CS633A Parallel Computing - Assignment 1

24 February 2021

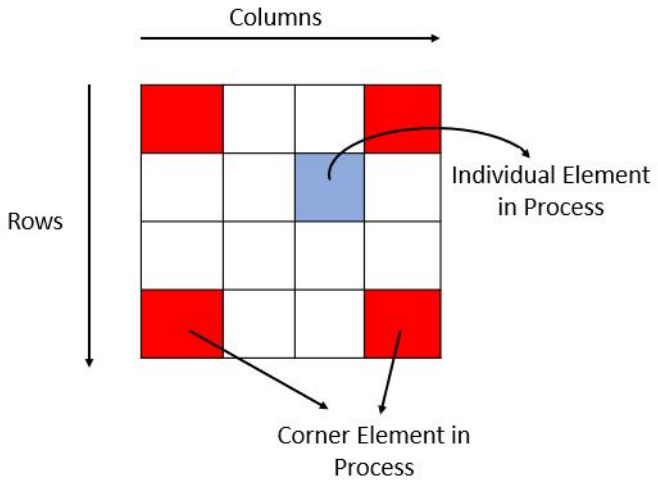
1 How to use

Run the **bash run.sh** command.
Note: NodeAllocator may take some time to run It simply does ssh to check the availability of the node and prioritize the node having lesser number of users.

2 Program Documentation

2.1 Process and Inside element

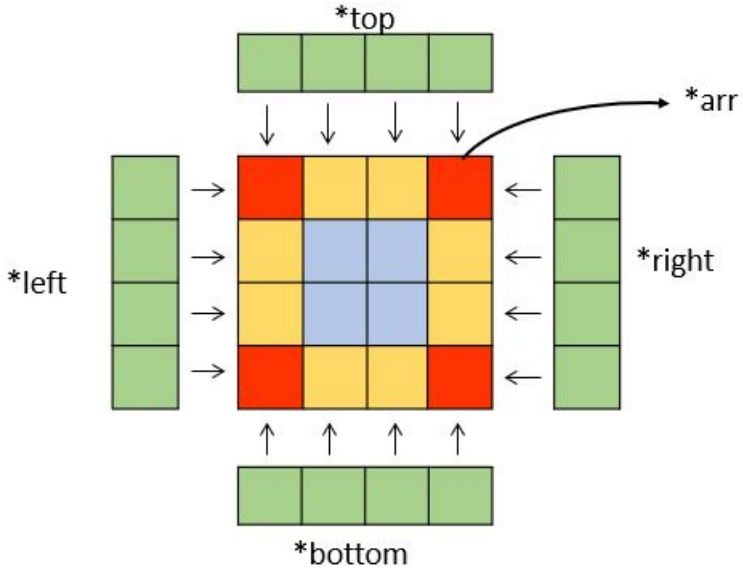
Every Process has elements inside it, every element have different (row, column) pair.



Every process is represented by structure which contains information of current process and whole Stencil:

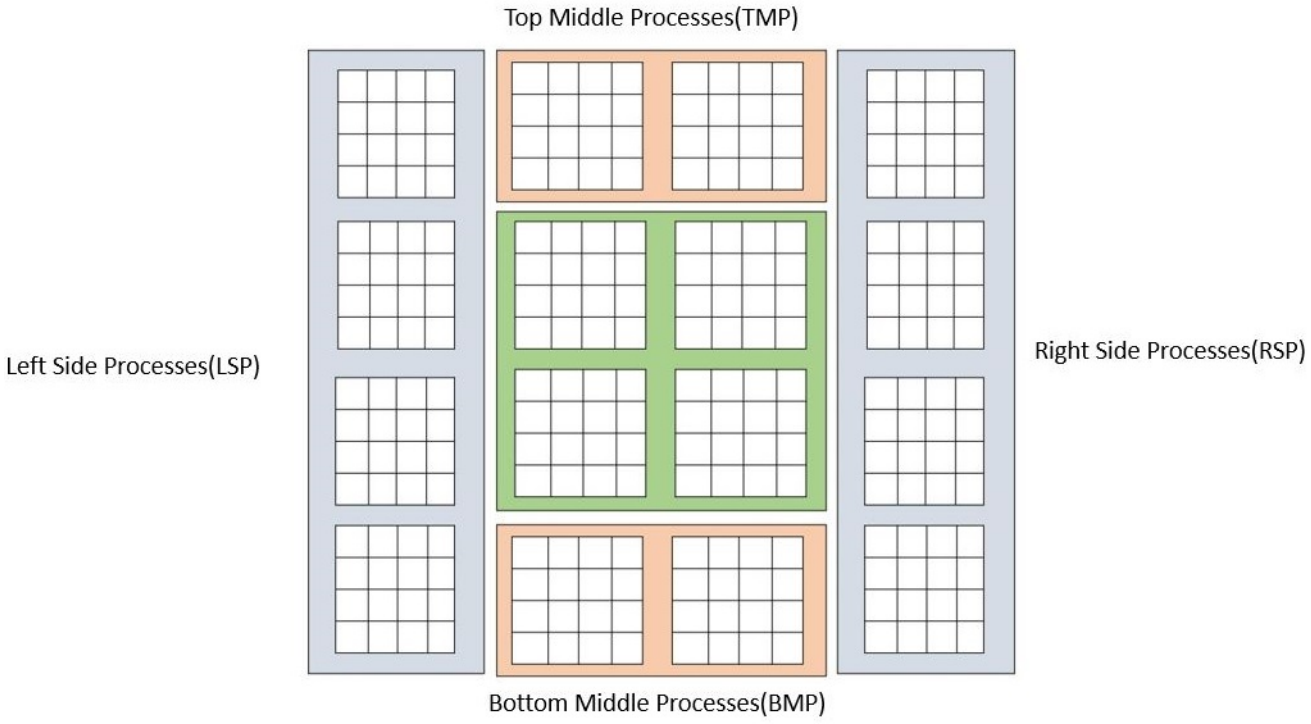
```
typedef struct data{
    int myrank;          //Rank of Process
    int size;            //Total Number of Process
    int sq;              //Processes Per Row
    int row;             //No. of Rows in each Process
    int col;             //No. of Columns in each Process
    double* arr;         //Process Array
    double* newarr;      //Temporary Array to Store intermediate values while calculation
    double* left;        //Stores Data Came From process which is Left to current process
    double* right;       //Stores Data Came From process which is Right to current process
    double* top;         //Stores Data Came From process which is Top to current process
    double* bottom;      //Stores Data Came From process which is Bottom to current process
}processData;
```

The Data is transferred to *top, *bottom, *left and *right vector from the neighbouring processes before calculation takes place. and during calculation the data is taken from these vectors as a shadow data of neighbouring processes.



2.2 Stencil

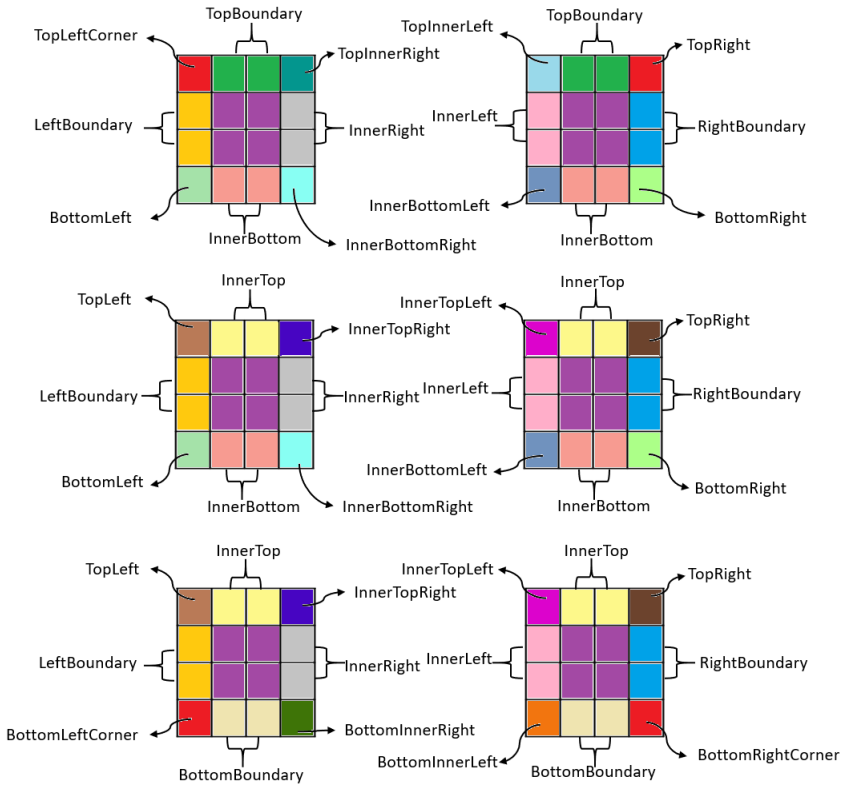
Stencil Consists of arrays of processes. Various processes inside stencil consists of different number of neighbouring processes and at differnet location. Hence we have divided processes based on their location in stencil.



The Above process map will be used while going through Calculation Part in the program. Processes inside Green box will fall into Middle process.

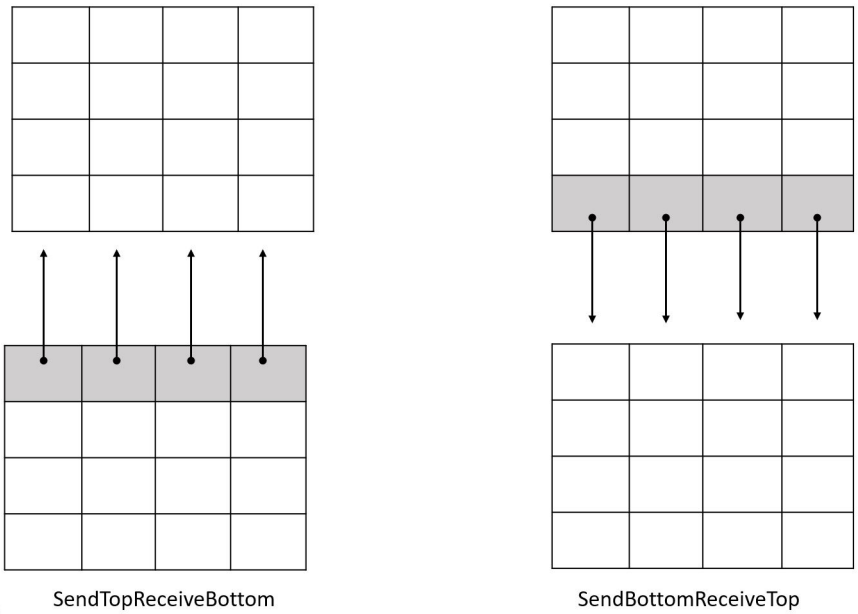
2.3 Element Function Mapping

Every element inside process needs different calculation. Hence we had made new function for every new location type. The following mapping will be helpful during calculation phase.

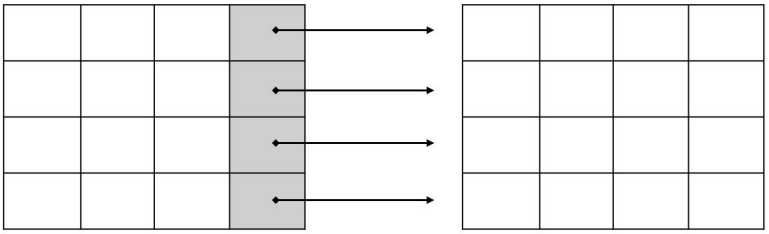


2.4 Transfer

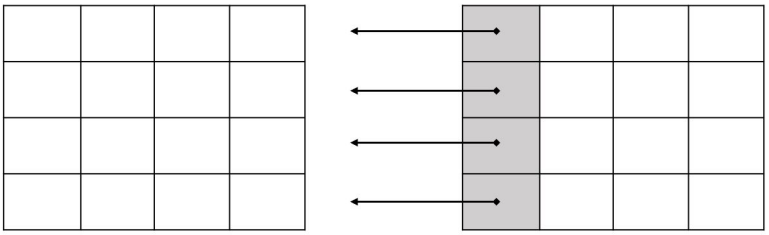
Transfer will be of 4 types as shown below.



SendTopReceiveBottom: The current process Will send its top row to process above it. And the transferred elements will be stored in bottom* vector of above process.
SendBottomReceiveTop: The current process Will send its bottom row to process below it. And the transferred row will be stored in top* vector of below process.



SendRightReceiveLeft



SendLeftReceiveRight

SendRightReceiveLeft: The current process Will send its right column to process right to it. And the transferred elements will be stored in left* vector of right process.
SendleftReceiveRight: The current process Will send its left column to process left to it. And the transferred row will be stored in right* vector of right process.

2.5 Graph Ploting

The time taken by the processess are saved in three file DataNormalversion.txt, DataVectorversion.txt and DataPackversion.txt. This data is processed by the PlotGraphs.py which plots the 4 box plots using seaborn library.

2.6 Program Skeleton

The program consists of various modules e.g. Transfer Normally, Transfer vector, Calculation of Left Side process etc.

- **main()**
  - Transfer
    - Normal
      - Tag Generator: There are multiple number of transfer having same send and recv pairs.
      - SendLeftReceiveRight()
      - SendRightreceiveLeft()
      - SendTopReceiveBottom()
      - SendBottomReceiveTop()
    - Pack
      - SendLeftReceiveRight()
      - SendRightreceiveLeft()
      - SendTopReceiveBottom()
      - SendBottomReceiveTop()
    - Vector
      - SendLeftReceiveRight()
      - SendRightreceiveLeft()
      - SendTopReceiveBottom()
      - SendBottomReceiveTop()
  - Calculation: Following functions will call different methode() for every element inside it.
    - LSP(): Left Side Process
    - RSP(): Right Side Process
    - TMP(): Top Middle Process
    - BMP(): Bottom Middle Process
  - **methods()**

methods()

- Boundary Case: For Elements which are at boundary of Stencil
  - Corner Case
    - TopLeftCorner()
    - TopRightCorner()
    - BottomLeftCorner()
    - BottomRightCorner()
  - Elements in boundary But are adjacent to different processes
    - TopInnerRight()
    - BottomInnerRight()
    - TopInnerLeft()
    - BottomInnerLeft()
    - TopLeft()
    - BottomLeft()
    - TopRight()
    - BottomRight()
  - Elements at boundary but do not share any relation with other processes
    - LeftBoundary()
    - RightBoundary()
    - TopBoundary()
    - BottomBoundary()
- Non-Boundary Case: For Elements which are not at boundary of Stencil
  - Elements Having 2 Neighbouring Process
    - InnerBottomLeft()
    - InnerBottomRight()
    - InnerTopLeft()
    - InnerTopRight()
  - Elements having only one Neighbour
    - InnerTop()
    - InnerBottom()
    - InnerLeft()
    - InnerRight()
  - Elements do not have any neighbours
    - MiddleMiddle()

3 Observation

The Normal Mode always has the worst result, hence it is better to use Pack or Vector Mode to transfer data when sending in bulk. Also, the process sometimes crashes while using Normal Mode may be due to heavy traffic. The Vector mode and Packed Mode gave almost similar results, They both are kind of the same thing. There is no big change in runtime due to a change in the number of processes but we can see the drastic change in runtime while using normal mode if we increase process size(size of N).

4 Issues Faced

- In Normal mode there were multiple send-receive between two process hence it was also necessary to keep track of row and column of data item. Hence we used Tag generator which keeps value of row and column in it.
- It was tricky to plot Graphs inside csewsX.
- Sending and Receiving in Packed version was little bit difficult.