CS633 Parallel Computing
Assignment-3

Shrey Bhatt, Roll no - 20111060
Saksham Jain, Roll no - 20111053

Brief overview of the problem

We are given a file named tdata.csv which contains data of temperature readings spanning several years for several thousand stations where each station is identified by a latitude and longitude. Our task is to find -
(1) year-wise minimum temperatures across all the stations for each year
(2) global minimum across all stations and all years.

Code Execution, Explanation and Data Distribution Strategy

Code Execution

To execute the code, simply run the file *run.py* by using the command **python3 run.py**. After running this, required output file and plot **(as file plot.png)** will also be generated. Please note that currently, the job script runs the executions 5 times, which can be changed by editing the variable **numexecs**. Moreover, the input csv filename can also be modified by updating the variable **DataFileName**. Also, the output file generated by the **src.c** is **output.txt** and the overall output file is stored in **output_all.txt**. Moreover, the hostfile is generated using the similar script as that in the previous assignment using groups of cse cluster. Furthermore, a timeout is introduced of **9 seconds** to check for nodes that do not respond for long time.

Code Explanation

Initially, the process with rank 0 reads the data from file and stores it an array (1D of dimension Number Stations x Number Years). Arrays Latd, Longtd, temprdt are used to store latitude, longitude and temperature data given in the file respectively. After reading the data, MPI_Barrier() is called to synchronize all the processes so that the time of other processes waiting for rank 0 to read file is not included in the overall time.

After executing MPI_Barrier(), we record the beginning time in sttime variable. Some sub-communicators are created to perform optimization over the MPI calls. These sub-communicators are created to reduce networks call by creating a sub-communicator for processes residing in the same node **(nodewc)** and a leader process on every node **(nodelc)**.

Then, the parameter information such as Number of Stations and Number of Years are broadcasted using these sub-communicators by putting both variables in a single Bcast call.

Then, we perform the data distribution (refer to section - Data Distribution Strategy) based on the number of processes running and also the number of nodes involved. After distributing the data, every process finds the minimum values of temperature for each year and the overall

minimum temperature as well. After finding these local minima values, the global minima values are calculated by performing an MPI reduction for both year-wise and overall values. One thing to note is that, we perform the reduction of both year-wise and overall data in same call by appending the overall value at end. For reduction as well, we use the node-wise subcommunicators by first performing reduction over nodewc and then over nodelc. The end time is finally recorded in ettime.

Using sttime and ettime, we calculate the time taken by each process then we are finding out the maximum time that is taken by any process using MPI_Reduce() call. Finally, the minima data and the time taken is over-written into output.txt. The following functions have been constructed for intermediate tasks -

- int parseHostID(char* hname,int hlen) - This function converts char hostid into int hostid.
- void getNumYears(char* linestr,int* nyears) - This function is used to find the number of years for which temperature data is given.
- void getColumnYears(char* linestr,int *yearsd,int nyear) - This function retrieves the data of years from column header.

Data Distribution Strategy

The data distribution strategy for the program involves distribution the rows of the data among processes. Every process works on every year. Since, the number of rows need not be divisible to number of processes, the array size is fixed such that the number of rows/stations is divisible by number of processes (adding some extra zero-rows at end so that the division is rounded of to the ceiling value). This is done in order to avoid vector variants like Scatterv, since they require calculation of count and displacement arrays. But by using the above size, the data is distributed using simple Scatter. Although this could lead to slight imbalance of data in the actual (excluding the pseudo) rows, it is very low compared to data volume since the number of station is very high compared to number of process. Once, the data is received, it is also taken care that any process does not consider the pseudo-rows by calculating the end of original data using the number of Stations and the number of processes. Also, note that the Scatter is done initially over **nodelc** and then over **nodewc**.

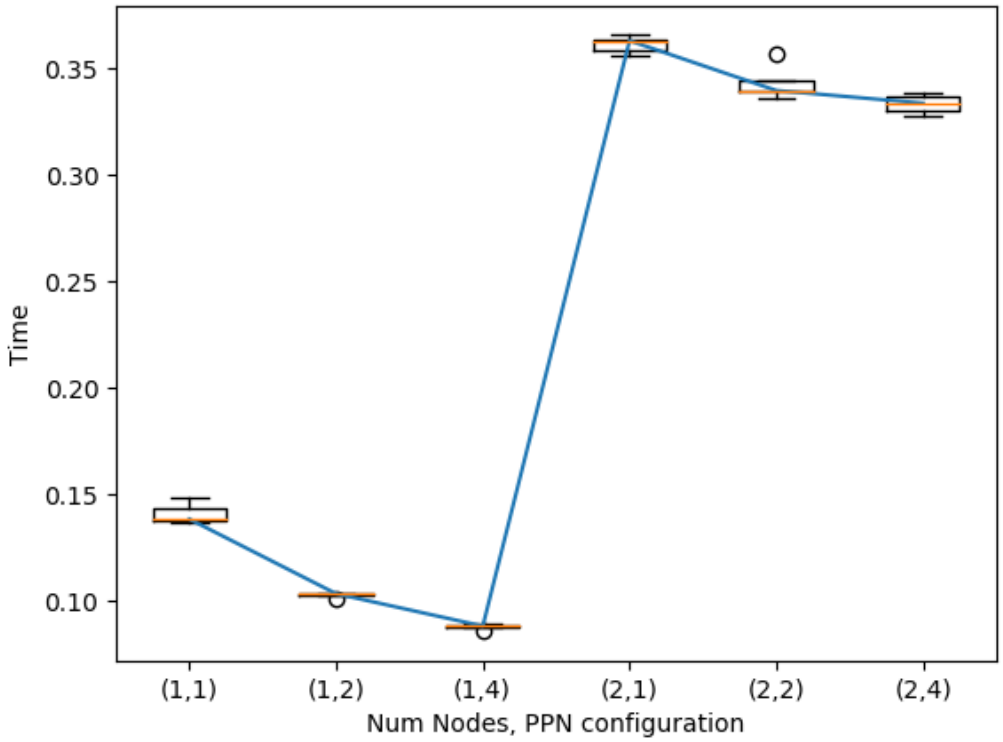
Optimization performed:

- As mentioned above, in both the Bcast and Reduce calls, we have coupled the data in order to reduce the number of calls and thereby any overhead resulting, even in form of networks calls.
- We have used node based sub-communicators so that the number of messages being communicated through network decreases.
- The data distribution strategy although creates a slight imbalance, but is done to avoid the use of vector variant wherein computation of counts and displacements are required.

Observations and Plots

The plot depicting the time of different configurations is as shown below. As mentioned, the above configurations were executed 5 times. For each configuration a box plot of the recorded time duration was constructed and a line connecting the median of those box plots is also shown. The Y-axis records the time taken by the different configurations in every execution. The X-axis marks the configuration describing the number of nodes and process per node.

Plot



As observed in the plot, for a fixed number of node value, when the process per node increases, the time decreases. The decrease is very steep for number of nodes = 1 and slight for number of nodes = 2. The steep but almost linear decrease in number of Nodes = 1 configuration suggests that the overhead occurred in distributing the was very less compared to the advantage obtained by decreasing the domain of the work. Also for number of node = 2, the decrease suggests that the benefit obtained by distributing the work was slightly better than the overhead of communication. Moreover, the sudden increase of time when considering number of nodes=1 and number of nodes=2 suggest that the work employed for each process was not so large enough to overcome the time when data is communicated between the nodes through network.