Fahad Fiaz – (303141) – G2

System Info:

Processor	i7-5500U , 2.40GHz
Cores	4
Operating system	Windows 64 Bit
Ram	8GB
Programming Language	Python 3.7.7

Exercise 1: Implement K-means?

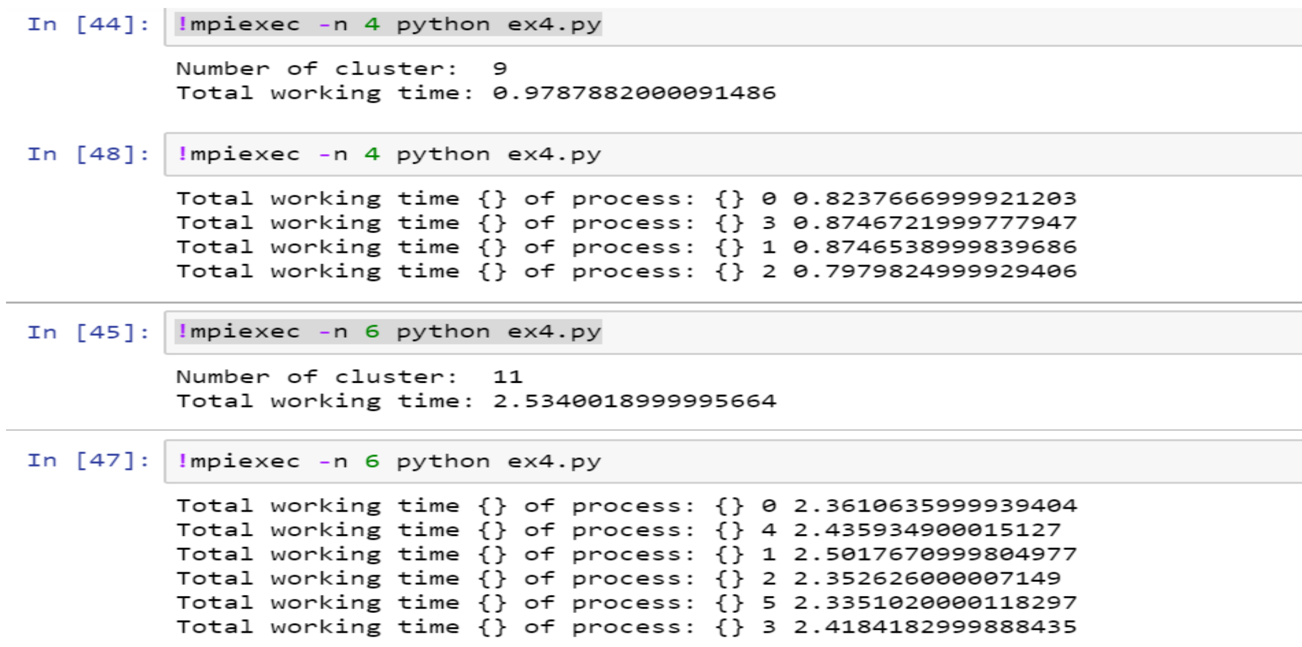
Steps:

1. Consider 1st process as master node and other as slave node.
2. Master process will create a centroid list by randomly generating indexes values, then taking specific rows against these specific indexes from the dataset. Then master node partition the data in equal chunks according to number of processes and scatter these chunks to slave nodes. It also sends centroid to slave processes.
3. Slave processes received chunk of data and centroid from master process. Then they calculated Euclidean distance. Then use this distance to find membership vector.
4. Master process then receive this membership vector from different slave processes and calculate the updated centroid.
5. This process continues until the updated centroid is same as previous centroid.

Time Taken:

P \ cluster	4	8	12
1	4.33	2.75	2.69
2	2.47	2.26	2.65
3	1.77	1.98	2.09
4	1.01	1.40	1.42
6	1.17	0.68	1.42
8	1.58	1.93	3.16

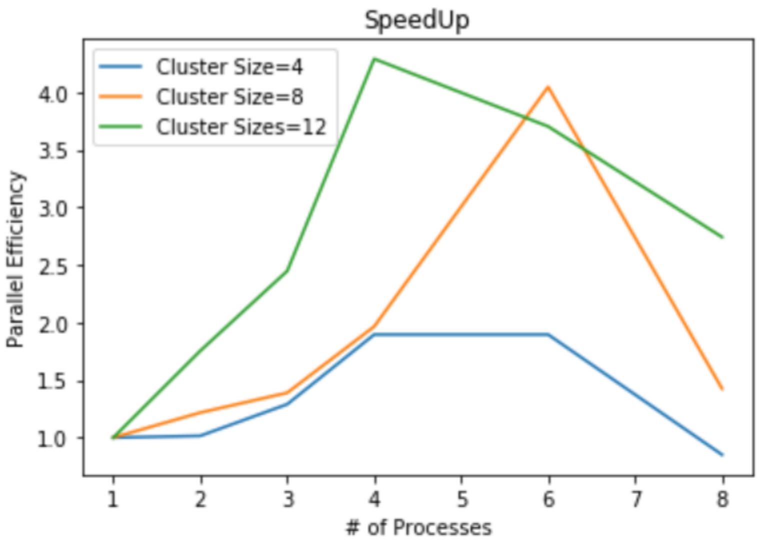
Code working for arbitrary number of clusters and doing parallel processing:



```
In [44]: !mpiexec -n 4 python ex4.py

         Number of cluster:  9
         Total working time: 0.9787882000091486

In [48]: !mpiexec -n 4 python ex4.py

         Total working time {} of process: {} 0 0.8237666999921203
         Total working time {} of process: {} 3 0.8746721999777947
         Total working time {} of process: {} 1 0.8746538999839686
         Total working time {} of process: {} 2 0.7979824999929406

In [45]: !mpiexec -n 6 python ex4.py

         Number of cluster:  11
         Total working time: 2.5340018999995664

In [47]: !mpiexec -n 6 python ex4.py

         Total working time {} of process: {} 0 2.3610635999939404
         Total working time {} of process: {} 4 2.435934900015127
         Total working time {} of process: {} 1 2.5017670999804977
         Total working time {} of process: {} 2 2.352626000007149
         Total working time {} of process: {} 5 2.335102000118297
         Total working time {} of process: {} 3 2.4184182999888435
```

Exercise 2: Performance Analysis?

Speedup is defined as the ratio of serial execution time to the parallel execution time which shows how many times a parallel program works faster than its serial version used to solve the same problem.[1]



The above graph shows super linear speedup till almost 4 processes but after that parallel time efficiency decrease because my system has 4 cores and if the number of processes that we run through mpi4py is more than then available cores then OS has to run multiple processes on one core. In this case there is context switching between processes and it increases the time taken to do our operations.

All the outputs available in file output.html