



Exercise 12

Task 1 (row_wise_matrix_mult.c)

Download the program `row_wise_matrix_mult.c` from the e-learning platform and study the source code carefully.

The program implements a parallel matrix multiplication using MPI. The matrix `matrix_a` is multiplied with the matrix `matrix_b`, and the result is stored in `matrix_c`. The computation is distributed across multiple MPI ranks.

The goal of this exercise is to understand the basic working principle of the program: How is the computation distributed, and which tasks are performed by the individual ranks?

Discuss your observations with your neighbor and answer the following questions:

- Which rank(s) are responsible for initializing and filling the matrices `matrix_a` and `matrix_b`?
- How are the matrices `matrix_a` and `matrix_b` distributed among the MPI ranks? Do all ranks receive the complete matrices, or does each rank receive only specific parts (e.g., rows, columns, or submatrices)?
- Which parts of the result matrix `matrix_c` are computed by an individual rank?
- What happens to a computed partial result of the matrix `matrix_c` immediately after a rank has finished its computation?
- What role does rank 0 play in the program, and what roles do the other ranks assume?
- When sending parts of the matrix `matrix_a`, MPI tags are used to retain information about the portion of the result matrix `matrix_c` that is currently being processed:
 - When is the tag set to the value 0 by the sender?
 - How do the receiving ranks react when they receive a message with tag 0?

Task 2: EduMPI Setup and Program Execution

In the following, the parallel matrix multiplication should be executed on the Fulda HPC cluster and analyzed for runtime behavior and performance.

Please carry out the following tasks on a **local computer in the Linux Lab**, as all required tools for performance analysis are available there.

Preparation

First, download the EduMPI GUI (version 1.1.9) from:

```
https://github.com/AnnaLena77/EduMPI_GUI_Download.git
```

EduMPI GUI is provided as an AppImage and does not require installation. If necessary, make the file executable:

```
chmod +x EduMPI_GUI-1.1.9.AppImage
```

When using EduMPI GUI for the first time, a passwordless SSH connection to the cluster must be set up. This is required to execute MPI programs via the GUI.

Instructions for setting up the SSH connection can be found in the *Help* menu of EduMPI GUI. Use your regular cluster credentials (fd number and password).

Connecting to the Cluster

When starting EduMPI GUI, log in to the cluster using your fd number and establish a connection to the performance database. The required settings can be configured in the *Settings* menu.

Cluster Settings

- IP address: 10.32.47.10 (default)
- Name: your fd number (e.g., fd023456)
- Path: /opt/edumpi (default)

EduStore Settings

- Hostname: 10.35.8.10 (default)
- Port: 5432 (default)
- Database: edumpi_td (default)
- User: edumpi (default)
- Password: edumpi

Program Execution

Execute the parallel matrix multiplication using EduMPI GUI.

Select the program `row_wise_matrix_mult.c` via the button in the right sidebar of the GUI. Use matrices of size 8000×8000 and execute the program with **256 MPI processes**. EduMPI GUI automatically distributes the specified number of processes across available nodes on the cluster. It is not necessary to specify the number of nodes to be used.

Start the program and observe the execution.

Task 3: Performance Analysis with EduMPI GUI

In the following, the program is divided into two program phases: the distribution of `matrix_b`, and the second program phase, where `matrix_a` is distributed row by row to the processes, which compute partial results and send them back to the master.

Use EduMPI GUI along with all available views (2D, 3D, Communication Matrix) and options (send/recv ratio, max send ratio, max recv ratio, late sender, late receiver) to analyze the performance of the matrix multiplication.

After program termination, use the Timeline to navigate through the execution. You can reopen the EduMPI run from the database EduStore via the *Settings* menu.

Answer the following questions based on your observations and the collected performance data:

- How and across how many nodes were the MPI processes distributed on the cluster?
- How long does it take to fully distribute `matrix_b`, and which MPI function is used for this?
- Analyze the first program phase. Which problems can you identify, and why do they occur?



- Examine the second program phase, where `matrix_a` is distributed row by row to processes that compute partial results and send them back to the master. What problems can be identified in this phase, and why?
- Would you describe the MPI communication in this program as centralized or distributed? How do you rate the efficiency and scalability of this parallel matrix multiplication?
- Considering the program over its total runtime, which MPI function:
 - takes the most time,
 - causes the largest communication overhead, and
 - transfers the largest amount of data?

Task 4: Performance Analysis with CUBE and TAU Paraprof

In addition to EduMPI GUI, traditional performance analysis tools can be used. To analyze the program with **CUBE** and **TAU Paraprof**, performance traces or profiles must first be generated on the cluster.

For this purpose, the tools **Score-P** and **Scalasca** can be used.

You may either adapt and use the provided SLURM script from the e-learning platform, or generate the required files via the *EduMPI Non-Visualization* menu.

The two performance analysis tools can be started from a terminal:

```
cube  
paraprof
```

In both tools, the generated file `summary.cubex`, which contains the collected performance data, can be opened.

Use CUBE and TAU Paraprof to analyze the program and try to answer the questions from **Task 3** again based on the information provided by these tools. Focus in particular on:

- Runtime distribution across MPI functions
- Communication overhead and performance issues
- Differences in perspective compared to EduMPI GUI analysis