Assignment 1: Parallel Matrix Multiplication

CS328 - Distributed and Cloud Computing

DDL: 23:59, Monday, October 16, 2023

1 Requirements

In this assignment, you are required to use Open MPI to implement parallel matrix multiplication. Consider two matrices, A and B, both with shape 500×500, with the elements in `double` type. In your code, there should be an MPI process that acts as the **master** to control the data distribution and gathering. The **master** process splits and distributes the computation of matrix multiplication to other **worker** processes **and itself**. All the processes then do the computation assigned to them, and finally send the results back to the **master**. Then the master process should compare the result with the brute force approach (calculating the matrix multiplication without parallelisation) to check the correctness of parallel computation.

When splitting the matrix into different processes, a proper mechanism should be designed to allow non-even splitting. For instance, when splitting the matrices of size 500 $\times$ 500 to 32 processes, there is no way to split the matrix in equal size onto each process. You are required to design a method to dynamically solve this problem.

Run your program with 1, 2, 4, 8, 16 and 32 MPI processes on one virtual machine (VM). For instance, if you run 4 processes (1 master and 3 workers), the command should be similar to: `mpirun -np 4 ./xx`. Then, plot a figure to show the scalability in terms of *number of MPI processes* vs. *computation time*. Make sure the VM is allocated with enough memory and CPUs. Analyse the figure and explain the performance.

Run your program with 1, 2, 4, 8, 16 and 32 MPI processes on the 2-VM cluster that you've created in Lab 2. Plot the relation between *number of MPI processes* vs. *computation time*. Compared with running on one VM, what are the differences when running on two VMs? Are there any abnormal behaviours? Describe your findings and provide possible explanations.

Bonus: Is there any way to optimise the parallelisation for improved computation efficiency?

2 Hints

A code stub (`mpi_matrix_demo.c`) will be provided, and you may use it to build your program.

You can use `MPI_Scatter` or/and `MPI_Bcast` to distribute data to worker processes, and use `MPI_Gather` to collect the result in the master process.

When running more MPI processes than the number of processors, use a hostfile to enable the node to be "oversubscribed" [1].

3 What to Submit

1. All source code files.
2. A report (using a provided template) in PDF format, including:
    - Description about how your code works (prove that your code provides the correct result by comparing it with the brute-force approach).
    - A figure, showing the performance when running on **one VM**. The X axis is the number of MPI processes and the Y axis is the running time.
    - A figure, showing the performance when running on **two VMs**. The X axis is the number of MPI processes and the Y axis is the running time.
    - **Detailed** description and explanation of any patterns/trends shown in the figures

Please write your report with as much detail as you can, since if your code runs without error, the marking will be based on the quality of your report. "Good quality" includes but is not limited to: a comprehensive description of your design with code and **plain words**, experiment setup and procedure, in-depth analysis of the results you get from the experiments, etc.

Pack all files into `SID_NAME_A1.zip`, where `SID` is your student ID and `NAME` is your name (e.g., 11710106_ 张三 _A1.zip).

[1] https://www.open-mpi.org/faq/?category=running#oversubscribing