README

How to execute the code

The Assignment1 directory contains the following files:

1. src.c
2. run.sh
3. Makefile
4. plot.py

The whole assignment can be executed by the script file "run.sh". For executing the code, make sure the path to mpich is pointed to user's installation directory and use the following commands to execute:

```
$ chmod 777 run.sh
$ ./run.sh
```

The run.sh script will create a folder named "data_files" inside which all data files will be generated. The plot script will make use of these data files and will generate the plots named "plotP.jpg" for each P. If it is required to run the code again, please use below command to remove any old versions of results:

```
$ make clean
```

Explanation for the code

The src.c is the soruce code which performs halo exchange and stencil computation operations in three different methods. It will take arguements N (number of data points per process), time steps, mode

Value of mode will be as below:

1. mode = 1 for for communication using multiple MPI Sends and Recvs
2. mode = 2 for communication using MPI Pack and Unpack
3. mode = 3 for for communication using MPI Derived data types.

For multiple MPI Sends and Recvs, we received individually sent elements into four buffers per process. These buffers store the elements sent by four neighbouring processes for stencil computation.

For MPI Pack and Unpack, we packed the data from sender side into a 1D array and sent to the neighbouring processes. At the receiver side, first we received the data into a temporary array and unpacked it into four 1-D buffers for each process. Data will be read from these buffers for stencil computation by each process.

For MPI Derived datatypes, we used MPI Vector datatypes to pack the data to be sent and at the receiver side we received data into four buffers. By using derived datatype, we don't need to copy data into temporary buffers for sending.

To allocate host nodes on fly we used Node allocator project which allocates the nodes based on their current loads.

Observations on results

1. We can observe that for any of the P value, as N (data points per process) increases, the execution time also increases for all three methods of data exchange. However this change is exponential when multiple MPI Sends and Recvs are used for communication. This behaviour is as expected. Because as we increase N, more data is to be exchanged between the processes. Hence increasing the time drastically.
2. For any of the P value, we can observe that time taken for multiple MPI Sends and Recvs is larger than MPI Pack Unpack and MPI derived datatypes.
3. Time taken by using MPI Pack, Unpack and by using MPI derived datatypes for all values of P is in the same range and the rate of increase in execution time is similar as we increase N. However for large values of N, it is observed that time for MPI Pack and Unpack is slightly larger than time for MPI derived datatypes since, there is copying overhead involved with MPI Pack Unpack.
4. We can see that, for some instances of P and N, the width of box plot is large. This can be due to impact of congestion in network acorss different executions particulary when P and N values are larger.