Report

Instructions to execute the code

The Assignment3 directory contains the following files:

1. src.c
2. run.sh
3. Makefile
4. script.py
5. plot.py
6. output.txt
7. plot.jpg

The whole assignment can be executed by the script file ‘run.sh’. For executing the code, make sure that the path to ‘mpich’ is pointing to user’s installation directory. Use the following commands to execute:

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

The 'run.sh' script will generate two files, 'output.txt' and 'times.txt'. 'output.txt' is the required output file with three lines. The first line contains yearwise minimum data in csv format, second line will contain global minimum data across all years and third line will contain the maximum execution time across all process. File 'times.txt' will store all the execution times and will be used for generating plots.

To generate plots, use the following commands:

```
$ python3 plot.py
```

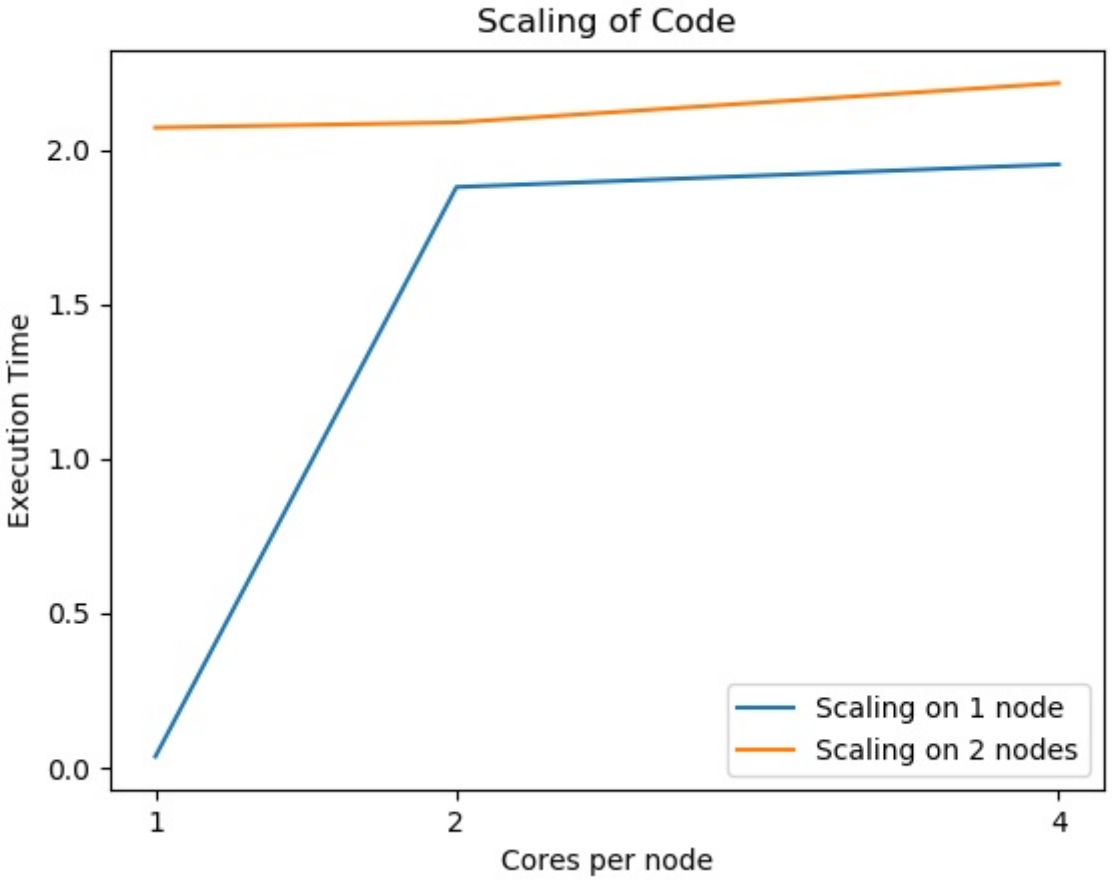
Note: To allocate the nodes dynamically, we have written a small script named 'script.py'. It takes two arguments, number of nodes and number of cores and generate hostfile named 'hostfile'.

Explanation of the code:

The src.c is the source code to compute the minimum temperature as asked in the Assignment3. It will take only one argument, name of the data file. We are reading data file through rank 0.

As mentioned, data file can have different numbers of rows and columns, so we first counted the rows and columns and then allocated buffer to place the data. To distribute the data, we use a strategy in which each process except rank 0 will get equal number (number_of_columns_in_data/number_of_process) of columns for computation and rank 0 will take care of remaining columns. To distribute the data to different processes, we have used MPI_Pack/Unpack. After reading data from input file, rank 0 will pack the correct amount of data for each process and sends it using MPI_Send. After receiving data, each process computes year-wise minimum in allocated years and send the result back to rank 0. Now rank 0 will compute the minimum temperature across all years and save year-wise minimum temperatures and global minimum temperature into file 'output.txt'. We timed the entire computation after rank 0 finished reading data from the file.

The file 'output.txt' will get overwritten each time the code will get executed for different configurations. So we save the execution times in a file "times.txt" for plotting.



Observations on results

1. We can observe that when we execute the code on single node and single core, in that case it took minimum time; possibly because it doesn't involve any data communication.

2. As we increase the number of cores, the execution time also starts increasing in both the cases. This is because of increase in data communication between the cores which is bandwidth limited.
3. One more thing we can observe is that using single core on 1 node results in much better peorformance than using 2 nodes with one core each. This is because using two nodes involve inter-node communication which is very expensive.
4. As number of cores is increasing, the execution time becomes almost constant for both cases. This is because, as the degree of parallelism is increasing, the inter-process communication is also increasing.