# Homework5 report

## Kangdong Yuan

## Introduction

In this part, the goal of this homework is to use MPI in C++. Mpi is Message Passing Interface, which enables a node to send and receive from other nodes. This Mpi is designed for parallel programming. The program I need to write is to use the function of MPI to implement multi-nodes running matrix multiplication. The matrix multiplication  I need to use is the cannon algorithm. The Cannon algorithm is a distributed algorithm for matrix multiplication for two-dimensional meshes. The cannon algorithm works better than Scalable Universal Matrix Multiplication Algorithm, because the cannon algorithm doesn't need a square 2D grid.

## System information

The system I run those programs is red-hat 7, and CPU is Intel E5 2680. And each node has 1 core, and the random-access memory is 1G size for every node.

## The learning process

I use this chapter to describe how I learned MPI and cannon, because I learned a lot in the process of learning these contents. Before starting to learn MPI, we have learned openmp, which is a library similar to MPI. But the openmp is very simple and easy to understand and write a parallel program. I only need to add openmp instructions before the program blocks that I want to do parallel computation, and openmp will help me run this program in parallel. But for MPI, I need to understand how different cores operate and transfer information, so MPI requires me to have a very deep understanding of parallel programs.

First, the biggest problem I encountered while learning MPI is that the functions for MPI to receive and send messages are very complicated. For example, the functions for receiving messages need to fill in many parameters, but I don't know what these parameters mean. The second point is that I don't have a clear roadmap for how to transmit information between different nodes. For example, I don't know when node 0 should receive information sent by other nodes.

But for the cannon algorithm, I think I have learned a lot. Cannon has two phases, Skew phase and Matrix multiply phase. Both phases require MPI to transmit information in different nodes.

After trying many times, I always fail in writing a MPI program. So I started following the instructions of the professor. Professor teaches us to write the SUMMA algorithm, SUMAA algorithm is very similar to cannon algorithm. The SUMMA algorithm just divides the matrix into many blocks. And use one node to broadcast and gather the information, and other nodes to compute the small blocks. Then, the professor taught us how to convert the SUMMA algorithm to cannon algorithm. For cannon algorithms, we need to divide the whole computation process into two parts. The first part is do a row and column shift when we get the original matrices. Then, we need to compute the result by doing the matrix multiplication for each small block. And we do the row and column after each computation. Finally, we can get the result matrix after the final shift.

Although the professor shows how to write this program in lecture. But I still meet a lot of problems, like bugs, slow running time, MPI error. I have cost a lot of time to fix, although I think my program is not perfect now.

## Testing

After completing this program, I start to test my program. I use pbs to request the resource for parallel computing. I tried N=4, and node =2, and it ran very quickly and returned the right answer. Then I tried N=20, and node =16. The time to request the resource take a while, and it ran about 5 mintues and returned the right answer.  Then I tried N=200, and node =64. The time to request the resource took half hour, and it ran about 45 mintues and returned the right answer. Finally I find that I can run my program on 100 nodes with N=500. However when I run my program with N=1000 and node=120, the program runs too long and passes the walltime.
From previous testing I can find that the scalability of my program is good, which can handle so many nodes to do matrix multiplication. Moreover, the cannon algorithm is very suitable for parallel computing because it divides the whole computation process into two phases with many small matrix multiplication.

## Conclusion

The cannon algorithm is a good algorithm in doing the parallel matrix multiplication. And the cannon algorithm needs to do the row shift and column shift, which require to send and receive the information betweens different nodes. So, MPI is a fitable tool to write the cannone algorithm. However, the MPI program does not show a higher efficiency than the openmp program. If I can choose, I will use openmp because it is easy to use.