Assignment 1 (100 marks)

Write a program to perform halo exchange (refer to Lecture 5) with neighbouring processes. The boundary processes need not exchange data in case of non-existing neighbours (for e.g. consider the process grid below, process 0 does not have a top and left neighbour, process 1 has three neighbours, and so on).

0	1	2
3	4	5
6	7	8

Assume a square number of processes for this problem. Every process owns a subdomain of data (doubles). For example, in the figure below, every process has 16 data points and 4 halo regions (shown in green). The halo regions need to be exchanged with four neighbouring processes (top, bottom, right, left) at every time step. Every process performs stencil computation followed by communication for the boundary points/cells. Next time step value at a point/cell P is computed as the average of the four neighbouring points/cells of P (left, right, top, bottom). E.g.
value (P, t+1) = [value (P_left, t) + value (P_right, t) + value (P_top, t) + value (P_bottom, t)] / 4.

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Compare the performance of data exchange using the following three methods:

(1) Multiple MPI_Sends, each MPI_Send transmits only 1 element (1 double in this case).
(2) MPI_Pack/MPI_Unpack and MPI_Send/MPI_Recv to transmit multiple elements at a time
(3) MPI_Send/Recv using MPI derived datatypes (contiguous, vector, ... whichever is suitable)

Perform the above experiments for the following configurations.

for execution in 1 to 5 // repeat each configuration 5 times
 for P (number of processes) in 16, 36, 49, 64
  for N (data points per process) in 16^2, 32^2, 64^2, 128^2, 256^2, 512^2, 1024^2
   mpirun –np P –f hostfile ./halo N <num_time_steps>

Execute with num_time_steps = 50, i.e. computation and communication is done for 50 time steps. Assume 2D square decomposition of P and N. Every process must randomly initialize the N data points that it owns at time step 0.

Use 8 processes per node. Time the entire data exchange for all the time steps per method, and output the times in the following format (one per line) for the three methods listed above (in the same order). Here, the times are as returned by the MPI_Wtime() function.
<time1>
<time2>
<time3>

Plot the time (in seconds) for each data size per method per process count. Use boxplots (from the 5 executions) for every data point in a plot. Plot the data corresponding to a process count in a separate file, i.e. submit 4 plot files. Plot the times for each N for the three methods in a file. Time in seconds (y-axis) and N (x-axis). You should report the total time taken by the main data transfer function, excluding initialization etc.

Execution and submission instructions

Create 'Assignment1' directory on git. It should necessarily contain the source code ('src.c'), 'Makefile', 'readme.pdf', job script ('run.py' or 'run.sh'), plot script and plots. The job script should compile your code using the "make" command. The job script should run all the configurations as specified above. I should be able to run the job script to execute your code (all configurations). The job script execution must generate the output data* files. Name your plots 'plotP.jpg' for each P. The 'readme.pdf' should contain a simple explanation of your code, your observations regarding performance from the plots, the 4 plots and any issues that you may have faced with the code, experimental setup etc. You must use a script that generates machinefile/hostfile on-the-fly based on the node status so that your jobs never fail. Optionally, you may use NodeAllocator to generate hostfile. The job script must generate the hostfile before the runs.

Suggestions

- Follow the above instructions carefully
- Use git.cse.iitk.ac.in as a real git repo
- Document your code

Due date: 25-02-2021 (There will be NO extensions)