Preliminary Performance Analysis of Multi-rail Fat-tree Networks

Noah Wolfe*, Misbah Mubarak†, Nikhil Jain‡, Jens Domke§, Abhinav Bhatele‡,
Christopher D. Carothers*, Robert B. Ross†

*Department of Computer Science, Rensselaer Polytechnic Institute, Troy, New York
†Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois
‡Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, California
§Institute of Computer Engineering, Technische Universität Dresden, Dresden, Germany

***Abstract*—Among the low-diameter, high-radix networks being deployed in next-generation HPC systems, dual-rail fat-tree networks are a promising approach. Adding additional injection connections (rails) to one or more network planes allows multi-rail fat-tree networks to alleviate communication bottlenecks. These multi-rail networks necessitate new design considerations, such as routing choices, job placements, and scalability of rails. We extend our fat-tree network model in the CODES parallel simulation framework to support multi-rail and multi-plane configurations in addition to different types of static routing, resulting in a powerful research vehicle for fat-tree network analysis. Our detailed packet-level simulations use communication traces from real applications to make performance predictions and to evaluate the impact of single- and multi-rail networks in conjunction with schemes for injection rail selection and intra-plane routing.**

I. Introduction

With next-generation systems shifting toward low-diameter, high-radix interconnection networks such as fat-tree and dragonfly, issues arise with respect to maximizing application performance on these networks. One approach being applied to some upcoming HPC systems, such as Summit [1], is the use of dual-rail fat-tree networks. Summit's new dual-rail fat-tree network will provide an injection bandwidth of 25 GB/s and deliver five times the compute performance of its predecessors [1]. It is known that the system will have large, compute-heavy nodes, but it is unclear how well the 25 GB/s bandwidth of the network will match up with the compute performance of 3,400 dense nodes. This raises additional issues that need to be addressed such as, how can applications efficiently use multiple rails in order to fully utilize the compute capacity.

In this work, we use the CODES simulation framework and the optimistic event scheduling capability of ROSS to conduct efficient yet detailed packet-level simulations [2]. We use CODES to evaluate design choices for multi-rail fat-tree networks by replaying communication traces of applications available via the Design Forward program [3]. Inspired by the Summit configuration, our CODES fat-tree network model leverages new static routing and multi-rail configuration extensions to emulate complete fat-tree systems with multiple injection links per compute node (multi-rail) connecting to multiple network planes. CODES also has the ability to model multiple jobs executing in parallel using different HPC job placement schemes that represent potential full system workloads. In order to mimic the large compute node performance, we map multiple MPI processes to a node. We use our detailed packet-level network model to evaluate the effectiveness of a dual-rail dual-plane network in improving performance by comparing with a corresponding single-rail single-plane network.

II. Background

In this section, we describe the functionality provided by the CODES and ROSS simulation frameworks and the communication patterns of the applications used for the evaluation.

A. ROSS Discrete-event Simulator

Rensselaer Optimistic Simulation System (ROSS) provides the parallel discrete-event simulation platform for the CODES network and storage simulation framework [4]. ROSS supports optimistic event scheduling which enables scalable simulations. ROSS has demonstrated super-linear speedup and is capable of processing 500 billion events/s with over 250 million logical processes (LPs) on 120k nodes of the Blue Gene/Q system at Lawrence Livermore National Laboratory [5].

B. CODES Simulation Framework

CO-Design of multi-layer **E**xascale **S**torage and data-intensive systems (CODES) provides a framework for exploring the design space of HPC interconnects, storage systems and workloads using ROSS. CODES supports high-fidelity packet-level models of several HPC interconnect topologies including dragonfly, slim fly, torus, and fat-tree networks [2], [6], [7], [8]. CODES also supports a variety of network workloads [2] including synthetic traffic injection and application communication traces collected using the DUMPI tracing tool [9]. In this paper, we focus on application communication traces from the Design Forward program [3].

C. Application Traces

Communication traces from the Design Forward program represent a variety of communication patterns and intensities, and application scales. In the following, we provide details of these applications and communication traces, which are the workloads for the fat-tree simulations in Section VI.

AMG: The AMG benchmark is a parallel algebraic multigrid solver developed at LLNL [10]. AMG decomposes the default unstructured domain into 3D chunks and solves a linear system arising from it. It sends a significant number of small messages and MPI operations account for 39.66% of the runtime for the trace with 13,824 MPI processes.

Crystal Router: Crystal Router represents the many-to-many communication pattern of the highly scalable Nek5000 spectral element code developed at ANL [11]. The MPI processes in Crystal Router perform large data transfers in the form of an n-dimensional hypercube. The trace for 1,000 MPI processes shows an overall communication time of 68.5% of the application's runtime.

Geometric Multigrid: The Geometric Multigrid miniapp implements a single production cycle of the linear solver used in BoxLib [12]. Multigrid processes communicate along the diagonals, resulting in many-to-many communication. The trace used in this paper has 10,648 MPI processes with roughly 5% of its runtime spent in MPI communication.

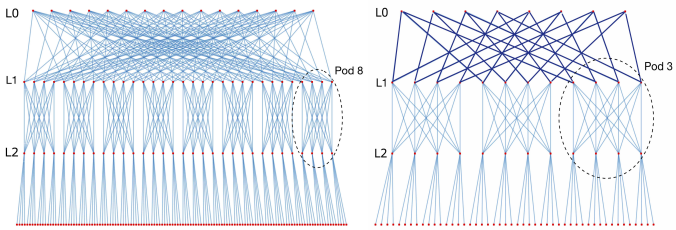
In each case, the simulations replay the traces using the communication time, not accounting for computation time.

III. Fat-tree Network Model

Full bisection fat-tree networks are suitable for the Summit supercomputer with its computationally dense CPU-GPU nodes. This section first describes the full fat-tree network model, followed by its pruned configuration, and finally we describe the multi-rail topology.

A. Full Bisection Fat-tree Configuration

The fat-tree network topology is composed of typically two or three switch levels [13]. For a given switch radix, k, each switch has $k/2$ links to switches in upper levels and $k/2$ links to switches or compute nodes in lower levels. The structure of the 3-level fat-tree breaks down into k *pods*. Each pod contains $k/2$ switches in the first and second levels, labeled L1 and L2, respectively (Figure 1a). L1 and L2 switches within each pod form a complete bipartite graph. Each L2 switch has $k/2$ connections down to compute nodes resulting in a total of $k \times k/2 \times k/2 = k^3/4$ compute nodes.



(a) Full fat-tree (b) Pruned fat-tree

Fig. 1: Example configurations of (a) full and (b) pruned 3-level fat-trees using 8-port switches yielding 128 and 48 compute nodes respectively (darker colored lines between L1 and L0 indicate a bundle of two links).

B. Pruned Fat-tree Configuration (Summit Approximation)

Modeling a fat-tree network that matches the future Summit system requires modifications to the full bisection fat-tree. Currently, Mellanox commodity EDR switches are available with up to $k = 36$ ports. Using 36-port switches and the standard 3-level fat-tree configuration [13] results in $36 \times (36/2)^2$ terminals, i.e., 11,664 compute nodes which is many more than is necessary for the 3,400-node Summit system.

An alternative design option, the pruned fat-tree, starts with the full 11,664-node 3-level fat-tree and then prunes/removes excess pods within the network (example shown in Figure 1b). Full connectivity between pods is maintained by adjusting L1 and L0 connections as needed. This process continues until the desired node count for the Summit system is reached. Using the presumed 36-port switch radix yields 11 pods, which are enough to construct a 3,564 node pruned fat-tree able to approximate the Summit system.

C. Multi-rail Topology (Summit Approximation)

Multi-rail networks are deployed to handle the increased injection load of dense compute nodes, as used in Summit, and support additional network planes. We assume that each network plane has the same topology, and all rails and their corresponding planes are independent of one another.

IV. Fat-tree Routing Algorithms

In this section, we describe our schemes used for traffic injection rail selection and intra-plane message routing.

A. Routing within a Fat-tree Network

A key influencing factor for application performance on a fat-tree topology is the routing algorithm, which determines the physical path traversed by packets in the network. In this paper, we focus on two routing approaches that we have integrated into our fat-tree network model.

1) Static Routing: A state-of-the-art routing algorithm for InfiniBand-based fat-tree topologies is the flow-oblivious and destination-based fat-tree routing [14]. We adapt the approach of the fail-in-place simulation framework [15], allowing us to make use of the optimized fat-tree routing available as part of the InfiniBand (IB) subnet manager, called OpenSM. We make use of the topology loader/generator and routing engine in [15], and load the extracted forwarding tables (LFT) back into our fat-tree model to accurately simulate the application traffic routed via the static fat-tree routing algorithm.

2) Adaptive Routing: This routing scheme aims to balance traffic on different links by making locally optimal decisions for forwarding packets in a switch, similarly to the minimal adaptive routing of the Connection Machine CM-5 [16]. For each port (or virtual channel) on a switch, a token-count is maintained to estimate the availability of a port. The initial value of token-count is set to the full capacity of the virtual channel buffer. The token-count is decremented when a packet is sent on a port and incremented when a credit is received from the receiver. When a packet arrives at a switch, we identify all ports on which the packet can be forwarded

assuming shortest path routing. The port with the maximum token-count is selected to enqueue the packet.

B. Multi-rail Injection

An additional scheduling layer directs packets by deciding which rail to transmit the packet on. Once the packet is injected, the intra-plane routing algorithm routes the packet to its destination. The following policies have been implemented:

1) Random Injection: The random approach offers a uniform distribution of traffic over all network rails regardless of traffic load and network congestion.

2) Adaptive Injection: The adaptive approach samples the occupancy of ports connecting the NIC to different rails and selects the one with the lowest number of packets.

V. Verification

The fat-tree network model used in this paper is an extension of previously validated work [17]. In this verification study, we ensure that the multi-rail fat-tree configuration observes expected full-bisection bandwidth under a synthetic bisection workload. Using static routing and bisection-pairing ensures that each pair of compute nodes has its own distinct path in the fat-tree network to transfer messages.

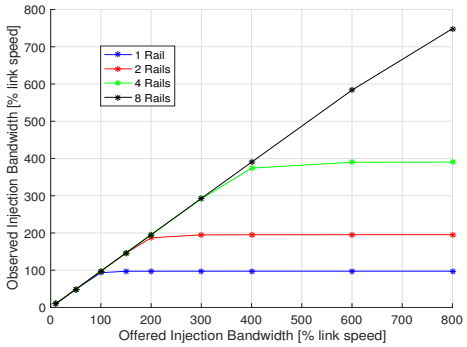


Fig. 2: Multi-rail verification using a synthetic bisection-pairing workload. The axes show the offered and observed injection bandwidth in percent of the link speed of EDR InfiniBand (12.5 GB/s).

All nodes should have an observed throughput performance equal to the injection load up until 100% link bandwidth, at which point the single-rail network will be fully saturated. At an injection rate beyond 100%, each additional rail in the network should provide a corresponding increase in observed bandwidth above the 100% injection load. These expectations are verified in Figure 2, confirming expected multi-rail network performance under ideal workload conditions.

VI. Evaluation

We focus on characterizing general trends, as well as quantifying network performance, for the dual-rail fat-tree Summit network in comparison with a comparable single-rail network for the three application traces.

A. Simulation Configuration

All simulations execute multiple traces running concurrently on the 3,564-node pruned fat-tree system using one AMG, one Crystal Router, and one Multigrid trace. The traces have 13,824, 1,000, and 10,648 MPI processes, respectively, and each compute node hosts eight MPI processes. All switches are simulated with a 90 ns switch traversal delay and one virtual channel with a buffer space of 64 KB per port. Flow control is performed with the use of 8 -byte credits and the network packet size is 4 KB. All links in the single-rail network are set to the EDR bandwidth of 12.5 GB/s and in the dual-rail network to FDR bandwidth of 7 GB/s.

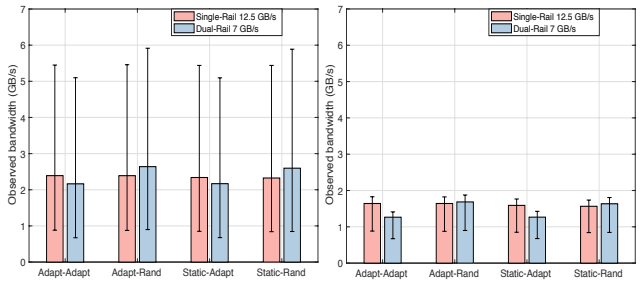
The results (Figure 3) focus on observed bandwidth which measures the rate at which each node is able to inject data into the network. Each compute node calculates the observed bandwidth by dividing the total amount of bytes transferred by the total time spent transferring the data. The sub-figure labeled "System Aggregate" represents the data aggregated over all active compute nodes in the 3,564-node system. The remaining application-specific sub-figures present data from the same parallel multi-job run extracted with respect to the corresponding application trace. The height of each bar represents the average bandwidth among all compute nodes, while the upper and lower limits on each bar indicate the maximum and minimum compute node bandwidths, respectively. Further, each cluster of bars has hyphenated x-axis labels indicating intra-plane routing and injection rail selection policies. For example, a "Static-Adapt" label indicates the use of static intra-plane routing with adaptive injection rail selection.

B. Performance Results

We sought out to determine the ability of the dual-rail network to match or even exceed network performance of a similar single-rail configuration. The dual-rail configuration has twice the number of switches and links with the slower FDR link speed of 7 GB/s. The single-rail configuration has half as many switches and links with the faster EDR link speed of 12.5 GB/s. Figure 3 presents the observed bandwidth performance of the single-rail and dual-rail network configurations under all combinations of intra-plane routing and injection rail selection policies.

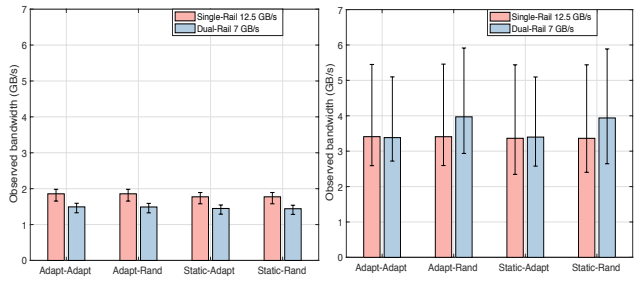
Overall, the observed bandwidth performance is almost identical for both configurations. A small difference in performance between the two networks can be seen depending on the application trace and dual-rail injection policy. For example, Crystal Router achieves better performance with single-rail for all rail selection and routing schemes with up to 25% improvement over dual-rail. Multigrid observes equal or better performance on the dual-rail configuration with up to 17% improvement compared to single-rail.

Finally, AMG sees split preference to single-rail and dual-rail depending on the dual-rail injection policy. Adaptive rail injection achieves up to 26% lower bandwidth on the dual-rail network while random injection allows for up to 4% improved bandwidth. The behavior results from the localized port selection which is oblivious to the network congestion.



(a) System Aggregate

(b) AMG



(c) Crystal Router

(d) Multigrid

Fig. 3: Network performance comparison of a single-rail 12.5 GB/s network and a dual-rail 7 GB/s network.

Hence, when injection bandwidth for an application trace falls below link bandwidth, which is the pattern in the bursty AMG and Multigrid workloads, output buffers on the compute nodes are emptied by the time new packets are generated. In this case, packets are repeatedly issued on the first rail and can lead to a slight network load imbalance and congestion within the network which compounds over time. In contrast, random rail injection is impervious to injection loads and balances communication among available rails at all times, resulting in less congestion and slightly higher bandwidth performance under bursty communication patterns.

VII. Conclusion

Understanding HPC interconnects and performance factors influencing them is vital to the best utilization of future HPC systems. In this paper, we extended the CODES fat-tree model to support multi-rail and multi-plane configurations, and studied the performance of a dual-rail fat-tree network modeling the proposed Summit supercomputer. We presented simulation results comparing dual-rail vs. single-rail performance.

Our comparison study shows that observed network bandwidth performance is highly dependent on the communication pattern. While the Crystal Router workload performs best on the single-rail network, Multigrid workload observes up to 17% performance improvement on the dual-rail, and AMG performs similarly on both network configurations. We have found that applications, such as Multigrid, which have bursty communication workloads that inject large quantities of small messages into the network at high rates, see benefit from the additional network rail.

Acknowledgment

This work is supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computer Research (ASCR) under contract DE-AC02-06CH11357. 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-713054).

References

[1] Oak Ridge National Laboratory, "*Summit, Oak Ridge's next High Performance Supercomputer*," https://www.olcf.ornl.gov/summit/.

[2] M. Mubarak, C. D. Carothers, R. B. Ross, and P. Carns, "Enabling Parallel Simulation of Large-Scale HPC Network Systems," in *IEEE Transactions on Parallel and Distributed Systems*. IEEE, 2016.

[3] Department of Energy, "*Design Forward - Exascale Initiative*," (Accessed on: Dec. 31, 2014). [Online]. Available: http://www.exascaleinitiative.org/design-forward

[4] C. D. Carothers, D. Bauer, and S. Pearce, "ROSS: A high-performance, low-memory, modular Time Warp system," *J. of Parallel and Distributed Comput.*, vol. 62, no. 11, pp. 1648–1669, Nov. 2002.

[5] P. D. Barnes, C. D. Carothers, D. R. Jefferson, and J. M. LaPre, "Warp speed: executing time warp on 1,966,080 cores," in *Proc. of the 2013 ACM SIGSIM Conf. on Principles of Advanced Discrete Simulation (PADS)*, May 2013, pp. 327–336.

[6] N. Wolfe, C. D. Carothers, M. Mubarak, R. Ross, and P. Carns, "Modeling a million-node slim fly network using parallel discrete-event simulation," in *Proceedings of the 2016 annual ACM Conference on SIGSIM Principles of Advanced Discrete Simulation*. ACM, 2016, pp. 189–199.

[7] M. Mubarak, C. D. Carothers, R. B. Ross, and P. Carns, "A case study in using massively parallel simulation for extreme-scale torus network codesign," in *Proc. of the 2nd ACM SIGSIM/PADS Conf. on Principles of Advanced Discrete Simulation*, 2014, pp. 27–38.

[8] ——, "Using massively parallel simulation for MPI collective communication modeling in extreme-scale networks," in *Proc. of the 2014 Winter Simulation Conf.*, 2014, pp. 3107–3118.

[9] Sandia National Labs, "*SST DUMPI trace library*," (Accessed on: Apr. 3, 2015). [Online]. Available: http://sst.sandia.gov/using_dumpi.html

[10] *Co-design at Lawrence Livermore National Laboratory*, "*Algebraic Multigrid Solver (AMG)*," (Accessed on: Apr. 19, 2015). [Online]. Available: https://codesign.llnl.gov/amg2013.php

[11] J. Shin, M. W. Hall, J. Chame, C. Chen, P. F. Fischer, and P. D. Hovland, "Speeding up nek5000 with autotuning and specialization," in *Proceedings of the 24th ACM International Conference for Supercomputing*. ACM, 2010, pp. 253–262.

[12] Department of Energy, "*AMR Box Lib*." [Online]. Available: https://ccse.lbl.gov/BoxLib/

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

[14] E. Zahavi, G. Johnson, D. J. Kerbyson, and M. Lang, "Optimized InfiniBand fat-tree routing for shift all-to-all communication patterns," *Concurr. Comput. : Pract. Exper.*, vol. 22, no. 2, pp. 217–231, Feb. 2010. [Online]. Available: http://dx.doi.org/10.1002/cpe.v22:2

[15] J. Domke, T. Hoefler, and S. Matsuoka, "Fail-in-place Network Design: Interaction Between Topology, Routing Algorithm and Failures," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '14. Piscataway, NJ, USA: IEEE Press, 2014, pp. 597–608. [Online]. Available: http://dx.doi.org/10.1109/SC.2014.54

[16] C. E. Leiserson, Z. S. Abuhamdeh, D. C. Douglas, C. R. Feynman, M. N. Ganmukhi, J. V. Hill, D. Hillis, B. C. Kuszmaul, M. A. St. Pierre, D. S. Wells, M. C. Wong, S.-W. Yang, and R. Zak, "The network architecture of the Connection Machine CM-5," in *SPAA '92: Proceedings of the 4th annual ACM Symposium on Parallel Algorithms and Architectures*. New York, NY, USA: ACM, 1992, pp. 272–285.

[17] 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.