



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

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

**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 CUDA is done on Google Colab.

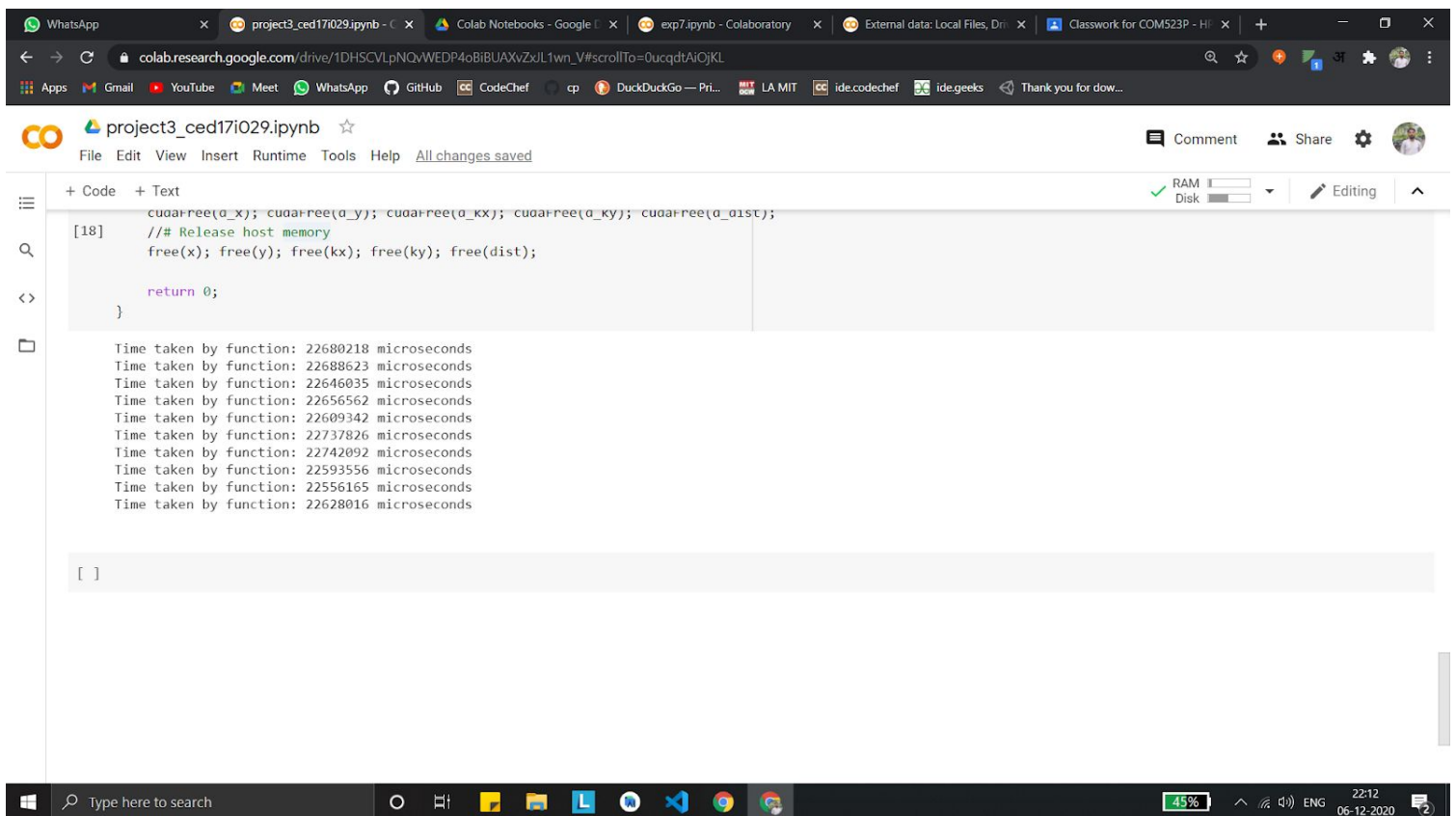
## How did i do parallel programming

I have used CUDA 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 by running calculateDistance function in device parallelly.

## Graph and table

[https://docs.google.com/spreadsheets/d/1RiWsOftH94EjRSFfPNMegSW8tQSAgpQqePwF\\_0k2A04/edit#gid=0](https://docs.google.com/spreadsheets/d/1RiWsOftH94EjRSFfPNMegSW8tQSAgpQqePwF_0k2A04/edit#gid=0)



The screenshot shows a Google Colab notebook interface. The browser tabs at the top include WhatsApp, project3\_ced17i029.ipynb, Colab Notebooks - Google, exp7.ipynb - Colaboratory, External data: Local Files, Dri..., and Classwork for COM523P - H... The address bar shows the Colab URL: colab.research.google.com/drive/1DHSCVLPnQvWEDP46BiBUAXvZxL1wn\_V#scrollTo=0ucqdtAiOjKL. The notebook title is project3\_ced17i029.ipynb. The code editor shows the following code:

```
+ Code + Text
[18]
    cudaFree(d_x); cudaFree(d_y); cudaFree(d_kx); cudaFree(d_ky); cudaFree(d_dist);
    /// Release host memory
    free(x); free(y); free(kx); free(ky); free(dist);

    return 0;
}
```

The output of the code execution is displayed below the code cell:

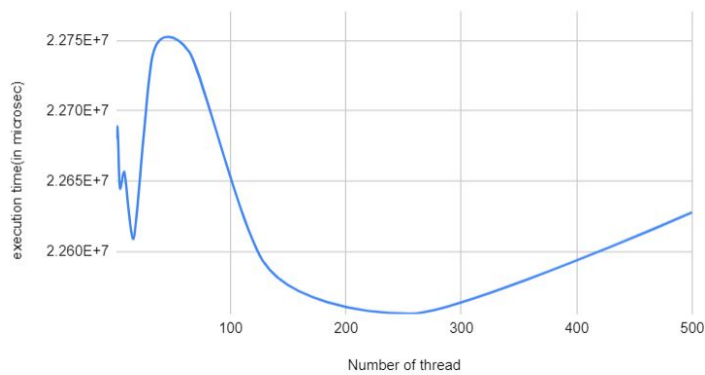
```
Time taken by function: 22680218 microseconds
Time taken by function: 22688623 microseconds
Time taken by function: 22646035 microseconds
Time taken by function: 22656562 microseconds
Time taken by function: 22609342 microseconds
Time taken by function: 22737826 microseconds
Time taken by function: 22742092 microseconds
Time taken by function: 22593556 microseconds
Time taken by function: 22556165 microseconds
Time taken by function: 22628016 microseconds
```

The bottom status bar shows the Windows taskbar with the search bar, taskbar icons, and system tray information: 45% battery, 22:12, 06-12-2020.

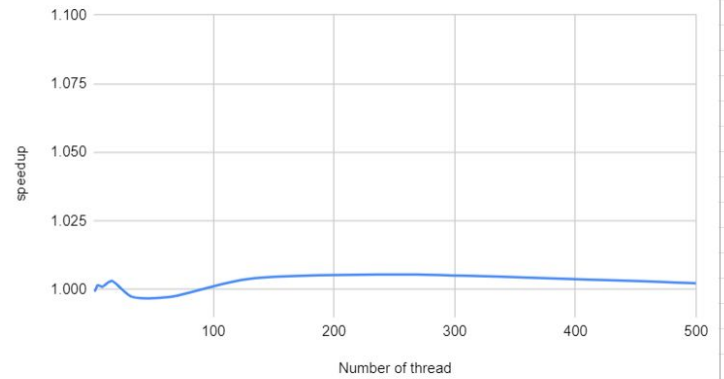
# CUDA implementation of Project Problem

Number of thread	execution time(in microsec)	speedup	parallelization fraction(f)
1	22680218	1	0
2	22688623	0.99962955	-0.0007411745337
4	22646035	1.001509447	0.002009563283
8	22656562	1.001044113	0.001192026839
16	22609342	1.00313481	0.003333348324
32	22737826	0.9974664245	-0.002621946693
64	22742092	0.997279318	-0.002771407532
128	22593556	1.003835695	0.003851126032
256	22556165	1.005499738	0.005491106053
500	22628016	1.002306963	0.002306265893

execution time(in microsec) vs. Number of thread



speedup vs. Number of thread



## Calculation of parallelization fraction

$T(1) = 22680218$  microseconds

Here , for  $P = 256$  the execution time is minimum

$T(P) = 22556165$  microseconds

$$\text{Speedup} = \frac{T(1)}{T(P)} = \frac{22680218}{22556165} = 1.005499738$$

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,  $f = 0.005491106053$  which means that approx. 0.55% of the program is parallelized.