CSCE 435 Fall 2023

Assignment 2: Matrix Multiplication using MPI

Instructions to compile and execute the matrix multiplication code:

1. Upload the files (**build.sh, CMakeLists.txt, mpi_mm.cpp, mpi.grace_job**) to your scratch directory after logging into grace portal.

2. Initialize the cmake build:

    a. $. build.sh

3. If further changes are made to the code, use make to re-build:

    a. $ make

4. Run the batch file, giving matrix size and number of processors:

    a. $ sbatch mpi.grace_job <m> <p>

5. After a job is complete, you'll be able to see the output in the output file corresponding to your jobid in the same directory as the source code. The Caliper performance measurements will be written to the .cali file. (You'll be able to find out whether a recent job has been completed or not by going to: grace dashboard > jobs > active jobs)

6. If you are running less than 48 processes, only request 1 node (--nodes = 1), and the specific number of cores you need (--ntasks-per-node = numCoresYouNeed). If you are running greater than 48 processes, you should increase the number of nodes you are requesting accordingly.

7. On your grace dashboard, when starting up the Jupyter notebook, for the module selection, choose **Python/3.10.4** and for the type of environment, choose **Module load + Python virtualenv**. You can change the number of hours to however much you believe you will need to work on the graphs but leave everything else as default. **Reference the slides if you do not know how to go to your grace dashboard.**

Assignment:

Summary

- The structure of lab 2 is the same as lab 1, except this time you have to annotate the regions with Caliper as well. If you desire, you can copy the MPI_Reduce and other sections from lab 1 to lab 2. Note, you cannot copy your entire lab 1 and replace it to lab 2.
- Use Caliper & MPI_Reduce to calculate the minimum, maximum and the average runtime taken by the "**receiving**" part, the "**calculation**" part, and the "**sending**" part of the **worker** processes. Plot **minimum time, maximum time, and average time** as functions of the **number of processes** where the number of processes will vary between {2, 4, 8, 16, 32, 64, 128}. (You may plot all the three times: min, max, avg on a single graph with different colors)
    - There will be different plots for:
        - Matrices of size 128x128 **[10 points]**
        - Matrices of size 1024x1024 **[10 points]**
        - Matrices of size 8192x8192 **[10 points]**
- Similarly, in the master process, compute the runtime for the "**whole computation**", for the "**initialization**" part, and for the "**sending and receiving**" part and plot graphs for **initialization time**, **send/receive time**, and **total time** vs number of processes where the number of processes will vary between {2, 4, 8, 16, 32, 64, 128}.
    - There will be different plots for:
        - Matrices of size 128x128 **[10 points]**
        - Matrices of size 1024x1024 **[10 points]**
        - Matrices of size 8192x8192 **[10 points]**
- Write down your observations on the variation of runtimes with the number of processes for various matrix sizes. **[15 points]**
- Correctness of the MPI_Reduce code changes **[10 points]**
- Correctly annotated Caliper regions **[5 points]**
- Used Thicket to generate plots **[10 points]**
- **Note:** Because you will need a max total of 128 processors, you will need to calculate the number of nodes needed. If you request 16 nodes and 4 tasks per node, it gives you a total of 64 processors max that you can use in your request. Modify the number of tasks per node and number of nodes in the mpi.grace_job file to get 128 processors you will need when running the program for 128 processors. **Remember: each node has 48 cores and therefore you can use it to run at most 48 processors.**
- For more information regarding the hardware: https://hprc.tamu.edu/kb/User-Guides/Grace/

Part 1 - Caliper

You will use Caliper's region and function annotations to gather performance measurements for the runs. The Caliper annotation API. Caliper functions used:

- CALI_CXX_MARK_FUNCTION
    - C++ macro to mark a function.
- CALI_MARK_BEGIN(name)
    - Mark begin of a user-defined code region.
- CALI_MARK_END(name)
    - Mark end of a user-defined code region.

The CALI_CXX_MARK_FUNCTION's are already implemented for you. Your assignment is to open and close the regions using CALI_MARK_BEGIN(name) and CALI_MARK_END(name). The region names are already defined for you.

If correctly implemented, the caliper config will automatically collect and aggregate the min, max, and average times into the caliper file.

Part 2 - MPI

1. Implement timers – Use MPI_Wtime to create timers to measure over the same regions that you marked with caliper.
2. MPI_Reduce - Use MPI_Reduce to calculate min, max, and sum (used to compute average) times for each of your worker timers. Be careful, if you use MPI_COMM_WORLD as your communicator, your values will be incorrect. This is because your master process will be implicitly included in your worker calculations. You need to either:
    a. Create a new MPI communicator that excludes the master process
    b. Initialize the variables you are reducing to in a way that avoids this problem.

Note: Your timers from Part 1 and their equivalent timers in Part 2 should be almost equal in comparison, as you are timing the same thing.

Part 3 – Plotting and Observations

You are to plot the graphs using Thicket from the Caliper files you had collected. You may use the included notebook as your starting point. Write down your observations from the plots and put the plots and the observations into the report.

Upload a .zip file on canvas containing:

- **A pdf of the report.**
- **mpi_mm.cpp file with your code changes.**
- **The .cali files**