Università della Svizzera italiana

Institute of Computing CI

High-Performance Computing Lab **Institute of Computing**

Student: Metehan Dündar Discussed with: Francesco Gnan

Solution for Project 5

1. Task 1 - Initialize and finalize MPI [5 Points]

In the `main.cpp` file, the MPI library is utilized to enable parallel execution of the code. The initialization and finalization of MPI are essential steps for creating a parallel environment.

`MPI_Init` initializes the MPI environment, and `MPI_Finalize` finalizes it before program termination. The `MPI_Comm_rank` and `MPI_Comm_size` functions are used to retrieve the current rank and the total number of ranks, respectively.

2. Task 2 - Create a Cartesian topology [10 Points]

In the `data.cpp` file, domain decomposition was implemented using MPI's Cartesian topology. This involved determining the number of subdomains in each dimension, creating a non-periodic Cartesian topology, and retrieving the coordinates of the current rank in the domain grid. This approach facilitates efficient communication between neighboring subdomains.

MPI functions is utilized to create a Cartesian topology, retrieve process coordinates, and establish neighbor relationships in the grid. The use of a Cartesian topology streamlines communication in the multidimensional grid.

3. Task 3 - Change linear algebra functions [5 Points]

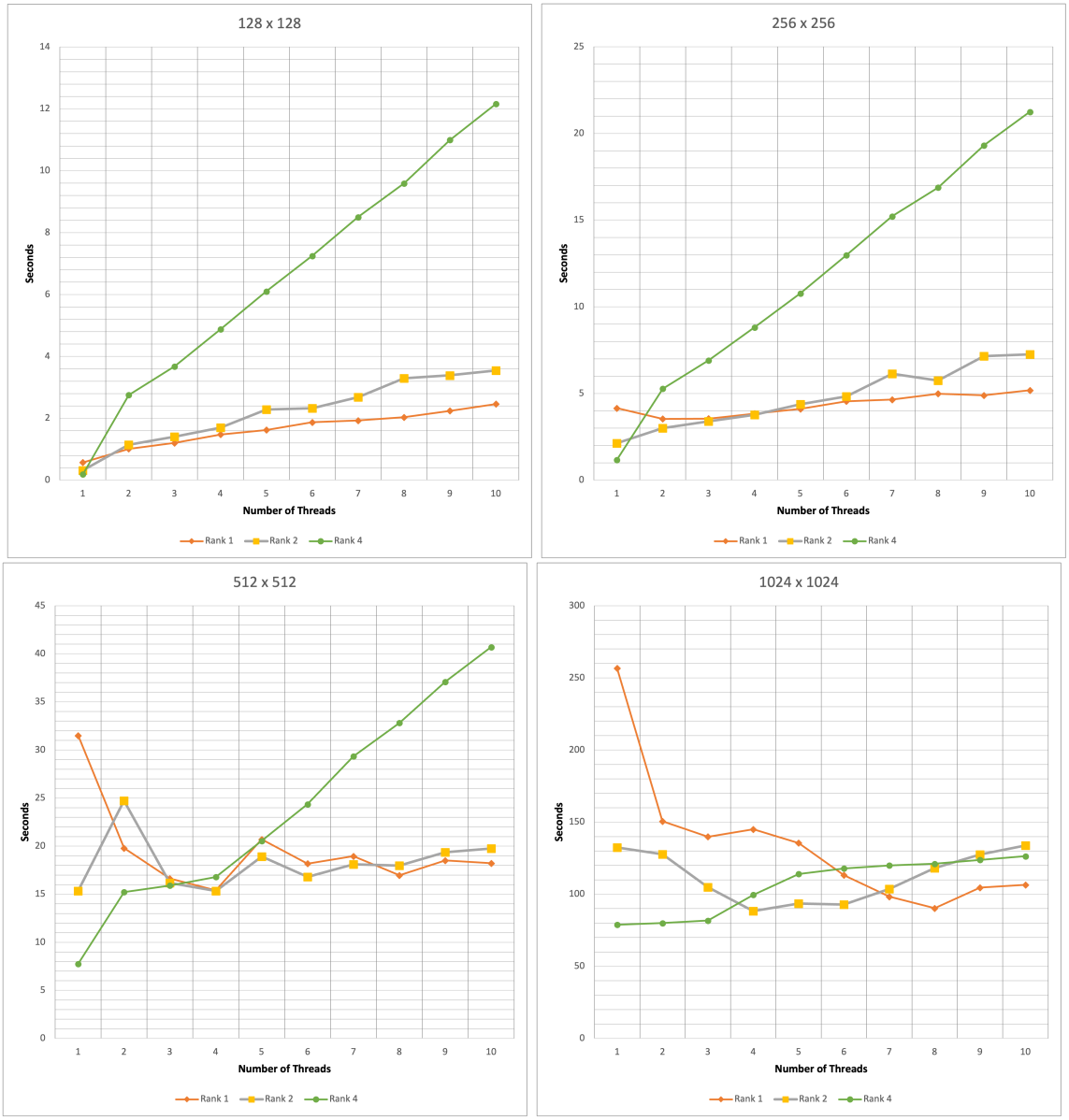
In the `linalg.cpp` file, linear algebra functions were modified to include collective operations using `MPI_Allreduce` for computing the dot product and norm. These modifications enhance the efficiency of linear algebra computations by introducing collective operations for global reductions, reducing the need for explicit communication. This optimization ensures a more synchronized and efficient parallel execution of linear algebra operations, improving overall performance.

4. Task 4 - Exchange ghost cells [45 Points]

In the `operators.cpp` file, efficient communication strategies were implemented for exchanging ghost cells in all directions (north, south, east, west). Non-blocking communication with `MPI_Isend` and `MPI_Irecv` were employed for overlap between computation and communication. Data was copied to send buffers before sending, and ghost cells were received before computing, minimizing idle time during communication.

5. Task 5 - Testing [20 Points]

Testing was conducted to assess the scalability of the implemented parallelization strategy. Grid sizes of **128x128**, **256x256**, **512x512**, and **1024x1024** were considered, with thread counts ranging from **1 to 10** and varying MPI ranks (**1, 2, 4**). Time-to-solution data was collected and analyzed to understand the impact of parallelization on performance.



5.1. Scalability Analysis

A strong scaling analysis was performed by plotting the time to solution against the number of threads for each grid size and MPI rank combination. The plots provide insights into how well the code scales with increasing computational resources. It is expected that, up to a certain point, increasing the number of ranks might lead to a decrease in the time to solution due to improved parallelism. However, beyond a certain point, communication overhead may counteract the benefits of increased parallelization.

This comprehensive testing and analysis provide valuable insights into the performance characteristics of the parallelized code under different conditions, aiding in the identification of optimal configurations for different computational resources.

5.1.1. Impact of Threads

As the number of threads increases, the execution time generally increases. This is likely due to increased overhead and contention for resources as more threads are utilized. However, the relationship is not strictly linear, and there might be a point where adding more threads does not lead to further improvements.

5.1.2. Impact of MPI Ranks

For a fixed number of threads, increasing the number of MPI ranks generally leads to an increase in execution time. This is expected because more communication overhead is introduced as the number of ranks grows. However, the impact may vary based on the grid size.

5.1.3. Impact of Grid Size

As the grid size increases, the execution time also increases. Larger grids require more computation, and the impact is more pronounced as the problem size grows.

5.1.4. Interaction Between Threads and Ranks

There are instances where increasing the number of MPI ranks results in significantly higher execution times, especially for larger grid sizes. This may indicate that the algorithm is not well-suited for a high degree of parallelism when combined with MPI, or there may be communication bottlenecks that are exacerbated with more ranks.

5.1.5. Optimal Configuration

The optimal configuration (in terms of both threads and ranks) depends on the specific characteristics of the algorithm, the problem size, and the architecture of the computing environment. Fine-tuning the combination of threads and ranks is crucial for achieving optimal performance.

5.2. Conclusion

In summary, achieving optimal performance in parallel computing involves a careful balance between computation and communication, and thorough experimentation is essential for understanding the complex interplay between threads, MPI ranks, and problem size.

The MPI parallelization of the diffusion equation solver has been successfully implemented, addressing various aspects such as domain decomposition, Cartesian topology creation, linear algebra function modifications, and efficient ghost cell exchange. The testing and scalability analysis provide valuable insights into the performance characteristics of the parallelized code under different conditions.

6. Task 6 - Quality of the Report [15 Points]

Additional notes and submission details

Submit the source code files (together with your used `Makefile`) in an archive file (tar, zip, etc.) and summarize your results and the observations for all exercises by writing an extended Latex report. Use the Latex template from the webpage and upload the Latex summary as a PDF to iCorsi.

- Your submission should be a gzipped tar archive, formatted like project_number_lastname_firstname.zip or project_number_lastname_firstname.tgz. It should contain:
  - all the source codes of your MPI solutions.

  - your write-up with your name project_number_lastname_firstname.pdf,

- Submit your .tgz through Icorsi.