CS633: Group 1: Assignment 1

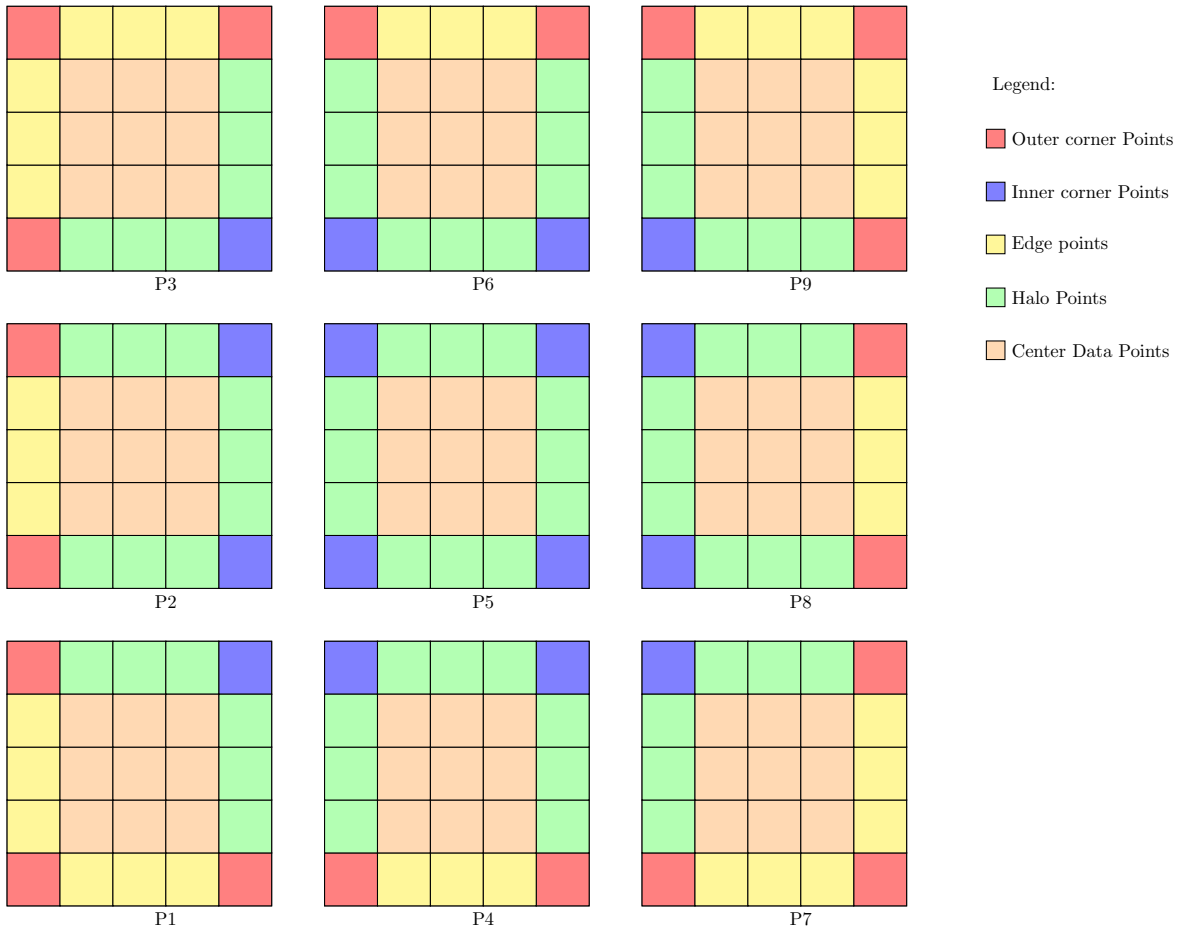
Manasvi Jain - 210581 - manasvij21@iitk.ac.in
Sarthak Kalankar - 210935 - sarthakk21@iitk.ac.in
Saugat Kannojia - 210943 - saugatk21@iitk.ac.in

March 2024

Explanation of Code

- We took the values of the various parameters from the arguments and initialised the data matrix with random values.
- We checked for existing neighbours by making flags for them.
- For the boundary processes, we communicated the desired data points to them from the neighbours(if applicable).
- For the above communication, we used the following in the given order:
  1. MPI_Pack for packing the boundary data points.
  2. MPI_Isend for sending the packed data.
  3. MPI_Recv for receiving the packed data.
  4. MPI_Unpack to unpack the received packed data.
- Handled **5-point** and **9-point** stencils accordingly
- Handled the corner cases of the processes containing non-existing neighbours

We have plotted the below figure to show how exactly the updation in the data matrix for a 5-point stencil happens and the data filling has been done in the following patterns which are shown with a different colour for each pattern.



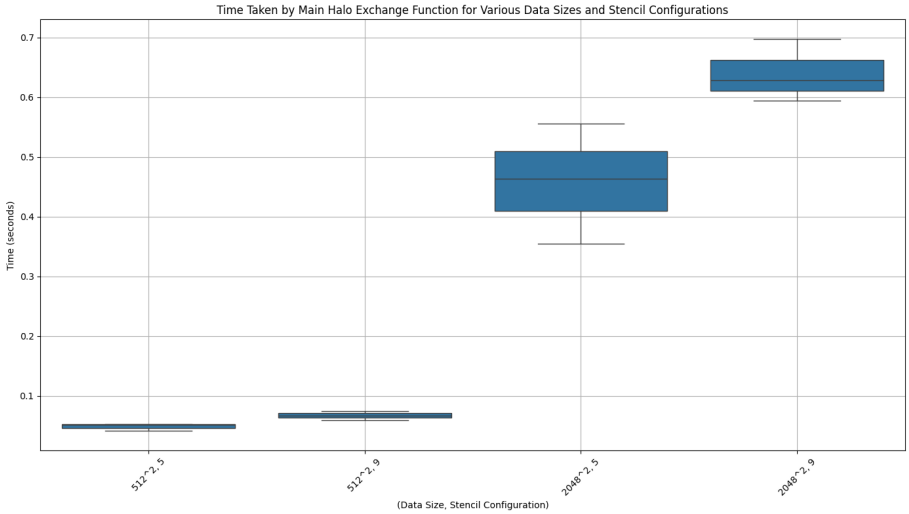
Performance Comparison

Observation 1:

Changing the stencil size from 5-point stencil to 9-point stencil **increases the total time taken** for the complete Halo Exchange for all the time steps, because the **communication time increases** as there are double the number of rows/columns to be communicated between two processes.

Observation 2:

Increasing the data size increases the total time taken because more amount of data points get packed and communicated as well as more computation time.



Optimizations

We have used Isend/Recv with MPI_Wait before unpacking to ensure non-blocking message passing takes place accurately which gives better safety and performance over blocking calls.