Comput Syst Sci & Eng (2019) 5: 259–281
© 2019 CRL Publishing Ltd

International Journal of
Computer Systems
Science & Engineering

A Survey and Systematic Categorization of Parallel K-means and Fuzzy-c-Means Algorithms

Ahmed A. M. Jamel[1]* **and Bahriye Akay**[2]†

[1] *Erciyes University, Institute of Natural and Applied Sciences, Department of Computer Engineering, 38039, Melikgazi, Kayseri, Turkey*
[2] *Erciyes University, Engineering Faculty, Department of Computer Engineering, 38039, Melikgazi, Kayseri, Turkey*

Parallel processing has turned into one of the emerging fields of machine learning due to providing consistent work by performing several tasks simultaneously, enhancing reliability (the presence of more than one device ensures the workflow even if some devices disrupted), saving processing time and introducing low cost and high-performance computation units. This research study presents a survey of parallel K-means and Fuzzy-c-means clustering algorithms based on their implementations in parallel environments such as Hadoop, MapReduce, Graphical Processing Units, and multi-core systems. Additionally, the enhancement in parallel clustering algorithms is investigated as hybrid approaches in which K-means and Fuzzy-c-means clustering algorithms are integrated with metaheuristic and other traditional algorithms.

Keywords: Clustering, Hadoop, Machine learning, Metaheuristic Algorithms, Multicore processing, parallel computing

1. INTRODUCTION

Data mining is a new inter-disciplinary field combining concepts from machine learning, statistics, databases, and parallel computing [1]. It refers to all comprehensive processes for discovering new patterns or building models from a given dataset. Additionally, it is the extraction of useful information or patterns from a massive amount of raw data [2]. Statistical analysis uses mathematical formulas to extract useful information based on specific models and using static data [3]. Machine learning is a branch of artificial intelligence modeling a system's behavior and predicting future results using instances of data or employing past experiences. The types of machine learning are supervised and unsupervised learning. The supervised learning supports the classification techniques such as support vector machine and artificial neural network. Unsupervised learning supports clustering algorithms such as K-means and fuzzy-c-means (FCM) algorithms.

Extracting useful information and reaching the specific data by traditional and sequential algorithms is challenging because of analyzing the data by them requires plenty of time and capacity. The main challenges of sequential clustering algorithms are scaling up to enormous sizes of databases. This case will lead to high computational and spatial costs. In order to obtain scalable machine learning algorithms, the researchers introduced parallel processing in which significant problems are divided into smaller chunks to be manipulated concurrently [4]. Recently, some multiple environments have been deployed to fulfill big data analysis in parallel, i.e., Hadoop, MapReduce, Spark, and the same circumstances used in the infrastructure of many well-known companies. Likewise, there has been a surge in processor manufacturing, where the companies have developed multi-core processors that can analyze and calculate data in parallel. The properties of parallel processing provide convenience in code writing, debugging, and testing. In some environments, a parallel processing topology consists of one master and many slave nodes. Master node distributes the chunk of dataset among slave nodes, and the associated job is accomplished concurrently. In other surroundings such as Graphical Processing Unit (GPU) and Central Processing Unit (CPU), the n data points are divided into p parts and each participating p processor is responsible for handling n/p data points.

*Email: 4010941311@erciyes.edu.t
†bahriye@erciyes.edu.t

Conversion of a sequential program into a parallel program is still challenging because most of the programs have been designed and implemented sequentially and converting a massive sequential program into a parallel program is a complicated task. Lack of tools for debugging, tracing, and testing the parallel programs is a disadvantage in parallel programming. Additionally, a program written in a parallel manner may not be ported in other environments. A parallel program written in Spark environment is different from that in MapReduce. The program written for several core processors may not perform well on a different number of core processors nor a program may not be complying with any other computing hardware and software architecture. The communication bottleneck in parallel programming is another issue when it is implemented in the MapReduce and multicore processors because the same communication channel is used for all the processors to access the data and result. All these challenges should be investigated by researchers and developers.

Due to the importance of parallel computing, this review article lists comprehensive research about parallel K-means and FCM algorithms. We investigated and focused only on the parallel K-means and FCM algorithms and their implementations in various platforms. We reviewed about 250 papers regarding K-means and FCM algorithms from different types of sources (i.e., Elsevier, Springer, IEEE, and other databases). We chose the papers related to parallel implementations, big data, and clustering algorithms.

We systematically categorized these studies according to the parallel file systems (Hadoop, MapReduce, and Spark) they are implemented on, according to the software libraries they use for CPUs or GPUs, and according to the enhancements proposed for the parallel K-means and FCM clustering. To the best of our knowledge, there is no report regarding multicore CPU and GPU studies and regarding the enhancements related to the parallel K-means and FCM algorithms in which metaheuristics and other algorithms are integrated to them. This paper will enable the readers to discover detailed information concerning the parallel K-means and FCM clustering algorithms.

The remaining of the paper is organized as the following: In Section 2, detailed information about parallel computing is given. In Section 3, we provide general information about clustering and K-means and FCM. In Section 4, a discussion is presented about the parallel K-means, and FCM clustering algorithms implemented on parallel file systems and implemented usingsoftware libraries. In Section 5, we discuss the enhancement of parallel K-means and FCM clustering algorithms combined with metaheuristics and other algorithms as a hybrid approach. In Section 6, the applications of parallel K-means and FCM clustering algorithms are mentioned. Finally, in Section 7, we give a conclusion the challenges, and our future investigations based on this survey study

2. PARALLEL COMPUTING

Parallel computing is expected to overcome the sequential processing bottlenecks, to provide scalable massive datasets, to enhance the performance and to improve the response and execution time [1]. Distributed memory machines (DMM) and shared-memory (SMP) are two different approaches in parallel computing. Both have the same goal in improving the performance of traditional data mining approaches, but they are built on different architectures. DMM data mining computers are communicating with each other using MPI library, while in SMP approach the computers deal with processors sharing memory or disk. This difference between the two approaches affects the algorithm design, performance, and cost model [5]. SMP systems have multiple processors which process with the same clock frequency and share the same memory (Figure 1). Since the communication between the processors is over shared memory, data sharing is quite fast. In such systems, the single operating system manages all processors and memory.

DMM architecture involves geographically distributed nodes and located over a wide area network (WAN) such as the internet. Furthermore, a distributed system is a group of computers working independently, and it appears to its users as a single cohesive system (Figure 2) [1, 3, 6].

The efficiency and the speed of parallel components in a distributed environment are affected by load balancing, data distribution, minimizing communication [1, 3, 5]. Load balancing is the ability to manage and distribute the flow of traffic and connection among nodes. It aims to increase the efficiency of the parallelization algorithms by minimizing execution time and maximizing resource utilization [7]. Data distribution provides each node of the structure to process datasets that are reduced the subset of the entire databases. Minimization of I/O reduces the amount of I/O processes being substantial benefits distributed in parallel data mining.

A cluster is a term referring to a group of computers, machines, servers—each of the elements inside a cluster called node. At least two nodes are needed to create a cluster. A cluster must include a master node which is responsible for distributing the job among the slaves and slave nodes which are responsible for fulfilling the job came from the master node. Speed-up, size-up, and runtime are the main metrics that are used to measure the quality of a cluster and the performance of an experiment. The runtime is the total time spent in the clustering process. The speed-up metric defined by Equation 1 is the ratio of the runtime of a task on one node(T_1) and the runtime of the same task on a cluster of m nodes (T_m) [8];

$$speedup(m) = \frac{T_1}{T_m} \tag{1}$$

When the number of nodes increases, the speed-up of the parallel algorithm rises as well. However, achieving a linear speedup is challenging because the communication cost increases as the number of clusters become large. The size-up measure is the duration that an algorithm takes on a particular cluster when the size of the dataset is (m) times the size of the original dataset. The size-up is given by Equation 2.

$$sizeup(DS, m) = \frac{T_{\mathrm{m}\,DS}}{T_{DS}} \tag{2}$$

T_{DS} indicates the runtime for clustering DS on a given cluster, and T_{mDS} indicates the run time for clustering m* DS on the same cluster.

Flynn's Taxonomy [9] categorizes the computer architectures into four classes based on several instruction streams and data streams, as shown in Figure 3. These classes are Single

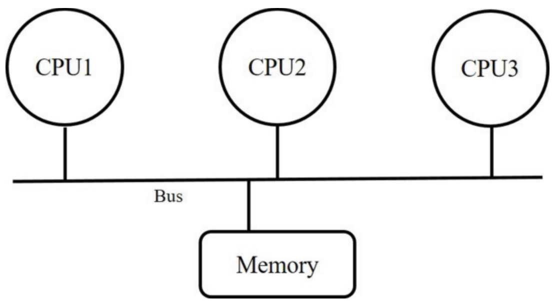

Figure 1 Shared memory architecture in parallel environment.

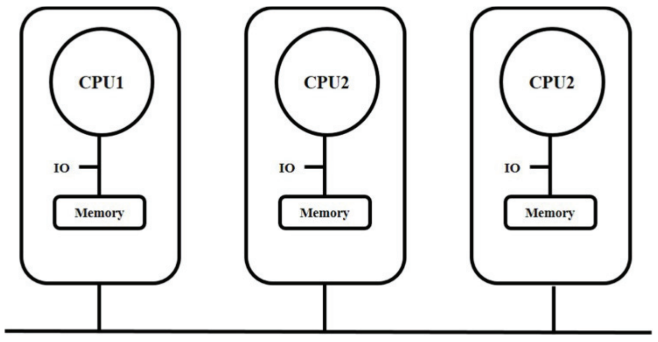

Figure 2 Distributed data mining environment.

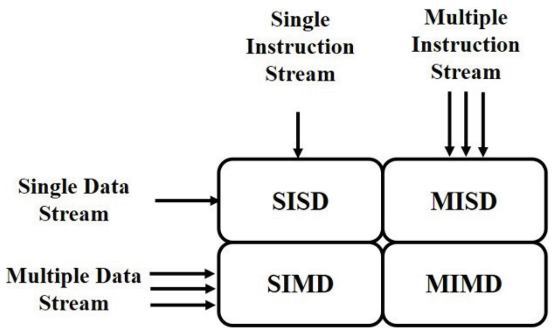

Figure 3 Flynn taxonomy.

Instruction Single Data (SISD), Single Instruction Multiple Data (SIMD) utilized in multi-core computers, Multiple Instruction Single Data (MISD), Multiple Instruction Multiple Data (MIMD) utilized for cluster applications. These architectures can be combined with execution model used in parallel computing, using threads.

A thread is the smallest processing unit and sequence of programmed instructions that can be managed independently by a scheduler, which is a part of the operating system. It allows one process to perform several tasks concurrently. In other words, a thread can perform several tasks in parallel. It means that the process will not need to wait until the user finishes working while another job starts.

Single threading: The process with a single thread performs one task.

Multi-threading: The processes with multiple threads perform several tasks (Figure 4).

3. CLUSTERING

Clustering is the process of placing data in similar groups. It is a branch of unsupervised data mining and useful in various fields such as pattern recognition, machine learning, image segmentation, computer graphics, learning theory [10]. The main criterion to obtain an optimal result in clustering is the similarity of data to its cluster and the dissimilarity of data to other clusters. In other words, the aim is to maximizing the intra-cluster similarities and minimizing inter-cluster similarities.

K-means and FCM algorithms belong to the clustering algorithms. The difference between them is that K-means assigns each data point exactly to one cluster, while FCM assigns each data point to multiple clusters with a degree of membership. Therefore, when datasets have some uncertain cases such as outliers, noises, FCM is likely to produce better results than K-means does.

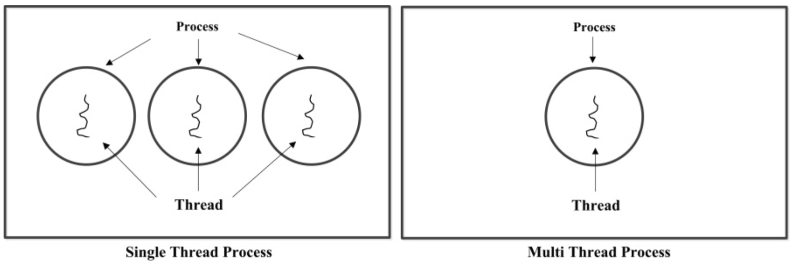

Figure 4 Single and multi-thread processes.

Algorithm 1

```
1: Determine k centroids
2: Calculate the distance between the data vector and cluster centroid by this equation:
   $$d_{ij} = \sqrt{\sum_{k=1}^{n} \left(x_{ik} - m_{jk}\right)^2} \quad (3)$$
   where x denoted to the vector of data and m to centroid respectively.
3: Assign each data vector to the closest centroid.
4: Take an average of vectors to obtain new centroid.
5: Repeat step 2–4 till the steps stabilize or converged.
```

3.1 Sequential K-means Algorithm

The main idea of the K-means algorithm is to divide a dataset consisting of *n* objects into *k* sets. The steps of the K-means algorithm are given in Algorithm 1.

The algorithm starts by initializing *k* by a scheme such as a random selection from the dataset. Then, each data vector is assigned to the nearest cluster centroid based on distance metrics. Euclidian (Equation 3), Cosine, Manhattan distances are commonly used metrics. When each dataset vector sare assigned to a centroid, a new cluster centroid is calculated for each cluster. The whole dataset is reassigned to new cluster centers. The steps of centroid assignment and re-calculation are repeated until the cluster centroids reach a stable state [11]. K-means algorithm aims to minimize the Within-Cluster Sum Of Square (WCSS) given by Equation 4 [12]:

$$\min \sum_{i=1}^{k} \sum_{x \in c} dis^2(x, m) \quad (4)$$

where x is the current vector or pattern, c is the set of vectors or patterns and m is cluster centroid. One way to minimize WCSS value is increasing the number of clusters k.

Algorithm 2

```
Let X = {$x_1$, $x_2$, $x_3$ …, $x_n$} be the set of data points and $V$ = {$v_1$, $v_2$, $v_3$ …, $v_c$} be the set of centers.
1: Select "c" cluster center randomly.
2: Calculate the fuzzy membership "$\mu_{ij}$" using:
   $$u_{ji} = \left(\sum_{k=1}^{c} \left(\frac{d_{ji}}{d_{ki}}\right)^{\frac{2}{m-1}}\right)^{-1} \quad \text{Where } d_{ji} = \left[x_i - v_j\right] \quad (5)$$
   where: 'n' is the number of data points. '$\mu_{ij}$' represents the membership of i$^{th}$ data to the j$^{th}$ cluster center and
   $$\sum_{j=1}^{c} u_{ji} = 1 \quad (6)$$
   'm' is the fuzziness index m ∈ 1, ∞.
   'd$_{ij}$' represents the Euclidean distance between i$^{th}$ data and the j$^{th}$ cluster center.
   'c' represents the number of cluster centers.
3: Calculate the fuzzy centers '$v_j$' using:
   $$v_i = \frac{\sum_{i=1}^{n} \left(u_{ji}\right)^m x_i}{\sum_{i=1}^{n} \left(u_{ji}\right)^m} \quad (7)$$
   where: '$v_j$' represents the j$^{th}$ cluster center.
4: Repeat steps 2 and 3 until the minimum '$J$' value is achieved $||U(l+1) - U(k)|| < \beta$ where 'k' is the iteration.'$\beta$' is the termination criterion between [0, 1]. 'U' = ($\mu$ij) n*c' is the fuzzy membership matrix. 'J' is the objective function defined by Equation 8.
   $$J(U, V) = \sum_{i=1}^{n} \sum_{j=1}^{c} \left(\mu_{ij}\right)^m \left\|x_i - v_j\right\|^2 \quad (8)$$
   where ||xi − vj|| is the Euclidean distance between ith data and jth cluster center.
```

3.2 Fuzzy-C-Means Algorithm

FCM algorithm is an improved and generalized version of K-means algorithm. It has been used in various fields such as image processing, remote sensing, and brain MR image segmentation [12–15]. The steps of the algorithm are given in Algorithm 2.

FCM algorithm starts with assigning membership of each data point to every cluster based on distance metric such as Euclidean metric. In FCM a dataset sample is assigned to a membership value (Equation 5) based on its similarity with the cluster center (centroid) whereas the membership values range is between 0 and 1 and the similarity means higher the membership value [16, 17]. A data point closer to a cluster center has a higher membership associated with this cluster center. The summation of membership of each data point should be equal to one (Equation 6). The cluster centroid is calculated based on Equation 7.

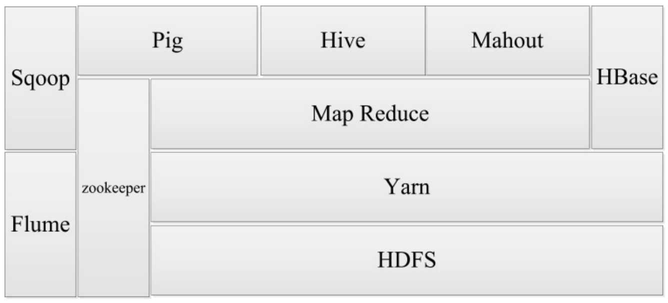

Figure 5 Hadoop ecosystem [Hurwits et al. 2013, [18]].

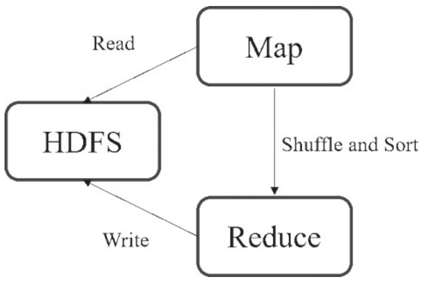

Figure 6 MapReduce structure.

4. PARALLEL K-MEANS AND FCM CLUSTERING ALGORITHMS REVIEW

In this section, we presented a review on the studies of parallel K-means and FCM clustering algorithms. In the first subsection, studies that were carried out on Hadoop, MapReduce, Spark are provided. Hadoop is an online-library written in java language used for parallelizing and processing large-scale data, and Hadoop has a component used for storing data named Hadoop Distributed file system (HDFS) (Figure 5). MapReduce is an element of Hadoop that used for parallel processing consisting of two separated tasks map allowing Hadoop to perform parallel processing quicker. Spark, Mahout, and Flink are one of Apache open source libraries, also used for processing large-scale data. After that, studies performed on a hardware environment that splits the big data along its multiprocessors such as CPU, GPU, and using MPI is presented.

4.1 Studies of Parallel K-Means and FCM Clustering Algorithm Implementations on Parallel File Systems

4.1.1 Studies with Hadoop and MapReduce

Hadoop is an open-source library written in JAVA programming language allowing parallel and distributed processing on large-scale data across multiple cluster machines. Hadoop stores and manages datasets on multiple machines. HDFS is a component in Hadoop that is responsible for storing data. In the Hadoop framework, datasets are split into blocks and distributed across cluster nodes. A block can be proliferated by replication factor to other nodes as a copy of the block. The main reason for replication is to prevent data loss when one of the nodes is damaged or removed [19].

MapReduce is a component in Hadoop allowing parallel processing of large datasets. The term refers to tow separates and distinct tasks in Hadoop framework. The first is map task, which converts the data into another format of information when elements are formed as <Key, value> pairs. The shuffle task collects and sorts the output data from map and directs it to reduce afterward; it takes the output from map task combines the data and store the result to the source as HDFS [20] (Figure 6).

Hai [8] accomplished Canopy, K-means, and FCM algorithms on Mahout and Hadoop framework. After implementation, the author compared three algorithms according to three metrics Run-time, Speed-up, and Size-up defined by Equations (1) and (2) respectively. The experiment was performed on four datasets with different sizes. By comparison among three algorithms according to the metrics, they reported that when the dataset and the number of nodes are the same, the running time of FCM algorithm is the longest one and Canopy the shortest. As far as the run time concerned, the study concluded that when the number of nodes and dataset in the cluster are same, the run time of FCM is the longest and Canopy the largest. The size-up is smallest, second smallest and largest to Canopy, K-mean, and FCM respectively. Finally, the speed-up of Canopy is the largest one among the three algorithms.

Lv et al. [11] accomplished a sequential and parallel K-means algorithm on Hadoop and MapReduce. Due to Hadoop's incompatibility with the images and their conversion to a text file, the transformation process was performed to transform each pixel from RGB to LAB value. LAB value refers to luminosity 'L' or brightness layer, chromaticity layer 'A*' refers to color falls along the red-green axis, and chromaticity layer 'B' refers to color falls along the blue-yellow axis. This space describes all colors visible to the human eyes. Initial clustering and the total error variance minimization for each cluster are implemented using Hadoop. Finally, the results are acceptable in a parallel process because the visualization of output and assessment by human eyes is more practical and designed for visualizing.

Garg and Trivedi [17] implemented the parallel FCM algorithm on five various datasets with different sizes based on multi-node Hadoop cluster and MapReduce being built using Amazon Elastic Cloud Computing (Amazon EC2). The datasets are converted to vector format. Distance calculation, creating membership value, and assigning data point to closest centroid are performed in the map function. Recalculation the centroid for each cluster is performed in reduce function. They concluded that as the number of nodes increases and the execution time reduces.

Moertini and Venica [19] implemented a technique that enhances the parallel implementation of the traditional K-means algorithm based on MapReduce. The enhancement techniques are: 1) data preprocessing by performing attributes selection, cleaning, and transformation along with the clustering process. 2) Reducing the number of iterations by computing initialization of centroids in map function. 3) Producing clustering patterns and generating clusters of quality measures. When the technique was tested on higher specification computer, they showed that the K-means algorithm based on MapReduce scaled better when it worked on two computers.

Zhao et al [20] adopted the K-means algorithm in MapReduce model, which was implemented by Hadoop to build the clustering process applicable to analyze big data. The performance of the proposed algorithm was evaluated according to speedup, scale-up, and size-up criteria. According to the evaluation metrics, the presented algorithm achieved a very good performance.

Xia et al. [21] introduced an algorithm for solving the traffic problem of subarea roads in Beijing by Parallel Three-Phase K-Means (Par3PKM) algorithm. They used large scale taxi movement's dataset. In order to increase the efficiency and scalability of Par3PKM, the optimal K-Means algorithm was applied on MapReduce framework in Hadoop. The optimization was performed by using Distributed Traffic Subarea Division (DTSAD) method and using large-scale GPS trajectories of taxicabs. The optimization and parallelization of Par3PKM's algorithm reduced memory usage and reduced computational cost, thus significantly improvement was achieved in traffic accuracy, efficiency, scalability, and reliability.

Liao et al. [22] presented an improving of parallel K-means algorithm based on MapReduce by decreasing the number of iterations, and selecting the initial centroids which are consistent with the distribution of the data and distance measure strategy. The proposed algorithm achieved higher processing speed and stability than the traditional K-means.

Liu and Cheng [23] compared both sequential and parallel K-means algorithms according to results efficiency and time consumption. The parallel K-means algorithm was implemented on a cloud system based on MapReduce. They reported that the parallel K-means algorithm is more efficient in terms of time consumption and more suitable for large-scale data processing than the sequential K-means algorithm.

Jiang and Zhang [24] proposed parallel k-Medoids on Hadoop based on MapReduce named HK-Medoids algorithm. The proposed algorithm was performed through three levels. The map function divided each data sample to the nearest cluster center (centroid). The combine function ran K-Medoids clustering algorithm with the clustered data to get a temporary center. In reduce function, the intermediate clusters were clustered again and calculated a new cluster center for every cluster. The algorithm was performed on three datasets. They reported that the speedup of HK-Medoids algorithm increases along with the extended number of the worker nodes, and the speedup decreases when dealing with small sizes datasets.

Lin et al. [25] presented K-means and FCM algorithms on land price dataset in Taichung City from 2006 to 2015. They utilized Hadoop HDFS and MapReduce with R language and visualized results on Google Maps. The project was applied on nine computer nodes. The nine nodes consist of three physical machines, and inside each machine, three virtual machines exist. The physical machines properties are Intel Core i7-4790 CPU and 8GB memory and virtual machines with 1Core CPU and 2GB memory. The master physical machine used a G860 CPU and 8GB memory. They reported that the computation time could be greatly reduced. Additionally, the inefficient memory issue can be solved with enough number of compute nodes. The performance of K-means is better compared with FCM algorithms.

Anchalia et al. [26] implemented the K-means algorithm based on MapReduce on a distributed network. This paper showed the employing of MapReduce for implementing the K-means algorithm. The designed system consists of one master node and seven slaves. The study achieved acceptable results in reducing the implementation costs of processing such huge volumes of data.

Yushui and Lishou [27] presented a clustering technique for clustering Chinese Commodity Information Web large scale dataset based on Hadoop and MapReduce by implementing parallel K-means algorithm. The calculation of the distance between each sample and centroid is performed in map function and calculating new cluster center is performed in Reduce function. They proposed using the range of Unicode characters to extract all the characters on the page and then using the Chinese Academy of Sciences word software ICTCLAS. They proved that the unstructured text of web content to be recognized by MapReduce, it should be converted to text vector. Experimental results showed that this method could achieve better improvement according to the speed-up and scalability of the clustering process.

Zhong and Liu [28] proposed an application of K-means algorithm on spatial data based on the Hadoop platform and MapReduce model. They analyzed the time complexity of parallel K-means. They concluded that the implementation of the K-means algorithm in a parallel manner gives effective results in execution time. However, the time complexity of the parallel K-means spatial clustering algorithm showed in Equation 9.

$$Time\ complexity = \frac{O(nkdt)}{PQ} \tag{9}$$

where P denoted to the nodes in the cluster and Q to the tasks respectively.

Abdouli et al. [29] offered a K-means algorithm to cluster Moroccan users in the social network Twitter. First, the data is converted to numeric feature; then, the K-means clustering algorithm is applied in Hadoop and MapReduce platforms by Python language and Natural Language Processing (NLP). Due to the improvement of usage of different dialects and languages and the changing of the meaning of the same words of Moroccan dialect, the system improvement is their future work.

Chen and Ying [30] carried out the K-means clustering algorithm in a cloud computing model based on Hadoop and MapReduce. They simulated a set of data to scatter them by the K-means algorithm and set k as three. They reported that the K-means algorithm performance on the cloud computing model is easy and quick in terms of data management.

Bandyopadhyay et al. [31] presented a parallel representation of K-means based on Hadoop and MapReduce architecture (HdK-Means) algorithm. The map function assigned the data points to its nearest centroid. The reduce function performed the summation of the data points in the same cluster and the number of the data points. The experimental results showed that the proposed HdK-means algorithm executes faster and more efficient on big data than the sequential K-means algorithm.

Kang and Park [32] tested the performance of parallel K-means algorithm by designing a MapReduce application. The distribution of the dataset into k partitions was performed by running on Twister and Hadoop, and the execution time of both frameworks was evaluated. The comparison between Twister and Hadoop showed that the average elapsed time decreases over 112, 0 times in Twister than Hadoop.

Boukhdhir et al. [33] proposed an enhancement of parallel K-means algorithm for clustering large datasets based on MapReduce in Hadoop framework. The steps are: removing outliers of datasets, selecting initial centroid systematically, and partitioning the dataset into k clusters by using MapReduce. In conclusion, the proposed method takes less time as compared to traditional K-means, PK-means, and Fast K-means.

In summary, due to the difficulty of clustering algorithms to find optimum minima, many researchers studied this problem as reviewed in this section. Different clustering algorithms and algorithms' versions such as K-means and FCM have been implemented on Hadoop MapReduce framework. The main challenges in performing parallel algorithms are how to keep the scalability of the implementation when a growing amount ofthe work is happened by adding resources to the system. Furthermore, also optimizing the speed-up, and size-up metrics is a challenge in parallel computing by using Hadoop MapReduce framework, but still, there is a gap, and more researches are required to obtain the best values. Regarding the datasets, usage of big datasets still not stated in many researches, therefore, using big data in a parallel environment, and its performance evaluation is an open field to study.

4.1.2 Studies with Spark, Mahout, and Flink

Apache Spark is an open-source library written in Scala programming language permitting parallel processing on large-scale data across multiple machines of clusters. Spark is simpler to use and faster than Hadoop. Resilient Distributed Datasets (RDD) is a fundamental data structure of Spark. It is an immutable distributed collection of objects. Each dataset in RDD divides into logical partitions, which may be computed on different nodes of the cluster. Its architecture consists of four components; Spark SQL, Sparks streaming, Machine learning library (MLib) and GraphX [34, 35].

Apache Mahout is an open-source algorithm library for scalable distributed and parallel implementation of machine learning on Hadoop. Apache Mahout is one of the components of the Hadoop ecosystem. It supports three main data science use case: 1) collaborative filtering by mining user behavior and makes product recommendation, 2) clustering by assigning items and putting them into classes, such that the items belonging to similar groups 3) classification by training the items from existing categorization and then assigning unclassified items to the best category [36].

Apache Flink is an open-source stream processing platform used for distributed, high performing, and accurate data streaming applications. Flink is developed by the Apache that performs datasets running with on thousands of nodes with excellent throughput and latency characteristics at large scale [37].

Wang et al. [34] proposed parallelization of K-means algorithm based on Spark clustering framework. They used four datasets, and three ways to select centroid of the K-means algorithm: randomly, sequentially and stochastically. The evaluation of cluster efficiency and a comparison was performed among the proposed methods based on Normalized Mutual Information (NMI) validation measure. They reported that the experiments on large-scale datasets and text datasets demonstrated the effectiveness of proposed algorithms.

Kusuma et al. [35] presented an intelligent parallel K-means algorithm based on Spark in the Hadoop platform. A parallel K-means algorithm was applied on datasets first as RDD and second as batch and comparing computational time between the executions using original and batch RDD. The intelligence in the algorithm comes from changing the original RDD to a batch of data the instances split from the original RDD and maintains the number of records in RDD. The operation is performed as map and reduce used the batch of data. The comparison of the efficiency of the cluster between intelligent K-means and traditional K-means was evaluated using silhouette measurement Equation 10. They concluded that the designed algorithm could speed-up the computational time in analyzing big data problems.

$$S = \frac{b - a}{\max(a, b)} \tag{10}$$

Jain and K Jain [36] accomplished categorization of Twitter users based on the user's interest and implemented their approach by Mahout over Hadoop platform. They used dataset consisted of about thirty thousand tweets. In this study, the tweets of the users were clustered according to their similarity. The proposed work consists of three steps: the first, all the tweets of a user are combined in one document. In the second, a pair of key and value is assigned in sequence file where user name acts as a key, and tweets act as value. In the third, the file is converted into vectors then the K-means was applied by using cosine and Euclidean distance measures. Experimental results of K-means and FCM on Mahout over Hadoop frameworks showed that FCM is slower than K-means in time execution, but gives better results. They proved that Cosine distance is a good measure for finding the distance in clustering text documents.

Li et al. [37] applied an improved K-means algorithm on the context of E-commerce datasets based on Flink platform. The improvement was performed by adopting a kernel density option based on characteristics of density distribution. The study concluded that the improved parallel K-means algorithm is an efficient method to process big data in the context of e-commerce based on the Flink framework.

Manzi and Tompkins [38] used parallel K-means clustering based on the Spark platform. They implemented Word Count and Sorting applications. The steps of the operation in the K-means Clustering implementation are: preparation of dataset to be parsed as text dataset, transformation parsed dataset into 3-dimensional floating points and parallelization of the iterative process of updating the centroids on the GPU. The study concluded that the K-means clustering is a good fit for GPU acceleration.

In another study of Jain and K Jain [39] applied parallel K-means and FCM algorithm on twitter dataset based on Mahout Platform. In order to manipulate the data, the authors converted the twitter dataset from text to vectors. The dataset was extracted from twitter. They collected tweets from different users. The dataset consisted of 30.000 tweets, and it clustered by K-means and FCM algorithms. The authors compared two algorithms according to the number of iterations and executing time. They reported that the FCM converges in less iteration, although it takes more time because of the more calculations when it compared with K-means. In conclusion, FCM is slower than the K-means algorithm in execution.

In summary, Spark, Mahout, and Flink are the file system libraries for processing large-scale data. Since Spark processes the data in memory, it is faster than other frameworks. Mahout is built based on Hadoop MapReduce and stores the data on HDFS. Due to its restriction by disk accesses, it is slow and does not handle iterative jobs very well. Since machine learning algorithms generally perform many iterations, this makes Mahout run slowly for this kind of applications. Flink, which is designed for processing streaming data, has a different approach to memory management. Flink transfers the pages to the disk when the memory is full. Therefore, it is faster than Mahout. There is a lack of studies on Spark, Mahout, and Flink file systems. More studies should be performed on these file systems, especially clustering streaming data on Flink.

4.2 Studies of Parallelization K-Means and FCM Using Software Libraries

In this section, a survey of studies based on software libraries is presented such as OpenMP, CUDA, and MPI based on multicore processors. We presented each field as an independent subsection. The processor manufacturers produce microprocessors with separate execution units. By this development, it has become easy to use multiple computers to work as clusters.

The clustering in multicore architecture starts with portioning the dataset and distributing each partition among cores. Each core computes its process according to a given algorithm, and then the outputs requested result.

4.2.1 Studies With Multicore CPUs Using Open MP Library

Open Multi-Processing (OpenMP) is an application programming interface (API) for parallel programming that supports multi-core computers shared memory multiprocessing programming in C, C++ and FORTRAN on much architecture including UNIX and Microsoft Windows platforms. It consists of a set of compiler directories, library routines.

Baydoun et al. [40] implemented parallel K-means algorithm by using several databases with different classes and features and RGB images by using CPU and GPU. In CPU they used both Cilk Plus (a library used for parallelization developed by Intel) and OpenMP. They concluded that there are effects of different clustering parameters on the performance of the parallel K-means algorithm. These effects based on properties of input data such as the number of clusters, number of samples, and number of features. In fact, not always depend on these factors but also depends on computer properties (e.g., CPU cores, memory capacity, parallel environment, and architecture). Finally, they found that OpenMP is more suitable than Cilk Plus for parallelization task.

Naik et al. [41] accomplished parallel K-means algorithm by using OpenMP. This work optimized and enhanced the execution time without affecting the accuracy. The outliers of datasets cause less efficiency of the clustering algorithm. In this study, before starting parallel implementation, the outlier analysis was performed by preprocessing data to be sorted out as an ascending order, and after that, the sorted data was divided into partitions to start clustering analysis process in a multi-core system for massive datasets. After experiments were performed on three datasets, the accuracy and execution time was reported. The CPU time partitioning method is improved when it compared with the sequential K-means method.

Honggang et al. [42] implemented a parallel K-means and shift mean algorithms on two datasets. In this study, parallel K-means consist of five steps: 1) randomly choosing k cluster centroids and broadcast to threads. 2) each thread computes cluster membership. 3) each thread calculates the partial sum and weight for every cluster centroid. 4) According to the partial sum and weights from each thread and computes the revised centroid. 5) repeat these steps until convergence. They concluded that a linear speed-up could be achieved in up to four threads with various parallelization techniques.

Chu et al. [43] presented various machine learning algorithms on a multi-core system. The tested algorithms were, K-means, logistic regression (LR), naive Bayes (NB), back propagation (NN), ICA, PCA, Locally Weighted Linear Regression (LWLR), Gaussian Discriminant Analysis (GDA), EM, and SVM. They proved that the speedup factor of parallel implementation of the algorithms is linear with an increasing number of processors and showed that the multi-core machines are generally faster than multi-processor machines because of the communication internal to the chip is much less costly.

Hadian and Shahrivari [44] designed a method for clustering large datasets based on 12-core CPU. The method splits the dataset into small blocks. Each of these chunks is clustered, and then the chunk clustering is aggregated to perform the final clustering. The maximum calculations capability of a multi-core machine was used to cluster the dataset in parallel. The method used K-means algorithm for finding cluster centroid and K-means++, which was helped the K-means algorithm to find and select an optimal cluster center quickly. When the proposed algorithm performed a single pass over the dataset, it worked appropriately for massive datasets; hence, the proposed method is suitable for systems with limited memory.

In summary, OpenMP library is used for parallelization data on multicore CPU, and it uses a scalable model that gives programmers a simple and flexible interface for developing parallel applications for platforms. The reviewed approaches focus on the parallelization of K-means and FCM algorithms on multicore CPU. We investigated that the main idea of the studies is distributing the implementation task on CPU cores to manipulate the data divided into chunks and work simultaneously. Therefore, the authors tried to evaluate the speed and performance by using more cores. More researches should be performed to compare parallelization between using the multicore approaches and the file system libraries such as Hadoop and Spark. The evaluation of performance should be compared according to the increasing number of clusters in the file systems and the increasing cores in the multicore approach.

4.2.2 Studies With Multicore CPUs and Using MPI Library

MPI is the most widely used library for exchanging messages between computers or nodes on a distributed memory environment, and it works with many programming languages such as C, FORTRAN, and Visual Net++. Besides, it can be used in a cluster, local network, or many other applications.

Rahimi et al. [13] improved a parallel FCM algorithm on (Open Source Cluster Application Resource) OSCAR software package and on nine nodes based on the MPI library. Parallel FCM algorithm divided the pixels of the image among the processors in which each processor handled and manipulated with n/p data points. The fuzzy membership function is distributed among the processors and then used for the computation of the membership. They reported that the parallel FCM algorithm could even obtain good liner speed up.

Zhang et al. [45] proposed a parallel K-means clustering algorithm based on MPI called MK-means. Data objects are divided into N objects among processes in which each process has N data objects. Randomly K data points are selected as initial cluster centroids. Each object is assigned to the closest centroid. The new centroids are calculated for each cluster. A final centroid is generated by a merge function and the produced result. A comparison is carried out between sequential K-means and MK-means according to the processing time. They concluded that MK-means is efficient in the clustering on large datasets, and it is relatively portable and stable.

Kwok et al. [46] presented a parallel FCM algorithm based on the Single Program Multiple Data (SPMD) model type with MPI. After performing experiments, they compared parallel FCM with parallel K-means algorithm and concluded that the FCM algorithm has ideal speed-up for large datasets. According to scale-up, the performance of parallel FCM experimentally is proved to be excellent.

Savvas and Sofianidou [47] accomplished a parallelization method of K-means algorithms for analyzing 1D data based on MPI. They explored the possibility of using the MPI to accomplish a popular clustering technique in a parallel manner. The worker node, which works on a parallel environment on different datasets produced the local centroids, and a number of data points are assigned to each one. After this information collected by the master node, the weighted arithmetic mean is applied on the centroid list and the global centroids are found by Equation 11.

$$Ci = \frac{\sum_{x=1}^{y} c_x n_x}{\sum_{x=1}^{y} n_x} \tag{11}$$

Ramesh et al. [48] designed a parallel K-means for analyzing agriculture dataset. They used Java programming as an MPI to cluster the dataset according to the structure of the soil. The topology consists of one master and five slaves. The parallelization process starts with the master node. It reads the data and divides it into N chunks and sends one of them to each slave node, randomly initializes cluster centroids, sends all of the centroids to all slaves and calculates the new cluster centroids by taking the average. Slave nodes receive a chunk of dataset and centroids from the master node and compute the cluster membership of the data points assigned to it, and then send the results back to the master node. They proved that the parallel K-means algorithm has a better performance and time complexity when compared with a traditional sequential K-means algorithm.

Kerdprasop and Kerdprasop [49] implemented parallel K-means and parallel Approximate K-means algorithms based on multicore processors by using MPI in the Erlang language. Experiments were performed on a personal computer with processor speed 1 GHz and 1 GB of memory. Each process handled n/p of data points, n indicated to data points to be divided into p parts which are the approximate size for the portion of data that will be processed by the p independent nodes. The centroid was updated by the master node, and the distance calculation is the responsibility of the slave nodes. The experimental results showed that the speedup is very high in parallel K-means of size between 50,000 to 200,000 points with more than 30% at the dataset.

In another study, Kerdprasop and Kerdprasop [50] carried out a parallel clustering K-means (PKM) and Approximate parallel K-means APKM algorithms by Erlang language based on the MPI scheme on multi-core processors. The experiments were performed by a desktop computer with AMD Athlon 64 X2 Dual-Cores CPU and processor speed with 2.2 GHz and 1 GB of memory. They concluded that the parallel K-means algorithm method speedups the computation time when tested with a multi-core processor. The approximation method produced excellent results in a short period of run time.

Sanpawat and Couch [51] designed the parallelization of the K-means algorithm based on MPI. They designed a master-slave technique with SPMD approach on a network of workstations. The master node process consists of choosing cluster centroids randomly and partitioning the data into subsets, sending each subset to each slave and receiving centroids broadcasted from slaves. Each slave node is responsible for receiving a vector of subset p from the master node. Each slave node calculates its local centroids and broadcasts the mean to every other slave node. The method computes the distance and choses new centroids, and sends them to master node. They used dataset consists of random numbers. They found that the time complexity of sequential K-means is reduced after parallel implementation. Additionally, the speedup of parallel K-means is obtained when N increases and reaches 600,000.

Othman et al. [52] presented a parallel K-means algorithm on DNA dataset based on MPI. They partitioned the RNA dataset on 1, 2,3,4,6 processors and artificial dataset on 1, 2, 4, 6 processors respectively. Each p node deals with n/p data points. Each p node performs distance calculation and centroid update in parallel. The higher speedup was obtained on ribosomal RNA dataset compared to an artificial dataset. They reported that the parallel K-means algorithm performs well when it deals with large datasets.

Joshi [53] proposed a parallel K-means algorithm based on multiprocessor by assuming the SPMD model using MPI. Each processor is manipulated with n/p dataset and the distance calculation between point and centroid is performed in a parallel manner. For initial centroid computation, they used the extension of the Bisecting K-means algorithm. In the study, an optimal level of speedup was not achieved, while the benefits of parallelization were observed.

In summary, the reviewed studies implemented parallel K-means and FCM algorithms based on MPI. In most of the studies, the authors used Master-Slave architecture in which the master divides the task among slaves. Generally, the master node is responsible for initializing, and it calculates new centroids while the slave node performs data point assignment process. They used both real and artificial datasets and used a different number of nodes in each study. The main difference between MPI and OpenMP is that MPI is a programming platform that provides the ability to parallelizing data over a distributed system in which an entire program can be parallelized over a network of computers or nodes over the same network while OpenMP provides the ability to parallelize data over shared memory system such as multicore processors. In MPI, since the nodes are mostly computers, they have their memory layout and their own set of cores while in OpenMP, the cores share memory between each other. The advantage of OpenMP is the communication among nodes is easy and relatively cheap.

4.2.3 Studies with GPU and CUDA

GPUs are programmable logic chips (processor) specialized for rendering images more quickly than a CPU because of its ability to perform parallel processing, which allows it to perform multiple calculations at the same time. Each block of GPU has local and shared memory, and three main memories are texture, constant, and global.

There are three major existing GPU-based K-means algorithms named UVk-means, GPU-minor, and HPk-means. UVk-means copies all the data points to the texture memory, which uses a cache mechanism; this way is useful to avoid the long-time latency of the global memory. GPU-Minor stores all input data in the global memory and load k number of cluster centers to the shared memory. HP-k-means uses shared memory for data that will be read multiple times and uses texture and constant memory to utilize the cache mechanism.

Compute Unified Device Architecture (CUDA) is a programming model and parallel computing platform developed by NVIDIA for GPUs. By the programming with CUDA language, developers can speed up computing applications. In the GPU, the processing of applications performs in synchronal with CPU. It runs the sequential part and GPU computes intensive piece of the application runs of thousands of GPU cores in parallel. Developers can use popular languages such as C++, FORTRAN, C++, MATLAB, and Python.

Al-Ayyoub et al. [14] presented a study of the implementation of a version of parallel FCM named brFCM on knee magnetic resonance imaging (MRI) and lung Computer Tomography (CT) images based on GPU. The brFCM aims to speed up the clustering operation. It consists of two main steps: data reduction and data clustering using the FCM algorithm. The parallelization process using GPU consists of four steps. First, initializing the algorithm variables and centers vector. Second, reducing the dataset. Third, updating the centroid vector, and finally updating the membership matrix. The authors compared the parallel and sequential implementations of brFCM with sequential and parallel implementations of the original FCM. The study concluded that the parallel brFCM on GPU is 2.24 times faster than its CPU implementation and 23.43 times faster than the parallel implementation of the traditional original FCM on GPU.

Sirotkovi et al. [54] presented a parallel image segmentation clustering by K-means algorithm by using CUDA on GPU. The proposed implementation was applied on two different CUDA and GPU new and old generation — image frames with a resolution of 512 × 512 pixels were used as input for the segmentation. The results were compared with the sequential version of the algorithm implemented on the CPU. They showed an improvement of the execution time of GPU compared with the sequential version of the algorithm implemented on CPU.

Baker and Balhaf [55] implemented a segmentation of white blood cell images (microscope images) by CPU, GPU, and then by the integration of CPU and GPU. Three types of color space are used HSI (Hue, Saturation, and Intensity), Lab, and RGB. These are a specific range of colors that can be sensed by the human eyes. K-means algorithm was applied on the images based on CPU and GPU with CUDA. The process of color space conversion from RGB to HIS and K-means implementation was performed by GPU in parallel, and the color extraction was performed on CPU. They concluded that the proposed hybrid approach has a more efficient performance on CPU and GPU, and the execution time is similar when the image size is small. However, GPU showed its efficiency when compared with CPU according to the increasing the size of the image. The study reported that the execution time and accuracy is improved based on the expression given by Equation 12.

$$Improvement = \text{CPUTime}/\text{GPUTime} \tag{12}$$

Fakhi et al. [56] developed a new version of parallel K-means algorithm to large-image segmentation based on a new generation of GPU with CUDA based on (MIMD) architecture. The role of the K-means algorithm in the clustering problem is dividing the problem into independent sub-problems. The process was performed on independent GPU computing units. The initialization of cluster centroids is applied on CPU, and the data assignment process is distributed on GPU's. They made a comparison according to total execution time between CPU based K-means algorithm indicated and GPU-based K-means. They showed that the performance of the GPU is better than the CPU.

Cuomo et al. [57] clustered a large dataset by the K-means algorithm based on GPU. They developed three implementations on GPU: first, they used only one data structure to store the data points and labels. Second, they used two different data structures, and then they modified the way to calculate the distance in the kernel. The study concluded that the parallelization approach gives improvement and excellent results in terms of execution time and speed-up.

Kakooei and Shahhoseini [58] designed an algorithm to find the initial centroids and dynamic center correction method based on GPU. Dynamic center correction is a mechanism to dynamically change the centroid of a cluster in case the current centers are not suitable for standard processing implementation. The speedup and accuracy were evaluated and compared with the previous researches results. They obtained better results than previous algorithms.

Li et al. [59] developed two different strategies for high and low dimensional datasets by using GPU. The first strategy for low dimensional datasets is that the GPU on-chip registers are utilized to minimize the memory access latency. The second strategy for high dimensional datasets is making use of GPU on-chip shared memory with on-chip registers. For low dimensional datasets, an algorithm is designed to exploit GPU on-chip registers to decrease the delay of data access. The authors also made a comparison and presented results of accelerated K-means and other parallel popular K-means implementation such as HPk-means, UVk-means, and GPU-Miner. The experimental results showed that the proposed GPU-based K-means algorithms are three to eight times faster than the best reported GPU-based algorithms.

Bai et al. [60] implemented the K-means algorithm on CPU and GPU based on CUDA and SIMD architecture. The data points assignment and k centroids re-calculation process of traditional K-means were performed in parallel on the GPU. The execution time of clustering processing was evaluated. They concluded that the speed of GPU based K-means is better than CPU based K-means.

Sharma et al. [61] proposed K-means, K-Nearest Neighbors (knn) and Back propagation algorithms separately in parallel on GPUs by using CUDA. They tested timing information. The parallel implementation achieves 75x speed-up in K-means and 146x speed-up in knn algorithms, but in the backpropagation algorithm, it does not hold useful results.

Bhavsar [62] designed a system using parallel K-means clustering algorithm on GPU using CUDA. Two cluster numbers 10 and 20 is used. He proved that when the number of clusters and data points increased, it provided better performance.

Zechner and Granitzer [63] accomplished a parallel K-means clustering algorithm that runs on a hybrid architecture of CPU and GPU. The calculation of the distance from data point to centroids is performed on GPU in parallel, and centroids updates are performed on CPU sequentially based on the results obtained from GPU. The first step is preparing data points and uploading them to the GPU. These data points are transferred only once. Then the role of CPU is labeling the data points and updating cluster centroids. The experimental study on synthetic data was given, and a maximum 14x speed increase was observed.

Farivar et al. [64] carried out three distinct stages operations for CUDA accelerated K-means algorithm. In the first stage: The CUDA hardware is initialized by specifying the suitable host and device memory storage area, and the initial set of centroids are evaluated and loaded the dataset into graphic card memory. In the second stage, each thread of the core is processed a single data point and calculated the distance between the points. Each centroid is computed to assign each data point to the nearest centroid. The data points are re-assigned to the closest centroid and computed the next centroid specification. In the third stage, the data points are relabeled to the nearest centroid and computed the next centroid estimation; this process is executed sequentially in the host. After the implementation of the K-means algorithm by CUDA programming language on an NVIDIA 8600 GT, it concluded that a 13x speed was achieved compared with baseline 3 GHz Intel Pentium.

Bhimani et al. [65] focused on implementing the K-means algorithm in three parallel frameworks (shared memory OpenMP, distributed memory MPI, and CUDA-c). They evaluated the results on small images with (300×300) pixels and large images with (1164×1200) pixels. The study concluded that the shared memory platform with OpenMP performs the best for smaller images while a GPU with CUDA-C performs the rest for larger images.

Kohlhoff et al. [66] designed an efficient version of K-means algorithm. The authors mentioned the apparent limitation of previously published highly performing high-occupancy code to relatively low dimensions is too restrictive for some applications and makes these implementations less useful. Hence, they applied the K-means algorithm without putting limits on the number of dimensions and data points and the number of clusters based on GPU. The feature and genes datasets were employed in the experiments. They witnessed how parallel sorting as an intermediate step helped to balance the speed-up over wide ranges of dimensionality, clusters, and data points. This approach achieved high performance by applying fast parallel sorting dataset based on parallel-prefix-sum with updating step in a subsequent iteration. Finally, the proposed approach on two types of GPU was compared with the implementation of CPU. They concluded that the proposed work on GPUs obtains a speedup of up to 200-fold over CPU reference code.

Gao et al. [67] introduced a parallel hybrid Harmony search (HS) with the K-means algorithm named HKA for documents clustering based on GPU with CUDA. In HKA, the fitness value of each solution vector is specified by the Average Distance of Documents to the Cluster centroid (ADDC). The difference between HS and HKA is adding the K-means algorithm to its steps for performing localized searching. The new solution vector is generated according to the result of the K-means algorithm. The results showed that when the cluster number is large, the implementation of the proposed hybrid algorithm with CUDA is 20 times faster than CPU.

Hooda and Nandal [68] presented parallel K-means algorithm both on CPU Intel Pentium D2.0 GHz and GPU NVidia GeForce 8800. In order to use a smaller number of threads and memory access, the two methods Forgy and Random Partition is used to initialize and random selection of k. The study showed that GPU is 6x faster than the CPU.

Wu and Hong [69] designed an efficient CUDA based reinforced algorithm for K-means clustering algorithm using triangle inequalities. This technique explored the trade-off between memory access coalescing and load balance. Due to the K-means algorithm performing more distance calculations, it may lead to increasing iteration, which can remove from the triangle inequalities method. This technique involved computing and sorting the inter-centroid distance steps before labeling step. In the technique, there are two matrixes ICD and RID. The distance between every two centroids is sorted by a k*k matrix called ICD. RID is k*k matrix representing the nearness of the distances. In the labeling points process, unnecessary calculations can be avoided by looking up the ICD and RID. For general input dataset, they designed a hybrid K-means algorithm. The experimental results proved that the presented hybrid algorithm could enhance the CPU based single threaded traditional K-means algorithm by up to 75x. They concluded that the hybrid CUDA algorithm is scalable.

Nistane and Shende [70] introduced parallel K-means and K-medoids algorithms based on GPU. K-Medoids centroids or medoids is calculated by minimizing the sum of dissimilarities between points labeled to be in a cluster and points determined as the centroids, rather than minimizing the square distance. The sequential and parallel performance of K-means and K-medoids clustering algorithms was compared according to the time elapsed. Speedup for K-means clustering is 1.55 and for K-medoids clustering is 0.91 when the cluster size of both is 10.

Kucukyilmaz [71] presented the parallel K-means algorithm based on shared memory architecture. The centroid calculation was divided into two parts. First, each processor is tried to find partial average square errors for all centroids in a parallel manner; the second step, the partial average results are gathered where the processors synchronize. The experimental results showed that the execution time increases when the size of dataset increases. When the total number of instances is higher than 40.000, the execution time also increases linearly. Due to the overall cost, the performance of the parallel processing cannot obtain the accomplishment of the sequential processing if the number of clusters is less than 10. The study concluded that the parallel implementation performed better than the sequential implementation did in case the total number of attributes is 90 attributes.

Wu et al. [72] implemented an acceleration of performance of GPU for analyzing billions of data points based on the K-means algorithm. They reported that the clustering problem with one billion 2-dimensional data points could be processed in less than 30 seconds per iteration compared with highly optimized CPU version on eight processed about 6 minutes per iteration. Furthermore, GPU-based implementation completed the 50 iterations in 26 minutes.

Shalom et al. [73] accelerated the execution time of FCM on large datasets based on GPU. They used multidimensional yeast gene expression dataset — the initial steps of the algorithm are performed by CPU. The distance calculation and new centroids assignments are performed by GPU. The authors performed the experiments on two different GPU cards, GeForce 8500GT and GeForce 8800GTX. The authors made a comparison between sequential and parallel enforcement. The experimental results showed an up to 140X speed-up on 8800GTX GPU card compared with the CPU enforcement, and the GPU enforcement showed up to 73X speed-up on 8500 GT GPU card.

In summary, according to the reviewed studies, the integration between CPU and GPU gives efficient results. In general, CPU performs centroid updates and data labeling sequentially, while GPU performs distance calculations in parallel. GPU based parallelization is better than CPU based parallelization. Since GPU is generally used for processing graphics and games, it is recommended for image segmentation. Furthermore, CPU with OpenMP performs best for smaller images while a GPU with CUDA performs well for larger images.

5. ENHANCEMENTS AND IMPROVEMENTS ON PARALLEL K-MEANS AND FCM CLUSTERING ALGORITHMS

Due to the sensitivity of the K-means algorithm to the selection of the initial centroids and it performs many iterations to cluster the data, many studies published to improve the mechanism of choosing optimal initial centroids and decreasing the number of iterations of the K-means and FCM clustering algorithms.

5.1 Enhancements of K-Means and FCM With Traditional Methods

Some studies suggest combining algorithms with K-means as a hybrid algorithm such as Canopy, K-means++ algorithms and Slope One, and others resolve the problems by parallelizing it in parallel frameworks such as Hadoop, MapReduce and GPU to increase efficiency.

Rathore and Shukla [74] presented an enhancement of the K-means algorithm by performing a small amount of modification on dataset processing technique. The approach improves the data quality by eliminating the outlier points in datasets using standard division method defined by Equation 13.

$$\text{Threshold} = \text{x} \pm 2\text{SD} \tag{13}$$

where x is the mean and SD is the standard deviation. They used five different datasets. The clustering process was performed by Hadoop and MapReduce framework. The proposed approach was compared with the classical K-means algorithm according to accuracy, error rate, and memory usage. The obtained results showed that the proposed method increases the clustering performance, and it is suitable for large data analysis.

Jinlan et al. [75] proposed refining initial centroids method to decrease the number of iterative procedure and the parallelism of cluster algorithms. The idea behind the refinement process is that the K-means algorithm selected the initial cluster centroids and then chose a small number of random samples of dataset. The selected samples are then being clustered via K-means to re-assign the cluster centers, and then the samples will be re-clustered. The new centroids then considered as the refined initial centroids. After these steps, the complexity analysis was performed. Hence, they reported that the iteration time of the K-means algorithm decreases, and the clustering performance is more efficient.

Swamy et al. [76] improved the K-means algorithm by using the initialization method originated by binary search technique to speedup the execution time of K-means. OpenMP was used as the parallel processing environment. The study concluded that the proposed approach takes less processing time as compared to other methods, and implementing parallel processing decreases the time taken for clustering the dataset.

Liangyu et al. [77] enhanced the random selection of the initial clustering center of K-means algorithm by carrying out Canopy algorithm based on Hadoop and MapReduce framework. The Canopy clustering algorithm is an unsupervised pre-clustering algorithm implemented before the K-means clustering algorithm. It is performed to speed up the clustering in case of studying with large-scale datasets. The study concluded that the data communication time is reduced, therefore the speed-up ratio is increased and the efficiency is enhanced. They concluded that the hybrid Canopy-K-means algorithm enhances clustering accuracy of about 8%.

Xiaojing and Yuanbo [78] proposed an approach to solve the initial centers of the K-means algorithm. Two centroids are selected by the convex hull and the solution of the heel point techniques. The Reuters news dataset was used as the data source. The parallelization of the enhanced algorithm was implemented using MapReduce for parallel processing.

Purohit and Shettar [79] developed a new enhanced K-means algorithm method to find optimal cluster centroids. This method took the average of all data and considered as the first centroid and a point with maximum distance from selected centroid considered as the second centroid — the method was implemented in MapReduce. The performance was evaluated by Silhouette Coefficient metric in Equation 10. The study concluded that the enhanced K-means algorithm achieved better accuracy in cluster formation than the traditional K-means algorithm.

Akthar et al. [80] accomplished a method modification of K-means clustering algorithm by improving the initial center. This method was applied by selecting optimum *k* data points in highly dense areas as initial cluster centers by implementing two techniques. The experiment was performed by using MapReduce and Hadoop framework. They compared proposed techniques by performance measurements including Precision, Recall, Macro Average Precision (MaAP) and Macro Average Recall (MaAR), Micro Average Precision (MiAP) and Micro Average Recall (MiAR), F-measure and execution time given in Equations 14–18, respectively.

$$\boldsymbol{Precision} = \frac{\#\ of\ relevant\ records\ reterived\ in\ a\ cluster}{Total\ \#\ of\ records\ reterived\ in\ that\ same\ a\ cluster} \tag{14}$$

$$\boldsymbol{Recall} = \frac{\#\ of\ relevant\ records\ reterived\ in\ a\ cluster}{Total\ \#\ of\ relevant\ records\ in\ the\ database} \tag{15}$$

$$\boldsymbol{MiAP} = \frac{\sum_{i=1}^{k} Correctly\ reterived\ documments\ in\ cluster\ C_i}{\sum_{i=1}^{k}\ Total\ \#\ of\ reterived\ documments\ in\ cluster\ C_i} \tag{16}$$

$$\boldsymbol{MiAR} = \frac{\sum_{i=1}^{k} Correctly\ reterived\ documments\ in\ cluster\ C_i}{\sum_{i=1}^{k} Total\ \#\ of\ correct\ documments\ of\ cluster\ C_i\ in\ the\ database} \tag{17}$$

$$f_{\boldsymbol{measure}} = \frac{\boldsymbol{2 * P * R}}{\boldsymbol{P + R}} \tag{18}$$

The experimental test was applied on a single system and then on three systems. They concluded that enhanced methods have better precision, average precision/recall, recall, micro/macro, less execution time, and higher f-measure count.

Zhang and Xia [81] implemented a new method to improve the K-means algorithm in choosing initial centroids to avoid the randomness of the K-means algorithm in choosing initial centroid. They set the initial clustering number as $\sqrt{N}$. The experiments were performed on artificial datasets. It showed that the enhanced K-means algorithm is excellent compared with random based K-means algorithm.

Saini et al. [82] carried out an algorithm named DisK-means in parallel based on the MapReduce model and Hadoop. They tried to reduce the number of iterations and selecting initial cluster centroids. The proposed algorithm was compared with the traditional K-means algorithm. The experiments were validated according to the cluster quality and execution time. In the cluster quality, the within-cluster sum of square metric was tested to show the efficiency of the outcomes. The study concluded that the proposed algorithm was reduced time execution when it compared to traditional K-means, and when the *k* is increases execution time decreases.

Lin et al. [83] introduced a hybrid parallel algorithm by combining K-means and Slope One on Hadoop and MapReduce framework. The goal of this study is to improve efficiency and consume less time in recommendation systems. The clustering users process was established on ratings of users by using Slope One algorithm to estimate the ratings of the target user's items. The authors compared a typical sequential hybrid and proposed parallel hybrid recommendation algorithm in terms of time consumption of different clusters. They reported that when the number of Hadoop nodes increases, the time consumption of the parallel algorithm decreases.

Thangarasu and Inbarani [84] proposed a modification of the K-means algorithm named Parallel Rough K-means (MPRK). This algorithm used to cluster large text document dataset. They concluded that the proposed algorithm provides better results and a proper method of clustering with less computational steps.

Dai [85] presented an implementation of the hybrid canopy and FCM algorithm based on Hadoop MapReduce named canopy-FCM algorithm. Due to the time-consuming of FCM in choosing an optimal initial center, the author utilized canopy algorithm to avoid this problem. The Canopy algorithm was used as the initialization of the FCM algorithm. The author parallelized two datasets for the experiment. The precision, recall, and speedup metrics were used to evaluate the quality of the clustering results. A comparison was carried out between canopy-FCM and FCM. They reported that when the number of the nodes in the Hadoop cloud platform is the same, the execution efficiency of the canopy-FCM algorithm is higher than FCM algorithm.

Yu and Ding [86] carried out an enhancement of hybrid FCM and Canopy algorithm to overcome the sensitivity of FCM in choosing initial clustering center. The Canopy algorithm was used to choose centroids. They improved Canopy-FCM algorithm by using max-min principle based on MapReduce to avoid the blindness of the Canopy algorithm. The authors reported that the improved Canopy-FCM algorithm based on MapReduce has better clustering quality and run more quickly than Canopy-FCM. Additionally, FCM algorithm based on MapReduce and also has better speed-up ratio than the Canopy-FCM based on the standalone (one) Hadoop node.

In summary, the reviewed studies focused on enhancing K-means and FCM algorithms. Because of K-means and FCM algorithms has difficulty in finding initial centroids, in order to avoid this problem, the clustering algorithms are integrated with other algorithms as a hybrid approach.

5.2 Enhancements of K-Means and FCM Algorithms With Metaheuristic Algorithms

Most of the studies introduce hybrid algorithms by combining clustering algorithms with metaheuristic algorithms such as Artificial Bee Colony (ABC), Particle Swarm Optimization (PSO), ant colony optimization (ACO), Firefly and genetic algorithms (GE). These hybrid algorithms are generally used for finding optimal centroids to improve the efficiency and scalability of the clustering algorithms.

Mathew and Vijayakumar [87] enhanced the K-means algorithm by utilizing the ability of the Firefly algorithm in finding initial centroids. The algorithm assigns each data point to clusters by using standard cluster validation techniques. They focused on finding optimal cluster centroids using the global optimum found by Firefly algorithm. They compared the modified parallel K-means algorithm (MPKM) with parallel K-means (PKM) and concluded that MPKM performance is better than PKM. The performance of MPKM was evaluated by classification error percentage (CEP) metric, which is defined by Equation 19.

$$\mathrm{CEP} = \frac{m}{n} * 100 \tag{19}$$

The effectiveness of the proposed algorithm was evaluated by metrics and evaluation Indexes by Accuracy, Davies-Bouldin (DB), Between Sum Squares (BSS) and WCSS Equations 20–23, respectively.

$$Accuracy = \frac{Number\ of\ Correct\ predictions}{Total\ od\ all\ cases\ to\ be\ predicted} \tag{20}$$

$$DB = \frac{1}{n}\sum_{i=1}^{n} \max\left\{\frac{S_n\left(Q_i\right) + S_n\left(Q_j\right)}{S_n\left(Q_i, Q_j\right.}\right\} \tag{21}$$

$$BSS = \sum_{i=1}^{k} |C_i|\left(m - m_i\right)^2 \tag{22}$$

$$WCSS = \sum_{j=1}^{k}\sum_{i=1}^{n}\left\|x_i^j - C_i\right\|^2 \tag{23}$$

Wang et al. [88] proposed a combination of PSO and parallel K-means algorithm named K-PSO implemented on Hadoop and MapReduce. The K-means algorithm utilized the PSO to improve its global search capability and increase its ability to process large amounts of data. They compared K-PSO with parallel and sequential K-means. The parallel K-PSO and parallel K-means efficiency are close to the performance of the sequential K-means, which is incompatible with what they initially expected. A possible reason is that the datasets have not divided into multiple splits.

Chaturbhuj and Chaudhary [89] implemented a combination of PSO with the K-means algorithm as a hybrid method to make K-means efficient and effective by solving its problem in selecting optimal initial centroids. The approach was implemented on Hadoop and MapReduce framework. In the PSO algorithm, the position of each particle represents k centroids. When PSO algorithm reached the maximum number of iterations, the current global best position chosen as the optimal initial centroids. They reported that the proposed algorithm is efficient and effective in clustering large scale of data.

M A and Abdul Nazeer [90] presented a combination of PSO with parallel K-means algorithm as a hybrid method to analyze large scale datasets by using Hadoop and MapReduce framework. The proposed method found the initial centroids of K-means and removed outliers from datasets using the convex hull approach. The authors evaluated the performance of their study according to the Sum of Square ratio (SS) Equation 24.

$$\mathrm{SS} = \mathrm{BSS}/\mathrm{TSS} \tag{24}$$

where TSS is defined by Equation 25:

$$\mathrm{TSS} = \mathrm{BSS} + \mathrm{WCSS}\ (\mathrm{Constant}) \tag{25}$$

They concluded that when the iteration number reduced, the performance of the proposed hybrid method increased compared with the enhanced parallel K-means clustering algorithm with MapReduce model (IPKCA).

In another study of Mathew and Vijayakumar [91] proposed a hyper method based on K-means and firefly algorithms. They used Firefly to choose initial centroids of parallel K-means algorithm and applied the proposed method on four datasets. They evaluated the proposed hybrid method with the metrics:accuracy defined by Equation 20, Davies-Bould in index defined by Equation 21, BSS defined by Equation 22, WCSS defined by Equation 23, Dunn index (DI) defined by Equation 26, and Silhouette Coefficient metrics defined by Equation 13. The authors compared the modified parallel algorithm MPKM with parallel K-means PKA and calculated the speedup as given in Equation (1).

$$DI = \min_{1 \le i \ge n}\left\{\min_{1 \le j \le n, i=j}\left\{\frac{d\left(c_i, c_i\right)}{\max\left(d'\left(c_k\right)\right)_{1 \le k \le n}}\right\}\right\} \tag{26}$$

Niknam and Amiri [92] accomplished an effective hybrid algorithm established on K-means PSO and ACO algorithm to cluster analysis named FAPSO-ACO-K. The PSO and ACO algorithms were utilized to determine K-means's optimal centroids. The algorithm was carried out and tested on several datasets. From the experimental results, the study concluded that the performance of the presented algorithm is better than the other stochastic algorithms such as ACO, GA, PSO, SA, and ABC.

Bhavani et al. [93] introduced a hybrid algorithm combining ACO and DE with the K-means algorithm named parallel K-means-DE-ACO and DE-K-means. The proposed approach aims to cluster genome sequence on Hadoop and MapReduce framework. The approach consists of three steps: storage of DNA sequences of genomes, feature signifier extraction for all genomes using MapReduce and clustering genome sequences by hybrid DE-K-means and DE-ACO-K-means algorithms. The study concluded that according to the speed of convergence and accuracy results, DE-ACO-K-means clustering algorithm converges faster and more accurate than DE-K-means algorithm.

Xiufeng and Changzheng [94] proposed a hybrid method by combining K-means and a pseudo-parallel GA algorithm (PPGA). Multiple populations can run by parallel GA algorithm on a single processor so that the method divided the initial population into several subpopulations. The subpopulations were improved independently according to the traditional process of GA and using of chromosome retreading. The proposed approach was examined in the analysis of big data. The study concluded that the proposed algorithm has higher convergence, accuracy, and speed. They proved that it is a feasible and efficient clustering algorithm.

Al-Shboul and Myaeng [95] carried out a hybrid algorithm combining K-means with GA algorithm to solve the K-means problem in selecting optimal initial centroids and to avoid its convergence to local minima. They proposed two approaches, GA initializes KM (GAIK), and KM initializes GA algorithm (KIGA). A comparative research study was carried out among the GA-based Clustering Algorithm (GCA), GAIK, KIGA, and FCM algorithms. In GKIK approach, GA algorithm was executed to start initial value of K-means algorithm to minimize the iteration numbers of K-means. In the KIGA approach, K-means was utilized to initialize the GA algorithm. The study concluded that KIGA is better and achieves high clustering accuracy than other algorithms.

Ahmadyfard and Modares [96] offered a hybrid algorithm combining PSO with the K-means algorithm named PSO-KM to solve K-means initialization problem by utilizing the ability of PSO's convergence to global minima. The study concluded that the proposed algorithm achieved good results than K-means and PSO algorithms.

Armano and Farmani [97] implemented a combination of K-means and ABC algorithms as a hybrid approach named k-ABC. The approach tried to improve the ability of the K-means algorithm in finding global optimum clusters by taking advantages of the ABC algorithm. The study concluded that the presented algorithm could find a global optimum cluster.

Karol and Mangat [98] proposed a hybrid algorithm combining PSO with K-means and FCM named KPSO and FCPSO, respectively. The proposed approach was tested on two real datasets. In general, the combined method starts with initializing the clusters and centroids generated by clustering algorithms, and then PSO runs on the clusters to obtain globally optimum clusters. The authors compared the proposed approaches and with traditional K-means and FCM. By using the F-measure metric, they reported that KPSO and FCPSO give more valid results as compared to all other tested algorithms on both datasets and also reported that FCPSO algorithm gives even better results than KPSO algorithm.

Izakian and Abraham [99] presented a hybrid approach by combining FCM and fuzzy PSO named FCM-FPSO. They utilized PSO to avoid the sensitivity of initial centroids of FCM algorithm. The authors applied the proposed technique on five real datasets. They reported that the hybrid FCM–FPSO algorithm obtained better results compared to FPSO and FCM algorithms in all of the datasets and showed that it could avoid getting stuck in local optima.

In this section, we reviewed combined approaches with metaheuristic algorithms and clustering algorithms. More study may be performed on the Spark environment. Because it has the ability to perform processes faster than Hadoop since spark processes the data in the memory, rather than a hard drive. The reviewed works showed that heuristic algorithms could help clustering algorithms in the initialization of the centroids.

6. APPLICATIONS BASED ON PARALLEL K-MEANS AND FCM CLUSTERING ALGORITHMS

Aitali et al. [15] proposed a parallel FCM algorithm with bias field correction named PBC-FCM for MRI brain image segmentation. The approach was implemented on GPU cards GT740m, gtx760, and gtx580, respectively. The MRI images were partitioned into pixels over the GPU, and each one is processed by one thread. The proposed algorithm starts by initializing the variables and centroids vector and transferring the data from CPU to GPU. One of two kernels calculates the membership function while the second kernel one calculates the conjectural bias field. The membership function calculation and centroid supdating are performed in GPU. The vector of cluster centers is calculated in CPU, and then transferred to GPU. The study reported that the GPU-based hardware is suitable for image segmentation, and GTX580 architecture provides high speedup compared to the others.

Zhang et al. [100] accomplished a parallel K-means clustering algorithm based on master and slave model. The computing environment is built by network stations and PVM for communication between nodes. The master node distributes dataset and produces new centroids, and the slave node allocates the data and performs a message transfer routine of PVM that is used to realize the relocation of the data among processors. The study concluded that the real acceleration ratio was produced and computing time was larger than communication time.

Rashmi et al. [101] carried out a parallel block approach based on parallel K-means algorithm to boost in computational time for processing the remote sensing images with more than 1000×1000 pixel dimension. The parallel block operation divided the input image into sub-images as two, four, eight blocks. Block processing approach was applied in parallel by distributing the task among the cores. The study concluded that when the number of cores increases, the proposed work gets more efficient and also the speed up increases with a rise in the size of an image pixel dimension.

Jian et al. [102] implemented a parallel K-means algorithm on MapReduce in Hadoop framework to analyze big data produced from customers' responses. WebCrawler was used to gather information from the internet, and then Hadoop was equipped with cloud computing servers for distributed computing. A neural network was employed to command the server to read the reviews and words like "Awesome", "Nice", and "Great". K-means algorithm was used to sort and group similar words and updates the database. Finally, the result was visualized as a Graphical User Interface (GUI) to customers. The study's outcomes do reading and understanding the reviews easy, and the intelligent learning system allows the user to understand and realize the goodness of the service easily.

Kraj et al. [103] presented a parallel K-means clustering algorithm in a high-performance multi threaded application (ParaKMeans). This software consists of the GUI, the application programming interface (API), and the Parallel Cluster web service. The software is designed for clustering genes in a laboratory in a modular way to provide both deployment flexibility in the user interface as well as it works as a client-server model. The experimental results showed that the speed of

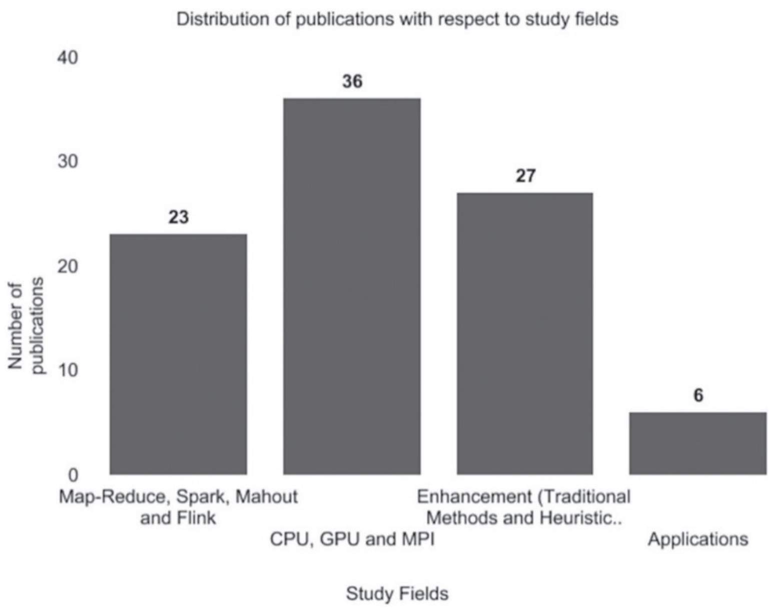

Figure 7 Distribution of publication with respect to study fields.

execution ParaKMeans increases when multiple computer nodes used.

Michopoulou et al. [104] accomplished a segmentation method of both normal and unfastened lumbar intervertebral discs of the spine from MRI images by FCM. In order to perform semi-automatic atlas-based disk segmentation, they used three different versions of the FCM algorithm. These versions were (atlas-fact, atlas-robust-fcm, (atlas-RFCM) and elastic-atlas-RFCM). For quantification and characterization of disc unfastened severity, mentioned approaches are used for utilizing the computer-aided diagnosis system and could be helped in computer-assisted spine surgery. The steps of the proposed approach are the registration of the image by the rigid landmark-based technique and the integration with the FCM algorithm. For each pixel of the disc image, the tissue class membership values are calculated. RFCM is an extension of FCM that forces the membership values to be like neighboring values. The RFCM produced spatially smooth membership functions. The proposed approach assumed a locally related and globally smooth transformation. The authors reported that better time efficiency and segmentation accuracy was provided by the atlas-RFCM approach.

In summary, K-means and FCM algorithms are commonly used in various applications. As we reviewed in this section, they are used in remote sensing, hotel management, and customer recommendation systems, gens clustering, and medical applications. Since K-means and FCM algorithms are easy to apply and implement, they still have potential in many research fields related to data analysis.

7. CONCLUSION

Due to the inefficiency and time consumption of sequential algorithms in analyzing large-scale datasets, it is essential to use parallel implementations of clustering algorithms in which the data is divided into smaller blocks, and each is assigned to a computing node. In the literature, some parallelization studies have been performed on K-means and FCM algorithms to improve their efficiency. In this study, we carried out a systematic and detailed review to obtain relevant literature on the implementations of parallel K-means and FCM algorithms. The number of reviewed papers is approximately 92, which are published between 2000 and 2018. As shown in Figure 7, among 92 papers reviewed; 23 papers are on the implementation of parallel K-means, and FCM algorithms on Hadoop, MapReduce, Spark, Mahout, and Flink frameworks. 20, 10, and 6 papers are implemented on GPU, MPI, and CPU, respectively.

There are 27 papers are about enhancing and improving parallel K-means and FCM algorithms. Among them, 13 papers are related to the enhancements by metaheuristic algorithms, and 14 of them are about improvements by traditional techniques, and the last 6 papers are about applications of parallel K-means and FCM algorithms. The distribution of publications concerning years is shown in Figure 8. The number of papers tends to increase year by year. Additionally, we summarized and classified the reviewed studies according to utilized algorithms and its purpose and application, as is shown in Table 1.

It was observed that the implementation of parallel K-means and FCM clustering algorithm in various frameworks achieves excellent performance in analyzing large-scale datasets. From Figure 7, it can be stated that the most popular research field in parallel K-means and FCM clustering algorithms are parallelizations on CPU, GPU-based hardware, and using MPI library. Since Spark is a new platform, the number of studies on this platform is limited. Additionally, it is seen that since the K-means algorithm cannot find optimal initial value, most studies have attempted to enhance and improve the K-means algorithm by combining it with different techniques and other algorithms to improve the optimal value and global search capability. It should be noted that there are several shortcomings which are open and can be studied in this field. Among them is that there is a limited number of researches regarding the Spark and Mahout parallel platforms which have high efficiency in solving machine learning problems. Enhancing the efficiency of the K-means and FCM algorithms regarding their execution times are not at the level of expected. Hence, new and more powerful techniques

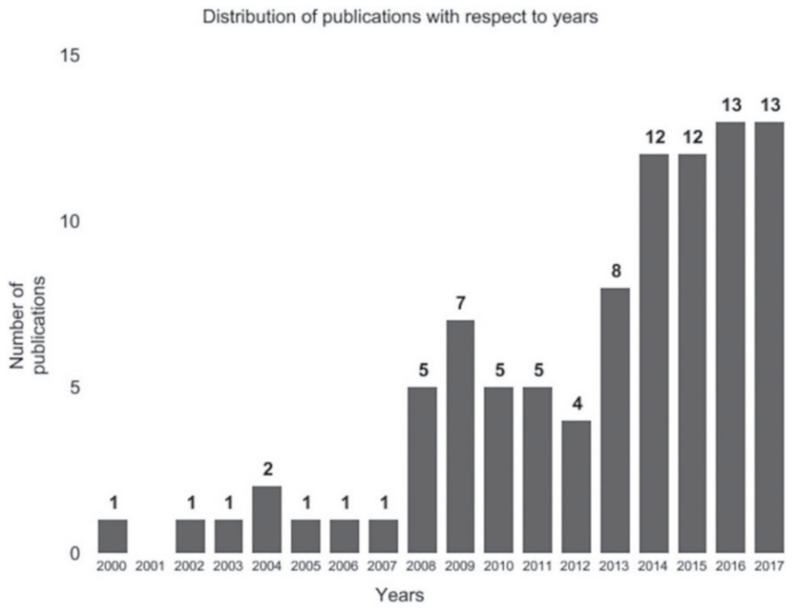

Figure 8 Distribution of publications with respect to years.

Table 1 Categorical view of the algorithms and their applications.

Algorithm	Purpose and Application	Publication
Parallel K-means based on MapReduce	Enhancement parallel K-means algorithm performance.	Moertini & Venica (2016) [19]
	Analyze big data.	Zhao et al. (2009) [20]
	Solve the traffic problem of subarea roads in Beijing caused by large scale taxi movements.	Xia et al. (2015) [21]
	Improvement of parallel K-means algorithm.	Liao et al. (2013) [22]
	Implement parallel K-means algorithm on cloud computing.	Liu & Cheng (2012) [23]
	The parallel K-means algorithm implemented on a distributed network with one master node and seven slave nodes.	Anchalia (2013) [26]
	Implement a sequential and parallel K-means algorithm on image file.	Lv et al. (2010) [11]
	Cluster large-scale Chinese commodity information web.	Yushui & Lishuo (2015) [27]
	Implementing parallel K-means on spatial data.	Zhong & Liu (2016) [28]
	Cluster Moroccan users in the social network Twitter.	Abdouli et al. (2017) [29]
	Implement parallel K-means algorithm on cloud computing.	Hao & Ying (2011) [30]
	Implement a parallel representation of K-means.	Bandyopadhyay et al. (2017) [31]
	Test the performance of parallel K-means algorithm.	Kang & PARK (2015) [32]
Parallel K-Medoids algorithm based on MapReduce	Acceleration of K-Medoids algorithm.	Jiang & Zhang (2014) [24]
Parallel K-means + FCM algorithms based on MapReduce	Implementing Parallel K-means and FCM algorithms on land price dataset in Taichung City.	Lin et al. (2017) [25]

Table 1 continued

Algorithm	Application	Publication
	Implement the parallel FCM algorithm on five different datasets with different sizes based on multi-node Hadoop cluster and MapReduce using Amazon Elastic Cloud Computing (Amazon EC2).	Garg & Trivedi (2014) [17]
Parallel Canopy, K-means and FCM algorithm based on MapReduce	Implementing Canopy, FCM, and K-means algorithms on Mahout and Hadoop framework.	Hai (2017) [8]
Parallel K-means based on Spark	Implementing a parallel representation of K-means.	Wang et al. (2016) [34]
	Intelligent K-means algorithm.	Kusuma et al. (2016) [35]
	Implement versions of Word Count and Sort applications.	Manzi & Tompkins (2016) [38]
Parallel K-means and FCM based on Mahout over Hadoop Platform	Categorization of twitter users.	Jain & S K Jain (2014a) [39]
	Clustering similar Twitter Users.	Jain & S K Jain (2014b) [36]
Parallel K-means based on Flink platform	Clustering context of E-commerce datasets.	Li et al. (2017) [37]
Parallel K-means based on Multicore CPU	Clustering several databases with different classes and featured and an RGB image of peppers.	Baydoun et al. (2016) [40]
	Enhancement parallel K-means algorithm performance.	Naik et al. (2013) [41]
	Analyze billions of data points.	Wu et al. (2009) [72]
	Testing the speedup factor.	Chu et al. (2007) [43]
Parallel K-means + shift mean algorithms based multicore CPU	Clustering Two datasets.	Wang et al. (2008) [42]
Parallel K-means, k-mean++ based multicore CPU	Clustering large datasets.	Hadian & Shahrivari (2014) [44]
Parallel K-means based on MPI	Implementing parallel and sequential versions on dataset.	Zhang et al. (2011) [45]
	Analyzing 1D data.	Savvas & Sofianido (2014) [47]
	Analyzing agriculture dataset.	Ramesh et al. (2010) [48]
	Design parallel clustering K-means PKM and Approximate parallel K-means APKM algorithms by Erlang language.	Kerdprasop & Kerdprasop (2010,a) [49]
	Designi parallel clustering K-means PKM and Approximate parallel K-means APKM algorithms by Erlang language.	Kerdprasop & Kerdprasop (2010,b) [50]
	Improving time complexity speedup of parallel K-means is increase when N increases.	Kantabutra & Couch (2000) [51]
	Implementing parallel K-means algorithm on DNA dataset.	Othman et al. (2004) [52]
	Implementing on three datasets.	Joshi (2013) [53]
Parallel FCM based on MPI	Implementing on large-scale dataset.	Kwok et al. (2002) [46]
	Improved a parallel FCM algorithm on OSCAR (Open Source Cluster Application Resource) software package.	Rahimi et al. (2004) [13]
Parallel K-means based on GPU	Image segmentation.	Sirotkovic et al. (2012) [54]
	White blood cell image segmentation.	Baker & Balhaf (2016) [55]
	Segmentation large-image.	Fakhi et al. (2017) [56]
	Clustering large dataset.	Cuomo et al. (2017) [57]
	Finding initial centroid and dynamic center correction.	Kakooei & Shahhoseini (2014) [58]

Table 1 continued

Algorithm	Application	Publication
	Speeding up of parallel K-means algorithm.	Li & et al. (2013) [59]
	Implementing K-means algorithm in CPU and GPUs.	Hong-tao et al. (2009) [60]
	Implementing with CUDA programming language and evaluating K-means performance.	Bhavsar (2017) [62]
	Parallel K-means in hybrid based on CPU and GPU.	Zechner &Granitzer (2009) [63]
	Acceleration of the K-means algorithm.	Farivar et al. (2008) [64]
	Implementing the K-means algorithm in three parallel frameworks (shared memory Open-MP, distributed memory MPI, and CUDA-c).	Bhimani et al. (2015) [65]
	Designed an efficient version of K-means algorithm without putting limits on the dimensions number, data points and a number of clusters.	Kohlhoff et al. (2013) [66]
	Document clustering.	Gao et al. (2012) [67]
	The implementation of the proposed hybrid algorithm with CUDA is 20 times faster than CPU.	Hooda1 & Nandal (2014) [68]
	Designing an efficient CUDA-based reinforced algorithm for K-means clustering using triangle inequalities.	Wu and Hong (2011) [69]
	Implementing a K-means algorithm in parallel based on shared memory architecture.	Kucukyilmaz (2014) [71]
Parallel FCM	Version of FCM named brFCM on lung CT and knee MRI images based on GPU in parallel.	Al-Ayyoub et al. (2015) [14]
	Multidimensional yeast gene expression dataset.	Shalom et al. (2008) [73]
	MRI brain image segmentation.	Kraj et al. (2008) [103]
FCM	Lumbar intervertebral discs image segmentation.	Michopoulou et al. (2009) [104]
Parallel K-means + knn + Back-propagation algorithms	Implementing with CUDA programming language.	Sharma et al. (2016) [61]
Parallel K-means + k-mediods based on GPU	Implementing parallel K-means and k-mediods.	Nistane & S. Shende (2013) [70]
Parallel K-means + Firefly	Enhancing Parallel K-means.	Mathew & Vijayakumar (2014) [91]
	Clustering data, Improving clustering accuracy by optimization of centroid.	Mathew & Vijayakumar (2015) [87]
Parallel K-means + PSO	Improve initial centroid of K-means.	Wang et al. (2012) [88]
	Solving K-means algorithm's problem in selecting optimal initial centroids.	Chaturbhuj & Chaudhary (2016) [89]
	Improve initial centroid of K-means and evaluate the result by Sum of Square SS measurement.	M A & K. Abdul Nazeer (2017) [90]
	Solving K-means initialization problem.	Ahmadyfard & Modares (2008) [96]
K-means + FCM + PSO	Clustering text document.	Karol & Magnat (2013) [98]
FCM + PSO	Solving FCM initialization problem.	Izakian & Abraham (2011) [99]

Table 1 continued

Algorithm	Application	Publication
K-means + ACO + PSO	Solving choosing optimal centroid problem.	Niknam &Amiri (2010) [92]
Parallel K-means + ACO + DE	Clustering DNA genome sequence.	Bhavani et al. (2010) [93]
K-means + GE algorithm	Solving the inefficiency of K-means algorithm in choosing optimal initial centroid.	Xiufeng & Changzheng (2010) [94]
	Solving K-means problem in selecting optimal initial centroids.	Al-Shboul, & Myaeng (2009) [95]
K-means + Artificial Bee Colony	Improve the ability of K-means algorithm in finding global optimum.	Armano & Farmani (2014) [97]
Parallel K-means algorithm	Enhancement of the K-means algorithm performance by performing a small amount of modification on dataset.	Rathore & Shukla (2015) [74]
	Refining of K-means algorithm and parallelize it.	Jinlan et al. (2005) [75]
	Improving K-means algorithm by using initialization method originated by using binary search technique and to speedup execution time of K-means.	Swamy et al. (2015) [76]
	Modification of K-means clustering algorithm by improving initial center.	Akthar et al. (2016) [80]
	Clustering large text document dataset.	Thangarasu & Inbarani (2016) [84]
	Parallel K-means algorithm and compare the efficiency with sequential version.	Zhang et al. (2006) [100]
	Implemented parallel block approach to solve problem according to increasing in computational time for processing the remote sensing images for pixel dimension more than 1000×1000.	Rashmi et al (2016) [101]
	Big data analysis in hotel customer response.	Jian et al. (2017) [102]
	Implementation of Parallel K-means algorithm suitable for general laboratory use.	Aitali et al. (2016) [15]
Parallel K-means + Canopy algorithm	Improve the random selection of initial clustering center K-means algorithm by Canopy algorithm.	Liangyu et al. (2015) [77]
Parallel FCM + Canopy	Utilize canopy algorithm to generate cluster center of FCM.	Dai (2016) [85]
	Used Canopy algorithm's ability in choosing cluster center.	Yu & Ding (2015) [86]
Parallel K-means + Slope One algorithms	Improving the efficiency and process less time consuming in recommendation system algorithm.	Lin et al. (2014) [83]
K-means algorithm	Solving initial centers of K-means algorithm.	Xiaojing & Yuanbo (2017) [78]
	Finding optimal cluster centroid.	V. Purohit & Shettar (2017) [79]
	Enhancement of parallel K-means algorithm for clustering very large datasets.	Boukhdhir et al. (2016) [33]
	Improving initial canter of K-means algorithm.	Chen & Shixiong (2009) [81]
	Reducing the number of iterations and selecting initial cluster centroids	Saini et al. (2016) [82]

and tools to increase efficiency and decrease the execution time are required in this field.

Furthermore, regarding the datasets, clustering non-linearly separated datasets are not stated in the reviewed papers. Therefore, there is a gap on clustering non-linearly big datasets by K-means or kernel K-means, which is our future work. Additionally, a survey on the other clustering algorithms can be future work as well.

REFERENCES

1. Zaki MJ (2000). Parallel and Distributed Data Mining: An Introduction. *Springer-Verlag Berlin Heidelb* 1–23. doi: 10.1007/3-540-46502-2_1.
2. Setiawan A, Budhi GS, Setiabudi DH, Djunaidy R (2017). Data Mining Applications for Sales Information System Using Market Basket Analysis on Stationery Company. In: *2017 International Conference on Soft Computing, Intelligent System and Information Technology (ICSIIT)*. pp 337–340.
3. Paul S (2011). Parallel and Distributed Data Mining. *InTech.*
4. Xiao H (2010). Towards parallel and distributed computing in large-scale data mining: A survey. *Parallel Comput* 1–30.
5. Paul S (2010). An Optimized Distributed Association Rule Mining Algorithm in Parallel and Distributed Data Mining with XML Data for Improved Response Time. *Int J Comput Sci Inf Technol*. doi: 10.5121/ijcsit.2010.2208.
6. Çelik A, Özmen A (2009). *A Comparative Study On Distributed Parallel Systems: Pvm And Mpi*. In: 5. Uluslararası Ýleri Teknolojiler Sempozyumu. pp 13–15.
7. Zamanifar K, Nematbakhsh N, Sadjady RS (2010). *A new load balancing algorithm in parallel computing*. 2nd Int Conf Commun Softw Networks, ICCSN 2010 449–453. doi: 10.1109/ICCSN.2010.27.
8. Hai M (2017). *A Performance Comparison Study of Parallel Clustering Algorithms in Cluster Environments*. In: IEEE 2nd International Conference on Big Data Analysis. pp 307–311.
9. Flynn MJ (1966). *Very High-Speed Computing Systems*. In: Proceedings of the IEEE. pp 1901–1909.
10. Anantathanavit M, Munlin M (2015). Using K-means radius particle swarm optimization for the travelling salesman problem. *IETE Tech Rev (Institution Electron Telecommun Eng India)* 33:172–180. doi: 101080/02564602.2015.1057770.
11. Lv Z, Hu Y, Zhong H, Wu J (2010). Parallel K-Means Clustering of Remote Sensing Images Based on MapReduce. *WISM 2010*, LNCS 6318 162–170.
12. Golghate AA, Shende SW (2014). Parallel K-Means Clustering Based on Hadoop and Hama. *IJCAT Int J Comput Technol* 1:33–37.
13. Rahimi S, Zargham M, Thakre A, Chhillar D (2004). *A Parallel Fuzzy C-Mean algorithm for Image Segmentation*. IEEE Annu Meet Fuzzy Information, NAFIPS 04 234–237.
14. Al-Ayyoub M, Abu-Dalo AM, Jararweh Y, et al (2015). A GPU-based implementations of the fuzzy C-means algorithms for medical image segmentation. *J Supercomput* 71:3149–3162. doi: 10.1007/s11227-015-1431-y.
15. Aitali N, Cherradi B, El Abbassi A, et al (2016). Parallel Implementation of Bias Field Correction Fuzzy C-Means Algorithm for Image Segmentation. *Int J Adv Comput Sci Appl* 7:375–383.
16. Choudhry MS, Kapoor R (2016). Performance Analysis of Fuzzy C-Means Clustering Methods for MRI Image Segmentation. *Procedia Comput Sci* 89:749–758. doi: 101016/j.procs.2016.06.052.
17. Garg D, Trivedi K (2014). *Fuzzy K-mean clustering in MapReduce on cloud based hadoop*. Proc 2014 IEEE Int Conf Adv Commun Control Comput Technol ICACCCT 2014 1607–1610. doi: 10.1109/ICACCCT.2014.7019379.
18. Hurwitz J, Nugent A, Halper F, Kaufman M (2013). *Big data for dummies.*
19. Moertini V, Venica L (2016). *Enhancing Parallel k-Means Using Map Reduce for Discovering Knowledge from Big Data*. IEEE Int Conf Cloud Compuing Big Data Anal 81–87.
20. Zhao W, Ma H, He Q (2009). Parallel K -Means Clustering Based on MapReduce. *CloudCom* 674–679.
21. Xia D, Wang B, Li Y, et al (2015). An Efficient MapReduce-Based Parallel Clustering Algorithm for Distributed Traffic Subarea Division. *Hindawi Publ Corp Discret Dyn Nat Soc.*
22. Liao Q, Yang F, Zhao J (2013). *An Improved parallel K-means Clustering Algorithm with MapReduce*. In: ICCT. pp 764–768.
23. Liu S, Cheng Y (2012). *Research on K-Means Algorithm Based on Cloud Computing*. In: International Conference on Computer Science and Service System. pp 1762–1765.
24. Jiang Y, Zhang J (2014). *Parallel K-Medoids Clustering Algorithm Based on Hadoop*. In: IEEE 5th International Conference on Software Engineering and Service Science.
25. Lin C, Liu J, Peng T (2017). *Performance Evaluation of Cluster Algorithms for Big Data Analysis on Cloud*. In: 2017 IEEE International Conference on Applied System Innovation IEEE-ICASI. pp 1434–1437.
26. Anchalia PP, K Koundinya A, Srinath NK (2013). *MapReduce Design of K-Means Clustering Algorithm*. In: International Conference on Information Science and Applications (ICISA).
27. Yushui G, Lishou Z (2015). *K-means Clustering Algorithm for Large-scale Chinese Commodity Information Web Based on Hadoop*. In: 14th International Symposium on Distributed Computing and Applications for Business Engineering and Science. pp 256–259.
28. Zhong Y, Liu D (2016). *The Application of K-Means Clustering Algorithm Based on Hadoop.* In: IEEE International Conference on Cloud Computing and Big Data Analysis. pp 88–92.
29. Abdouli AEL, Hassouni L, Anoun H (2017). Mining Tweets of Moroccan Users using the Framework Hadoop, NLP, K-means and Basemap. In: *Intelligent Systems and Computer Vision (ISCV).*
30. Hao C, Ying Q (2011). *Research of Cloud Computing based on the Hadoop platform*. In: International Conference on Computational and Information Sciences. pp 181–184.
31. Bandyopadhyay SS, Halder AK, Chatterjee P, et al (2017). *HdK-means: Hadoop based parallel K-means clustering for big data*. In: 2017 IEEE Calcutta Conference (CALCON). pp 452–456.
32. Kang Y, Park YB (2015). *The performance evaluation of k-means by two MapReduce frameworks, Hadoop vs. Twister*. In: International Conference on Information Networking (ICOIN). pp 405–406.
33. Boukhdhir A, Lachiheb O, Salah Gouider M (2015). *An improved MapReduce Design of Kmeans for clustering very large datasets*. In: IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA).
34. Wang B, Yin J, Hua Q, et al (2016). *Parallelizing K-Means-Based Clustering on Spark*. In: International Conference on Advanced Cloud and Big Data. pp 31–36.
35. Kusuma I, Ma'Sum MA, Habibie N, et al (2016). *Design of intelligent k-means based on spark for big data clustering*. *2016 Int Work Big Data Inf Secur IWBIS* 2016 89–95. doi: 101109/IWBIS.2016.7872895.
36. Jain E, K Jain S (2014). *Using Mahout for clustering similar Twitter Users Performance Evaluation of K-Means and its*

comparison with Fuzzy K-Means. In: 2014 5th International Conference on Computer and Communication Technology (ICCCT). pp 29–33.

37. Li C, Tian G, Cai K (2017). *Improved K-means based on flink platform and its application in e-commerce big data*. 2017 Chinese Autom Congr 7261–7264. doi: 101109/CAC.2017.8244089.
38. Manzi D, Tompkins D (2016). *Exploring GPU acceleration of apache spark*. In: Proceedings - 2016 IEEE International Conference on Cloud Engineering. pp 222–223.
39. Jain E, K Jain S (2014). *Categorizing twitter users on the basis of their interests using hadoop/mahout platform*. In: 9th International Conference on Industrial and Information Systems, ICIIS. pp 1–5.
40. Baydoun M, Dawi M, Ghaziri H (2016). *Enhanced Parallel Implementation of the K-Means Clustering Algorithm*. 3rd Int Conf Adv Comput Tools Eng Appl 7–11.
41. Naik DSB, Kumar SD, Ramakrishna S V (2013). *Parallel Processing Of Enhanced K-Means Using OpenMP*. In: 2013 IEEE International Conference on Computational Intelligence and Computing Research. pp 1–4.
42. Honggang W, Jide Z, Hongguang L, Jianguo W (2008). *Parallel clustering algorithms for image processing on multi-core CPUs*. In: International Conference on Computer Science and Software Engineering, ICSSE. pp 450–453.
43. Chu C-T, Kim SK, Lin Y-A, et al (2007). MapReduce for Machine Learning on Multicore. *Adv Neural Inf Process Syst* 19 281–288. doi: 10.1234/12345678.
44. Hadian A, Shahrivari S (2014). High performance parallel k-means clustering for disk-resident datasets on multi-core CPUs. *J Supercomput* 69:845–863. doi: 10.1007/s11227-014-1185-y.
45. Zhang J, Wu G, Hu X, et al (2011). *A parallel K-means clustering algorithm with MPI*. In: 2011 4th International Symposium on Parallel Architectures, Algorithms and Programming, PAAP 2011. pp 60–64.
46. Kwok T, Smith K, Lozano S, Taniar D (2002). *Parallel Fuzzy c-Means Clustering for Large Data Sets*. In: European Conference on Parallel Processing. Springer, Berlin, Heidelberg, pp 365–374.
47. Savvas IK, Sofianidou GN (2014). *Parallelizing K-means algorithm for 1-D data using MPI*. Proc Work Enabling Technol Infrastruct Collab Enterp WETICE 179–184. doi: 10.1109/WETICE.2014.13.
48. Ramesh V, Ramar K, Babu S (2013). Parallel K-Means Algorithm on Agricultural Databases. *Int J Comput Sci* 10:710–713.
49. Kerdprasop K, Kerdprasop N (2010). A lightweight method to parallel k-means clustering. *Int J Math Comput Simul 4*.
50. Kerdprasop K, Kerdprasop N (2010). *Parallelization of K-Means Clustering on Multi-core Processors*. ACS'10 Proc 10th WSEAS Int Conf Appl Comput Sci 472–477.
51. Sanpawat K, Couch AL (2000). Parallel K-means Clustering Algorithm on NOWs. *Tech J* 1:243–248.
52. Othman F, Abdullah R, Nur'Aini AR, Abdul Salam R (2004). *Parallel k-means clustering algorithm on DNA dataset*. In: International Conference on Parallel and Distributed Computing: Applications and Technologies. Springer, Berlin, Heidelberg, pp 248–251.
53. Joshi MN (2003). Parallel K- Means Algorithm on Distributed Memory Multiprocessors. *Comput 9*.
54. Sirotkovi J, Dujmi H, Papi V (2012). *K-means image segmentation on massively parallel GPU architecture*. MIPRO, 2012 Proc 35th Int Conv 489–494.
55. Bani Baker QB, Balhaf K (2017). *Exploiting GPUs to accelerate white blood cells segmentation in microscopic blood images*. In: 2017 8th International Conference on Information and Communication Systems, ICICS 2017. pp 136–140.
56. Fakhi H, Bouattane O, Youssfi M, Hassan O (2017). New optimized GPU version of the k-means algorithm for large-sized image segmentation. *Intell Syst Comput Vision, ISCV 2017*. doi: 101109/ISACV.2017.8054924.
57. Cuomo S, De Angelis V, Farina G, et al (2017). A GPU-accelerated parallel K-means algorithm. *Comput Electr Eng* 1–13. doi: 10.1016/j.compeleceng.2017.12.002.
58. Kakooei M, Shahhoseini HS (2014). *A parallel k-means clustering initial center selection and dynamic center correction on GPU*. In: 22nd Iranian Conference on Electrical Engineering, ICEE 2014.
59. Li Y, Zhao K, Chu X, Liu J (2013). Speeding up k-Means algorithm by GPUs. *J Comput Syst Sci* 79:216–229. doi: 10.1016/j.jcss.2012.05.004.
60. Hong Tao B, Li Li H, Dan Tong O, et al (2009). *K-means on commodity GPUs with CUDA*. 2009 WRI World Congr Comput Sci Inf Eng CSIE 3:651–655. doi: 10.1109/CSIE.2009.491.
61. Sharma R, Vinutha M, Moharir M (2016). *Revolutionizing Machine Learning Algorithms using GPUs*. In: International Conference on Computational Systems and Information Systems for Sustainable Solutions. pp 318–323.
62. Bhavsar HA (2017). Parallelizing K-Means Clustering Using GPU. *IJARIIE* 3:1451–1457.
63. Zechner M, Granitzer M (2009). *Accelerating k-means on the graphics processor via CUDA*. In: The 1st International Conference on Intensive Applications and Services, INTENSIVE 2009. pp 7–15.
64. Farivar R, Rebolledo D, Chan E, Campbell R (2008). A parallel implementation of k-means clustering on GPUs. *Pdpta* 13:212–312.
65. Bhimani J, Leeser M, Mi N (2015). *Accelerating K-Means clustering with parallel implementations and GPU computing*. In: 2015 IEEE High Performance Extreme Computing Conference, HPEC. pp 1–6.
66. Kohlhoff KJ, Pande VS, Altman RB (2013). K -means for parallel architectures using all-prefix-sum sorting and updating steps. *IEEE Trans Parallel Distrib Syst* 24:1602–1612.
67. Gao Z, Li E, Jiang Y (2012). *A Gpu-Based Harmony K-Means Algorithm For Document Clustering*. In: IET International Conference on Information Science and Control Engineering. Shenzhen, pp 2–5.
68. Hooda H, Nandal R (2014). Implementation of k-Means Clustering Algorithm in CUDA. *Int J Enhanc Res Manag Comput App*l 3:829–833.
69. Wu J, Hong B (2011). *An efficient k-means algorithm on CUDA*. In: IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum. pp 1740–1749.
70. Nistane KD, Shende SW (2013). GPU Accelerated Clustering Techniques. *Int J Sci Res.*
71. Kucukyilmaz T (2014). Parallel K-Means Algorithm for Shared Memory Multiprocessors. *J Comput Commun* 2:15–23.
72. Wu R, Zhang B, Hsu M (2009). Clustering billions of data points using GPUs. In: *the combined workshops on UnConventional high performance computing workshop plus memory access workshop, ACM*. pp 1–5.
73. Shalom S, Dash M, Tue M (2008). *Graphics hardware based efficient and scalable fuzzy c-means clustering*. In: Proceedings of the 7th Australasian Data. pp 179–186.
74. Rathore P, Shukla D (2015). *Analysis and performance improvement of K-means clustering in big data environment*. In: 2015 International Conference on Communication Networks (ICCN). pp 43–46.
75. Jinlan T, Lin ZHU, Suqin Z, Lu L (2005). Improvement and Parallelism of k-Means Clustering Algorithm. *Tsinghua Sci Technol* 10:277–281.

76. Swamy P, Raghuwanshi MM, Gholghate A (2015). *An Improved Approach for k-Means Using Parallel Processing*. In: 2015 International Conference on Computing Communication Control and Automation. pp 358–361.
77. Liangyu D, Dongping X, Zhenzhen L (2015). *The Improvement and Implementation of Clustering Algorithm Based on Multi-core Computing*. In: IEEE 14 International conf. on Cognitive Informatics & Cognitive Computing. pp 405–411.
78. Xiaojing W, Yuanbo L (2017). *Research on Improved K-Means Algorithm Based on Hadoop*. In: 4th International Conference on Information Science and Control Engineering. pp 593–598.
79. V. Purohit B, Shettar R (2015). *A MapReduce framework to implement Enhanced K- means algorithm*. In: International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT). pp 361–363.
80. Akthar N, Ahamad MV, Ahmad S (2016). *MapReduce Model of Improved K-Means Clustering Algorithm Using Hadoop MapReduce*. In: Second International Conference on Computational Intelligence & Communication Technology. pp 192–198.
81. Chen Z, Shixiong X (2009). *K-means clustering algorithm with improved initial center.* In: 2nd International Workshop on Knowledge Discovery and Data Mining. pp 790–792.
82. Saini A, Minocha J, Ubriani J, Sharma D (2016). *New Approach for Clustering of Big Data: DisK-Means*. In: International Conference on Computing, Communication and Automation (ICCCA2016). pp 122–126.
83. Lin K, Wang J, Wang M (2014). *A Hybrid Recommendation Algorithm Based on Hadoop*. In: The 9th International Conference on Computer Science & Education (ICCSE 2014).
84. Thangarasu M, Inbarani HH (2016). *MPRK algorithm for clustering the large text datasets*. In: IEEE International Conference on Advances in Computer Applications, ICACA. pp 224–229.
85. Dai W (2016). An Improved Hybrid Canopy-Fuzzy C-Means Clustering Algorithm Based on MapReduce Model. *J Comput Sci Eng* 10:1–8.
86. Yu Q, Ding Z (2015). *An improved parallel Fuzzy C-Means algorithm based on MapReduce.* In: 8th International Conference on BioMedical Engineering and Informatics. pp 634–638.
87. Mathew J, Vijayakumar R (2015). *Enhancement of Parallel K-Means Algorithm.* In: IEEE Sponsored 2nd International Conference on Innovations in Information, Embedded and Communication systems (ICJJECS).
88. Wang J, Yuan D, Jiang M (2012). *Parallel K-PSO Based on MapReduce*. In: IEEE 14th International Conference on Communication Technology. pp 1203–1208.
89. Chaturbhuj KS, Chaudhary G (2016). *Parallel Clustering of large data set on Hadoop using Data mining techniques*. In: World Conference on Futuristic Trends in Research and Innovation for Social Welfare (WCFTR16).
90. M A A, Abdul Nazeer KA (2017). *Improved Parallel Clustering with Optimal Initial Centroids.* In: International Conference on Recent Advances in Electronics and Communication Technology. pp 114–120.
91. Mathew J, Vijayakumar R (2014). *Scalable Parallel Clustering Approach for Large Data Using Parallel K Means and Firefly Algorithms.* In: International Conference on High Performance Computing and Applications (ICHPCA). pp 1–8.
92. Niknam T, Amiri B (2010). An efficient hybrid approach based on PSO, ACO and k -means for cluster analysis. *Appl Soft Comput* 10:183–197. doi: 101016/j.asoc.2009.07.001.
93. Bhavani R, Sudha Sadasivam G, Kumaran R (2011). *A Novel Parallel Hybrid K-means-DE-ACO Clustering Approach for Genomic Clustering using MapReduce.* World Congr Inf Commun Technol 132–137.
94. Xiufeng G, Changzheng X (2010). *K-means Multiple Clustering Research Based on Pseudo Parallel Genetic Algorithm.* In: International Forum on Information Technology and Applications. pp 30–33.
95. Al-Shboul B, Myaeng S-H (2009). I*nitializing K-means using genetic algorithms*. In: International Conference on Computational Intelligence and Cognitive Informatics (ICCICI 09). pp 114–118.
96. Ahmadyfard A, Modares H (2008). *Combining PSO and k-means to enhance data clustering.* 2008 Int Symp Telecommun 688–691. doi: 10.1109/ISTEL.2008.4651388.
97. Armano G, Farmani MR (2014). Clustering Analysis with Combination of Artificial Bee Colony Algorithm and k-Means Technique. *Int J Comput Theory Eng* 6:141–145. doi: 107763/ijcte.2014.v6.852.
98. Karol S, Mangat V (2013). Evaluation of text document clustering approach based on particle swarm optimization. *Open Comput Sci* 3:69–90. doi: 102478/s13537-013-0104-2.
99. Izakian H, Abraham A (2011). Fuzzy C-means and fuzzy swarm for fuzzy clustering problem. *Expert Syst Appl* 38:1835–1838. doi: 10.1016/j.eswa.2010.07.112.
100. Zhang Y, Xiong Z, Mao J, Ou L (2006). *The study of parallel k-means algorithm.* In: 6th World Congress on. Dalian, China, pp 5868–5871.
101. Rashmi C, Chaluvaiah S, Kumar GH (2016). An Efficient Parallel Block Processing Approach for K -Means Algorithm for High Resolution Orthoimagery Satellite Images. *Procedia Comput Sci* 89:623–631. doi: 10.1016/j.procs.2016.06.025.
102. Jian M-S, Fang Y-C, Wang Y-K, Cheng C (2017). *Big data analysis in hotel customer response and evaluation based on cloud.* In: 19th International Conference on Advanced Communication Technology (ICACT). pp 791–795.
103. Kraj P, Sharma A, Garge N, et al (2008). ParaKMeans: Implementation of a parallelized K-means algorithm suitable for general laboratory use. *BMC Bioinformatics* 9:1–13. doi: 10.1186/1471-2105-9-200.
104. Michopoulou SK, Costaridou L, Panagiotopoulos E, et al (2009). Atlas-based segmentation of degenerated lumbar intervertebral discs from MR images of the spine. *IEEE Trans Biomed Eng* 56:2225–2231. doi: 10.1109/TBME.2009.2019765.