Assignment 1 : Matrix Multiplication using MPI

CSCE-435 Fall 2025
Texas A&M University

September 17, 2025

1 Compile and execute project

1. Upload the starter code to your home directory after logging into the grace portal.
2. Navigate to the directory that the files were uploaded to after logging into the grace portal.
3. Allocate an interactive node using

```
$ salloc --nodes=1 --ntasks=1 --cpus-per-task=4 --mem=8G --time=02:00:00 --
partition=short
```

   Running `salloc` will allocate a node for you and you will receive the output

```
salloc: Granted job allocation [job_id]
salloc: Nodes [node_id] are ready for job
```

   where `node_id` is the node allocated to you. Also keep note of the `job_id` associated with this request as you will need it to free up the node later.

4. `ssh` into the allocated node as

```
$ ssh [node_id]
```

   Make sure your prompt is now `user_id@node_id`

5. Initialize the CMake build using the command:

```
$ . build.sh
```

6. After your build is done, deallocate the node as

```
$ scancel [job_id]
```

   where `job_id` refer to the job of allocating a node you received when running `salloc` in step 3.

7. Run the batch file by running the following command, making sure to specify the matrix size (**m**) and the number of processors(**p**):

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

8. Once the job is completed, you will be able to see the output file named according to the job id.

9. **Whenever** a change is made to the code, allocate a node using `salloc` as described in step 3 through 6 and run make to re-build:

```
$ make
```

Make sure to free up the node after the build is done.

2 Assignment

Overview

In this assignment, you will be analyzing how the given MPI code scales with different matrix sizes and number of processes. You will mark regions of the code and observe how the different parts of the code scale as the number of processes increases.

Worker Processes

Use `MPI_Reduce` to calculate the *minimum*, *maximum* and *average* runtime taken by the **receiving**, **calculation** and **sending** part of the worker processes. These regions are marked in the code using comments.

Master Process

Use `MPI_Reduce` to calculate the *minimum*, *maximum* and *average* runtime taken by the **whole computation**, **initialization** and **sending & receiving** part of the worker processes. These regions are marked in the code using comments.

Implementation

- **Implement timers:** Use `MPI_Wtime` to create timers to measure over the regions marked in the code.(Read the comments in provided code)
- **MPI_Reduce:** Use `MPI_Reduce` to calculate the min, max and average times for each of the worker timers.
  NOTE: 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
  - Create a new MPI communicator
  - Initialize the variables you are reducing to in a way that avoids this problem

Inputs(For the 500 section only!!)

For both the **master** and **worker** process, vary the process and the matrix size as follows

- **Number of processes:** [2,4,8,16,32,64].
- **Matrix Size** [128 ×128, 1024 ×1024, 8192 ×8192].

Plot the runtimes observed for the regions marked previously. The plots can be made using excel. You may plot all three times: max, min, avg in a single graph with different colors.

NOTE: Each node has 48 cores so you will have to change the number of nodes in order to get 64 processors.

NOTE: Only request the minimum number of nodes required to get the required number of processes. For example do if you want 32 processes, do not request 8 nodes with 4 tasks per node. Instead request 1 node with 32 tasks pernode.

Inputs(For the honors section(200) only!!!!)

For both the **master** and **worker** process, vary the process and the matrix size as follows

- **Number of processes:** [2,4,8,16,32,64,128].
- **Matrix Size** [128 ×128, 1024 ×1024, 8192 ×8192].

Plot the runtimes observed for the regions marked previously. The plots can be made using excel. You may plot all three times: max, min, avg in a single graph with different colors.

NOTE: Each node has 48 cores so you will have to change the number of nodes in order to get 64 processors.

NOTE: Only request the minimum number of nodes required to get the required number of processes. For example do if you want 32 processes, do not request 8 nodes with 4 tasks per node. Instead request 1 node with 32 tasks pernode.

Analysis

Analyze the plots and explain the observations. Write down the explanations for the trends observed for each plot.

Grading

Exercise	Points
Worker process time plots(10 points for each matrix size)	30
Master process time plots(10 points for each matrix size)	30
Observations	20
Correctness of code	20

3 Submission

Submit a `.zip`file on Canvas containing

- A **pdf** with the plots and your observations.
- `mpi_mm.cpp` file with your code changes