Implementation of Parallelized K-means and K-Medoids++ Clustering Algorithms on Hadoop Map Reduce Framework

Maithri .C, Chandramouli .H

Abstract: ***The electronic information from online newspapers, journals, conference proceedings website pages and emails are growing rapidly which are generating huge amount of data. Data grouping has been gotten impressive consideration in numerous applications. The size of data is raised exponentially due to the advancement of innovation and development, makes clustering of vast size of information, a challenging issue. With the end goal to manage the issue, numerous scientists endeavor to outline productive parallel clustering representations to be needed in algorithms of hadoop. In this paper, we show the implementation of parallelized K-Means and parallelized K-Medoids algorithms for clustering an large data objects file based on MapReduce for grouping huge information. The proposed algorithms combines initialization algorithm with Map Reduce framework to reduce the number of iterations and it can scale well with the commodity hardware as the efficient process for large dataset processing. The outcome of this paper shows the implementation of each algorithms.***

Keywords : Big Data, Clustering algorithms, Hadoop, K-means, K-Medoids, K-Medoids++, MapReduce.

I. INTRODUCTION

Clustering of large data is a common problem in data mining and computational geometry. It enables to group a set of multidimensional points into different classes so that different multidimensional points with inter-point similarity and intra-point difference can be classify into the same cluster. In scientific data analyses the basic requirement is scalability and performance, to attain this requirements various parallel clustering algorithms[1,2,3] and its implementation techniques are used.

The major disadvantages of parallel clustering algorithms are:

1. They reckon that at the same time all the objects can be placed in the main memory.
2. The parallel systems itself adopted an restricted programming models. This restricted mode on parallelism made the processing of the large data an falloff mode in automatically computing systems.

The MapReduce [4,5] is a programming framework defined on hadoop system. This was introduced by Google. The MapReduce technique facilitates the processing of data records in large data files on clusters of computers. This framework was used by Google first for the purpose of serving Web page indexing service on Google search engine. The MapReduce framework is found beneficial by the beginner developers as this contain library routines which supports the creation of parallel programs by never considering the impact of cluster communication, Job scheduling and monitoring or failure handling processes. MapReduce is an highly scalable programming paradigm that run on a large clusters of computers in an optimized manner [6,7,8,9].

The K-Means Algorithm is an iterative method that does the clustering of data which are defined as a set of n points into k clusters or groups of points. This approach is suitable when the dataset is present before the processing to the information. But, the algorithm fails when the data set varies and increases during real time. This results in the mis-processing of the dataset and false results on the information to be extracted from the data set. So, the K-Means algorithm is need be re-defined to have the knowledge of upcoming data set and need to adopt of online data. In this paper, an implementation for the K-Means algorithm is introduced which is capable of extending the processing step to the cluster and take new dataset for processing.

The K-Means algorithm for clustering is proposed [10,11] in different research areas independently are very variant and sensitive to outers noise. As a result, the K-Medoid++ algorithm was proposed which is more better solution than K-Means, where the medoids are used to replace the mean of the multidimensional points in the same class. The replacement rule of the medoid of a specific cluster is based on the cost. The non-medoid with the less cost will replace the current medoid which makes K-Medoid less sensitive to noise of the outers.

As for K-Medoids++ algorithm, the performance is good when processing small size data set but the performance will decrease when the scale of the spatial data is increased[12]. The K-Medoids++ clustering algorithm is time consuming while processing massive two dimensional points though it is robust against outers. As a result, the more efficient K-Medoids++ clustering algorithm should be proposed and this paper pays attention on this issue.

II. PRELIIMINARIES

In this section, the fundamental concepts of Hadoop, Mapreduce, K-Means and K-Medoids++ are given.

Revised Manuscript Received on December 15, 2019.

***** Correspondence Author

Maithri .C*, Associate Professor, Dept. of CSE, Kalpataru Institute of Technology, Tiptur, INDIA, Email: maithri.c.prashanth@gmail.com

Dr. Chandramouli .H, Professor, Dept. of CSE, East Point College of Engineering and Technology Bangalore, INDIA, Email: hcmcool123@gmail.com

A. MapReduce

MapReduce is a distributed and an efficient scheduling model for task resource allocation originally from functional programming language. The main ideology of this model is using map and reduce functions to spilt and combine the data. A MapReduce program is consists of a Map function which is designed for sorting and filtering the input data and a Reduce function which is designed for summing up the data. The MapReduce framework is designed for managing the distributed servers and running massive computing jobs in parallel. It manages all data transfers between different tasks of the framework with redundancy and high fault tolerance. The basic programming model of MapReduce architecture is demonstrated in Fig. 1.

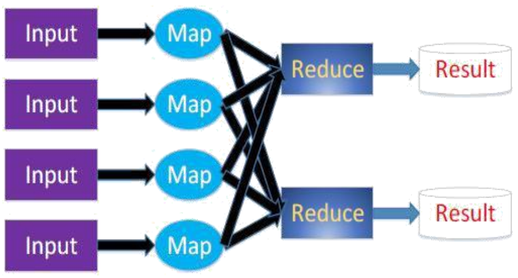


Fig.1: Basic programming model of MapReduce.

Map-function

The HDFS [13] stores input dataset as a sequence of record which is in the format (key, value) pair, which is called as the record of one dataset. The start point of the data file is represented with the key which will be the offset of record. The record content in the data file will be represented as a value. The map function will read this content to split the data file.

Combine-function

The combiner is applied on the spitted data of same map function. This is done after the completion of the map function execution. This function will not have any communication overhead since the intermediate data will be available in the datanode itself. The combine function performs an add operation on all the values of the data objects assigned in the same cluster. This function determines the mean value for each cluster and keeps the recording of samples in the same datanode.

Reduce-function

After the completion of the combine function execution. Each datanode will contain the partial sum in the data objects as the samples in the same cluster. The reduce function performs the addition of samples and calculates the total number of sample data present in the same cluster. Thus, the reduce function provides the new centroid of the samples which will be used in next iteration. The algorithm for the reduce function is shown in Algorithm 3.

B. Hadoop and HBase

MapReduce in Hadoop system is a software programming framework which works of distributed environment for processing the huge datasets on a cluster of computer machines. The basic feature of Hadoop is that it can automatically handle the hardware failure so that the cloud computing clusters can be built on commodity hardware.. HDFS is used for data storage and distributing the data into different blocks in different clusters. HDFS ensures the high fault tolerance by copying the data blocks into different nodes and the data locality information in HDFS is used for the communication between the nodes.

A typical instance of Hadoop system contain a one namenode and several datanodes. The namenode consists of a JobTracker, TaskTracker, ResourceManager, and SecondaryNamenode. The JobTracker which is the service for assigning the jobs to the datanodes with the data. The Task Tracker which is a node on cluster accepts the jobs from the JobTracker. A datanode mostly act as a TaskTracker and DataNode. NameNode is the control node and DataNode is the work node of HDFS.

C. K-Means

K-means algorithm for clustering is most widely used in most of the application that are developed for scientific labs and industrial applications. This is a method of clustering technique, in which the dataset with n objects is divided into k clusters in which each object belongs to the one of the k clusters with the nearest mean value.

The K-Means algorithm is very simple and straightforward:

- Select the value of k as the initial set of centroids.
- Repeat step 3 and 4 for all data records in dataset.
- Calculates the nearest point from that centroid list for the given Dataset.
- Create K cluster by adding each point to its nearest centroid.
- Find the new global centroids for all clusters using the equations.

The basic properties of K-means algorithm are:

- Efficient for processing big data files with proper hardware architecture.
- It works using the numeric values for data and computes the centroid as numbers only.
- The shapes of clusters generated will always looks as a convex structure.

Amira Boukhdhir et. al. have [14] proposed an improved version of K-Means. This algorithm works on very large dataset. The algorithm shows the execution time comparison over the implementation of traditional KMeans, Parallel KMeans and Fast KMeans algorithms. This algorithm has removed the outlier for the numerical. This Map Reduce technique used to for generating initial centroids and forming the clusters. But, this implementation has limitations that the value of the number of centroids must be given as input by user. This algorithm is design only to work with numerical data. This algorithm determines the number of clusters also numbers of clusters are not determined automatically.

Van Hieu D. and Meesad P. [15] has proposed an algorithm of KMeans for reducing execution time. They implemented the KMeans by cutting off the last iterations defined in K-Means. This algorithm reduced the number of iteration to 30%, so 30% of executing time is reduced, and accuracy is high. The main drawback is the selecting the initial centroids randomly makes the cluster instable cluster. The outcome of this execution will adds the noise points. Thus this technique may produce inaccurate result.

Li Ma et. al [16] has proposed an solution for traditional KMeans and shown improvement in the quality of standard KMeans clustering technique. The technique developed is of selecting the k centroid values systematically with the number of clusters based on the initial centroid point defined. This algorithm also reduced the noise points inclusion an solved the outliers problem. This algorithm has produced quality clusters. But this algorithm requires more computation time.

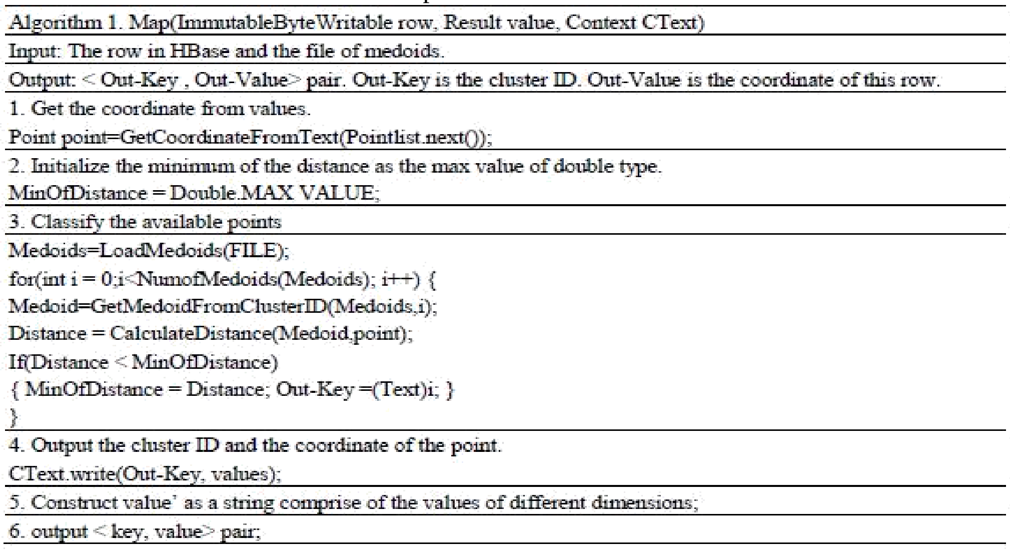
D. K-Medoids

The K-Means clustering is an popular clustering algorithm defined and used, due to its excellent performance. However, it is easily influenced by the outliers, the K-Medoid algorithm is proposed by using the medoids which are the elements in the date set as the center of the clusters. It attempts to minimize the cost between non-medoids of the cluster and the medoid of the cluster.

K-Medoids algorithm, was proposed by Kaufman and Rousseeuw [17] which is explained step by step as:

1. Select Data points k in the dataset arbitrarily as the initial medoids.
2. Classify each non-medoid or data point which is nearest to the medoid point based on computed Euclidean distance or cosine distance.
3. Calculate the new medoid with the minimum cost for each cluster.
4. Continue Step (2) until the total cost remain the same.

The K-Medoid algorithm works effectively while clustering datasets with small size, but it doesn‘t support for massive datasets due to the internal complexity. Hence, the paralleled K-Medoids++ algorithm is proposed with the effective initialize algorithm.



```
Algorithm 1. Map(ImmutableByteWritable row, Result value, Context CText)
Input: The row in HBase and the file of medoids.
Output: < Out-Key , Out-Value> pair. Out-Key is the cluster ID. Out-Value is the coordinate of this row.
1. Get the coordinate from values.
Point point=GetCoordinateFromText(Pointlist.next());
2. Initialize the minimum of the distance as the max value of double type.
MinOfDistance = Double.MAX VALUE;
3. Classify the available points
Medoids=LoadMedoids(FILE);
for(int i = 0;i<NumofMedoids(Medoids); i++) {
Medoid=GetMedoidFromClusterID(Medoids,i);
Distance = CalculateDistance(Medoid,point);
If(Distance < MinOfDistance)
{ MinOfDistance = Distance; Out-Key =(Text)i; }
}
4. Output the cluster ID and the coordinate of the point.
CText.write(Out-Key, values);
5. Construct value' as a string comprise of the values of different dimensions;
6. output < key, value> pair;
```

III. METHODOLOGY

A. K-Means

Currently, data sets are generated by Wikipedia, telecommunications systems, and sensors looks to be so large, that standard KMeans clustering algorithms fails to execute in iterative manner over such data to obtain the meaningful information. Thus, there is a necessity to enhance this clustering algorithm to suit large datasets processing. Hadoop is a platform to store big datasets which are in petabytes in size. Hence, the present work is to develop an parallel KMeans algorithm that is defined using Map-Reduce model over the Hadoop Framework.

Through this proposal, an enhancement for the KMeans algorithm called as Parallel KMeans is introduced which is capable of extending the processing of huge dataset to form the cluster and continue the processing of the incoming dataset. This algorithm implemented as the following outlined steps that is performed:

1. Arbitrarily select C which is the centroid of data records in cluster.
2. select the threshold value for convergence by which centroids will not have the data records that is more than this distance. No further iterations are done for the formulation of cluster.
3. Calculate the distance between the centroid and data object using Euclidean or Cosine distance algorithm for each data point and cluster center.
4. Add data points to its nearest cluster center.
5. After completion of add all the data points to the cluster centroid then the new centroid is computed with convergence threshold value using

$$V_i = \left(1/_{D_i}\right)\sum_{j=1}^{c_i}(C_j)$$

Where D_i is the number of data points in the ith cluster, C_j is the cluster centers in jth cluster center. The V_i is the new cluster centroid created for C_j cluster and are set of data points.
6. Again recompute the distance for each data point to new centroid and assign them into a particular cluster.
7. All the data point is reassigned to the cluster centroid list until the convergence threshold is met and the step 3 is performed otherwise.

Function Mapper for Hadoop

For the map function, the input data file is stored on HDFS in the form of a list of records in a line by line which contain the (key, value) pairs. Map function is loaded into one or more datanodes for the parallel execution. The datanodes that execute map function are called as Mappers. The input data file is a large number of records which are not have any arrangement and relevance. This file is defined as unsupervised input dataset. The input dataset is spitted randomly and each split is distributed as input records for Mapper. The mapper executes the defined operation of finding out the data object nearer to the centroid using the mathematical distance function such as Euclidean or cosine distance is a proximity measure. The process is repeated for calculating the distance of data point with each of the centroid. Then, the data point is added to the group of centroid with the closest distance. Once all the data points are added to each centroids. The mapper will executes an sorting of the data object and regrouping the data objects to formulate the cluster group in that mapper. This sorting and formulation of cluster group is performed in the Reducer. Algorithm for the Mapper of Hadoop is shown in Algorithm 2.

Algorithm 2: Mapper

```
Input: Data objects set D = {$d_1$, $d_2$,….., $d_n$}, where n>0
Centroid set C = {$c_1$, $c_2$, ... ,$c_m$} where m>0 and m<=n
Output: set of data as ($C_i$, $D_j$) where $1 \le i \le n$ and $1 \le j \le m$
Procedure
MD←{$d_1$,$d_2$,………,$d_n$};
centroids←{$c_1$,$c_2$,..., $c_m$};
P is distance measure which are computed using the Distance
Measure algorithms such as Euclidean or Cosine algorithm.
(where pi is the coordinate of p in dimension i )
for all $x_i$є MD such that 1≤i≤n do
    bestC←null
    dist_min←∞
    for all $c_j$єcentroids such that 1≤j≤n do
        dist← distance ($x_i$, $c_j$)
        if bestC = null || dist<dist_min then
            dist_min←dist
            bestC ← cj
        end if
    end for
output<< (bestC, xi)
i+=1;
end for
return output
```

Function Reducer for Hadoop

The output of the Mapper is stored in the intermediate files for the processing in reduce function. Again, the reduce function is loaded in the datanodes to perform parallel execution on the intermediate data file. This datanode is now called as reducer. The reducer will accepts the intermediate data file in the form of a records of <centroid, data object> pairs and performs the Grouping function, then sorts the groups and finally calculates new centroid values. For every centroid, the Reducer will find a new value for centroid which is based on datapoint that are get assigned to it in that iteration. This is done by finding the mean of data points in each cluster. The computed new centroid list is stored as the reducer output. This algorithm is explained in Algorithm 3.

Algorithm 3: Reducer

```
Input: (centroid, data object) where centroid is the new key
generated in Mapper, and value is the data points assigned by
Mapper.
Output: (centroid, data object) where key = centroid and value
= new_centroid which is the new_centroid value calculated
for that centroid.
Procedure
    output←output produced from Mappers
    V← { }
    newCentroids ← null
    for all B ∈ records in output do
        centroid ←B.key
        object ←B.value
        V[centroid] ← object
    end for
    for all Mapper generated centroids ∈ V do
        newCentroid← null
        Objects_sum← null
        Objects_num← null
    end for
    for all object ∈ V[centroid] do
        Objects_sum += object
        Objects_num += 1
    end for
    newCentroid ← (Objects_sum / Objects_num)
    newCentroids << (newCentroid)
    return newCentroidList
```

B. K-Medoids

Initialization algorithm of K-Medoids++

The K-Medoids algorithm finds cluster medoids by minimizing the intra-class discrepancy. However, the k-Mediods algorithm has the following theoretical drawbacks.

- The number of medoids should be given in advance. However, the number of medoids is hard to determine in many cases.
- The initial medoids should be selected manually which will increase the number of iterations.

The proposed k-Mediods++ algorithm addresses the second concern by using the initialization algorithm proposed in Ref. [8] to decrease the number of iterations. The explicit algorithm is as follows.

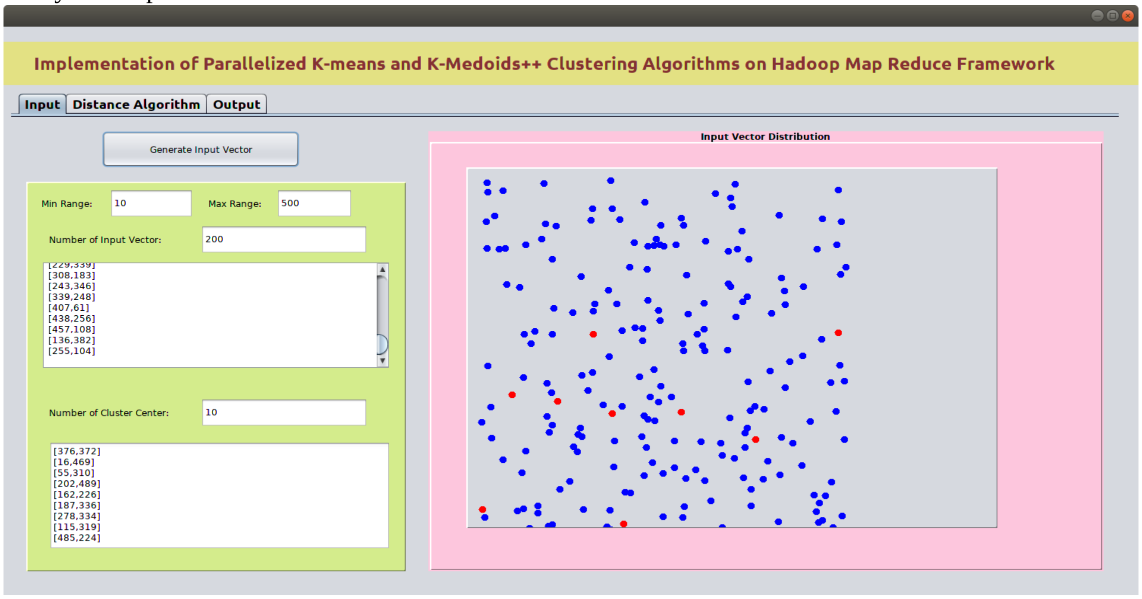
- The initial medoid is chosen randomly among all of the spatial points.
- For each spatial point, compute the distance between and the nearest medoids which is termed as D( ) and sum all the distances to.
- The next medoid is determined by using weighted probability distribution. Specifically, a random number between zero and the summed distance is chosen and the corresponding spatial point is the next medoid.

Step (2) and Step (3) are repeated until medoids have been chosen.

MapReduce Data Flow Units:

The basic algorithm for K-Medoids++ spatial clustering can be summarized to three steps as follows:

- Generating points as initial medoids using the initialization algorithm.
- Map the Key-Value pairs.
- The keys of output are identifiers of clusters and the values of the output are the coordinates of spatial points. The Reduce key, value pairs.


Implementation of Parallelized K-means and K-Medoids++ Clustering Algorithms on Hadoop Map Reduce Framework
Input
Distance Algorithm
Output
Generate Input Vector
Input Vector Distribution

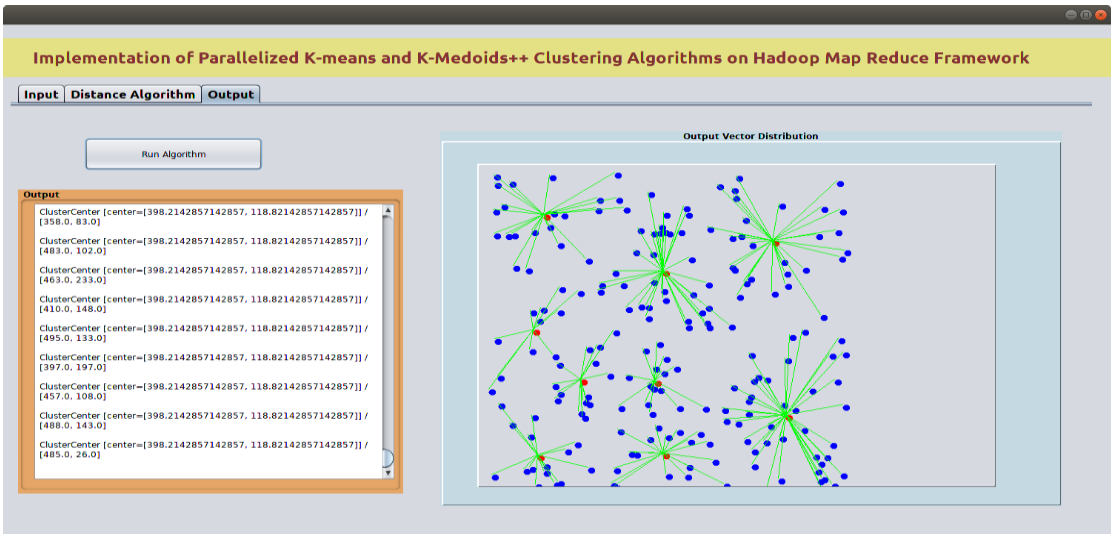
Min Range: 10 Max Range: 500
Number of Input Vector: 200

```
[229,339]
[308,183]
[243,346]
[339,248]
[407,61]
[438,256]
[457,108]
[136,382]
[255,104]
```

Number of Cluster Center: 10

```
[376,372]
[16,469]
[55,310]
[202,489]
[162,226]
[187,336]
[278,334]
[115,319]
[485,224]
```

Fig 2: The process of input data set generation.


Implementation of Parallelized K-means and K-Medoids++ Clustering Algorithms on Hadoop Map Reduce Framework
Input
Distance Algorithm
Output
Run Algorithm
Output Vector Distribution

Output

```
ClusterCenter [center=[398.2142857142857, 118.82142857142857]] /
[358.0, 83.0]

ClusterCenter [center=[398.2142857142857, 118.82142857142857]] /
[483.0, 102.0]

ClusterCenter [center=[398.2142857142857, 118.82142857142857]] /
[463.0, 233.0]

ClusterCenter [center=[398.2142857142857, 118.82142857142857]] /
[410.0, 148.0]

ClusterCenter [center=[398.2142857142857, 118.82142857142857]] /
[495.0, 133.0]

ClusterCenter [center=[398.2142857142857, 118.82142857142857]] /
[397.0, 197.0]

ClusterCenter [center=[398.2142857142857, 118.82142857142857]] /
[457.0, 108.0]

ClusterCenter [center=[398.2142857142857, 118.82142857142857]] /
[488.0, 143.0]

ClusterCenter [center=[398.2142857142857, 118.82142857142857]] /
[485.0, 26.0]
```

Fig 3: Outcome of KMeans execution.

The keys of the input and the output are identifiers of clusters, and the values of the input are the coordinates of spatial points. The values of the output are coordinates of the new medoids.

The three steps can be simplified as follows. The initial medoids are generated. Applying K-Medoids in MapReduce architecture is shown in algorithm1.

IV. RESULTS AND DISCUSSIONS

For implementation of these algorithms the hadoop system is configured on HP Proliant dl580 g5 with 128gb ram and Xeon x7350 CPU 2.93ghz as namenode and 5 units of Intel i3 6006u-4gb ram systems as datanode.

A sample dataset vector is generated with the minimum range as value 10 and maximum range as value 500. The number of input vectors generated is 200 and the number of cluster centers generated is 10.

The figure2 shows the process of input generation. The input and cluster center vector distribution is shown by converting the vector values into Cartesian coordinate values and point distribution chart is drawn as shown in figure2. A process is created which executes the KMeans algorithm on the Hadoop Distributed File System. Generates the output of new cluster centers. The generated new cluster center output is shown in the figure3.

V. CONCLUSION

Here we conclude with an efficient K-Medoids++ algorithm with the initial considerations, algorithm is proposed based on hadoop framework which has been commonly used by academia and industry. There are two visions in improving the K-Medoids algorithm. Firstly, initial medoids searching algorithm is integrated into K-Medoids++ clustering to find the suitable medoids as to decrease a total number of iterations. Next, the K-Medoids++ algorithm is proposed in MapReduce framework. Here, K-Medoids++ clustering algorithm can be used with massive high dimensional data analysis effectively and scales well on commodity hardware.

Further, the Proposed Parallel Algorithm is to be implemented for multi node environment using hadoop map reduce framework to scale the large amount of data, such that data loss doesn't happen and huge big data applications such as bio informatics, academics and industry.

REFERENCES

[1]. G Rasmussen, E.M., Willett, P. "Efficiency of Hierarchical Agglomerative Clustering Using the ICL Distributed Array Processor". Journal of Documentation 45(1), 1–24 ,1989.

[2]. J Li, X., Fang, Z. "Parallel Clustering Algorithms. Parallel Computing, Volume 11, Issue 3, Pages 275-290, 1989.

[3]. Clark F. Olson, Parallel Algorithms for Hierarchical Clustering. Parallel Computing, Volume 21, Issue 8, Pages 1313-132, 1995.

[4]. Dean, J., Ghemawat, S. "MapReduce: Simplified Data Processing on Large Clusters". In: Proc. Of Operating Systems Design and Implementation, San Francisco, CA, pp. 137–150, 2004.

[5]. Dean, J., Ghemawat, S. MapReduce: Simplified Data Processing on Large Clusters. Communications of The ACM 51(1), 107–113, 2008.

[6]. Ranger, C., Raghuraman, R., Penmetsa, A., Bradski, G., Kozyrakis, C. "Evaluating MapReduce for Multi-core and Multiprocessor Systems". International Symposium on High-Performance Computer Architecture (HPCA), Phoenix, 2007.

[7]. Lammel, R. "Google's MapReduce Programming Model" -Revisited. Science of Computer Programming, pp1–30, 2008.

[8]. Hadoop: Open source implementation of MapReduce, http://lucene.apache.org/hadoop/

[9]. Ghemawat, S., Gobioff, H., Leung, S. The Google File System. In: Symposium on Operating Systems Principles, pp. 29–43, 2003.

[10]. MacQueen, J "Some Methods for Classification and Analysis of Multivariate Observations". In: Proc. 5th Berkeley Symp. Math. Statist, Prob., pp.281–297 1967.

[11]. Borthakur, D. The Hadoop Distributed File System: Architecture and Design, 2007.

[12]. Xu, X., Jager, J., Kriegel, H.P. A Fast Parallel Clustering Algorithm for Large Spatial Databases. Data Mining and Knowledge Discovery 3, 263–290 , 1999.

[13]. Han J., Kamber M. and Tung A., in: HJ Miller and J Han., Eds, Spatial clustering methods in data mining: A survey. In Geographic Data Mining and Knowledge Discovery, CRC Press, pp 12-30,2009.

[14]. Boukhdhir, O. Lachiheb and M. S. Gouider, "An improved mapReduce design of kmeans for clustering very large datasets," 2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA), Marrakech, 2015, pp. 1-6. doi: 10.1109/AICCSA.2015.7507226

[15]. Van Hieu D., Meesad P., Fast K-Means Clustering for Very Large Datasets Based on MapReduce Combined with a New Cutting Method. In: Nguyen VH., Le AC., Huynh VN. (eds) Knowledge and Systems Engineering. Advances in Intelligent Systems and Computing, vol 326. Springer, Cham, 2015.

[16]. Li Ma, Lei Gu, Bo Li, Yue Ma and Jin Wang. "An Improved K-means Algorithm based on Mapreduce and Grid." International Journal of Grid Distribution Computing Vol.8, No.1, pp.189-200, 2015.

[17]. L. Kaufman and P. J. Rousseeuw, Finding Groups in Data, An Itroduction to Cluster Analysis, John Wiley & Sons, Brussels, Belgium, 1990.

AUTHORS PROFILE

Maithri .C, is an Associate Professor in the Department of Computer Science and Engineering at Kalpataru Institute of Technology, Tiptur. She is currently pursuing PhD under VTU, Belagavi. She has published several research papers and international conference papers. She is an active member in ISTE.

Dr. Chandramouli.H received his PhD in the year of 2014 and currently working as a Professor in the Department of Computer Science and Engineering at East Point College of Engineering and Technology, Bengaluru. He has 22 years of rich experience in the academics. He has published more than 25 research articles in National and International Journals. He holds CSI membership and an active member in CSI events. His research area includes wireless sensor network, Resource allocation in Networking Big Data Analytics