Data Parallelism in Linear Regression using MPI

Course: CSE 708 Programming Massively Parallel Systems
by Dr. Russ Miller

Submitted by:

Ritika Rekhi (ritikare@buffalo.edu)

Linear Regression

- Linear Regression is used when we want to predict the value of one variable using the value of another variable.
- The variable whose value we want to predict is called as the **dependent variable**.
- The variable that are used to predict the value of the above dependent variable is called as the **independent variable**.
- Linear regression is widely applicable in fields like economics and finance, marketing, healthcare, real estate, sports, and education, showcasing its versatility in data analysis and predictive modeling.
- In this project, I will analyze a household power consumption dataset containing approximately 2M records to predict current intensity of households based on various readings.

Source: https://www.ibm.com/topics/linear-regression

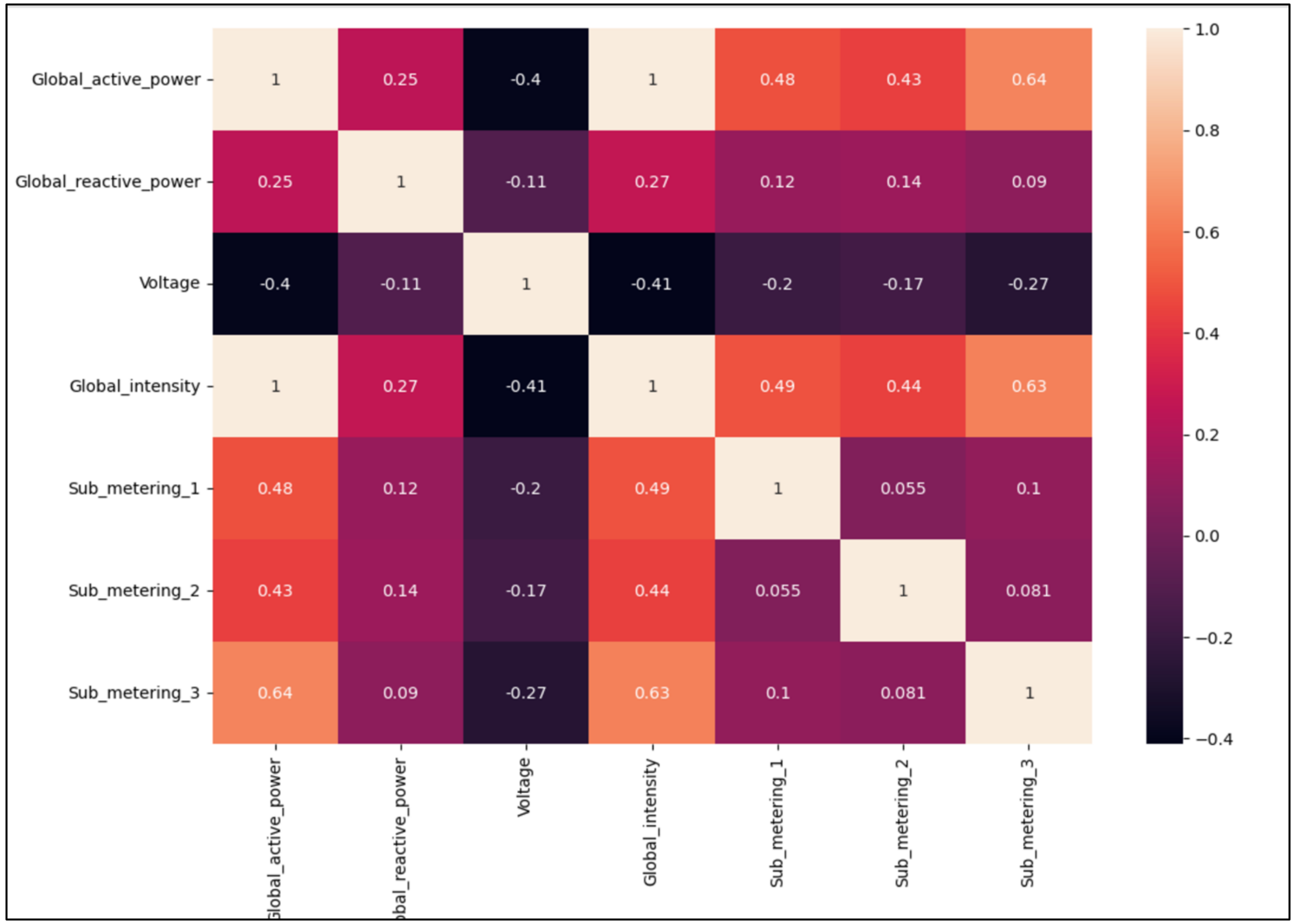
Household Power Consumption

This dataset is a time series dataset which contains **2,075,259 measurements** gathered between December 2006 and November 2010. Based on the definition of Linear Regression, the variables in our dataset would be categorised as follows:

- Independent Variables
  - **Global Active Power :** Household global minute-averaged active power (in kilowatt)
  - **Global Reactive Power:** Household global minute-averaged reactive power (in kilowatt)
  - **Voltage:** Minute-averaged voltage (in volt)
  - **Sub-metering 1:** It corresponds to the kitchen, containing mainly a dishwasher, an oven and a microwave (hot plates are not electric but gas powered).
  - **Sub-metering 2:** It corresponds to the laundry room, containing a washing-machine, a tumble-drier, a refrigerator and a light.
  - **Sub-metering 3:** It corresponds to an electric water-heater and an air-conditioner.
- Dependent Variable
  - **Global Intensity:** Household global minute-averaged current intensity (in ampere)

Source: https://www.kaggle.com/datasets/uciml/electric-power-consumption-data-set

- Based on the image on the right, it can be clearly seen that the independent and the dependent variables have a high correlation value. Thus, the fitted line that we would get after performing Linear Regression would be able to give the value of Global intensity accurately.



	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	Sub_metering_2	Sub_metering_3
Global_active_power	1	0.25	-0.4	1	0.48	0.43	0.64
Global_reactive_power	0.25	1	-0.11	0.27	0.12	0.14	0.09
Voltage	-0.4	-0.11	1	-0.41	-0.2	-0.17	-0.27
Global_intensity	1	0.27	-0.41	1	0.49	0.44	0.63
Sub_metering_1	0.48	0.12	-0.2	0.49	1	0.055	0.1
Sub_metering_2	0.43	0.14	-0.17	0.44	0.055	1	0.081
Sub_metering_3	0.64	0.09	-0.27	0.63	0.1	0.081	1

Figure 1: Image showing correlation between the dependent and the independent variables.

Linear Regression Training Algorithm for a Single Machine

- Linear Regression is all about finding the best fit line that would help in predicting the value of new data points. Since this problem has multiple independent variables, we would need to find the value for the formula below:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 \ldots\ldots \beta_n X_n$$

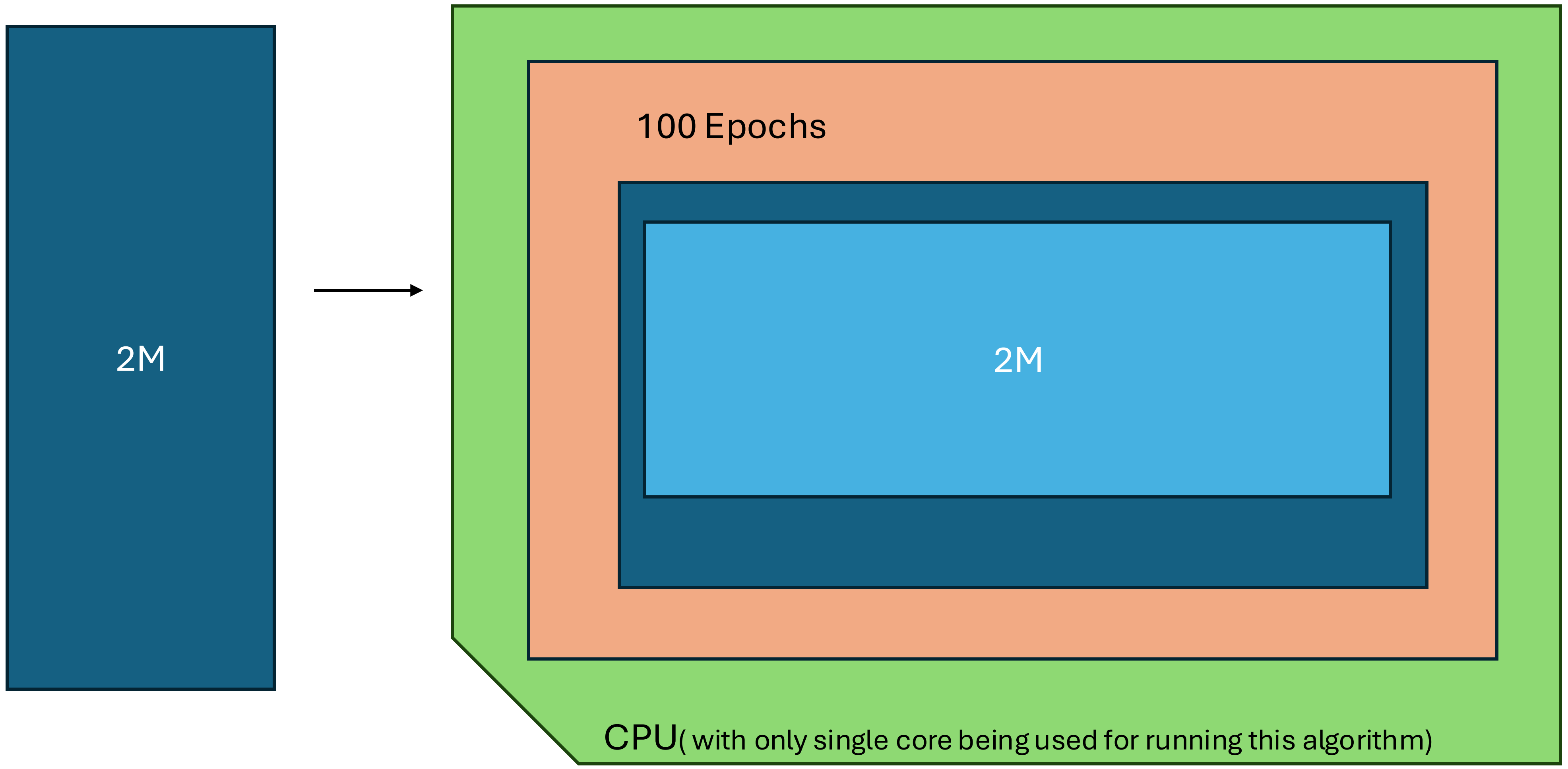
  where, β_0 is the y-intercept and $\beta_1, \beta_2, \beta_3 \ldots \beta_n$ are the coefficient of each independent variable.

- The **dependent** and **independent variables** also get converted to a **matrix** for calculations.
- This algorithm would be leveraging **Stochastic Gradient Descent**, which uses gradient descent on mini-batches of data to get appropriate values of y-intercept and the independent variable coefficients. The formula used for the calculation would be as follows:

$$w_j = w_j + \alpha \sum_i (y_i - w^T x_i) x_{ij}$$

  where w_j is the coefficient value corresponding to that independent variable or the y-intercept, α is the learning rate with which we want to update the coefficient values each iteration, w^T is the weight matrix having the y-intercept values and independent variable coefficient values, $\sum_i (y_i - w^T x_i)$ tells the difference between the predicted and actual value and x_{ij} is the j attribute value for the i training example.

- Step 1: Initialize the y-intercept, coefficients of independent variables and initial loss.
- Step 2: Repeat the below steps for 100 epochs:
    - Step 2a): Repeat the below steps for each of the data in the dataset:
        - Step 2a.a): Get the record.
        - Step 2a.b): Get the predicted value based on the weights.
        - Step 2a.c): Calculate the error by comparing the predicted value with the actual value.
        - Step 2a.d): Update the values of y-intercept and coefficients of independent variables.
- Step 3: Check the accuracy of the model against the test dataset.


2M
100 Epochs
2M
CPU(with only single core being used for running this algorithm)

Problems with this approach

- The size of the independent variables' matrix is huge.
- As the size of the data grows, it might be possible we might run out of RAM to perform the calculations.
- The time taken for convergence might also be longer as we might need to iterate over the same data again and again until we get the desired loss.
- This can be performed in parallel instead of running the entire flow on a single machine.

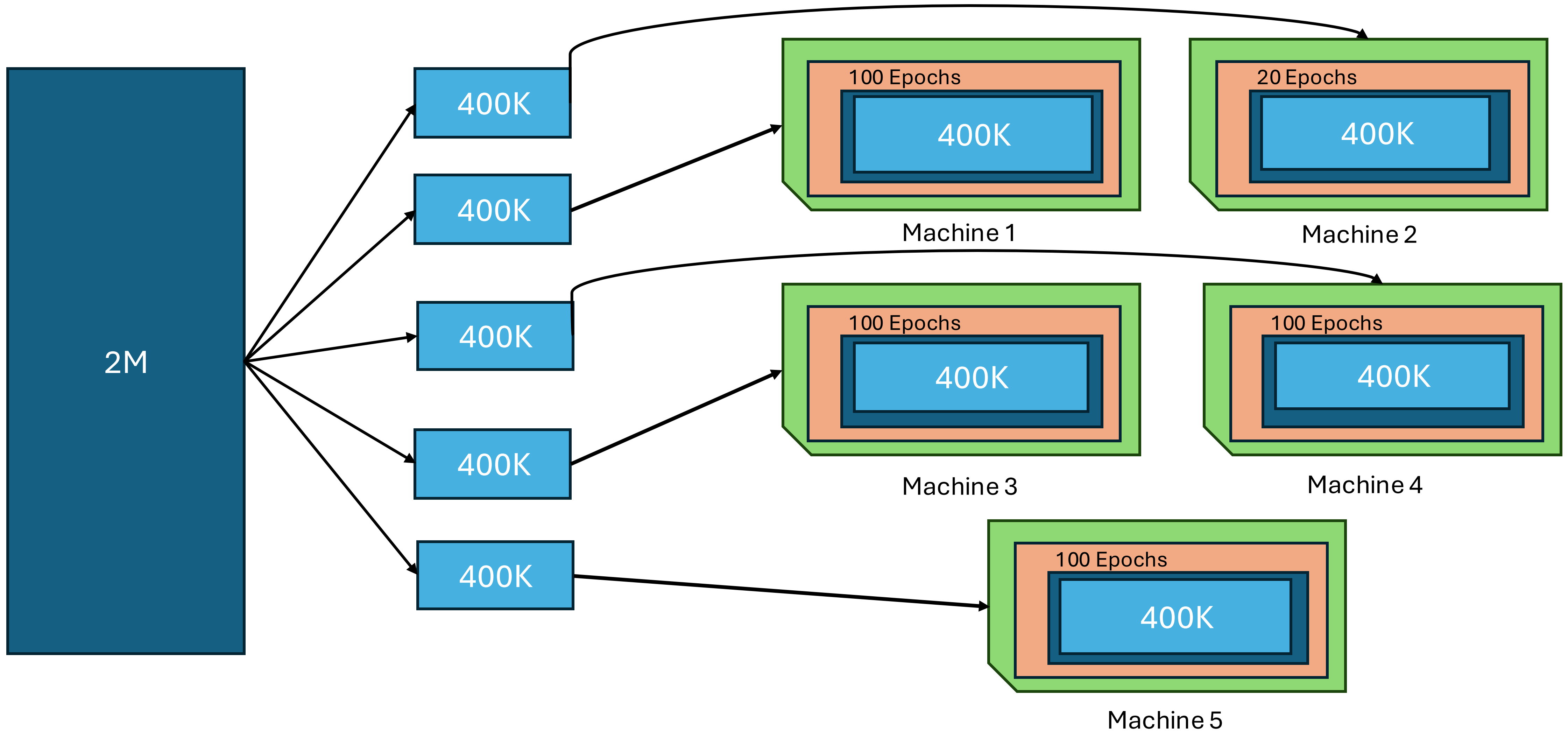
Data Parallelism in Linear Regression using MPI

- Based on the previous algorithm, we can clearly offload some calculation to the other machines.
- This way we will not run out of RAM when dealing with data that might increase in size.
- To perform data parallelism:
  - The 2M records would be split among the different number of machines.
  - Each machine will then operate on its own data.
  - When the gradient will be calculated on each machine, they would share the gradient values with other machines and each machine would then take an average of those values to update the model parameters.

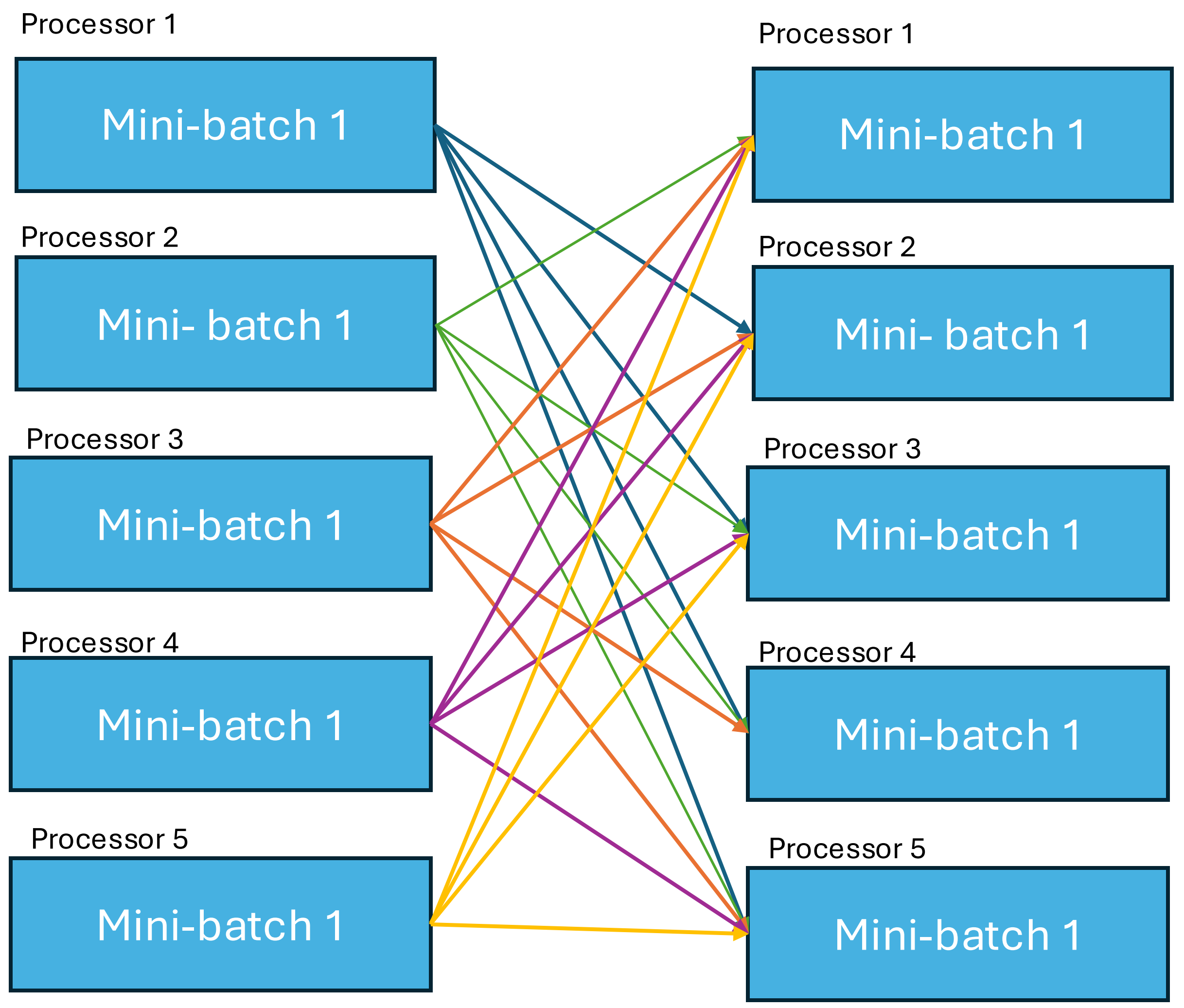
Linear Regression Algorithm in parallel

- Step 1: Initialize the y-intercept, coefficients of independent variables and initial loss.
- Step 2: Repeat the below steps for 100 epochs:
  - Step 2a): Repeat the below steps for each of records in the data assigned to that node:
    - Step 2a.a): Get the record.
    - Step 2a.b): Get the predicted value based on the weights.
    - Step 2a.c): Calculate the error by comparing the predicted value with the actual value.
    - Step 2a.d): Update the values of y-intercept and coefficients of independent variables.
    - Step 2a.e): After every n^{th} record in the dataset:
      - Step 3a.c): Share the gradient values with all the hosts.
      - Step 3a.e): Update the values of y-intercept and coefficients of independent variables using the averaged gradient values.
- Step 3: Check the accuracy of the model against the test dataset.

Diagram flow for Parallelized Linear Regression



MPI Communication Flow



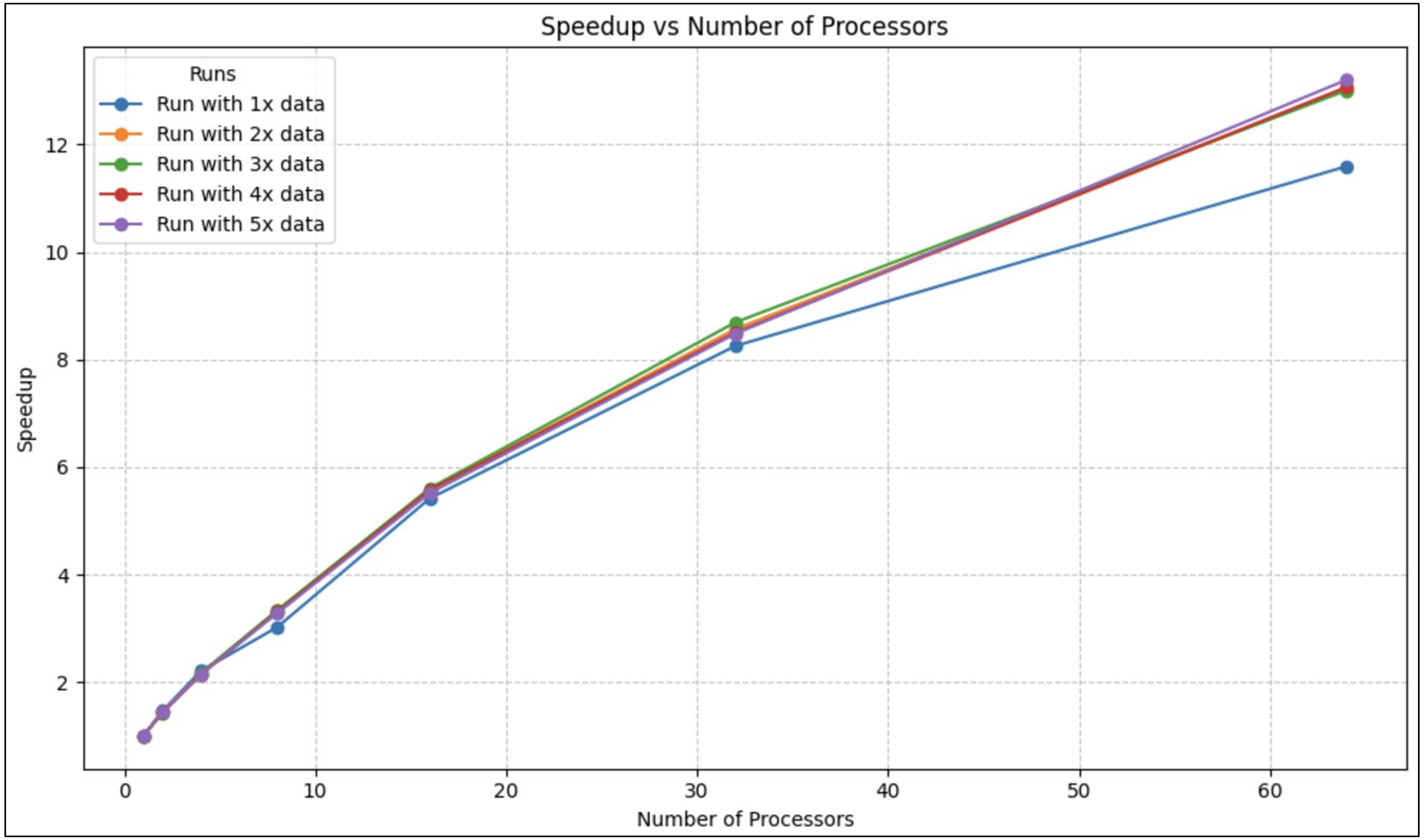
Implementation

- **MPI_Allgather** function was used to send the values to all the hosts so that all the machines are at the same compute level at the same time.
- To ensure that the accuracy of the Linear regression model did not drop with increase in the number of processosrs, a series of experiments were run with the processors communicating after every **50, 100 and 1000 records** in the dataset.
- Based on the results, it was better to communicate after every **100th record** in the dataset, so that the accuracy of the model does not decrease with increase in the number of processors.

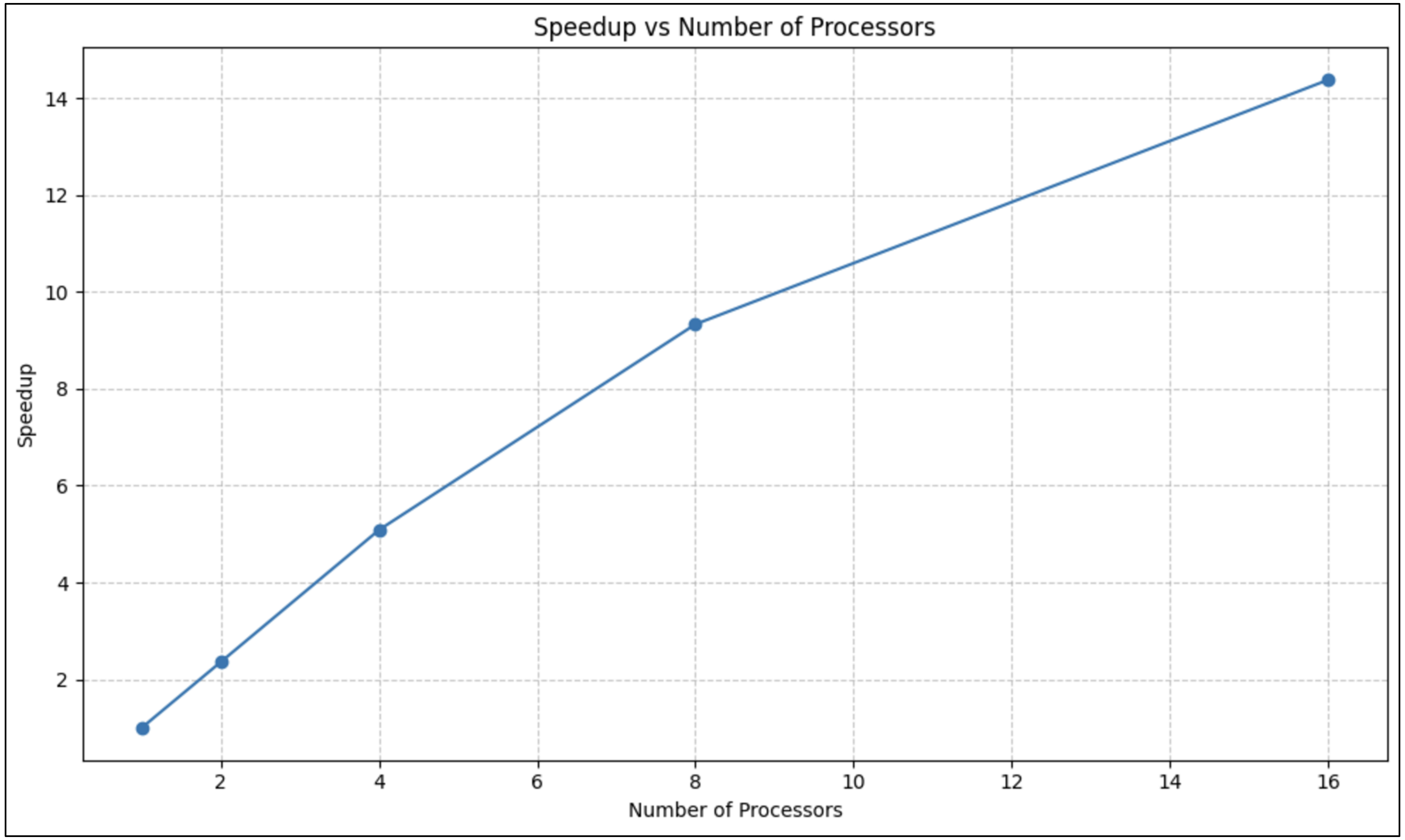
Results

Processors	Run 1 with 1x data - Time(ms)	Run 2 with 2x data - Time(ms)	Run 3 with 3x data - Time(ms)	Run 4 with 4x data - Time(ms)	Run 5 with 5x data - Time(ms)
1	8688.3206	17169.6961	25625.2576	34155.8526	42771.9858
2	5946.6893	11815.9860	17879.3098	23786.1215	29469.7481
4	3936.2508	8037.9675	11834.9987	15961.6785	19906.5969
8	2877.2035	5163.2822	7685.0937	10344.8398	13039.2069
16	1603.8647	3082.4627	4570.6412	6127.9872	7751.8223
32	1052.9779	2004.4688	2948.6935	4016.9285	5044.8213
64	749.5328	1317.4484	1970.6555	2614.2083	3240.4022

Results- Amdahl's Law



Results – Gustafson's Law



Lessons Learned Data Parallelism for Linear Regression

- The data that is distributed among different processors should have all types of data to get a better fitted line.
- When dealing with a time-series data, it is necessary that the order is maintained so that we get accurate values for the coefficients of the independent variables.
- It is also necessary that there is sufficient amount of data available on each of the machines so that the Linear Regression model is trained accurately.