Fast adjacent communication among with RDMA, MPI-RMA, and Double buffering

Kota Yoshimoto
Kogakuin University
Japan
em22027@ns.kogakuin.ac.jp

Akihiro Fujii
Kogakuin University
Japan

Teruo Tanaka
Kogakuin University
Japan

1. INTRODUCTION

In large-scale scientific computing programs, parallelization by MPI communication is generally used. MPI is convenient because it can be executed on many computers. However, inter-process communication often becomes a bottleneck in highly parallel computers. There is an interface called RDMA (Remote Direct Memory Access) to reduce the delay caused by communication.

We have previously evaluated the performance with different numbers of neighbors [1]. In this research, we conducted a performance analysis includingchange of the number of processes per node . We also evaluated the performance of one-sided communication using MPI.

2. RDMA and RMA with Double Buffering

RDMA communication is one-sided communication. By using dedicated memory, it can read and write data without going through the destination node's program. There are two types of RDMA communication: Put communication and Get communication. In this experiment, we used Put communication. There is RMA (Remote Memory Access) communication using MPI. This is also one-sided communication. It is realized by MPI_Put, MPI_Win_fence, MPI_win_lock/unlock.

In these methods with double buffering, two buffers for communication are prepared, and when the data in the buffer is being read, the other one communicates. Each process has two communication buffers so that it can read one buffer while receiving communication on the other buffer. This reduces the cost of communication and speeds up the process.

3. Numerical Experiment

In this experiment, we used a computer called Wisteria-O at Tokyo University. It is equipped with a system called FX1000.

In this experiment, MPI_Isend/Irecv, MPI_RMA (MPI_Put, MPI_Win_fence), MPI_RMA with double buffering (MPI_Put, MPI_Win_lock/unlock), RDMA, and RDMA with double buffering were evaluated by adjacent communication, respectively. The total number of processes in the tests was fixed at 128.

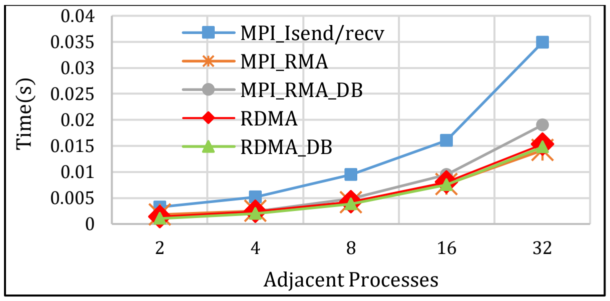

Figure 1. Number of adjacent communication when adjacent processes is changed

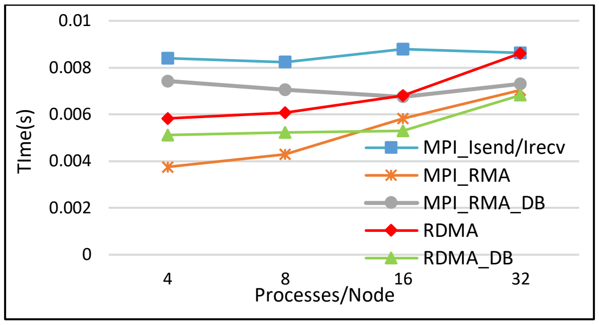

Figure 2. When changing the number of nodes

Figure 1 shows adjacent communication when changing the number of adjacent processes. The number of adjacent processes was changed from 2 to 32 and the processes per node ratio was 4. In Figure. 1, the vertical axis is time of adjacent communication and the horizontal axis is the number of adjacent processes. As a result, all one-sided communication methods had similar performance and were faster than MPI_Isend/Irecv.

Figure 2 shows adjacent communication when changing the number of processes per node. The number of adjacent processes was 32 and the processes per node was from 4 to 32. In Figure 2, the vertical axis was time and the horizontal axis was processes per nodes. As a result, MPI_RMA was the fastest when process per node was small. Also, RDMA with double buffering was the fastest when the processes per node was large.

4. Conclusion

MPI_Isend/Irecv, MPI_RMA, RDMA and RDMA with double buffering were evaluated using adjacent communication. MPI_RMA is effective when the number of processes per node is small. However, when there are many processes per node, RDMA using double buffering is also effective. We will present numerical experiment result with different inter-process topologies and different message sizes in the poster.

ACKNOWLEDGMENTS

This work is supported by JSPS KAKENHI Grant Number JP22K12063, “Joint Usage/Research Center for Interdisciplinary Large-scale Information Infrastructures” and “High Performance Computing Infrastructure” in Japan (Project ID: jh230060-NAH).

REFERENCES

[1] Kota Yoshimoto, Akihiro Fujii, Teruo Tanaka, RDMA with Double Buffering for Adjacent Communication, HPCAsia2022 poster session (2022).