George Papen, Stefan Savage, Srinivasan Seshan, Geoffrey M. Voelker, David G. Andersen, Michael Kaminsky, George Porter, and Alex C. Snoeren. 2015. Scheduling Techniques for Hybrid Circuit/Packet Networks. In *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies (CoNEXT '15)*. Article 41, 13 pages. https://doi.org/10.1145/2716281.2836126

[61] Robert Lucas, James Ang, Keren Bergman, Shekhar Borkar, William Carlson, Laura Carrington, George Chiu, Robert Colwell, William Dally, Jack Dongarra, Al Geist, Rud Haring, Jeffrey Hittinger, Adolfy Hoisie, Dean Micron Klein, Peter Kogge, Richard Lethin, Vivek Sarkar, Robert Schreiber, John Shalf, Thomas Sterling, Rick Stevens, Jon Bashor, Ron Brightwell, Paul Coteus, Erik Debenedictus, Jon Hiller, K. H. Kim, Harper Langston, Richard Micron Murphy, Clayton Webster, Stefan Wild, Gary Grider, Rob Ross, Sven Leyffer, and James Laros III. 2014. DOE Advanced Scientific Computing Advisory Subcommittee (ASCAC) Report: Top Ten Exascale Research Challenges. (2 2014). https://doi.org/10.2172/1222713

[62] Lailong Luo, Deke Guo, Wenxin Li, Tian Zhang, Junjie Xie, and Xiaolei Zhou. 2015. Compound graph based hybrid data center topologies. *Frontiers of Computer Science* 9, 6 (01 Dec 2015), 860–874. https://doi.org/10.1007/s11704-015-4483-5

[63] William M. Mellette, Rob McGuinness, Arjun Roy, Alex Forencich, George Papen, Alex C. Snoeren, and George Porter. 2017. RotorNet: A Scalable, Low-complexity, Optical Datacenter Network. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '17)*. 267–280. https://doi.org/10.1145/3098822.3098838

[64] G. Michelogiannakis, K. Z. Ibrahim, J. Shalf, J. J. Wilke, S. Knight, and J. P. Kenny. 2017. 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)*. 228–237. https://doi.org/10.1109/CCGRID.2017.33

[65] S. H. Mirsadeghi, J. L. TrÃd'ff, P. Balaji, and A. Afsahi. 2017. Exploiting Common Neighborhoods to Optimize MPI Neighborhood Collectives. In *2017 IEEE 24th International Conference on High Performance Computing (HiPC)*. 348–357. https://doi.org/10.1109/HiPC.2017.00047

[66] M. A. Mollah, P. Faizian, M. S. Rahman, X. Yuan, S. Pakin, and M. Lang. 2018. A Comparative Study of Topology Design Approaches for HPC Interconnects. In *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. 392–401. https://doi.org/10.1109/CCGRID.2018.00066

[67] Giovanni Neglia, Vincenzo Falletta, and Giuseppe Bianchi. 2004. Is TCP Packet Reordering Always Harmful?. In *Proceedings of the The IEEE Computer Society's 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS '04)*. 87–94.

[68] L. Nie, D. Jiang, L. Guo, S. Yu, and H. Song. 2016. Traffic Matrix Prediction and Estimation Based on Deep Learning for Data Center Networks. In *2016 IEEE Globecom Workshops (GC Wkshps)*. 1–6. https://doi.org/10.1109/GLOCOMW.2016.7849067

[69] K. Padmaraju, D. F. Logan, T. Shiraishi, J. J. Ackert, A. P. Knights, and K. Bergman. 2014. Wavelength Locking and Thermally Stabilizing Microring Resonators Using Dithering Signals. *Journal of Lightwave Technology* 32, 3 (Feb 2014), 505–512. https://doi.org/10.1109/JLT.2013.2294564

[70] I. Plander and M. Stepanovsky. 2017. MEMS technology in optical switching. In *2017 IEEE 14th International Scientific Conference on Informatics*. 299–305. https://doi.org/10.1109/INFORMATICS.2017.8327264

[71] Rastin Pries, Michael Jarschel, Daniel Schlosser, Michael Klopf, and Phuoc Tran-Gia. 2012. Power Consumption Analysis of Data Center Architectures. In *Green Communications and Networking*, Joel J. P. C. Rodrigues, Liang Zhou, Min Chen, and Aravind Kailas (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 114–124.

[72] Francesco Redaelli, Marco D. Santambrogio, and Donatella Sciuto. 2008. Task Scheduling with Configuration Prefetching and Anti-fragmentation Techniques on Dynamically Reconfigurable Systems. In *Proceedings of the Conference on Design, Automation and Test in Europe (DATE '08)*. 519–522. https://doi.org/10.1145/1403375.1403500

[73] G. Rodriguez, C. Minkenberg, R. Beivide, R. P. Luijten, J. Labarta, and M. Valero. 2009. Oblivious routing schemes in extended generalized Fat Tree networks. In *2009 IEEE International Conference on Cluster Computing and Workshops*. 1–8. https://doi.org/10.1109/CLUSTR.2009.5289145

[74] Arjun Roy, Hongyi Zeng, Jasmeet Bagga, George Porter, and Alex C. Snoeren. 2015. Inside the Social Network's (Datacenter) Network. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication (SIGCOMM '15)*. ACM, 123–137. https://doi.org/10.1145/2785956.2787472

[75] SAMTEC. 2019. *PCIe Optical Half Cables Application Note*. Technical Report. http://suddendocs.samtec.com/notesandwhitepapers/pcie_half_cable_app_

note.pdf

[76] V. Sasikala and K. Chitra. 2018. All optical switching and associated technologies: a review. *Journal of Optics* 47, 3 (01 Sep 2018), 307–317. https://doi.org/10.1007/s12596-018-0452-3

[77] S. Scott, D. Abts, J. Kim, and W. J. Dally. 2006. The BlackWidow High-Radix Clos Network. In *33rd International Symposium on Computer Architecture (ISCA'06)*. 16–28. https://doi.org/10.1109/ISCA.2006.40

[78] John Shalf, Sudip Dosanjh, and John Morrison. 2011. Exascale Computing Technology Challenges. In *Proceedings of the 9th International Conference on High Performance Computing for Computational Science (VECPAR'10)*. Springer-Verlag, Berlin, Heidelberg, 1–25.

[79] J. Shalf, S. Kamil, L. Oliker, and D. Skinner. 2005. Analyzing Ultra-Scale Application Communication Requirements for a Reconfigurable Hybrid Interconnect. In *Proc. SC2005: High performance computing, networking, and storage conference*.

[80] Y. Shen, A. Gazman, Z. Zhu, M. Y. The, M. Hattink, S. Rumley, P. Samadi, and K. Bergman. 2018. Autonomous Dynamic Bandwidth Steering with Silicon Photonic-Based Wavelength and Spatial Switching for Datacom Networks. In *2018 Optical Fiber Communications Conference and Exposition (OFC)*. 1–3.

[81] Yiwen Shen, Storm Madeleine Glick, and Keren Bergman. 2019. Silicon photonic-enabled bandwidth steering for resource-efficient high performance computing. In *Proceedings Volume 10946, Metro and Data Center Optical Networks and Short-Reach Links II (SPIE)*, Vol. 10946.

[82] Yiwen Shen, Maarten H. N. Hattink, Payman Samadi, Qixiang Cheng, Ziyiz Hu, Alexander Gazman, and Keren Bergman. 2018. Software-defined networking control plane for seamless integration of multiple silicon photonic switches in Datacom networks. *Opt. Express* 26, 8 (Apr 2018), 10914–10929. https://doi.org/10.1364/OE.26.010914

[83] Arjun Singh, Joon Ong, Amit Agarwal, Glen Anderson, Ashby Armistead, Roy Bannon, Seb Boving, Gaurav Desai, Bob Felderman, Paulie Germano, Anand Kanagala, Jeff Provost, Jason Simmons, Eiichi Tanda, Jim Wanderer, Urs Holzle, Stephen Stuart, and Amin Vahdat. 2015. Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Googleś Datacenter Network. In *Sigcomm '15*.

[84] Ankit Singla, P. Brighten Godfrey, and Alexandra Kolla. 2014. High Throughput Data Center Topology Design. In *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation (NSDI'14)*. USENIX Association, 29–41.

[85] Ankit Singla, Chi-Yao Hong, Lucian Popa, and P. Brighten Godfrey. 2012. Jellyfish: Networking Data Centers Randomly. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation (NSDI'12)*. USENIX Association, 17–17.

[86] C. Sun, M. Wade, M. Georgas, S. Lin, L. Alloatti, B. Moss, R. Kumar, A. H. Atabaki, F. Pavanello, J. M. Shainline, J. S. Orcutt, R. J. Ram, M. PopoviÄĞ, and V. StojanoviÄĞ. 2016. A 45 nm CMOS-SOI Monolithic Photonics Platform With Bit-Statistics-Based Resonant Microring Thermal Tuning. *IEEE Journal of Solid-State Circuits* 51, 4 (April 2016), 893–907. https://doi.org/10.1109/JSSC.2016.2519390

[87] Mohammad Mahdi Tajiki, Behzad Akbari, and Nader Mokari. 2017. Optimal Qos-aware Network Reconfiguration in Software Defined Cloud Data Centers. *Comput. Netw.* 120, C (June 2017), 71–86. https://doi.org/10.1016/j.comnet.2017.04.003

[88] K. Tang, X. He, S. Gupta, S. S. Vazhkudai, and D. Tiwari. 2018. Exploring the Optimal Platform Configuration for Power-Constrained HPC Workflows. In *2018 27th International Conference on Computer Communication and Networks (ICCCN)*. 1–9. https://doi.org/10.1109/ICCCN.2018.8487322

[89] Y. Tang, H. Guo, and J. Wu. 2018. OCBridge: An Efficient Topology Reconfiguration Strategy in Optical Data Center Network. In *2018 Optical Fiber Communications Conference and Exposition (OFC)*. 1–3.

[90] Y. Tarutani, Y. Ohsita, and M. Murata. 2014. Virtual network reconfiguration for reducing energy consumption in optical data centers. *IEEE/OSA Journal of Optical Communications and Networking* 6, 10 (Oct 2014), 925–942.

[91] E. Tasoulas, E. G. Gran, T. Skeie, and B. D. Johnsen. 2016. Fast hybrid network reconfiguration for large-scale lossless interconnection networks. In *2016 IEEE 15th International Symposium on Network Computing and Applications (NCA)*. 101–108. https://doi.org/10.1109/NCA.2016.7778601

[92] Yutaka Urino, Tatsuya Usuki, Junichi Fujikata, Masashige Ishizaka, Koji Yamada, Tsuyoshi Horikawa, Takahiro Nakamura, and Yasuhiko Arakawa. 2014. High-density and wide-bandwidth optical interconnects with silicon optical interposers. *Photon. Res.* 2, 3 (Jun 2014), A1–A7. https://doi.org/10.1364/PRJ.2.0000A1

[93] Chao Wang, Frank Mueller, Christian Engelmann, and Stephen L. Scott. 2008. Proactive Process-level Live Migration in HPC Environments. In *Proceedings of the 2008 ACM/IEEE Conference on Supercomputing (SC '08)*. IEEE Press, Piscataway, NJ, USA, Article 43, 12 pages. http://dl.acm.org/citation.cfm?id=1413370.1413414

[94] Chang-Heng Wang, Tara Javidi, and George Porter. 2015. End-to-end scheduling for all-optical data centers. 406–414. https://doi.org/10.1109/INFOCOM.2015.7218406

[95] Guohui Wang, David G. Andersen, Michael Kaminsky, Michael Kozuch, T. S. Eugene Ng, Konstantina Papagiannaki, Madeleine Glick, and Lily B. Mummert. 2009. Your Data Center Is a Router: The Case for Reconfigurable Optical Circuit Switched Paths.. In *HotNets*, Lakshminarayanan Subramanian, Will E. Leland, and Ratul Mahajan (Eds.). ACM SIGCOMM.

[96] Ke Wen, Payman Samadi, Sébastien Rumley, Christine P. Chen, Yiwen Shen, Meisam Bahadroi, Keren Bergman, and Jeremiah Wilke. 2016. Flexfly: Enabling a Reconfigurable Dragonfly Through Silicon Photonics. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '16)*. IEEE Press, Piscataway, NJ, USA, 15:1–15:12.

[97] Yiting Xia, Xiaoye Steven Sun, Simbarashe Dzinamarira, Dingming Wu, Xin Sunny Huang, and T. S. Eugene Ng. 2017. A Tale of Two Topologies: Exploring Convertible Data Center Network Architectures with Flat-tree. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '17)*. ACM, 295–308. https://doi.org/10.1145/3098822.3098837

[98] Xu Yang and Zhiling Lan. 2016. Cooperative Batch Scheduling for HPC Systems.

[99] P. Yebenes, J. Escudero-Sahuquillo, P. J. Garcia, F. J. Quiles, and T. Hoefler. 2017. Improving Non-minimal and Adaptive Routing Algorithms in Slim Fly Networks. In *2017 IEEE 25th Annual Symposium on High-Performance Interconnects (HOTI)*. 1–8. https://doi.org/10.1109/HOTI.2017.11

[100] Keren Bergman Yiwen Shen, Madeleine Strom Glick. 2019. Silicon photonic-enabled bandwidth steering for resource-efficient high performance computing. , 10946 - 10946 - 9 pages. https://doi.org/10.1117/12.2509060

[101] Ziyi Zhu, Yiwen Shen, Yishen Huang, Alexander Gazman, Maarten Hattink, and Keren Bergman. 2019. Flexible Resource Allocation Using Photonic Switched Interconnects for Disaggregated System Architectures, In Optical Fiber Communication Conference (OFC) 2019. *Optical Fiber Communication Conference (OFC) 2019*, M3F.3. https://doi.org/10.1364/OFC.2019.M3F.3