Prototyping of Offloaded Persistent Broadcast on Tofu2 Interconnect

Extended Abstract

Yoshiyuki Morie, Masayuki Hatanaka, Masamichi Takagi, Atsushi Hori, and Yutaka Ishikawa
Riken Advanced Institute for Computational Science, JAPAN
Kobe, Hyogo
yoshiyuki.morie@riken.jp

ABSTRACT

With the increasing scale of parallel computers, it has become more important to reduce communication time. Overlapping computation and communication is one effective method for hiding communication delay. Although standard non-blocking collective communication is an overlap method, it requires generating a communication command sequence for each collective communication. In contrast, persistent non-blocking collective communication can generate the sequence at initialization and reuse it at the start of collective communication. Moreover, if the sequence can be offloaded to a network device, more efficient execution is possible without using CPU cycles.

In this paper, a persistent non-blocking broadcast is implemented using the offloading functionality of the Tofu2 interconnect on the Fujitsu FX100 supercomputer, a successor to the K computer. We report the performance improvement by offloading persistent non-blocking collective communication on a real machine.

CCS CONCEPTS

• **Networks → Network experimentation**;

KEYWORDS

Persistent Collective communication, Offloading

ACM Reference Format:
Yoshiyuki Morie, Masayuki Hatanaka, Masamichi Takagi, Atsushi Hori, and Yutaka Ishikawa. 2017. Prototyping of Offloaded Persistent Broadcast on Tofu2 Interconnect. In *Proceedings of SC17, Denver, Colorado USA, Nov 2017,* 3 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Collective communication latency is a major concern, as it has become longer with the increase of scale of parallel computers. Overlapping computation and communication is one effective method for hiding communication delay. Recent networks have offloading functionality, so non-blocking communication is more important. Although non-blocking collective communication exists as an overlapping method, it requires generating a command sequence every time a non-collective communication is issued.

In contrast, persistent non-blocking collective communication can be separated into the initialization and the collective communication body, so the initialization can be removed from the loop. In such a persistent non-blocking collective communication, the sequence of communication commands is generated at the time of initialization. The communication command sequence is then reused for each collective communication instead of the sequence needing to be re-generated. Moreover, if the sequence can be offloaded to a network device, then more efficient execution is possible without using CPU cycles. Although some implementations of persistent non-collective communication exist[7], there is no implementation of persistent non-collective communication that uses the offload functionality.

The offloading functionality was added to the Tofu2 interconnect on the FX100 supercomputer, a successor to the K computer. In this paper, an offloaded persistent non-collective broadcast for the Tofu2 interconnect is implemented. We will report the performance improvement from offloading persistent non-blocking collective communication on a real machine.

2 RELATED WORK

As an example of using the Tofu2 offloading functionality, the reference implementation of non-blocking collective communication was proposed in [2]. However, this is not an implementation that takes into account the network topology, so it is not practical enough in terms of performance. Although offloaded persistent neighborhood collective is proposed in Tofu2, offloaded persistent non-blocking collective communication is not included[6].

As other network devices, Mellanox ConnectX-2 and Bull eXascale Interconnect[4] have an offload capability which can be used to implement the non-blocking collectives. However, no implementation of persistent non-blocking collectives has been proposed[5].

3 OFFLOADING MECHANISM ON TOFU2

Tofu2 has a framework for hardware-driven distributed execution of communication commands called session mode CQ (SMCQ).

As shown in Figure 1, SMCQ consists of a transmit order queue (TOQ) and three pointers: the producer, consumer, and scheduling pointers. The TOQ descriptor has the information for RDMA communication such as command type, source, destination, and session progress step (SPS) field. The producer pointer points to the end of

SC17, Nov 2017, Denver, Colorado USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

command sequences enqueued by the application. The consumer pointer indicates the end of commands executed.

When an RDMA engine processes an incoming RDMA operation packet, the scheduling pointer can be advanced by the SPS fields of the packet. Therefore, the progress of the command processing in SMCQ is driven by incoming packets and does not need the help of the CPU.

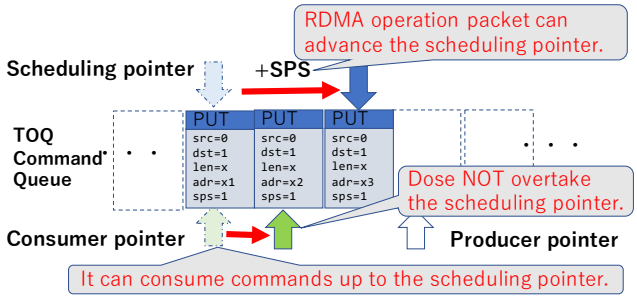

Figure 1: Overview of session mode CQ.

4 IMPLEMENTATION OF OFFLOADED PERSISTENT NON-BLOCKING BROADCAST

The implementation target broadcast algorithm is the Trinaryx3 algorithm used in the K computer[1]. This is an algorithm for improving the communication bandwidth by mapping multiple pipeline paths for the torus network. Blue Gene/L uses a similar algorithm[3].

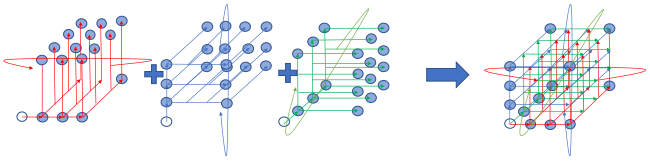

Figure 2: Overview of Trinaryx3.

The Trinaryx3 algorithm is an extension of the Trinary algorithm in which broadcast data are divided into three fragments, each of which is transmitted using the Trinary algorithm. The three pipeline paths of the trinary tree are mapped to the 3D torus in Figure 2. Each pipeline communication is offloaded and executed by SMCQ.

5 PERFORMANCE EVALUATION

Figure 3 shows the performance of an offloaded persistent non-blocking broadcast on the FX100. The offloaded persistent non-blocking broadcast is better than the non-blocking broadcast because the persistent non-blocking collective is able to complete creating the sequence of communication commands and exchanging the remote memory information at MP_Bcast_init().

Figure 3 shows the execution times of MPI_Bcast_init and MPI_Start. MPI_Bcast_init is relatively expensive and should not be repeated. Also, the cost of MPI_Start is sufficiently small, indicating that offloading is effective.

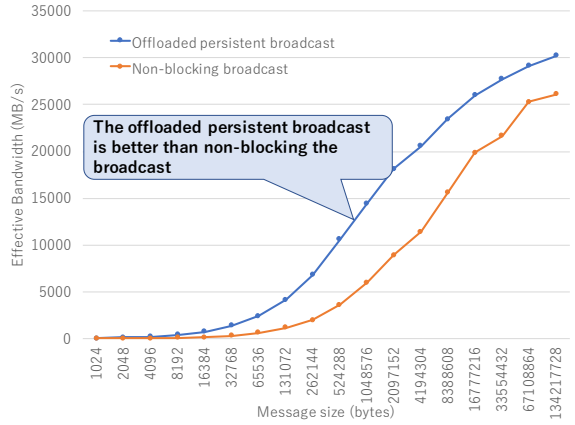

Figure 3: Effective bandwidth (MB/s) (12 processes).

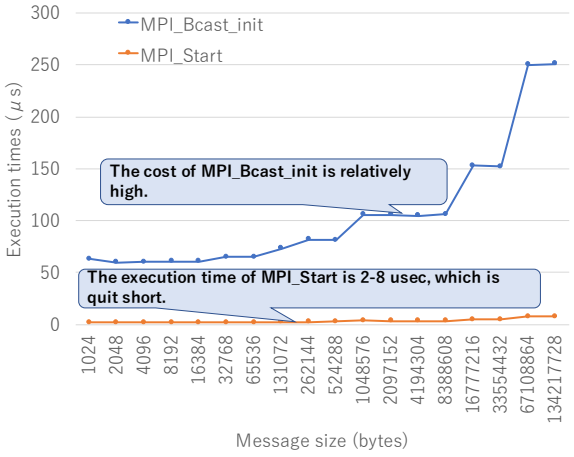

Figure 4: Execution times of MPI_Bcast_init and MPI_Start (μs) (12 processes).

6 CONCLUSIONS AND FUTURE WORK

This paper reports the effectiveness of offloading persistent non-blocking collective communication using the Tofu2 interconnect. Results of the evaluation demonstrate that an offloaded non-blocking broadcast outperforms a non-blocking broadcast. Moreover, the paper shows the relationship between the generation of the communication command sequence and the cost of offloading the command sequence of collective communication, revealing the importance of the interface of persistent collective communication and that of cost reduction by hardware offloading. Future work includes an evaluation of the implemented offloaded persistent non-blocking broadcast on a large-scale machine. The other collective communications will also be ported into offloaded persistent non-blocking collective communications on Tofu2.

REFERENCES

[1] T. Adachi, N. Shida, K. Miura, S. Sumimoto, A. Uno, M. Kurokawa, F. Shoji, and M. Yokokawa. 2013. The design of ultra scalable MPI collective communication

on the K computer. *Computer Science - Research and Development* 28, 2-3 (may 2013), 147–155.

[2] Y. Ajima, T. Inoue, S. Hiramoto, S. Ando, M. Maeda, T. Yoshikawa, K. Hosoe, and T. Shimizu. 2014. Tofu Interconnect 2. In *2014 IEEE 22nd Annual Symposium on High-Performance Interconnects*. 57–62.

[3] G. Almasi, P. Heidelberger, C. J. Archer, X. Martorell, C. C. Erway, J. E. Moreira, B. Steinmacher-Burow, and Y. Zheng. 2005. Optimization of MPI collective communication on BlueGene/L systems. In *the 19th annual international conference on Supercomputing*. 253–262.

[4] S. Derradji, T. Palfer-Sollier, J. Panziera, A. Poudes, and F. W. Atos. 2015. The BXI Interconnect Architecture. In *IEEE 23rd Annual Symposium on High-Performance Interconnects (HOTI'15)*. 18–25.

[5] S. Di Girolamo, P. Jolivet, K. D. Underwood, and T. Hoefler. 2016. Exploiting Offload-Enabled Network Interfaces. *IEEE Micro* 36, 4 (aug 2016), 6–17.

[6] M. Hatanaka, M. Takagi, A Hori, and Y. Ishikawa. 2017. Offloaded MPI Persistent Collectives using Persistent Generalized Request Interface. In *EuroMPI/USA 2017*. (to appear).

[7] I. B. Morgan. 2017. *Prototyping the Reference Implementation of Persistent Non-Blocking Collective Communication in MPI*. Ph.D. Dissertation. Auburn University.