Cache-Oblivious MPI All-to-All Communications on Many-Core Architectures

Shigang Li
SKL Computer Architecture,
Institute of Computing Technology,
Chinese Academy of Sciences
shigangli.cs@gmail.com

Yunquan Zhang
SKL Computer Architecture,
Institute of Computing Technology,
Chinese Academy of Sciences
yunquan.cas@gmail.com

Torsten Hoefler
Department of Computer Science,
ETH Zurich
htor@inf.ethz.ch

Abstract

In the many-core era, the performance of MPI collectives is more dependent on the intra-node communication component. However, the communication algorithms generally inherit from the inter-node version and ignore the cache complexity. We propose cache-oblivious algorithms for MPI all-to-all operations, in which data blocks are copied into the receive buffers in Morton order to exploit data locality. Experimental results on different many-core architectures show that our cache-oblivious implementations significantly outperform the naive implementations based on shared heap and the highly optimized MPI libraries.

Keywords cache-oblivious algorithms, MPI_Alltoall, many-core

1. Introduction

Many-core architectures tend to come with massive intra-node parallelism, deep memory hierarchies, and complex Networks-on-Chip (NoC). The Message Passing Interface (MPI) [5] is used ubiquitously for communication in parallel applications. The performance of MPI collective communications, which often determines the scalability of applications, becomes increasingly dependent on the intra-node component. Intra-node communication is essentially cache line transfer on the NoC. Thus, it is imperative to design algorithms for collective communications with high cache efficiency.

However, designing optimal communication algorithm in terms of cache efficiency is non-trivial. There are two major challenges. Firstly, traditional algorithms for MPI collectives [6] mainly focus on reducing the latency and bandwidth overhead over the network, but ignore the cache complexity. The second challenge comes from the diversity of the many-core architectures: processors may have different memory hierarchies, such as a two-level cache for Intel® Xeon Phi™ KNC or a three-level cache for Intel® Xeon® E7; furthermore, there are various arrangements of main memory, including Uniform Memory Access (UMA) or Non-Uniform Memory Access (NUMA).

Cache-oblivious algorithms [1] are asymptotically optimal in terms of cache complexity without considering any hardware parameters. Thus, these algorithms enable portable performance on different architectures. To carry these benefits towards implementations of MPI collectives, we propose cache-oblivious algorithms for MPI all-to-all style operations. The key idea is to arrange the order of transferring the send buffers to the corresponding receive buffers in Morton order [4]. We also demonstrate the performance advantages of our cache-oblivious algorithms on different many-core architectures.

2. Cache-Oblivious Algorithms for All-to-All Operations

We design the algorithms for intra-node collectives based on a shared heap. The technique of creating a globally shared heap for all MPI processes has been discussed in our previous work [2, 3]. All the send and receive buffers are allocated in a shared heap. Then, each process can directly access all the send and receive buffers, which enables more opportunities to exploit the data locality, and design cache-oblivious algorithms.

For MPI_Alltoall, also known as all-to-all personalized exchange, every process sends a distinct data block to every other process. Each process can view all the send buffers as a 2D matrix, of which each dimension is equal to the number of processes and each element represents a data block; and so do the receive buffers. We name these two matrices as 'send-buffer matrix' and 'recv-buffer matrix', respectively. In a naive implementation of MPI_Alltoall, each process copies a column of the send-buffer matrix into its receive

buffer (i.e., a row of the recv-buffer matrix). The access to the send-buffer matrix exhibits poor spatial locality because of the row-major property of the matrix. To have good spatial locality for both send-buffer and recv-buffer matrices, we use Morton order [4] (also known as Z-order) to guide the copying of data blocks into recv-buffer matrix. Figure 1 shows MPI_Alltoall with 8 processes based on Morton order. The Z-shaped curve is equally divided into 8 segments and each one is handled by a process. Following the Z-shaped curve, the spatial locality of MPI_Alltoall is exploited.

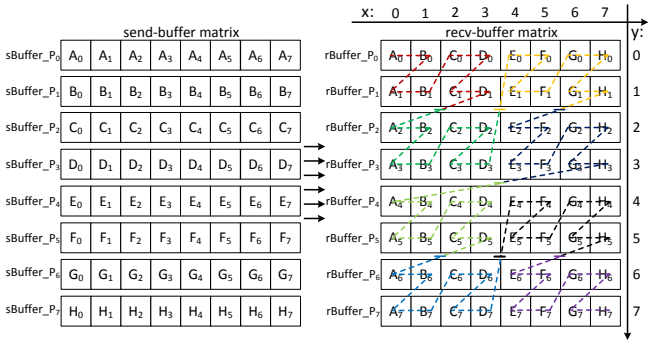

Figure 1. MPI_Alltoall with 8 processes based on Morton order.

For MPI_Allgather, also known as all-to-all broadcast, each process sends the same data block to all other processes. Each process can view all the send buffers as a vector (we call it "send-buffer vector") and the receive buffers as a matrix (we call it "recv-buffer matrix"). In a naive implementation of MPI_Allgather, each process copies the send-buffer vector into its receive buffer (a row of the recv-buffer matrix), which exhibits poor temporal locality. As for MPI_Allgather, we use Morton order to guide the copying of data blocks from send-buffer vector into recv-buffer matrix. In this way, the temporal locality of send-buffer vector is exploited.

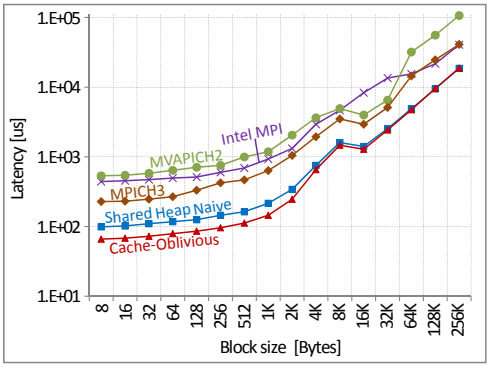

Figure 2. Latency comparison of MPI_Alltoall between traditional MPI, naive shared heap, and the cache-oblivious implementation on Intel Xeon Phi.

Preliminary experiments are conducted on Intel Xeon Phi KNC 5110P and Intel Xeon E7-8890 v3. Xeon Phi is a UMA architecture with 60 cores. Xeon E7-8890 is a NUMA architecture with 72 cores. Figure 2 presents the latencies of MPI_Alltoall with different block sizes on Xeon Phi. On Xeon Phi, our cache-oblivious implementation for MPI_Alltoall outperforms the naive implementation based on shared heap by 40% on average when the block size is less than 16 KB, and outperforms MPICH3 by 211% on average over all the block sizes. On Xeon E7-8890, our cache-oblivious implementation for MPI_Alltoall also has significant performance advantage. These demonstrate that our cache-oblivious algorithms achieve portable performance improvement on different architectures.

3. Conclusion

As supercomputers evolve into exascale, data movement is growingly expensive. We propose cache-oblivious algorithms for MPI all-to-all style collectives to exploit data locality. Experimental results show that our cache-oblivious algorithms achieve portable performance improvement on both UMA and NUMA architectures. Architecture trends indicate massive intra-node cores and deep memory hierarchies will be necessary to reduce power consumption and contention on buses. We foresee that the benefit of our cache-oblivious algorithms will be more significant on such future machines.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant No. 61502450, Grant No. 61432018, and Grant No. 61521092; National Major Research High Performance Computing Program of China under Grant No. 2016YFB0200803.

References

[1] M. Frigo, C. E. Leiserson, H. Prokop, and S. Ramachandran. Cache-oblivious algorithms. *ACM Transactions on Algorithms (TALG)*, 8(1):4, 2012.

[2] S. Li, T. Hoefler, and M. Snir. NUMA-aware shared-memory collective communication for MPI. In *Proceedings of the 22nd international symposium on High-performance parallel and distributed computing*, pages 85–96. ACM, 2013.

[3] S. Li, T. Hoefler, C. Hu, and M. Snir. Improved MPI collectives for MPI processes in shared address spaces. *Cluster Computing*, 17(4):1139–1155, 2014.

[4] G. M. Morton. *A computer oriented geodetic data base and a new technique in file sequencing*. International Business Machines Company New York, 1966.

[5] MPI Forum. MPI: A Message-Passing Interface standard. Version 3.0, September 2012.

[6] R. Thakur, R. Rabenseifner, and W. Gropp. Optimization of collective communication operations in MPICH. *International Journal of High Performance Computing Applications*, 19(1): 49–66, 2005.