Exploring Application Performance on Fat-Tree Networks in the Presence of Congestion

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

Sanil Rao
Department of Computer Science
University of Virginia
Charlottesville, VA
sr7cb@virginia.edu

Ian Karlin
Livermore Computing
Lawrence Livermore National Lab
Livermore, CA
karlin1@llnl.gov

***Abstract*—Network congestion, which occurs when multiple applications simultaneously use shared links in cluster network, can cause poor communication performance, decreasing the performance and scalability of parallel applications. Many studies are performed while clusters also run other production workloads, which makes it harder for them to isolate causes and their effects. To look at congestion in a more controlled setting we used dedicated access time on an HPC cluster and measured the performance of three HPC applications with different communication patterns run with varying amounts and types of background traffic. This enables us to assess the relative sensitivity of the applications to congestion caused by different traffic patterns. Our tests show that the applications were not significantly impacted by even the most aggressive neighboring patterns, with all the performance degradation being 7% or less, pointing to the resiliency of the fat-tree topology.**

I. Introduction

Communication performance is an important part of overall application performance, especially at large scale. Past studies showed that applications running on torus [1] and dragonfly [2] networks exhibit performance variability due to network congestion caused by nearby jobs. Fat-tree networks, although common in current production HPC systems, have received less attention. In this poster, we analyze the results of running several HPC applications with different communication patterns on a fat-tree network with different levels and types of background traffic. We work to understand how much congestion is required to significantly impact application performance, and how this varies from application to application. This study is an extension of work from last year [3] that focused on measuring applications' sensitivity to different network placements. In this study we intentionally create significantly more congestion and show that application performance degradation never exceeds 7%.

II. Application Overview

We used three applications with different messaging patterns and message sizes to represent some of the key communication patterns common in many HPC applications. **AMG** is a multigrid solver for linear systems used in many applications, including simulations of elastic and plastic deformations. The chosen test problem used a 27-point nearest neighbor stencil. Communication starts local, but becomes non-local on coarser grids. Small latency bound messages dominate communication, and the app's average message size is 4 kB. **UMT** is a deterministic (S_n) radiation transport mini-app. It exchanges messages to neighbors in a 3D pattern across its faces, resulting in a 7 point stencil pattern with an average message of 53 kB. **pF3D** is a multi-physics code that simulates laser-plasma interaction experiments at the National Ignition Facility. It models a 3D domain in which the dominant communication patterns are FFTs computed across 2D slabs, with 144 MPI ranks assigned to each slab.

All of our tests were run on the Quartz cluster at LLNL. Quartz is a 3-level fat-tree built with 100 Gbps Omni-Path switches and employing FTree routing to connect its 2,688 nodes together. Each node contains two 18-core 2.1 GHz Broadwell processors. The network has a 2 to 1 tapper at the leaf switches and 32 nodes per leaf switch. Each leaf switch has 31 user nodes and 1 system node. The second level switches connect 256 nodes together with two links to each leaf switch. We will refer to a 256 node group as a pod.

To understand how Quartz responds to different levels of network congestion we ran multiple experiments alongside different traffic patterns designed to cause congestion. Each application used 224 nodes and 8064 MPI tasks per application. We left 24 nodes, 3 per switch, free so we could run run an additional program that causes congestion. We refer to this application as a bully and use it to understand how the applications performs when the network is under different types of stress.

We first measured a baseline of all of our applications running within a pod using 28 nodes per switch to understand their performance in a best case scenario. This experiment had no background traffic. In later bully runs, except the incast bully, applications were spread over two pods using 14 nodes per leaf switch. Our first two experiments involved using some or all of the remaining nodes to send messages back and forth between pairs of nodes on neighboring leaf switches. We ran two flavors of this our "light" bully had 10 communicating pairs and our "full" bully had 17 communicating pairs. With 10 communicating pairs the bully is consuming 62.5% of the

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

TABLE I

MEAN PERFORMANCE DIFFERENCE COMPARED TO CONTROL FOR EACH APPLICATION AND TYPE OF CONGESTING TRAFFIC. NEGATIVE VALUES INDICATE THAT THE APPLICATION PERFORMED WORSE THAN CONTROL. *Italic orange values* ARE STATISTICALLY SIGNIFICANT RESULTS AT THE $\alpha = 0.05$ SIGNIFICANCE LEVEL, WHILE **BOLD GREEN VALUES** ARE ALSO STATISTICALLY SIGNIFICANT AT $\alpha = 0.01$

Congestion Type	AMG	UMT	pF3D
Light bully	-0.70%	**-0.68%**	0.00%
Adjacent leaf bully	**-1.74%**	-0.52%	-0.02%
Bisection bully	**-6.47%**	**-1.94%**	0.00%
Incast bully	*-0.74%*	0.01%	-0.14%
Interleaved	**-3.17%**	*-0.47%*	*-0.66%*

bandwidth between the two switches, while with 17 pairs it is attempting to consume more than 100%. Our next test, the bisection bully, involved the "full" bully communicating between pairs of nodes in different pods. While all of the neighboring leaf switch test's traffic only goes to the second level of the fat-tree, all of the bisection bully's traffic must go to the top level, which stresses the bisection bandwidth of the network.

In addition we used an incast pattern to create congestion. This 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. Our final experiment randomly interleaved the different applications on the system. We ran 10 applications at once on random nodes to try and mimic real world allocations and interference on production systems.

III. RESULTS

Overall, the bisection bully caused the largest performance impact. Since all of the traffic that the bisection bully creates traverses the diameter of the network, it competes with the application for available bandwidth on more links, creating more opportunities to slow down the application. In most cases, we observed that the slowest bully MPI rank received at least 49.5 Gbps of congesting traffic, but in practice, when running with other applications, the slowest bisection bully only received about 30 Gbps, while the slowest adjacent leaf bully rank received 45 Gbps or more. The missing bandwidth in all cases is likely consumed by two sources: the bullies self congesting, and the application congesting with the bully. We suspect that the static routes on Quartz may result in hot spots that are the key cause of this degradation.

Unlike AMG and UMT, pF3D was essentially unaffected by congesting background traffic. PF3D communicates in a domain of 18×8 MPI ranks, which maps almost optimally in these experiments. Since Quartz has 36 cores per compute node, the x direction FFT communication is node local, and the y direction FFT communication takes place in four-node groups. When run using 28 nodes per leaf switch, as in Control, each four-node group is connected to the same leaf switch, and when run with 14 nodes per leaf switch, only one group is split across two leaf switches within the same pod. When pF3D was interleaved with other applications, more of the four-node groups were split across leaf switches, increasing its susceptibility to network congestion, and resulting in some performance degradation. Previous work [1] has shown pF3D is sensitive to neighboring applications causing congestion on a torus where the FFTs can not be isolated from other actors. However, here we see that on topologies with many nodes per leaf switch, e.g., fat-tree and dragonfly, the FFTs can be isolated to reduce the impact of this type of congestion.

One unexpected result was the small effect the incast bully caused. Because incast patterns can cause catastrophic, rapidly-spreading congestion trees [4], we expected a substantial impact from the incast pattern. Additionally, prior to our experiments, we simulated incast traffic with a trace of UMT's communication to prototype this experiment, and the simulation demonstrated performance degradation greater than 15%. During the experiment, the incast pattern was only able to drive approximately 81 Gbps of traffic per receiver. If this is due to head-of-line blocking [5] causing the incast bully to congest with itself, then the bully caused the desired behavior. In this case, the negligible impact on application performance may have been due to static routing limiting the congestion to a few links out of each leaf switch. An alternative explanation we are also exploring is that the MPI rendezvous protocol acted as a form of back pressure, and prevented congestion trees from forming.

IV. CONCLUSIONS AND FUTURE WORK

To understand how congestion impacts the performance of different applications, we ran a variety of experiments and gathered their performance results. Overall, we found that the applications and network we tested were not significantly impacted by congestion, especially compared to similar results on dragonfly or torus networks. Although the bully experiments used more compact allocations than typical, the bullies constantly communicated unlike real applications, and performance losses were always less than 7%. This suggests that the 2:1 taper allows reasonable application performance even in the presence of aggressive background traffic.

AMG, which is latency sensitive and also has the most irregular communication pattern, was most impacted by congestion. We plan to look into how these features contributed to its sensitivity to congestion. Another significant finding is that pF3D's default mapping localized the majority of traffic to within a leaf switch, rendering it mostly unaffected by congestion due to background traffic.

For future work we plan to try similar mappings for UMT and AMG to determine if we can reduce the impact of congestion on these applications. We also want to analyze the switch counter data and look at the routing tables to determine if performance degradation was caused by hot spots in the network. If so, we would like to re-run these experiments on a machine with adaptive routing to see if this prevents relative hot spots from forming.

REFERENCES

[1] A. Bhatele, K. Mohror, S. H. Langer, and K. E. Isaacs, "There goes the neighborhood: Performance degradation due to nearby jobs," in *Proceed-*

ings 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.

[2] X. Yang, J. Jenkins, M. Mubarak, R. B. Ross, and Z. Lan, "Watch out for the bully!: Job interference study on dragonfly network," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '16. Piscataway, NJ, USA: IEEE Press, 2016, pp. 64:1–64:11.

[3] P. Taffet, I. Karlin, E. A. Leon, and J. Mellor-Crummey, "Understanding the impact of fat-tree network locality on application performance," in *The International Conference for High Performance Computing, Networking, Storage and Analysis (SRC Posters)*, Denver, Colorado, Nov. 12-17 2017.

[4] 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.

[5] P. J. Garcia, J. Flich, J. Duato, I. Johnson, F. J. Quiles, and F. Naven, "Decongestants for clogged networks," *IEEE Potentials*, vol. 26, no. 6, pp. 36–41, Nov. 2007.