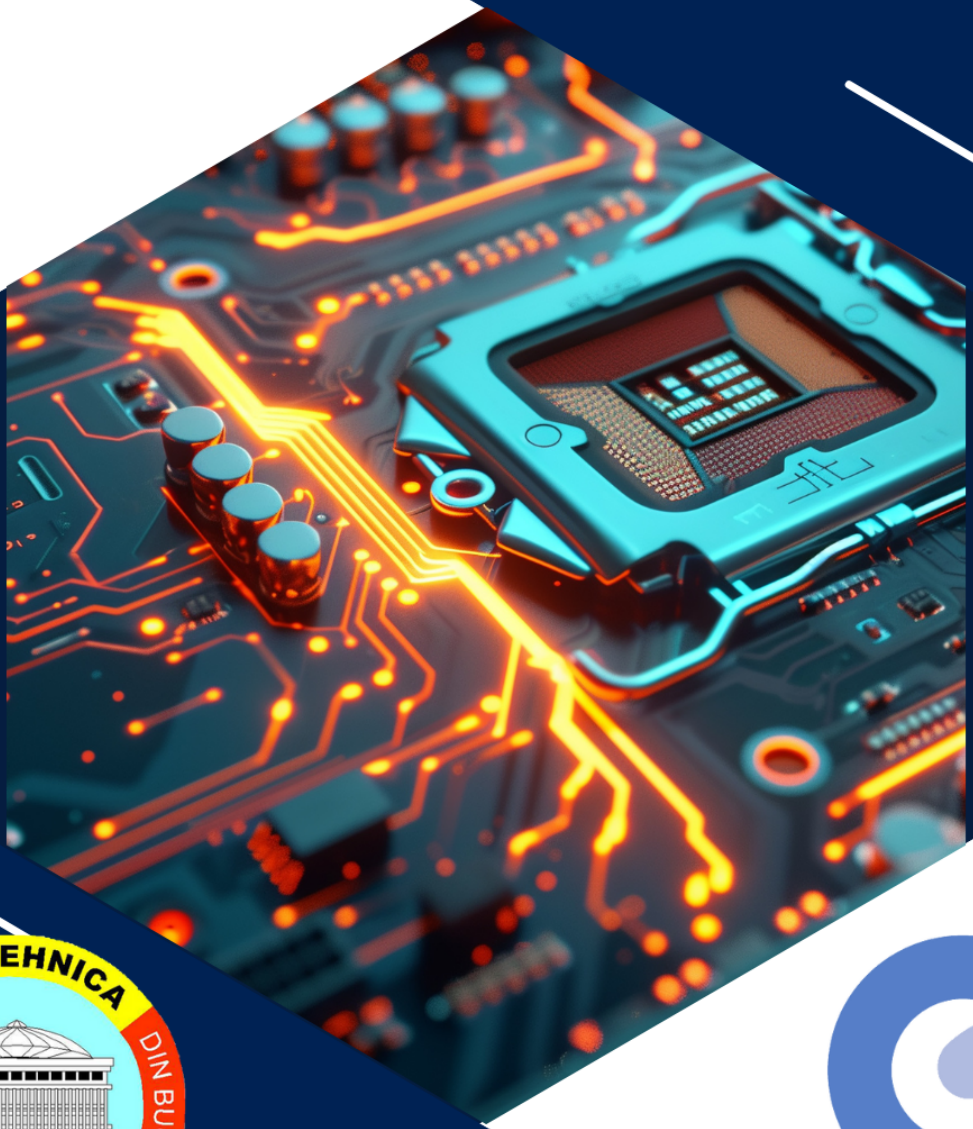
COLLABORATIVE CALCULATIONS

DUȚU ALIN CĂLIN



UNIVERSITATEA POLITEHNICA DIN BUCUREȘTI
1818

CONTENTS

1 INTRODUCTION

1.1 Context

In the era of artificial intelligence and machine learning, which is the trend of today's industry, the computational power of our computers plays a crucial role in terms of speed and efficiency. Since a query in ChatGPT is 15 times more energy-consuming than a Google search[1] it is important to look for efficient strategies that reduce the carbon footprint of such tools.

The MPI is a standard that includes point-to-point message-passing, collective communications, group and communicator concepts, process topologies, environmental management, process creation and management, one-sided communications, extended collective operations, external interfaces, I/O, some miscellaneous topics, and multiple tool interfaces that will increase the efficiency and the speed required to do a task.[2]

1.2 Objectives

This project aims to provide an overview of the MPI standard and serves as practice for testing the protocol capabilities. The following objectives are taken into consideration:

- Research the efficiency of the MPI protocol
- Create and develop one topology of processes
- Design the code with tolerance to failure

2 PROPOSED SOLUTION

2.1 Solution presentation

The project aims to implement a topology of processes that receives a vector as input after topology initialization and returns the vector with its elements doubled.

A visualization of the project's topology can be found below:

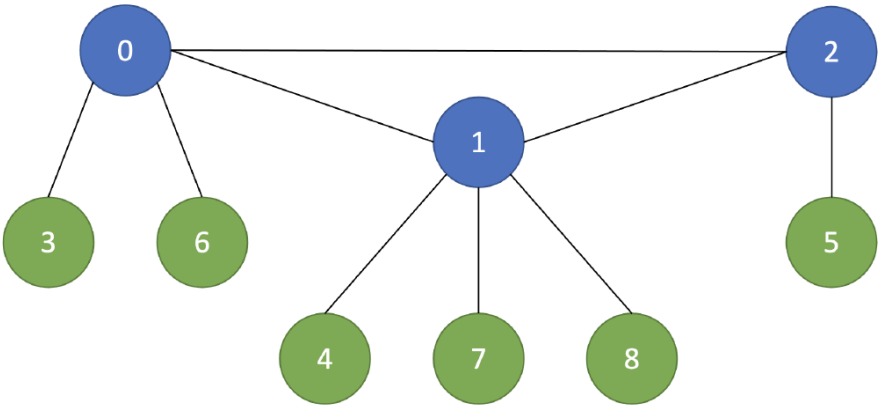


Figure 1: Topology

In this topology there are 2 types of entities:

- Coordinator - Receives the input, splits the tasks evenly and sends them to workers, receives back the result, and reconstructs to create the final solution. These are indicated with blue dots
- Worker - The entity that receives tasks from coordinators, executes them and returns the result. These are indicated with green dots

3 IMPLEMENTATION DETAILS

3.1 Proposed implementation

3.1.1 MPI Initialization

This program starts with the initialization of the topology. First, the program initializes the MPI protocol which consists of the following:

- The MPI configurations are being set
- The processes are being set
- The ranks for all processes are being set and provided

3.1.2 Topology formation

Next, each coordinator reads their topology and then sends it to the other coordinators, in which case, we consider 2 scenarios.

The first scenario is when the complete topology is up without any problems in which case we consider a ring-like topology.

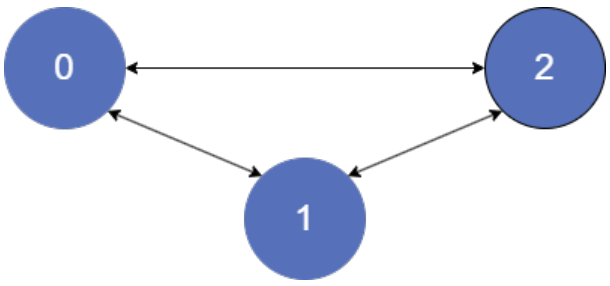


Figure 2: Coordinator's communication line - First scenario

The coordinators send their topology to the next neighbor, in the following manner: $0 \rightarrow 2$, $2 \rightarrow 1$, $1 \rightarrow 0$. By doing that, one coordinator will know its topology and the one from the previous neighbor. However, the same coordinator doesn't know the topology of the next neighbor, so the next neighbor sends its topology to the current coordinator, and the following topologies are sent: $0 \rightarrow 1$, $1 \rightarrow 2$, $2 \rightarrow 0$.

The second scenario is when the complete topology is up, but the connection between coordinators 0 and 1 is down. In this case, coordinator 2 acts as an intermediary to make coordinator 0 reach coordinator 1.

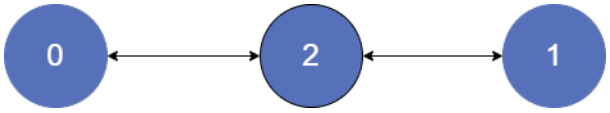


Figure 3: Coordinator's communication line - Second scenario

In this scenario, coordinator 2 receives the topologies for 0 and 1 meaning that coordinator 2 will know the whole topology. Next, 2 will send the coordinator 0's topology to 1 and coordinator 1's topology to 0.

3.1.3 Task execution

In the first scenario, coordinator 0 reads the whole vector, grabs its part, and sends the corresponding parts to coordinators 1 and 2.

Next, the coordinators split their parts into same-sized tasks and sent them to their workers using the scatter function.

The worker receives the task, which consists of a part of the vector and doubles each element. When one worker is done it sends the result back to its coordinator. The coordinators wait for their workers to finish, receive their output, and assemble the results using the gather function.

In the end, coordinators 1 and 2 send their parts to coordinator 0 which will assemble the final result and write it in the output.

The difference between the first and the second scenario is the transmission of parts between coordinators 0 and 1 will always go through coordinator 2 as it is the only communication line between 0 and 1.

3.2 Testing

For testing the proposed solution, the following methods have been used:

- Checker - Used for testing general cases
- Manual testing - Used for testing corner cases

4 EVALUATION

When evaluating the MPI protocol, a great metric should measure the time it takes over a multi-threaded execution of the program and a sequential execution. For this purpose, two more programs have been developed to perform the same task, but one does it sequentially and the other uses threads. The time is calculated based on the average of multiple runs. These are the results:

N	Sequential	Pthreads	MPI
10000	0,003s	0,004s	0,582s
50000	0,035s	0,014s	0,580s
100000	0,045s	0,040s	0,583s
1000000	0,564s	0,488s	0,835s

Table 1: Time execution

The performance of the MPI protocol seems abysmal compared with Pthreads and Sequential running, however, one interesting thing is when the initialization times are not considered, then the results tell a different story.

N	Sequential	Pthreads	MPI
10000	0,003s	0,002s	0,274s
50000	0,034s	0,012s	0,273s
100000	0,043s	0,034s	0,276s
1000000	0,561s	0,447s	0,527s

Table 2: Time execution without initialization

This table tells the story of MPI that initialization takes longer than other methods and its efficiency can be noticed in highly computational tasks as the differences in time between medium-sized and big-sized inputs are much smaller in MPI than the other methods.

The reason behind longer times on smaller inputs seems to be the design of the method itself, as it requires some of the processes to coordinate workers rather than contribute to solving the task. The trade-off of this mechanic is the workload of splitting tasks which is being taken considerably by the coordinators which results in an equitable workload between the main process and the coordinators.

5 CONCLUSIONS

In conclusion, this program aims to provide an overview of the MPI protocol by developing a simple program that doubles the elements of a vector. The analysis in the evaluation phase provided a clear view of when MPI excels and when it is not recommended to use. Overall, MPI is a protocol designed to significantly reduce the time it takes to compute complex and computationally demanding tasks.

BIBLIOGRAPHY

[1] Wim Vanderbauwhede. Emissions from ChatGPT are much higher than from conventional search. https://limited.systems/articles/google-search-vs-chatgpt-emissions/. Latest access: 25.08.2024.

[2] Message Passing Interface Forum. *MPI: A Message-Passing Interface Standard Version 4.1*, November 2023.

[3] APD team @ Polytechnic University of Bucharest. Calcule colaborative in sisteme distribuite. https://archive.curs.upb.ro/2021/pluginfile.php/439968/mod_resource/content/11/Tema%203%20-%20Enunt.pdf. Latest access: 28.08.2024.