



**INDIAN INSTITUTE OF INFORMATION TECHNOLOGY
DESIGN AND MANUFACTURING KANCHEEPURAM**

**PROJECT REPORT
ON
K-MEAN CLUSTERING ALGORITHM
USING MPI**

**SUBMITTED BY
AMAR KUMAR
(CED17I029)
TO
DR. NOOR MAHAMMAD**

Problem Statement

A Pizza company wants to open its delivery centres across a very big city.

The challenges they will face is that they need to analyze areas from where pizza is being ordered frequently.

They need to figure out the locations for the pizza stores within all these areas in order to keep the distance between the store and delivery points minimum.

Resolving these challenges includes a lot of analysis and mathematics. We would learn here about how clustering can provide a meaningful and easy method of sorting out such real life challenges.

My Solution to this problem

I have used an unsupervised machine learning algorithm to solve this problem. I have used the K-Means clustering algorithm for solving the above problem.

My dataset contains x and y coordinates of locations of different people/their houses across the city and some randomly generated pizza centers(According to my algorithm, the number of pizza centers can vary from 1 to 100).

We have to find the optimum location for these pizza centers so that each house can get the delivery fastest.

Step1 : For this, I have calculated the distance of every house from every pizza center. The house which is nearest to a particular pizza center, will be served by that particular pizza center in the first iteration.

Step2 : Now the location of the pizza center will change according to the location of houses which come under it. X coordinate of the pizza center will be calculated by taking the mean of x coordinates of the location of houses. Similarly Y coordinate of the pizza center will be calculated by taking the mean of y coordinates of the location of the houses.

Step3 : For further iteration repeat from Step1

Note 1 : More the number of iterations, more optimal will be the location of the pizza center. But the location of the pizza center will not vary after a certain number of iterations as it will start converging to an optimal location of the pizza center such that a house can get its delivery faster.

Note 2 : This implementation of k-mean clustering algorithm in MPI is done on HPC server.

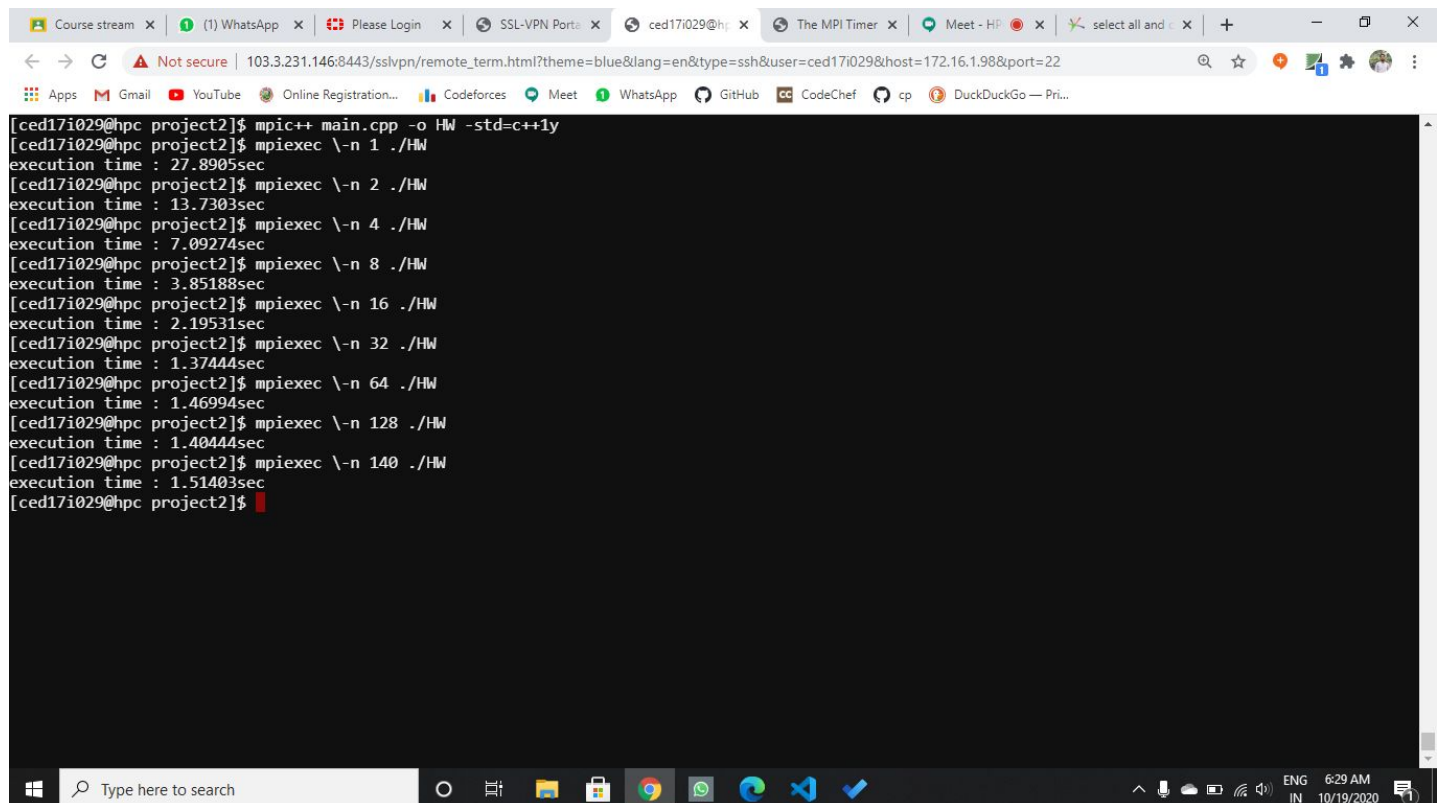
How did i do parallel programming

I have used MPI technique to parallelize my program.

From my solution to the problem statement, we can clearly see that we have to calculate the distance between pizza center and location of houses. So I have parallelized the program for calculating the distance. I.e. distributed the work of calculating distance between master and worker.

Graph and table

https://docs.google.com/spreadsheets/d/1rrEXtpddJWioTHJEkV_5FgHw532DAWcloMXupw36CIU/edit#gid=0



The screenshot shows a terminal window with the following commands and output:

```
[ced17i029@hpc project2]$ mpic++ main.cpp -o HW -std=c++1y
[ced17i029@hpc project2]$ mpiexec \-n 1 ./HW
execution time : 27.8905sec
[ced17i029@hpc project2]$ mpiexec \-n 2 ./HW
execution time : 13.7303sec
[ced17i029@hpc project2]$ mpiexec \-n 4 ./HW
execution time : 7.09274sec
[ced17i029@hpc project2]$ mpiexec \-n 8 ./HW
execution time : 3.85188sec
[ced17i029@hpc project2]$ mpiexec \-n 16 ./HW
execution time : 2.19531sec
[ced17i029@hpc project2]$ mpiexec \-n 32 ./HW
execution time : 1.37444sec
[ced17i029@hpc project2]$ mpiexec \-n 64 ./HW
execution time : 1.46994sec
[ced17i029@hpc project2]$ mpiexec \-n 128 ./HW
execution time : 1.40444sec
[ced17i029@hpc project2]$ mpiexec \-n 140 ./HW
execution time : 1.51403sec
[ced17i029@hpc project2]$
```

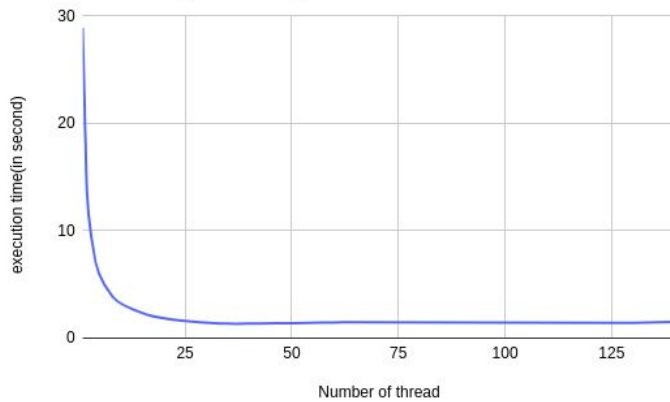
The terminal window is part of a web browser interface. The browser's address bar shows a URL for a remote terminal session. The browser's taskbar at the bottom shows various application icons and the system clock indicating 6:29 AM on 10/19/2020.

K-Mean clustering using MPI

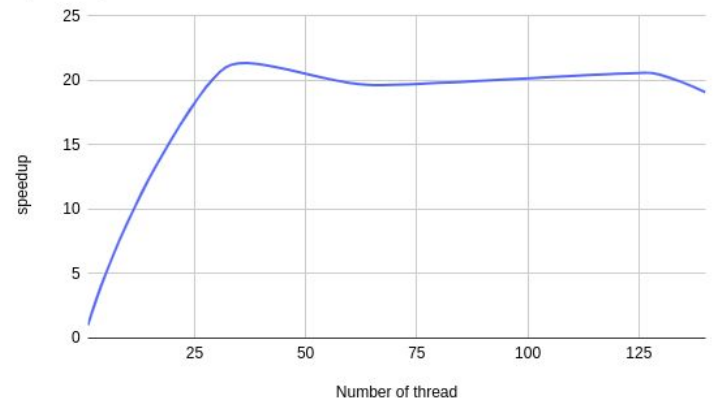
number of data points = 100000 , number of iterations = 50, number of pizza center = 100

Number of thread	execution time(in second)	speedup	parallelization fraction(f)
1	28.8905	1	0
2	13.7303	2.104141934	1.049493778
4	7.09274	4.073249548	1.005994358
8	3.85188	7.500363459	0.9904835747
16	2.19531	13.1601004	0.9856135869
32	1.37444	21.01983353	0.9831492996
64	1.46994	19.65420357	0.9641857006
128	1.40444	20.5708325	0.9588787203
140	1.51403	19.08185439	0.9544114163

execution time(in second) vs. Number of thread



speedup vs. Number of thread



Calculation of parallelization fraction

$T(1) = 28.8905$ seconds

Here , for $P = 32$ the execution time is minimum

$T(P) = 1.37444$ seconds

$$\text{Speedup} = \frac{T(1)}{T(P)} = \frac{28.8905}{1.37444} = 21.01983353$$

From Amdahl's Law,

$$\text{Speedup} = \frac{1}{(f/P) + (1-f)} \text{ Where , } f = \text{Parallelization factor } P = \text{Thread Number}$$

$$\text{So, } f = \frac{(1-T(P)/T(1))}{(1-(1/P))}$$

Therefore, $f = 0.9831492996$ which means that approx.98% of the program is parallelizable.