K Means Clustering using OpenMP and MPI

Akshay Jain
Information Technology
NITK, Surathkal
akshayjain.191it102@nitk.edu.in

Vasanthram S Joshi
Information Technology
NITK, Surathkal
vasanthramsjoshi.191it155@nitk.edu.in

Anand Kumar
Information Technology
NITK, Surathkal
anand.191it104@nitk.edu.in

***Abstract*—In data mining, clustering is a technique in which the set of objects are assigned to a group called clusters. Clustering is the most essential part of data mining. K-means clustering is the basic clustering technique and is the most widely used algorithm. It is also known as nearest neighbor searching. It simply clusters the data-set into a given number of clusters. Numerous efforts have been made to improve the performance of the K-means clustering algorithm. In this project we have been briefed in the form of a review of the work carried out by the different researchers using K-means clustering. We have discussed the limitations and applications of the K-means clustering algorithm as well. This project presents a current review about the K means clustering algorithm.**

***Index Terms*—component, formatting, style, styling, insert,K-means clustering, nearest neighbor searching, clusters and data mining.**

I. Introduction

K-means clustering is a method of vector quantization, originally from signal processing, K-Means Clustering is an Unsupervised learning algorithm, which groups the unlabeled data-set into different clusters, it aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean. k-means clustering minimizes within-cluster variances.

Given a data set of items, with certain features, and values for these features, the algorithm will categorize the items into k groups or clusters of similarity. To calculate the similarity, we can use the Euclidean distance, Manhattan distance, Hamming distance, Cosine distance as measurement.

II. Literature survey

There are many clustering algorithms for big data which are based on distributed and parallel computation. Mac Queen in 1967 first proposed this technique. The standard algorithm was first proposed by Stuart Lloyd in 1957 as a technique for pulse code modulation. Sometimes it is referred to as Lloyd-Forgy because in 1965, E.W.Forgy published essentially the same method. According to K. A. Abdul Nazeer the major drawback of the k-means algorithm is the selection of initial centroids which produce different clusters. But final cluster quality in algorithms depends on the selection of initial centroids. Two phases included in the original k-means algorithm: first for determining initial centroids and second for assigning data points to the nearest clusters and then recalculating the clustering mean. But this enhanced clustering method uses both the phases of the original k-means algorithm. This algorithm combines a systematic method for finding initial centroids and an efficient way for assigning data points to clusters. But still there is a limitation in this enhanced algorithm that is the value of k, the number of desired clusters, is still required to be given as an input, regardless of the distribution of the data points.

According to Y. S. Thakare, the performance of k-means algorithm which is evaluated with various databases such as Iris, Wine, Vowel, Ionosphere and Crude oil data Set and various distance metrics. It is concluded that performance of k-means clustering is dependent on the data base used as well as distance metrics.

To understand and learn the data, classify those data into remarkable collections. So, there is a need for data mining techniques. R. Amutha proposed that when two or more algorithms of the same category of clustering technique are used then best results will be acquired. Two k-means algorithms: Parallel k/h-Means Clustering for Large Data Sets and A Novel K-Means Based Clustering Algorithm for High Dimensional Data Sets.

Parallel k/h-Means algorithm is designed to deal with very large data sets. Novel K-Means Based Clustering provides the advantages of using both HC and K-Means. Using these two algorithms, space and similarity between the data sets present each node is extended.

III. Sequential program logic (Algorithm)

1. Initialise the clusters

The algorithm needs to start somewhere, so we need to come up with a crude way of clustering points. To do this, we can select first k points or randomly select k points which become ‘centroids’, then assign each datapoint to its nearest centroid. The result of this is k clusters. While this is a naive initialisation method, it does have some nice properties - more densely populated regions are more likely to contain centroids (which makes logical sense).

2. Compute the centroid of each cluster

Technically Lloyd’s algorithm computes the centroid of each partition of 3D space via integration, but we use the reasonable approximation of computing the centre of mass of the points in a given partition. The rationale behind this is that the centroid of a cluster ‘characterises’ the cluster in some sense.

3. Assign each point to the nearest centroid and redefine the cluster

If a point currently in cluster 1 is actually closer to the centroid of cluster 2, surely it makes more sense for it to belong to cluster 2? This is exactly what we do, looping over all points and assigning them to clusters based on which centroid is the closest.

4. Repeat steps 2 and 3

We then repeatedly recompute centroids and reassign points to the nearest centroid. There is actually a very neat proof that this converges: essentially, there is only a finite (though massive) number of possible partitions, and each k-means update at least improves the WCSS. Hence the algorithm must converge.

The Algorithm for the K-Means Clustering is explained in the below steps:

Step-1: Select the number K to decide the number of clusters.

Step-2: Select random K points or centroid. (It can be different from the input data-set).

Step-3: Assign each data point to their closest centroid, which will form the predefined K clusters.

Step-4: Calculate the variance and place a new centroid of each cluster.

Step-5: Repeat the third steps, which means re-assign each data point to the new closest centroid of each cluster.

Step-6: If any reassignment occurs, then go to step-4 else go to FINISH.

Step-7: The clusters are ready.

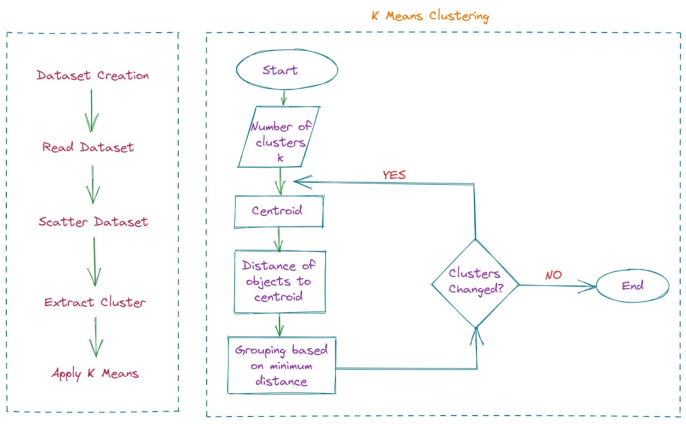


Fig. 1. K Means Clustering Flowchart

For our project, we are using two distance calculation measures :-

1. Euclidean Distance

It is just a distance measure between a pair of samples a and b in an n-dimensional feature space.

2. Cosine Distance

Cosine distance is a measure of the similarity between two vectors based on the cosine angle between them.

$$\text{Point } a : (a1, a2, \ldots, an)$$
$$\text{Point } b : (b1, b2, \ldots, bn)$$

$$\text{Euclidean Distance} = \sqrt{(a1 - b1)^2 + (a2 - b2)^2 + \ldots (an - bn)^2}$$

$$\text{Cosine Distance} = \frac{(a1 * b1) + (a2 * b2) + \ldots + (an * bn)}{\sqrt{(a1^2 + a2^2 + \ldots + an^2) * (b1^2 + b2^2 + \ldots + bn^2)}}$$

Fig. 2. Distances equations

IV. Parallel Program Logic

In above sequential algorithm, Follwing are the steps which we have done in parallel in order to make it execute faseter.

1) MPI has been used to scatter the dataset.
2) MPI has been used to broadcast centroids after extracting cluster.
3) OpenMP has been used to implement the run function of K means clustering to get the centroid id nearest to any point and update its distance.

The detailed explanation of the algorithm can be listed as follows : Suppose we have P processors, where the master node is defined by the Rank 0, and dataset with N points represented in the space with a n-dimensional vectors. Then the master node:

1) Loads the dataset and scatters it among nodes, assigning to each of them N/P points and R Remaining points are assigned to the first R nodes.
2) Reads initial configuration parameters: no. of dimensions, no. of clusters, max iterations.
3) Chooses K points as initial centroids based on the user's choice either first K points or the random K points and broadcast them to the other nodes.

K-means 'loop'

4) In each node the following steps are executed:
   - For each point in the local dataset find the membership among the K clusters. Each point must be in one and only one cluster. Distance calculation between point and centroids is performed in parallel using OpenMP.
   - Within each cluster, sum 'dimension-to-dimension' all the points that belong to that cluster. We obtain a local for each cluster, in each node. [See Fig.3].
5) Once we get the local summations, with MPI Allreduce operation, we can store the sum of the local summations (global sum) in each node. In the same way we can obtain the global number of points in each cluster and store that value in each nodes.
   So to re-calculate a centroid, we can simply divide the global sum of that cluster over the number of points belong to it.
6) Repeat from point 4 until termination.

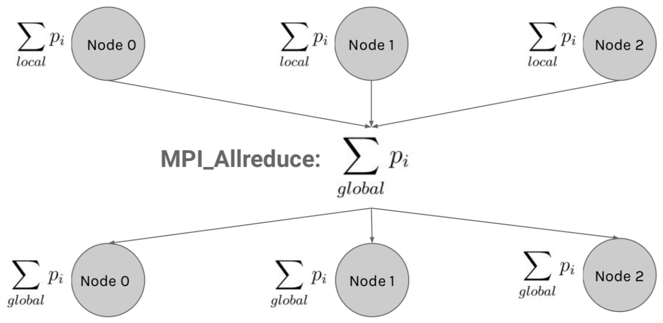


Fig. 3. Operations at each node

Termination Condition

K-means converges when no more point changes its membership status. Since our dataset is distributed among nodes, we cannot know directly if no more changes occurs. To deal with this, we have defined a flag called notChanged in each node that is set when no more changes in the local dataset happen. Those flags are collected by nodes to check if there are changes or not. Furthermore, in order to avoid unnecessary long computation, we have limited the number of iteration that can be executed and user can give the max iteration which they want to run the algorithm for according to their needs.

V. Results

The final working of our implementation consists of three parts

- Building the dataset
- Sequential K-means algorithm
- Parallel K-means algorithm

1. Building the dataset : User has to enter three values namely number of points , dimension of points, number of clusters, and max-iteration and accordingly the data set will be created.

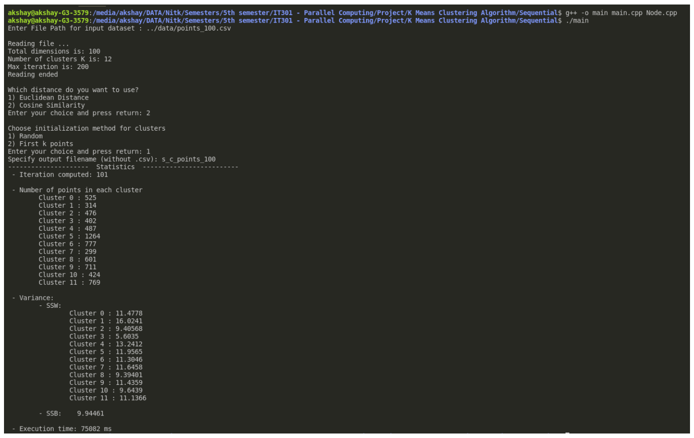
2. Sequential K-means algorithm : The program expects the following input from the user :

- Path to input data-set file
- Distance which we want to consider i.e Euclidean or cosine similarity
- Initialisation method of clusters i.e Random k points or First k points
- Output file name without csv extension

Once all above mentioned inputs are entered properly the sequential program runs and shows various statistics related to data points and clusters as shown in the picture below :

3. Parallel K-means Algorithm : Again the overall user interface is same for parallel algorithm. Just the internal implementation is different, so here also the program expects the following inputs :

- Path to input data-set file

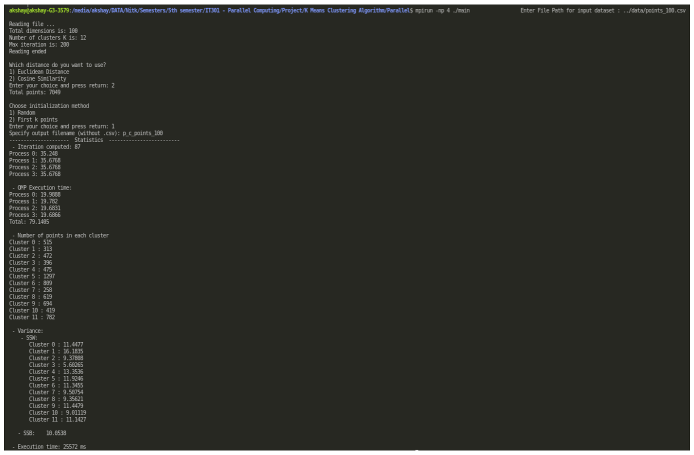


```
akshay@akshay-G3-3579:/media/akshay/DATA/Nitk/Semesters/5th semester/IT301 - Parallel Computing/Project/K Means Clustering Algorithm/Sequential$ g++ -o main main.cpp Node.cpp
akshay@akshay-G3-3579:/media/akshay/DATA/Nitk/Semesters/5th semester/IT301 - Parallel Computing/Project/K Means Clustering Algorithm/Sequential$ ./main
Enter File Path for input dataset : ../data/points_100.csv

Reading file ...
Total dimensions is: 100
Number of clusters K is: 12
Max iteration is: 200
Reading ended

Which distance do you want to use?
1) Euclidean Distance
2) Cosine Similarity
Enter your choice and press return: 2

Choose initialization method for clusters
1) Random
2) First k points
Enter your choice and press return: 1
Specify output filename (without .csv): s_c_points_100
--------------------  Statistics  ------------------------
 - Iteration computed: 101

 - Number of points in each cluster
        Cluster 0 : 525
        Cluster 1 : 314
        Cluster 2 : 476
        Cluster 3 : 402
        Cluster 4 : 487
        Cluster 5 : 1264
        Cluster 6 : 777
        Cluster 7 : 299
        Cluster 8 : 601
        Cluster 9 : 711
        Cluster 10 : 424
        Cluster 11 : 769

 - Variance:
        - SSW:
                Cluster 0 : 11.4778
                Cluster 1 : 16.0241
                Cluster 2 : 9.40568
                Cluster 3 : 5.6035
                Cluster 4 : 13.2412
                Cluster 5 : 11.9565
                Cluster 6 : 11.3046
                Cluster 7 : 11.6458
                Cluster 8 : 9.39401
                Cluster 9 : 11.4359
                Cluster 10 : 9.6439
                Cluster 11 : 11.1366

        - SSB:    9.94461

 - Execution time: 75082 ms
```

Fig. 4. Sequential Program's Statistics

- Distance which we want to consider i.e Euclidean or cosine similarity
- Initialisation method of clusters i.e Random k points or First k points
- Output file name without csv extension

Once all above mentioned inputs are entered properly the Parallel implementation of the program runs and shows various statistics related to data points and clusters which are depicted below :



```
akshay@akshay-G3-3579:/media/akshay/DATA/Nitk/Semesters/5th semester/IT301 - Parallel Computing/Project/K Means Clustering Algorithm/Parallel$ mpirun -np 4 ./main          Enter File Path for input dataset : ../data/points_100.csv

Reading file ...
Total dimensions is: 100
Number of clusters K is: 12
Max iteration is: 200
Reading ended

Which distance do you want to use?
1) Euclidean Distance
2) Cosine Similarity
Enter your choice and press return: 2
Total points: 7049

Choose initialization method
1) Random
2) First k points
Enter your choice and press return: 1
Specify output filename (without .csv): p_c_points_100
--------------------  Statistics  ------------------------
 - Iteration computed: 87
Process 0: 35.248
Process 1: 35.6768
Process 2: 35.6768
Process 3: 35.6768

 - OMP Execution time:
Process 0: 19.9888
Process 1: 19.782
Process 2: 19.6831
Process 3: 19.6866
Total: 79.1405

 - Number of points in each cluster
Cluster 0 : 515
Cluster 1 : 313
Cluster 2 : 472
Cluster 3 : 396
Cluster 4 : 475
Cluster 5 : 1297
Cluster 6 : 809
Cluster 7 : 258
Cluster 8 : 619
Cluster 9 : 694
Cluster 10 : 419
Cluster 11 : 782

 - Variance:
    - SSW:
       Cluster 0 : 11.4477
       Cluster 1 : 16.1835
       Cluster 2 : 9.37808
       Cluster 3 : 5.60265
       Cluster 4 : 13.3536
       Cluster 5 : 11.9246
       Cluster 6 : 11.3455
       Cluster 7 : 9.50754
       Cluster 8 : 9.35621
       Cluster 9 : 11.4479
       Cluster 10 : 9.01119
       Cluster 11 : 11.1427

   - SSB:    10.0538

 - Execution time: 25572 ms
```

Fig. 5. Parallel Program's Statistics

In case of parallel program execution the user has to provide the number of processor they want to use for the MPI while giving the command for running the parallel algorithm. In MPI we start taking timings after the I/O operation of data placement into the nodes. I/O operation of data placement is much more than computational time.

As shown in the two figures the execution time for parallel implementation for 7049 points and 100 dimensions gives approximately 3 time faster execution time than sequential execution this can further make a huge time difference when number of points becomes too large.

VI. Conclusion

K means clustering algorithm is not so easy to parallelize as it requires frequent communication between each nodes. As it needs frequent communication between the nodes using only the MPI would not give expected speedup because of communication overhead. We used the hybrid model of MPI and OpenMP to get better performance.

The loop where each point is assigned membership to right cluster is threaded in a particular node is being done using the OpenMP and rest other things we did using MPI as mentioned earlier.

Here We used OpenMP and MPI to accomplish the task of parallel computing. We have implemented the parallel execution of K Means Clustering which is a Machine Learning Algorithm used to classify n data items into k clusters using some distance metrics. The two distance metrics we considered are Euclidean Distance and Cosine Distance. Data set Generation to execute the project is also done separately. At the end, we are comparing the time taken by sequential implementation of K-Means with our parallel implementation of K-Means.

References

[1] ”Algorithms Sequential & Parallel: A Unified Approach” – Russ Miller and Lawrence Boxer

[2] “Parallel k-Means Clustering for Quantitative Ecoregion Delineation using Large Data Sets” – Jitendra Kumara, Richard T. Millsa, Forrest M. Homana, William W. Hargroveb

[3] “Parallel k/h-Means Clustering for Large Data Sets” - Kilian Stoel and Abdelkader Belkonieneiene

[4] K-means Clustering Documentation

[5] Euclidean distance Documentation

[6] Cosine similarity Documentation

[7] MPI Resources

[8] OpenMP Resources