Generating Better Initial Centroids Over Hadoop for K-Means Clustering

Vineesh Cutting[1] Prateek Singh[2]
[1]M.Tech Scholar [2]Assistant Professor
[1,2]Department of Computer Science & Engineering
[1,2]SHIATS, Allahabad

Abstract— Clustering is one of the traditional data mining technique used for grouping of various kinds of data to perform better analyses. K-Means being most desirable Algorithm for clustering. With the advancement in Technology, the data at many domains is generated at higher rates reaching size greater than Petabyte. Harnessing Hadoop and K-Means resulted in faster processing of large data set. However, random initial centroids have to be provided in traditional K-Means algorithm. The Convergence to be reach highly depends on the set of initial centroids. This paper represents an efficient and simplified technique for generating set of better initial centroids as an input to K-Means Clustering over Hadoop. The experimental result shows better performance in clustering compared to random initial centroids.

Key words: Data Mining, K-Means clustering, Random initial centroids, Better initial centroids, Hadoop, MapReduce

I. INTRODUCTION

The Clustering is the process of organizing objects into different groups called as clusters. Each object belonging to certain group share some common properties in accordance to the other objects present in similar group.

The result of cluster analysis is the group of objects such that objects in group are similar to each other and dissimilar to the objects belonging to other group.

For this, clustering, unsupervised machine learning technique is used. Numerous methods have been proposed to solve clustering problem [1], [2]. Mac Queen in 1967 gave K-Means Clustering technique that became very famous due to its simplicity and came to use in various domains.

K-Means technique takes set of initial centroids to start clustering of n objects into k mutually excessive groups. Distance of each object to its centroid is minimized in each iteration.

However, the technology advancement in computer hardware technology, powerful computers, storage media, and data collection equipments has provided a tremendous growth in the volume of the text documents. With the increase in the number of electronic documents, it is hard to organize, analyse and present these documents efficiently by putting manual effort [3]. These have brought challenges for the effective and efficient organization of text documents automatically [4].

Increase in volume and variety of data gave rise to many problems such as estimating no. of clusters, improving scalability, set of initial centroids [5] due to k-means sensitivity towards initial centroids.

In this paper we propose technique to generate better initial centroids for k-means clustering over Apache™ Hadoop [6] to harness the power of distributed computing with clustering technique. Hadoop [7] is an open-source framework designed by Doug Cutting in 2006 at Yahoo that works on MapReduce concept. MapReduce Parallel Programming model was first introduced by Dean and Ghemawat from Google.

The rest of the paper is organized as follows:

Section II represents fundamental of Hadoop/MapReduce. Section III reviews K-Means Clustering over Hadoop. In Section IV the proposed technique for generation of initial centroids is discussed. In Section V, the result represents performance of the experiments for proposed method. Section VI represents conclusion followed by future scope.

II. FUNDAMENTAL OF HADOOP/MAPREDUCE

Hadoop is a distributed software solution, distributed framework, load of commodity machines, to store and process the data. It is highly scalable, flexible and fault-tolerant capable of processing TBs and PBs of data. Two main components of Hadoop are: File storage and Distributed Processing System.

Hadoop uses “HDFS (Hadoop distributed file system)” that provides low cost storage and high throughput. HDFS uses replication factor to store replicas of file across collection of servers in a cluster. HDFS ensures data availability by keeping the track of data and monitoring the data blocks and also balances the data across data storage nodes.

Hadoop uses MapReduce framework for distributed processing of data. The processing is done on data nodes and instead of collecting all data in one node and processing it, Hadoop sends job to all data nodes resulting in distributed processing. The job for MapReduce is mainly written in java but further development led to the execution of jobs written C, C++, PHP, Python, Pearl, etc.

MapReduce Internals:

A. Input Format

Determines how files are parsed into the MapReduce pipeline [8].

B. Split Phase

Data is divided into input splits based on input format. Input splits equate to map task which runs in parallel.

C. Mappers

Transforms the input splits into key value pairs based on user-defined code.

D. Shuffle and Sort

Shuffle and sorts data by the given key and outputs to reducer.

E. Reducers

Aggregate the results according to user-defined code.

F. Output Format

Determines how data is put back on HDFS.

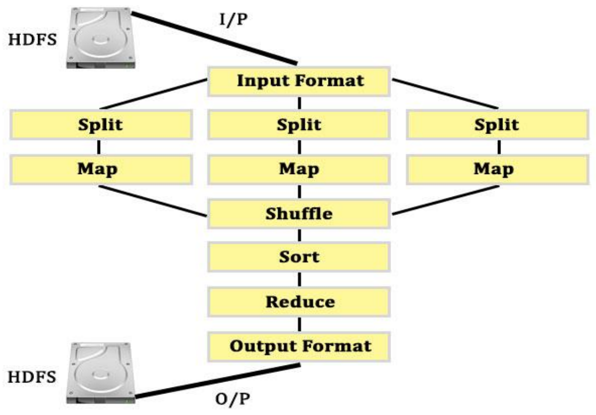

Fig. 1: Map Reduce Internals

III. K-MEANS CLUSTERING OVER HADOOP

The input for K-Means over Hadoop [10] is given as <key,value> pair, where key is the 'centroid' and 'value' is serialized data nodes(objects) that are need to be clustered. These keys and values are maintained in HDFS in separate files. Centroid file contains initial centers either entered by the user or selected randomly from the data nodes(objects) to be clustered. These centers form 'key' for <key,value> pair during Mapper phase.

Once Mapper is invoked, it receives the data in the form of <key,value> pair, it then compares the distance between the all the data nodes(objects) with each of the centers mentioned, prior to Mapper call, in the centroid file provided. If the data note is 1-dimentional then the difference in the value of center and data note is calculated. Once the computation ends, the data note(object) is assigned to the nearest center i.e. center having minimum distance with respect to data note or in case of 1-dimentional data, center having minimum difference. After completion of Mapper phase new <key,value> pairs are made with 'key' as centers and 'value' as large set of data points having minimum distance to that key.

Under Reduce phase, key is taken and mean of all values is calculated that acts as a new centroid for that particular cluster [9]. This process is done for all the keys, resulting in set of new centroids for cluster along with data points assigned to it. Once the centroids for every cluster is updated, the new centers and set of data points are re-written to the disk according to the Output Format making it ready for next iteration.

The process continues until convergence, a stage where the shifting of centroids stops i.e. there is no change in the value of updated centroids.

IV. PROPOSED METHOD

Traditional k-Means algorithm requires initial data points as center that are selected randomly. Since k-Means is highly sensitive to initial centroids so random initial centroids may produce different clustering results, which can lead large no. of iterations to reach Convergence if performed over large data sets.

New proposed method takes up the data points and generates set of initial centroids for K-Means clustering.

1) Step1: Inputting data.
   a) If first iteration, read the data nodes from data file.
   b) Else read data nodes from last iteration.
2) Step2: Mapping Phase.
   a) If first iteration, Map each 'value' i.e. data points to key 1. And follow Step3.
   b) Else take list of centroids from previous iteration as 'key' and compare each data point to each of the centroid.
   c) Centroid having maximum difference in value with data point is selected.
   d) Key, value pair is made with centroid as 'key' and data point with max distance its 'value'.
3) Step3: Emit out key value pairs to reducer.
4) Step4: Reducer Phase.
   a) If first iteration, under reducer, sum up all values for each key and count number of key. Aggregated value is divided then divided by count of key resulting in first initial centroid. The centroid forms the 'key' and data points forms 'value'. And follow Step5.
   b) Else take key and iterate over all its values to select maximum value.
   c) Selected value is then divided by 2.
   d) This divided value acts as next initial centroid.
   e) The updated centroid forms the 'key' and data points forms the 'value'.
5) Step5: Outputting the data.
   Re-write the key and its values to disk in a file.
6) Step6: Repeat as per value of k where k is no. of clusters required.

A. Algorithms for first iteration

Algorithm 1: Implementing KMEANS Function

```
1) Procedure KMEANS FUCNTION1.
2) LOAD cluster file from DIRECTORY.
3) CREATE new JOB
4) SET MAPPER to map class defined.
5) SET REDUCER to reduce class defined.
6) Paths for output directory.
7) SUBMIT JOB
8) END
```

Algorithm 2: Mapper Design for KMEANS Clustering

```
1) Procedure KMEANMAPDESIGN1
2) LOAD cluster file
3) CREATE s_id = 1//acts as a key
4) MAP
   For each element ri , map(ri , s_id)        //mapping each value to s_id
5) COLLECT OUTPUT // <key,value> pairs ready for reducer phase
```

Algorithm 3: Reducer Design for KMEANS Clustering

```
1) Procedure KMEANREDUCEDESIGN1
2) FETCH key
3) ITERATE over all values of that key
4) FOR key = 1
      Calculate $\frac{1}{value\ count}\sum$ values      //mean is calculated
5) COLLECT OUTPUT
6) WRITE calculated mean to output DIRECTORY
```

B. Algorithms for second iteration and further

Algorithm 1: Implementing KMEANS Function

```
1) Procedure KMEANS FUCNTION2.
2) LOAD cluster file from DIRECTORY.
3) LOAD output from first iteration
4) CREATE itr = 0;
5) WHILE (itr < k)// k is the no. of clusters
     a) CREATE new JOB
     b) SET MAPPER to map class defined.
     c) SET REDUCER to reduce class defined.
     d) Paths for output directory.
     e) SUBMIT JOB
     f) INCREMENT itr
6) END
```

Algorithm 2: Mapper Design for KMEANS Clustering

```
1) Procedure KMEANMAPDESIGN2
2) CALL read (cluster file)
3) CALL read (mean from first iteration)
4) CREATE arraylist<mCenters> // forms key in
   <key,value> pair
5) ADD m to mCenters   // array list contains will add
   centroids
6) ASSIGN correct data point
   For each element  , find the farthest data point  to
   cluster
7) FOREACH mCenters
     a) CALCULATE maxDist(ri, mCenters)
     b) Assign farthest ri to mCenters
8) COLLECT OUTPUT               // <key,value> pairs
   ready for reducer phase
```

Algorithm 3: Reducer Design for KMEANS Clustering

```
1) Procedure KMEANREDUCEDESIGN2
2) FETCH key
3) CREATE maxVal = 0
4) ITERATE over all values of specific key
   a) If( value > maxVal )
         maxVal = value
5) CALCULATE maxVal / 2      //next initial centroid
6) COLLECT OUTPUT
     a) WRITE result to output DIRECTORY on HDFS
     b) WRITE previous calculated initial centroid to same
        output DIRECTORY on HDFS
```

V. Results

We have implemented the Normal K-Means Algorithm over Hadoop in java language and above proposed method for generating better initial centroids for k-means clustering over Hadoop. Data Set Used – weather dataset from the National Climatic Data Center. No. of data sets used – 4. No. of Nodes (data points processed) for each dataset – 41,92,914 (approx.). Size of data set 1 – 502.4 MB, data set 2 - 543.6 MB, data set 3 – 503. MB and data set 4 – 532.6 MB.

Since the algorithm is built over Hadoop, it is tested over both Single Node [11] architecture and Multi-Node [12] Architecture. Configuration for Hadoop – Single Node (Intel® Core™ i7-4700MQ CPU @ 2.40GHz, 8GB RAM, Ubuntu 14.10 64-bit) and Multi-Node (1st node - Intel® Core™ i7-4700MQ CPU @ 2.40GHz, 8GB RAM, Ubuntu 14.10 64-bit) (2nd node - Intel® Core™ i5-4200M CPU @ 2.50GHz, 4GB RAM).

Initial centroids were generated from one dimensional data nodes prior to k-means clustering up to value k=3 i.e. 3 initial centroids for 3 clusters. These centroids were then provided as input to K-Means over Hadoop to initiate clustering process.

In Fig. 2, the generated initial centroids are compared to randomly selected initial centroids to achieve convergence for 502.4 MB data set. Fig. 3, represents the similar reduction in iterations to reach the state of convergence for 543.6 MB data set.

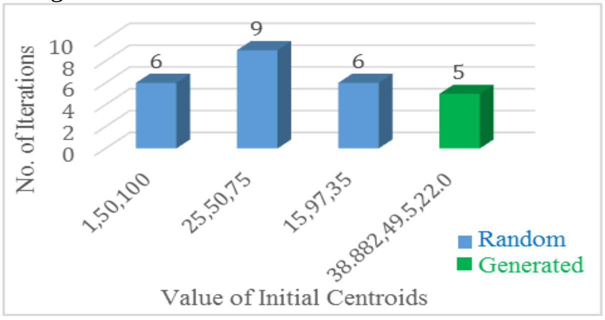

Fig. 2: Comparison of Iterations to achieve convergence for 502.4 MB data set

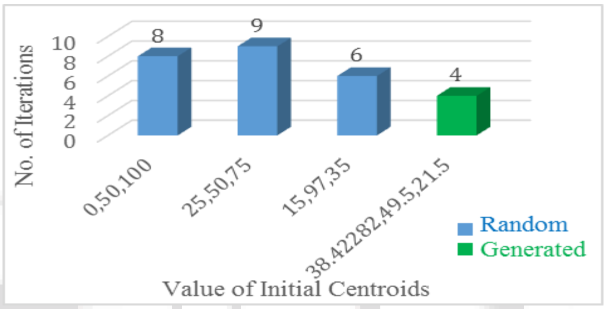

Fig. 3: Comparison of Iterations to achieve convergence for 543.6 MB data set

In figure 4, the generated initial centroids show fair reduction in execution time to reach the state of convergence compared to randomly selected initial centroids for 502.4 MB data set while figure 5 represents more reduction in execution time for 543.6 MB data set to reach convergence.

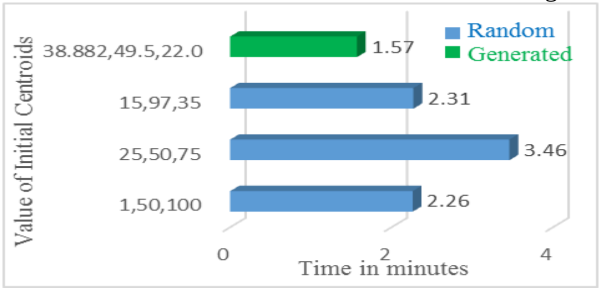

Fig. 4: Comparison of Execution time to Convergence for 502.4 MB data set

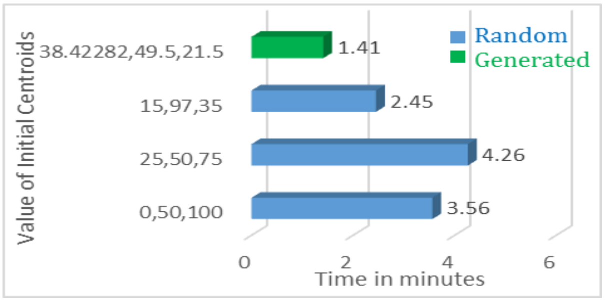

Fig. 5: Comparison of Execution time to Convergence for 543.6 MB data set

VI. Conclusion

This paper presents a new and easy technique to generate initial set of centroids for one dimensional data set. The proposed method fairly reduces the no. of iterations to reach convergence as K-Means is highly sensitive to set of initial centroids. The execution time for K-Means Clustering job to finish has also reduced making the technique useful for large sets of data that would generally require large amount of time to reach convergence as it has been observed in case of randomly selected initial centroids. The result may vary for different data sets.

No matter how good anything can be there is always scope for improvement. Primary area of improvement in any algorithm is its accuracy [9]. Another area of improvement is the implementation of Combiner Class to combine the intermediate output of Mapper for further reduction in execution time. For higher dimensions, modification to the proposed method can be done.

References

[1] Fahim A. M., Salem A. M., F.A. Torkey and M.A. Ramadan, “An Efficient enhanced k-means clustering algorithm,” Journal of Zhejiang University, 10(7): 6261633, 2006.
[2] S. Revathi and Dr. T. Nalini, “Performance Comparison of Various Clustering Algorithm,” IJARCSSE, vol. 3, issue 2, February 2013.
[3] RekhaBaghel and Dr. RenuDhir, “A Frequent Concepts Based Document Clustering Algorithm,” Int’l Journal of Computer Applications, vol. 4, no.5, pp. 0975 – 8887, July 2010.
[4] Huang, “Similarity measures for text document clustering,” Proc. of the 6th New Zealand Computer Science Research Student Conference NZCSRSC, pp. 49-56, 2008.
[5] J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S.H. Bae, J. Qiu, and G. Fox. Twister, “A runtime for iterative mapreduce”, In Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, pages 810-818. ACM, 2010.
[6] Apache Hadoop. http://hadoop.apache.org/
[7] J. Venner, (June 22, 2009), Pro Hadoop. Apress.
[8] Learning BigData with CBT Nuggets at: http://www.cbtnuggets.com/
[9] Prajesh P Anchalia, Anjan K Koundinya and Srinath N K, “MapReduce Design of K-Means Clustering Algorithm”, 978-1-4799-0604-8/13/ IEEE, 2013.
[10] K-Means over Hadoop, (visited on 21st December, 2015) at: http://cmj4.web.rice.edu/MapRedKMeans.html
[11] Description of Single Node Cluster Setup, (visited on 22nd December, 2015) at: http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster/
[12] Description of Multi Node Cluster Setup (visited on 23rd December, 2015) at: http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/.