Hindawi
Scientific Programming
Volume 2020, Article ID 8884926, 12 pages
https://doi.org/10.1155/2020/8884926

Review Article

A Survey of Parallel Clustering Algorithms Based on Spark

Wen Xiao[1] **and Juan Hu**[2]

[1]*Key Laboratory of Unmanned Aerial Vehicle Development & Data Application of Anhui Higher Education Institutes, Wanjiang University of Technology, Maanshan 243000, China*
[2]*Maanshan Engineering Technology Research Center for Wireless Sensor Network and IntelliSense, Wanjiang University of Technology, Maanshan 243000, China*

Correspondence should be addressed to Wen Xiao; cyees@163.com

Received 16 May 2020; Revised 30 July 2020; Accepted 4 August 2020; Published 1 September 2020

Academic Editor: Antonio J. Peña

Clustering is one of the most important unsupervised machine learning tasks, which is widely used in information retrieval, social network analysis, image processing, and other fields. With the explosive growth of data, the classical clustering algorithms cannot meet the requirements of clustering for big data. Spark is one of the most popular parallel processing platforms for big data, and many researchers have proposed many parallel clustering algorithms based on Spark. In this paper, the existing parallel clustering algorithms based on Spark are classified and summarized, the parallel design framework of each kind of algorithms is discussed, and after comparing different kinds of algorithms, the direction of the future research is discussed.

1. Introduction

Clustering is one of the most important unsupervised machine learning tasks. Its purpose is to divide data points into groups or clusters so that data points in the same cluster are similar to each other but are very different from data points in other clusters. Clustering is widely used in text mining, information retrieval, social network analysis, image and video analysis, and other fields. In the past decades, researchers have put forward many clustering algorithms, such as K-Means [1], K-Medoids [2], DBSCAN [3], BIRCH [4], OPTIGRID [5], FCM [6], PCM [7], CURE [8], CHAMELEON [9], DENCLUE [10], OPTICS [11], WaveCluster [12], STING [13], CLIQUE [14], FADE [15], CLARA [16], CLARANS [17], and ORCLUS [18], which have achieved good results over small-scale set of data points. Some clustering algorithms such as Possibilistic Fuzzy C-Mean realize the fuzzy segmentation of data points based on probability, which are applied to image segmentation and other fields [19–22].

With the rapid development of information technology such as sensors, computers, and communication, the data generated by people and various devices is growing explosively, and we have entered the era of big data. Big data can be defined and described generally with 5V [23] which is Volume, Variety, Value, Velocity, and Veracity. Owing to the above characteristics, the traditional clustering algorithms cannot meet the needs of big data clustering, so a parallel clustering algorithm is needed to meet the challenges.

Apache Spark is a new generation of parallel process platform for big data, which has merits such as easy to use, versatile, and automatic fault-tolerant. Specifically, Spark uses Resilient Distributed Datasets (RDDs) to store data in memory, which can significantly improve the performance of machine learning tasks requiring multiple iterations. Compared with the classic big data parallel process platform named Hadoop, Spark has an order of magnitude advantage in performance [24] and is suitable for clustering that requires multiple iterations in particular. Many parallel clustering algorithms based on Spark have been proposed in recent researches, which significantly improve the efficiency and accuracy of big data clustering.

Some researchers have reviewed the clustering algorithms for big data. In [25], the characteristics of big data are discussed in detail, and the classification and clustering

algorithms based on MapReduce are summarized and discussed. The article [26] discusses the characteristics of different types of clustering algorithms and the main challenges in dealing with big data and makes a comparative analysis of the major clustering algorithms. The paper [27] discusses the application, opportunities, and challenges of big data and briefly describes the latest technologies used to process big data. The paper [28] classifies the existing nonparallel clustering algorithms and compares the accuracy, scalability, and performance of different types of algorithms in dealing with big data through experiments. Currently, although Spark is the most popular big data parallel processing platform, and there are more and more parallel clustering algorithms based on Spark, there is no specific overview and discussion of parallel clustering algorithms based on Spark.

This paper provides an overview of parallel clustering algorithms based on Spark using the research methods of literature survey and classification, classifies the parallel clustering algorithms based on Spark proposed in the literature, summarizes the parallel implementation framework of each type of algorithms, and compares different types of algorithms.

The main contributions of this paper are as follows:

(1) The proposed parallel clustering algorithms based on Spark are studied in taxonomy

(2) The design framework and characteristics of each kind of parallel clustering algorithms based on Spark are discussed

(3) Different kinds of algorithms are compared and discussed, and the prospect of future research direction is discussed

This paper is organized as follows. In Section 2, we summarize the classical clustering algorithms and introduces the preliminaries. In Section 3, the main parallel clustering algorithms based on Spark are classified, and the design framework and characteristics of each kind of algorithms are discussed. In Section 4, different kinds of algorithms are compared, and the prospect of future research is discussed.

2. Preliminaries

2.1. Problem Definition. Let x_i be a data point depicted by p attributes $(x_{i1}, x_{i2}, \ldots, x_{ip})$, and $N = \{x_1, x_2, \ldots, x_n\}$ is a collection of n data points. The set of clusters is expressed as $C = \{C_1, C_2, \ldots, C_m\}$, where $C_1 \cap C_2 \cap \ldots \cap C_m = \varnothing$. A cluster C_i can be represented by a special data point in this cluster or a statistical value of the data point in this cluster. This data point or statistical value is called the centroid of the cluster, which is expressed as c_i. The aim of clustering is to allocate the data points in N to m clusters. In the following, $\mathrm{dist}(x_i, x_j)$ is used to represent the distance (or similarity) between two data points; $\mathrm{dist}(C_i, C_j)$ is used to represent the distance (or similarity) between two clusters, and $\mathrm{dist}(x_i, C_i)$ is used to represent the distance (or similarity) between a data point and a cluster.

2.2. Classical Clustering Algorithm. Clustering is a traditional machine learning task. In the past decades, researchers have proposed many nonparallel clustering algorithms. Classical clustering algorithms can be divided into five categories.

2.2.1. Partitioning-Based. This is the most commonly used clustering algorithm, which divides data points into multiple mutually exclusive clusters. In the process of partition, each data point is usually divided into the nearest cluster according to the distance between the data point and the cluster. The most famous algorithm in this category is K-means [1], which randomly selects m data points as the initial centroid, divides the remaining data points into clusters closest to them, updates the centroid of the cluster with the average value of the data points in the cluster after the division, and iterates the above process until the division is stable or meets the iteration stop condition (generally the maximum number of iterations or the best approximation function value). This kind of algorithm has four key aspects: the selection of initial centroid, the measurement of the distance between data points and clusters, the update method of the centroid, and the design of approximation function. Other algorithms in this category are optimized in the above four aspects. For example, Intelligent Keams [29] uses abnormal data points as the initial centroid; K-Medoids [2] uses the center of cluster as the updated centroid; FCM [6] and PCM [7] use different approximation functions. The clustering algorithm based on a partition is easy to understand and realize, but it has several obvious disadvantages: firstly, the selection of initial centroid has a significant impact on the clustering speed and accuracy; secondly, the partition is based on the distance between the data point and centroid of the cluster, which is only suitable for finding spherical clusters; thirdly, the clustering quality is significantly affected by outliers; fourthly, it needs to specify the number of clusters in advance; sometimes, this parameter is difficult to determine priorly.

The partition-based clustering algorithms depend heavily on the selection of initial centroid. If the initialization centroid is not selected properly, it will have a very serious impact on the quality of clustering results. From another point of view, clustering can be seen as a process of continuously optimizing the selection of the centroids of clusters and searching the best centroid of the cluster by heuristic searching. Many heuristic search methods can be used for continuous optimization, including artificial bee colony [30] and particle swarm optimization (PSO) [31]. These methods simulate the intelligence of the colony or the process of natural evolution. Generally, they converge quickly and have a good effect.

2.2.2. Hierarchical-Based. This kind of algorithm organizes data points into a tree of clusters with a hierarchical structure, in which the leaf is the data point, the root node is all data points, and the other nodes in the tree represent a cluster. The parent nodes in the tree represent the cluster obtained by agglomerating clusters represented by the child

node, and the child node represents the cluster obtained by the devising cluster represented by the parent node. Therefore, the hierarchical-based clustering algorithm can be carried out in the top-down direction in a division way or in a bottom-up way in an agglomeration way. In the former way, all data points are regarded as a cluster firstly, and then the cluster is divided into several subclusters recursively. In the latter way, each data point is regarded as a cluster firstly, and then two or more clusters agglomerate recursively. The above iterative process ends when the iterative stop condition is reached (generally, the number of clusters is used), using the distance or similarity between clusters as the basis for cluster division or agglomeration. BIRCH [4], CURE [8], and CHAMELEON [9] are the representative algorithms of this kind. This kind of algorithm has an obvious disadvantage, that is, the operation of division or agglomeration is irreversible, and improper operation can lead to low-quality clusters.

2.2.3. Density-Based. The clustering algorithms based on partition and hierarchy can only find spherical clusters, and it is difficult to find clusters of arbitrary shapes. The density-based clustering algorithm regards cluster as a dense area separated by sparse area in data space. If a data point belongs to a cluster, it must be in a dense area; that is to say, it has more than one predefined threshold number of neighbors in a special radius. The discovery of clusters can start from a data point and extend in any direction according to the density. Therefore, clusters of any shape can be found, and the outliers are naturally filtered. DBSCAN [3] is a typical representative algorithm of this kind. It identifies the dense area and the core point by calculating the number of neighbors in the field of this point. Multiple dense areas can be connected to form a cluster according to density reachable between core points. Data points that cannot be included in any cluster (i.e., data points that are not in any dense area) can be identified as an outlier. These kinds of well-known algorithms include DENCLUE [10] and OPTICS [11].

2.2.4. Model-Based. This kind of clustering algorithm assumes that data points are generated according to a certain probability distribution model, and the clustering process is to adapt all data points to some predefined mathematical models. Therefore, this kind of algorithm can automatically identify the number of clusters and outliers in data points according to the selected mathematical model. The commonly used probability distribution models are the Gaussian mixture model (GMM) [32], mixture model for cluster analysis [33], etc., and the typical algorithms are EM [34], etc.

2.2.5. Grid-Based. The above discussed four kinds of clustering algorithms are all data-driven, which directly partition or identify data points, while the grid-based clustering algorithm is space-driven. This kind of algorithm divides the data space of data points into a fixed number or size of grid units, and clustering is carried out on grid units instead of data points. Because the number of grid units is far less than the number of data points and only needs to scan the data points in the grid once to get the statistical information of the unit, the grid-based clustering algorithm is faster, and the performance is independent of the number of data points. WaveCluster [12] and STING [13] are representative algorithms of this kind. Although the clustering algorithm based on the grid has faster speed, it needs to predefine the number or size of grid units. If the data points' distribution is irregular, the use of fixed number or uniform size of the grid unit to divide the data points may lead to poor quality of clusters and long clustering time and is not suitable for processing high-density data.

2.3. Spark. Apache Spark [35] is one of the most popular big data parallel processing platforms at present. Because it uses RDD based on memory to store input data and intermediate results, compared with Hadoop [36], it reduces a lot of I/O operations, especially those suitable for machine learning tasks which require multiple iterations [37].

A typical Spark cluster consists of one master node and several worker nodes. The master node is responsible for managing the resources of the worker nodes and assigning tasks to the worker nodes, while the worker nodes perform the corresponding distributed tasks. The working model of Spark is shown in Figure 1.

Spark distributes RDD to worker nodes in the cluster to realize distributed storage and provides a batch of functions for performing parallel operations on RDD which cross worker nodes. Commonly used functions are as follows:

map (f): use a user-defined function *f* to convert each record in RDD to a new record, and return an RDD containing the new record

mapPartitions (f): use a user-defined function *f* to convert each record in the local RDD partition into a new record, and return an RDD containing the new record

filter (f): use a user-defined function *f* to filter the records in RDD and return an RDD containing the filtering results

reduce (f): use a user-defined function *f* to aggregate data in RDD

reduceByKey (f): use a user-defined function *f* to aggregate data with the same key in pair RDD

takeSample (s): use a user-defined generator seed *s* to get a sample of RDD records

collect (): return the records in RDD to the master node as an array of objects

3. Taxonomy and Framework of Parallel Clustering Algorithm Based on Spark

The algorithms that have been proposed are basically to transplant the classical clustering algorithms to the Spark platform. They use RDD to store datasets in a distributed

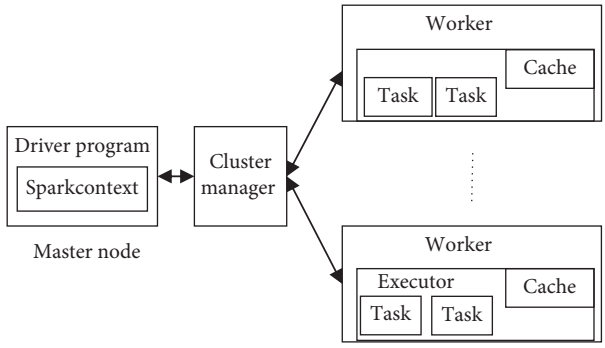

Figure 1: Working model of Spark framework.

way and use the functions provided by Spark to realize the parallel execution of key steps, so as to achieve parallel clustering. The proposed algorithms can be roughly divided into four categories.

3.1. Parallel K-Means Clustering Algorithm. K-means is the most famous and commonly used clustering algorithm. It has three key steps: the selection of initial centroid, distribution of every data point into the nearest cluster by calculating the distance between the data point and each cluster centroid, and updating of each cluster centroid. Many K-means variant algorithms take different strategies in these three key steps. The implementation framework of this kind of algorithm based on Spark is shown in Figure 2.

As can be seen from Figure 2, the three key steps of the K-means algorithm can be executed in parallel using the functions provided by Spark. After loading data points into RDD, this kind of algorithm can use takesample() function to randomly select initial centroid or use filter() function to customize filtering rules. The distance between every data point and all centroids can be calculated by map() or mapPartitions (). Each data point is divided into clusters represented by the nearest centroid. This is the result of the division store in pair RDD, where the key is the cluster ID and the value is the data point. Algorithms of this kind use reducebykey() function to update the centroid of clusters. If the iteration end condition has been met (generally the number of iterations or the change proportion of data point distribution), stop the iteration, and use collect() to collect the clustering results from each worker node. Otherwise, redistribute data points and update centroid.

The framework of parallel K-means algorithm in MLlib [38], a Spark-based machine learning algorithm library, is exactly the same as that in Figure 2. It randomly selects the initial centroids, uses Euclidean distance (1) to measure the distance between data points, and uses the average value of data points in the cluster to update the centroid of the cluster (2). This algorithm is easy to understand and use, but the Euclidean distance measurement will be invalid if the dimension of the data point is high, and the mean value of data points in the cluster is used to update the centroid, which

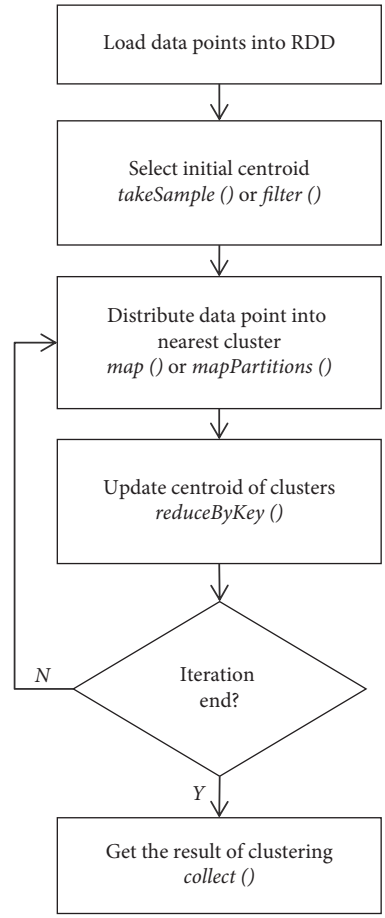

Figure 2: Parallel implementation framework of the K-means clustering algorithm based on Spark.

does not support fuzzy clustering. So, this algorithm is only suitable for the deterministic clustering of low dimensional data. The parallel K-means proposed in [39, 40] is similar to the parallel K-means in MLlib, which is based on the classic K-means algorithm implemented in Spark:

$$\operatorname{dist}(x_i, x_j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \cdots + (x_{ip} - x_{jp})^2}, \tag{1}$$

$$c_i = \left(\frac{1}{|C_i|} \sum_{j=1}^{|C_i|} x_{j1}, \frac{1}{|C_i|} \sum_{j=1}^{|C_i|} x_{j2}, \ldots, \frac{1}{|C_i|} \sum_{j=1}^{|C_i|} x_{jp} \right). \tag{2}$$

Wang et al. proposed a series of optimization strategies for parallel K-means algorithm in MLlib in [41]. This algorithm can use a simple and fast random selection method to select initial centroids or use the method proposed in [42] to select high-quality initial centroids according to probability distribution or use the distributed centroid selection method [43] to speed up the selection of initial centroids from big data. If the dimension of the data point is high, this algorithm uses cosine distance [44] (3) or KL distance [45] (4) to measure the distance between data points:

$$\operatorname{dist}\left(x_i, x_j\right) = \frac{\sum_{n=1}^{p} x_{in} * x_{jn}}{\sqrt{\sum_{n=1}^{p} x_{in} *} \sqrt{\sum_{n=1}^{p} x_{jn}}}, \tag{3}$$

$$\operatorname{dist}\left(x_i, x_j\right) = \sum_{n=1}^{p} x_{in} * \log \frac{x_{in}}{x_{jn}}. \tag{4}$$

The selection of initial centroids has a significant impact on algorithms of K-means kind. Improper selection of initial centroids may lead to long execution time or poor clustering quality. Therefore, many types of research have explored the optimization choice of initial centroids. In [46], a parallel intelligent K-means based on Spark is proposed. The difference between this algorithm and K-means is the way of a generation method of initial centroids. Intelligent K-means [29] looks for the center gravity or average of data points in advance and initializes the centroids which are farthest from the center gravity. This method of selecting the initial centroid takes into account the distribution of data points but is greatly affected by outliers. Similar algorithms include [47], which uses bat [48] and firefly [49] to optimize the selection of initial centroids.

Lu et al. proposed a parallel clustering algorithm, which uses a tabu search strategy [50, 51] to optimize the updating of centroids [52]. Given a cluster C_i and its centroid x_i, a domain $N(i)$ (5) with a radius R_i can be created. The new centroid can only be selected from $N(i)$. The value of R_i can be determined by the average (6) of the distance between data points in the cluster. If the centroid of a cluster is updated from x_i to x_j in an iteration, x_i will be added to the tabu list, and x_i will no longer be an alternative centroid in the specified t-round iteration. Although this method of centroid updating is more complex than the classical K-means, centroid updating of each cluster can be performed in parallel on Spark, so it can still generate new centroids of higher quality quickly, making each cluster closer, making the distance between all data points DistSum (7) as small as possible:

$$N(i) = \left\{ x_j \,\middle|\, \operatorname{dist}\left(x_j, x_i\right) \le R_i, x_j \in C_i, x_j \ne x_i \right\}, \tag{5}$$

$$R_i = \frac{\sum_{i=1}^{|C_i|} \operatorname{dist}\left(x_i, x_j\right)}{|C_i|}, \quad \left(x_i, x_j \in C_i, x_i \ne x_j\right), \tag{6}$$

$$DistSum = \sum_{1}^{m} \sum_{i}^{|C_i|} \operatorname{dist}\left(x_i, x_j\right). \tag{7}$$

Another way to update the centroid is to use a heuristic search method, which is also called evolutionary computation. Its characteristic is to simulate the method of biological evolution or natural selection which chooses the best solution among the new solutions generated randomly. The representative algorithms of this kind of methods are artificial bee colony (ABC) [30], Particle Swarm Optimization (PSO) [31], etc. Yan et al. proposed a parallel ABC algorithm based on Spark [53]. The process of clustering is a simulation of bees' search for high-quality food sources. ABC algorithm divides all data points into three categories: food source (centroid), employed bee (assigned data point), and the unemployed bee (unassigned data point). In each iteration, a random partition solution is generated first, and then the partition probability of each data point is calculated. By comparing with the previous partition solution, whether to update the solution is determined until the iteration stops. Similarly, the KMPSO [54] proposed by Matthew et al. implements a parallel PSO algorithm based on Spark.

Fuzzy C-means (FCM) is the most classical fuzzy clustering algorithm, which is proposed by Bezdek [6]. The clustering result of this algorithm is not to divide each data point into clusters but to generate the probability that each data point belongs to a cluster. The difference between FCM and K-means mainly lies in the method of centroid updating and the design of objective function. It uses equation (8) to update centroid and equation (9) as an objective function. The objective of iteration is to minimize the weighted average of all data points belonging to a cluster:

$$c_i = \frac{\sum_{i=1}^{|C_i|} w_{im}^{c} x_i}{\sum_{i=1}^{|C_i|} w_{im}^{c}}, \tag{8}$$

$$J_m = \sum_{i=1}^{n} \sum_{j=1}^{m} u_{ij}^{c} \left\| x_i - c_i \right\|^2. \tag{9}$$

Neha et al. discussed the design ideas of three famous variants of the FCM algorithm named LFCM [55], resFCM [55], RSIO-FCM [56], and proposed a parallel clustering algorithm SRSIO-FCM [57] based on Spark.SRSIO-FCM divides the set of data points N into multiple subsets randomly $\{N_1, N_2, \ldots, N_S\}$, clustering the first subset N_1 with the classical K-means algorithm to generate the set of centroids $\{c_{11}, c_{12}, \ldots, c_{1k}\}$. In the following iteration, when dealing with N_i, the centroids of $N_1 \sim N_{i-1}$ is calculated according to the membership to generate the recommended centroid set of N_i, $c = \{c_{i1}, c_{i2}, \ldots, c_{ik}\}$, which is used as the initial centroid to cluster N_i until all subsets are clustered.

Spark-PCM [7] proposed by Zhang et al. is a parallel version of the Possibilistic C-Means (PCM) [7] based on Spark. PCM uses the possibility model instead of the probability model of FCM, which is less affected by outliers. Because the possibility model matrix needs to be updated repeatedly, Spark-PCM uses the distributed matrix operation library Marlin [58] to accelerate the matrix update operation.

In order to improve the traditional distance measurement method which only supports numerical attributes (such as equations (1), (3), and (4)), Mohamed et al. [59] implemented the Spark-based K-Prototypes [60]. The difference between K-Prototypes and K-means is the measurement method of the distance between data points and centroid. It uses equation (10) to measure the distance between data points and centroid so that it can support data points with categorical attributes:

$$\operatorname{dist}\left(x_i, c_i\right) = \sum_{r=1}^{m_r} \sqrt{\left(x_{ir} - c_{ir}\right)^2} + \sum_{t=1}^{m_t} \delta\left(x_{it}, c_{it}\right). \tag{10}$$

3.2. Parallel Hierarchical Clustering Algorithm. The hierarchical-based clustering algorithm organizes all data points into a tree structure, which can agglomerate data points from the bottom-up direction or divide the set of data points from the top-down direction. Obviously, the method based on agglomeration is more suitable for parallel execution. Single linkage hierarch clustering (SHC) [61] is a representative clustering algorithm based on agglomeration. It needs multiple iterations. Each iteration will merge the nearest data points or clusters with each other until the end condition of iteration is met (generally the number of clusters or the number of iterations).

In order to realize the agglomeration of contiguous data points or clusters, the set of data points needs to be divided according to the data space in advance. The framework of the parallel hierarchical clustering algorithm based on Spark is shown in Figure 3.

Jin et al. proposed a parallel SHC algorithm based on Spark named SHAS [62]. The framework of SHAS is the same as Figure 3, which mainly includes three stages: data point division, local clustering, and cluster merging. In the local clustering stage, the method proposed in [63] is introduced to transform the clustering into a problem of finding a minimum spanning tree (MST) of a complete graph. All the vertices in the graph are data points, and the weight of the edge is the distance between data points. This method improves the efficiency of local clustering and cluster merging. Firstly, SHAS divides the set of data points into roughly equal partitions $\{N_1, N_2, \ldots, N_S\}$ and constructs a complete graph for every partition. Then, bipartite graphs (C_s^2 in total) are constructed for each pair of data partitions. After MSTs of each complete graph are constructed, each bipartite graph is combined until only one MST is left.

MLlib, a machine learning algorithm library based on Spark, also contains a parallel hierarchical clustering algorithm, which is based on the bisecting K-means [64], and the design idea comes from the paper [65]. Because bisecting K-means is a hierarchical clustering algorithm based on the division in the top-down direction, this algorithm can only start from a single node with all data points. It uses K-means to further divide clusters in each iteration. The parallelism degree of this algorithm is low, and it is only suitable for training data with a small size [66].

3.3. Parallel Density Clustering Algorithm. The clustering algorithm based on partition or hierarchy can only find spherical clusters, and the clustering quality is greatly affected by outliers, while the clustering algorithm based on density overcomes the above two shortcomings. Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [3] is one of the most representative density-based clustering algorithms, which can identify clusters of any shape and outliers efficiently. DBSCAN uses the number of data points denoted by $|N_{\text{Eps}}(p)|$ contained in $N_{\text{Eps}}(p)$ which is a neighborhood with radius r of data point p (Figure 4) as the density of data point p and divides all data points into three categories according to the density of data

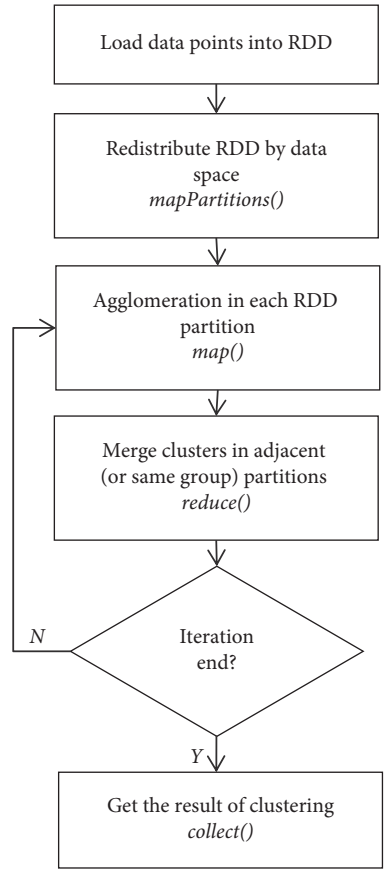

Figure 3: Parallel implementation framework of the hierarchical clustering algorithm based on Spark.

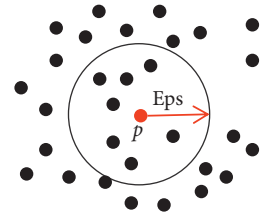

Figure 4: $N_{\text{Eps}}(p)$: the neighborhood of data point p with radius r.

points: core points, boundary points, and outliers. The core point is the data point whose density is greater than the specified threshold MinPts. The boundary point is the data point included in N_{Eps} of a core point but not the core point; the outlier is the data point not in N_{Eps} of any core point. N_{Eps} of all the density-connected core points can form a cluster (Figure 5). Data points that do not belong to any cluster are outliers. Therefore, the density-based clustering algorithm mainly consists of two key steps: finding the core points by calculating the density of data points and merging N_{Eps} (representing a subcluster) of the density-connected core points into a cluster:

$$\text{density}(P) = \left|N_{\text{Eps}}(p)\right|. \tag{11}$$

The framework of a parallel density clustering algorithm based on Spark is shown in Figure 6.

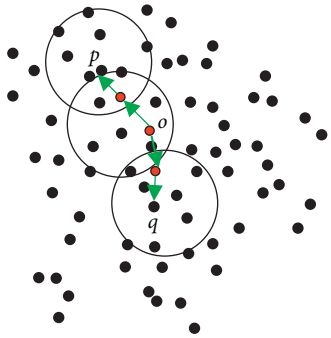

Figure 5: Core point p and q are density-connected.

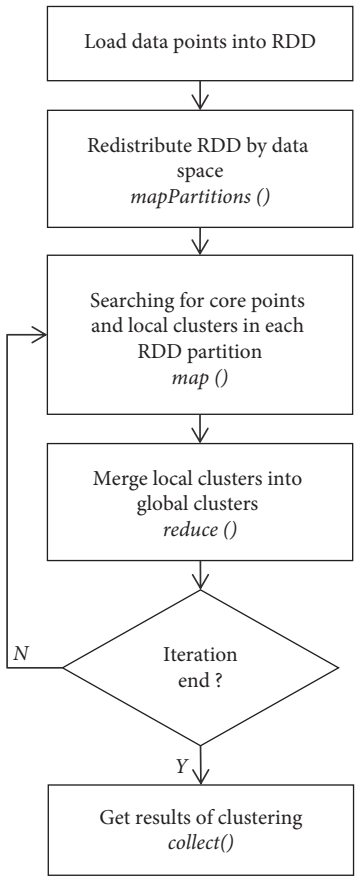

Figure 6: The framework of a parallel density clustering algorithm based on Spark.

Fang et al. proposed a parallel density clustering algorithm named parallel DBSCAN [67] based on Spark and classic DBSCAN. According to the analysis of DBSCAN, more than 90% of its execution time is used to calculate the density of data points and find the core points. Therefore, parallelizing this step can significantly improve the performance of DBSCAN. Parallel DBSCAN mainly consists of three stages: in the first stage, data points are redistributed according to the strategy of grid and secondary expansion partition [68] according to the data space. Each grid is called a local area, the adjacent parts between the local areas are called boundary areas, and the data points in the boundary area are called boundary points (Figure 7). Because the

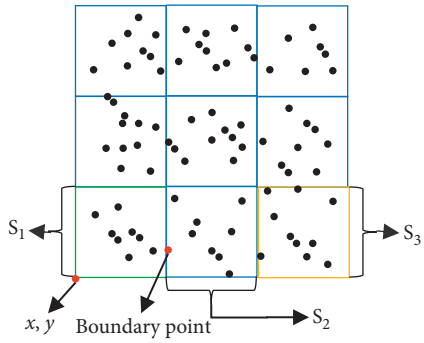

Figure 7: Data grid and boundary points generated after data points are divided by space.

boundary points may be density-connected with data points in multiple local areas, the algorithm enlarges the grid appropriately to include the boundary points (Figure 8).

In the second stage, the classical DBSCAN is executed in each cell in parallel to generate local clusters. Finally, in the third stage, all local clusters are merged and relabeled to generate global clusters. The authors in [67] also discuss the optimization of parallel DBSCAN in terms of data transmission, serialization, parameter configuration, etc. The experimental results show that parallel DBSCAN has good acceleration under various Spark operating environments than classic DBSCAN.

Amar et al. proposed a parallel clustering algorithm SparkSNN [69] based on Spark and Shared Nearest Neighbor (SNN) [70]. Different from the parallel DBSCAN which uses Euclidean distance (equation (1)) to measure the distance between data points, SparkSNN uses the number of data points in the neighborhood intersection of data points as the method for distance measurement (equation (12)), the density of x_i is defined as the sum of the similarities between x_i and data points in N_{Eps} (equation (13)), and the core point (also is the centroid of subcluster) is the point whose density is greater than the specified threshold MinPs. SparkSNN, like parallel DBSCAN, also includes three main stages: redistributing data points, local clustering, and global merging. The difference is that the local clustering stage uses SNN. Compared with the DBSCAN algorithm, the SNN algorithm not only has the advantages of discovering clusters of arbitrary shapes and recognizing outliers well but also has a better effect on the collection of data points with a high dimension or uneven density:

$$\text{dist}\left(x_i, x_j\right) = N_{\text{Eps}}\left(x_i\right) \cap N_{\text{Eps}}\left(x_j\right), \tag{12}$$

$$\text{dentisy}\left(x_i\right) = \sum \text{dist}\left(x_i, x_j\right) x_j \in N_{\text{Eps}}\left(x_i\right), \quad x_i \neq x_j. \tag{13}$$

Liu et al. proposed a parallel clustering algorithm Parallel DP [71] based on Spark Graphx and Density Peaks [72]. The main difference between this algorithm and the parallel DBSCAN algorithm is the selection of centroid of clusters. The parallel DBSCAN selects the first found core point as the centroid of the cluster, which can only ensure that the density of centroid is not less than the predefined threshold MinPts. The Parallel DP calculates the density of each data

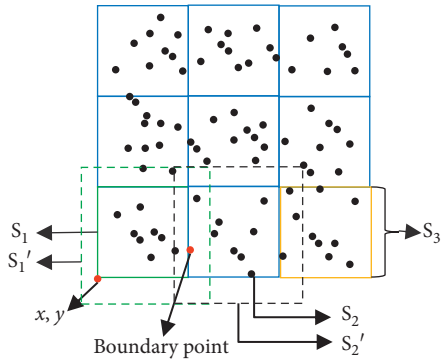

Figure 8: Expansion of the grid to include boundary points.

point and the minimum distance to other data points with higher density $\delta(x_i)$ (equation (14)) and selects the data point with the largest δ as the centroid of the cluster. In other words, the densest data point in the cluster is selected as the centroid of the cluster, which makes the cluster compact and more conducive to merging local clusters to generate global clusters:

$$\delta(x_i) = \min\left(\text{density}\left(x_j\right)\right) x_j \in N_{\text{Eps}}(x_i). \tag{14}$$

The framework of the parallel clustering algorithm [73] proposed by Liang et al. is consistent with Figure 6, but special methods are adopted in three stages. In the data point redistribution stage, the locality sensitive hashing (LSH) function [74] is used to achieve better load balancing and spatial density estimation; in the local clustering stage, the classic kernel density and high-density nearest neighbor (KNN) [75] are used, and the Gauss model is used to represent the local cluster to reduce the amount of data transmitted by the network and the amount of computation in the merging stage. In the stage of merging local clusters into global clusters, the density connection based on the Gauss model is used to merge multiple models (local clusters).

3.4. Parallel Model Clustering Algorithm. The model-based clustering algorithm is based on the assumption that the data points come from the data source which contains multiple subpopulations. The data points in each subpopulation conform to a certain probability distribution, and the data point set is a mixture of multiple subpopulations. The most commonly used model is Gaussian Mixture Models (GMMs) [32], which regards the cluster as a Gaussian distribution, and the set of data points is a mixture of multiple Gaussian distributions with different parameters (equation (15)). The process of clustering is to divide the data points into a Gaussian distribution, which directly generates clusters. Therefore, the model-based clustering algorithm has more advantages in running speed than other types of algorithms:

$$f(x) = \sum_{i=1}^{k} f\left(x \,\middle|\, u_{i,\Sigma_i}\right) P(C_i). \tag{15}$$

MLlib provides a parallel clustering algorithm based on GMM. It uses the expectation maximization (EM) [34] for sampling data points to find one or more variable Gaussian distributions and training Gaussian mixture model and generates the mean value and standard deviation of each Gaussian distribution closer to the real situation through multiple iterations. After training, the Gaussian mixture model is used to classify all data points and get a predefined number of clusters. In addition to the GMM model, MLlib also provides a parallel Latent Dirichlet Allocation (LDA) [76] algorithm, which is mainly used for text clustering. LDA is a probability model of the corpus. It is supposed that a text is randomly mixed by multiple latent topics; each latent topic can be identified by the probability distribution of words.

The above two parallel model clustering algorithms provided by MLlib both include three main stages: data sampling, training probability model, and classification. These algorithms firstly get some samples from all data points, then use the samples to train the probability model, and finally use the trained probability model to classify all data points in parallel.

3.5. Parallel Clustering Algorithm for Special Dataset. A high-dimensional dataset refers to the set of data points described by a large number of attributes. Data points with high dimensionality will bring serious challenges to the clustering algorithm based on distance measurement, which not only affects the performance but also has a significant impact on the accuracy of clustering. This is called "the curse of dimensionality" [77]. There are two main methods to cluster a high-dimensional dataset: one is subspace clustering, which refers to clustering a subset of all attributes rather than all attributes; the other is dimension reduction method, which combines or converts the original attributes to form new attributes, so as to reduce the dimension.

Zhu et al. proposed a parallel subspace clustering algorithm named CLUS [78] based on Spark, which parallelized the classic subspace clustering algorithm SUBCLU [79]. If the dimension of data points is p, there are $2^p - 1$ subspaces, and the subspaces with the same dimension can be clustered in parallel. CLUS starts from p subspaces with a single dimension and generates all clusters in these subspaces. In the k-th iteration, any two (k-1)-dimensional candidate clusters are merged, and the merged results are pruned monotonically to generate a k-dimensional subspace. The DBSCAN algorithm is used to cluster the newly generated k-dimensional subspace, which requires at most logP iterations. In the above process, each worker node needs to process all data points in the subspace. Some data points need to be copied repeatedly between multiple nodes. It is difficult to redistribute the dataset, and the amount of data transmission between nodes is also large.

Spectral clustering is a dimension reduction method. Zhu et al. put forward a parallel spectral clustering algorithm based on Spark, named SCoS [80]. This algorithm parallelized the four main steps of the spectral clustering algorithm: building similarity matrix, building Laplacian matrix and normalization, feature vector calculation, and parallel clustering of feature vector matrix. Some methods of performance optimization are used in the algorithm of SCoS:

Table 1: Comparison of key features of various algorithms.

	Clusters shape	Handling outliers	Input parameters	Data points' distribution
K-means	Spherality	No	m n	Random
Hierarchy	Spherality	No	m	Random or by data space
Density	Arbitrary	Yes	Eps MinPts	Data space
Model	Mathematical model	Yes	m $k_1, k_2, \ldots, k_i$	Random

m: number of clusters; n: maximum number of iterations; Eps: radius of data point neighborhood; MinPts: minimum density of core data point; $k_1, k_2, \ldots, k_i$: parameters required for a specific model.

parallel computation based on multiround iteration is adopted to improve the speed of building similarity matrix; Sparse representation and storage are adopted to further reduce the data transmission between storage and nodes; ScalaPack, a numerical linear algebra computing library, is used to accelerate the parallel solution of feature vectors.

In addition to the high-dimensional data, there are also parallel clustering methods [81, 82] for the streaming data realized by Spark streaming and parallel clustering methods [83–86] for the graph data realized by Spark graph x, which expands the types of the dataset that can be parallel clustered based on Spark.

4. Conclusion and Prospection

4.1. Summary of Proposed Algorithms. To sum up, many researchers have proposed many parallel clustering algorithms based on Spark, and their main common features are as follows:

(1) These parallel clustering algorithms are all based on the classical clustering algorithm; there is no significant change in the algorithm design. They all improve the clustering efficiency by parallelizing the main steps of the classical algorithm.

(2) These parallel clustering algorithms use RDD provided by Spark to store data points, but most of them need to distribute data points according to the characteristics of data points or algorithms.

(3) These algorithms use the functions provided by Spark to realize the parallel operation of each data partition and make full use of the characteristics of Spark based on memory computing to improve the efficiency of multiple iterations.

The comparison of key features of various algorithms is shown in Table 1.

4.2. Prospection of Future Works. Big data can be defined and described generally with 5V [23] which are Volume, Variety, Value, Velocity, and Veracity. From the above discussion, we can see that the proposed parallel clustering algorithms based on Spark mainly solve the problem of the large scale of data. In addition to the huge Volume of data, big data also includes the following features.

4.2.1. Variety. The data types are very diverse. In addition to the structured two-dimensional quantitative data which is most used, it also includes categorical data, high-dimensional data, and three-dimensional data such as time-series data.

4.2.2. Value. Because of the low value density of data, a data mining algorithm is needed to find the important information contained in the data.

4.2.3. Velocity. The processing of big data requires not only fast batch processing speed but also real-time data processing.

4.2.4. Veracity. Generally, big data contains some wrong data or noise data.

According to the above characteristics of big data, the research on parallel clustering algorithm based on Spark can be carried out from five aspects:

(1) Research on a parallel clustering algorithm that can deal with more types of data: categorical data is different from quantitative data, its value is discrete, there is no natural order between different categorical values, and there is no distance measurement information. Therefore, it is necessary to further study the parallel clustering algorithm which can process categorical data. Generally, big data has a large number of attributes. Because "the curse of dimension" will lead to the failure or meaningless of classical distance measurement methods, it is necessary to further study the parallel clustering algorithm based on subspace or dimension reduction technology to process high-dimensional data. Three-dimensional data are usually time- or location-related data, such as gene-sample-time series in bioinformatics or item-time-location data in market analysis. It is necessary to further study the parallel clustering algorithm that can process three-dimensional data to find clusters across time or location.

(2) Research on parallel clustering algorithm which can find the significant clusters in the data: big data with a large amount of data and attributes will lead to a great increase in the number of clusters generated by

clustering, and the clustering results become redundant and poorly interpretable. The concept of maximum cluster or significant cluster can be used to study the parallel clustering algorithm of important clusters in data to improve the interpretability of clustering results and reveal the important information in data.

(3) Research on parallel clustering algorithm with better performance: grid-based clustering algorithm transforms the clustering of data objects into clustering of grid units, which has better performance. We can further study the grid-based parallel clustering algorithm supported by Spark. Because the performance of grid-based clustering algorithm is closely related to data space and grid size, and the number of data points has little effect on the performance of this kind of algorithm, it is more suitable for big data clustering.

(4) Research on a parallel clustering algorithm which can deal with streaming data: in addition to clustering of stored data in batch, we also need to study a parallel clustering algorithm which can process streaming data in real time to further improve the real time and responsiveness of clustering.

(5) Research on a parallel clustering algorithm that can deal with noise: noise data generally contains the values of error, missing or unknown, which often appears in big data. Therefore, further research on parallel clustering algorithm which can deal with noise is also an important research direction.

Conflicts of Interest

The authors declare that there are no conflicts of interest in connection with the work submitted.

References

[1] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, no. 14, Berkeley, CA, USA, January 1967.

[2] H.-S. Park and C.-H. Jun, "A simple and fast algorithm for K-medoids clustering," *Expert Systems with Applications*, vol. 36, no. 2, pp. 3336–3341, 2009.

[3] M. Ester, H. P. Kriegel, J. Sander, and X. Xiaowei, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Kdd*, vol. 96, no. 34, Portland, OR, USA, August 1996.

[4] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch," *ACM Sigmod Record*, vol. 25, no. 2, pp. 103–114, 1996.

[5] A. Hinneburg and D. A. Keim, "Optimal grid-clustering: towards breaking the curse of dimensionality in high-dimensional clustering," in *Proceedings of the 25th International Conference on Very Large Databases*, Edinburgh, Scotland, September 1999.

[6] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Springer Science & Business Media, Berlin, Germany, 2013.

[7] R. Krishnapuram and J. M. Keller, "The possibilistic c-means algorithm: insights and recommendations," *IEEE Transactions on Fuzzy Systems*, vol. 4, no. 3, pp. 385–393, 1996.

[8] S. Guha, R. Rastogi, and K. Shim, "Cure: an efficient clustering algorithm for large databases," *Information Systems*, vol. 26, no. 1, pp. 35–58, 2001.

[9] G. Karypis, E.-H. Han, and V. Kumar, "Chameleon: hierarchical clustering using dynamic modeling," *Computer*, vol. 32, no. 8, pp. 68–75, 1999.

[10] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics," *ACM Sigmod Record*, vol. 28, no. 2, pp. 49–60, 1999.

[11] A. Hinneburg and D. A. Keim, "An efficient approach to clustering in large multimedia databases with noise," in *Proceedings of the KDD'98: Fourth International Conference on Knowledge Discovery and Data Mining*, vol. 98, New York, NY, USA, August 1998.

[12] G. Sheikholeslami, S. Chatterjee, and A. Zhang, "Wavecluster: a multi-resolution clustering approach for very large spatial databases," in *Proceedings of the 24th VLDB Conference*, vol. 98, New York, NY, USA, August 1998.

[13] W. Wang, J. Yang, and R. Muntz, "STING: a statistical information grid approach to spatial data mining," in *Proceedings of the 23rd VLDB Conference*, vol. 97, Athens, Greece, August 1997.

[14] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic subspace clustering of high dimensional data for data mining applications," in *Proceedings of the International Conference on Management of Data*, pp. 94–105, Seattle, WA, USA, June 1998.

[15] A. Quigley and P. Eades, "FADE: graph drawing, clustering, and visual abstraction," in *Proceedings of the 8th International Symposium on Graph Drawing*, pp. 197–210, Williamsburg, VA, USA, September 2000.

[16] W. R. Fox, L. Kaufman, and P. J. Rousseeuw, "Finding groups in data: an introduction to cluster analysis," *Applied Statistics*, vol. 40, no. 3, pp. 486-487, 1991.

[17] R. T. Ng and J. Han, "Efficient and effective clustering methods for spatial data mining. 51," in *Proceedings of the 20th VLDB Conference*, vol. 88, no. 9, pp. 144–155, Santiago, Chile, September 1994.

[18] C. C. Aggarwal and P. S. Yu, "Redefining clustering for high-dimensional applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, no. 2, pp. 210–225, 2002.

[19] C. L. Chowdhary, M. Mittal, P. Kumaresan, P. A. Pattanaik, and Z. Marszalek, "An efficient segmentation and classification system in medical images using intuitionist possibilistic fuzzy C-mean clustering and fuzzy SVM algorithm," *Sensors*, vol. 20, no. 14, p. 3903, 2020.

[20] C. L. Chowdhary and D. P. Acharjya, "Clustering algorithm in possibilistic exponential fuzzy c-mean segmenting medical images,"*Journal of Biomimetics, Biomaterials and Biomedical Engineering*, vol. 30, Trans Tech Publications Ltd, 2017.

[21] C. L. Chowdhary and D. P. Acharjya, "Segmentation of mammograms using a novel intuitionistic possibilistic fuzzy c-mean clustering algorithm," in *Nature Inspired Computing*, pp. 75–82, Springer, Singapore, 2018.

[22] P. G. Shynu, H. MdShayan, and C. Chowdhary, "A fuzzy based data perturbation technique for privacy preserved data mining," in *Proceedings of the 2020 International Conference on Emerging Trends in Information Technology and Engineering (Ic-ETITE)*, IEEE, Tamilnadu India, February 2020.

[23] Ishwarappa and J. Anuradha, "A brief introduction on big data 5Vs characteristics and Hadoop technology," *Procedia Computer Science*, vol. 48, pp. 319–324, 2015.

[24] J. L. Reyes-Ortiz, L. Oneto, and D. Anguita, "Big data analytics in the cloud: spark on Hadoop vs MPI/OpenMP on beowulf," *Procedia Computer Science*, vol. 53, pp. 121–130, 2015.

[25] B. Zerhari, A. AitLahcen, and S. Mouline, "Big data clustering: algorithms and challenges," in *Proceedings of International Conference on Big Data, Cloud and Applications (BDCA'15)*, Tetuan, Morocco, May 2015.

[26] A. Ayed Ben, M. B. Halima, and A. M. Alimi, "Survey on clustering methods: towards fuzzy clustering for big data," in *Proceedings of the 2014 6th International Conference of Soft Computing and Pattern Recognition (SoCPaR)*, IEEE, Tunis, Tunisia, August 2014.

[27] C. L. P. Chen and C.-Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: a survey on big data," *Information Sciences*, vol. 275, pp. 314–347, 2014.

[28] A. Fahad, N. Alshatri, Z. Tari et al., "A survey of clustering algorithms for big data: taxonomy and empirical analysis," *IEEE Transactions on Emerging Topics in Computing*, vol. 2, no. 3, pp. 267–279, 2014.

[29] L. Rutkowski, "Clustering for data mining: a data recovery approach," *Psychometrika*, vol. 72, no. 1, pp. 109-110, 2007.

[30] C. Zhang, D. Ouyang, and J. Ning, "An artificial bee colony approach for clustering," *Expert Systems with Applications*, vol. 37, no. 7, pp. 4761–4767, 2010.

[31] T. Cura, "A particle swarm optimization approach to clustering," *Expert Systems with Applications*, vol. 39, no. 1, pp. 1582–1588, 2012.

[32] R. P. Browne, P. D. McNicholas, and M. D. Sparling, "Model-based learning using a mixture of mixtures of Gaussian and uniform distributions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 4, pp. 814–817, 2011.

[33] G. J. McLachlan and K. E. Basford, *Mixture Models: Inference and Applications to Clustering*, Vol. 38, Marcel Dekker, New York, NY, USA, 1988.

[34] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EMAlgorithm," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39, no. 1, pp. 1–22, 1977.

[35] A. Spark, Apache spark, 2018.

[36] White and T. Hadoop, *The Definitive Guide*, O'Reilly Media, Inc., Sebastopol, CA, USA, 2012.

[37] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: cluster computing with working sets," in *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing*, Berkeley, CA, USA, July 2010.

[38] E. R. Sparks, A. Talwalkar, V. Smith et al., "MLI: an API for distributed machine learning," in *Proceedings of the 2013 IEEE 13th International Conference on Data Mining*, IEEE, Dallas, Texas, USA, October 2013.

[39] B. Shangguan and P. Yue, "SPARK processing of computing-intensive classification of remote sensing images: the case on K-means clustering algorithm," in *Proceedings of the 2018 26th International Conference on Geoinformatics*, IEEE, Kunming, China, June 2018.

[40] X. Mallios, V. Vassalos, T. Venetis, and A. Vlachou, "A framework for clustering and classification of big data using spark," in *Proceedings of the OTM Confederated International Conferences on the Move to Meaningful Internet Systems*, Springer, Rhodes, Greece, October 2016.

[41] B. Wang, J. Yin, Q. Hua, and Z. Wu, "Parallelizing k-means-based clustering on spark," in *Proceedings of the 2016 International Conference on Advanced Cloud and Big Data (CBD)*, IEEE, Chengdu, China, August 2016.

[42] D. Arthur and S. Vassilvitskii, *K-Means++: The Advantages of Careful Seeding*, Stanford, Stanford, CA, USA, 2006.

[43] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii, "Scalable k-means++," 2012, https://arxiv.org/abs/1203.6402.

[44] Y. Zhao and K. George, *Criterion Functions for Document Clustering: Experiments and Analysis*, University of Minnesota, Minneapolis, MN, USA, 2002.

[45] J. Cao, Z. Wu, J. Wu, and H. Xiong, "SAIL: summation-based incremental learning for information-theoretic text clustering," *IEEE Transactions on Cybernetics*, vol. 43, no. 2, pp. 570–584, 2013.

[46] I. Kusuma, M. Anwar Ma'sum, N. Habibie, W. Jatmiko, and H. Suhartanto, "Design of intelligent k-means based on spark for big data clustering," in *Proceedings of the 2016 International Workshop on Big Data and Information Security (IWBIS)*, IEEE, Jakarta, Indonesia, October 2016.

[47] V. Santhi and R. Jose, "Performance analysis of parallel K-means with optimization algorithms for clustering on spark," in *Proceedings of the International Conference on Distributed Computing and Internet Technology*, Springer, Bhubaneswar, India, January 2018.

[48] X.-S. Yang, "Bat algorithm: literature review and applications," 2013, https://arxiv.org/abs/1308.3900.

[49] X.-S. Yang, "Firefly algorithms for multimodal optimization," in *Prcoeedings of the International Symposium on Stochastic Algorithms*, Springer, Sapporo, Japan, October 2009.

[50] F. Glover, "Tabu search-part I," *ORSA Journal on Computing*, vol. 1, no. 3, pp. 190–206, 1989.

[51] F. Glover, "Tabu search-part II," *ORSA Journal on Computing*, vol. 2, no. 1, pp. 4–32, 1990.

[52] Y. Lu, B. Cao, C. Rego, and F. Glover, "A Tabu search based clustering algorithm and its parallel implementation on spark," *Applied Soft Computing*, vol. 63, pp. 97–109, 2018.

[53] Y. Wang and Q. Qian, "A spark-based artificial bee colony algorithm for large-scale data clustering," in *Proceedings of the 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, IEEE, Exeter, United Kingdom, June 2018.

[54] M. Sherar and F. Zulkernine, "Particle swarm optimization for large-scale clustering on apache spark," in *Proceedings of the 2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, IEEE, Honolulu, HI, USA, November 2017.

[55] T. C. Havens, J. C. Bezdek, C. Leckie, L. O. Hall, and M. Palaniswami, "Fuzzy c-means algorithms for very large data," *IEEE Transactions on Fuzzy Systems*, vol. 20, no. 6, pp. 1130–1146, 2012.

[56] N. Bharill and A. Tiwari, "Handling big data with fuzzy based classification approach," in *Advance Trends in Soft Computing*, pp. 219–227, Springer, Cham, Switzerland, 2014.

[57] N. Bharill, A. Tiwari, and A. Malviya, "Fuzzy based scalable clustering algorithms for handling big data using apache spark," *IEEE Transactions on Big Data*, vol. 2, no. 4, pp. 339–352, 2016.

[58] R. Gu, Y. Tang, C. Tian et al., "Improving execution concurrency of large-scale matrix multiplication on distributed data-parallel platforms," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 9, pp. 2539–2552, 2017.

[59] H. Kacem, M. Aymen Ben, C. B. N'Cir, and N. Essoussi, "KP-S: a spark-based design of the K-prototypes clustering for big data," in *Proceedings of the 2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*, IEEE, Hammamet, Tunisia, October 2017.

[60] Z. Huang, "Clustering large data sets with mixed numeric and categorical values," in *Proceedings of the 1st Pacific-Asia Conference on Knowledge Discouvery and Data Mining*, River Edge, NJ, USA, February 1997.

[61] S. C. Johnson, "Hierarchical clustering schemes," *Psychometrika*, vol. 32, no. 3, pp. 241–254, 1967.

[62] C. Jin, "A scalable hierarchical clustering algorithm using spark," in *Proceedings of the 2015 IEEE First International Conference on Big Data Computing Service and Applications*, IEEE, Redwood City, CA, USA, March 2015.

[63] W. Hendrix, "Parallel hierarchical clustering on shared memory platforms," in *Prcoeedings of the 2012 19th International Conference on High Performance Computing*, IEEE, Pune, India, December 2012.

[64] S. M. Savaresi and D. L. Boley, "On the performance of bisecting K-means and PDDP," in *Proceedings of the 2001 SIAM International Conference on Data Mining Society for Industrial and Applied Mathematics*, Chicago, IL, USA, March 2001.

[65] S. Hong, W. Choi, and W.-K. Jeong, "GPU in-memory processing using spark for iterative computation," in *Proceedings of the 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, IEEE, Madrid, Spain, May 2017.

[66] A. Shobanadevi and G. Maragatham, "Studying the performance of clustering techniques for biomedical data using spark," in *Proceedings of the 2017 International Conference on Intelligent Sustainable Systems (ICISS)*, IEEE, December 2017.

[67] F. Huang, Q. Zhu, J. Zhou et al., "Research on the parallelization of the DBSCAN clustering algorithm for spatial data mining based on the spark platform," *Remote Sensing*, vol. 9, no. 12, p. 1301, 2017.

[68] Y. He, H. Tan, W. Luo, H. Mao, and D. Ma, "Mr-dbscan: an efficient parallel density-based clustering algorithm using mapreduce," in *Proceedings of the 2011 IEEE 17th International Conference on Parallel and Distributed Systems*, IEEE, Tainan, Taiwan, December 2011.

[69] A. M. Aryal and S. Wang, "SparkSNN: a density-based clustering algorithm on spark," in *Proceedings of the 2018 IEEE 3rd International Conference on Big Data Analysis (ICBDA)*, IEEE, Shanghai, China, March 2018.

[70] L. Ertöz, M. Steinbach, and V. Kumar, "Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data," in *Proceedings of the 2003 SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics*, San Francisco, CA, USA, May 2003.

[71] R. Liu, X. Li, L. Du, S. Zhi, and M. Wei, "Parallel implementation of density peaks clustering algorithm based on spark," *Procedia Computer Science*, vol. 107, pp. 442–447, 2017.

[72] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492–1496, 2014.

[73] M. Liang, Q. Li, Y. Geng, J. Wang, and Z. Wei, "REMOLD: an efficient model-based clustering algorithm for large datasets with spark," in *Proceedings of the 2017 IEEE 23rd International Conference on Parallel and Distributed Systems (ICPADS)*, IEEE, Shenzhen, China, December 2017.

[74] M. Datar, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, Brooklyn, NY, USA, June 2004.

[75] Y.-A. Geng, Q. Li, R. Zheng, F. Zhuang, R. He, and N. Xiong, "RECOME: a new density-based clustering algorithm using relative KNN kernel density," *Information Sciences*, vol. 436-437, pp. 13–30, 2018.

[76] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, no. 5, pp. 993–1022, 2003.

[77] E. Müller, S. Günnemann, I. Assent, and T. Seidl, "Evaluating clustering in subspace projections of high dimensional data," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 1270–1281, 2009.

[78] Bo Zhu, A. Mara, and M. Alberto, "CLUS: parallel subspace clustering algorithm on spark," in *Proceedings of the East European Conference on Advances in Databases and Information Systems*, Springer, Poitiers, France, September 2015.

[79] K. Kailing, H.-P. Kriegel, and Peer Kröger, "Density-connected subspace clustering for high-dimensional data," in *Proceedings of the 2004 SIAM International Conference on Data Mining.Society for Industrial and Applied Mathematics*, Lake Buena Vista, FL, USA, April 2004.

[80] G. H. Zhu, S.-B. Huang, C.-F. Yuan et al., "SCoS: the design and implementation of parallel spectral clustering algorithm based on spark," *Chinese Journal of Computers*, vol. 41, no. 4, pp. 868–885, 2018.

[81] O. Backhoff and E. Ntoutsi, "Scalable online-offline stream clustering in apache spark," in *Proceedings of the 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, IEEE, Barcelona, Spain, December 2016.

[82] Y. Gong, R. O. Sinnott, and R. Paul, "Rt-dbscan: real-time parallel clustering of spatio-temporal data using spark-streaming," in *Proceedings of the International Conference on Computational Science*, Springer, Wuxi, China, June 2018.

[83] Q. Zhou and J. Wang, "SparkSCAN: a structure similarity clustering algorithm on spark," in *Proceedings of the National Conference on Big Data Technology and Applications*, Springer, Harbin, China, December 2015.

[84] A. I. Taloba, M. R. Riad, T. Hassan, and A. Soliman, "Developing an efficient spectral clustering algorithm on large scale graphs in spark," in *Proceedings of the 2017 Eighth International Conference on Intelligent Computing and Information Systems (ICICIS)*, IEEE, Cairo, Egypt, December 2017.

[85] E. Ivannikova, "Scalable implementation of dependence clustering in apache spark," in *Proceedings of the 2017 Evolving and Adaptive Intelligent Systems (EAIS)*, IEEE, Ljubljana, Slovenia, May 2017.

[86] D. Liu and J. Li, "PGCAS: a parallelized graph clustering algorithm based on spark," in *Proceedings of the International Conference on Big Scientific Data Management*, Springer, Beijing, China, November 2018.