CS633: Assignment 2

Shrilakshmi S K (211012) Prashant Sharma (210750) Venkaat Balaje (211159)

April 15, 2024

1 Augmentations to Assignment 1 Code

- The first process in each row is the leader rank of the subcommunicator. Here each row is a subcommunicator. Extra memory allocations for leader rank are made through arrays `up_recv_leader` and `down_recv_leader`.
- Created subcommunicators (each row has its own subcommunicator) `row_comm` using `MPI_Group_incl` method.
- We have changed the up-down communication step. Instead of sending and receiving data from up/down processes, we have gathered all the halo data at leader rank (root) and sent/received data only from/by leader ranks and then scattered the data from leader rank (root) to all processes in row subcommunicator.

2 Optimizations

- We allocate space only if the neighbouring processes exist.
- We observed that the average time is lesser in Right Sends followed by Left Sends Algorithm compared to Odd Even Algorithm. Hence we choose the former algorithm to communicate in the final code.
- We have also not used `MPI_Unpack` after `MPI_Recv`. We have adjusted the indices accordingly in stencil computation part to eliminate the use of `MPI_Unpack`, hence removing the copying overhead and optimize the code further.
- We have created subcommunicators for each row to `MPI_Gather` and `MPI_Scatter` the data instead of using `MPI_Send` and `MPI_Recv`, as former use optimal collective communication algorithms than latter.

3 Performance Observations

We recorded 5 observations on HPC2010 to plot the boxplot as follows.

	N = 4096*4096 P = 8	N = 4096*4096 P = 12	N = 8192*8192 P = 8	N = 8192*8192 P = 12
without leader	4.341593	9.207783	18.443207	35.908174
with leader	4.286254	8.808371	18.039222	34.893481
without leader	4.343023	9.222252	18.645009	35.470010
with leader	4.288961	8.817914	17.848242	34.926677
without leader	4.332190	9.190236	18.132423	36.285214
with leader	4.301278	8.803493	18.835474	35.838921
without leader	4.401983	9.203429	19.542931	35.398143
with leader	4.582103	9.245412	18.949313	35.849104
without leader	4.289231	9.164815	18.389274	34.438923
with leader	4.194273	8.739609	17.348453	35.243045

Table 1: Time in (s) for P_x = 4, seed = 101, num_time_steps = 10

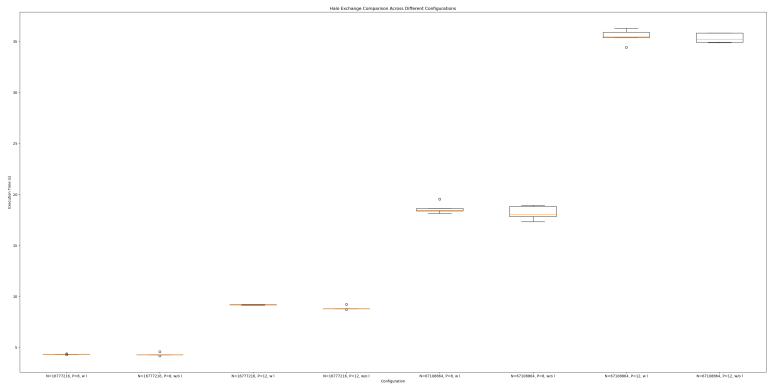

Figure 1: Boxplot of Halo Exchange Across Different Configurations

3.1 Complexities

		Maximum communication data size sent or received by a single process (bytes)	Maximum number of additions to calculate sum for stencil computation by a single process
1	`N = 4096*4096, P = 8`	32768	32768
2	`N = 4096*4096, P = 12`	32768	32768
3	`N = 8192*8192, P = 8`	65536	65536
4	`N = 8192*8192, P = 12`	65536	65536

Table 2: Stencil communication & computation complexity of a single process

3.2 Observations

- **Difference between observations on Prutor and HPC:** Prutor gave similar times for a given `N` for `P = 8` and `P = 12`. For `N = 4096*4096`, the times were 5.691562 and 5.786639 respectively for `P = 8` and `P = 12`. But as shown in the table, the times almost double for same `N` from `P = 8` to `P = 12`. As the table in Section 3.1 indicates, the complexities for `P = 8` and `P = 12` for same `N` are same. This doubling of time could be attributed to the fact that nodes are increasing from 2 (`P = 8`) to 3 (`P = 12`) and the middle row in the latter case has to communicate both up and down unlike the former case. This applies for `N = 8192*8192` as well.

- Both communication and computation complexities increase by 200% from Case 1 to Case 3. Despite this, the time increases by almost 400-500%. The reason could be that, both communication and computation requires memory access of `data` matrix. These huge number of memory accesses may not be linearly scalable as nodes have different level of cache (NUMA architecture). Other reasons also could be load imbalance (This is particularly important as the data initialization depends on i and j which in turn depends on N) and resource contention.

One more reason could be that all processes do not synchronize after an iteration and hence in next iteration, the earlier processes are blocked by `MPI_Send`, `MPI_Recv`, `MPI_Scatter` and `MPI_Gather` calls. This could propagate more delay in further iterations.

- The time with leader is lesser than time without leader in most cases. This is attributable to the fact that we are doing less inter-node communications in case of with leader. We are only doing 2 inter-node communications in `P = 8` and 4 inter-node communications in `P = 12` with leader as opposed to 8 inter-node communications in `P = 8` and 16 inter-node communications in `P = 12` without leader.

4 Group Details and Contribution

Group: 43
Shrilakshmi S K - 211012 - shrilakshm21@iitk.ac.in - 40%
Prashant Sharma - 210750 - kprashant21@iitk.ac.in - 33%
Venkaat Balaje - 211159 - venkaatb21@iitk.ac.in - 27%