Assignment 1: Parallel Matrix Multiplication

CS328 - Distributed and Cloud Computing

DDL: 23:59, October 27, 2024

1 Introduction

Matrix Multiplication can be processed in a parallel fashion. As shown in Figure 1, the first matrix can be sliced into pieces where each piece possesses several rows. Then, these pieces can be multiplied by the second matrix separately and concurrently. Finally, the local multiplication results can be stacked together to construct the final output.

2 Requirements

In this assignment, you are required to use **Open MPI**[1] to implement parallel matrix multiplication and test it with **Docker**[2]. There are two matrices, A and B, both of shape 500×500, with the elements in `double` type. The **root** process should control the data distribution and gathering. Other **worker** processes should receive the data from the root, compute, and send the local result back to the root. Finally, the root process should compare the result with the brute force approach (computation without parallelization) to check the correctness. The root process should also hold a piece of the input data after distribution and perform part of the computations for simplification.

After completing the code, run it locally with 1, 2, 4, 8, 16 and 32 (utilise the option `--oversubscribe` when necessary) processes and draw a line chart to analyze the relationship between the number of processes and the computation latency.

Finally, use **Docker Compose**[3] to set up 3 containers and test the code in this distributed environment. To be specific, use the following command to enter one of the containers and execute your program in a distributed manner over all containers:

```
docker exec -it [container_name] /bin/bash
```

A reference code (`mpi_matrix_base.c`) has been provided, and you may use it to build your program.

[1] Open MPI: https://www.open-mpi.org/

[2] Docker: https://www.docker.com/

[3] Docker Compose: https://docs.docker.com/compose/

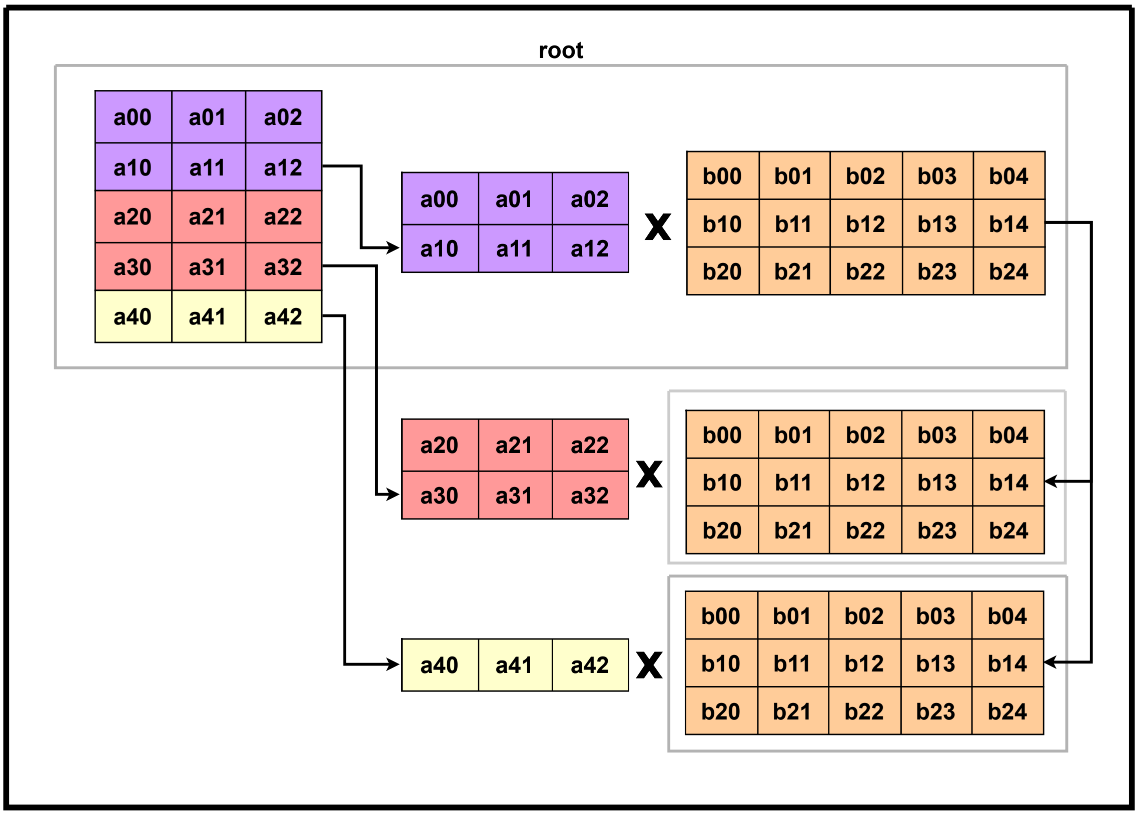

Figure 1: Parallel Matrix Multiplication.

3 Hints

Use `MPI_Scatter`, `MPI_Bcast` to distribute data to worker processes, and use `MPI_Gather` to collect the result in the root process. Visit the official documentation and identify which function variant is required!

4 What to Submit

1. Source code folder, including `mpi_matrix.c`, `Dockerfile`, `compose.yaml` and other relevant code files.
2. A report (using the provided template on Blackboard) in PDF format, including:
   - Description about how your code works (also prove that it provides the correct result by comparing it with the brute force approach).
   - A line chart, with the number of processes as X axis and the running time as Y axis (try to analyze the pattern shown in the line chart).
   - Screenshots of the code execution result (both local and the distributed environment constructed by Docker Compose), docker image building result.

Pack all files into `SID_NAME_A1.zip`, where `SID` is your student ID and `NAME` is your pinyin/English name (e.g., `11710106_ZhangSan_A1.zip`).