Design and Evaluation of Topology-Aware Scatter and AllGather Algorithms for Dragonfly Networks

Nathanaël Cheriere
ENS Rennes
nathanael.cheriere@ens-rennes.fr

Matthieu Dorier
Argonne National Laboratory
mdorier@anl.gov

1. INTRODUCTION

Traditional network topologies of supercomputers such as torus become too expensive for the petascale, thus new topologies such as Dragonfly [5] have been designed. This cost-efficient design is used in numerous machines such as Cori and Edison at NERSC [1] and the Theta machine at ANL [2]. This opens opportunities to optimize the communication operations for this topology.

This poster focuses on two communication operations: AllGather and Scatter. For AllGather, we studied two algorithms from MPICH (BRUCK, RING) and one original (TAR). For Scatter, we looked at one algorithm used in MPICH (TREE), a simple one (LIN), and two inspired by Xiang and Liu [10] (GLF, LLF). Using simulation, we recorded the behavior of those algorithms on a Dragonfly topology and observed intersting results.

2. RELATED WORK

Because collective communication operations are critical for efficiency, many authors have improved them. Thakur et al. [9] detail the algorithms used in MPICH, they minimize the latency for short messages and bandwidth usage for larger ones, but they do not consider the topology of the network.

Jain et al. [4] have designed algorithms for networks with a PERCS topology [3] that is close to the Dragonfly. However, the algorithms they have developped can not be used on the Dragonfly topology as they exploit the specificities of PERCS to improve the communication operations.

Xiang and Liu [10] have proposed broadcast algorithms for Dragonfly. The present work extends their algorithms to other collective comunication operations; in particular, the *group-first* and *router-first* algorithms are the inspiration source for our GLF and LLF scatter algorithms, respectively.

3. METHODOLOGY

To get the results presented on the poster, we simulated the execution of all algorithms on a Dragonfly network using CODES [8], a network simulation framework that has already demonstrated its accuracy [6, 7]. We considered a network of 5,256 terminals with minimal routing (each packet is sent through the shortest path). The characteristics of the hardware is taken from the Cray XC30 which uses a Dragonfly network. Thus, the local links and terminal links have a bandwidth of 5.25 Gb/s and the global links have 4.7 Gb/s.

We focus our attention on the run time: from the first message sent to the last message received.

© 2016 ACM. ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

4. RESULTS

4.1 AllGather

We simulated the various AllGather algorithms with an allocation size varying from 128 to 4,096. Each terminal had 1 KiB to send to the other terminals. Results, presented on the poster, show that TAR is faster than RING by 10% for an allocation size of 128 and up to 2 times faster on an allocation of 4,096 terminals. BRUCK is faster than TAR for small allocations (under 1024 terminals) but slower by up to a factor of 2 for larger allocations. TAR sends more messages ($O(n^2)$), while BRUCK sends fewer messages but of larger size ($O(n \times log_2(n))$). Sending fewer messages improves BRUCK's run time when the allocation is small, but when the allocation is larger, the less-controlled patterns of communication of BRUCK (and RING) create contention in the network and decrease the run time. With TAR, the contention is reduced thanks to a carefully built ring that minimizes the link usage.

This minimization can also be observed with the average number of hops per message of TAR, which is lower than that of both RING and BRUCK. On average, the number of hops (number of routers that forwarded the message) for BRUCK and RING is 3.8. A high number of hops also increases the chances of interferences with other jobs.

Overall, BRUCK should be used for small amounts of data (up to a MiB) while TAR should be used for bigger transfers.

4.2 Scatter

We simulated the Scatter operation with allocation sizes varying from 128 to 4,096, each terminal receiving exactly 1 KiB of data.

The most surprising observation is that the naive algorithm, LIN, is 1.5 to 2 times faster than the other algorithms. This performance is due to the size of the buffers in routers. Messages sent by LIN are small and fit into the routers' buffers (1 KiB in 16 KiB). Besides, messages are sent to terminals in a random order, so the used buffers are not always the same and, thus, are never filled. On the contrary, the other algorithms send larger messages that fill the buffers quickly, and the sender has to wait for them to be emptied before sending the rest of the message. We studied the buffer saturation in the router attached to the root (that must send all the data) and observed that LIN does not saturate any buffer.

Beside LIN, the other algorithms implementing Scatter do not have significant performance differences between them. Adding

background traffic does not change the trends, but the performance for small allocations are slightly degraded.

Overall, LIN is twice faster than the other algorithms for the Scatter operation and also minimizes the overall amount of data transfered.

5. DISCUSSION: CAN WE ALSO IMPROVE THE PERFORMANCES OF ALLGATHER?

In the case of AllGather, all terminals in the allocation are sending data, which creates contention. Thus, it is not trivial to ensure that no buffer is saturated because of the amount of data transfered and the multiple senders. Moreover, because the run time of the algorithms is determined by the slowest link due to the ring structure, if only one buffer saturates, the performance of the algorithm drops.

Improving AllGather by reducing the saturation of the buffers is far from trivial and needs the creation of new algorithms for the AllGather problem.

6. CONCLUSION

In this work, we have designed multiple algorithms that reduce the amount of data transferred through the network. The simulations of the AllGather collective algorithms on CODES show expected results: RING can easily be improved, but has limitations when the allocation's size is small. Scatter however shows unexpected results: LIN is a lot faster than the other algorithms. To fully optimize algorithms for a network, all the parameters of the network must be taken into account: its topology, but also its hardware.

7. REFERENCES

[1] NERSC, Cray XC30 supercomputer. http://www.nersc.gov/users/computational-systems/edison/configuration/. Access: 2016-05-16.

[2] Theta, early model of the ALCF. http://aurora.alcf.anl.gov/. Access: 2016-05-16.

[3] B. Arimilli, R. Arimilli, V. Chung, S. Clark, W. Denzel, B. Drerup, T. Hoefler, J. Joyner, J. Lewis, J. Li, et al. The PERCS high-performance interconnect. In *IEEE 18th Annual Symposium on High Performance Interconnects (HOTI)*, 2010.

[4] N. Jain, J. Lau, and L. Kale. Collectives on two-tier direct networks. In *European MPI Users' Group Meeting*, pages 67–77, 2012.

[5] J. Kim, W. J. Dally, S. Scott, and D. Abts. Technology-driven, highly-scalable dragonfly topology. In *ACM SIGARCH Computer Architecture News*, 2008.

[6] M. Mubarak, C. D. Carothers, R. Ross, and P. Carns. Modeling a million-node dragonfly network using massively parallel discrete-event simulation. In *High Performance Computing, Networking, Storage and Analysis (SCC)*, 2012.

[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 *ACM SIGSIM/PADS Conference on Principles of Advanced Discrete Simulation*, 2014.

[8] M. Mubarak, C. D. Carothers, R. B. Ross, and P. Carns. Enabling parallel simulation of large-scale HPC network systems. *IEEE Transactions on Parallel and Distributed Systems*, 2016.

[9] R. Thakur, R. Rabenseifner, and W. Gropp. Optimization of collective communication operations in MPICH. *International Journal of High Performance Computing Applications*, 1(14):42–66, 2005.

[10] D. Xiang and X. Liu. Deadlock-free broadcast routing in dragonfly networks without virtual channels. *IEEE Transactions on Parallel and Distributed Systems*, 2015.