IEEE Copyright Notice

Accepted to be Published in: Proceedings of the 2019 IEEE 26th Symposium on High-Performance Interconnects (HOTI 26), Aug. 15-16, 2019 Santa Clara, CA, USA

The First Supercomputer with HyperX Topology: A Viable Alternative to Fat-Trees?

(Extended Abstract)

J. Domke and S. Matsuoka
RIKEN Center for Computational Science (R-CCS), RIKEN

I. R. Ivanov, Y. Tsushima, T. Yuki, A. Nomura, and S. Miura
Tokyo Institute of Technology

N. McDonald, D. L. Floyd, and N. Dubé
Hewlett Packard Enterprise (HPE)

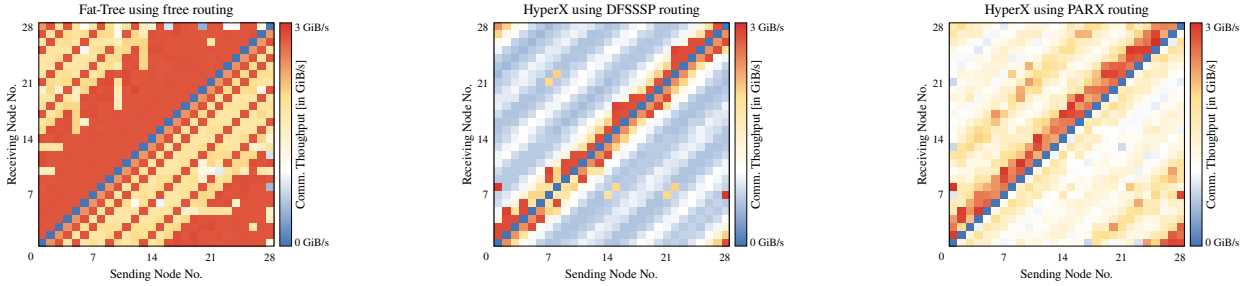


Figure 1: Measured throughput with mpiGraph for 28 compute nodes of our Fat-Tree/HyperX, dual-rail supercomputer

***Abstract*—The state-of-the-art topology for modern supercomputers are Folded Clos networks, a.k.a. Fat-Trees. The node count in these massively parallel systems is steadily increasing. This forces an increased path length, which limits gains for latency-sensitive applications, because the port count of modern switches cannot be scaled accordingly. Additionally, the deployment methodology for today's Fat-Trees requires the extensive use of costly active optical cables. A novel, yet only theoretically investigated, alternative is the low-diameter HyperX. To perform a fair side-by-side comparison between a 3-level Fat-Tree and a 12x8 HyperX, we constructed the world's first 3 Pflop/s supercomputer with these two networks. We show through a variety of benchmarks that the HyperX, together with our novel communication pattern-aware routing, can challenge the performance of traditional Fat-Trees.**

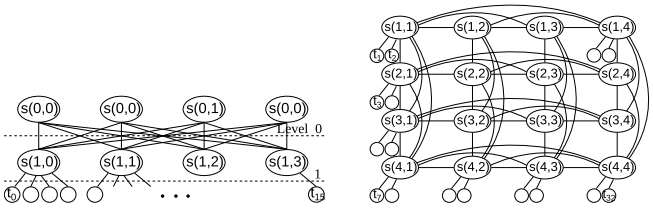
1. Introduction

The recent generation of supercomputer underwent a drastic scale-out effect, e.g., reaching the extremes of K or Sunway TaihuLight [1] with over 80,000 and over 40,000 nodes, respectively, to tackle the ending of Moore's law. The interconnect in these HPC systems faces increasing demands for ultra-low latency from legacy HPC workloads, and new needs for high throughput of large messages from deep learning (DL) frameworks. Deploying Clos or Fat-Tree topologies will provide the needed throughput, but at a high cost. Furthermore, additional tree levels (to achieve this scale-out) will negatively affect the observable latency. Low-diameter, "electrically-optimized" topologies have been proposed, such as Dragonfly [2], Dragonfly+ [3], or Slimfly [4]. Another alternative, assuming economical co-packaging of optics and fiber shuffles, is the HyperX [5] which could provide high-throughput and low network dimensionality.

Similar to other direct and low-diameter topologies, the HyperX is affected by a potential bottleneck between two adjacent switches, which is usually bypassed by non-minimal, adaptive routing [6]. Figure 1 visualizes this adverse effect of static, minimal routing on a HyperX. Here, the measured average intra-rack bisection bandwidth drops from 2.26 GiB/s (Fat-Tree; left) to mere 0.84 GiB/s for our HyperX, since at worst seven traffic flows may be minimally routed along a single cable. To mitigate the issue (seen in right-most heatmap), we develop the non-minimal, static PARX routing, increasing the average bandwidth by 66% to 1.39 GiB/s.

This extended abstract outlines how we rewire a many-node, peta-scale supercomputer, the recently decommissioned TSUBAME2 system, to analyze the viability of the HyperX topology. We modify TSUBAME2's second network plane while keeping the 1st plane in the original state, and develop mitigation strategies for the aforementioned bottlenecks resulting from the lack of adaptive routing. This approach facilitates our fair comparison by allowing us to empirically quantify the capabilities of both the HyperX and Fat-Tree topology. To stress our network topologies, we sample from a broad set of available benchmarks and attempt various usage scenarios. The first set of benchmarks are of synthetic form (to evaluate raw MPI performance), while the second set is comprised of scientific (proxy-)applications, e.g., selected from the Exascale Computing Project (ECP) proxy applications [7], which are used by the HPC community for procurement purposes. In short, this paper highlights:

1) our fair, in-depth comparison of HyperX and Fat-Tree using numerous HPC workloads,
2) a novel topology- and communication-aware routing for HyperX to mitigate bottlenecks, and
3) that even a statically routed HyperX is competitive for many realistic HPC and DL workloads.



(a) 4-ary 2-tree with 16 nodes (b) 32-node 2D(4x4) HyperX

Figure 2: Two full-bisection bandwidth topologies (Fig. 2a and Fig. 2b; some nodes omitted) sketched; Each rack of our 672-node HPC system connects with 2 edge switches to a 3-level Fat-Tree and with 4 switches to a 12x8 HyperX

2. k-ary n-tree vs. HyperX Topology

The k-ary n-tree topology [8], a.k.a. Folded Clos or Fat-Tree, is the de facto standard topology to connect large-scale HPC systems. Figure 2a depicts a small 4-ary 2-tree. The advantage of Folded Clos networks is that all admissible (non-incast) traffic theoretically yields the same throughput [8], given a congestion-free message forwarding [9].

The HyperX topology [5] is a generalization of all flat integer lattice networks with fully connected dimensions, e.g., HyperCube, Flattened Butterfly, see [6, Sec. 3.3] for details. The network shown in Fig. 2b is a full-bisection bandwidth, 2-dimensional version. This topology is designed as low-diameter, direct network to fit current high-radix routers. Theoretically, given appropriate message forwarding, a HyperX network with merely 50% bisection bandwidth can still provide 100% throughput for uniform random traffic. This approach reduces the network costs of the system.

Unfortunately, no large-scale system with HyperX topology exists. However, the Tokyo Institute of Technology decommissioned their TSUBAME2 supercomputer recently, and allowed us to modify its topology to construct the first prototype. TSUBAME2 consisted of over 1,500 compute and auxiliary nodes, achieving a ≈6 Pflop/s theoretical peak performance. Its nodes were interconnected by two full-bisection bandwidth Fat-Trees [10], using QDR InfiniBand (IB).

The IB networks employ four 36-port Voltaire 4036 edge switches per rack (two per network plane), 6 Voltaire Grid Director 4700 switches per plane, and thousands of fiber optic cables (AOC) in between. Both 18-ary 3-trees were undersubscribed, meaning that only 15 compute nodes (not 18) are connected to each edge switch. To achieve a fair comparison of Fat-Tree vs. HyperX topology, we re-wire one of the original network planes as HyperX. Unfortunately, the lack of adaptive routing (AR) features in our IB hardware, see [11, Sec. 18.2.4.3] and Section 3, should theoretically diminish our achievable results. Given our labor, hardware, and facility constraints, we constructed the largest possible HPC system with 12x8 2D HyperX network connecting to 7 nodes per switch. Hence, this topology has a bisection bandwidth of 57.1%. Our network hosts 672 compute nodes (distributed across 24 racks) which connect to 96 IB switches. Excluding the additional auxiliary rack, our new system features a theoretically peak performance of 2.7 Pflop/s.

3. Bottleneck Mitigation for HyperX

The HyperX requires AR, like other low-diameter topologies, to prevent congestion on the shortest path which may only be connected by a single (or few) cable. The middle plot of Figure 1 demonstrates this bottleneck. Because our QDR IB technologies lacks AR, and only offers static and flow-oblivious routing, we develop two mitigation measures.

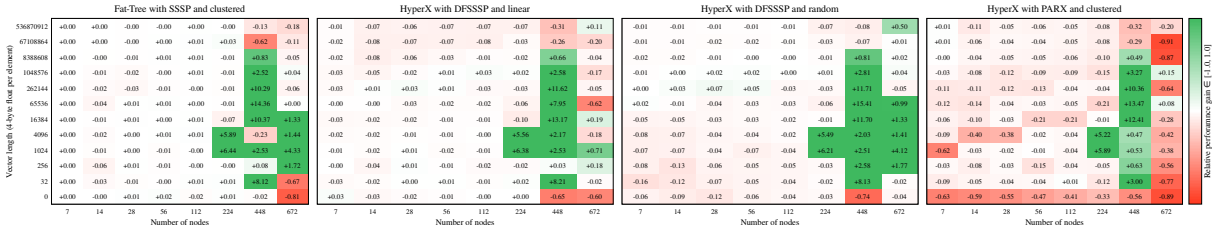
One trivial solution to the bottleneck issue, arising from static routing, is to distribute the allocated nodes of an application across the system, with the potential negative effect on latencies. Hence, the number of node-adjacent switches, and path diversity between them, is increased. An optimized mapping, for example through topology- or routing-awareness [12, 13], is desired. However, we generally would not endorse using HyperX without adaptive routing, and hence only test a simple random assignment heuristic.

Alternatively, redirecting the traffic flow along non-minimal paths may be possible to utilize more links to increase throughput, as done by AR on Dragonfly [2, 14]. Unfortunately, statically routing along non-minimal paths is not trivial in IB, due to the destination-based virtual cut-through switching [11, ¶3.4.3]. Due to space constraints, we only sketch our approach hereafter. We design a novel topology-aware routing for HyperX and messaging scheme for applications, called **P**attern-**A**ware **R**outing for Hyper**X**, which satisfies the following four criteria: (1) route small messages along shortest paths; (2) utilize non-minimal paths for large messages; (3) $\forall\{$node pairs$\}$ the choice between (1) and (2) exists; and (4) the routing is fault-tolerant and loop-/deadlock-free. Criteria (1–3) aim for low latency and high throughput, whereas criterion (4) is necessary for our imperfect HyperX, i.e., few faulty cables. Deadlock-freedom became apparent after initial tests with SSSP routing [15].

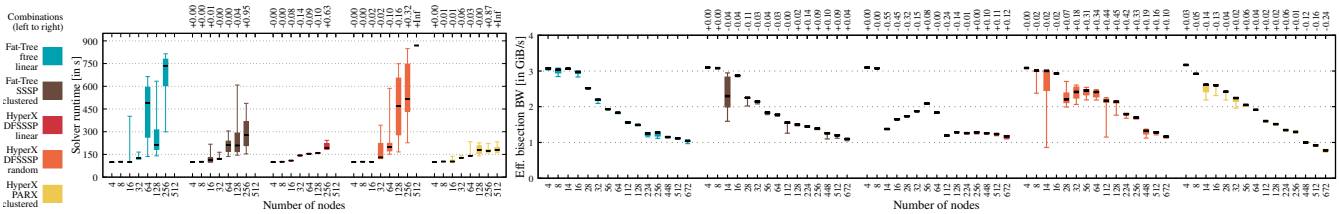
PARX relies on LMC-based multi-pathing [11, ¶7.11.1] and smart HyperX-quadrant enumeration to set up 4 simultaneous paths towards each compute node. By virtually removing switch-to-switch links in certain quadrants, we can enforce valid path calculations along non-shortest paths. Further tuning of static routes is achieved by pre-collecting low-level P2P message information [16] for each application, and inject this data into PARX before our real benchmarking (similar to SAR [17]). Lastly, we modify Open MPI's `bfo` point-to-point messaging layer (PML) [18] to chose between (1) and (2) depending on message size (512 KiB threshold) instead of allowing it to do concurrent multi-pathing.

4. Methodology

We evaluate HyperX's raw latency/throughput for small messages, common in HPC [19], and large message size, typical in modern deep learning, with three benchmark sets: a subset of *Intel MPI Benchmarks (IMB)* [20], *Netgauge's eBB* [21], and *Baidu's DeepBench Allreduce* [22]. Additionally, we test a variety of HPC workloads by using a subset of the Exascale Computing Project (ECP) proxy-apps [7], RIKEN R-CCS' Fiber Suits [23], and Trinity [24] and CORAL [25] procurement applications: (1) *AMG*, *CoMD*,



(a) Baidu's Allreduce (relative gain over Fat-Tree / ftree / linear)



(b) ECP's SWFFT benchmark

(c) Netgauge's eBB (effective bisection bandwidth)

Figure 3: Network / scientific application benchmarks; Fig. 3a: relative perf. gain over baseline for best measured result; Fig. 3b/3c: whisker plots for 10 trials (# above indicates relative gain over baseline for the best run of both combinations)

MiniFE, and *SWFFT*; (2) *FFVC*, *mVMC*, and *NTChem*; (3) *MIMD Lattice Computation (MILC)*; and (4) *LLNL's qb@ll (Qbox)*, the last two of which are known to be highly network-intensive. Lastly, we run three commonly known HPC benchmarks which are used to establish world-wide supercomputer rankings: *High Performance Linpack (HPL)* [1], *High Performance Conjugate Gradients (HPCG)* [26], and *Graph 500 Benchmark* [27, 28]. All these workloads will be evaluated in two scenarios and five different topology, routing, and placement combinations:

Capability Evaluations. The decommissioned TSUBAME2 enabled us to execute capability runs sequentially, without inter-job interference, to determine the idealized achievable performance on the given topology. We start from a single HyperX switch, i.e., seven nodes (or 4 nodes for pow2) and weak scale the benchmarks up to 672 nodes (or 512, respectively), by doubling the node count in each step. However, we impose a 15 min execution time limit per run.

Capacity Evaluations. Since a multi-job execution model is more realistic [29] — where concurrent jobs compete for resources and inter-job interference degrades message latency [30] — we select 14 of our benchmarks and each gets a dedicated node allocation (32 or 56 nodes, respectively) assigned. Using 664 compute nodes (or 98.8%), we submit all 14 jobs (configured for infinite iterations) simultaneously and let them execute for 3 h to compare the number of runs across our five routing and placement configurations.

Routing and Placement. Both routing and MPI rank placement can positively/negatively affect applications' communication. Hence, we test the above for five combinations:

1) Fat-Tree / ftree routing [31] / linear placement;
2) Fat-Tree / SSSP routing [32] / clustered placement;
3) HyperX / DFSSSP routing [33] / linear placement;
4) HyperX / DFSSSP routing / random placement; and
5) HyperX / PARX routing / clustered placement,

and collect performance data, e.g., latency or solver runtime.

5. Evaluation

All experiments are done according to the conditions listed in Section 4. Hence, apart from replacing faulty nodes, the only difference between trials is the fabric and placement.

Figure 3 shows a sample of the collected data. The results for the ring-based Allreduce implemented by Baidu (cf. Fig. 3a) reveals a noteworthy issue with ftree routing. However, the Fat-Tree itself is not responsible for the performance degradation, since SSSP routing on the Fat-Tree performs equally well as our HyperX. On average, the novel PARX routing appears to yield the lowest performance for MPI micro-benchmarks, such as Allreduce. Analyzing the data of other MPI benchmarks, not shown here due to space constraints, indicates that the severe performance loss induced by PARX stems from the fact that we needed to switch from Open MPI's default `ob1` PML to the (apparently less tuned) `bfo` PML, as explained in Section 3.

In contrast, PARX and our HyperX network are not only able to match the performance of the Fat-Tree, but are also able to outperform it for other traffic patterns and workloads. For example, for the 1 MiB payloads used by Netgauge's eBB (cf. Fig. 3c), where the software overhead of `bfo` diminishes, we observe the advantage resulting from the non-minimal routing. Especially for dense rank allocation on subpartitions of the HyperX, this mitigation technique performs nearly as well as the randomized placement approach, and PARX is able to outperform the Fat-Tree / ftree combination by 2%–6% for the mid-range of the nodes counts.

For realistic HPC workloads, HyperX is on par with the Fat-Tree, depending on routing and placement combination. But most conspicuously, our HyperX together with PARX routing is the only option to consistently execute the SWFFT benchmark at a scale of 512 compute nodes. Figure 3b shows that all 10 executions terminated in less than 233 s. All other tested combinations failed for the 512-node SWFFT, except one valid 869 s run using HyperX / DFSSSP / random.

6. Conclusion

The high-speed interconnect, used to connect the compute nodes in modern HPC systems, has a major impact on achieving scalability and throughput for our scientific workloads. To explore alternative topology designs, we constructed a large-scale, 672-node system with two separate networks, i.e., a 3-level Fat-Tree and a 12x8 2D HyperX, from the remains of the decommissioned TSUBAME2 supercomputer.

Using a variety of MPI and HPC benchmarks, as well as a set of proxy-application which are usually used during system procurement and which resemble real HPC workloads, we evaluated both network topologies for two usage scenarios. While our HyperX installation only offers roughly half-bisection bandwidth, which translates into a deployment cost reduction compared to our full-bisection Fat-Tree, we see from our fair, 1-to-1 comparison that the HyperX is able to compete with our 18-ary 3-tree. The lack of adaptive routing — otherwise generally required for HyperX topologies — in our dated QDR-based InfiniBand equipment makes this achievement by the HyperX even more astonishing.

To overcome the bottlenecks arising from the lack of AR and from applying minimum-path, static routing to the HyperX topology, we developed and tested two mitigation approaches: (1) our novel PARX routing and (2) an alternative MPI rank placement. Our routing prototype shows potential, but real adaptive routing will further improve the effectiveness of upcoming HyperX installations. Regardless, we hope for future adaptation of our approach of minimal plus non-minimal routing and optimization for traffic demands[‡].

Acknowledgements

We would like to thank Hewlett Packard Enterprise for funding the research. Without the hard labor of our 40 university students and volunteers, this project would have been impossible. Moreover, we would like to thank Fujitsu for donating a few spare switches, and also acknowledge the support by JSPS KAKENHI Grant Number JP19H04119 and JST CREST Grant Number JPMJCR1687.

References

[1] E. Strohmaier *et al.*, "TOP500," Nov. 2018. URL: http://www.top500.org/

[2] J. Kim *et al.*, "Technology-Driven, Highly-Scalable Dragonfly Topology," *ACM SIGARCH Comput. Architecture News*, vol. 36, no. 3, pp. 77–88, Jun. 2008.

[3] A. Shpiner *et al.*, "Dragonfly+: Low Cost Topology for Scaling Datacenters," in *2017 IEEE 3rd International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era (HiPINEB)*, Feb. 2017, pp. 1–8.

[4] M. Besta and T. Hoefler, "Slim Fly: A Cost Effective Low-diameter Network Topology," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '14. Piscataway, NJ, USA: IEEE Press, 2014, pp. 348–359.

[5] J. H. Ahn *et al.*, "HyperX: Topology, Routing, and Packaging of Efficient Large-scale Networks," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, ser. SC '09. New York, NY, USA: ACM, 2009, pp. 41:1–41:11.

[6] D. Abts and J. Kim, *High Performance Datacenter Networks: Architectures, Algorithms, and Opportunities*, ser. Synthesis Lectures on Computer Architecture. Morgan & Claypool Publishers, 2011. URL: http://dx.doi.org/10.2200/S00341ED1V01Y201103CAC014

[7] Exascale Computing Project, "ECP Proxy Apps Suite," 2018. URL: https://proxyapps.exascaleproject.org/ecp-proxy-apps-suite/

[8] F. Petrini and M. Vanneschi, "k-ary n-trees: high performance networks for massively parallel architectures," in *11th International Parallel Processing Symposium*, Apr. 1997, pp. 87–93.

[9] T. Hoefler *et al.*, "Multistage Switches are not Crossbars: Effects of Static Routing in High-Performance Networks," in *Proceedings of 2008 IEEE International Conference on Cluster Computing*. IEEE Comput. Soc., Oct. 2008.

[10] GSIC, Tokyo Institute of Technology, "TSUBAME2.5 Hardware and Software," Nov. 2013. URL: https://www.gsic.titech.ac.jp/sites/default/files/spec25e1.pdf

[11] InfiniBand® Trade Association, "InfiniBandTM Architecture Specification Volume 1 Release 1.3 (General Specifications)," Mar. 2015.

[12] T. Hoefler and M. Snir, "Generic Topology Mapping Strategies for Large-scale Parallel Architectures," in *Proceedings of the 2011 ACM International Conference on Supercomputing (ICS'11)*. Tucson, AZ: ACM, Jun. 2011.

[13] A. H. Abdel-Gawad *et al.*, "RAHTM: Routing Algorithm Aware Hierarchical Task Mapping," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '14. Piscataway, NJ, USA: IEEE Press, 2014, pp. 325–335.

[14] G. Faanes *et al.*, "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*, ser. SC '12. Los Alamitos, CA, USA: IEEE Comput. Soc. Press, 2012, pp. 103:1–103:9. URL: http://dl.acm.org/citation.cfm?id=2388996.2389136

[15] OpenFabrics Alliance, "OpenSM," Dec. 2018. URL: https://github.com/linux-rdma/opensm

[16] K. Brown *et al.*, "Hardware-Centric Analysis of Network Performance for MPI Applications," in *2015 21th IEEE International Conference on Parallel and Distributed Systems (ICPADS)*. Melbourne, Australia: IEEE Press, Dec. 2015.

[17] J. Domke and T. Hoefler, "Scheduling-Aware Routing for Supercomputers," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '16. Piscataway, NJ, USA: IEEE Press, 2016. URL: http://dl.acm.org/citation.cfm?id=3014904.3014922

[18] E. Gabriel *et al.*, "Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation," in *Proceedings, 11th European PVM/MPI Users' Group Meeting*, Budapest, Hungary, Sep. 2004, pp. 97–104.

[19] B. Klenk and H. Fröning, "An Overview of MPI Characteristics of Exascale Proxy Applications," in *High Performance Computing: 32nd International Conference, ISC High Performance 2017*, ser. ISC '17, Frankfurt, DE, Jun. 2017.

[20] Intel Corporation, "Intel® MPI Benchmarks User Guide," Sep. 2018. URL: https://software.intel.com/en-us/imb-user-guide

[21] T. Hoefler *et al.*, "Netgauge: A Network Performance Measurement Framework," in *High Performance Computing and Communications, Third International Conference, HPCC 2007, Houston, USA, September 26-28, 2007, Proceedings*, vol. 4782. Springer, Sep. 2007, pp. 659–671.

[22] Baidu, Inc., "baidu-allreduce," Feb. 2017. URL: github.com/baidu-research

[23] RIKEN AICS, "Fiber Miniapp Suite," 2015. URL: fiber-miniapp.github.io/

[24] National Energy Research Scientific Computing Center (NERSC), "NERSC-8 / Trinity Benchmarks," Apr. 2016. URL: https://www.nersc.gov/users/computational-systems/cori/nersc-8-procurement/trinity-nersc-8-rfp/nersc-8-trinity-benchmarks/

[25] M. Leininger, "CORAL Benchmark Codes," Jun. 2014. URL: https://asc.llnl.gov/CORAL-benchmarks/

[26] J. Dongarra *et al.*, "HPCG Benchmark: a New Metric for Ranking High Performance Computing Systems," University of Tennessee, Tech. Rep. ut-eecs-15-736, Jan. 2015. URL: https://library.eecs.utk.edu/pub/594

[27] R. C. Murphy *et al.*, "Introducing the Graph 500," in *Proceedings of the Cray User's Group Meeting (CUG)*, May 2010, p. 5.

[28] K. Ueno *et al.*, "Extreme scale breadth-first search on supercomputers," in *2016 IEEE International Conference on Big Data (Big Data)*, Dec. 2016.

[29] G. P. Rodrigo Álvarez *et al.*, "HPC System Lifetime Story: Workload Characterization and Evolutionary Analyses on NERSC Systems," in *Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC '15. New York, NY, USA: ACM, 2015, pp. 57–60, event-place: Portland, Oregon, USA.

[30] N. Jain *et al.*, "Partitioning Low-Diameter Networks to Eliminate Inter-Job Interference," in *2017 IEEE International Parallel and Distributed Processing Symposium, IPDPS 2017, Orlando, FL, USA, May 29 - June 2, 2017*, ser. IPDPS '17. IEEE Comput. Soc., May 2017, pp. 439–448.

[31] E. Zahavi, "D-Mod-K Routing Providing Non-Blocking Traffic for Shift Permutations on Real Life Fat Trees," Tech. Rep., Aug. 2010. URL: https://webee.technion.ac.il/publication-link/index/id/574

[32] T. Hoefler *et al.*, "Optimized Routing for Large-Scale InfiniBand Networks," in *17th Annual IEEE Symposium on High Performance Interconnects (HOTI 2009)*, Aug. 2009.

[33] J. Domke *et al.*, "Deadlock-Free Oblivious Routing for Arbitrary Topologies," in *Proceedings of the 25th IEEE International Parallel & Distributed Processing Symposium (IPDPS)*. Washington, DC, USA: IEEE Comput. Soc., May 2011.

[‡] Our evaluation toolchain, incl. PARX routing and collected data, is readily available for download: **https://gitlab.com/domke/t2hx**, giving other network researchers the option to reuse/adapt our routing approach and to perform similar studies.