An Efficient Parallel Algorithm for Clustering Big Data based on the Spark Framework

Zineb Dafir*
Faculty of Science of Rabat, Mohammed V University
Rabat, Morocco

Said Slaoui
Faculty of Science of Rabat, Mohammed V University
Rabat, Morocco

***Abstract*—The principal objective of this paper is to provide a parallel implementation focused on the main steps of the parameter-free clustering algorithm based on K-means (PFK-means) using the Spark framework and a machine learning-based model to process Big Data. Thus, the process consists of parallelizing the main tasks of the first stage of the PFK-means clustering algorithm using successive RDD functions. Then, the parallel K-means provided by Spark MLlib is invoked by setting the cluster centers and the number of clusters determined in the previous step as input parameters of the parallel K-means. Furthermore, a comparison between the parallel designed algorithm and the parallel K-means was conducted using UCI data sets in terms of the sum of squared errors and the processing time. The experimental results, performed locally using the Spark framework, demonstrate the efficiency of the proposed solution.**

***Keywords*—Clustering; big data; spark; parallel computing; parallel K-means**

I. Introduction

Today's digital world is experiencing unprecedented progress, causing the generation of an immense amount of data presented in various formats and derived from many sources, including social networks, the internet of things, mobile apps, multimedia, financial services, and ERP systems. Hence, managing this huge amount of data requires efficient techniques and tools to ensure that the data is processed efficiently to make the right decision according to the application domain [1]. Researchers and scientists are constantly contributing to and developing new machine learning algorithms, as well as designing more efficient frameworks capable of dealing with the continuous flow of data generated each lapse of time [2] [3].

Cluster analysis is one of the appropriate data mining techniques for ensuring efficient Big Data processing [4] [5]. Indeed, the purpose of cluster analysis is to place similar data objects in the same group to construct disjoint clusters. The concept of similarity is firmly based on a distance measure depending on the characteristics of the data object's features. In the same context, several Big Data clustering algorithms were designed based on specific computing platforms aiming to either distribute the clustering tasks on several nodes to perform the necessary calculations in a parallel way or by using a server with high capacities in terms of CPUs, memories, and I/O resources [6] [7] [8].

This paper proposes a new parallel clustering algorithm based on Spark [9], designed to enhance the sequential PFK-means algorithm to leverage distributed systems and process big data. As a result, the proposed parallel implementation is carried out using open-source Hadoop, the Spark framework with RDDs, and a machine learning-based model. More specifically, the proposed algorithm aims to parallelize the two principal stages of the PFK-means algorithm using Spark RDDs functions. The first one consists of applying successive RDDs functions in order to perform the computation of the Euclidean distances, build the initial clusters, and finally update the different clusters to obtain the cluster centers and, consequently, the number k. The second stage invokes the use of the parallel K-means provided by Spark mLlib with as input parameters the cluster centers and the number of clusters discovered in the previous phase. In that respect, a comparison between the developed parallel algorithm and the parallel K-means provided by Spark mLlib was conducted on some UCI data sets based on the sum of the squared error measure and the processing time. The results obtained show that the suggested solution yields more efficient clustering compared to the parallel K-means with the random initialization mode. The main contributions of this paper are as follows:

- Improve the approach to determining the initial cluster centers of the cluster construction step
- Suggest a parallel implementation of the main steps of the PFK-means clustering algorithm based on spark RDDs in order to process big quantitative data.
- Establish a comparison between the designed algorithm and the parallel K-means algorithm provided by spark mLlib.

The remainder of this paper is organized as follows: Section II discusses a literature review. Section III gives a short presentation of parallel computing using the Spark framework. Section IV provides the design and implementation of the parallel developed algorithm. Section V shows the results of the experiment. Finally, Section VI concludes the paper and presents future perspectives.

II. Literature Review

Over the years, clustering algorithms have undergone several evolutions and improvements to face the challenges of managing heterogeneous and complex data generated from multiple and varied digital sources. In particular, K-means [10] [11] is one of the most widely used clustering algorithms due to its simplicity and efficiency in processing data. This algorithm has experienced several adaptations and improvements

*Corresponding authors.

to address initialization issues, exploit distributed systems and handle Big Data [12] [13] [14]. In this context, several research studies based on different parallel computing architectures have been carried out to give birth to powerful and scalable algorithms.

Among these algorithms, a parallel adaptive Canopy-K-means algorithm [15] aims to solve the manual selection problem for the distance threshold T2 in the Canopy process. In this regard, it is improved by adaptive parameter estimation using the MapReduce-based Model and the Spark framework. The clustering results on the Stanford Large Network Dataset Collection (SNAP) data set and self-built Dimension demonstrate the efficiency of the proposed solution.

Another improvement in the parallel K-means is introduced in [16], which is an improved K-Means Clustering Algorithm for Big Data Mining based on the density of data points to construct the corresponding clusters. Thus, the algorithm was parallelized on the Hadoop platform using the distributed database to improve its efficiency and decrease the running time. The simulation results prove that the enhanced solution outperforms the classical K-means and the DBSCAN algorithm in terms of clustering accuracy by 10%.

In the same context, a clustering center selection method [17] was developed to solve the issue of the random nature and limited quality of initial cluster center selection. In addition, a parallel implementation using the Spark computing framework was suggested to perform Big Data clustering and therefore obtain higher execution efficiency, accuracy, and good stability in big data.

More recently, another improved initialization method [18] for the K-means algorithm was implemented to enhance the initial points selection strategy using sparse reconstruction. Moreover, the parallel version of the algorithm was performed based on the MapReduce framework and Hadoop cluster using the real customer data from the JD Mall. The experimental results demonstrate its efficiency in processing large amounts of data.

Another contribution aiming to address the K-means algorithm initialization issue is a parallel clustering algorithm based on grid density [19]. Indeed, the process of the algorithm is based on specific strategies, including, locality sensitive hash function(DP-LSH), the adaptative grouping strategy (AGS), and the MapReduce framework to operate the cluster centers in parallel and therefore increase the performance of the proposed approach.

III. Parallel Computing using Spark Framework

Parallel computing is one of the efficient solutions to process large amounts of data using computing platforms. Indeed, these platforms allow the distribution of the calculations according to different architectures [20]. Such platforms can be either horizontally scaled platforms or vertically scaled platforms [7]. In particular, Spark is a horizontally scalable cluster computing framework [9], performing parallel processing for data-intensive applications with working sets. These applications are managed by the driver program, which allows them to execute the user's main function as well as the parallel operations in a cluster. Hence, it has the advantage

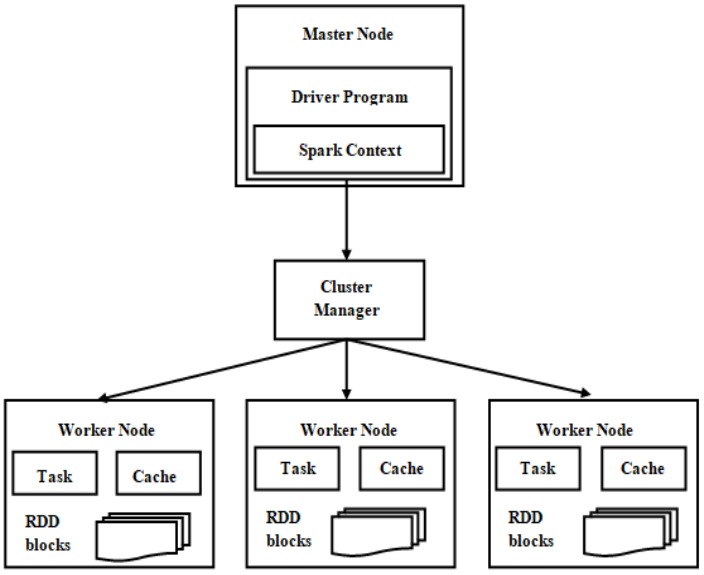

Fig. 1. Spark Architecture.

of executing tasks by ensuring locality-aware scheduling, fault tolerance, and load balancing [9]. The spark is distinct from MapReduce since it handles iterative jobs and enables users to run queries on a Big Data set by loading into memory only the required data set. More specifically, the Spark framework is structured around two main abstractions: resilient distributed dataset (RDD) and shared variables. The RDD is a collection of items distributed through the cluster nodes that can persist in memory to be reused in the event of parallel operations. The shared variables are also used in parallel operations to share the necessary variables among jobs or between jobs and the driver program. In addition, Spark is powerful and outperforms the Hadoop MapReduce framework by 10× in interactive machine learning workloads. Fig. 1highlights the basics of spark architecture.

IV. The Proposed Parallel PFK-means

This section first introduces an overview of the PFK-means clustering algorithm for processing quantitative data, then gives a detailed description of the implementation of the parallel algorithm designed for Big Data processing, and finally presents the algorithm describing the various Spark RDD functions that allow the parallelization of the tasks of the suggested algorithm.

A. The Sequential PFK-means

The PFK-means algorithm [21] is a parameter-free clustering algorithm aiming to construct progressively homogeneous clusters until the appropriate number of clusters is automatically detected. This heuristic is a combination of the E-transitive heuristic [22] adjusted for quantitative data, and the traditional K-means [10] [11]. Indeed, the sequential version of the PFK-means algorithm performs a partitioning clustering based on two main stages. The first one aims to construct clusters successively based on the similarity between the cluster centers and the data objects using the Euclidean

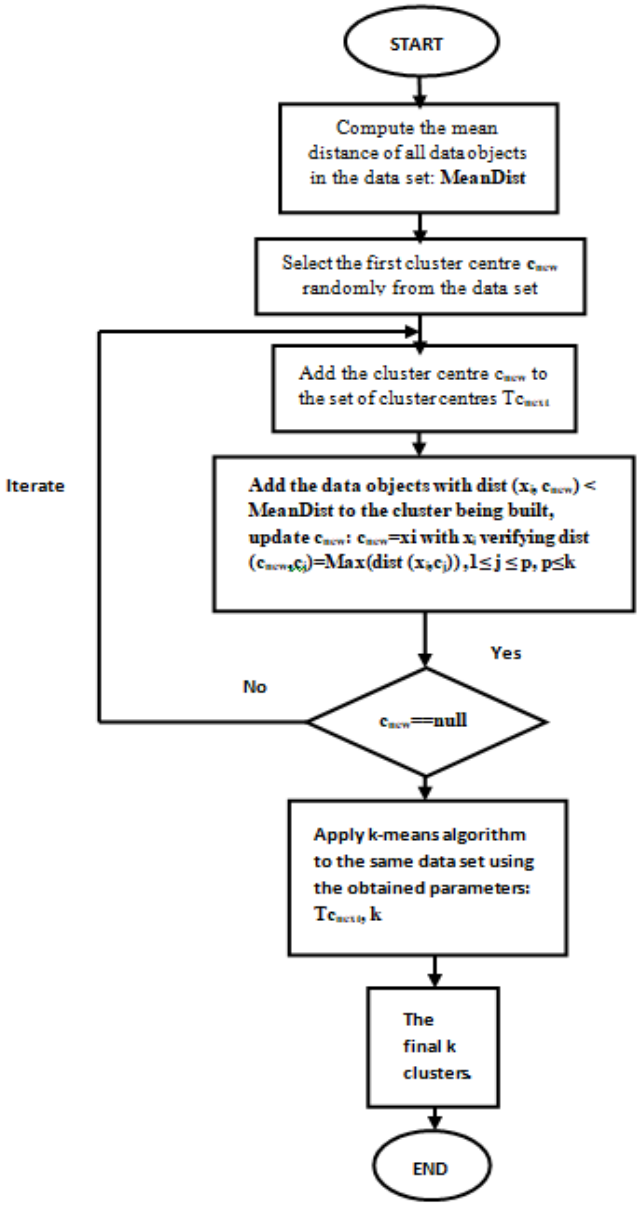

Fig. 2. Sequential PFK-means Flowchart.

distance calculation and consequently, gets the initial cluster centers and the number of clusters k, which are the two primary input parameters for the K-means algorithm. The second stage consists of applying the K-means algorithm to the same data set taking into consideration the cluster centers obtained and the number of clusters k reached in the first stage. The first stage of the PFK-means algorithm is an independent clustering algorithm that enables the discovery of clusters with appropriate cluster centers. The K-means algorithm is applied to enhance the quality of the clusters obtained. Fig. 2 resumes the main steps of the PFK-means algorithm.

B. Design and Implementation of the Parallel PFK-means

The PFK-means is a sequential and iterative clustering algorithm, suitable for processing unsupervised machine learning data sets. However, the efficiency of this algorithm will be

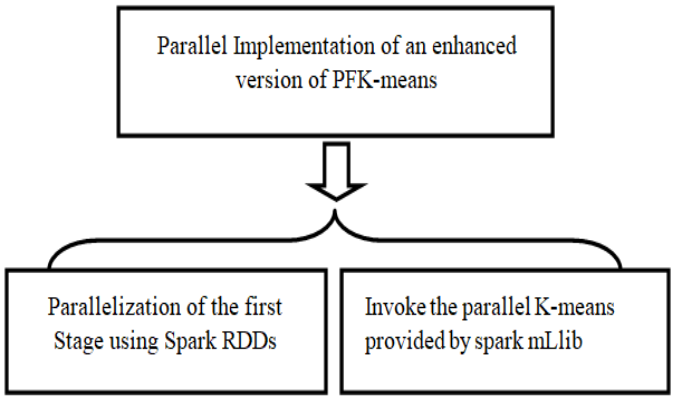

Fig. 3. The Design of the Parallel Implementation.

restricted when dealing with a large amount of data. Among the solutions to improve the performance of this algorithm and make it suitable for processing Big Data is the parallelization of its tasks using the open-source Hadoop, Spark RDD, and Machine Learning-based Model. In that respect, the implementation of the parallel version of PFK-means consists of the parallelization of its two main stages. In this way, the first stage consists of the parallelization of three principal tasks, including the calculation of the total average of the Euclidean distances, the construction of the initial clusters, and assigning the data objects to the appropriate clusters as well as updating the discovered cluster centers. The next stage involves the use of the parallel K-means provided by spark mLlib, using as input parameters the discovered cluster centers as well as the number of clusters automatically detected in the first stage. Fig. 3 illustrates the design of the parallel implementation of the PFK-means. In the following, the steps of the initial stage will be thoroughly described.

1) Calculation of the Total Average of the Euclidean distances: In order to compute the total average of the Euclidean distances, the process starts by reading the data set being processed locally into RDDs partitions since it is the first step. Then these RDDs' partitions are transformed using the Map function with the purpose of calculating the Euclidean distance between each two data objects, which is performed in several iterations. Therefore, in each iteration, the Reduce function sums the partial Euclidean distances for each partition Map. Then, after completing all the iterations, another Reduce function is applied to the final result, in order to obtain the total average of the Euclidean distances. Fig. 4 depicts the process of spark RDDs to compute the total average Euclidean distance of the processed data set.

Although the use of Spark is optimal since it avoids storing the data on a hard disk, the calculation of the average of the Euclidean distances of the data set is costly. It is, therefore, essential to optimize this calculation. The first way involves calculating the average of the Euclidean distances of a sample instead of processing the entire data set using the Map and Reduce functions. The second solution consists of first reordering the whole data set randomly and then applying the MapPartitions function, which allows decomposing the data

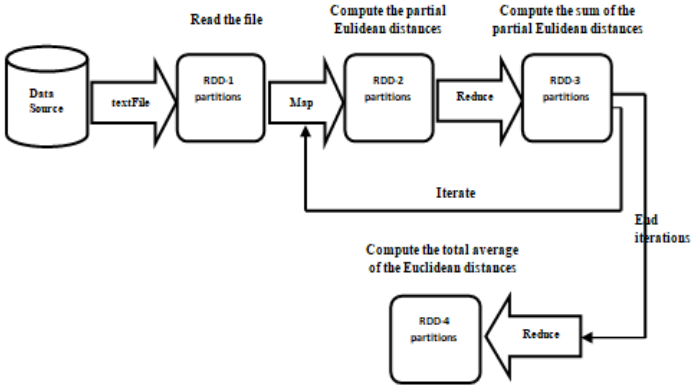

Fig. 4. The Parallel Process of the Average of Euclidean Distance Calculation.

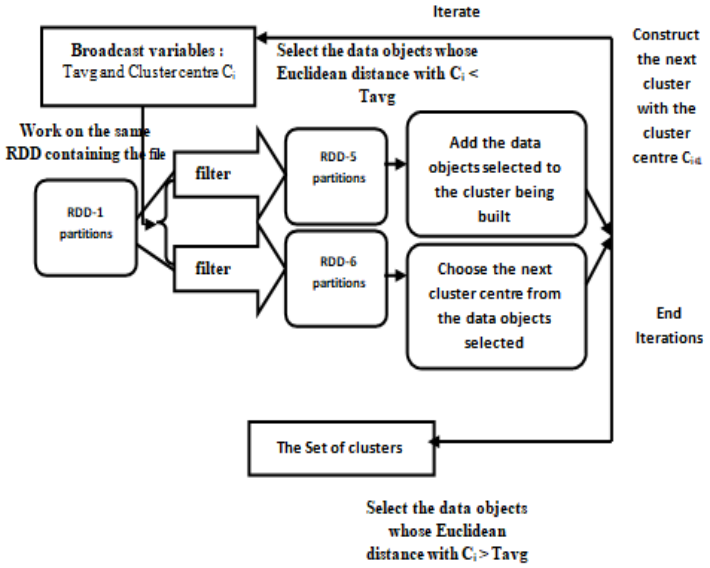

Fig. 5. The Parallel Process of Initial Cluster Construction.

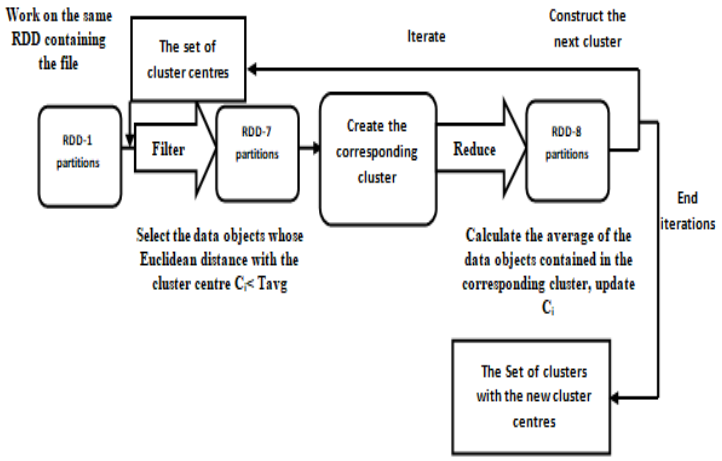

Fig. 6. The Parallel Process of Updating Clusters.

set into a specific number of partitions. Then, each partition performs the computation of the two-by-two Euclidean distances of a part of the data. Thus, the result produced by the MapPartitions function is transmitted to the Reducer, allowing the calculation of the total average of the Euclidean distances. In this solution, the MapPartitions and Reduce functions are invoked only once, so the calculation time is much reduced compared to the solution explained previously.

2) The Construction of the Initial Clusters: This step starts by choosing the first data object in the data set as the first cluster center. Then, the data objects similar to this cluster center are gathered in such a way that the Euclidean distance between this cluster center and each data object in the whole data set is less than the total average of the Euclidean distances calculated in the previous step. Contrary to the sequential version of the algorithm in which the construction of the first cluster is realized in n iterations, the construction of the first cluster in the parallel version of the algorithm is done using the filter function, which selects the elements similar to the cluster center by calculating the Euclidean distances in parallel. Subsequently, the cluster center of the second cluster is determined in such a way that the data object similar to the first cluster center whose Euclidean distance is the greatest is compared to the data objects not similar to the center, so the data object whose Euclidean distance yields an average value is chosen as the second cluster center. Consequently, the determination of the cluster center is realized using the filter function, which allows selecting the cluster center among the objects not similar to the first cluster center by calculating the Euclidean distances in parallel. In this way, the other cluster centers are determined successively to follow the same process. This process is finished when all the data objects are classified and there are no more data objects to choose from as a new cluster center. And lastly, it is important to mention that during the construction of the clusters, each data object can be placed in several clusters, and therefore the clusters achieved at the end of this step are overlapped. This allows for the enlargement of the size of the clusters and, accordingly, improves the quality of the cluster centers obtained. Fig. 5 shows the cluster construction process using the Spark RDDs functions.

3) Assigning the Data Objects to the Appropriate Clusters and Update Cluster Centers: After having built the initial clusters in the previous step, this step consists of assigning to each cluster center found in the previous step the data objects similar to it by using the filter function. This function selects the data objects by performing the calculations of the Euclidean distances in parallel. Then, the Reduce function is applied to the RDD returned by the Filter function to update the value of the cluster center by computing the average of the data objects contained in the corresponding cluster. In this step, each data object can be associated with only one cluster, and consequently, there will be no overlaps in the resultant clusters. Fig. 6 describes the process of obtaining the final hard clusters with the new cluster centers using the Filter and Reduce functions. Once the hard clusters with updated cluster centers and the number of cluster centers detected automatically have been obtained, the next step is to apply

the parallel K-means provided by Spark mLlib with the input parameters obtained from the first stage, which are the cluster centers and the number of clusters.

C. The Parallel Proposed Algorithm

The following pseudo-code (Algorithm 1) describes the stages of the parallel implementation of the PFK-means algorithm. Hence, the process begins by reading the file containing the set of quantitative data (lines 1 and 2). Subsequently, it seeks to calculate the average of the Euclidean distances of the data set using one of the previously explained methods (line 3). The next step is to build the initial clusters to obtain the list of cluster centers to be used in the next step (from 4 to 17). Afterward, the process aims to assign the data objects to the appropriate centers and update the different centers (from 18 to 24). The last step intends to invoke the parallel K-means provided by Spark mLlib using the parameters obtained from the previous step (from 25 to 27).

Algorithm 1 The Parallel Implementation of PFK-means

Input:A set of n data objects X
Output:The initial cluster centers Tc_{next}. The number of clusters automatically computed
begin

1: ReadFile = sc.textFile("the file path")
2: dtSet = ReadFile.map(lines).cache()
3: TotalAverage = ComputeTotalAverage()
4: sc.broadcast($TotalAverage$)
5: $NextCenter = newList[0]$, $Next = True$
6: **while** $Next$ **do**
7: sc.broadcast($NextCenter$)
8: Cluster = dtSet.filter($dist(X, NextCenter) < TotalAverage$)
9: NextC= dtSet.filter($dist(X, NextCenter) > TotalAverage$)
10: SortedList = NextC.sortBy($dist(X, FarthestElement)$)
11: $NextCenter = MeanValue(SortedList.collect())$
12: Add the constructed Cluster to Tc_{next}
13: **if** $NextCenter == Null$ **then**
14: $Next == False$
15: **end if**
16: **end while**
17: Save the set of centers in $Centers$
18: $i \leftarrow 0$
19: **while** $i < ClustersSize$ **do**
20: Cluster = dtSet.filter($dist(X, Centers[i]) < TotalAverage$)
21: UpdateCenter=Cluster.reduce($computeMean(X, Y)$)
22: Remove the filtred elements from the dataset dtSet
23: Save the new cluster center in $NewCenters$
24: **end while**
25: t_{max} = $maxIterations$,dt = $dtSet$,k = $NewCentersSize$
26: $initializationMode = KMeansModel(NewCenters)$
27: FinalC=KMeans.train($dtSet$,k,t_{max},$initializationMode$)
end begin

TABLE I. DESCRIPTION OF THE REAL-LIFE DATA SETS

data set	Data size	Attributes	Cluster number
Iris	150	4	3
Wine	178	13	3
Pima Indian Diabetes	768	8	2
Letter-Recognition(LR)	20000	16	26

V. EXPERIMENTS

For the purpose of implementing the parallel version of PFK-means with Spark RDDs in stand-alone mode, a machine of 12GB memory and 1TB hard disk was configured with Spark (version 3.0.3) and Hadoop (2.7 version) and PyCharm IDE by installing useful Pyspark libraries (PySpark version is 3.9). Subsequently, the proposed algorithm was evaluated on real-world data sets extracted from the UCI machine learning repository based on the sum of squared errors and the execution time to measure the clustering performance. In addition, the parallel PFK-means were compared with the parallel K-means provided by the mLlib library to demonstrate the efficiency of the developed solution. It is also crucial to notice that the parallel implementation of PFK-means has been tested locally in the PyCharm IDE and, therefore, the size of the data sets has been limited.

A. Data Sets Description

Table I provides a short description of three real-world datasets, extracted from the UCI machine learning repository [23], serving to validate the efficiency of the parallel implementation of PFK-means, including Iris, Wine, and Letter Recognition (LR). In this way, each data set is represented by a data size, a determined number of clusters, and the number of attributes that constitutes the vectors of the data sets.

B. Results Achieved in Terms of the Sum of Squared Errors

One of the effective metrics to evaluate the quality of clustering is the sum of the squared errors. This measure is obtained by calculating the sum of the Euclidean distances between each cluster center and the set of data objects belonging to this cluster. Therefore, the sum of the results found corresponds to the sum of squared errors of the set of clusters. Accordingly, the more minimal this measure is, the better the result is achieved.

Table II reports the sum of squared errors of the clustering result obtained by running the parallel PFK-means and the parallel K-means provided by Spark mLlib on the Letter Recognition (LR) data set. The execution of parallel PFK-means performs clustering by automatically detecting the number of clusters that can be varied by changing the position of the new cluster center determined in each iteration of the cluster construction step of the algorithm. Thus, the reported results are obtained by iterating the parallel K-means one, three, and ten times, respectively. In other words, for the parallel PFK-means, the reiteration is applied just to the second phase of the algorithm, which consists of applying the parallel K-means implementation of Spark mLlib using the cluster centers and the number of clusters found in the first phase of the algorithm. In this respect, the acquired results are compared

TABLE II. THE SUM OF SQUARED ERRORS OF THE CLUSTERING RESULTS ON LETTER-RECOGNITION DATA SET

number of clusters	parallel PFK-means	parallel K-means (mLlib)	number of iterations
24	**117015.22**	118400.60	1
24	**113035.25**	113995.74	3
24	111162.61	**110717.19**	10
26	**114813.58**	118262.82	1
26	**111198.35**	112354.80	3
26	**109506.13**	109853.67	10

TABLE III. THE SUM OF SQUARED ERRORS OF THE CLUSTERING RESULTS ON IRIS, WINE, AND PIMA DATA SETS

data set	parallel PFK-means	parallel K-means (mLlib)
Iris	**97.34**	97.70
Wine	**18889.14**	19015.23
Pima	**49403.81**	64315.08

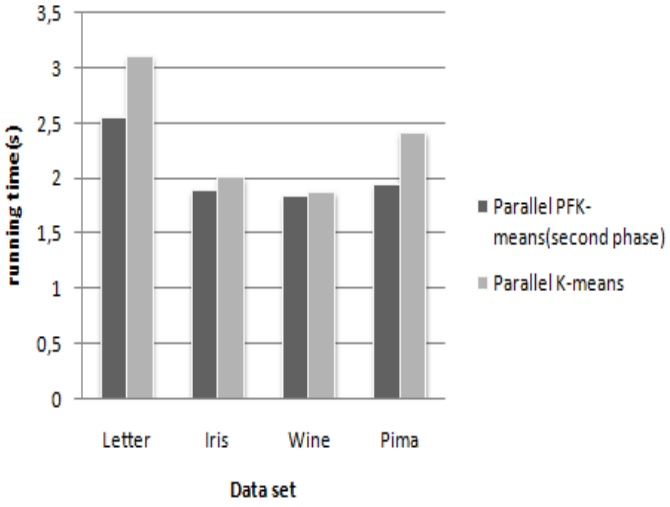

Fig. 7. Clustering Time on Real-world Data Sets.

with the parallel K-means of Spark mLlib using the random initialization mode.

By observing the results displayed in the Table II, it can be seen that the parallel PFK-means outperforms the parallel K-means in terms of the sum of squared errors, except for the tenth iteration with k = 24, for which the parallel K-means exceeds the proposed algorithm. It is also noteworthy that the PFK-means produces good results from the first iteration, implying that the cluster centers discovered in the first phase of the algorithm are well-positioned in comparison to the cluster centers discovered using the random initialization mode. Besides the fact that the parallel PFK-means allows discovering the number of clusters automatically, the result given by the developed algorithm is stable. Table III presents a comparison of the parallel PFK-means and the parallel K-means in terms of the sum of squared errors for the real-world data sets Iris, Wine, and Pima. The results clearly show that the execution of the parallel K-means after determining the input parameters (number of clusters and the cluster centers) from the first phase of the developed parallel algorithm consumes less time compared to the execution of parallel K-means with the random initialization mode for the three data sets used.

C. Results of the Processing Time

In order to evaluate the performance of the parallel PFK-means concerning running time, the second phase of the proposed algorithm has been compared with the parallel K-means of Spark mLlib using the random initialization mode, which allows to randomly generate the cluster centers. In this case, the number of clusters k used is the number detected by the first phase of the parallel PFK-means, and therefore the number of cluster centers must be fixed before the execution. On the other hand, by running the second phase of the PFK-means, the number of clusters is automatically detected and assigned to the parallel K-means as well as the cluster centers.

Fig. 7 illustrates the execution time of the second stage of the parallel PFK-means compared to the execution time of the parallel K-means on Iris, Wine, Pima, and Letter-Recognition data sets. Fig. 7 shows that the PFK-means outperforms the parallel K-means for all the data sets used.

D. Discussion of the Obtained Results

According to the experiments that were conducted on real-world data sets, it was proved that the proposed algorithm is a parameter-free clustering algorithm since it can automatically determine the set of cluster centers and the number of clusters without specifying any input parameters, and therefore the suggested solution is suitable for unsupervised learning. However, running the parallel version of k-means provided by Spark mLlib requires fixing the number of clusters as well as specifying the initial cluster centers. Moreover, the execution of the developed algorithm using real-world data sets gives significant results compared to the results achieved by the parallel implementation of K-means provided by Spark mLlib in terms of the sum of squared errors and execution time. In addition to that, the parallel implementation of the method allows the distribution of the tasks constituting the main steps of the algorithm on several nodes, which allows for the processing of very large files.

VI. CONCLUSION AND PERSPECTIVE

The main concern of this paper is to design a new parallel clustering algorithm based on the main steps of the PFK-means clustering algorithm for processing large quantitative data sets by bringing a significant improvement in the way of determining the initial cluster centers in the cluster construction step. Thus, it exposes a general presentation of the parallel computing platforms and the main process of the Spark framework. Moreover, this paper presents a preview of the sequential version of the PFK-means algorithm [21]as well as a detailed explanation of the proposed parallel algorithm. Furthermore, experiments based on UCI data sets were conducted based on the sum of squared errors and the execution time. The results have clearly shown that the suggested parallel algorithm is an efficient clustering algorithm as it can automatically construct the initial cluster centers as well as the number of clusters without having to specify any parameters beforehand. Besides, it has been demonstrated that the developed algorithm outperforms the parallel K-means provided by Spark mLlib in terms of squared errors and processing time.

In our future research, we intend to concentrate on creating a remote server configuration to distribute the different tasks of the algorithm on multiple nodes and therefore process very large files. In addition, we will establish a deep analysis to compare both the stand-alone mode implementation and cluster mode implementation. Furthermore, we will focus on the implementation of the proposed algorithm using other similarity measures and various data sets. It may also be possible to apply the proposed solution to perform real-time processing.

REFERENCES

[1] M. Naeem, T. Jamal, J. Diaz-Martinez, S. A. Butt, N. Montesano, M. I. Tariq, E. De-la Hoz-Franco, and E. De-La-Hoz-Valdiris, "Trends and future perspective challenges in big data," in *Advances in intelligent data analysis and applications*. Springer, 2022, pp. 309–325.

[2] A. E. Ezugwu, A. M. Ikotun, O. O. Oyelade, L. Abualigah, J. O. Agushaka, C. I. Eke, and A. A. Akinyelu, "A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects," *Engineering Applications of Artificial Intelligence*, vol. 110, p. 104743, 2022.

[3] K. El Handri and A. Idrissi, "Parallelization of $top\_\{k\}$ algorithm through a new hybrid recommendation system for big data in spark cloud computing framework," *IEEE Systems Journal*, vol. 15, no. 4, pp. 4876–4886, 2020.

[4] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*. Elsevier, 2011.

[5] R. Garcia-Dias, S. Vieira, W. H. L. Pinaya, and A. Mechelli, "Clustering analysis," in *machine learning*. Elsevier, 2020, pp. 227–247.

[6] H. Ibrahim Hayatu, A. Mohammed, and A. Barroon Isma'eel, "Big data clustering techniques: Recent advances and survey," *Machine Learning and Data Mining for Emerging Trend in Cyber Dynamics*, pp. 57–79, 2021.

[7] Z. Dafir, Y. Lamari, and S. C. Slaoui, "A survey on parallel clustering algorithms for big data," *Artificial Intelligence Review*, pp. 1–33, 2020.

[8] Y. Lamari and S. C. Slaoui, "Pdc-transitive: An enhanced heuristic for document clustering based on relational analysis approach and iterative mapreduce," *Journal of Information & Knowledge Management*, vol. 17, no. 02, p. 1850021, 2018.

[9] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 10)*, 2010.

[10] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA, 1967, pp. 281–297.

[11] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[12] M. Ahmed, R. Seraj, and S. M. S. Islam, "The k-means algorithm: A comprehensive survey and performance evaluation," *Electronics*, vol. 9, no. 8, p. 1295, 2020.

[13] A. Ashabi, S. B. Sahibuddin, and M. Salkhordeh Haghighi, "The systematic review of k-means clustering algorithm," in *2020 The 9th International Conference on Networks, Communication and Computing*, 2020, pp. 13–18.

[14] S. Bouhout, Y. Oubenaalla, and E. H. Nfaoui, "Comparative study of two parallel algorithm k-means and dbscan clustering on spark platform," in *International Conference on Advanced Intelligent Systems for Sustainable Development*. Springer, 2020, pp. 245–262.

[15] D. Xia, F. Ning, and W. He, "Research on parallel adaptive canopy-k-means clustering algorithm for big data mining based on cloud platform," *Journal of Grid Computing*, vol. 18, no. 2, pp. 263–273, 2020.

[16] W. Lu, "Improved k-means clustering algorithm for big data mining under hadoop parallel framework," *Journal of Grid Computing*, vol. 18, no. 2, pp. 239–250, 2020.

[17] X. Lu, H. Lu, J. Yuan, and X. Wang, "An improved k-means distributed clustering algorithm based on spark parallel computing framework," in *Journal of Physics: Conference Series*, vol. 1616, no. 1. IOP Publishing, 2020, p. 012065.

[18] Y. Liu, X. Du, and S. Ma, "Innovative study on clustering center and distance measurement of k-means algorithm: mapreduce efficient parallel algorithm based on user data of jd mall," *Electronic Commerce Research*, pp. 1–31, 2021.

[19] Y. Mao, D. Gan, D. S. Mwakapesa, Y. A. Nanehkaran, T. Tao, and X. Huang, "A mapreduce-based k-means clustering algorithm," *The Journal of Supercomputing*, vol. 78, no. 4, pp. 5181–5202, 2022.

[20] B. Balusamy, S. Kadry, A. H. Gandomi *et al.*, *Big Data: Concepts, Technology, and Architecture*. John Wiley & Sons, 2021.

[21] S. Slaoui and Z. Dafir, "A parameter-free clustering algorithm based k-means," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 3, 2021.

[22] S. C. Slaoui, Z. Dafir, and Y. Lamari, "E-transitive: an enhanced version of the transitive heuristic for clustering categorical data," *Procedia Computer Science*, vol. 127, pp. 26–34, 2018.

[23] A. Asuncion and D. Newman, "Uci machine learning repository," 2007.