JOURNAL OF AERONAUTICAL MATERIALS
ISSN: 1005-5053

Vol. 43, Issue-02, 2023
pp. 133-145

EMPOWERING LARGE-SCALE CLUSTER ANALYSIS: INTRODUCING THE OPTIMIZED REPARTITIONED K-MEANS ALGORITHM ON HADOOP

Dr. S.Shiva Prasad[1*] and Dr. Ramu Vankudoth[2]

[1]Professor & HOD, Department of CSE – Data Science, Malla Reddy Engineering College(A), Secunderabad, Telanagna, India

[2]Assistant Professor, Department of CSE – Data Science, Malla Reddy Engineering College(A), Secunderabad, Telanagna, India

ABSTRACT:

The Internet of Things (IoT) has become a significant source of data in today's world. These vast amounts of data can be utilized in various applications within smart cities, improving the quality of human life through convenience and comfort. The demand for effective data mining techniques to extract valuable insights from this abundant resource has grown substantially. To process such data efficiently, suitable computing techniques, such as distributed computing, are essential. Distributed computing is a computer approach that allows for high computational processing over multiple interconnected systems. Each network-connected system is referred to as a node, and the collective set of nodes is referred to as a cluster. Clustering, a crucial data analysis method, involves partitioning the primary dataset into multiple subsets based on similarities among data points. In this study, we applied optimized K-Means clustering techniques to handle data-intensive distributed applications and compared the results with different available tools. Among these tools is Hadoop, which originated MapReduce and Google File System (GFS) from Google. Hadoop provides a reliable and scalable infrastructure for efficiently managing extensive data processing tasks.

Keywords: Internet of Things (IoT), data mining techniques, GFS, Optimized K Means, Hadoop

1. Introduction

In recent years, the rapid growth of mobile Internet, cloud computing, IOT, social network services, and other developing technologies has resulted in an explosion of data. The focus has shifted towards achieving fast and effective analysis of knowledge and maximizing the benefits of data property[2]. Traditional methods of data analysis have become inadequate due to the "four Vs" model of big data, encompassing variety, volume, velocity, and value. As a result, new techniques including distributed analysis, features extraction, and sample have received a lot of attention[3,4].

The Hadoop framework provides applications with transparent reliability and data movement capabilities. It uses the MapReduce processing model, which divides the programme into smaller work fragments that can be run on any nodes in the clusters. Hadoop also contains a distributed filing system for storing data on processing nodes, resulting in a high aggregate bandwidth throughout the cluster. MapReduce and the Decentralized Document Storage Solution are both designed to manage node failures automatically [5-7].

Cloud computing has emerged as a promising field of research in recent years, providing a path for affluent societies. Data maintenance in cloud computing presents challenges that can be

mitigated by increasing data accuracy and reliability. Various methods, including the use of sensors in machines and systems like nuclear power plants, gas turbines, and wind turbine farms, continuously gather data for system monitoring, fault prediction, and monitoring in cloud computing. One proposed approach for fault prediction is the Case-Based Reasoning (CBR) approach, which utilizes past cases to find solutions for new problems. The information of past cases is stored in the Case Base.

2. Literature Survey

There are two primary categories of systems capable of performing large-scale data analytics on a cluster: Parallel Database Systems and MapReduce-based systems. Research on Parallel Database Systems dates back to the late 1980s [8]. The performance and scalability differences between these two systems are explained in [9] and [10], while [11] delves further into the scalability issues.

MapReduce is a popular parallel processing paradigm used for handling large data sets on a cluster. In [12], it is described as a simple and powerful interface. Additionally, it achieves Exceptional performance achieved on extensive networks of affordable personal computers [1]. It provides an overview of CBR principles, methods, and systems, focusing on task-method decomposition and the representation of cases that heavily influence CBR.

In [12] employ the MapReduce model for creating certain components of the framework. Authors proposed, the Map Join Reduce model [5] to develop the ECV framework component. This approach will enhance the efficiency and scalability of data handling within the proposed framework, particularly for heterogeneous data sets.

Arthur and Vassilvitskii[15] conducted a preliminary implementation of the k-means clustering algorithm, known as k-means++. K-means++ utilizes a careful seeding method to optimize both speed and accuracy. By conducting experiments on four datasets, researchers observed that this algorithm exhibits $O(\log k)$-competitiveness when compared to the optimal clustering solution. Additionally, the experimental results revealed that k-means++ surpasses traditional k-means in both speed (70% faster) and accuracy, achieving significantly improved results ranging from 10 to 1000 times better. Kanungo et al [16] also proposed an $O(n\text{^}3/\epsilon\text{^}{-d})$ algorithm for the k-means problem, which is $(9 + \epsilon)$-competitive but relatively slow in practice. Another study by Xiong et al. focused on identifying performance measures for K-means clustering. The impact of data distributions on K-means clustering performance was investigated, leading to improvements in handling datasets with varying cluster sizes. The study revealed that entropy sometimes provided misleading information, prompting the proposal of a coefficient of variation (CV) as a validation metric for clustering outcomes.

3. Cluster Analysis

Data mining is an interdisciplinary field that encompasses various definitions and methods for extracting patterns from data. These methods include discrimination and characterization, classification ,frequent pattern mining, correlations and relationships, and regression, clustering and outlier analysis.

Clustering, in particular, is an intriguing component of data mining with applications in a variety of domains. Clustering's purpose is to find latent structures in data and organise them

into useful groupings. Cluster analysis involves dividing a large dataset into smaller subsets, with each subset representing a distinct cluster. This division is achieved by minimizing differences between objects belonging to different clusters while maximizing similarities among objects within the same cluster. Similarity and dissimilarity measurements are applied to quantitative data based on feature values that describe the objects, and distance measures such as supremum Manhattan and Euclidean distance are used[10].

3.1 Algorithms for Hierarchical Clustering:

Hierarchical clustering techniques can be performed in two ways: Agglomerative (top-bottom) and Divisive (bottom-top). An initial object is chosen in the Agglomerative technique and neighboring objects are successively merged based on minimum, maximum, or average distance measures. This process continues until the desired cluster is formed. The Divisive approach initiates with a single cluster comprising a set of objects, subsequently dividing the cluster into additional clusters iteratively until the desired number of clusters is attained. Hierarchical clustering algorithms such as BIRCH, CURE, ROCK, Chameleon, Echidna, Wards, SNN, GRIDCLUST, and CACTUS can form clusters with non-convex and arbitrary hyper-rectangular shapes [2].

3.2 Clustering techniques:

Clustering techniques based on density divide data objects into core points, boundary points, and noise points. Clusters are formed by connecting core points based on density. DBSCAN, OPTICS, DBCLASD, GDBSCAN, DENCLU, and SUBCLU are examples of clustering algorithms that can generate arbitrary shapes [2].

3.3 Grid-based clustering methods

Grid-based clustering methods divide the dataset into cells in a grid pattern. This grid arrangement is used to construct clusters. Grid algorithms construct clusters using subspace and hierarchical clustering techniques. Grid algorithms provide faster processing than other clustering algorithms. However, uniform grid techniques may not be enough to generate the intended clusters. Adaptive grid techniques such as MAFIA and AMR are used to build clusters with various shapes to overcome this limitation [3].

4. Methodology

4.1 Analytics Based on Big Data

The act of studying enormous quantities of information in order to uncover hidden patterns, correlations, and relevant insights that may be used to make educated decisions is known as big data analytics. To effectively analyze such vast and complex data, it becomes necessary to scale up hardware platforms. Choosing the right platform is crucial to meet user requirements within shorter timeframes. There are various big data platforms accessible, each with their own set of features. To choose the best platform for a certain application, one must

first grasp its pros and disadvantages. The chosen platform should be capable of handling increased processing demands, especially when developing analytics-based solutions [5].

The data for big data analytics is sourced from various channels, including smartphones and the data they make and consume, such as sensors embedded in common objects that provide location and weather data, social network posts, digital images and videos, and transaction records. This vast and continuously updated data is collectively referred to as big data. Online and startup firms were among the first to harness the power of big data, with companies like Facebook, Google, and LinkedIn being built around its utilization. Big data refers to datasets that are large, complex, and consist of structured, semi-structured, and unstructured data. Traditional software tools struggle to handle such data effectively. Many organizations face challenges due to the sheer volume, velocity, and variety of data that surpasses their current processing capacity. This information is often disorganized, fragmented, and difficult to get.

4.2 Algorithm for K- means

The K-means clustering algorithm is a popular method for partitioning a dataset into K distinct clusters. The algorithm aims to minimize the within-cluster sum of squares, also known as the squared Euclidean distance, between data points and their respective cluster centroids. The equation for the K-means clustering algorithm can be expressed in modules:

1. Initialization
2. Assignment
3. Update:
4. Repeat Steps 2 and 3 until convergence:

- Calculate the new Euclidean distances between data points and the updated cluster centroids.
- Reassign data points to the cluster with the closest centroid.
- Recalculate the cluster centroids.

The algorithm iterates until the cluster assignments and centroids stabilize, typically when there is no further change in the cluster assignments or a maximum number of iterations is reached. The objective of the K-means algorithm is to find the optimal cluster centroids that minimize the within-cluster sum of squares. This is achieved through an iterative process of assigning data points to clusters and updating the centroids until convergence is reached.

K-means clustering solves

$$\text{Arg min } c\sum i=1k, \sum x\in ci, d(x,\mu i)=\text{arg min } c\sum i=1k \quad (1)$$

The equation you provided is the objective function for the K-means clustering algorithm, which seeks to minimize the sum of squared distances between data points (x) and their respective cluster centroids (μi).

The notation "arg minc" denotes that we are finding the value of c (the cluster assignments) that minimizes the sum of squared distances. The outer summation ∑i=1k represents the sum over all clusters, where k is the total number of clusters. The inner summation ∑x∈ci d(x,μi) calculates the squared Euclidean distance between each data point x in cluster ci and its centroid μi.

By minimizing this objective function, the algorithm determines the optimal assignment of data points to clusters and the optimal positions of the cluster centroids. This process iterates

until convergence, resulting in a clustering solution that minimizes the within-cluster sum of squares.

$$\sum x \in ci \ \|x-\mu i\|^2 \qquad (2)$$

- $\sum x \in ci$ denotes the summation over all data points x belonging to cluster ci.
- $\|x-\mu i\|2$ represents the Euclidean distance between a data point x and its centroid μi.

This problem is nontrivial and is actually classified as NP-hard, meaning that finding the global minimum solution is challenging for the K-means algorithm. It may get stuck in different solutions instead.

The K-means clustering algorithm, when implemented using MapReduce, can be divided into the following stages:

1. Initialization: The newly received information set is partitioned into sub-datasets, which are then constructed into Key, Value> lists and passed to the map function.

- As the first clustering centroids, k points are chosen at random from the datasets.

2. Mapper: - To update the cluster centroids, compute the distance between each point in the provided datasets and the k centroids.

- Assign each data point to the cluster closest to it until all of the data has been analysed.
- Return the ai, zj> pair, where ai indicates the cluster zj's centre.

3. Reducer:

- Read the pairings ai, zj> from the Map stage.
- Collect all data records and form k clusters by assigning each data point to the nearest cluster.

-Calculate the average of each cluster to identify the new cluster center, which leads to the repositioning of the clusters.

The initial cluster positions must be carefully picked, and there are several common methods:

1. Forgy: Assign the k cluster coordinates to k randomly picked observations from the dataset.
2. Random partitioning: At random, assign a cluster to each observation and compute means as per step 3.

The initial cluster positions must be carefully picked, and there are several common methods:

1. Forgy: Assign the k cluster coordinates to k randomly picked observations from the dataset.
2. Random partitioning: At random, assign a cluster to each observation and compute means as per step 3.

This iterative approach is repeated until the centroids reach a stable solution.

Data points with cluster memberships are output.

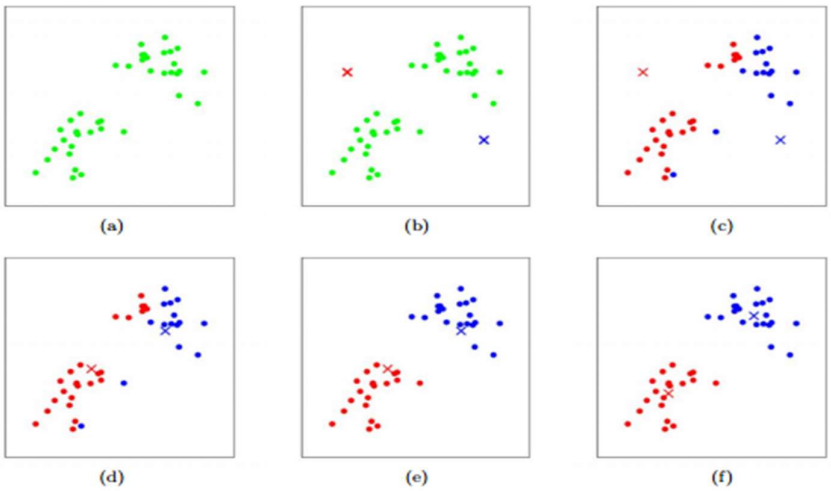

Figure 1. K-means algorithm.

In the given visualization, training examples are represented by dots, and cluster centroids are represented by crosses. The process of k-means clustering is illustrated through several steps.
(a) The original dataset is shown.
(b) Random initial cluster centroids are selected.
(c-f) Two iterations of the k-means algorithm are shown.
Training examples are assigned to the nearest cluster centroid in each iteration, which is represented by colouring the training examples the same colour as the given cluster centroid. Each cluster centroid is moved to the mean of the points assigned to it after assignment. The goal of the clustering issue is to form cohesive clusters from the supplied training set x(1),...,x(m). While feature vectors x(i) Rn are provided for each datum, no labels y(i) are provided, making this an unsupervised learning task. The purpose is to predict k centroids and label each data point with c(i).

5. Clustering by K-Means With Map-Reduction Technique

The first stage in building the MapReduce code for the K-means algorithm is to specify and analyse the implementation's input and output. The input is made up of pairs, where the "key" is the cluster mean and the "value" is a serializable vector from the dataset. Two files are required to implement the Map and Reduce functions. The first file comprises clusters with centroid values, whereas the second file contains the clustered items. These two files are created as the first step in clustering data using the K-means algorithm and Apache Hadoop's MapReduce technique.

The MapReduce procedures for K-means clustering are implemented using a specific algorithm. Before running the Map function, the set of centroids is initially placed in the Hadoop Distributed File System (HDFS) input directory, and they comprise the "key" field in the pair. The Mapper procedure contains instructions for calculating the distance between the requested dataset and the cluster centroids in the cluster set. The Mapper function computes the distance between the object value and each cluster centroid, while also keeping note of

which cluster the item is closest to. When the distance calculation is finished, the object is assigned to the nearest cluster.

The centroid of each cluster is recomputed when all objects have been allocated to their respective clusters. The Reduce procedure does this re calculation, as well as adjusting the cluster structure to prevent the creation of clusters with extreme sizes. Finally, the new collection of objects and clusters is written back to memory and is ready for the next iteration when the centroid of each cluster is updated.

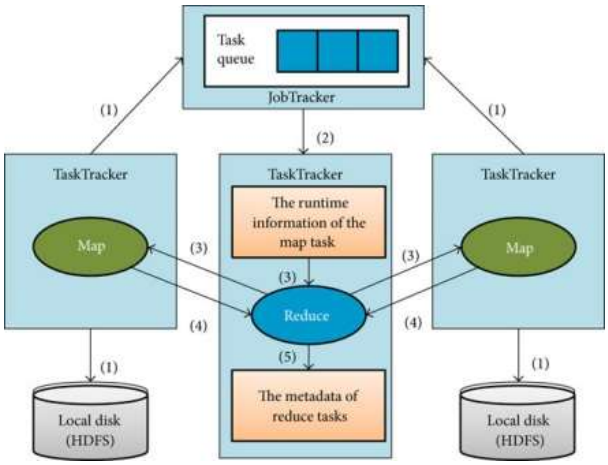

Figure 2: The acquisition of metadata decreases tasks.

5.1 Repartitioning

Algorithm 1: Repartitioning algorithm.

Data: A = [a1, a2, ..., an], K

Result: R: an index of subsequence

Step 1: low ← max(a1, a2, ..., an)

Step 2: high ← n

Step 3: num ← 0

Step 4: while low < high do

Step 5: mid ← (low + high) / 2

Step 6: for each ai ∈ A do

Step 7: sum ← sum + ai

Step 8: if sum > mid then num++

Step 9: sum ← ai and R ← R ∪ i

Step 10: end

Step 11: end

Step 12: if num ≤ K then high ← mid - 1

Step 13: else if num > K then low ← mid + 1

Step 15: end

Step 16: return R;

The repartitioning algorithm takes a sequence of values A and the desired number of

subsequences K as input. It aims to divide the sequence into K subsequences based on certain conditions, optimizing the repartitioning process.

The algorithm initializes the variables low and high based on the maximum value in the sequence A. It also initializes the variable num to keep track of the number of subsequences created.

It then enters a loop where it calculates the midpoint (mid) between low and high. For each value ai in A, it accumulates the sum and checks if it exceeds the midpoint. If it does, it increments num, resets the sum, and adds the index i to the result R.

After the loop, it checks the value of num. If it is less than or equal to K, it updates high to mid - 1, indicating that the desired number of subsequences has been achieved or can be reduced. If num is greater than K, it updates low to mid + 1, indicating that more subsequences are needed.

Finally, when the loop ends, the algorithm returns the resulting indices of the subsequences in R.

5.2 Comparative Cluster Techniques

Volume:

Volume refers to the capability of an algorithm to handle large amounts of data. When considering clustering algorithms, volume can be assessed based on two factors:

a. The data set's size.

b. A high degree of dimensionality.

Handling Outliers: The size of the data collection is an important concern in the context of volume. A data set is made up of attributes that can be category, nominal, ordinal, interval, or ratio-based. Many clustering methods are built to handle both numerical and categorical data. Another factor to consider is high dimensionality. As the size of the data set grows, so does the number of dimensions, resulting in the curse of dimensionality. Outliers, or noisy data, present a challenge to clustering algorithms since they are difficult to combine with other data points.

Variety:

The ability of a clustering method to handle different sorts of data sets, such as numerical, categorical, nominal, and ordinal data, is referred to as variety. The variety criterion is used while evaluating clustering techniques.

a. Data set type.

c. Cluster form.

Although the size of a data set can vary, several clustering methods are designed to handle enormous data sets for big data mining.

Cluster shape is affected by the amount and type of data set, which influences the shape of the clusters created by the algorithm.

Velocity:

The computational efficiency of a clustering algorithm, as measured by the running time complexity, is referred to as velocity. The time complexity of an algorithm determines how efficiently it can perform computations.

A clustering algorithm with lower time complexity will have faster execution. Time complexity calculations are often performed using Big O notation.

Value:

To accurately process data and form clusters with minimal computation, input parameters play a crucial role in clustering algorithms. Finding the right combination of input parameters can greatly impact the value provided by the algorithm.

5.3 MapReduce Processing Model

Hadoop MapReduce is a parallel processing framework designed to handle big data efficiently. It consists of two main functions: Map and Reduce.

The Map function is responsible for filtering and sorting large data sets. It processes the input data and performs necessary transformations or operations on each data item. This function is executed in parallel across multiple nodes, enabling efficient processing of big data.

The Reduce function performs the summarization operation by combining the results generated by the Map function. It takes the intermediate outputs and produces a final output that is more concise and meaningful. The Reduce function is executed on the master node, where it collects and combines the outputs from the Map phase.

Hadoop's HDFS (Hadoop Distributed File System) and MapReduce framework were inspired by Google's filing system, known as Google File System (GFS). GFS is a distributed file system developed by Google, It offers organised and efficient access to data utilising massive clusters of commodity servers. The master node accepts input data during the Map phase and divides the overall problem into smaller sub-problems. In a multi-level tree structure, these sub-problems are subsequently divided across worker nodes. Each worker node handles its assigned sub-problem, runs the Map function, and returns the intermediate results to the master node. The Reduce function aggregates the outputs of all sub-problems, which are collected on the master node, during the Reduce phase. The final output, representing the consolidated outcome of the entire MapReduce process, is subsequently generated. Each Map function is linked to a Reduce function, which allows for efficient data processing and aggregation.

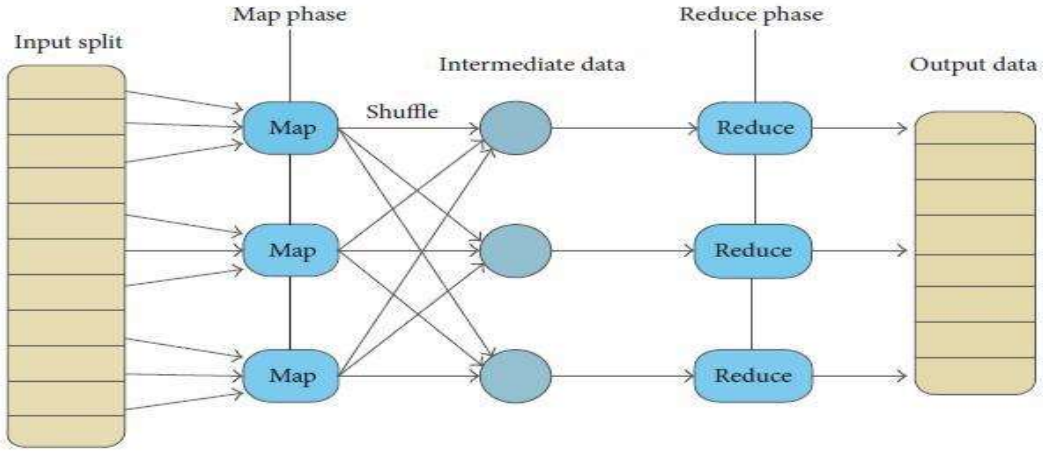

Fig 3: Model of Map Reduce Programming

The Map Reduce operation mechanism is as follows:

(1) Input: The Hadoop-based MapReduce framework requires the implementation of Map and Reduce functions, as well as the specification of input and output locations and other operational parameters. The huge input data in the directory is separated into independent data blocks during this stage, which are subsequently processed in parallel by the Map function.

(2) Map: The MR framework views the application's input as a collection of key-value pairs

(key, value>). The user-defined Map function is run in the Map stage to process each key-value pair, yielding a new set of intermediate key-value combinations <key, value>.
(3) Shuffle: The Shuffle stage is used to ensure that the input to the Reduce function is sorted. For each Reduce job, the framework leverages HTTP to extract the relevant key-value pairs <key, value> from the Map outputs. Based on the key values, the MapReduce framework groups the input for the Reduce phase.
(4) Reduce: The intermediate data is traversed for each unique key in this phase, and the user-defined Reduce function is executed. The Reduce function takes as input key, a list of values and returns a set of new key-value pairs <key, value>.
(5) Output: In the last stage, the Reduce phase results are written to the given output directory location, producing the necessary MapReduce computation output.

6. Discussion of the Findings

6.1 Setup for the experiment

We put up a four-node cluster on AWS to implement the k-means algorithm. The cluster is set up as follows:

1. a single Master Node (of the m4 instance type)
2. Four Slave Nodes (instance type m4)
3. Apache Hadoop 2.4.1
4. JDK 1.7

We used the cluster in this distributed context to implement and execute the optimised repartitioned k-means clustering method, and we saved the results for analysis.

6.2 Data Set Description:

For scaling the optimized repartitioned k-means clustering algorithm, we used a smart city dataset. Specifically, we utilized a pollution dataset European Environment Agency (EEA) Air Quality Data. The EEA provides air quality data for European countries. Their dataset includes measurements of pollutants like PM2.5, PM10, SO2, CO, and O3 where it consisting of 449 files. Each file contains approximately 17,500 observations related to the pollutant ratios of five attributes.

6.3 Evaluations:

We executed the scaled k-means algorithms over Hadoop MapReduce on 10 distinct data samples to evaluate their performance. We generated and analysed the inter-cluster and intra-cluster similarity metrics after executing the algorithm. Distance between clusters: The distance (d(i,j)) between two clusters is calculated by measuring the distance between their centroids.
Distance between clusters: This metric computes the distance between any two objects in a cluster.

The table and image below show the experimental results of the K-means algorithm on several data samples with k=3 clusters.

R Table 1. Execution results of An optimized repartitioned K-means cluster algorithm

Sample	Sample size	Inter – Cluster Density	Intra -Cluster Density
S1	68290	0.546309	0.629142
S2	1576718	0.571887	0.750337
S3	2368512	0.563767	0.74014
S4	3153530	0.5684	0.748691
S5	3942470	0.577079	0.812399
S6	4732842	0.621366	0.686724
S7	5522887	0.573842	0.75722
S8	6312932	0.565958	0.783907
S9	7099392	0.57797	0.714998
S10	7887974	0.582288	0.781926

Each row in the table corresponds to a specific sample, and the corresponding values in the Inter-Cluster Density and Intra-Cluster Density columns indicate the performance of the clustering algorithm for that particular sample. The goal of the algorithm is to minimize the inter-cluster density and maximize the intra-cluster density, as it signifies better separation between clusters and higher cohesion within clusters.

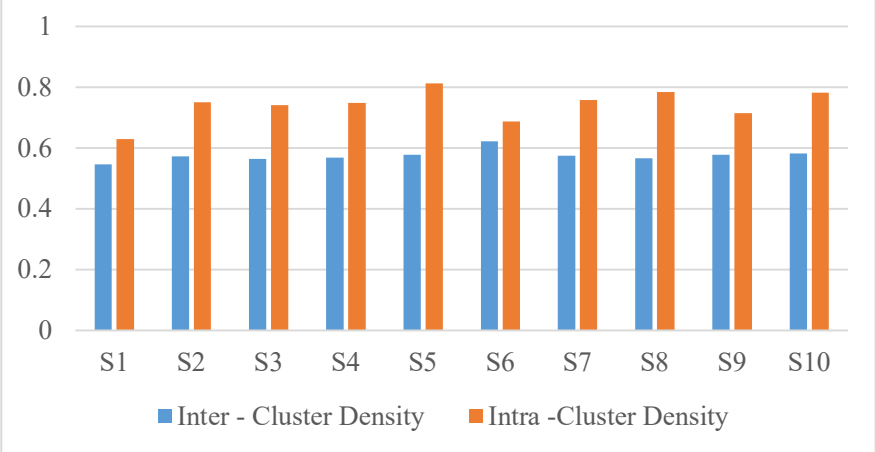

Figure 4. Comparison between Inter-cluster and Intra Cluster

The results for each sample show how the algorithm performed in terms of clustering quality, as indicated by the density values. It can be observed that the performance varies across different samples, with some samples achieving better clustering results compared to others.

Sample S5 has the Inter-Cluster Density with a value of 0.577079. This suggests that the data points within each cluster in S6 are closely related and exhibit high cohesion.

Sample S5 has the highest Intra-Cluster Density with a value of 0.812399. This indicates that the centroids of different clusters in S5 are well-separated, suggesting distinct

and non-overlapping clusters. Sample S6 has the highest Inter-Cluster Density with a value of 0.621366. This indicates that the data points within each cluster in S6 are closely related and exhibit high cohesion. Inter-Cluster Density: The values of inter-cluster density range from 0.546309 to 0.582288 across the different samples. The inter-cluster density represents the separation or dissimilarity between different clusters. Higher values indicate greater dissimilarity between clusters. Intra-Cluster Density: The values of intra-cluster density range from 0.629142 to 0.812399 across the different samples. The intra-cluster density represents the compactness or similarity within each cluster. Lower values suggest that the objects within a cluster are closer to each other in terms of their attributes.

7. Conclusion

In this, propose an optimized repartitioned k-means clustering algorithm specifically tailored for large datasets containing around 10 million objects. The algorithm harnesses the power of Hadoop and MapReduce to achieve high-performance analysis of big data. Its objective is to compute inter and intra cluster measurements, optimizing cluster density. The proposed approach addresses the challenges associated with clustering large datasets effectively. The results demonstrate promising outcomes in terms of intra-cluster density and inter-cluster density. However, further extensive evaluation and experimentation are required to establish the benchmark for clustering quality.

References

1. Murty, M.N. Clustering large data sets, Soft Computing approach to Pattern Recogniton and Image Processing 53, 41-63 (World Scientific, Singapore, 2002) ISBN: 981-238-251-8
2. Jain, A. K. and Dubes, R.C. Algorithms for clustering data (Prentice Hall, New Jersey, 1988)
3. Fasulo, D. An Analysis of Recent Work on Clustering Algorithms (1999), http://citeseer.ist.psu.edu/fasulo99analysi.html
4. MacQueen, J. Some methods for classification and analysis of multivariate observations, Proc. 5th Berkeley Symp. Math. Stat. and Prob., 1, 281-97 (Univ. of Calif. Press, 1967)
5. Kaufman, L. and Rousseeuw, P.J. Finding Groups in Data. An Introduction to Cluster Analysis (John Wiley & Sons, Canada, 1990)
6. Berkhin, P. Survey of Clustering Data Mining Techniques (2002), http://citeseer.ist.psu.edu/berkhin02survey.html
7. Jain, A. K. Data Clustering: 50 Years beyond K-Means, Pattern Recognition Letters. 31, 8, 651-666. (2009)
8. Su, T. and Dy, J. G. In search of deterministic methods for initializing K-means and Gaussian mixture clustering, Intelligent Data Analysis 11 (4), 319-338, (IOS Press, 2007)
9. Bradley, P. S. and Fayyad, U. M. Refining Initial Points for K-Means Clustering, Proc. 15th Intl Conf on Machine Learning, 91 – 99, (Morgan Kaufmann Publishers, San Francisco, 1998) ISBN: 1-55860-556-8 , http://research.microsoft.com/~fayyad
10. Fayyad, U., Reina, C. and Bradley, P.S. Initialization of Iterative Refinement Clustering Algorithms, Proc. 4th Int Conf. on Knowledge Discovery and Data Mining (KDD98), (AAAI press, 1998), http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.44.4060
11. Pena, J.M., Lozano, J.A. & Larranaga, P., An empirical comparison of four initialization

methods for the K Means algorithm, Pattern Recognition Letters , 20 , 10, 1027 - 1040 (1999) ISSN:0167-8655

12. Meila, M. and Heckerman, D. An experimental comparison of several clustering and initialization methods (1998), http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.8.1159

13. Zhang, Z., Dai, B. T. and Tung, A. K.H. On the Lower Bound of Local Optimums in K- Means Algorithm, Proc. 6 th Intl Conf on Data Mining ICDM'06, 775 – 786 (IEEE Computer Society, Washington DC, 2006) ISBN: 0-7695-2701-7, DOI: 10. 1109/ ICDM.2006.118
14. Daoud, M. A., Venkateswarlu, N. B. and Roberts, S. New Methods for the Initialization of Clusters, Pattern Recognition Letters 17, 5, 451 - 455 (1996) ISSN: 0167-8655
15. D. Arthur and S. Vassilvitskii, “k-means++: the advantages of careful seeding,” in Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1027–1035, 2007.
16. T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, “A local search approximation algorithm for k-means clustering,” Computational Geometry: Theory and Applications, vol. 28, no. 2-3, pp. 89–112, 2004.