# CS633 - Assignment-2

## Group - 21

Deven Joshi - 210326 (devenaj21@iitk.ac.in)
Shishir Gujarey - 210977 (shishirg21@iitk.ac.in)
Shreyash Kumar - 211009 (shreyashk21@iitk.ac.in)

August 30, 2024

---

## Introduction

The assignment continues on the front of the **halo exchange** paradigm, incorporating **topology-aware hierarchical communication** and comparing the performance of this heuristic with previously obtained timings.

---

## Initialization

- **buf:** local matrix for each process; initialized with random seed at the beginning.
- **send_dirn_2:** the four arrays are used to send the packed arrays for 9-point stencil
- **recv_dirn_2:** the four arrays are used to receive the packed arrays for 9-point stencil
- **gather_buf_dirn:** array to receive rows' data from processes of the same node
- **recv_gather_dirn:** buffers for receiving data for the inter-node commmunication
- **arr_dirn**: the four arrays are used to unpack the received arrays and are finally used for average computation by the cells(except for interior cells)

---

## Communication

The following calls have been used to facilitate communication within the MPI environment:

- **MPI_ISend:** non-blocking function used to initiate send operation between processes
- **MPI_Recv:** blocking function used to receive the data sent by another process
- **MPI_Barrier:** blocking function used to synchronize all processes in a communicator
- **MPI_Reduce:** collective operation used to used to cumulate data from all processes

- **MPI_Gather:** collective call used to accumulate data from all processes in the intra comm to the leader
- **MPI_Scatter:** collective call used to distribute data from from leader process to all processes in the intra comm

---

# Code Explanation

## 4.1 Without Leader

- **Initialization:** The code parses command-line arguments, including the number of processes in each dimension (Px), the grid size (N), the number of time steps (steps), the random seed (seed), and the stencil type (stencil). Each process then initializes its sub-domain with random values using the provided seed. Processes then determine whether they have neighboring processes to communicate with using flags.
- **Halo-Exchange:** Each process packs its outlying data in the *send_dirn* using *MPI_Pack*. Packed data is then sent to the corresponding process where the data is required. The packed data is then received in the *Recv_dirn* using *MPI_Recv*. The received data is then unpacked into the *arr_dirn* using *MPI_Unpack*.
- **Computation:** Values in the cells are averaged out using the 9-point stencil techniques. Various edge cases are handled using flags, taking care of the denominators of the average.
- **Reporting:** The code calculates the execution time for 10 time steps of the aforementioned algorithm for each process and reports the maximum time using *MPI_Reduce*. Note that this is the overall execution time for the problem statement.

## 4.2 With Leader

- The side-wise communications and the computation remains the same as before.
- For the top and bottom communications, we make use of the topology of the system. We make separate communicators for each node. Instead of using a large number of inter-node communications, we use *MPI_Gather* to send the required data to the leader process of each communicator, which shares the data with the leader of the receiver process via the global comm, and the receiving leader process then communicates the data with the intra comm using *MPI_Scatter*

---

# Optimizations

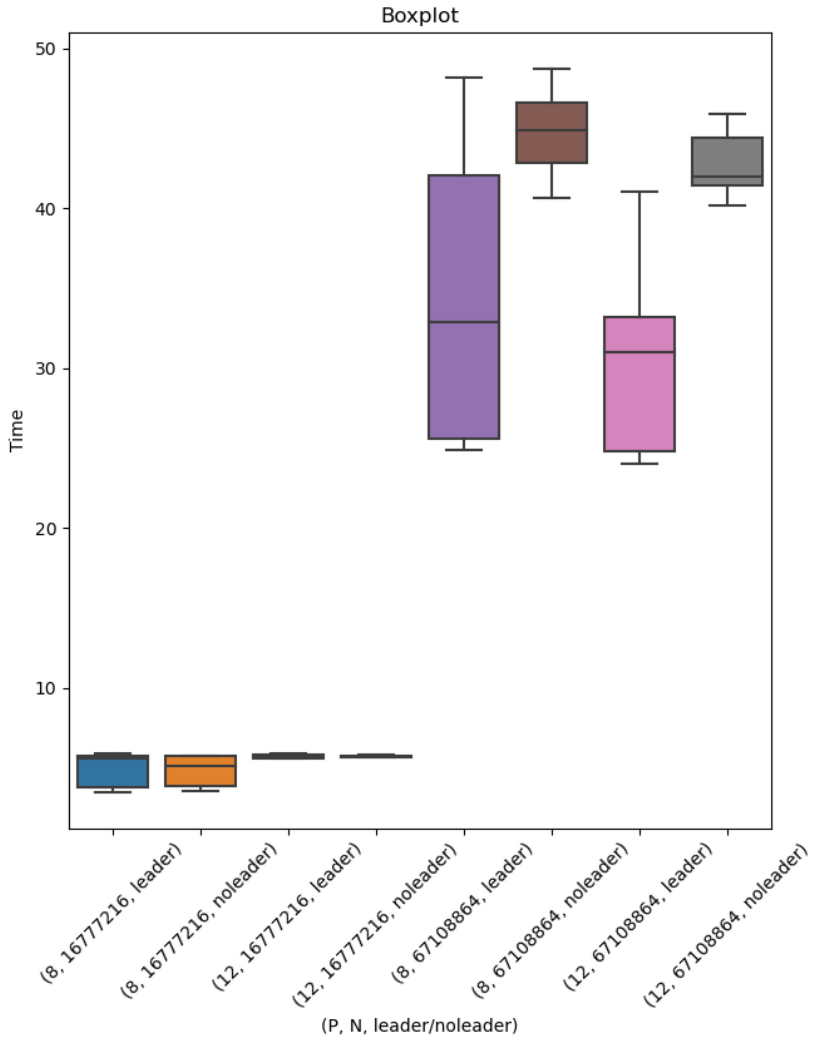
- We have used *MPI_ISend* instead of *MPI_Send*, which allows for asynchronous communication and overlapping communication with computation, potentially improving overall performance by enabling the sending process to perform other tasks while the message is being sent.
- For the 9-point stencil computation, we have packed two arrays and sent them as one, reducing the number of communications between the processes, thereby reducing process execution time.
- In the *With Leader* segment, we have used topology aware hierarchical communication, reducing the number of inter-node communications and the overall time taken by the algorithm

# Timing Data

On executing the above code three times, we get the following output:

| P | N | Leader/ No Leader | Execution Time | | | | |
|---|---|---|---|---|---|---|---|
| | | | Time 1 | Time 2 | Time 3 | Time 4 | Time 5 |
| 8 | 16777216 | Leader | 3.479132 | 3.770416 | 5.552796 | 5.880188 | 5.74752 |
| 8 | 16777216 | No Leader | 3.534736 | 3.864568 | 5.078565 | 5.766754 | 5.751837 |
| 12 | 16777216 | Leader | 5.553156 | 5.562327 | 5.914341 | 5.751143 | 5.798096 |
| 12 | 16777216 | No Leader | 5.621475 | 5.64281 | 5.662887 | 5.748588 | 5.806583 |
| 8 | 67108864 | Leader | 25.577896 | 24.845275 | 48.202177 | 32.889295 | 42.063179 |
| 8 | 67108864 | No Leader | 42.877243 | 40.612666 | 44.860047 | 46.607665 | 48.723078 |
| 12 | 67108864 | Leader | 23.984294 | 24.778516 | 41.038509 | 33.159878 | 31.034048 |
| 12 | 67108864 | No Leader | 41.948859 | 40.184045 | 44.393407 | 41.398799 | 45.932362 |

# Plot and Observations



From the above plot, we can make the following observations:

- There is a huge difference between the time taken for N=16777216 and N=67108864. Hence, we can say that the time taken largely depends on the input data size (number of data points per process).
- For the same N and P, we can see that the time taken for data exchange with leader ranks is lesser than the time taken without leader ranks. While this difference is not that apparent for N=16777216, it becomes more evident for N=67108864. Hence, having leader ranks reduces the time taken for data exchange by reducing the number of inter-node communications. As the datasize increases, this reduction in the number of inter-node communications becomes an increasingly significant factor for the time taken for data exchange.

---

## Contributions

- Deven Joshi - 33.33%
- Shishir Gujarey - 33.33%
- Shreyash Kumar - 33.33%