Available online at www.sciencedirect.com

ScienceDirect

Future Computing and Informatics Journal 3 (2018) 247–261
http://www.journals.elsevier.com/future-computing-and-informatics-journal/

Partition based clustering of large datasets using MapReduce framework: An analysis of recent themes and directions

Tanvir Habib Sardar, Zahid Ansari*

Computer Science and Engineering, P.A. College of Engineering, Mangalore, India

Received 28 October 2017; revised 17 April 2018; accepted 12 June 2018
Available online 18 June 2018

Abstract

Data clustering is one of the fundamental techniques in scientific analysis and data mining, which describes a dataset according to similarities among its objects. Partition based clustering algorithms are the most popular and widely used clustering technique. In this information era, due to the digitization of every field, the huge volume of data is available to data analysts. The quick growth of such datasets makes decade old computing platforms, programming paradigms, and clustering algorithms become inadequate to obtain knowledge from these datasets. To cluster such large datasets, Hadoop distributed platform, MapReduce programming paradigm and modified clustering algorithms are being used to shrink the computational time by distributing clustering job across multiple computing nodes. This paper provides a comprehensive review of Hadoop and MapReduce and their components. This paper aims to survey recent research works on partition based clustering algorithms which use MapReduce as their programming paradigm. In many recent works, the traditional partition based clustering algorithms like K-means, K-prototypes, K-medoids, K-modes and Fuzzy C-means are modified for MapReduce paradigm in order to obtain different clustering objectives on different datasets for reducing the computational time. The contribution of this paper is (1) to provide an overview of clustering challenges in real world large dataset clustering and the role of MapReduce programming paradigm and its supporting platforms in dealing the challenges for several tasks in different datasets and (2) to review recent works in partition based clustering using MapReduce paradigm for different clustering objectives for different datasets employing different strategies.

Keywords: Data mining; Partition-based clustering; MapReduce; Hadoop

1. Introduction

1.1. An overview of cluster analysis

Data clustering techniques are considered among one of the popular data mining techniques. Data mining techniques are widely used for extracting knowledge from different datasets [1]. Data mining methods provide interesting and meaningful patterns by analysis of large repositories of datasets [2]. Clustering or cluster analysis is one of the elementary data mining techniques used in scientific data analysis [3]. Clustering is also known as unsupervised learning which produces different clusters of input dataset and is able to place any new input in the suitable cluster [4]. The outcome of clustering leads to setting down on numerous clusters or groups by determining the resemblances of the data points of a dataset [5]. This is performed such that data points in the same cluster/groups are alike and data points belonging to other clusters/groups are unlike [6]. The objective of a clustering process is determined by a specific necessity provided by the user [7]. Clustering algorithms are widely and extensively used for data analysis in different domains and datasets such as biological

* Corresponding author.
E-mail addresses: tanvir.cs@pace.edu.in (T.H. Sardar), zahid_cs@pace.edu.in (Z. Ansari).
Peer review under responsibility of Faculty of Computers and Information Technology, Future University in Egypt.

https://doi.org/10.1016/j.fcij.2018.06.002

data [8,9], weblog data [10], image data [11], text data, market data, telecommunication data, medical data etc.

Clustering techniques are still significant research issue in data mining process and it is basically a continuous focus on data mining for finding optimum clusters [12]. Clustering techniques have some typical requirements as explained below [13]:

- Clustering algorithms require dealing large data volumes effectively.
- Clustering algorithms should provide with quality clusters from diverse input datasets containing binary, numerical and categorical data attributes.
- The shape of clusters obtained from clustering algorithms should be of arbitrary shape. For example, depending on input datasets clustering result require to generate oval shaped, “Z” shaped or “S” shaped clusters.
- Clustering algorithms should generate quality clustering from datasets containing low or high dimensional data.
- It is common for a voluminous dataset to encompass noise in terms of missing, inaccurate or erroneous data. A clustering algorithm is considered robust is it can produce good quality clusters with a noisy input dataset.
- The resulting clusters should be easy to understand and unambiguous.

However, all the above requirements are practically unavailable in any particular clustering algorithm. Clustering algorithms can be categorized into five major groups based on its inherent method used for data classification in clusters. These are i) Partitioning algorithms ii) Hierarchical algorithms iii) Density-Based algorithms iv) Grid-Based algorithms and v) Model-Based algorithms [14]. Varieties of algorithms under each group are already available in the literature and were effectively applied to real-life data mining problems [15].

The most popular and widely used algorithms are arguably partitioning based due to the fact that it is simple to understand, easy to implement and time complexity is minimal as compared to other techniques [4]. Partition based clustering algorithms create k number of partitions of a given data set, each representing a cluster. A typical data clustering process starts with a set of data objects and partitions them into k clusters founded on similarity distance measures such as Euclidean distance. Some of the requirements that are accomplished by partition-based clusters are: i) Each cluster must consist of at least one data object in it and ii) in non-fuzzy clustering algorithms, each object should be present in only one cluster [14]. K-means, K-medoids, K-modes, and FCM are examples of partition based algorithms [14].

Clustering algorithms can also be categorized as Soft Clustering and Hard clustering [16]. Traditional clustering methods are hard in nature. Here one object must be part of a single cluster after convergence of the clustering process. Traditional clustering algorithms are modified with soft computing techniques to meet real-world requirements of clustering. This modification leads it to a soft technique of clustering. Hence, hard clustering denotes parting data objects into separate groups maintaining every particular data object goes to accurately in one group. In soft clustering, a particular data object can associate in more than one group, and connected to the other data objects through a membership calculation. The membership value of a data object shows the orientation of that data object with respect to a specific cluster [17].

Fuzzy sets are widely incorporated with clustering algorithms for soft clustering. It enables the clustering result to demonstrate the level of closeness of a particular object to a cluster. Soft clustering assigns each data object a degree of membership valued within the range of 0−1 with different clusters. Fuzzy clusters are implicit and represent the true nature of a clustering than its non-fuzzy version [16,18]. If a data object possesses membership value closer to 1 for a particular cluster then it indicates that this object is suitable for that particular cluster.

In terms of quality of clustering, fuzzy clustering provides better clustering than non-fuzzy clustering. Fuzzy clusters provide more insight and knowledge about the proximity of the data objects to the different clusters. The overlapping clusters can also be easily recognized by the membership values in case of fuzzy clustering [19]. There are many recent research works found in the area of fuzzy clustering in order to obtain different clustering objectives such as web log analysis, image segmentation, and bioinformatics to name a few [20]. The most popular and widely used hard and fuzzy partitioning algorithms are K-means and Fuzzy C-means respectively [21].

1.2. Challenges due to growing size of data and parallel techniques

Over the years, dataset volumes are developing quickly due to business and manufacturing process automation, and less expensive storage device availability. These voluminous datasets generated by an enterprise are sometimes distributed across their branch locations. One of the reasons for this growing quantity of data is web server log file maintenance, which records user's activity in a particular website. Analyzing these voluminous web server log files provide an enterprise the behavior of any particular user [17]. The job of data analysts becoming difficult to effectively analyze and retrieve knowledge from such voluminous data generated from almost all field of work [22], like bioinformatics [23,24], biomedical [25,26], cheminformatics [27], web [28] to name a few. Clustering these large datasets effectively becomes a problem for traditional clustering algorithms [3]. Traditional clustering algorithms take huge time in clustering such voluminous datasets [14]. The existing data analysis tools also suffer similar inefficiency problem while clustering voluminous datasets [29]. Therefore, scalable and parallel clustering algorithms are required to efficiently cluster such a large volume of data. Also, a parallel computing model is required for parallel clustering of such large volume of data [30].

At the same time, when dataset sizes are increasing, the computer processing paradigm is also changing to multi-core and many-core systems. There is an increasing shift

in computing technique to parallel and distributed mechanisms. MapReduce, proposed by Google in 2004, is a programming paradigm which process large datasets in parallel among a series of connected nodes. MapReduce performs a task by dividing the task into two functional blocks named map and reduce. MapReduce automatically handles input and output related mechanisms require for map and reduce function execution [31]. Hadoop is a framework which enables efficient large dataset processing by application programs written based on the MapReduce paradigm [32]. Hadoop works on master-slave architecture where one master coordinates many slaves. Hadoop has two major parts: Storage and Processing. The storage part is managed by Hadoop Distributed File System (HDFS) and processing part is designed based on MapReduce programming paradigm. Using these two parts, Hadoop can efficiently process very large datasets by managing the process of the distribution of data in the nodes, processing the data in the nodes and accumulating result from the slave nodes [33,34]. This paper summarizes a literature review on efforts made on parallelizing partition based clustering algorithms using MapReduce framework.

The following are the challenges faced in dealing with large datasets:

- The challenge of effectively analyze and retrieve knowledge from real-world voluminous datasets is faced by data analysts.
- It is challenging to parallelize traditional clustering algorithms in MapReduce paradigm for large dataset clustering.

1.3. The different utilization of the MapReduce framework: an overview

Real-world datasets such as web log data, image data, and bio-medical data necessitate a huge amount of storage space. Sometimes it consists of terabytes of storage space. MapReduce [35,36] is the best choice for clustering of these voluminous datasets in distributed clusters and multi-core systems [37,38]. Many researchers have been conducted to make MapReduce familiar to the users and to make MapReduce best for processing large datasets [31]. In order to exploit better efficiency though MapReduce processing of data-intensive [35] real-world datasets, many recent project works have improved the APIs and experimented MapReduce using different configuration parameters [39–45]. MapReduce is successfully implemented for processing large datasets for different applications on distinct platforms [39,46–49]. Although MapReduce is designed to process large data sets on a cluster of computing nodes, it is also used for developing applications for multi-core computers [47,50,51].

The name of MapReduce is given as its execution is largely dependent on two functions named map and reduces. The input to reduce function is the output of the map functions. The input dataset is firstly divided into parts and then map functions are allocated with a particular data chunk. The processing result of this dataset is then fed to reducer for further processing and accumulation of results [52]. The MapReduce model becomes so popular due to the following reasons:

- The biggest advantage is that it provides automatic parallelization and distribution.
- It's tolerant to faults. Individual tasks can be retried.
- A clean abstraction for developers provided.
- MapReduce programs are usually written in Java, which in turn one of the popular and most widely used by developers.
- Hadoop comes with standard status and monitoring tools.

1.4. MapReduce design

In MapReduce, the parts of a dataset are processed separately by map functions (known as mappers also). The reduce functions (known as reducers also) are provided with the results from the mappers. In this way, MapReduce process a large dataset by partitioning the jobs between mappers and reducers [51]. The dataset input to the MapReduce must first transform into key and value pairs as the mappers and reducers can only work in this format.

Mapper: $(k_1, v_1) \rightarrow [(k_2, v_2)]$

Reducer: $(k_2, |v_2|) \rightarrow [(k_3, v_3)]$

where, k_1 & k_2 are in-key and out-key respectively, and v_1 and v_2 are in-value and out-value respectively. k_3 and v_3 are final keys and final value respectively. $|v_2|$ is the out value list.

The mappers execute in parallel on different data splits of an input dataset and output intermediate pairs of keys and values. The reducers obtain these values from mappers and calculate the final value for each key. Fig. 1 provides the working style of MapReduce with mappers and reducers.

1.5. A MapReduce program execution strategy

The input dataset is automatically partitioned into subsets of the original datasets and allotted to mapper functions for parallel processing in different nodes of a cluster. The intermediate values and the keys outputted by the mappers are then allotted to the reducers. The number of partitions (p) and the partitioning function are specified by the user [53]. The following sequence takes place when the user's application program runs on MapReduce paradigm: (1) application program creates separate processes for master and slave nodes, (2.1) master node assigns nodes for map job and (2.2) also assigns nodes for reduce job, (3) input dataset is partitioned and each partition is assigned to a particular mapper node, (4) output of the mapper job is stored in a file on local nodes correspondingly, (5) the key and intermediate values stored in

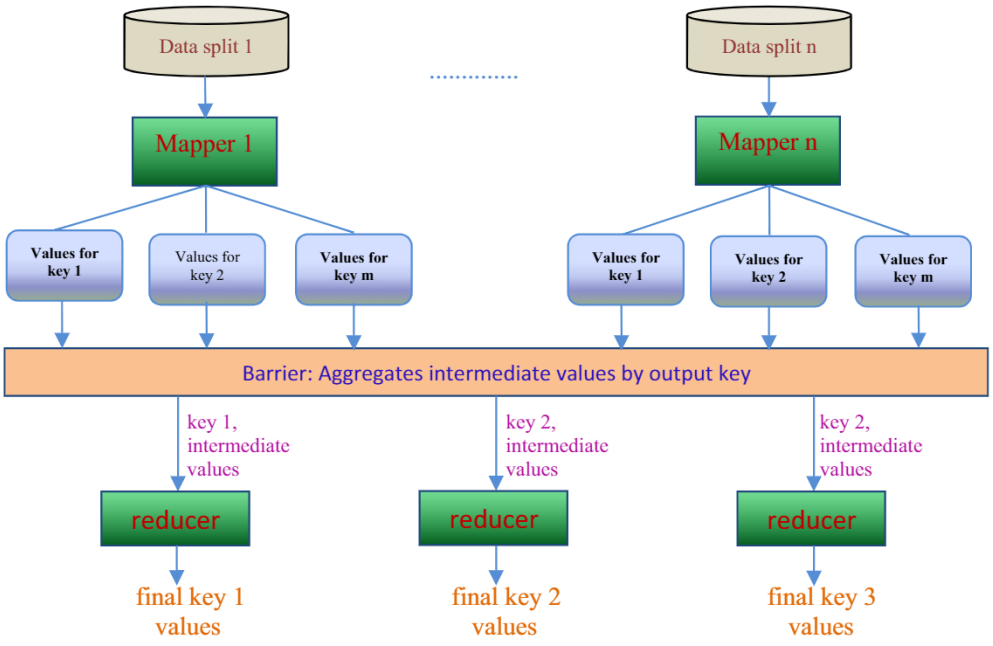

Fig. 1. MapReduce overview.

local files are sent to nodes assigned for reduce job, (6) after reduction the final results are stored in the master node. The MapReduce execution strategy is provided in Fig. 2.

1.6. MapReduce applications

Google in 2008 used to process 20 PB of data daily with the help of 100,000 MapReduce processes [54]. These data in turn required framework for analyzing its large volume efficiently in short span of time. One of the smartest solutions for such a problem is obviously MapReduce programming paradigm which helps it to execute in parallel by distributing a job into a number of clusters of nodes. As provided in section 1.4, MapReduce is extensively used for getting the advantage of large data processing in several problem domains [55]. There are many applications customized for MapReduce paradigm to

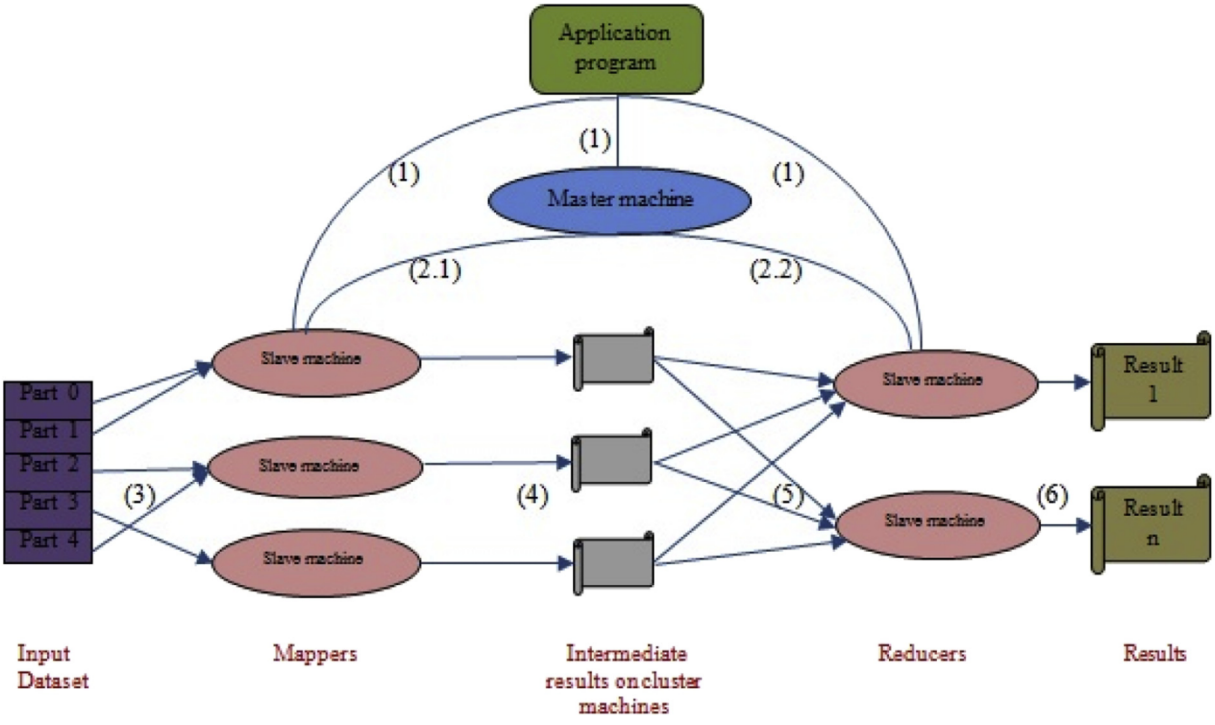

Fig. 2. MapReduce strategy of Data processing.

attain certain objectives. Some of the examples includes analysis of different kinds of log files [56], determination of crops for agricultural planning and use of pesticides and fertilizers [57], pre-processing and extraction of user sessions from the web server log files [58], providing current and yearly trends of crops prices to the farmers [59], segmentation and classification in medical image analysis [60] etc. It is also found useful in applications such as web-graph inverting [61], SMS message mining [62], encoding and decoding large twister code [63], parallelizing particle swarm optimization algorithms [64], calculation of response time using generalized stochastic petri nets for analysis of efficiency in communication systems [55] and to optimize spatial queries [65].

In Table 1, we have provided details of some of the tasks parallelized using MapReduce:

2. Implementations of MapReduce programming paradigm

2.1. Hadoop

Hadoop [32] is an open source distributed computing framework for large dataset processing using MapReduce [51]. Hadoop executes a program in a cluster of connected nodes following master-slave architecture. The master node partitions the datasets transfers the datasets to the slaves and combines the result from slave nodes after processing. It can process petabytes of data efficiently through a cluster of nodes made of commodity hardware [51]. Hadoop consists of several parts and subprojects [32,70]. The two important parts of Hadoop are provided below:

- HDFS: Hadoop Distributed File System is the storage part of Hadoop.
- MapReduce: This is the computational paradigm used by Hadoop for processing data.

HDFS is an efficient and fault-tolerant distributed file system designed by Hadoop. HDFS of the master node implicitly partitions an input dataset, is distributed among slave nodes and recollects the final result from the slaves to the master. HDFS in the master node contains the metadata to store all this information. Fig. 3 provides an overview of HDFS File System.

- Client: The client [71] allows a user to access the HDFS file system.
- HDFS Master Node: The significant tasks like splitting the input dataset, monitoring and tracking slave node execution of a particular data split, replication and transfer of a data split evenly across slaves, managing the file list as well as blocks in a file, storing the attributes of files and managing other meta-data are performed by HDFS of master node [51].
- HDFS Data Nodes (Slaves): Slave nodes receive data splits from the master node and stores in its local HDFS file system. It also maintains a meta-data for keeping track of the data blocks it operates on. Slave HDFS cooperates with master HDFS by sending the periodic signal which means that the slave is working fine. Slaves can also transfer blocks to other slaves as per the instructions from master node. Hadoop can be implemented in 3 ways: [72].
- Standalone: In the standalone installation of the Hadoop only one java process executes for providing functionalities.
- Pseudo-Distributed: As the name implies, this Hadoop installation provides a flavor of Hadoop execution containing the master and slave HDFS and processes in a single machine. This kind of implementation can be used for just developing and testing a Hadoop based project but not for actual execution.
- Fully Distributed: This is a real Hadoop installation which provides a full fledge distributed storage and processing on a master-slave architecture. The processing of a job in the fully distributed mode for large data definitely provides efficiency as all the nodes can contribute as per their respective storage and processing power.

2.2. Other implementation of MapReduce

The following paragraphs describe the working mechanism of different non-Hadoop platforms for implementation of the MapReduce programming paradigm:

- GridGain: The GridGain [73] is also an open-source execution platform for MapReduce. Unlike Hadoop, GridGain is able to process real-time and non-transactional data in a distributed fashion. Even low latency applications can be efficiently processed by GridGain. GridGain provides an efficient mechanism such that a slave never waits for data. Grid Gain is documented properly and appealing to the beginners [53].
- Phoenix: Phoenix [47] is an efficient shared memory system that provides fault tolerance and resource management implicitly and works using the MapReduce model. It is well suitable for multiprocessors and multicores [53]. Mappers and reducers are crafted by Phoenix by threads. Other than mappers and reducers, Phoenix works on two special functions one of which is used for splitting the data before each step of a data processing job and another function for comparison of keys. A function named scheduler is also used by Phoenix to start MapReduce task, allocating I/O buffer locations and running mappers and reducer threads.
- Mars: Mars [48] implements MapReduce for graphical processing units. Mars actually created to provide GPUs with a standardized programming platform for its extensive parallel computing system by hiding the inner complexity of the GPUs [53]. Mappers are reducers are designed by threads in Mars. The thread assignment process is done on Mars before the MapReduce starts the mappers and reducers as the GPUs don't provide with thread scheduling. The transformation of the input dataset into <key, value> pair is done through processors by Mars as GPUs don't fetch data from disk directly [48]. One

Table 1
Tasks parallelized using MapReduce.

Objective	Tasks implemented using MapReduce	Dataset	Reference
Producing current and yearly trends of crops prices to the farmers	Data is stored in HDFS in such a format which consists of unique id for the commodity, market, and date after map operation. This transformed data is used by the users to query current prices in the different market for a particular crop.	Agricultural datasets from various sourcesare collected with the help of web scrapping tool	[59]
Crops determination for agriculture planning	Support Vector Machine, Regularized Greedy Forest (RGF), Decision Tree.	Agricultural dataset	[66]
Transforming soil and crop sensing to determine use of pesticides and fertilizers	Digital Farm Record which consists of soil temperature, moisture percentage etc.	Agricultural dataset	[57]
Determining quality of crop generalization	Regression on agricultural info. Degree index. It uses vector regression model and other classifiers.	Agricultural dataset	[67]
Preprocessing the log files	Analyze web server log file to check successful client requests and web crawler requests. The referred URL with empty values are removed also.	Log files collected from NASA Kennedy Space Centre	[58]
Exploring user session from log file	Splits all the pages visited by the IP address based on unique identity and timeout, when the time between page requests exceeds 30 min.	Log files collected from NASA Kennedy Space Centre	[58]
Analyzing different kinds of log files	Data visualization and Statistical analysis are implemented by a job scheduling algorithm	Email, Web server; Firewall logs, Call Centre logs	[56]
Segmentation and classification in medical image analysis	Support Vector Machines: The search for optimal SVM parameters. First, a file containing all possible parameter Combinations were generated. The latter serves as the input for the Hadoop job, where each line containing a value couple represents an independent map task.	ImageCLEF 2011 medical dataset	[60]
Content based image retrieval	Gaussian density functions and Huang-Hilbert Transform.	DDSM image database	[68]
Detection of hidden useful patterns from biomedical datasets by using clustering	Latent Semantic Indexing, Agglomerative Hierarchical clustering, spherical K-means algorithm	Large volumes biomedical dataset repository from PubMed.	[69]

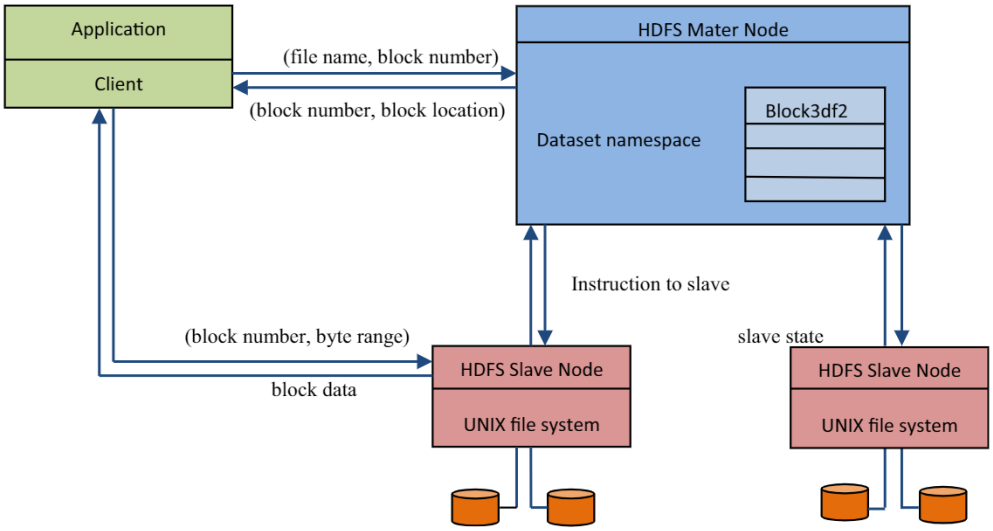

Fig. 3. HDFS file system.

thread in Mars is provided with one chunk of dataset making the number of chunks is equal to the number of threads.

- Map-Reduce-Merge: Map-Reduce-Merge [74] is better suitable to be specified as a modification of MapReduce rather than a MapReduce implementation framework. Unlike MapReduce, Map-Reduce-Merge can process mixed type of datasets simultaneously [53,74]. Map-Reduce-Merge can effectively execute relational algebra and join operations. The results obtained from the reducers can be joined by a specially designed merge phase in this model. This merge phase enables Map-Reduce-Merge with a better processing capability of large datasets than MapReduce.

3. Literature review on partition based clustering using MapReduce

Surve and Paddune (2014) [75] used MapReduce for clustering of remote sensing images. At the beginning, the color images from the remote sensing images are first transformed using the CIELAB color space. So at First, a transformation of the original image dataset results in a text file which represents, in each line, a pixel with its red, green and blue proportion. The differences in the two proportions of color are measured using Euclidean distance. There are two MapReduce jobs for calculations used in proposed K-means, one job of assigning pixels into a particular centroid and the other job for error variance checking of the pixels. The proposed work demonstrates its effectiveness in the remote sensing dataset clustering. The similar experiments are also conducted by Zhenhua Lv et al. (2010) [76] for datasets containing similar input data. One of important observation found from the experiments of this work is that the transformation of images to a text file creates a large data file and consumes a significant amount of execution time.

In order to obtain efficiency while clustering large dataset, K-means distance calculation is parallelized using MapReduce by Li et al. (2011) [52]. This work is experimented on different image datasets obtained from the UCI Machine Learning Dataset Repository. The proposed algorithm is provided in Fig. 4 and particulars are summarized below:

1) Select K points randomly from the input dataset space. These are considered as initial centroid values.
2) Assign data point subsets to the mappers in order to calculate the distance from data points and centroids.
3) After each data points are assigned to a centroid, reducers recalculate centroid positions.

The steps 2 and 3 are repeatedly executed until the new centroid values become same as compared to the old centroid value. Experimental results show that this method produces good quality clusters efficiently.

An efficient clustering on document dataset is implemented using the MapReduce paradigm by Kumar and Chandavarkar (2015) [77]. Since the MapReduce programming model requires the key and value pairs to be submitted to the MapReduce job, the key is selected as cluster center and value is the vector space values of document dataset. The Map function takes these two files as the input, the initial cluster center file in HDFS form the key field and the other file consisting of respective vector value representation of the dataset. The distance calculation from cluster center to each point in the data set is performed in the Map function, simultaneously recording the cluster to which the given vector is nearest. After processing all the vectors, the vector values are allocated to the adjacent cluster. Immediately after the Mapper execution, the computation is recalculated until it reaches a convergent point. This recalculation part goes into the Reduce routine, it also restructures the clusters to avoid the clusters with extreme size that is the clusters with too fewer data vectors or too many data vectors. These newly created clusters are written back to the disk which will be loaded as input to the next iteration. The author claims that the proposed work efficiently clustered the dataset.

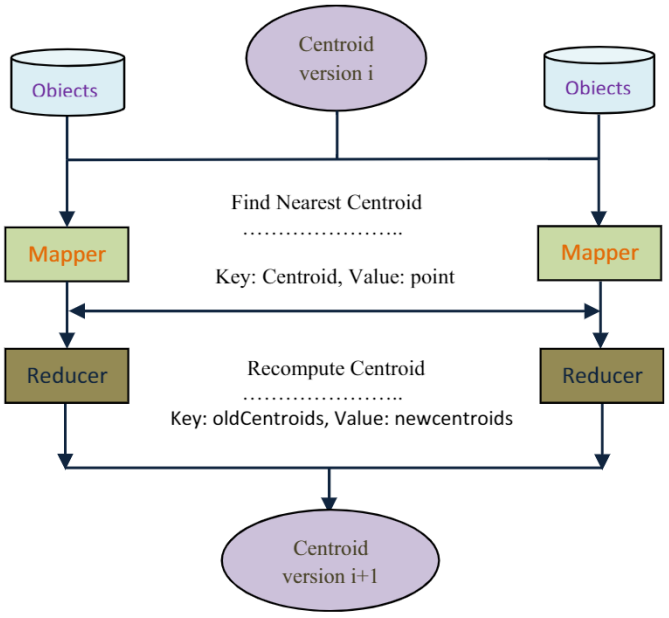

Fig. 4. Parallel K-means algorithm with MapReduce as in Ref. [52].

In a similar work on document dataset [33], the authors have designed, experimented and evaluated a MapReduce based parallel K- Means named PK-Means on a 10 node Hadoop cluster. The clustering outcome provides good quality clusters in reduced computational time as compared to traditional K-means while clustering large datasets. Table 2 provides a summary of execution time taken by PK-Means as compared to traditional K-means. This proposed algorithm is executed in few phases as specified briefly below:

Phase 1: This phase preprocess a text dataset and transforms it into a normalized numeric dataset using the vector space model. This job is accomplished using two MapReduce processes. First MapReduce process calculates the parameter values for the vector space model like term frequency-inverse document frequency. The second MapReduce process excerpts terms from the normalized dataset and produces vector space of the dataset. This phase also specifies the number of iteration for the algorithm and randomly selects centroids.

Phase 2: This step uses a mapper function so as to calculate the distance between centroids and data points. As the data size it produces is high, a combiner function is allotted which calculates the mean value of data points respectively to each centroid. The output of the combiner is sent to the reducer function.

Phase 3: This step receives output from the phase 2 and employs a reducer function to recalculate the centroid value. This process of phase 2 and 3 iterates until the new centroid value becomes same.

Table 2
The Result of traditional and PK-means Algorithm as in Ref. [33].

Algorithm used	Traditional K-means	Parallel K-means
Computers used	1	10 (Hadoop cluster)
Time taken	30 min	10 min

Although the above-reviewed works used a document dataset, there are many works available in the literature which did not use any specific dataset. Rather, those works are conducted on a dataset of points. In one of such work, the Authors [78] have modified K-means with MapReduce to cluster 15 million 2-D and 3-D data objects. K-means algorithm is parallelized and clustered on commodity hardware. This study provides an insight into the fact that as the data size for clustering is small, a MapReduce based clustering algorithm becomes ineffective in exploiting the efficiency of a data clustering job. It is because the MapReduce model consists of network and process overhead, which trade off against efficiency for small datasets. However, an advantage of this work is that it has used a combiner function to gain efficiency in clustering. A similar technique is implemented in Ref. [79].

Like the previous study, a combiner is also used by Anchalia et al. (2014) for a MapReduce based K-means clustering [80]. This combiner function significantly shrinks the task of the reducer function which in turns makes the clustering to consume less time. The execution of this proposed algorithm over 10 million data points is found efficient compared to the non-combiner based MapReduce implementation. As an experimental observation author found that shuffling in MapReduce for 50 thousand points is nearly 4 s, for 500 thousand points it becomes 30 s and for 5 million data points it consumes 207 s. Thus, to reduce this consumption of execution time in shuffling and sorting in MapReduce, a combiner is designed between mappers and reducers. This combiner function demonstrates a good speed up in clustering job.

Gowru and Potnuri (2015) [81] modified K-means using MapReduce as follows: the map step finds out the cluster center values randomly into memory from a sequence input file. Iteration occurs over every cluster center for all key/value pairs in the input file. It then computes the distances and keeps the nearest center which has the lowest distance into a defined vector. Cluster centers are then written down with its computed vector to the file system. The reducer receives all the associated vectors of each cluster center then sum up associated vector values and calculates the new mean of the cluster. In this point, the algorithm compares the values between the old and the new cluster center. If the centroids values are not same then the algorithm iterates else it converges.

Li et al. (2015) [82] implemented MapReduce based K-means algorithm for the large dataset in cloud computing model. This algorithm provides an effective way to select initial centroids in order to obtain good quality clusters. The proposed algorithm consists of two main parts. The first part is to select initial cluster centers and divide the sample dataset into partitions for parallel processing. The second part assigns mappers and reducer on these data partitions. The proposed algorithm execution is depicted in Fig. 5. Experiments are conducted on a dataset obtained from a coal group enterprise. The experimental results demonstrate that the proposed algorithm provides quality clusters while consuming less computational time than the traditional K-means.

Anchalia et al. (2013) [83] proposed a parallel K-means using the MapReduce model for text data clustering. The text

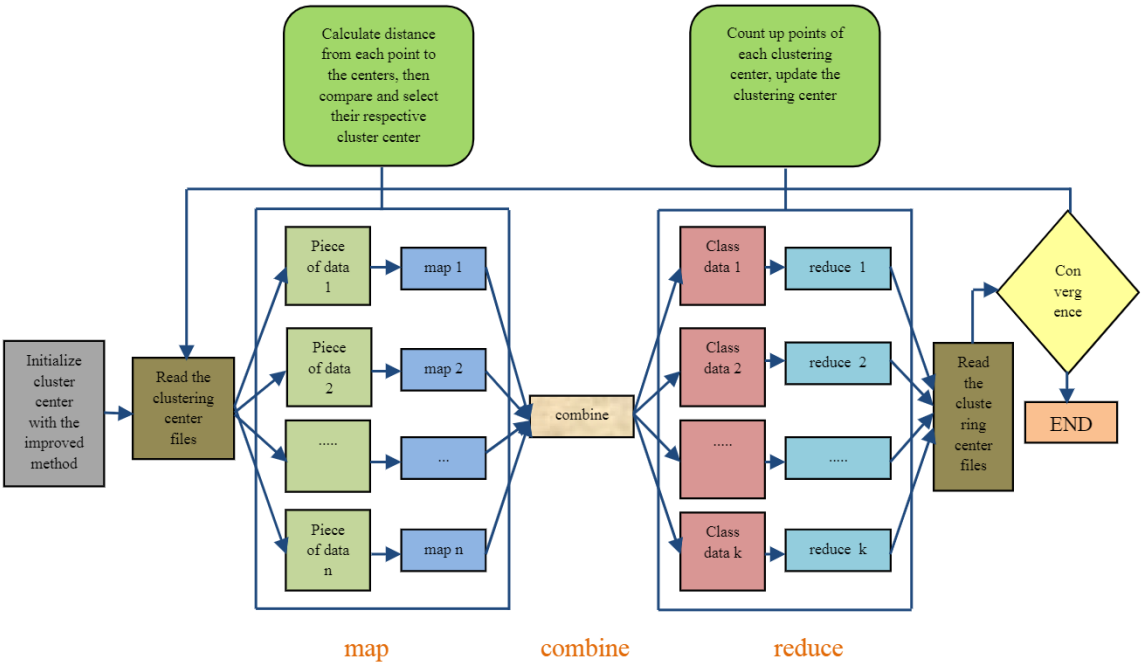

Fig. 5. Process of proposed K-means as in Ref. [82].

data is transformed to vector space model first. This transformed dataset is then inputted to the Hadoop master node. The master node creates two files: one which consists of randomly chosen centroid values from the datasets and the second, which consist of vector spaced representation of the text datasets. These two files are sent to slave nodes by the master. The mappers in the slave nodes compute the distance between centroids and vectors and assigned vectors to the centroids as per the lesser distance it computes. The restructure of these clusters are then performed through the reducers by recalculation of the centroid values. This work has not implemented any combiner between mappers and reducers. This work is also silent about the experimental result of clustering in terms of numeric quantity such as scale-up, cluster up etc.

Zhao et al. (2009) [84] provides a modification of K-means using MapReduce named PK-Means. The workflow design of this study is similar to the previous review of [83], except that this work is designed with a combiner function between mapper and reducer. The network communication overhead is reduced due to the effective use of combiner in this work. The authors used speed up, scale up and size up the comparison in order to evaluate the efficiency of the proposed algorithm. The work is found efficient in clustering large dataset using Hadoop cluster of commodity hardware. It is observed that the result of clustering in terms of speed up does not provide linearity with respect to increase in data size. The author stated that the main reason of this nonlinearity is the communication overhead between the nodes of a Hadoop cluster. The authors have not specified the type of the dataset used for experimentation.

In order to efficiently cluster a large mixed type dataset, HajKacem et al. (2016) [85] proposed a modification of the K-prototype algorithm using MapReduce. A pruning technique is employed in mappers, named triangle inequality method [86], is used in this work to achieve a reduction in distance computations of the K-prototype. The distance calculation between cluster centers and data objects are performed in mappers and recalculation of cluster centers are performed by reducers after receiving output from mappers. The clustering provides a linear speed up with respect to increase in dataset size. The experimental results also depict that the efficiency of the proposed algorithm is only effective when the input data objects are larger than 2 million.

Srinivasulu et al. (2015) [87] parallelized K-medoids using MapReduce. The mappers are provided with a global array which contains randomly chosen medoids objects. Based on these medoids, mappers compute the distance with other data objects. The combiner functions are employed, which gather intermediate outcomes from mappers. As combiners work on the same node it reduces the communication overhead for the reducers. The medians and centroids are then fetched to the reducers where recalculation of the median is performed.

The shortfall of K-means in noisy data clustering is solved by the K-medoids algorithm, but K-medoids is not efficient for large datasets because of its higher execution time. To solve this shortfall of K-medoids, the author in Ref. [88] proposed parallel K-Medoids algorithm named HK-medoids. This technique used in this work is similar to the review in Ref. [87]. The dataset is clustered by this work is the classical iris feature dataset. The algorithm HK-medoids is found effective and obtained a linear speedup.

HBase provides efficient storage of sparse information using a distributed model of HDFS. Authors Yue et al. (2016) [89] have modified K-medoids using MapReduce, named K-medoids++, and experimented on spatial datasets. The input

spatial datasets are kept on HBase as an ordered file of coordinates. The key of map function is the row number in the HBase dataset and the value is a string of the corresponding coordinate. The spatial points are split and sent to mappers. As a result, the distance between medoids and data objects is computed in parallel. All the initial medoids are stored in the file of medoids and provide to the mappers as input. The mappers calculate the medians using distance calculation between centroids and data objects. The reducers, after receiving centroid and respective data object coordinates from the mapper, recalculates the centroid value. The clustering results show that K-medoids is effective in clustering large datasets.

In order to cluster dataset using MapReduce based parallel K-modes, Tao et al. [90] first randomly selected K initial categorical objects from the dataset. These modes are input to the mapper functions which calculate the distance between data objects and modes and then assign objects to a cluster based on the highest similarity between them. This work provides the result obtained from m mappers to n reducers. Reducer process will generate X HDFS files containing X categorical clusters, where each of X is obtained from the mode of a cluster. The study is conducted on US census datasets and demonstrates speed up in terms of computational time for large dataset clustering. The study also reveals that the proposed technique is equivalent in terms of computational time for clustering of small datasets.

Fuzzy C-means is parallelized using MapReduce by Garg et al. (2015) [91]. Canopy method, used in this work for generating the data points, is also parallelized using MapReduce. The canopy generated data objects are considered as centroids and vectors as the data objects. The mapper in this work calculates the distance from the centroids and canopy generated data objects and assigns the membership value to each data object with respect to the centroids. The data objects with highest membership value are assigned in the cluster by the mappers and this result is transferred to the reducers. The reducer recalculates the centroid value and iteratively fed to the mapper again until reaches to a particular iteration. The result of this technique demonstrates a speed up for large datasets.

The Stopping criterion of FCM is that it shall continue its execution until centroids don't alter its value outside the threshold of convergence and neither the data objects assigned to the centroid don't alter. This increase in a number of iterations for large datasets overlapping clusters and becomes problematic for in-memory data management. In order to deal with this problem, Mathew and Chandran (2015) [92] used MapReduce. The map job calculates the Euclidean distance between randomly selected centroid points and the vector points. Membership matrix is then calculated using Euclidean distance. Additionally, it manages a data cache to keep track of all data objects associated to a particular centroid. The reducer performs the recalculation of the centroid values and fed the newly calculated centroid value to the mappers, iteratively, till convergence.

Garg et al. (2013) [93] compares K-mean and FCM by implementing both in MapReduce and sequential approach. The algorithms implemented in MapReduce as follows: Mappers compute the distance and spills out a key-value pair <centroid_id, datapoint> for K-means. This identifies the associativity of the data point with the cluster. Reducers, on the other hand, work with specific cluster_id and a list of data points associated with it. New mean is computed with the help of a reducer and the reducer also writes to a new centroid file. Now it is the choice of the user regarding the number of iterations to be used, a method of algorithm termination or comparison with the centroid in the previous iteration. A single iterative method of K-means on MapReduce used in this work is shown in Fig. 6. However, this work requires many MapReduce jobs for clustering requiring much of its computational requirement [94].

The above-stated drawback of parallel FCM using MapReduce is solved by Ghadiri et al. (2016) [94], by an enhancement of classical FCM named BigFCM. The associativity of a data object with a centroid is specified by a weight. The mapper reads randomly selected cluster centers from the dataset and transfers the preprocessed records to another MapReduce process: combiner. In the case of more than one combiner, a key defines them. Combiner executes the FCM by fetching the centroids from the cache and data objects from the mapper. Big FCM combiner provides weights to the objects for each centroid and fed the value to the reducer. The reducers recalculate and discover new centroids. The reducer outputs are gathered by a single reducer which combines the result and discovers the final centroids. The experimental results show that the algorithm is way faster in terms of computational time and generates high-quality clusters. The overall process of this algorithm is shown below in Fig. 7.

In Table 3, we have provided details of some of the recent works of partition-based clustering using MapReduce.

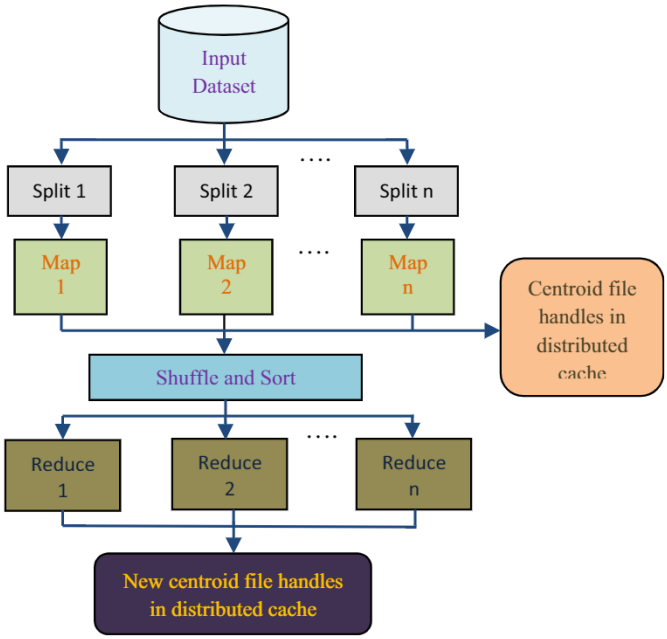

Fig. 6. Single iteration of K-means on MapReduce as in Ref. [93].

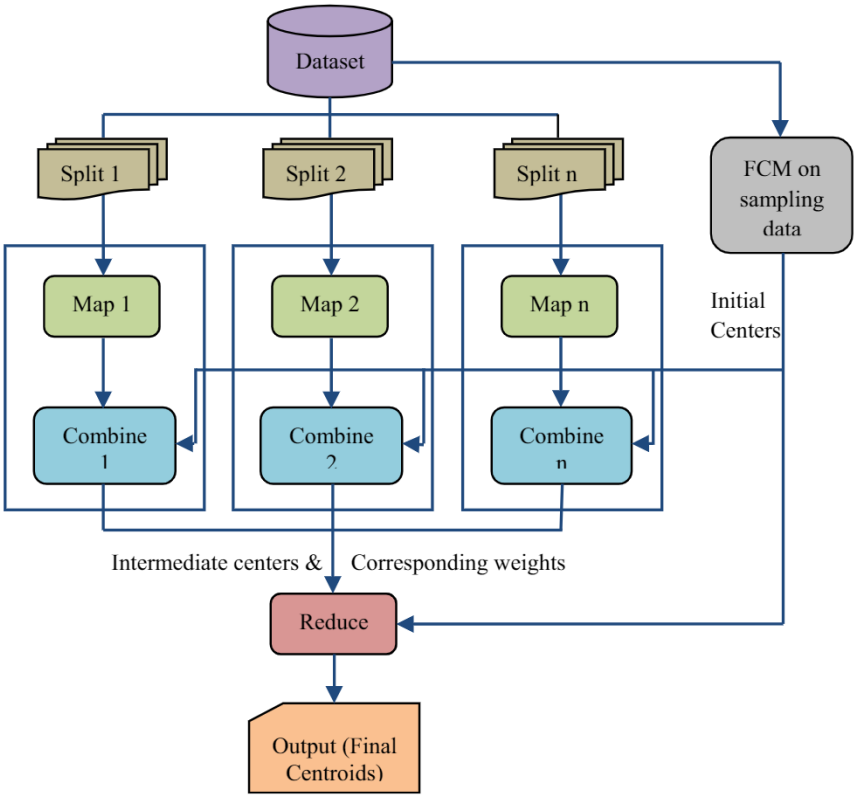

Fig. 7. The overall process of BigFCM as in Ref. [94].

4. Conclusion

This survey paper provides a comprehensive study of the recent works on partition based clustering algorithms modified using the MapReduce programming paradigm in order to execute on top of the Hadoop distributed computing environment. We have explained the MapReduce framework by elaborating MapReduce design and program execution Strategy. We have also reviewed and briefed different frameworks which use MapReduce paradigm. The most popular framework for MapReduce paradigm is Hadoop. Hence we have provided a Hadoop distributed computing framework into greater details. The recent tasks in the literature which implements different tasks using MapReduce to obtain different objective for different datasets are briefed in a table. Then we analyzed recent works on partition based clustering using MapReduce framework. The authors have modified MapReduce based clustering by incorporating different techniques in the MapReduce paradigm and clustering algorithms. Table 1 provides an accumulated discussion on different works in MapReduce which is used for different task parallelization. Similarly, Table 2 provides an overview on different partition based clustering algorithms used in different datasets. Table 2 also summarizes the contribution of these partition based clustering algorithms, specially, in terms of the working of the algorithmic parts (how the traditional algorithms are divided into mappers, combiners and reducers) over MapReduce paradigm.

In general, the empirical study allows us to draw the following conclusions for clustering using MapReduce:

- Hadoop cluster nodes made with commodity hardware are used to execute clustering algorithms modified in MapReduce.
- Introducing a combiner between the mapper and reducer functions decreases the amount of data to be written by the mapper and the amount of data to be read by the reducer. This technique considerably reduces the redundant MapReduce data transfer that results in a significant reduction in the time required for clustering.
- Hadoop operates only on text data and when an image dataset is processed and represented as text, the overhead in the representation and processing is huge.
- Clustering in MapReduce has no obvious advantage when dealing with smaller dataset, but it greatly shortens the computing time and archives good speedup ratio when dealing with large-scale data.
- In order to decrease the cost of network communication, a combiner function can be developed to deal with a partial combination of the intermediate values with the same key within the same map task.
- The clustering performance of MapReduce can be evaluated using two key observations: scale up and size up. Scale up is keeping the same data size and experimenting with different Hadoop cluster size. Size up is experimenting with different data size by keeping the same Hadoop cluster size.
- The input dataset distribution among the nodes has a profound effect in gaining efficiency in MapReduce based algorithms.

Table 3
Few Recent Works of Partition-based Clustering using MapReduce.

Algorithm Modified	Short Description	Input Dataset	Reference
K-means	Input images are transformed into text files. One MapReduce job distributes pixels into a centroid and second job moves pixels to minimize the error variances.	Remote sensing images	[75]
K-means	Mapper allots data objects to centroids with minimum distance centroid while reducer recalculates the value of the centroids when all objects have been assigned.	Dataset consists of different types of images	[52]
K-means	Mapper calculates the distance between vector points and centroids. Reducer computes the new cluster centers.	Document dataset	[33]
K-means	A combiner function is created that receives output from each mapper then calculates a local centroid value. The reducer calculates global centroid based on combiner output.	50,000 data points	[80]
K-prototypes	Mapper performs the pruning process. It also assigns each data point to the nearest cluster. Reducer is devoted to update the new cluster centers. The process of calling the two functions iterate several times until convergence.	Synthetic dataset, real data set which consists of data about TCP connections	[85]
K-medoids	A global variant medoids array contains the information about medoids of the clusters. Mapper computes the closest medoid for each data objects. A combiner combines the intermediate results of the map task. Reducer sums all the data objects and then computes the total quantity of data objects allocated to the same cluster. Thereafter, new medians are computed and used for next iteration.	Not provided	[87]
K-modes	The mapper calculates the similarity between the modes and the objects and allocate to a cluster whose similarity is maximum. Many mapper and reducers exchange data clustering operation to obtain a final result by a reducer.	US Census dataset	[90]
FCM	Mapper calculates the distance between cluster centers and the data objects from canopy generated dataset where centroid points received through canopy method is taken as key and data objects as value. Then Mapper computes the membership matrix for each object. The centroid recalculation for each cluster is achieved by a reducer function. The new centroid value is passed to Map. Iteratively it executes till convergence.	Iris, Synthetic control and KDD cup dataset	[91]
FCM	Mappers compute the distance and spill out a key-value pair <centroid_id, datapoint> for K-means. This identifies the associativity of the data point with the cluster. Reducers, on the other hand, work with specific cluster_id and a list of data points associated with it. New mean is computed with the help of a reducer and the reducer also writes to a new centroid file.	Not Mentioned	[93]
FCM	The mapper reads randomly selected cluster centers from a data cache and transfers the preprocessed records to the combiner. Combiner executes the FCM by fetching the centroids from the cache and data objects from the mapper. Combiner provides weights to the objects for each centroid and fed the value to the reducer. The reducers recalculate and discover new centroids.	Different datasets like Pima Indian Diabetes, intrusion detection and Iris datasets	[94]

- It is important to optimize the MapReduce configuration parameters for efficient clustering.
- The shuffling process of MapReduce consumes a significant processing time of the clustering job.
- The quality of clustering is highly dependent on the proper accumulation of partial clustering results from slave nodes to the master node.
- The distance calculation part of partition based algorithms is performed in map functions and the centroid recalculation part is calculated by reducers.
- The fault tolerance in MapReduce applications has a trade-off with the performance gain. The more the replication factor data objects possess, the larger it consumes time for the node to node data transfer.
- The MapReduce clustering algorithms are data intensive than computation intensive and a good performance gain in clustering requires minimization of disk I/O and reduction in communication among nodes.

This review provides us the following directions for implementation of clustering large datasets using MapReduce programming model:

[1] Hadoop cluster of general purpose computers can be used for the efficient parallel clustering of large datasets using MapReduce model.
[2] It is required to convert traditional partition based clustering algorithms to MapReduce model by expressing them into Map and Reduce functions. It is a challenging task to determine iterative parts and serial parts of clustering algorithms to express it in mapper and reducer respectively.
[3] The MapReduce model can be enhanced in order to obtain higher effectiveness in clustering. For an example, the data distribution among nodes become uneven (a load balancing problem) for the skewed datasets, where few keys are allocated with a large portion of data.
[4] In order to make MapReduce based clustering more effective, several optimizations in system level and code level can be performed.
[5] The system level optimization of the MapReduce model includes specification of data replication number and uniform transfer of the dataset among nodes etc.
[6] The code level optimization of the MapReduce model includes the design of a combiner function between mapper and reducer, incorporation of different techniques in the clustering algorithms for a good quality clustering (like pruning, particle swarm optimization etc.).
[7] It is required to implement an effective technique in the MapReduce model to accumulate good quality clusters from partial clustering results obtained from slave nodes.
[8] It is required to design MapReduce based clustering algorithms in such a way that it can tune the fault tolerance of the system by controlling the replication factor of the dataset. This adjustment technique on data replication factor should be focused on Hadoop cluster size, dataset feature, and objective of the clustering.
[9] In order to exploit efficiency in MapReduce based clustering, the focus should be given to reduce the number of MapReduce processes involved in clustering.
[10] A general core pseudo-code structure can be designed for parallelization of partition based clustering algorithms using MapReduce. This pseudo-code can be extended so as to parallelize different partition based clustering algorithms using MapReduce with different techniques like preprocessing, pruning, the addition of a combiner, particle swarm optimization etc.
[11] In order to experiment and observe the efficiency of partition based clustering algorithms using MapReduce, a relationship between data size and the number of Hadoop cluster nodes should be studied.
[12] Similarly, a relationship between speedup, data size, and code scalability is also required to be studied for observing the efficiency of partition based clustering algorithms.

Conflicts of interest

The authors declare that they have no conflict of interest.

Ethical approval

This article does not contain any studies with human participants or animals performed by any of the authors.

Acknowledgement

This study was facilitated by Government of Karnataka (GoK) Vision Group on Science and Technology (VGST) CISEE (2015−16) scheme grant GRD No-461.

References

[1] Nikhil R, Tikoo N, Kurle S, Pisupati HS, Prasad GR. A survey on text mining and sentiment analysis for unstructured web data. Int J Emerg Technol Innov Res 2015;1292:2−4.
[2] Cristobal Romero, Sebastian Ventura. Educational data mining: a survey from 1995 to 2005. Expert Syst Appl 2007;33(1):135.
[3] Zhang Bin, Forman George. . Distributed data clustering can Be efficient and exact. ACM SIGKDD Explor Newsl 2000;2(2):34.
[4] Baser, Saini. A comparative analysis of various clustering techniques used for very large datasets. Int J Comput Sci Commun Netw 2013;3(4):271.
[5] Hruschka ER, Campello RJ, Freitas AA. A survey of evolutionary algorithms for clustering. IEEE Trans Syst Man Cybern 2009;39(2):133.
[6] Alsabti K, Ranka S, Singh V. An efficient K-means clustering algorithm. Electr Eng Comput Sci Paper 1997;43.
[7] Maitrey S, Jha CK, Gupta R, Singh J. Enhancement of CURE clustering technique in data mining. In: National conference on development of reliable information systems. Techniques and related issues (DRISTI); 2012. p. 7−12.
[8] Liao SH, Chu PH, Hsiao PY. Data mining techniques and applications-a decade review from 2000 to 2011. Elsevier Expert Syst Appl 2012; 39(12):11303.
[9] Andreopoulos B, An A, Wang X, Schroeder M. A roadmap of clustering algorithms: finding a match for a biomedical application. Bioinformatics 2009;10(13).
[10] Tanvir Habib Sardar and Zahid Ansari. Detection and confirmation of web robot requests for cleaning the voluminous web log data. In: IEEE

international conference impact of E-Technology on US (IMPETUS); 2014. p. 13–9.
[11] Chitade AZ, Katiyar SK. Colour based image segmentation using K-means clustering. Int J Eng Sci Technol 2010;2(10):5319.
[12] Sharma, Ramya. A review on density based clustering algorithms for very large datasets. Int J Emerg Technol Adv Eng 2013;3(12):398.
[13] Namrata Gupta, Bijendra Agrawal, Rajkumar Chauhan. Survey on clustering techniques of data mining. Am Int J Res Sci Technol Eng Math 2015;5(2):2272–6.
[14] Fahad A, Alshatri N, Tari Z, Alamri A, Khalil I, Zomaya AY, Foufou S, Bouras A. A survey of clustering algorithms for big data: taxonomy and empirical analysis. IEEE Trans Emerg Topics Comput 2014;2(3):267.
[15] Dharmarajan A, Velmurugan T. Applications of partition based clustering algorithms: a survey. In: IEEE international conference on computational intelligence and computing research (ICCIC); 2013.
[16] Bora, Gupta. Comparative study between fuzzy clustering algorithm and hard clustering algorithm. Int J Comput Trends Technol 2014;10(12): 108.
[17] Prabha, Sujatha. Reduction of big data sets using fuzzy clustering. Int J Adv Res Comput Eng Technol 2014;3(6):2235.
[18] Jain AK, Murty MN, Flynn PJ. Data clustering:a review. ACM Comput Surv 1999;31:264.
[19] Falasconi M, Gutierrez A, Pardo M, Sberveglieri G, Marco S. A stability based validity method for fuzzy clustering. Elsevier Pattern Recognit 2010;43(44):1292.
[20] Hoppner Frank. Booked named fuzzy cluster analysis: methods for classification. Data Analysis and image recognition. 1999.
[21] Jafar, Sivakumar. A comparative study of hard and fuzzy data clustering algorithms with cluster validity indices. In: Proceeding of international conference on emerging research in computing. Information. Communication and applications (ERCICA); 2013.
[22] Zhanquan, Fox. A parallel clustering method study based on MapReduce. In: 1st international workshop on cloud computing and information security; 2013.
[23] Fox, Qiu. Case studies in data intensive computing: large scale DNA sequence analysis. The million sequence challenge and biomedical computing technical report. 2009.
[24] Qiu XH, Ekanayake J, Fox GC, Gunarathne T, Beason S. Computational methods for large scale DNA data analysis. In: Microsoft eScience workshop; 2009.
[25] Blake, Bult. Beyond the data deluge: data integration and bio-ontologies. J Biomed Inf 2006;39(3):314.
[26] Qiu J. Scalable programming and algorithms for data intensive life science applications. OMICS A J Integr Biol 2010;15(4):235.
[27] Guha R, Gilbert K, Fox G, Pierce M, Wild D, Yuan H. Advances in cheminformatics methodologies and infrastructure to support the data mining of large. Heterogeneous chemical datasets. Curr Comput Aided Drug Des 2010;6(1):50.
[28] Chang, et al. Mining semantics for large scale integration on the web: evidences. Insights and challenges. ACM SIGKDD Explor Newsl 2004; 6(2):67.
[29] Suryawanshi, Wadne. Big data mining using map reduce: a survey paper. IOSR J Comput Eng 2014;16(6):37.
[30] Saravanan, Maheswari. Analyzing large web log files in a hadoop distributed cluster environment. Int J Comput Technol Appl 2014;5(5): 1677.
[31] Agarwal, Zeba. Map reduce: a survey paper on recent expansion. Int J Adv Comput Sci Appl 2015:209–15.
[32] http://hadoop.Apache.org/.
[33] Zhou P, Lei J, Ye W. Large-scale data sets clustering based on MapReduce and hadoop. J Comput Inf Syst 2011;7(16):5956.
[34] Ghemawat S, Gobioff H, Leung ST. The Google file system. 19th symposium on operating systems principles. 2003. p. 23–43. New York.
[35] Dean, Ghemawat. Mapreduce: simplified data processing on large clusters. Commun ACM 2008;51(1):107.
[36] McNabb AW, Monson CK, Seppi KD. Parallel PSO using MapReduce. In: IEEE congress on evolutionary computation; 2007. p. 7–14.
[37] Jiang W, Ravi VT, Agrawal G. A map-reduce system with an alternate API for multi-core environments. In: Proceedings of the 2010 10th IEEE/ACM international conference on cluster. Cloud and grid computing; 2010. p. 84–93.
[38] Kang SJ, Lee SY, Lee KM. Performance comparison of OpenMP, MPI, and MapReduce in practical problems. Adv Multimed 2015; 575687.
[39] Ekanayake J, Pallickara S, Fox G. MapReduce for data intensive scientific analyses. In: IEEE fourth international conference on eScience; 2008. p. 277–84.
[40] Isard M, Budiu M, Yu Y, Birrell A, Fetterly D. Dryad: distributed data-parallel programs from sequential building blocks. ACM SIGOPS - Oper Syst Rev 2007;59:41–3.
[41] Olston C, Reed B, Srivastava U, Kumar R, Tomkins A. Pig Latin: a not-so-foreign language for data processing. In: Proceedings of the 2008 ACM SIGMOD international conference on management of data; 2008. p. 1099–110.
[42] Pike R, Dorward S, Griesemer R, Quinlan S. Interpreting the data: parallel analysis with sawzall. Sci Program 2005;13(14):277.
[43] Seo S, Jang I, Woo K, Kim I, Kim JS, Maeng S. HPRM: prefetching and pre-shuffling in shared MapReduce computation environment. In: IEEE international conference on cluster computing and workshops; 2009. p. 1–8.
[44] Thusoo A, Sarma JS, Jain N, Shao Z, Chakka P, Anthony S, et al. Hive: a warehousing Solution over A Map-reduce framework. Proc VLDB Endow 2009;2(2):1626–9.
[45] Zaharia M, Konwinski A, Joseph AD, Katz RH, Stoica I. Improving MapReduce performance in heterogeneous environments. In: Book named OSDI. 8(4); 2008. p. 7.
[46] Gillick D, Faria A, DeNero J. MapReduce: distributed computing for machine learning. Berkley 2006;18.
[47] Ranger C, Raghuraman R, Penmetsa A, Bradski G, Kozyrakis C. Evaluating MapReduce for multi-core and multiprocessor systems. In: IEEE 13th international symposium on high performance computer architecture; 2007. p. 13–24.
[48] He B, Fang W, Luo Q, Govindaraju NK, Wang T. Mars: a MapReduce framework on graphics processors. In: Proceedings of the 17th international conference on parallel architectures and compilation techniques; 2008. p. 260–9.
[49] Farivar R, Verma A, Chan EM, Campbell RH. Mithra: multiple data independent tasks on a heterogeneous resource architecture. In: IEEE international conference on cluster computing and workshops.; 2009. p. 1–10.
[50] Chu CT, Kim SK, Lin YA, Yu Y, Bradski G, Olukotun K, et al. Map-reduce for machine learning on multicore. Adv Neural Inf Process Syst 2007;19:281.
[51] Kumar A, Kiran M, Prathap BR, et al. Verification and validation of MapReduce program model for parallel K-means algorithm on Hadoop cluster. In: Fourth international conference on computing communications and networking technologies (ICCCNT); 2013.
[52] Li HG, Wu GQ, Hu XG, Zhang J, Li L, Wu X. K-means clustering with bagging and MapReduce. In: 44th Hawaii international conference on system sciences (HICSS); 2011. p. 1–8.
[53] Vijayalakshmi V, Akila A, Nagadivya S. The survey on MapReduce. Int J Eng Sci Technol 2012;4(7):3335.
[54] Vaidya Madhavi. Survey of parallel data processing in context with MapReduce. Comput Sci Inf Technol 2011;3:69.
[55] Haggarty OJ, Knottenbelt WJ, Bradley JT. Distributed response time analysis of gspn models with MapReduce. Simulation 2009;85(88): 497.
[56] Reshma Chaudhari, Naykar Savita Dilip, Vandhan Himali Jaywant, Shelke Pratibha Ashok, Kavita S Kumavat. A novel approach for generic log analyzer. Int J Recent Innov Trends Comput Commun 2015;3(10):5748–51.
[57] Yadav R, Rathod J, Nair V. Big data meets small sensors in precision agriculture. Int J Comput Appl 2015:0975–8887.
[58] Savitha, Vijaya. An efficient analysis of web server log files for session identification using hadoop MapReduce. In: Int. conf. on advances in communication network and computing. CNC; 2014.

[59] Roopam Dad, Nikhil Shanmugam, Prapti Singh, S.M Nalawade. Analysis of agriculture commodity prices using MapReduce model. Int J Adv Res Comput Commun Eng 2015;4(4):69.
[60] Markonis D, Schaer R, Eggel I, Müller H, Depeursinge A. Using MapReduce for large-scale medical image analysis. In: Proceedings of the 2012 IEEE second international conference on healthcare informatics. Imaging and Systems Biology; 2012.
[61] Alexandrov A, Ewen S, Heimel M, Hueske F, Kao O, Markl V, et al. Mapreduce and pact - comparing data parallel programming models. BTW 2011:25−44.
[62] Xia Tian. Large-scale SMS messages mining based on map-reduce. In: International symposium on computational intelligence and design1; 2008. p. 7.
[63] Feldman Jon. Using many machines to handle an enormous error-correcting code. In: Information theory workshop. ITW'06 Punta del Este. IEEE; 2006. p. 180−2.
[64] McNabb Andrew W, Monson Christopher K, Seppi Kevin D. Parallel PSO using MapReduce. In: IEEE congress on evolutionary computation; 2007.
[65] Zhang S, Han J, Liu Z, Wang K, Feng S. Spatial queries evaluation with MapReduce. In: Eighth international conference on grid and cooperative computing; 2009. p. 287−92.
[66] Kumar R, Singh MP, Kumar P, Singh JP. Crop selection method to maximize crop yield rate using machine learning technique. In: International conference on smart technologies and management for computing. Communication controls energy and materials (ICSTM); 2015. p. 138−45.
[67] Zhang L, Jiang J, Liu Z. Research on the AID measurement based on improved SVM algorithm. In: Second International symposium on electronic commerce and security1; 2009. p. 519.
[68] Jai-Andaloussi S, Elabdouli A, Chaffai A, Madrane N, Sekkaki A. Medical content based image retrieval by using the hadoop framework. In: 20th international conference on telecommunications (ICT); 2013. p. 1−5.
[69] Ioannou ZM, Nodarakis N, Sioutas S, Tsakalidis A, Tzimas G. Mining biological data on the cloud−a MapReduce approach. In: IFIP international conference on artificial intelligence applications and innovations; 2014. p. 96−105.
[70] Shvachko K, Kuang H, Radia S, Chansler R. The Hadoop distributed file system. In: IEEE 26th symposium on mass storage systems and technologies (MSST); 2010. p. 1−10.
[71] Lin J, Dyer C. Data-intensive text processing with mapreduce. Synthesis Lectures on Human Language Technologies 2010;3(1):1.
[72] Kalola, Bhave. Weblog analysis with map-reduce and performance comparison of single v/s multinode hadoop cluster. Int J Recent Innov Trends Comput Commun 2014;3692:2−11.
[73] http://www.gridgain.com/.
[74] Yang HC, Dasdan A, Hsiao RL, Parker DS. Map-reduce-merge: simplified relational data processing on large clusters. In: International conference on management of data proceedings of the 2007 ACM SIGMOD; 2007. p. 1029−40.
[75] Surve, Paddune. A survey on hadoop assisted K-means clustering of Hefty volume images. Int J Comput Sci Eng 2014;6(3):113.
[76] Zhenhua Lv. Parallel K-means clustering of remote sensing images based on MapReduce. Web information systems and mining. Springer; 2010. p. 162−70.
[77] Uday Kumar Sr, Chandavarkar Naveen D. Implementation of K-Means clustering algorithm in hadoop framework. Int J Res Appl Sci Eng Technol (IJRASET) 2015;3(5):829−33.
[78] Mishra, Badhe. Improved map reduce K mean clustering algorithm for Hadoop architecture. Int J Eng Comput Sci 2016;17144:2−7.
[79] Patil, Vaidya. K-means clustering with MapReduce technique. Int J Adv Res Comput Commun Eng 2015:4−11.
[80] Anchalia Prajesh. Improved MapReduce K-means clustering algorithm with combiner. In: 16th International conference on computer modelling and simulation (UKSim); 2014. p. 386−91.
[81] Gowru, Potnuri. Parallel two phase K-means based on MapReduce. Int J Adv Res Comput Sci Manag Stud 2015;22:3−11.
[82] Li ZH, Song XD, Zhu WH, Chen YX. K-means clustering optimization algorithm based on MapReduce. In: International symposium on computers & informatics (ISCI 2015); 2015. p. 198−203.
[83] Anchalia PP, Koundinya AK, Srinath NK. MapReduce design of K-means clustering algorithm. In: International conference on information science and applications (ICISA); 2013. p. 1−5.
[84] Zhao W, Ma H, He Q. Parallel K-means clustering based on MapReduce. Cloud computing. Springer; 2009. p. 674−9.
[85] HajKacem MA, N'cir CE, Essoussi N. An accelerated MapReduce-based K-prototype for big data. In: Federation of international conferences on software technologies: applications and foundations; 2016. p. 13−25.
[86] Drake Hamerly. Accelerating Lloyds algorithm for K-means clustering. In: Partitional clustering algorithms; 2015. p. 41−78.
[87] Srinivasulu L, Reddy AV, Akula VG. Improving the scalability and efficiency of K-medoids by map reduce. Int J Eng Appl Sci (IJEAS) 2015; 88:2−4.
[88] Jiang, Zhang. Parallel K-medoids clustering algorithm based on hadoop. In: 5th IEEE International conference on software engineering and service science (ICSESS); 2014. p. 649−52.
[89] Yue X, Man W, Yue J, Liu G. Parallel K-medoids++ spatial clustering algorithm based on MapReduce. Cornell University Library; 2016. https://arxiv.org/ftp/arxiv/papers/1608/1608.06861.pdf.
[90] Tao G, Xiangwu D, Yefeng L. Parallel K-modes algorithm based on MapReduce. In: Third international conference on digital information networking and wireless communications; 2015. p. 176−9.
[91] Garg D, Gohil P, Trivedi K. Modified fuzzy K-mean clustering using MapReduce in hadoop and cloud. In: IEEE international conference on electrical computer and communication technologies; 2015. p. 1−5.
[92] Mathew, Chandran. Parallel implementation of fuzzy clustering algorithm based on MapReduce computing model of hadoop −a detailed survey. Int J Comput Sci Inf Technol 2015;6(5):4740.
[93] Garg D, Trivedi K, Panchal B. A comparative study of clustering algorithms using MapReduce in hadoop. Int J Eng Res Technol 2013;6(10): 2999.
[94] Ghadiri N, Ghaffari M, Nikbakht MA. BigFCM: Fast. Precise and scalable FCM on Hadoop. Future Generat Comput Syst 2017;77:29−39.