POLITECNICO DI TORINO
Repository ISTITUZIONALE

Reconfigurable Optical Datacom Networks by Self-supervised Learning

Original
Reconfigurable Optical Datacom Networks by Self-supervised Learning / Liu, C.-Y., Chen, X., Proietti, R., Li, Z., Yoo, S.J.B.. - ELETTRONICO. - (2021), pp. 23-27. (2021 ACM SIGCOMM 2021 Workshop on Optical Systems, OptSys 2021 USA 23 August 2021) [10.1145/3473938.3474509].

Availability:
This version is available at: 11583/2973498 since: 2022-11-30T13:37:56Z

Publisher:
Association for Computing Machinery, Inc

Published
DOI:10.1145/3473938.3474509

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
ACM postprint/Author's Accepted Manuscript

(Article begins on next page)

16 June 2026

Reconfigurable Optical Datacom Networks by Self-supervised Learning

Che-Yu Liu[★], Xiaoliang Chen[†], Roberto Proietti[★], Zhaohui Li[†], S. J. Ben Yoo[★]

[★]University of California, Davis, [†]Sun Yat-Sen University
chenxliang7@mail.sysu.edu.cn,sbyoo@ucdavis.edu

ABSTRACT

This paper presents a self-supervised machine learning approach for cognitive reconfiguration in a Hyper-X-like flexible-bandwidth optical interconnect architecture. The proposed approach makes use of a clustering algorithm to learn the traffic patterns from historical traces. A heuristic algorithm is developed for optimizing the connectivity graph for each identified traffic pattern. Further, to mitigate the scalability issue induced by frequent clustering operations, we parameterize the learned traffic patterns by a deep neural network classifier. The classifier is trained offline by supervised learning to enable classification of traffic matrices during online operations, thereby facilitating cognitive reconfiguration decision making. Simulation results show that compared with a static all-to-all interconnection, the proposed approach can improve throughput by up to 1.76× while reducing end-to-end packet latency and flow completion time by up to 2.8× and 25×, respectively.

CCS CONCEPTS

• **Networks** → **Data center networks**; **Network performance evaluation**; **Network architectures**; **Network management**;

KEYWORDS

Reconfigurable datacom networks, Flexible bandwidth optical interconnect, Clustering, Deep neural network

ACM Reference Format:
Che-Yu Liu[★], Xiaoliang Chen[†], Roberto Proietti[★], Zhaohui Li[†], S. J. Ben Yoo[★]. 2022. Reconfigurable Optical Datacom Networks by Self-supervised Learning. In *Proceedings of ACM Conference (OptSys'2021).* ACM, New York, NY, USA, 5 pages. https://doi.org/

1 INTRODUCTION

Today's data center (DC) and high-performance computing (HPC) systems leverage multi-hierarchy tree topologies [1, 14] and electronic packet switching. Their poor scalability and energy efficiency make it difficult to sustain the ever-growing demand for cloud services and HPC applications (e.g., scientific computing, deep learning workloads). Thanks to recent advances in scalable and manufacturable silicon-photonic technologies [13], flat and disruptive optical interconnect architectures scaling up to tens of thousands of terminals by enabling direct (thus, low-diameter) wavelength switching between racks [4, 13, 16] have been recognized as promising solutions for meeting the above challenges.

On the other hand, the application-driven nature of the communication patterns between the computing nodes makes DC/HPC traffic exhibit high dynamicity and spatial nonuniformity. Therefore, effective reconfiguration schemes are essential to fully exploit the benefits of optical interconnects and ensure consistent system performance [5, 18]. The authors in [16] proposed a hybrid optical/electrical switching-based reconfigurable network topology leveraging heuristic algorithms for determining the wiring and bandwidth steering schemes for different traffic profiles. In [17], the authors leveraged a deep learning approach to learn the mapping between the traffic distribution and the optimal topology configuration. Note that reconfiguration operations can be very costly and even cause traffic disruptions as they involve reconfiguring optical switches and routing tables in all related top-of-racks (ToRs). Hence, it is desirable to implement effective reconfiguration policies guiding when reconfiguration should be performed. The work in [15] applied deep reinforcement learning to learn autonomic reconfiguration policies from repeated trials and errors. However, such an approach was only demonstrated under a small-scale system and can hardly scale. Our previous work in [7] proposed a cognitive reconfiguration policy relying on performance (i.e., latency, packet loss rate) estimations by deep neural network (DNN) models. Nevertheless, training the DNN models requires collecting a large amount of performance data, introducing non-negligible operation overheads. Meanwhile, the approach still makes use of a fixed-threshold-based policy applied to the performance estimations.

In this paper, we target a Hyper-X-type flexible-bandwidth optical interconnect architecture [2] and propose a self-supervised machine learning (ML) approach for cognitive reconfiguration decision making. We first discuss the data and control plane arrangement of the considered interconnect architecture in Section 2. In Sections 3 and 4, we detail the proposed cognitive reconfiguration design and present the evaluation results. Finally, we conclude the paper with Section 5.

2 SYSTEM ARCHITECTURE

This paper considers a Hyper-X-like directly connected architecture, where the ToR switches are fully connected in each dimension. Each ToR switch connects to a certain number of servers (or computing nodes). Fig. 1 shows an architecture with N ToRs in each dimension of a 2D Hyper-X. Each column (or row) is a cluster (often referred to as portable data centers - PODs) with N ToR switches per POD. The intra-POD connectivity in each dimension is augmented with wavelength-space selective optical switches that allow reconfiguring the topology and bandwidth (number of links between ToR pairs) based on certain algorithms and policies (which will be discussed in Section 3). Fig. 1(a-top) shows the default connectivity

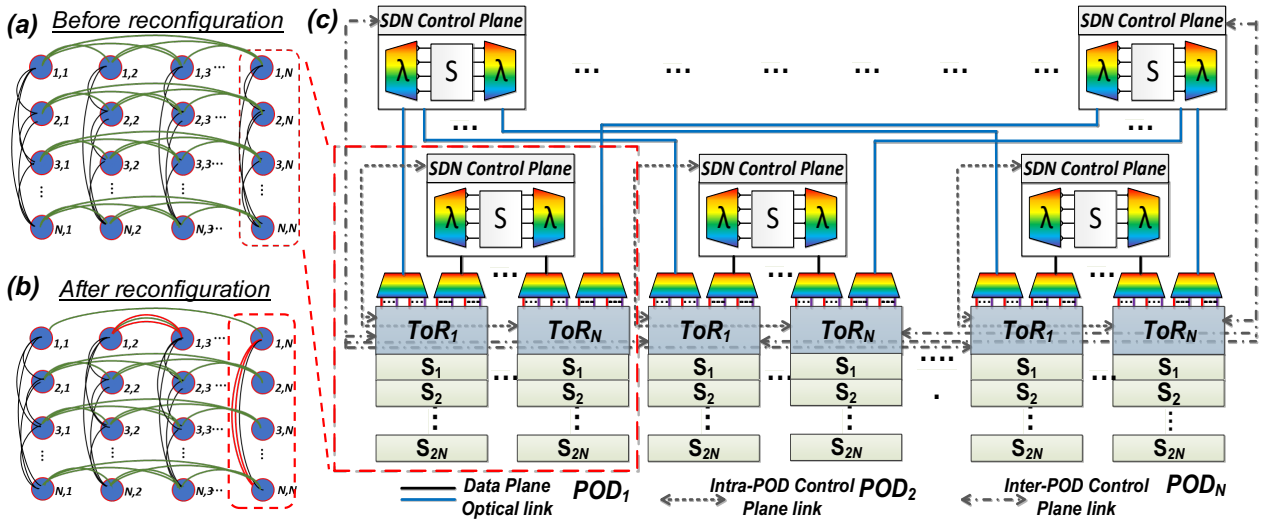

Figure 1: (a) 2D-Hyper-X architecture: each blue node represents a ToR switch. (Top) Default all-to-all connectivity. (Bottom) Example of reconfiguration to increase the bandwidth between specific ToR pairs. (b) Detailed diagram of the architecture with $N = k/4$ PODs, with each PODs containing $2N$ servers and N k-port ToRs ($k = 4N$).

inside a POD, which is all-to-all. This is ideal for the case where the traffic is evenly distributed. In the common case of uneven traffic distributions with hotspots, it can be desirable to break the all-to-all connectivity to give more bandwidth (links) where and when is needed. This is shown in Fig. 1(a-bottom), where the red links represent the links/bandwidth that have been steered from other ToRs. As shown in Fig. 1(b), each ToR switch uses $2N$ ports for servers (intra-rack) and $2N$ ports for inter-rack (N ports for each dimension of the Hyper-X). This means that the oversubscription of the network is $1:1$ (other oversubscription values are possible and represent a trade-off between scalability, performance and cost). We assume that WDM TRXs with N wavelengths are used for inter-rack ports and that the radix of the optical switches is N. If we consider state-of-the-art switch ASICs with $k = 4N = 128$ ports at 100 Gb/s, this 2D Hyper-X architecture can scale to $k^3/32 = 65,536$ servers, while requiring optical switches with limited radix $N = k/4 = 32$ and number of wavelengths $N = k/4 = 32$.

To facilitate scalable management of the 2D Hyper-X architecture, we consider a distributed software-defined networking (SDN) control plane arrangement, where a dedicated SDN controller is deployed for each switching domain (i.e., a POD or inter-POD switching fabric). The SDN controllers perform parallel reconfiguration and exchange necessary information to facilitate system-wide optimization.

3 RECONFIGURATION DESIGN

Fig. 2 shows the layout of control plane functionality facilitating the proposed reconfiguration design. An SDN controller communicates with the related ToRs and optical switches through SDN protocols (e.g., OpenFlow, P4) to perform real-time monitoring of data plane states and distribute configuration instructions. Interfacing with the SDN controller, the reconfiguration module drives cognitive reconfiguration decision making. Specifically, the reconfiguration

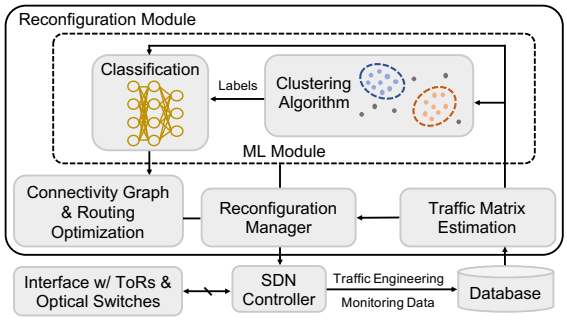

Figure 2: Layout of control plane functionality.

module makes use of a compound of ML and traditional optimization tools. The reconfiguration manager constantly estimates the traffic demand between ToRs by retrieving information such as flow counters from the traffic engineering database [10]. Each time the reconfiguration manager observes changes in the estimated traffic matrix, it invokes the ML module to generate a reconfiguration solution. Let $\mathbb{D}$ denote the set of traffic matrices recorded over time and D_t represent the traffic matrix monitored at time t. An unsupervised learning module is first employed to learn traffic patterns from $\mathbb{D}$, i.e., by clustering traffic matrices in $\mathbb{D}$ according to their mutual distances. In this work, we applied the density-based clustering algorithm developed in [9] owing to its capability of detecting clusters of arbitrary shapes. Meanwhile, we adopt the Euclidean distance as the distance metric. Because traffic matrices belonging to the same clusters exhibit similar patterns, we make the connectivity graph & routing optimization module (will be detailed later) precompute a common configuration scheme for each cluster $\mathbb{C}_i$. This way, the reconfiguration manager can promptly

Algorithm 1 Procedures of calculating connectivity graphs.

1: **Input**: $S, N, \mathbb{C}_i$
2: **Output**: G
3: $x_s \leftarrow N-1$, $y_s \leftarrow N-1, \forall s \in S$
4: $G_{u,v} \leftarrow 0, \forall u, v \in S$
5: $D \leftarrow \frac{1}{|\mathbb{C}_i|} \sum_{D_t \in \mathbb{C}_i} D_t, W \leftarrow D$
6: **for** $n \in [1, N-1]$ **do**
7: $r_s \leftarrow 0, \forall s \in S$
8: **for** each ToR $s \in S$ **do**
9: $w_u \leftarrow \begin{cases} \max\{W_{s,u} \cdot x_s, W_{u,s} \cdot y_u\}, x_s \cdot y_u > 0 \\ -\infty, else \end{cases}, \forall u \in S \setminus s$
10: **if** $r_s == 1$ OR $\max_u w_u \leq 0$ **then**
11: continue
12: **end if**
13: $u^* \leftarrow \arg\max_u (w_u)$
14: $x_{u^*} \leftarrow x_{u^*} - 1, y_{u^*} \leftarrow y_{u^*} - 1$
15: $x_s \leftarrow x_s - 1, y_s \leftarrow y_s - 1$
16: $G_{s,u^*} \leftarrow G_{s,u^*} + 1, G_{u^*,s} \leftarrow G_{u^*,s} + 1$
17: $W_{s,u^*} \leftarrow W_{s,u^*} - C, W_{u^*,s} \leftarrow W_{u^*,s} - C$
18: $r_{u^*} \leftarrow 1$
19: **end for**
20: **end for**

decide the reconfiguration solution by simply identifying whether D_t and D_{t-1} belong to the same cluster, and if not, which cluster D_t resides in. More importantly, by exploiting the inherent structure of the traffic data, we can determine the most effective reconfiguration point while largely eliminating unnecessary reconfiguration operations.

Each execution of the unsupervised learning module requires traversing the whole data set, which can be computationally costly when the size of $\mathbb{D}$ is large. Therefore, we further introduce a supervised learning module for online operations, which we train with knowledge extracted by the unsupervised learning module, similar to the idea presented in [6]. More specifically, we design the supervised learning module with a DNN architecture, which takes as input D_t and outputs the predicted cluster ID. The complexity of the supervised learning module (after it has been trained) is only related to the scale of the DNN implemented, and therefore, does not incur scalability issues. In addition, for D_t that cannot be classified into any of the existing clusters (i.e., an outlier), the reconfiguration manager instructs the SDN controller to maintain all-to-all interconnects for the related ToRs.

Next, we elaborate on the design of the connectivity graph & routing optimization module. In particular, we apply a two-phase optimization approach where connectivity graphs and routing schemes are determined successively. Algorithm 1 shows the procedures of calculating the connectivity graph G for a set of ToRs S and a traffic cluster $\mathbb{C}_i$. In *Lines* 3-4, we initialize the number of available input (x_s) and output ports (y_s) in each ToR s to be $N-1$, and the number of links configured for each ToR pair as 0. *Line* 5 calculates the mean of $\mathbb{C}_i$ as the reference traffic matrix. The for-loop covering *Lines* 6-20 traverses all the ports and iteratively adds links to ToR pairs with larger amounts of traffic to be served. Within the loop, we first set r_s as 0 for all ToRs with *Line* 7 to indicate that the corresponding ports of these ToRs have not yet been touched. The inner loop from *Line* 8 to 19 determines for each ToR s a target ToR u^* to whom the current port should be connected. In *Line* 9, we assign each other ToR u a weight by calculating the products of the amounts of traffic remaining to be served between s and u ($W_{s,u}$ and $W_{u,s}$) and the numbers of available ports in them (x_s and y_u), i.e., to encourage adding links to ToR pairs with larger demands and more spare ports. Note that, since the connectivity between ToRs needs to be bidirectional, we set the larger value of a product from the two directions as the weight. Meanwhile, we exclude ToRs that do not have any spare port by setting the related weights to be negative infinity. With *Lines* 10-12, we skip ToRs whose current ports have already been configured or for which sufficient capacities have been allocated. *Line* 13 picks ToR u^* with the largest weight as the target ToR. Finally, we update the connectivity graph G, the information of port utilization and traffic volume remaining to be served, and ToR port utilization indicator in *Lines* 14-18. After G has been determined, we apply the equal-cost multipath routing algorithm [11] to determine the routing scheme for each flow, for its advantage of facilitating load balancing and robustness against network failures.

4 PERFORMANCE EVALUATION

We evaluated the performance of the proposed design by packet-level simulations using the NetBench simulator [12]. A POD of 64 ToRs was considered. In the simulations, we assumed that each link has a capacity of 10 Gbps and is associated with a delay of 20 ns. The buffer size of each ToR port was set as 150,000 bytes. We made servers generate packet flows following Poisson processes. The source and destination nodes of each flow were selected based on the traffic distributions derived from the traffic matrices for evaluation. The upper bound of Alizadeh Web Search distribution [3] was used to determine the sizes of flows. To emulate various traffic patterns, we used both synthetic traffic matrices and real HPC application traces, i.e., algebraic multi-grid (AMG), center for exascale simulation of advanced reactors (CESAR) and FFT [7, 8]. The two synthetic traffic matrices were generated by randomly selecting 30% and 50% of the ToR pairs to generate demands following a uniform distribution. We expanded the data set by adding zero-mean random noises to each of these traffic matrices and obtained a total of 1,000 instances.

We first assessed the performance of the ML modules in terms of their accuracy in detecting clusters or classifying traffic matrices. For the clustering algorithm, we set ε and $MinPts$ to be 46 and 100, respectively, using the method discussed in [6]. With this setup, the clustering algorithm successfully detected five clusters (cluster ID ranging from 0 to 4) along with 77 outliers (labeled as belonging to cluster -1) from the entire data set. Based on the clustering results, we implemented and trained a DNN classifier of two fully-connected layers (15 neurons in each layer). The normalized confusion matrix in Fig. 3(a) shows the accuracy performance of the DNN classifier on the testing data set (15% of the entire data set randomly drawn). The results indicate that the DNN classifier can achieve a classification accuracy of at least 94.5% for samples belonging to the five clusters. For outliers with much fewer samples used in training, the classifier can still achieve accuracy over 88%.

Next, we evaluated the system-level performance of the reconfiguration design and compared it with a baseline always adopting an all-to-all interconnection scheme (like in a regular Hyper-X network). Figs. 3(b) and (c) present the results of the average packet latency and $99th$ percentile FCT for the AMG trace under different flow arrival rates λ (i.e., network loads). We can see that the performance of the two approaches is comparable at low loads

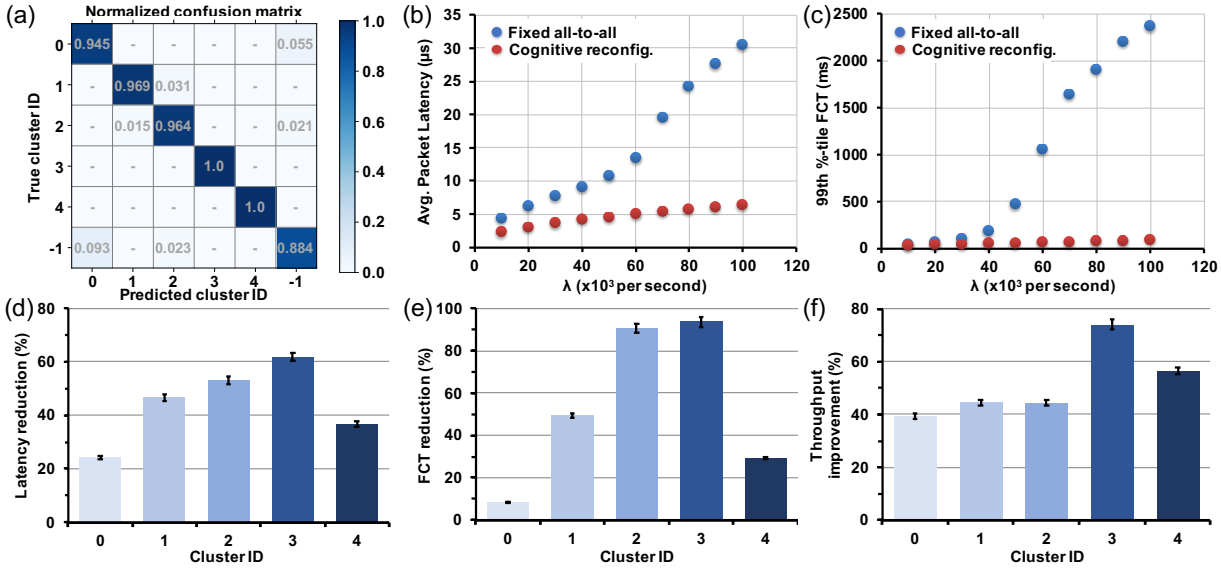

Figure 3: Results: (a) classification accuracy of the DNN classifier; (b)-(c) comparisons of average packet latency and $99th$ percentile flow completion time (FCT) between the proposed design and the baseline; (d)-(f) latency, FCT and throughput improvements from the proposed design against the baseline with respect to different clusters.

($\lambda < 40,000$) when the links are underutilized. As we kept increasing λ, the performance of the reconfiguration design remains stable, whereas that of the baseline deteriorates dramatically. This is because the proposed approach can effectively steer the bandwidth to where it is needed, thus offering much higher throughput and mitigating link congestion. The results for traffic matrices belonging to other clusters show similar trends. We summarize the performance gains of the proposed design against the baseline with respect to different clusters ($\lambda = 60,000$) in Figs. 3(d)-(f). The results show that the proposed design outperforms the baseline in all the cases. It can also be seen that the benefits from reconfiguration vary significantly (e.g., the reduction in FCT from the proposed design can range from below 10% to as high as 96%), depending on how skewed traffic matrices are [e.g., CESAR (cluster 3) versus FFT (cluster 4)].

5 CONCLUSION

In this paper, we proposed a self-supervised ML approach to assist cognitive bandwidth reconfiguration considering a Hyper-X-type optical interconnect architecture with link bandwidth flexibility. Evaluation results show that our approach achieves significant latency, FCT and throughput improvements over the regular all-to-all baseline. Future research directions include: (1) developing tools for joint connectivity graph and routing optimization and (2) extending the proposed design to hyper-scale systems composed of multiple parallel intra- and inter-POD switching fabrics.

6 ACKNOWLEDGEMENTS

This work was supported in part by the NSF ECCS Award #1611560 and the NSFC Project U2001601.

REFERENCES

[1] N. R. Adiga, M. A. Blumrich, D. Chen, P. Coteus, A. Gara, M. E. Giampapa, P. Heidelberger, S. Singh, B. D. Steinmacher-Burow, T. Takken, M. Tsao, and P. Vranas. 2005. Blue Gene/L torus interconnection network. *IBM Journal of Research and Development* 49, 2.3 (2005), 265–276.

[2] 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*. 1–11.

[3] Mohammad Alizadeh, Shuang Yang, Milad Sharif, Sachin Katti, Nick McKeown, Balaji Prabhakar, and Scott Shenker. 2013. PFabric: Minimal near-Optimal Datacenter Transport. *SIGCOMM Comput. Commun. Rev.* 43, 4 (Aug. 2013), 435–446.

[4] Hitesh Ballani, Paolo Costa, Raphael Behrendt, Daniel Cletheroe, István Haller, Krzysztof Jozwik, F. Karinou, S. Lange, Kai Shi, B. Thomsen, and H. Williams. 2020. Sirius: A Flat Datacenter Network with Nanosecond Optical Switching. *In Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication* (2020).

[5] Nick Barrow-Williams, Christian Fensch, and Simon Moore. 2009. A communication characterisation of Splash-2 and Parsec. In *2009 IEEE International Symposium on Workload Characterization (IISWC)*. 86–97.

[6] X. Chen, B. Li, R. Proietti, Z. Zhu, and S. J. B. Yoo. 2019. Self-Taught Anomaly Detection With Hybrid Unsupervised/Supervised Machine Learning in Optical Networks. *J. Lightw. Technol.* 37, 7 (April 2019), 1742–1749.

[7] Xiaoliang Chen, Roberto Proietti, Marjan Fariborz, Che-Yu Liu, and S. J. Ben Yoo. 2021. Machine-learning-aided cognitive reconfiguration for flexible-bandwidth HPC and data center networks [Invited]. *J. Opt. Commun. Netw.* 13, 6 (2021), C10–C20.

[8] U.S. DOE. 2020. Characterization of DOE Mini-apps. https://portal.nersc.gov/project/CAL/doe-miniapps.htm. (11 2020). Accessed: 11-2020.

[9] M. Ester, H. Kriegel, J. Sander, and X. Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of KDD*. 226–231.

[10] N. Farrington, G. Porter, S. Radhakrishnan, H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat. 2010. Helios: A Hybrid Electrical/Optical Switch Architecture for Modular Data Centers. *SIGCOMM Comput. Commun. Rev.* 40, 4 (Aug. 2010), 339–350.

[11] Christian Hopps. 2000. Analysis of an Equal-Cost Multi-Path Algorithm. RFC 2992. (Nov. 2000).

[12] Simon Kassing, Asaf Valadarsky, and Ankit Singla. 2016. NetBench. https://github.com/ndal-eth/netbench. (2016).

[13] Gengchen Liu, Roberto Proietti, Marjan Fariborz, Pouya Fotouhi, Xian Xiao, and S. J. Ben Yoo. 2020. Architecture and Performance Studies of 3D-Hyper-FleX-LION for Reconfigurable All-to-All HPC Networks. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '20)*. IEEE Press, Article 26, 16 pages.

[14] F. Petrini and M. Vanneschi. 1997. k-ary n-trees: high performance networks for massively parallel architectures. In *Proceedings 11th International Parallel Processing Symposium*. 87–93.

[15] Yu Shang, Xiaoliang Chen, Roberto Proietti, Bingli Guo, Shanguo Huang, and S. J. Ben Yoo. 2019. DeepAutonet: Self-driving Reconfigurable HPC System with Deep Reinforcement Learning. In *Asia Communications and Photonics Conference (ACPC) 2019*. S3C.4.

[16] Min Yee Teh, Zhenguo Wu, and Keren Bergman. 2020. Flexspander: augmenting expander networks in high-performance systems with optical bandwidth steering. *J. Opt. Commun. Netw.* 12, 4 (2020), B44–B54.

[17] Mowei Wang, Yong Cui, Shihan Xiao, Xin Wang, Dan Yang, Kai Chen, and Jun Zhu. 2018. Neural Network Meets DCN: Traffic-Driven Topology Adaptation with Deep Learning. *Proc. ACM Meas. Anal. Comput. Syst.* 2, 2, Article 26 (June 2018), 25 pages.

[18] Guojun Yuan, Roberto Proietti, Xiaoli Liu, Alberto Castro, Dawei Zang, Ninghui Sun, CheYu Liu, Zheng Cao, and S. J. B. Yoo. 2016. ARON: Application-Driven Reconfigurable Optical Networking for HPC Data Centers. In *ECOC 2016; 42nd European Conference on Optical Communication*. 1–3.