International Journal of Engineering & Technology, 7 (4) (2018) 5601-5606

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET
doi: 10.14419/ijet.v7i4.21571
Research paper

Evaluating K-means multidimensional big data clusters through MapReduce paradigm

Agnivesh [1] *, Rajiv Pandey [1], Amarjeet Singh [2]

[1] AIIT, Amity University, Lucknow, India
[2] Shriram Institute of Management and Technology, Kashipur, Uttarakhand, India
**Corresponding author E-mail: agniveshpandey13@gmail.com*

Abstract

In the era of big data, with the increasing use of large-scale data-driven applications, clustering and extracting useful information from big datasets has posed challenges. Prevailing clustering algorithms need globally optimized solutions for big datasets. K-means algorithm for clustering is of great interest because of its simplicity. However, there are certain limitations in K-means for analyzing big data which leave scope for successive improvements. This research work presents a new K-means clustering algorithm by improving K-means in MapReduce paradigm. The proposed work presents a method to find initial seeds of clusters instead of randomly selecting them which is a major drawback in standard K-means for clustering big data. The research minimizes MapReduce iteration dependence also. Moreover, the presented algorithm takes into consideration between cluster separation and within cluster compactness to achieve high performance. To obtain efficiency, cloud computing is applied in which Amazon Elastic MapReduce 5.x is used. It distributes the job of clustering among various nodes in parallel using low cost machines. The proposed work is simulated on some real datasets from UC Irvine Machine Learning Repository. The results confirm that the research work models an effective algorithm for clustering Big Data.

Keywords: *Big Data; Cloud Computing; Clustering; Hadoop; K-Means.*

1. Introduction

Big data is a phenomenon which provides technological transformation because of it being dynamic. It originates with the fact that there is a lot more information floating around these days than ever before and it is being put to astounding new uses. The significance of big data does not revolve around how much data we have but what we do with it. We can take data from any source and analyze it to find answers that enable savings of time and cost along with development of new products and smart decision making.

Clustering is a powerful, frequently used big data analytics and prediction technique. The method finds meaningful groups of entities and differentiates clusters formed for a dataset [1]. The capability to automatically group similar items enables analysts to discover hidden similarities and essential concepts while combining a large amount of data into a few groups [2]. This allows users to envision a large amount of data. Standard K-means clustering performs well when applied to small datasets.

K-means algorithm is most widely used clustering technique to find homogenous objects based on distance vectors suited to small datasets. Pre-specifying the number of clusters is the primary requirement to apply K-means. It is a trial-and-error [3] process to find the exact number of clusters for a given dataset. Moreover, initial centers are selected randomly. These steps of conventional K-means algorithm makes cluster analysis of big datasets critical. Hence, it is a need of the time to enhance K-means algorithm to suit large datasets.

Cloud Computing has made remarkable progress in the recent past. With this development, MapReduce paradigm evolved as a competent technique for generating distributed large-scale data processing applications. MapReduce [4] is a coding based model developed by Google for processing multidimensional big datasets. A user can conveniently use MapReduce model for solving area-specific problems. Today, MapReduce is widely applied by several large organizations. Nevertheless, coding map-reduce functions is very difficult to solve real world problems as the framework is rigid.

2. Literature review

Researchers turned to clustering big data as data volume is continuously rising these days and it is need of the time to extract valuable insights from big data. Researchers have been endeavoring to improve further and further big data analytics. Many algorithms have been proposed in this direction. The presented work reviews some of these works which are most widely known.

Researchers in [5] worked on multidimensional large dataset to discuss issues while clustering large datasets with MapReduce. They proposed a parallel clustering method through Hadoop MapReduce framework which focused on a key factor of reducing I/O and network costs. Faster execution and scaling were two crucial points to deal with. The drawback of this method is that it is a hard-clustering process. For the datasets having many overlapping clusters, it may not be the best solution.

Deriving from the behavior of birds in a flock, one of the solutions has been found. In the recent past, the graphics processing unit (GPU) gained attraction. It is able to solve very quickly problems of parallelism. Researchers in [6] applied this concept using CUDA platform from NVIDIA GPU. Weakness of this work lies in its complexity $O(n^2)$. The results generated are not within reasonable period of time. The research work is carried out using

Copyright © 2018 Agnivesh et. al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

single GPU. Parallelism becomes essential to process the dataset that may not be contained in a single disk.
Similarly, Andrade et al. in [7] used DBSCAN algorithm on GPU to gain high performance. However, the method is based on calculating a proximity radius between objects. For those objects which are beyond the proximity radius, this method may not be applicable. In 2013, Xiao Cai and co-authors [8] concisely reflected on collection of data from various sources, each source representing a different aspect of the data. Since each source has its own individual aspect, therefore, clustering of big data here becomes difficult. They presented a novel method to combine composite representation of large scale data which specifies no fix criteria. The method may not be categorized to produce optimized results.
Furthermore, researchers in [9] presented an algorithm to improve K-means clustering by obtaining initial centroids. First, the algorithm calculates mean of the dataset. Then, distance between data points is calculated until d reaches to mean. These two data points will be the candidate for initial centroids. This process is cumbersome for clustering big data and makes it comparatively more iterative than reasonable.
Kodinariyal and Makwana reviewed K-means clustering on determining number of clusters [10].
Authors in Cui et al. [11] mentioned MapReduce to be unstable due to iterations which involve restarting jobs again and again. They proposed a new model of processing big data using K-means algorithm devoid of iterations. The merging technique proposed by them does not seem to be appropriate and therefore clustering quality may not be as required in big data clustering.
Researchers in [12] worked on clustering text documents for similarity check. Deliberating that output is fully dependent on input of number of clusters, they affirmed keyword as input. They created subset of documents for achieving desired results. Divide and Conquer strategy applied in this research is in itself a major drawback.
Tsai with co-researchers in [13] write that one of the main reasons of failure in clustering large datasets by traditional clustering methods is that most of them are designed for centralized systems. In their paper they proposed to solve this problem by an algorithm which they termed as MapReduce Black Hole (MRBH) which accelerated clustering. The weakness is that they applied random generation and initialization of stars.
Problem of initial cluster centers in K-means algorithm was tried to be solved by researchers in Wu et al. [14] by sampling the large dataset and used convex hull and opposite Chung points. They applied MapReduce framework for parallel execution of the algorithm. The limitation of this work lies in the fact that the Chung points find only two initial cluster centers.
Authors of [15] compared K-means and K-mediods. They applied nearly ten thousand transactions of KEEL dataset repository. Results described that K-mediods is better than K-means in terms of execution time, noise minimization and choosing initial center.
Further, researchers studied K-means, Fuzzy C-Means (FCM), hierarchical clustering algorithm like Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) and grid-based clustering algorithm Clustering In QUEst (CLIQUE) which are the prevailing effective algorithms. Ajin and Lekshmy [16] compared them in context of big data. A new Inter and Intra K-means Clustering (KM-I2C) algorithm was developed in [17] by changing clustering distance metric that used parallelization tools through Hadoop. However, the method selects initial centers randomly from the dataset which is a major limitation in clustering big data as it results in generating less optimized clusters.
Rehioui et al. [18] in their research presented a new version of DENCLUE called DENCLUE-IM which avoids complexities of other DENCLUE algorithms for speedy calculation of big data. They compared the proposed approach with DENCLUE, DENCLUE-SA and DENCLUE-GA. However, for execution in multiprocessor environment, partitioning of dataset may find a difficult proposition. Moreover, fixed number of clusters can make it difficult to predict the value of K.
A concept of big data in form of streams and how to process them was presented by Giacomo Aletti and the co-author in their research in [19]. Datasets having components of characteristics which find correlation were considered. They used Mahalanobis distances for assignment of data to clusters estimating total number of clusters. The proposed method can be applied only in the cases where data points are low dimensional.
Researchers in [20] worked on clusters where boundaries are not firm with certainty. There are lower and upper approximations. The algorithm proposed by them was based on weighted distance measure with Gaussian function for computing the new center for each cluster. The main drawback of the presented algorithm is that it is highly complex in nature consuming more computing time than the other algorithms.
Vijay et al. [21] presented an algorithm which they termed as Variance Based Moving K-means (VBKM). They applied a new distance metric. Moreover, the research work used a different approach for moving data points between clusters to minimize within cluster distance. The researchers have not taken into account selection of initial centers of clusters and as such the results may remain lacking in achieving optimized output.

3. Background

3.1. Exploring traditional k-means algorithm

K-means segregates the data set into K parts where K denotes a positive integer number and stands as a user input to the algorithm. Each cluster has a centroid. The algorithm checks the positions of these centroids as the algorithm iterates. Random values are put to initialize the centroids before the first iteration. The algorithm stops as soon as centroids locations become static during iteration [22]. This well-known algorithm performs two steps per iteration:-

1) Allocate each object x_i to a closest cluster centroid c_j. This allocation is obtained by Euclidean measure between the object and the cluster's centroid (obtained at a previous iteration).
2) Update the centroids of the clusters based on new clusters members.

The above two steps continue to a finite number of repetitions, till occurrence of any alterations in the centers.
Suppose $\{x_1,....,x_n\}$ are the given observations of a multidimensional huge dataset D with d dimensions. The objective is to determine a set of K or cluster centroids $C = \{c_1,\ldots,c_K\}$ that minimizes the within cluster distortion given in equation 1:

$$W(C) = \sum_{i=1}^{n} \|x_i - c\|^2 \quad (1)$$

The above equation defines a cluster assignment rule as

$$f(x) = \arg \min_{f\epsilon\{1,..,K\}} \|x - a_f\|^2 \quad (2)$$

Where a_f is one of the initial centroid.
The problem of choosing K, the number of clusters and cluster centers, can be considered to be a model determination problem.

Algorithm 1: Standard K-means
Input: Dataset D={x_1, xn} and
K (number of clusters to be generated)
Output: K clusters
Select centers C randomly from D
If u > iter then /* loop repeats until convergence
For each x_i-compute distance from all centers C_j
Dist (xi, Cj)
Assign xi to closest Cj
Min (dist (xi, Cj)
Calculate new centroid

$$n_i = \frac{1}{c_i} \sum_{i=1}^{c_i} x_i$$

End for
End if
Exit

3.2. Big data analytics technology

Conventional data warehouse fails to handle unstructured and semi-structured data since they are based on relational data model. Moreover, data warehouses are incapable of processing sets of big data that are regularly updated or may be continually, as in the case of real-time data such as weather data. That is why many organizations that collect process and analyze big data turn to NoSQL databases as well as Hadoop.
Hadoop, an open source framework addresses two main issues of big data which includes storage and processing using distributed concepts with commodity hardware. This framework literally carries large amount of data and when demand arises, performs data analysis.
Hadoop contains two core components. These are Hadoop Distributed File System (HDFS) and MapReduce. HDFS is specially designed file system for storing huge datasets with clusters of commodity hardware and streaming access patterns. MapReduce is the processing component of Apache Hadoop. It processes data in distributed environment. MapReduce can be programmed in various languages such as Java, Python and Ruby.
Traditional K-means algorithm works in MapReduce paradigm in the following way:
The algorithm inputs two vectors. First vector represents our data and the second vector contains K-centers which are sometimes just a subset of the input vectors, but sometimes they are random points or points-of-interest to which we are going to cluster them. In a MapReduce version we use our cluster center-vectors always like keys, and the input vectors are simple values.
Map steps are given below:

- Read the cluster centers into memory from a sequence file
- Iterate over each cluster center for each input key/value pair.
- Measure the distances and save the nearest center which has the lowest distance to the vector.
- Write the cluster center with its vector to the file system.

Reduce steps (we get associated vectors for each center) are given below:

- Iterate over each value vector and calculate the average vector. (Sum each vector and divide each part by the number of vectors we received).This is the new center, save it into a Sequence File.
- Check the convergence between the cluster center that is stored in the key object and the new center.
- If they are not equal, increment an update counter.
- Run this whole thing until nothing was updated anymore.

Algorithm 2: K-means$_{MapReduce}$
Input: Dataset D ={$x_1,\ldots,x_n$} and
K (number of clusters to be generated)
Output: Final set of clusters
i = 0
for all d Є D do
initial_centroid = select(K,d)
input data file from directory
write initial centroid to a file
previous_centroid = initial_centroid while condition true doset mapper to map class define
set reducer to reduce class define
end while
new_ cluster_ centroid = read centroid values to a file
if update ((new_cluster, old_cluster) > 0)
previous_centroid = new_centroid
else
update new_centroid to result
i++
repeat until convergence
result = read centroid values to a file

4. Proposed work

To analyze big data, finding the optimized initial centers is essential. To achieve this, the algorithm takes an extensive value of K, selects randomly one of the data point as initial centroid and then applying likelihood distribution, the remaining K-1 points are sampled by using the equation:

$$L(x) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{K}(\text{distance}(x_i,C_j)}{\sum_{i=1}^{n}\sum_{i=1}^{K}\max(\text{distance}(x_i,C_j)} \quad (3)$$

Various algorithms available today to cluster datasets are lacking in estimating cluster similarity for creating well defined clusters. A K-means algorithm will give globally optimized results with high levels of within cluster similarity and low levels of between cluster similarities. The proposed work splits big data into definite clusters based on between cluster and within cluster similarities. Final clusters generated after execution of the experiment will be well separated if the betweencluster metric value will be maximized and each cluster produced will be tight if the withincluster distance measure will be minimized.
The proposed work modifies map phase and reduce phase of standard K-means MapReduce algorithm. It consists of two algorithms, illustrated as Algorithm 3 and Algorithm 4. There are two new distance measures- between$_{cluster}$ and within$_{cluster}$ used in Map phase. The between$_{cluster}$ metric calculates the average separation of clusters which is given by equation 4:

$$\text{between}_{\text{cluster}} = \frac{1}{n_c}\sum_{i=1}^{n_c}\frac{\|\sigma(v_i)\|}{\|\sigma(x)\|} \quad (4)$$

where n_c is the number of clusters, v_i is centroid of the i^{th} cluster and σ is variance.
The within$_{cluster}$ metric calculates similarity measure between the data in a cluster. It is defined by finding density which is given by equation 5 as follows:

$$\text{within}_{\text{cluster}} = \frac{1}{n_c(n_c-1)}\sum_{i=1}^{n_c}\sum_{\substack{i=1\\ i\neq j}}\{\frac{\text{density}(m_{ij})}{\text{density}(v_i),\text{density}(v_j)\}} \quad (5)$$

Here m_{ij} = midpoint of the distance between v_i and v_j cluster centroids.
As given in [23] the density function is defined by the number of points in a hyper-sphere whose radius is equal to the average standard deviation of clusters. Precisely the average standard deviation of clusters derived is as referred below:

$$\text{stndev} = \frac{1}{n_c}\sum_{i=1}^{n_c}\|\sigma(v_i)\| \quad (6)$$

Algorithm 3: Proposed Work Map Phase
Input: D: dataset having n number of dimensions in each mapper
K: K clusters
$C_{1,\ldots,K}$: Initial cluster centers obtained from equation (3)
Output: output vector<key, value>
vector_new: New Centroid Vector
value = 0
vector_new = 0
while all d Є D
while all C_i Є C do
cc =: Ø /* cc denotes center nearest to the data point
between_cluster = MAX_VALUE
within_cluster = MAX_VALUE
while all d_i Є D do
l(d_i) =: euc (d_i, d_j) /* euc is Euclidean distance between data points
i =: 0 /* i represents number of iterations
c =: 0

while all x_i Є X do
minDist =: DistanceMeasurer(d_i, c_j)
if (curr_centr = 0 or 1(d_i) < minDist) then
update intra_cluster using equation (4)
else
update inter_cluster using equation (5)
cc++
i++
generate output vector <key, value> with each data point and the centroid of its cluster
repeat until convergence

Algorithm 4: Proposed Work Reduce Phase
Input: (key, value): where key is centroid, and value is data point assigned to the centroid
O_v: Output vector from mappers
Output: vector_new : Averaging_Vector (AV)
vector_new = 0
AV = null
while each x Є O_v do
centroid = x.key
data point = x.value
AV = data point
while each c_i Є M do
Sum_vector = null
Num_vector = null
while each o_i Є O do
Sum_vectors += vector
Num_vector ++
AV = sum_vector/num_vector
Outputvector = AV
Return AV

5. Experimental design

5.1. Machines

The research work is simulated on Amazon Elastic MapReduce 5.x using Amazon Web Service (AWS) Elastic Compute Cloud (EC2) resources having 8 (2.5 GHz Intel Xeon E5-2670v2) CPUs. Apache Hadoop 2.7.3 version is used on Ubuntu Server, 64 bit operating system with high I/O performance. Hadoop streaming utility is applied to write mapper and reducer executable in Java. Dataset is stored in Amazon S3. Table 1 gives information about the machines used in the experiment. Instance is a virtual server in Amazon EC2 executing applications on AWS framework. Two types of nodes are used in EMR cluster of EC2 instances: master node and core node. Master node controls the entire cluster and usually runs master components of distributed applications. Core node handles data storage and runs parallel computation tasks on data which is needed by the installed application.

Table 1: Hardware Configuration

Node	Instance Type	Memory	Storage	No. of Instance
Master: Master Instance Group	m3. xlarge	15GB	80 GB	1
Core: Core Instance Group (1-2)	m3. xlarge	15 GB	80 GB	2
Core: Core Instance Group (3-4)	m3. xlarge	15 GB	80 GB	2

5.2. Dataset statistics

Datasets used in the experiment are real datasets imported from UCI Machine Learning Repository. The datasets are YouTube Multiview Video Games Dataset and Daily and Sports Activities Datasets. After data cleaning, these raw datasets are transformed into understandable format. YouTube Multiview Video Games dataset is segregated into Dataset1, Dataset2 and Dataset3. Daily and Sports Activities dataset is partitioned into DSA1, DSA2, DSA3 and DSA4 datasets. The datasets vary in size extending from 128 MB to 1 GB nearly. The datasets are sparse. The details about the datasets are given in Table 2. We applied subset of these datasets.

YouTube Multiview Video Games Dataset has nearly 120 thousands records with 1 million attributes. Characteristics of this dataset are multivariate. Attributes contain integer and real values.

Daily and Sports Activities Dataset is collected by providing sensors to monitor motion of sports activities at the rate 19 per day involving 8 persons in their own individual styles for 5 minutes. There are 9120 records in the dataset with 5625 number of attributes. The dataset has the typicality that it is time-series multivariate dataset. Data values are real as recorded by the collectors. Fig. 1 presents a subset of the datasets which is input to the proposed system.

Table 2: Statistics of 7 Large-Scale Real Datasets

Dataset	No. of Samples	No. of Dimensions	Size (MB)
Dataset1	97,935	838	706
Dataset2	97,935	838	706
Dataset3	97,935	838	706
DSA1	2,85,000	45	128
DSA2	5,70,000	45	257
DSA3	11,40,000	45	467
DSA4	22,80,000	45	934

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11
1	8.1305	1.03490	5.4217	-0.009461	0.001915	-0.003424	-0.78712	-0.069654	0.15730	0.70097	5.6829
2	8.1305	1.02020	5.3843	-0.009368	0.023485	0.001953	-0.78717	-0.068275	0.15890	0.71829	5.6005
3	8.1604	1.02010	5.3622	0.015046	0.014330	0.000204	-0.78664	-0.068277	0.15879	0.69849	5.6612
4	8.1603	1.00520	5.3770	0.006892	0.018045	0.005649	-0.78529	-0.069849	0.15912	0.72799	5.6393
5	8.1605	1.02750	5.3473	0.008811	0.030433	-0.005346	-0.78742	-0.068796	0.15916	0.71572	5.6441
6	8.1454	1.02010	5.3919	0.007001	0.029523	-0.008080	-0.78716	-0.070603	0.15905	0.71323	5.6611
7	8.1454	1.04950	5.4518	-0.005710	0.014217	-0.021708	-0.78720	-0.067744	0.15836	0.71813	5.6466
8	8.1376	0.99744	5.5111	0.010574	0.030563	0.002868	-0.78690	-0.069828	0.15892	0.72801	5.6392
9	8.1307	1.06460	5.3921	-0.009296	0.030531	-0.008107	-0.78717	-0.069696	0.15891	0.71331	5.6296
10	8.1155	1.02020	5.4215	0.005122	0.027022	0.005606	-0.78643	-0.069156	0.15773	0.71320	5.6708
11	8.1605	1.04960	5.3773	-0.012963	0.017953	-0.012616	-0.78646	-0.070042	0.15707	0.72073	5.6440
12	8.1306	1.02040	5.3396	0.001632	0.043022	-0.005414	-0.78596	-0.068679	0.15908	0.71806	5.6515
13	8.1157	1.02780	5.3693	-0.002114	0.023404	0.000138	-0.78657	-0.070697	0.15747	0.75738	5.6247
14	8.1306	1.02770	5.3694	0.005111	0.016129	-0.004380	-0.78706	-0.069229	0.15583	0.71311	5.6733

Fig. 1: Input Dataset to the Proposed System.

6. Results and discussion

6.1. Performance

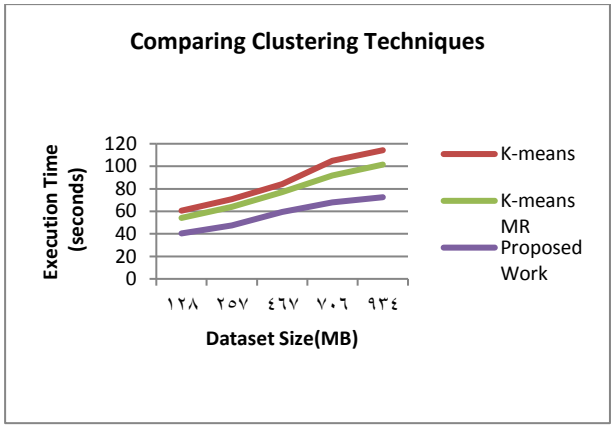


Fig. 2: Comparing Execution Times with Dataset Size.

Fig. 2 illustrates the execution times of 15 clusters for the datasets compared with proposed work, K-means MapReduce and parallel K-means algorithms. K-means and K-means MapReduce use Euclidean metric to check similarity. Our proposed algorithm uses between_cluster and within_cluster distance measures to generate better quality big data clusters with higher within cluster similarity and lower between cluster similarities.

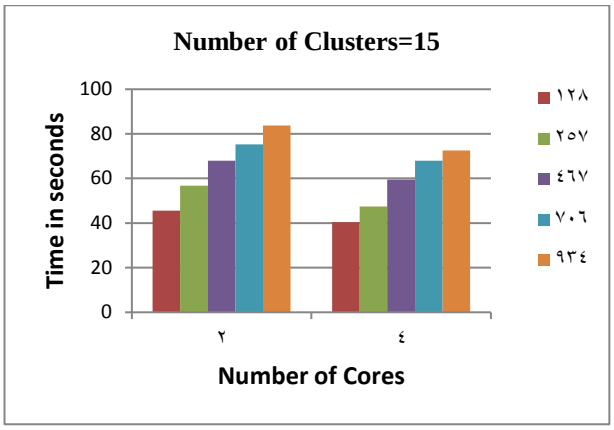


Fig. 3: Comparing Execution Time with 15 Clusters by Proposed Work.

The outcome of the number of core machines on the clustering algorithms is also studied in the research work. Fig. 3 and Fig. 4 show the execution times of 15 clusters for cluster analysis by proposed work and K-means MapReduce when operated on 2 and 4 numbers of cores.

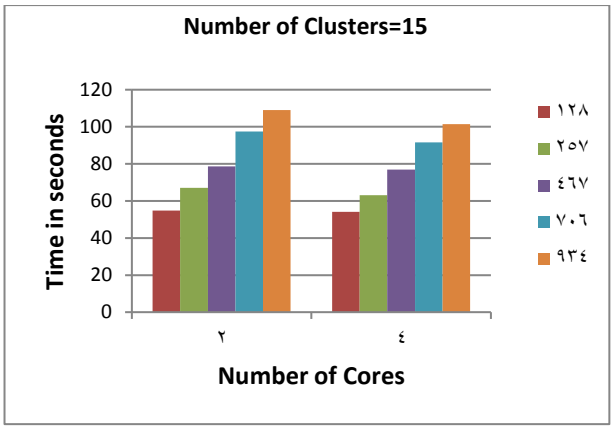


Fig. 4: Comparing Execution Time with 15 Clusters By K-Means Mapreduce.

These results exhibit that the level of scalability has appreciable impact on the process of clustering. With increase in the number of parallel processing, there is significant reduction in execution time by the algorithms.

Table 3: Execution Time in Seconds

Dataset	Standard K-Means	K-means MapReduce	Proposed Work
128	60.57	54.13	40.37
257	70.88	63.04	47.42
467	84.00	76.92	59.52
706	104.97	91.57	67.94
934	114.25	101.49	72.51

Table 3 presents execution times of the three comparing algorithms for K= 15 clusters with dataset size ranging from 128 MB to 934 MB.

Figure 1 represents dataset sizes in mega bytes along x-axis and execution time of the algorithms in seconds along y-axis. The red curve in the graph represents execution time of standard K-means which varies between 60 seconds to 114 seconds. The green curve in the graph represents execution time of K-means MapReduce algorithm which varies between 54 seconds to 101 seconds. The blue curve in the graph represents execution time of our proposed work which varies between 40 seconds to 72 seconds.

Evidently proposed algorithm reduces remarkably the time of execution compared to other algorithms. Here, smallest size dataset is 128 MB and the largest size dataset is 934 MB. The execution time is reduced by application of the proposed algorithm by 19% and 13% respectively compared to standard K-means and K-means MR in execution of smallest dataset. Whereas the reductions in execution times are 42% and 29% respectively compared to standard K-means and K-means MR in execution of largest dataset.

Proposed algorithm outperforms the other algorithms. It is prominently more advantageous to apply for the larger datasets.

6.2. Cluster validity

Cluster validity is a term widely referred when assessment of the results of a clustering algorithm is performed. For measuring "goodness" of a clustering result, there are several validity indices which are applied. The Davies-Bouldin index is one of the popular validity indices. The index is determined by equation 7 as:

$$DB_i = \frac{1}{K} \sum_{i=1}^{K} S_{ij} \quad (7)$$

Here

$$S_{ij} = \max_{i,j=1,..,K} \frac{s_i + s_j}{d_{ij}}, i \neq j$$

Here K = number of cluster, S_{ij} = similarity measure of clusters, d_{ij} = distance between two centroids, and s_i = dispersion measure of a cluster.

Lower DB_i represents high separation between clusters and high similarity within clusters.

Table 4: DB_i for Distinct K

K	DBi of Proposed Work	DBi of Standard K-means	DBi of K-means MapReduce
10	0.237538	0.321101	0.241037
11	0.209694	0.358620	0.265912
12	0.235571	0.347195	0.254349
13	0.217063	0.362689	0.276085
14	0.228975	0.350901	0.276954
15	0.210966	0.362449	0.288975

Table 4 shows values of DB_i for proposed work, standard K-means and K-means MapReduce algorithms for varied number of clusters ranging from K=10 to K=15 for various dataset sizes. It is observed that the work under reference has lower values of DB_i compared to the other two algorithms exhibiting distinctly good quality clusters generation. Fig. 5 is showing outputs produced after execution of the proposed algorithm.

```
Cluster means:
        V1          V2          V3             V4             V5
1  8.116742 1.032412 5.453933  0.0074504167  0.0324826667
2  7.936725 1.163225 5.641625 -0.0054682500 -0.0131535000
3  8.129767 1.023578 5.367744  0.0059478889  0.0257396667
4  7.862033 1.087883 5.751172  0.0068742222  0.0168597222
5  8.112580 1.026134 5.424500  0.0057418000  0.0349638000
6  7.968350 1.122425 5.600450 -0.0036170000 -0.0003027500
7  7.827500 1.109933 5.859722  0.0007972778  0.0004756111
8  7.985260 1.122180 5.528320  0.0006954000  0.0138013000
9  7.960825 1.127242 5.635375  0.0059259167  0.0231860833
10 8.137382 1.038327 5.388564  0.0042058182  0.0323459091
11 8.012500 1.047220 5.541740  0.0029870000  0.0612746000
12 7.930808 1.084958 5.650025  0.0090770000  0.0594514167
13 7.741800 1.175800 6.143850  0.0288580000  0.0698370000
14 8.130600 1.020350 5.358250 -0.0011415000  0.0322905000
15 7.930000 1.140500 5.316900  0.0351320000  0.2147600000
         V8          V9          V10         V11         V12
1  -0.06950825 0.1557700 0.7109808 5.651833 7.963492  0.01
2  -0.07001250 0.1321100 0.6259675 5.804550 7.866725  0.01
3  -0.06919378 0.1567456 0.7206122 5.635811 7.964100  0.01
4  -0.06656267 0.1196533 0.6875500 5.713939 7.916000  0.01
5  -0.06953920 0.1530780 0.6837940 5.692600 7.976260  0.01
6  -0.06920950 0.1346250 0.6671550 5.779025 7.885625  0.01
```

Fig. 5: Outputs of the Proposed Work.

7. Conclusion

Clustering Big Dataset is a very difficult issue which depends on the shape of the dataset and the problem under consideration. Currently, Hadoop MapReduce is most sought-after technique to cluster big data. In this research work, we proposed improvements in MapReduce K-means algorithm to overcome its limitations in clustering big data. The presented work includes two algorithms. First algorithm modifies Map phase. It inputs a large value of K, computes location of initial seeds and applies between$_{cluster}$ and within$_{cluster}$ distance measures. Second algorithm reformed Reduce

phase by aggregating these centroids and generates final set of quality clusters. We verified the performance of our work on AWS public datasets. The results confirm that the proposed work models a powerful tool for clustering big datasets. Experimental results also prove that the proposed work excels K-means parallel and K-means MapReduce algorithms.

References

[1] Han J, Pei J & Kamber, M. Data Mining: Concepts and Techniques, Third Edition, Elsevier, 2011.

[2] Fang W, Sheng VS, Wen X & Pan W. "Meteorological data analysis using MapReduce". Sci World J. 2014;2014. https://doi.org/10.1155/2014/646497.

[3] Pham DT, Dimov SS & Nguyen CD. (2005), "Selection of K in K-means clustering", Proc. IMechE Vol. 219 Part C: J. Mechanical Engineering Science, https://doi.org/10.1243/095440605X8298.

[4] Du W, Qian D, Xie M & Chen W. "Research and Implementation of MapReduce Programming Oriented Graphical Modeling System", *2013 IEEE 16th International Conference on Computational Science and Engineering*, Sydney, NSW, 2013, pp. 1332-1337. https://doi.org/10.1109/CSE.2013.197.

[5] Cordeiro RLF, Traina Junior C, Traina AJM, López J, Kang U & Faloutsos C. "Clustering very large multidimensional datasets with MapReduce", In: Proceedings of KDD'11, ACM, California, August 21–24. 2011.

[6] Cui X, Charles JS & Potok T. "GPU enhanced parallel computing for large scale data clustering". Future Generation Computer Systems, 29(7), 1736-1741, (2013). https://doi.org/10.1016/j.future.2012.07.009.

[7] Andrade G, Ramos G, Madeira D, Sachetto R., Ferreira R & Rocha L. (2013). "G-DBSCAN: A GPU accelerated algorithm for density-based clustering". Procedia Computer Science, 18, 369-378 https://doi.org/10.1016/j.procs.2013.05.200.

[8] Cai X, Nie F & Huang, H. "Multiview K-means Clustering on Big Data". Proceedings of the Twenty-Third International Conference on Artificial Intelligence, Pages 2598-2604, Beijing, China, August 03-09, 2013, ISBN:978-1-57735-6332-2

[9] Ghosia U, Ahmad U & Ahmad M. "Improved K-Means Clustering Algorithm by Getting Initial Centroids", World Applied Sciences Journal 27 (4): 543-551, 2013, ISSN 1818-4952, © IDOSI Publications, 2013, DOI: 10.5829/idosi.wasj.2013.27.04.1142.

[10] Kodinariya1 TM & Makwana PR. "Review on determining number of Cluster in K-Means Clustering", International Journal of Advance Research in Computer Science and Management Studies, ISSN: 2321-7782 (Online), Volume 1, Issue 6, November 2013.

[11] Cui X, Zhu P, Yang X, Li K & Ji C. "Optimized big data clustering using MapReduce". The journal of Supercomputing (2014). Volume 70, Issue 3, pp 1249–125970: 1249. https://doi.org/10.1007/s11227-014-1225-7 .

[12] Bide P & Shedge R. "Improved Document Clustering using k-means algorithm", *2015 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, Coimbatore, 2015, pp. 1-5. https://doi.org/10.1109/ICECCT.2015.7226065.

[13] Tsai CW, Hsieh CH & Chiang MC. "Parallel black hole clustering based on MapReduce", In: Proceedings of IEEE International Conference on Systems, Man and Cybernetics, 2015. https://doi.org/10.1109/SMC.2015.445.

[14] Wu K, Zeng W, Wu T & An Y. "Research and improve on K-means based on hadoop". Software Engineering and Service Science (ICSESS). 2015 6th IEEE conference, 23-15 september, 2015. https://doi.org/10.1109/ICSESS.2015.7339068.

[15] Arora P, Deepali & Varshney S. "Analysis of K-Means and K-Medoids Algorithm For Big Data", Volume 78, 2016, Pages 507-512. https://doi.org/10.1016/j.procs.2016.02.095.

[16] Ajin VM & Kumar LD. "Big data and clustering algorithms". 2016 International Conference on Research Advances in Integrated Navigation Systems (RAINS). https://doi.org/10.1109/RAINS.2016.7764405.

[17] Shridhar, C., Kasivishwanath, N. and Reddy, P. C. Clustering large datasets using K-means modified inter and intra clustering (KM-I2C) in Hadoop. Journal of Big Data (2017). Springer. https://doi.org/10.1186/s40537-017-0087-2.

[18] Rehioui, H., Idrissi, A. Abourezq, M. and Zegrari, F. DENCLUE-IM: A New Approach for Big Data Clustering. Procedia Computer Science, Volume 83, 2016, pages 560-567, ELSEVIER. https://doi.org/10.1016/j.procs.2016.04.265.

[19] Aletti, G. and Micheletti, A. A clustering algorithm for multivariate data streams with correlated components. Journal of Big Data 2017:48. https://doi.org/10.1186/s40537-017-0109-0.

[20] Zhang, T. and Ma Fumin, F. (2017). "Improved rough *k*-means clustering algorithm based on weighted distance measure with Gaussian function", International Journal of Computer Mathematics, 94:4, 663-675, https://doi.org/10.1080/00207160.2015.1124099.

[21] Vijay V, Raghunath VP, Singh A & Omkar SN. "Variance Based Moving K-Means Algorithm," *2017 IEEE 7th International Advance Computing Conference (IACC)*, Hyderabad, 2017, pp. 841-847. https://doi.org/10.1109/IACC.2017.0173.

[22] Jain AK & Dubes RC. (1988). Algorithm for Clustering Data, Prentice Hall.

[23] Halkidi M, Batistakis Y & Vazirgiannis M. Journal of Intelligent Information Systems (2001) 17: 107. https://doi.org/10.1023/A:1012801612483.

[24] Saini A, Minocha J, Ubriani J & Sharma D. "New approach for clustering of big data: DisK-means," *2016 International Conference on Computing, Communication and Automation (ICCCA)*, Noida, 2016, pp. 122-126. https://doi.org/10.1109/CCAA.2016.7813702.

[25] Pandove D & Goel S. "A comprehensive study on clustering approaches for big data mining," *2015 2nd International Conference on Electronics and Communication Systems (ICECS)*, Coimbatore, 2015, pp.1333-1338. https://doi.org/10.1109/ECS.2015.7124801.

[26] Qiao J & Zhang Y. "Study on K-means method based on Data-Mining," *2015 Chinese Automation Congress (CAC)*, Wuhan, 2015, pp. 51-54. https://doi.org/10.1109/CAC.2015.7382468.

[27] UCI Machine Learning Repository: https://archive.ics.uci.edu (2015).