CS633A

Parallel Computing

Assignment 3

Data Decomposition Simulator

Aditya Jain
20111004

Mohit Kumar
20111034

Instructor:
Dr. Preeti Malakar

April 19, 2021

1 Steps to run the code

- **NOTE:** For automated plot generation, please install python and following python libraries - matplotlib and numpy.
- Execute run script from terminal as :- $ ./run.sh
- Above command will make the program, generate the host file on fly, execute the program with all required configurations, and will generate the output files which are required to generate the plot.

2 Implementation Details

2.1 Implementation Details

2.1.1 Decomposition Strategy

- First, we made rank 0 to read the csv file into a 2d array *arr* in column major form. So each row of this array represents the data for a year across all the stations.
- Then we decomposed the data array *arr* year-wise among all the processes. If the data array contains data for Y years, we decomposed our data as floor(Y/size) (where size is equal to the total number of processes launched.) This data is shared to other processes through **MPI_Scatter()** function call. If Y is not a multiple of size, then there are data for some years which is not distributed by MPI_Scatter(), so we allocate the data for such years to the process 0.
- **Example:** In our sample file, there is data for 41 years from 1960 to 2000. If size=8, then 41/8 = 5. So data for five years is distributed among all the eight processes. Since 41 is not a multiple of 8, there exist the data for one year (i.e. 2000) which is not distributed through MPI_Scatter(), data for this year is allocated to the process with rank 0.

2.1.2 Reasoning for such decomposition:

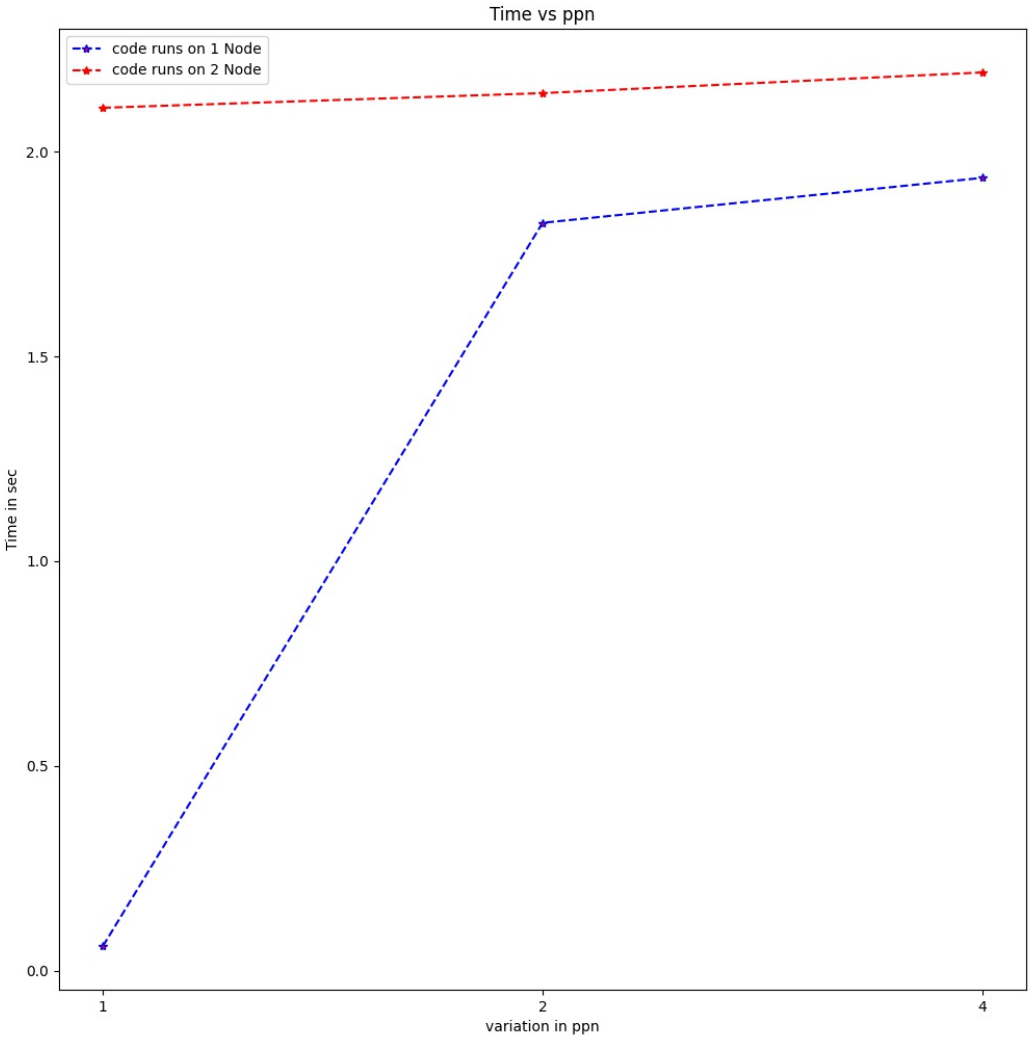
- Once each process has received the data for Y/size years, each of them calculates the minimum temperature value for their assigned years. This calculated result is then sent to the rank 0 through **MPI_Gather()** function call.
- After receiving the result values of each year, rank 0 calculates the minimum temperature value among all these values to obtain the global minimum temperature over all the stations and all years.
- The rank 0 then writes the resulting data into the file output.txt.

2.1.3 Technqiue used to calcutate time,maximum across all processes

- Our code uses **MPI_Reduce()** to calculate maximum time taken across all processes to do the work.Rank 0 is used as root process.
- Every process feed the time taken in **MPI_Reduce()**.
- **MPI_Reduce()** calcuate the maximum of time and resultant is feed into output variable at root process rank 0.
- The rank 0 then writes the maximum time into the file output.txt.

3 Simulation Results

3.1 Plot for variation in time vs ppn



4 Simulation Analysis

4.1 Time taken by only 1 process is minimum

- If the number of process is 1, then the program will do sequential work i.e no parallelisation.
- In this case, the process has not to distribute the data as there is only one process.
- So,the process has only to do computation and not the decomposition of data.
- In the sample data file provided ,computation is of easy nature i.e calculating minimum.
- This causes only one process to take very less time as compare to other cases.

4.2 Time taken by 2 Node is greater than 1 Node for a fixed ppn

- If the code uses only one node, then all the communication is intra-node or inter-process communication.
- If the code uses 2 node, then the data decomposition and other type of communication would require inter-node communication as well as intra-node communication.
- The time taken in inter-node communication depends on many factor like contention,bandwidth, data-size etc.
- Since,the computation is of easy nature .The communication is dominating factor in time taken by code.
- In several run, it was observed that if the code uses 2 node it takes more time as compared to 1 node used by code for fixed ppn.