Testing the Limits of Tapered Fat Tree Networks

Philip Taffet
Dept. of Computer Science
Rice University
Houston, TX
ptaffet@rice.edu

Sanil Rao
Dept. of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA
sanilrao@cmu.edu

Edgar A. León and Ian Karlin
Livermore Computing
Lawrence Livermore National Laboratory
Livermore, CA
{leon,karlin1}@llnl.gov

***Abstract*—HPC system procurement with a fixed budget is an optimization problem with many trade-offs. In particular, the choice of an interconnection network for a system is a major choice, since communication performance is important to overall application performance and the network makes up a substantial fraction of a supercomputer's overall price. It is necessary to understand how sensitive representative jobs are to various aspects of network performance to procure the right network. Unlike previous studies, which used mostly communication-only motifs or simulation, this work employs a real system and measures the performance of representative applications under controlled environments. We vary background congestion, job mapping, and job placement with different levels of network tapering on a fat tree. Overall, we find that a 2:1 tapered fat tree provides sufficiently robust communication performance for a representative mix of applications while generating meaningful cost savings relative to a full bisection bandwidth fat tree. Furthermore, our results advise against further tapering, as the resulting performance degradation would exceed cost savings. However, application-specific mappings and topology-aware schedulers may reduce global bandwidth needs, providing room for additional network tapering.**

***Index Terms*—Communication performance, system procurement, networking, application benchmarking, performance measurement, high performance computing.**

I. Introduction

Buying a new machine is an optimization problem. Inputs into this optimization problem include target applications, the scale at which applications are expected to run, the cost of components, and the performance impact of various trade-offs. The system interconnect is typically among the most significant costs in a machine procurement, sometimes as much as 25% of the cost of a high-performance computing (HPC) system. Because of cost, both empirical and simulation studies [1]–[3] have explored various network configurations to determine the optimal configuration in a fixed-budget procurement. Using a less expensive network enables reinvestment of the savings into other parts of the system, e.g. additional nodes, more RAM, or faster CPUs, ultimately resulting in a faster system. However, a network must be chosen carefully because networks that do not provide sufficient bandwidth to meet application demands severely degrade performance [4].

While there are general rules of thumb for network design, the optimal design depends on expected workload and node architecture [5]. Many forward-looking studies attempt to determine the optimal system balance for a future procurement, but few look backwards to evaluate if the correct decision was made. This work takes a view back at previous work to evaluate the decision made to use a 2:1 tapered network [1] for Quartz, a system at Lawrence Livermore National Laboratory (LLNL). We use three applications from this study, including the two that were most impacted by network tapering and one of the most latency sensitive applications. Building on our work previously presented in poster form at SC17 and SC18 [6], [7], we add controlled congestion to study application performance under adverse traffic conditions. Our work makes the following contributions:

- We affirm the decision to procure a 2:1 tapered fat tree [1] and show that the network on the production machine delivers good application throughput even with adversarial traffic.
- Additional tapering, e.g. 4:1, is likely unprofitable for application throughput without further mitigation strategies.
- Application mapping strategies can enable further cost-effective network tapering in fat trees.

The rest of this paper discusses experiments using applications running under adverse conditions on real hardware.

II. Background and Related Work

There are multiple network topology options and ways to run machines with those topologies. The torus topology was formerly common and is still in use on some machines, although it has fallen out of favor to lower radix networks. Torus networks can map jobs to the topology well [8]; however, they can suffer from inter-job interference [4]. One approach to mitigate inter-job interference used in BlueGene systems is partitioning the network so that each allocation has an electrically isolated partition [9]. This leads to very consistent performance, but at the cost of idling nodes more frequently.

The dragonfly topology is another popular topology for large scale systems due to an optimized design that minimizes the need for expensive long optical cables [10]. However, they depend on good adaptive routing to minimize congestion and system noise, and many production implementations have not been good enough [11].

The third commonly deployed topology is a fat tree or folded Clos [12]. Fat trees also can suffer from inter-job interference, and some work has suggested isolating jobs to address this [13], but it is often less severe in practice. Fat trees are more expensive than comparable dragonfly networks because they require many optical cables, but the cost of fat trees can be significantly reduced by tapering at the edge switches [1]. As we show in this paper, tapered fat trees without adaptive routing are robust to network noise.

Several studies have evaluated network performance using network simulators [2], [14], [15]. Simulation allows for full control of the entire simulated system, but simulations rely on either communication patterns or motifs [16], which abstract away some nuances of application performance. In particular, communication patterns cannot account for complex interactions and often only a few timesteps of an application can be run. This paper uses empirical studies of network performance rather than simulation. Empirical measurements enable the use of real applications on real hardware, and provide a more accurate view of performance. Furthermore, these measurements take into consideration larger scale issues, such as aggregate throughput, rather than micro-level congestion events needed to design network routing policies. Using real hardware, however, limits the experiments we can perform and the types of measurements we can collect.

III. Experimental Methodology

In this section we describe the methodology we employ for our experiments. This includes the applications we used for tests, the problems we ran, the machines we ran them on, and how those machines were configured.

A. Applications

In our experiments, we used three applications with different messaging patterns and message sizes. These applications include key communication patterns common in many HPC applications. Our mix of applications includes a range of message sizes and various common collectives. Applications used either MVAPICH2 or Intel MPI implementations.

AMG [17] is a multigrid solver for linear systems used in many applications, including simulations of elastic and plastic deformations. Our chosen test problem solves the linear system resulting from a 27-point nearest neighbor stencil. Communication starts as a 27-point halo exchange and global `AllReduce` operations on the finest grid. As the computation progresses, more communication becomes non-local with a larger number of communication partners on coarser grids and more frequent `AllReduce` operations. Message size varies widely, from 8 bytes to over 100 kB, but latency-bound messages, both point to point and `AllReduce`, tend to dominate its communication.

UMT [18] is a deterministic (Sn) radiation transport mini-app. It exchanges messages to neighbors in a 3D wavefront pattern across the faces in its unstructured grid, resulting in a 7 point stencil pattern.

pF3D [19] is a multi-physics code that simulates laser-plasma interaction experiments at the National Ignition Facility. pF3D is used to understand the effects of backscatter light, a side-effect of the laser. The code models a 3D domain in which the dominant communication pattern, taking up to 30% of the runtime in some cases [19], is FFTs computed across 2D slabs. For our test problem, 144 MPI ranks were assigned to each slab, and the FFTs were 16 kB per rank.

TABLE I
Overview of applications used in experiments. The last three columns refer to the large problem size.

Application	% Time Spent in MPI[1]	Average Message Size[2]	Message Count
AMG	26%	4.8 kB	404 M
UMT	45%	1.6 MB	1.2 M
pF3D	25%	16 kB	160 M

B. System Overview

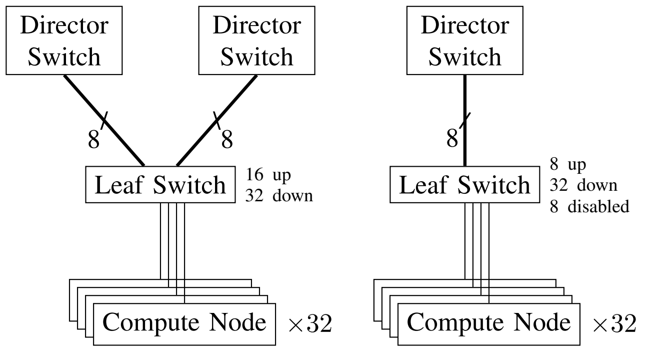
Quartz [20], a petascale system at LLNL, was used for all experiments in this paper. Quartz supports a large mix of unclassified applications and job sizes [1]. Quartz consists of 2,688 nodes with dual 18-core Broadwell CPUs connected by a 3-level 100 Gbps Omni-path [21] fat tree network with a 2:1 taper, i.e., as illustrated in Figure 1a, each leaf switch has 32 links to compute nodes and 16 links to second level switches. Each second-level switch connects to eight leaf switches, grouping 256 nodes together in a “pod”. Quartz’s network is statically routed using the Ftree routing algorithm [22]. One node from each leaf switch is reserved for system support tasks (e.g. login or gateway nodes), leaving 31 user-accessible nodes per leaf switch and 248 per pod. Due to difficulty tuning Omni-path QoS features [21] during initial system installation, Quartz normally runs with these features disabled; we did not re-enable them for our experiments. However, to eliminate the impact of interference from other users running applications during our experiments, we reserved dedicated time on the system.

We also ran some experiments with the network configured as a 4:1 taper instead of the default 2:1 taper. In this configuration, shown in Figure 1b, each leaf switch had 32 links to compute nodes and 8 links to second level switches; the other 8 ports were disabled. If we were to deploy a similar system with a 4:1 taper we would expect a 3-4% full system cost reduction. The reduction is due to a 50% reduction in optical cables and director switches, along with about a 15% reduction in top of rack switches.

C. Problem Selection

We ran each application with a small problem and a large problem. Small problems ran on 30 nodes (28 for pF3D, which requires a multiple of 4 nodes)–approximately one leaf switch worth of nodes–while large problems ran on 224 nodes–approximately one pod worth of nodes. All problems were run with “MPI everywhere” and used one MPI rank per core, i.e., 36 ranks per node, meaning that small problems used approximately 1,000 MPI ranks and large problems used approximately 8,000 MPI ranks. Many applications on Quartz run this way.

[1]Includes time spent due to load imbalance and system noise
[2]Only includes messasges part of primary communication pattern



(a) 2:1 taper (b) 4:1 taper

Fig. 1. Quartz's network with the two configurations used. Each director switch is composed of two levels of internal switches which are not illustrated.

TABLE II
OVERVIEW OF LOCALITY EXPERIMENTS. APPLICATIONS EXECUTED SMALL PROBLEMS EXCEPT WHERE INDICATED.

Name (Abbrev.)	Nodes per leaf sw.	# of leaf sw.	Pattern
Control (Cs)	30	1	Linear
Split 2 (S2s)	15	2	Linear
Split 3 (S3s)	10	3	Linear
Split 6 (S6s)	5	6	Linear
Random (RDs)	Variable	Variable	Random
Random, large (RDl)	Variable	Variable	Random

AMG and UMT report application-specific figures of merit that we use for measuring performance. Since pF3D is an application instead of a benchmark, we compute our own figure of merit for pF3D. Specifically, we take a constant multiple of reciprocal time, where time is measured by the application as CPU time and messaging time but not time spent performing file I/O operations. Due to a configuration error, for pF3D experiments with large problems, FOM calculations only take into account the first full timestep which consists of 10 light physics sub-cycles and 1 hydro physics sub-cycle. Although this may increase the variance in our measurements, we do not believe it has a significant impact on our results.

D. Locality Experiments

To measure the impact of locality on communication performance, we measured the performance of each application on the small problem on nodes split across 2, 3, and 6 leaf switches, all part of the same pod. During each experiment, applications split across s leaf switches shared those leaf switches with $s-1$ applications instances part of the same experiment. Results for these experiments are compared to a control experiment with all nodes for each application connected to a single leaf switch. Table II summarizes locality experiments. Additionally, we also ran applications on both small and large problems with random allocations, and we ran each small locality experiment block and cyclic task mapping strategies to investigate how task mapping impacts performance and how it interacts with locality.

TABLE III
OVERVIEW OF CONGESTION EXPERIMENTS. NODE COUNT IS PER LEAF SWITCH. APPLICATIONS EXECUTED LARGE PROBLEMS EXCEPT WHERE INDICATED.

	Nodes running		
Name (Abbrev.)	Bully	App	Type of traffic
Control (Cl)	0	28	None
Control, small (Cs)	0	30	None
Light leaf bully (LBl)	10	14	Large messages to adjacent leaf switch
Adjacent leaf bully (ABl)	17	14	Large messages to adjacent leaf switch
Adjacent leaf bully, small (ABs)	15	15	Large messages to adjacent leaf switch
Bisection bully (BBl)	17	14	Large messages crossing network bisection
Incast bully (IBl)	3	28	7:1 incast pattern to a single leaf switch

E. Congestion Experiments

To understand how each application responds to different levels of network congestion we ran multiple experiments alongside different traffic patterns designed to cause congestion. Applications running the small problems were split across two leaf switches while 15 nodes from each leaf switch ran an application designed to cause congesting traffic. For most experiments, applications running the large problems used 14 nodes per leaf switch, leaving 17 nodes per leaf switch free to cause congestion. Table III describes a collection of communication traffic patterns that compete for network bandwidth against a code that is the object of our study. We refer to applications whose sole purpose is to inject communication traffic patterns designed to cause congestion as *bullies* and use them to understand how our applications under study perform when the network is under different types of stress.

Because each leaf switch has 16 up-links, the light bully should consume at most 62.5% of available bandwidth while other bullies can consume all available bandwidth, routing permitting. Unlike other bullies, congesting traffic created by the bisection bully application travels between pods, crossing the bisection of the network. The incast bully pattern had three nodes in each of seven leaf switches sending to the eight leaf switch in the pod, oversubscribing links in that leaf switch. The incast pattern mimics I/O traffic and can cause severe performance degradation due to congestion trees if the network doesn't handle it well [23].

IV. NOTABLE FINDINGS

We ran several trials for each locality and congestion experiment and measured the performance of each run. This section summarizes and interprets the results. To assess the statistical significance of each result, we use a Student's t-test. *Italic orange* values are statistically significant at the $p = 0.05$ level while **bold green** values are also statistically significant at the $p = 0.01$ level. Statistical significance takes into account

TABLE IV
PERFORMANCE DEGRADATION COMPARED TO CONTROL. NEGATIVE VALUES INDICATE EXPERIMENT WAS SLOWER THAN CONTROL.

Experiment	AMG	UMT	pF3D
S2s	-0.11%	0.10%	0.47%
S3s	*-2.14%*	-0.04%	0.01%
S6s	*-2.79%*	-0.37%	0.39%
RDs	-1.48%	-0.09%	0.18%
RDl	**-3.17%**	*-0.47%*	*-0.45%*
LBl	-0.70%	**-0.68%**	0.22%
ABl	**-1.74%**	-0.52%	0.19%
ABs	0.15%	-0.20%	0.40%
BBl	**-6.47%**	**-1.94%**	0.22%
IBl	*-0.74%*	0.01%	0.08%

TABLE V
PERFORMANCE DEGRADATION COMPARED TO CONTROL RUN ON A 4:1 TAPER AND COMPARED TO THE SAME EXPERIMENT ON A 2:1 TAPER.

Experi-ment	AMG		UMT	
	vs. 4:1 control	vs. 2:1 experiment	vs. 4:1 control	vs. 2:1 experiment
Cl	–	**-1.41%**	–	**-0.65%**
RDl	**-7.72%**	-0.19%	**-1.54%**	*-0.44%*
LBl	**-3.15%**	**-3.84%**	*-0.83%*	*-0.81%*
ABl	**-7.72%**	**-7.42%**	**-1.54%**	**-1.67%**
BBl	**-9.29%**	**-4.37%**	**-3.88%**	**-2.61%**
IBl	*-0.62%*	**-1.29%**	0.00%	**-0.66%**

the magnitude of the difference, run-to-run variance, and the number of successful runs for each experiment.

A. Robustness of 2:1 Tapered Fat Trees

The strongest finding from our experiments was that the impact of decreasing locality and adding congestion was smaller than expected. As shown in Table IV, most experiments did not result in statistically significant performance degradation. Furthermore, other than in two cases, the performance impact was less than 3%. Overall, this is strong evidence that a 2:1 tapered fat tree provides enough bandwidth to meet the communication demands of these applications.

This matches other experimental studies with applications [1] but is smaller than results that use communication-only motifs. This is not surprising, since computation essentially dilutes the impact of communication performance. Furthermore, overlapping communication with useful computation, as all three applications do, can hide communication slowdowns. Studies and simulations using communication-only motifs are important and useful, but their results must be understood as an exaggeration of the impact on real applications.

Even with adversarial traffic causing heavy background congestion, the performance impact was almost always below the cost savings from using a tapered network. This suggests that the procurement decision of using a tapered network was correct and, since the cost savings were re-invested in more nodes, actually increased overall system performance.

B. Comparison with 4:1 Taper

To explore further tapering of future machines, we also ran several experiments after disabling one of the director switches, essentially mimicking a system with a 4:1 taper. Table V shows the results for experiments using AMG and UMT[3]. Performance degradations are noticeably higher than with the 2:1 taper in almost every experiment. For example, the light leaf bully (LBl) and adjacent leaf bully (ABl), which both cause congestion on the links in the tapered level, had relatively modest impact on AMG with a 2:1 taper but have severe impact with a 4:1 taper. Furthermore, the threshold for the intensity of background traffic at which significant performance degradations begin to occur decreases substantially with the additional reduction in global bandwidth. While the light leaf bully had a small performance impact with a 2:1 taper, it has a substantial impact on AMG with a 4:1 taper.

To evaluate the impact of using a 4:1 tapered network in a future machine, we consider the performance of each experiment relative to the same experiment in the 2:1 tapered network. Performance with a 4:1 taper is typically about 1–2% worse than with a 2:1 taper, but sometimes much higher. Since switching from a 2:1 taper to a 4:1 taper only saves about 3% of the machine cost, using a 4:1 taper is not clearly beneficial. Although reinvesting the cost savings may result in a net performance increase for some jobs, this suggests a significant number of jobs would experience a net performance degradation. Thus, we do not recommend additional tapering beyond the current 2:1 taper for future systems.

C. Communication-Local Mappings

Another clear observation from Table IV is that the magnitude of performance impact varies by application, with AMG the most affected and pF3D the least affected. Interestingly, there is little correspondence between time spent in MPI (from Table I) and sensitivity to congestion, suggesting that sensitivity is a product of an application's specific communication behavior and how it maps onto the system.

We explored, in particular, the finding that pF3D is essentially unaffected by congestion and discovered it is due to pF3D's logical-to-physical mapping. pF3D performs FFTs using collectives on subcommunicators within each slab comprised of 144 consecutive MPI ranks, which is equivalent to 4 compute nodes. Thus, pF3D's logical-to-physical mapping ensures that traffic from the dominant communication pattern is localized to within each group of 4 nodes. On a relatively compact allocation, as were used in these experiments, each 4 node group was often connected to the same leaf switch. Since links connected to compute nodes are not shared, traffic localized to a single leaf switch is essentially impervious to network congestion from background traffic, making pF3D highly robust against congestion.

If all applications used communication-localizing mappings, the reduction in global bandwidth demands could justify using a 4:1 tapered network. However, allocations of nodes given by

[3]Due to a configuration error, we do not have results for pF3D.

TABLE VI
PERFORMANCE OF CYCLIC COMPARED TO BLOCK. POSITIVE VALUES INDICATE CYCLIC MAPPING WAS FASTER.

Experiment	AMG	UMT	pF3D
Cs	**3.61%**	1.64%	**10.29%**
S2s	**3.58%**	**1.53%**	**9.67%**
S3s	**5.65%**	**1.44%**	8.61%
S6s	**6.24%**	**1.59%**	9.14%
RDs	**4.98%**	**1.23%**	**8.18%**
ABs	**3.16%**	**1.79%**	**10.05%**

batch schedulers in a production environment are typically not as compact as the allocations we used in many experiments, meaning that it may be necessary to use a locality-aware batch scheduler to actually reap the benefits of communication-local mappings or to tolerate a tapered network.

D. *Performance of Shared Memory MPI Transport*

Although not originally a goal of our investigation, we discovered interesting and unexpected results from comparing locality experiments using the block task mapping strategy with the cyclic strategy. Since block mapping ensures that MPI ranks with adjacent rank numbers often run on the same compute node, and we conjectured that many applications' communication patterns have some locality in the MPI rank space, we expected that block mapping would improve performance by keeping more messages node-local. Essentially, some messages that could be transported with shared memory in a block mapping would instead must be transmitted over the network under a cyclic mapping.

Our initial results in Table VI show the cyclic mapping was faster than block in every instance. In fact, the performance difference is often fairly substantial, for example, pF3D's performance on the small problem in the control experiment is over 10% higher with cyclic mapping compared to block.

Upon further investigation, we discovered that although the cyclic strategy did cause more network load, it performed better unless the network was extremely heavily loaded. For certain message sizes, the DMA engine of the 100 Gbps Omni-Path NIC was able to transfer data between two nodes faster than one process could transfer data to another process running on the same node. This problem affected all MPI implementations available on Quartz. Using the kernel sampling feature of HPCToolkit [24], we discovered that message transfers over shared memory were implemented using the `process_vm_readv` system call [25] to copy 1 MB buffers between address spaces with a single copy. We wrote a simple benchmark application that called `process_vm_readv` directly, but were not able to drive more than 8.3 GB/s of throughput compared to 13.1 GB/s for `memcpy` and 10.5 GB/s NIC throughput. This suggests that software optimization of `process_vm_readv` may be possible. Figure 2 shows the performance using `MPI_Isend` for various message sizes with the 2018.0 version of Intel's MPI implementation. Results for other MPI distributions and on some other systems at LLNL exhibited similar behavior.

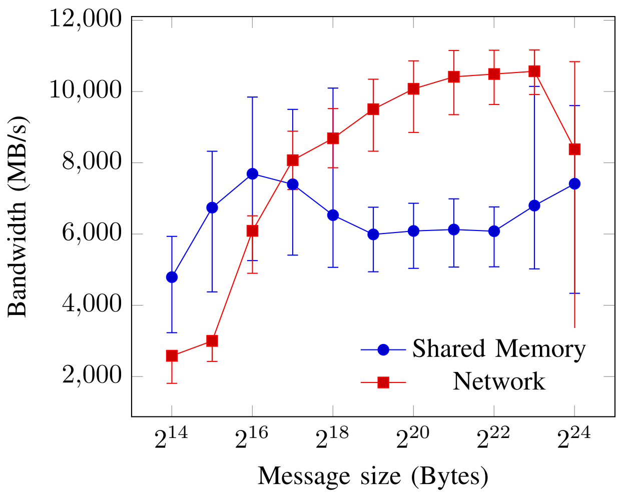


Fig. 2. Mean achieved throughput with OSU bidirectional point-to-point bandwidth benchmark [26] when two MPI ranks run on the same compute node and transfer messages via shared memory (blue) or on different compute nodes and transfer messages through the network (red). Error bars indicate the minimum and maximum values observed over approximately 100 trials.

Unfortunately, switching to cyclic has significant drawbacks as well. Since network bandwidth is shared among all MPI ranks on a compute node, bandwidth per rank decreases sharply as more MPI ranks per node attempt to send inter-node messages simultaneously. Furthermore, since the cyclic mapping tends to increase traffic on the network, it also makes applications more sensitive to network congestion and locality. For example, pF3D's mapping with cyclic no longer employs local communication; the S3s experiment ran **1.52%** slower than control when both S3s and control used cyclic compared to essentially no change with block.

Our finding that the shared memory MPI transport mechanism performs worse than expected in some cases creates a conflict for developers attempting to optimize their mapping; the fastest mappings for an application are likely those that demand the most bandwidth from the network. Since this appears to be a software issue, we expect that results with the block mapping are most relevant for procurements. However we also recommend that developers experiment with alternative mappings for current machines, and that future nodes contain a block transfer engine to accelerate intra-node transfers.

V. CONCLUSION AND FUTURE WORK

In this paper, we show that application performance on Quartz's 2:1 tapered fat tree network is not significantly impacted under adversarial traffic conditions, validating the conclusions reached in previous work [1]. Furthermore, our analysis shows that if a 4:1 taper were used, it would result in significant performance degradation and only save about half (3-4% of system cost) the capital costs saved from 1:1 to 2:1. Thus, a 4:1 tapering may not be cost effective.

Independent of tapering, our initial application task mapping and job placement experiments show significant performance advantages. By keeping traffic local, these techniques may reduce the use of global links and improve network latency. Additional experiments and analysis are needed to demonstrate that a diverse set of applications can benefit from this and also to understand the impact on job scheduling and throughput.

In future work, we plan to investigate the combined impact of task mapping and job placement techniques with more aggressive tapering, which may result in additional cost savings. Another approach to mitigate congestion is to use adaptive routing and quality of service mechanisms present in some modern networks to prioritize traffic and take advantage of underutilized links. Finally, we plan to extend this work to GPU systems. In a GPU system, messages are typically more frequent due to computation acceleration, and larger since one MPI rank often uses the whole GPU. This will require more bandwidth per node, but less bandwidth per unit of compute. The impact of these GPU-driven changes on tapering, however, is likely small as application messaging patterns and job placement are the primary driver of global traffic.

Acknowledgment

We would like to thank John Mellor-Crummey for his valuable feedback. This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. LLNL-CONF-794561.

References

[1] E. A. León, I. Karlin, A. Bhatele, S. H. Langer, C. Chambreau, L. H. Howell, T. D'Hooge, and M. L. Leininger, "Characterizing parallel scientific applications on commodity clusters: An empirical study of a tapered fat-tree," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '16. Piscataway, NJ, USA: IEEE Press, 2016, pp. 78:1–78:12. [Online]. Available: http://dl.acm.org/citation.cfm?id=3014904.3015009

[2] N. Jain, A. Bhatele, S. White, T. Gamblin, and L. V. Kale, "Evaluating HPC networks via simulation of parallel workloads," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '16. Piscataway, NJ, USA: IEEE Press, 2016, pp. 14:1–14:12. [Online]. Available: http://dl.acm.org/citation.cfm?id=3014904.3014923

[3] E. A. León, C. Chambreau, and M. L. Leininger, "What do scientific applications need? An empirical study of multirail network bandwidth," in *International Conference on Advanced Communications and Computation*, ser. INFOCOMP'17. Venice, Italy: IARIA, Jun. 2017.

[4] A. Bhatele, K. Mohror, S. H. Langer, and K. E. Isaacs, "There goes the neighborhood: Performance degradation due to nearby jobs," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, ser. SC '13. New York, NY, USA: ACM, 2013, pp. 41:1–41:12. [Online]. Available: http://doi.acm.org/10.1145/2503210.2503247

[5] C. Zimmer, S. Atchley, R. Pankajakshan, B. E. Smith, I. Karlin, M. Leininger, A. Bertsch, B. S. Ryujin, J. Burmark, A. Walker-Loud, K. M. A., and O. Pearce, "An evaluation of the CORAL interconnects," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, ser. SC '19, Nov. 2010.

[6] P. Taffet, I. Karlin, E. A. Leon, and J. Mellor-Crummey, "Understanding the impact of fat-tree network locality on application performance," in *SC 17*, Denver, CO, Nov. 2017. [Online]. Available: http://sc17.supercomputing.org/SC17%20Archive/src_poster/poster_files/spost153s2-file2.pdf

[7] P. Taffet, S. Rao, and I. Karlin, "Exploring application performance on fat-tree networks in the presence of congestion," in *SC 18*, Dallas, TX, Nov. 2018. [Online]. Available: https://sc18.supercomputing.org/proceedings/tech_poster/poster_files/post250s2-file3.pdf

[8] A. Bhatele, T. Gamblin, S. H. Langer, P.-T. Bremer, E. W. Draeger, B. Hamann, K. E. Isaacs, A. G. Landge, J. A. Levine, V. Pascucci, M. Schulz, and C. H. Still, "Mapping applications with collectives over sub-communicators on torus networks," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, ser. SC '12. Los Alamitos, CA, USA: IEEE Computer Society Press, Nov. 2012, pp. 97:1–97:11. [Online]. Available: http://dl.acm.org/citation.cfm?id=2388996.2389128

[9] D. Chen, N. Eisley, P. Heidelberger, R. Senger, Y. Sugawara, V. Salapura, D. Satterfield, B. Steinmacher-Burow, and J. Parker, "The IBM Blue Gene/Q interconnection network and message unit," 12 2011, pp. 1 – 10.

[10] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Technology-driven, highly-scalable dragonfly topology," in *2008 International Symposium on Computer Architecture*, June 2008, pp. 77–88.

[11] T. Groves, Y. Gu, and N. J. Wright, "Understanding performance variability on the Aries dragonfly network," in *2017 IEEE International Conference on Cluster Computing (CLUSTER)*, Sep. 2017, pp. 809–813.

[12] C. E. Leiserson, "Fat-trees: Universal networks for hardware-efficient supercomputing," *IEEE Transactions on Computers*, vol. C-34, no. 10, pp. 892–901, Oct 1985.

[13] S. D. Pollard, N. Jain, S. Herbein, and A. Bhatele, "Evaluation of an interference-free node allocation policy on fat-tree clusters," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*, ser. SC '18. Piscataway, NJ, USA: IEEE Press, 2018, pp. 26:1–26:13. [Online]. Available: https://doi.org/10.1109/SC.2018.00029

[14] A. Bhatele, N. Jain, M. Mubarak, and T. Gamblin, "Analyzing cost-performance tradeoffs of HPC network designs under different constraints using simulations," in *Proceedings of the 2019 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, ser. SIGSIM-PADS '19. New York, NY, USA: ACM, 2019, pp. 1–12. [Online]. Available: http://doi.acm.org/10.1145/3316480.3325516

[15] N. Jain, A. Bhatele, L. H. Howell, D. Böhme, I. Karlin, E. A. León, M. Mubarak, N. Wolfe, T. Gamblin, and M. L. Leininger, "Predicting the performance impact of different fat-tree configurations," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '17. New York, NY, USA: ACM, 2017, pp. 50:1–50:13. [Online]. Available: http://doi.acm.org/10.1145/3126908.3126967

[16] S. D. Hammond, K. S. Hemmert, M. J. Levenhagen, A. F. Rodrigues, and G. R. Voskuilen, "Ember: Reference communication patterns for exascale." Aug. 2015.

[17] J. Park, M. Smelyanskiy, U. M. Yang, D. Mudigere, and P. Dubey, "High-performance algebraic multigrid solver optimized for multi-core based distributed parallel systems," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '15. New York, NY, USA: ACM, 2015, pp. 54:1–54:12. [Online]. Available: http://doi.acm.org/10.1145/2807591.2807603

[18] L. Howell, "Characterization of UMT2013 performance on advanced architectures," 12 2014.

[19] S. H. Langer, A. Bhatele, and C. H. Still, "pF3D simulations of laser-plasma interactions in National Ignition Facility experiments," *Computing in Science Engineering*, vol. 16, no. 6, pp. 42–50, Nov. 2014.

[20] TOP500. (2018) LLNL CTS-1 Quartz - Tundra Extreme Scale, Xeon E5-2695v4 18C 2.1GHz, Intel Omni-Path. [Online]. Available: https://www.top500.org/system/178971

[21] M. S. Birrittella, M. Debbage, R. Huggahalli, J. Kunz, T. Lovett, T. Rimmer, K. D. Underwood, and R. C. Zak, "Enabling scalable high-performance systems with the Intel Omni-Path architecture," *IEEE Micro*, vol. 36, no. 4, pp. 38–47, Jul. 2016.

[22] E. Zahavi, G. Johnson, D. J. Kerbyson, and M. Lang, "Optimized InfiniBand fat-tree routing for shift all-to-all communication patterns," *Concurrency and Computation: Practice and Experience*, vol. 22, no. 2, pp. 217–231, 2010. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.1527

[23] G. F. Pfister and V. A. Norton, ""Hot spot" contention and combining in multistage interconnection networks," *IEEE Transactions on Computers*, vol. 100, no. 10, pp. 943–948, 1985.

[24] J. Mellor-Crummey, L. Adhianto, M. Fagan, M. Krentel, and N. Tallent, *HPCToolkit User's Manual*, Rice University, May 2019. [Online]. Available: hpctoolkit.org/manual/HPCToolkit-users-manual.pdf

[25] *process_vm_readv(2) Linux Programmer's Manual*, Mar. 2012. [Online]. Available: https://linux.die.net/man/2/process_vm_readv

[26] (2019) MVAPICH :: Benchmarks. [Online]. Available: http://mvapich.cse.ohio-state.edu/benchmarks/