Parallel Implementation of K-means clustering algorithm

S Srikiruthika & K N Nakshatiraa

# Introduction

K-means algorithm is an iterative approach to partition the given dataset into K different subsets or clusters, where each data point belongs to only one cluster. It assigns data points into a cluster such that the sum of squared distances between data points is minimum. The lesser the variation we have inside the cluster, the more the homogenous clusters we get.

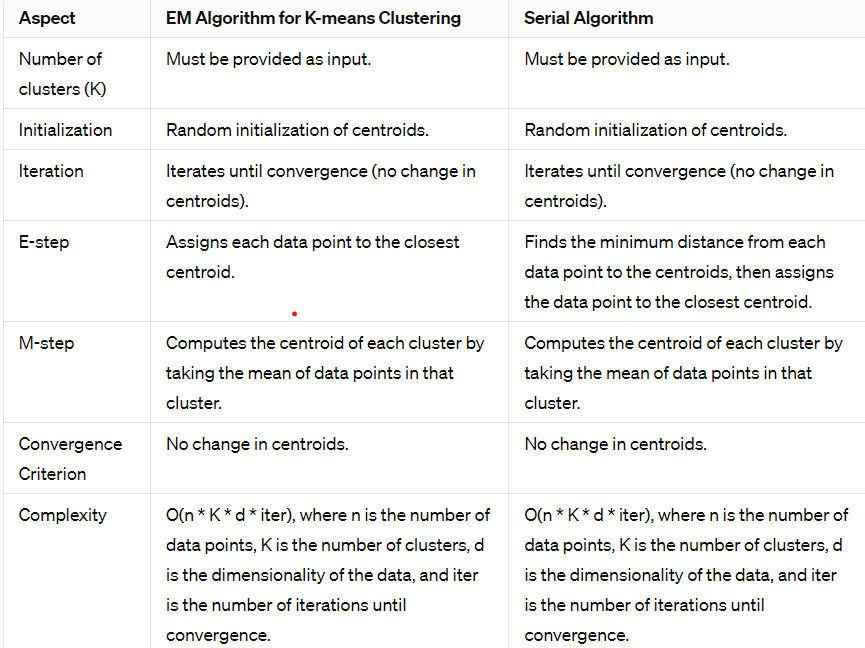
# Serial Algorithm and Complexity Analysis

The general algorithm we follow,

1. Provide the number of clusters K
2. Initialize K random centroids of the clusters from the dataset
3. Keep iterating the following steps until we get no change to the centroid data point.
   1. Find the minimum distance from each data point to the centroids.
   2. Assign the data point to the closest centroid.
   3. Compute the centroid of each cluster by taking the mean of data points in that cluster.

This approach is known as Expectation-Maximization. Where the E-step is assigning different data points to the closest cluster in each iteration.

Whereas M-step is calculating the mean of data points in the cluster after each iteration.



# Scope of parallelism

The clustering algorithm requires massive computations, with distance between each data point and each centroid being calculated. Since calculation of the appropriate centroid for each data point is independent of the others, the algorithm provides a good scope for parallelism.

However, there is a bottleneck. The threads need to communicate among themselves to keep the centroid values updated, as more than one thread might try to access the same centroid point. In that case, it is imperative to ensure that both threads do not try to modify the centroid at the same time, as it might result in corrupted values and false sharing.

# Parallelization strategies

For parallelization of the *k-means* algorithm, a **data-parallelism** approach was adopted. The *N* points were equally split among the number of threads (in case of an imperfect data split among threads, the remainder points are allotted to the last thread)

Key features of the algorithm:

**1.Initialization:** The first *K* data points are chosen as

the initial centroids

**2.Data-parallelism:** Each thread assigned *N/num\_threads*

number of points

**3.Thread function:** Each thread runs a loop (with a *max\_iter* value of 100), in every iteration of which it computes the closest cluster centroid for every point assigned to it, and then assigns the point to that cluster. After every point is assigned to a cluster, the global (shared) cluster centroids are updated with the values computed for their coordinates from the points that were assigned to that cluster. This is again an instance of *data-parallelism*.

**4.** **Stopping-condition:** The *L2-norm* is computed for every cluster centroid coordinates by comparing against corresponding values in the previous iteration. The norms are then summed and compared against the *threshold* value, chosen to be *1e-6*. All threads break from the iterations loop as soon as the *delta* value goes below the *threshold.*

Results and Discussions:

The efficiency curves decrease for every instance run with an increase in the number of threads. This is again due to the diminishing returns property on including an extra thread - a consequence of the **Amdahl’s Law.**

The efficiency increases if the problem size is increased keeping the number of processing elements constant. And this fact is clearly visible for 16 threads.

The efficiency at *num\_threads = 1* goes above 1.0 in certain runs due to superlinear speedups perhaps due to out-of-control system latencies during particular executions.

The efficiency curves demonstrate **best efficiency at the larger-sized datasets** (800,000 and 1 million) and the **worst-performance at smaller-sized ones** (1000)

# References:

1. Inderjit S. Dhillon, Dharmendra S. Modha, in ‘A Data Clustering Algorithm On Distributed Memory Multiprocessors’
2. [https://medium.com/@dilekamadushan/introduction-to-k-means-clust](https://medium.com/%40dilekamadushan/introduction-to-k-means-clust) ering-7c0ebc997e00