# Distributed Data Analytics

## *Exercise Sheet 3*

Mohsan Jameel, Florian Pal

Information Systems and Machine Learning Lab

University of Hildesheim

Name: Pedram Babakhani

Student Number: 276848

Email: Pedram_Babakhani@yahoo.com

# Exercise 1.

Clustering is the task of assigning a set of objects into groups (called clusters) so that the objects in the same cluster are more similar (in some sense or another) to each other than to those in other clusters. As a naïve example: animals can be clustered as land animals, water animals and amphibians.

• k-means clustering is a method of clustering which aims to partition n data points into k clusters (n >> k) in which each observation belongs to the cluster with the nearest mean.

• The nearness is calculated by distance function which is mostly Euclidian distance or Manhattan distance.(I useed Euclidian)

• One important assumption to be made is the data points are independent of each other. In other words there exists no dependency between any data points.

In the following you can see the time execution comparison between parallel and sequential approach.

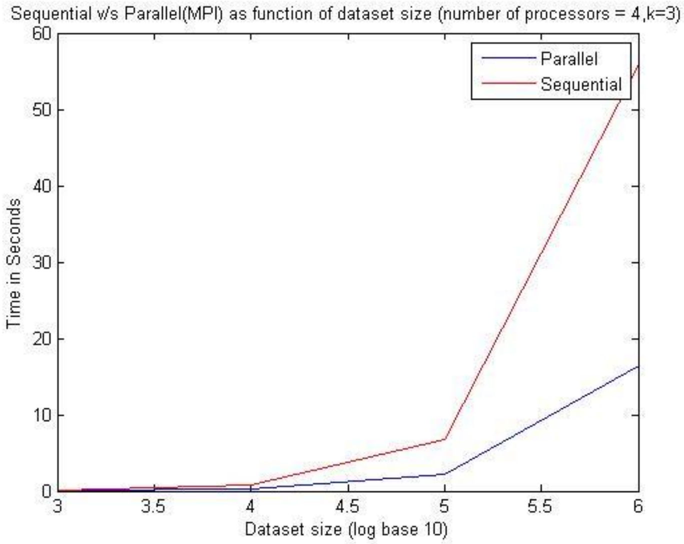


Figure 1: Time execution between parallel and sequential[1]

In the following, figure 2 demonstrate the speedup and time execution vs number of processes, we can see as we increase number of processes we will get better speedup but we have to find the best K and best number of Processor.

[1] Buffalo university https://www.cse.buffalo.edu/faculty/miller/Courses/CSE633/Chandramohan-Fall-2012-CSE633.pdf

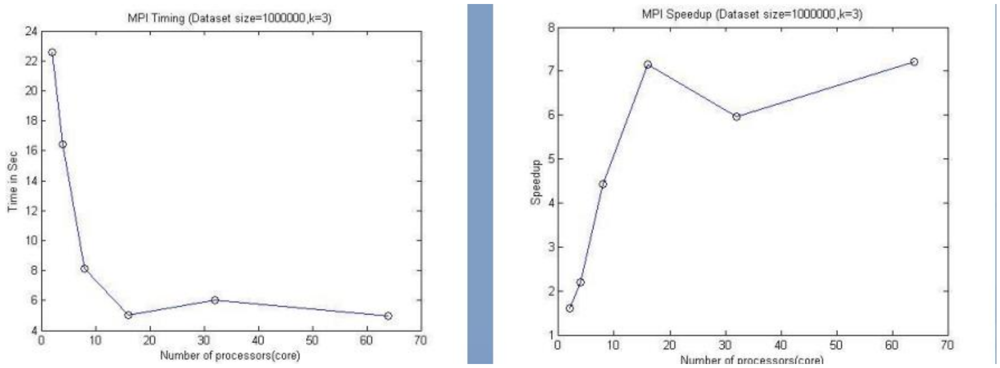
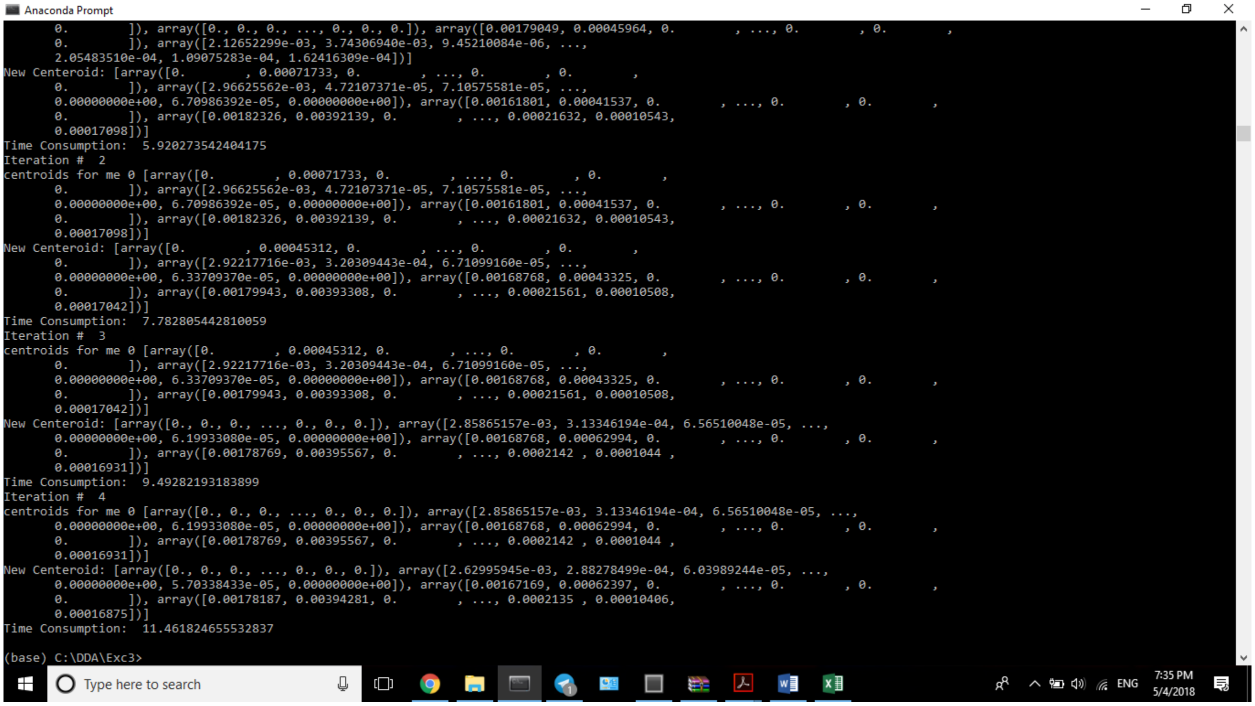


Figure 2: Time execution and speed up vs number of processes[2]

First of all I victories the texts in folders (only 2 folders because of memory limitation), After that I split the vector horizontally and send for all processes, every process try to find Euclidian Distance and then find which of them are closer the update the membership and assign the data to a cluster. I used dictionary for membership, actually it is dictionary of list which keeps the membership of every class. On the other hand at the end I will calculate new centroids by mean of the values in membership vector.
Then at the end, process 0 receives the membership from other processes and calculate new centroids and calculate time execution. In the following section I will compare my results with the real results.

Figure 3 shows the output of terminal.



```
       0.        ]), array([0., 0., 0., ..., 0., 0., 0.]), array([0.00179049, 0.00045964, 0.        , ..., 0.        , 0.        ,
       0.        ]), array([2.12652299e-03, 3.74306940e-03, 9.45210084e-06, ...,
       2.05483510e-04, 1.09075283e-04, 1.62416309e-04])]
New Centeroid: [array([0.        , 0.00071733, 0.        , ..., 0.        , 0.        ,
       0.        ]), array([2.96625562e-03, 4.72107371e-05, 7.10575581e-05, ...,
       0.00000000e+00, 6.70986392e-05, 0.00000000e+00]), array([0.00161801, 0.00041537, 0.        , ..., 0.        , 0.        ,
       0.        ]), array([0.00182326, 0.00392139, 0.        , ..., 0.00021632, 0.00010543,
       0.00017098])]
Time Consumption:  5.920273542404175
Iteration #  2
centroids for me 0 [array([0.        , 0.00071733, 0.        , ..., 0.        , 0.        ,
       0.        ]), array([2.96625562e-03, 4.72107371e-05, 7.10575581e-05, ...,
       0.00000000e+00, 6.70986392e-05, 0.00000000e+00]), array([0.00161801, 0.00041537, 0.        , ..., 0.        , 0.        ,
       0.        ]), array([0.00182326, 0.00392139, 0.        , ..., 0.00021632, 0.00010543,
       0.00017098])]
New Centeroid: [array([0.        , 0.00045312, 0.        , ..., 0.        , 0.        ,
       0.        ]), array([2.92217716e-03, 3.20309443e-04, 6.71099160e-05, ...,
       0.00000000e+00, 6.33709370e-05, 0.00000000e+00]), array([0.00168768, 0.00043325, 0.        , ..., 0.        , 0.        ,
       0.        ]), array([0.00179943, 0.00393308, 0.        , ..., 0.00021561, 0.00010508,
       0.00017042])]
Time Consumption:  7.782805442810059
Iteration #  3
centroids for me 0 [array([0.        , 0.00045312, 0.        , ..., 0.        , 0.        ,
       0.        ]), array([2.92217716e-03, 3.20309443e-04, 6.71099160e-05, ...,
       0.00000000e+00, 6.33709370e-05, 0.00000000e+00]), array([0.00168768, 0.00043325, 0.        , ..., 0.        , 0.        ,
       0.        ]), array([0.00179943, 0.00393308, 0.        , ..., 0.00021561, 0.00010508,
       0.00017042])]
New Centeroid: [array([0., 0., 0., ..., 0., 0., 0.]), array([2.85865157e-03, 3.13346194e-04, 6.56510048e-05, ...,
       0.00000000e+00, 6.19933080e-05, 0.00000000e+00]), array([0.00168768, 0.00062994, 0.        , ..., 0.        , 0.        ,
       0.        ]), array([0.00178769, 0.00395567, 0.        , ..., 0.0002142 , 0.0001044 ,
       0.00016931])]
Time Consumption:  9.49282193183899
Iteration #  4
centroids for me 0 [array([0., 0., 0., ..., 0., 0., 0.]), array([2.85865157e-03, 3.13346194e-04, 6.56510048e-05, ...,
       0.00000000e+00, 6.19933080e-05, 0.00000000e+00]), array([0.00168768, 0.00062994, 0.        , ..., 0.        , 0.        ,
       0.        ]), array([0.00178769, 0.00395567, 0.        , ..., 0.0002142 , 0.0001044 ,
       0.00016931])]
New Centeroid: [array([0., 0., 0., ..., 0., 0., 0.]), array([2.62995945e-03, 2.88278499e-04, 6.03989244e-05, ...,
       0.00000000e+00, 5.70338433e-05, 0.00000000e+00]), array([0.00167169, 0.00062397, 0.        , ..., 0.        , 0.        ,
       0.        ]), array([0.00178187, 0.00394281, 0.        , ..., 0.0002135 , 0.00010406,
       0.00016875])]
Time Consumption:  11.461824655532837

(base) C:\DDA\Exc3>
```

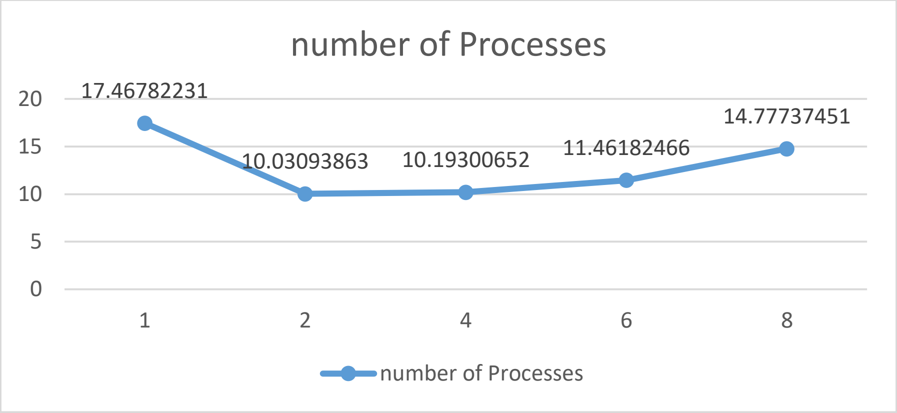
Figure 3: Terminal output

[2] Buffalo university
https://www.cse.buffalo.edu/faculty/miller/Courses/CSE633/Chandramohan-Fall-2012-CSE633.pdf

## Exercise 2.

I calculate time by process zero, because it takes longer than others because it has to do split the data then apply k-means and gather data from others. Graph 1 shows the execution time simulated by k=4 and different number of processes.

As we can see 2 processes will be the best number of processes when k=4 in k means algorithm.



Graph 1: execution time vs number of processor

Generally, when we increase number of processor it does not mean that we are decreasing time execution, sometimes synchronization between processes takes longer than the task we are operating parallel then my result sounds to be true because in figure 2 (left part) we can see that always more processor does not decrease the time execution.

Note: all time execution is simulated by my code, you can run my code and check the result. Furtheremore, I have put comment enough in my code Hope it would be enough.