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 method wherein the set of gadgets are assigned to a set known as clusters. Clustering is the most critical part of records mining. K-manner clustering is the primary clustering method and is the maximum extensively used set of rules. It is likewise called nearest neighbor searching. It really clusters the datasets right into a given wide variety of clusters. Numerous efforts were made to enhance the overall performance of the K-manner clustering set of rules. In this paper we were briefed withinside the shape of an overview of the paintings finished via means of the special researchers the use of K-manner clustering. We have mentioned the restrictions and packages of the K-manner clustering set of rules as well. This paper gives a contemporary overview approximately the K manner clustering set of rules. Keywords: K-manner clustering, nearest neighbor searching, clusters and data mining.**

***Index Terms*—**

I. Introduction

K-means Clustering is a vector quantization approach that originated in signal processing. K-Means Clustering is an unsupervised learning process that divides an unlabeled data set into groups. It attempts to split n observations into k clusters, with each observation assigned to the cluster with the closest mean. Within-cluster variances are minimised using k-means clustering.

In current years there was a great boom withinside the extent of information. To draw significant insights from this mountain of information we want algorithms that can carry out evaluation in this information. Clustering is the technique of grouping of information or dividing big information set into smaller information units of a few similarity in order that the gadgets withinside the equal cluster are extra much like every different and extra exceptional from the gadgets withinside the different group. Clustering is an critical evaluation approach this is hired to big datasets and unearths its software withinside the fields like seek engines, advice systems, information mining, understanding discovery, bioinformatics and documentation. Nowadays, the information being generated isn't handiest large in extent, however is likewise saved throughout numerous machines everywhere in the world. We want to technique this information in parallel to lessen the price of processing. But clustering isn't unfastened from weaknesses such that the person has to specify the quantity of clusters to be generated earlier than the begin of the set of rules, which may be very hard mainly while we're managing large information. Another trouble which might also additionally rise up in making use of the K-Means clustering set of rules is the decision trouble, which might be over-decision or beneath neath-decision. Over-decision is a kingdom while the very last quantity of clusters generated is extra than the real quantity of clusters and beneath neath decision is the communicate of over-decision.

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. The Euclidean distance, Manhattan distance, Hamming distance, and Cosine distance may all be used to calculate similarity.

Parallel computing is presently as much a portion of everyone's life as individual computers, keen phones, and other innovations are. You clearly get it, since you've got set out upon the MPI Instructional exercise site. Whether you're taking a lesson around parallel programming, learning for work, or basically learning it since it's fun, you have got chosen to memorize an ability that will stay fantastically profitable for a long time to come.

OpenMP is a collection of compiler instructions and an API for C, C++, or FORTRAN applications that allows for parallel programming in shared-memory settings. Parallel districts are identified by OpenMP as portions of code that will execute in parallel. Application designers embed compiler orders into their code at parallel regions, and these orders educate the OpenMP run-time library to execute the locale in parallel.

In addition to giving directives for parallelization, OpenMP allows developers to select between several levels of parallelism. They can choose the number of threads manually, for example. It also enables developers to determine if data is shared between threads or is thread-specific. For Linux, Windows, and Mac OS X, OpenMP is supported by a number of open-source and commercial compilers.

You've also made the right step toward improving your parallel programming skills by understanding the Message Passing Interface, in our opinion (MPI). Although MPI is more basic than most parallel programming tools (such as Hadoop), it is a wonderful place to start learning about parallel programming.

Before the 1990s, programmers didn't have the same opportunities as us. It was a tough and time-consuming process to write parallel programmes for various computing systems. Many libraries existed at the time that may aid in the development of parallel applications, but there was no universally approved method for doing so.

At the time, the majority of parallel applications were in the disciplines of science and research. The message passing paradigm was the most popular among library users. What is the model for passing messages? It simply means that a programme sends messages to other processes in order to complete a task. In practise, this paradigm works effectively for parallel applications.

II. Literature survey

There are a variety of big data clustering algorithms that are based on distributed and parallel computation. Mac Queen suggested this strategy for the first time in 1967.

In 1957, Stuart Lloyd devised the standard technique as a pulse code modulation mechanism. It is frequently referred to as Lloyd-Forgy since E.W.Forgy presented an approach that is essentially the same in 1965.

According to K. A. Abdul Nazeer, the k-means algorithm's main flaw is the choice of initial centroids, which results in distinct clusters. In algorithms, however, the quality of the final cluster is dictated by the initial centroids chosen. The original k-means algorithm included two phases: the first was used to create initial centroids, while the second was used to allocate data points to the nearest clusters and recalculate the clustering mean. This improved clustering approach, on the other hand, employs both phases of the original k-means algorithm. This method combines a systematic method for selecting initial centroids with an efficient method for allocating data points to clusters.

Regardless of the data point distribution, the amount of k, or the number of desired clusters is still required to be given as an input in this upgraded algorithm.

According to Y. S. Thakare, the performance of the k-means algorithm is evaluated using a variety of databases and distance measures, including Iris, Wine, Vowel, Ionosphere, and Crude oil data sets. The results show that the performance of k-means clustering is influenced by the data base and distance metrics utilised.

Classify the data into noteworthy collections to better comprehend and study it. As a result, data mining techniques are required. When two or more algorithms from the same category of clustering approach are employed, the best results are obtained, according to R. Amutha. Parallel k/h-Means Clustering for Large Data Sets and A Novel K-Means Based Clustering Algorithm for High Dimensional Data Sets are two k-means algorithms.

The parallel k/h-Means algorithm is optimised for huge data sets. The advantages of employing both HC and K-Means are combined in Novel K-Means Based Clustering. Space and similarity between the data sets present in each node are enhanced using these two techniques.

III. Sequential program logic (Algorithm)

1. Cluster initialisation

As the algorithm must begin somewhere, we must devise a primitive method of clustering points. To do so, we can choose k points or pick k points at random to serve as 'centroids,' and then assign each datapoint to the centroid closest to it. As a result, there are k clusters. While this is a rudimentary method of initialization, it has certain benefits: Centroids are more prone to appear in heavily populated places (which makes logical sense).

2. Compute the centroid of each cluster

The centroid of each 3D partition is calculated by integration in Lloyd's method, however we use the reasonable approximation of identifying the centre of mass of the points in a particular partition. This is because the cluster's centroid 'characterises' it in some way.

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

Isn't it more logical to put a point in cluster 1 in cluster 2 if it's closer to the centroid of cluster 2? We do just that, cycling over all locations and allocating them to clusters based on the centroid that is closest to them.

4. Repeat steps 2 and 3

We then recalculate centroids and reassign locations to the nearest centroid many times. There's a tidy argument that this converges: there are only a finite (albeit vast) number of potential partitions, and each k-means update improves the WCSS at the very least. Hence the algorithm must converge.

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

Step 1: To determine the number of clusters, choose the number K.

Step 2: Choose a random set of K points or the centroid. (It may or may not be the same as the input data-set.)

Step 3: Assign each data point to the centroid that is closest to it, forming the preset K clusters.

Step 4: Determine the variance and reposition each cluster's centroid.

Step 5: Re-assign each data point to the cluster's new nearest centroid by repeating the third steps.

Step-6: If there is a reassignment, go to step-4; otherwise, move to FINISH.

Step 7: The clusters are now complete.

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

The cosine distance between two vectors is a measure of their similarity based on their cosine angle.

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.

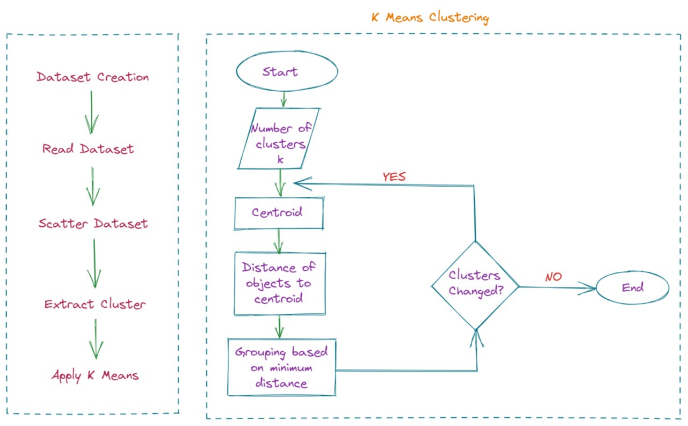


Fig. 1. K Means Clustering Flowchart

Point a : (a1, a2, ..., an)
Point b : (b1, b2, ..., 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

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 following is a full explanation of the algorithm:

Assume we have P processors, with Rank 0 as the master node, and a dataset with N points represented in space by n-dimensional vectors. The master node follows:

1) Loads the dataset and distributes it across nodes, assigning N/P points and R to each. The first R nodes receive the remaining points.
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:
   - Find the membership of the K clusters for each point in the local dataset. Each point must belong to a single cluster. The distance between the point and the centroids is calculated in parallel using OpenMP.
   - Sum all the points that belong to that cluster 'dimension-to-dimension' inside each cluster. In each node, we receive a local for each cluster. [See Fig.3].
5) We may use MPI Allreduce to store the total of the local summations (global sum) in each node after we get the local summations. We can also get the total number of points in each cluster and save that value in each node in the same way.
   To recalculate a centroid, just divide the cluster's global sum by the number of points that belong to it.
6) Repeat from point 4 until termination.

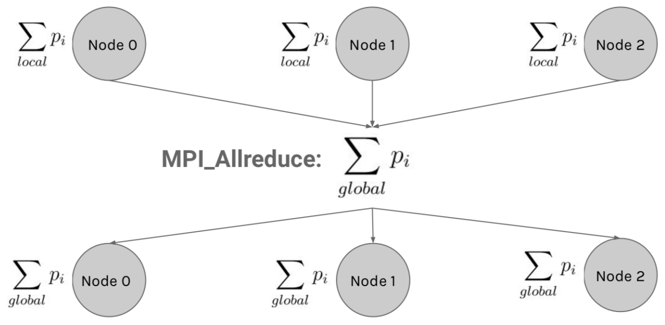


Fig. 3. Operations at each node

Termination Condition

When no more points alter their membership status, K-means converges. We can't tell whether there are no more modifications because our dataset is scattered among nodes. To deal with this, each node has a flag called notChanged that is set when the local dataset does not change any more. Nodes gather these flags to see whether there have been any modifications. Furthermore, in order to avoid excessively long computations, we have restricted the number of iterations that may be performed, and the user can choose the maximum number of iterations for which the algorithm should be conducted.

V. Results

The final working of our implementation consists of three parts

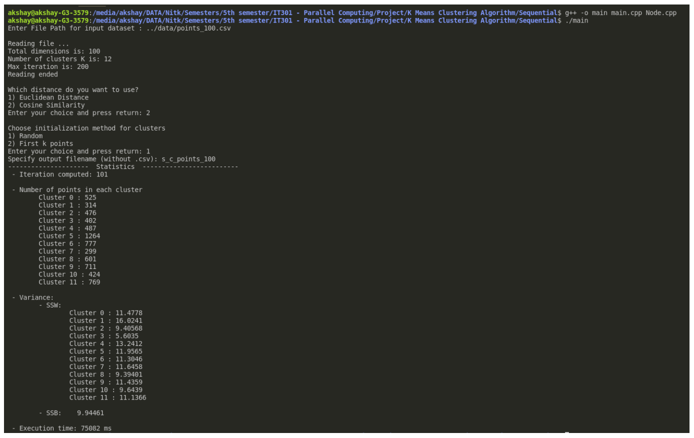
- Building the dataset
- Sequential K-means algorithm
- Parallel K-means algorithm

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 :



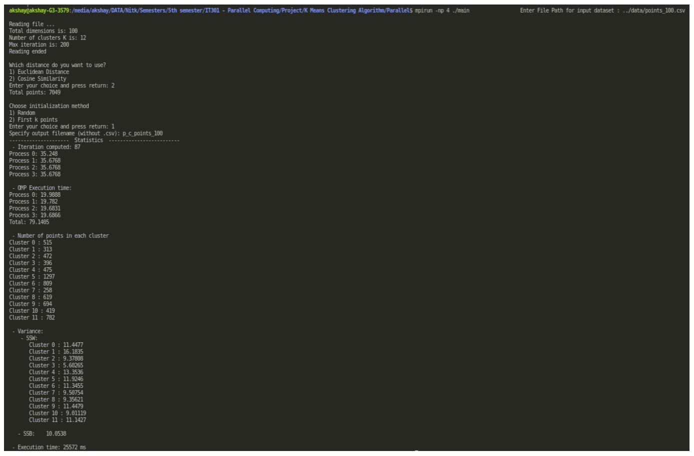
```
akshay@akshay-G3-3579:/media/akshay/DATA/Nitk/Semesters/5th semester/IT301 - Parallel Computing/Project/K Means Clustering Algorithm/Sequential$ g++ -o main main.cpp Node.cpp
akshay@akshay-G3-3579:/media/akshay/DATA/Nitk/Semesters/5th semester/IT301 - Parallel Computing/Project/K Means Clustering 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

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
- 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 :



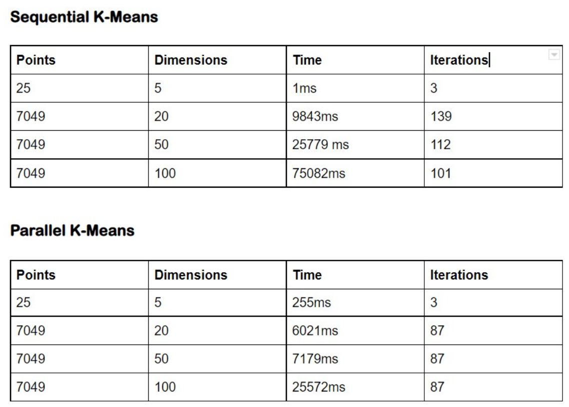
```
akshay@akshay-G3-3579:/media/akshay/DATA/Nitk/Semesters/5th semester/IT301 - Parallel Computing/Project/K Means Clustering Algorithm/Parallel$ mpirun -np 4 ./main          Enter File Path for input dataset : ../data/points_100.csv

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. After the I/O operation of data insertion into the nodes, we start taking timings in MPI. The time it takes to do an I/O operation for data placement is substantially longer than the time it takes to perform a computation.

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.

Time and iteration count is being shown in the picture below :



Sequential K-Means

Points	Dimensions	Time	Iterations
25	5	1ms	3
7049	20	9843ms	139
7049	50	25779 ms	112
7049	100	75082ms	101

Parallel K-Means

Points	Dimensions	Time	Iterations
25	5	255ms	3
7049	20	6021ms	87
7049	50	7179ms	87
7049	100	25572ms	87

Fig. 6. Comparison Of Parallel and Sequential K-Means

VI. Conclusion

The K means clustering technique is difficult to parallelize because it necessitates frequent communication among nodes. Because frequent contact between the nodes is required, relying just on the MPI would not provide the promised speedup due to communication cost. To improve performance, we employed a hybrid model combining MPI and OpenMP.

The loop in which each point is assigned membership to the correct cluster is threaded in a specific node is done using OpenMP, and the remainder of the work was done with MPI, as previously described.

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. Euclidean Distance and Cosine Distance are the two distance measures we evaluated. 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