


Article

Improved k-Means Clustering Algorithm for Big Data Based on Distributed Smartphone Neural Engine Processor

Fouad H. Awad *  and Murtadha M. Hamad

College of Computer Science and Information Technology, University of Anbar, Ramadi 31001, Iraq;
dr.mortadha61@uoanbar.edu.iq

* Correspondence: fouad.hammadi@uoanbar.edu.iq

Abstract: Clustering is one of the most significant applications in the big data field. However, using the clustering technique with big data requires an ample amount of processing power and resources due to the complexity and resulting increment in the clustering time. Therefore, many techniques have been implemented to improve the performance of the clustering algorithms, especially for k-means clustering. In this paper, the neural-processor-based k-means clustering technique is proposed to cluster big data by accumulating the advantage of dedicated machine learning processors of mobile devices. The solution was designed to be run with a single-instruction machine processor that exists in the mobile device's processor. Running the k-means clustering in a distributed scheme run based on mobile machine learning efficiently can handle the big data clustering over the network. The results showed that using a neural engine processor on a mobile smartphone device can maximize the speed of the clustering algorithm, which shows an improvement in the performance of the cluttering up to two-times faster compared with traditional laptop/desktop processors. Furthermore, the number of iterations that are required to obtain (k) clusters was improved up to two-times faster than parallel and distributed k-means.

Keywords: big data; clustering; neural engine; k-means; parallel computing



Citation: Awad, F.H.; Hamad, M.M. Improved k-Means Clustering Algorithm for Big Data Based on Distributed Smartphone Neural Engine Processor. *Electronics* **2022**, *11*, 883. <https://doi.org/10.3390/electronics11060883>

Academic Editor: Miin-shen Yang

Received: 25 February 2022

Accepted: 9 March 2022

Published: 11 March 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Currently, we are in a data flood era, as proven by the massive amounts of continuously generated data at unprecedented and ever-increasing scales. In the recent decade, machine learning techniques have become increasingly popular in a wide range of large and complex data-intensive applications, such as astronomy, as well as medicine, biology, and other sciences [1]. These strategies offer potential options for extracting hidden information from the data. However, as the era of big data approaches, the growth of dataset collection in such a large and complex way makes it difficult to deal with it using conventional learning methods, as the learning process for traditional datasets is not designed for and may not work well with large amounts of data. Most classical machine learning algorithms are built to process data that are loaded into memory [2], which is no longer true in the big data context.

The usefulness of the massive volumes of data can be achieved only if the meaning of those data is guaranteed, and proper information can lead to the right path. The information-gathering process from huge unstructured or semi-structured data is the so-called clustering technique. Clustering is a technique of grouping elements based on the similarity of their characteristics and returns those elements as clusters. Thousands of clustering algorithms have been published based on this concept, and k-means is one of the most used. k-means is widely used with a wide range of applications due to its simplicity of implementation and its effectiveness. In the literature, different modifications have been proposed for improving the performance and efficiency of the k-means clustering algorithm.

Big data analytics can extract useful information from numerous amounts of data generated by a variety of sources [3]. Although computer systems and Internet technologies

have seen the evolution of computing hardware following Moore's law for several decades, the issues of processing large-scale data persist as we approach the age of big data [4]. Almost every major company wants to collect massive volumes of data from its customers or underlying infrastructure, then mine them to provide relevant insights in a timely way. These data assist firms in providing better customer service and increasing profits. During each trading session, the New York Exchange collects over 1 TB of trade data [5]. The ability to analyze data in real-time can help traders make key trading decisions. About 23% of all digital data are thought to include useful information that may be used by businesses, government agencies, policymakers, and individual users. Furthermore, many researchers have been working with IoT technologies to improve and solve the critical issues of the performance and data analysis of big datasets [6]. The IoT-based solutions for big data open opportunities to point out the importance and requirements of managing big data in the IoT environments and highlight the future of big data in real-world applications such as the IoT [7]. Multi-view data, which have become more common in recent years, represent one of the applications of utilizing the IoT field with big data in terms of clustering algorithms as multi-view data require a special handling technique [8].

Clustering, which is considered an unsupervised learning approach, is an important technique for data mining and retrieving information from big data. However, using the standard k-means algorithm has some drawbacks, which are: much space and a high processing speed are required when the dataset is very large, while obtaining the best number of (k) requires running the algorithm multiple times to obtain the optimal results. These necessities require a large amount processing unit power, memory space, and energy. In general, there are two main techniques to improve the performance of the clustering system and reduce the energy consumption, which are: powering down the node utilization and matching the workload specifications with the system hardware [9]. Therefore, researchers are trying to improve the performance of the k-means algorithm by taking advantage of parallel computing, distributed computing, and the latest processing technology.

This paper proposes an efficient and high-performance solution to improve the k-means clustering by:

1. Maximizing the performance of the k-means algorithm by running it on the dedicated neural engine processor of smart mobile devices by editing the code and steps of the k-means algorithm to run on the single-instruction-based machine with an ARM-based processor;
2. Dividing the k-means clustering technique into two parts. The first part, which contains the main tasks of the k-means algorithm, runs on a neural engine processor, while the second part, which contains the general tasks of the k-means technique, runs on the mobile general-purpose processor;
3. Using a distributed Spark server, multiple mobile devices can process the k-means algorithm many times in a distributed network and give the optimal result to the server to select the optimal number of (k).

This paper contains seven sections. The Section 1 discusses the importance of big data and clustering algorithms in real-world applications and especially focuses on the main characteristics and limitations when working with big data. The big data clustering algorithms and main techniques are discussed in Section 2. Section 3 presents the main advantages and problems in the current big data platform. Section 4 presents the recent research in big data clustering. Section 5 presents the proposed solution. Section 6 discusses the implementation and results of the proposed solution. Section 7 is dedicated to the conclusions.

2. Big Data Clustering

Clustering is the technique of splitting data objects into a set of clusters based on the similarities and differences of the objects within datasets. It is considered as one of the important methods to recognize the set of objects based on their similarities and generate a pattern from a dataset that is unlabeled (unsupervised). Typically, the technique of

big data clustering can be categorized into two types [10]: single-machine clustering and multiple-machine clustering techniques, as is illustrated in Figure 1. The single-machine clustering technique is designed to cluster the data on a single device without the ability to take advantage of a gridded or distributed high-performance system. However, the multiple-machine clustering technique is much faster and more adaptable and can handle the new big data challenges much better [11]. Therefore, in this article, the main focus was on multi-machine clustering.

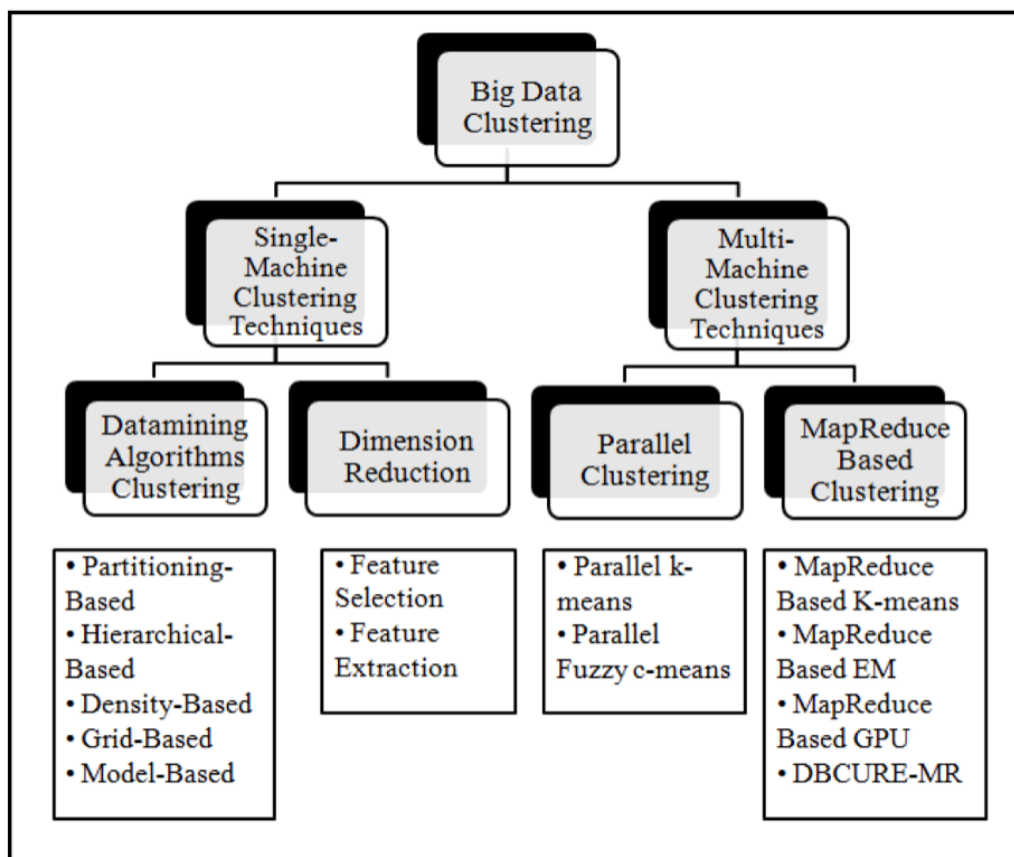


Figure 1. Big data clustering techniques.

2.1. Multi-Machine Clustering

Multi-machine clustering is divided into three main types based on the mechanism of taking advantage of multi-machine performance. The multi-machine term is used for a single or many devices that have a multi-processing ability. Typically, the multi-machine technique is divided into two main categories, which are:

1. Parallel clustering: In general, parallel computing is used to handle many tasks simultaneously, which is typically designed to handle heavy tasks. Therefore, as clustering is considered a heavy task, parallel computing is used with it. In this section, some distributed clustering and parallel processing techniques that can deal with big data are examined. The parallel clustering splits the data partitions that are distributed on different machines [12]. This improves the scalability and speeds up the clustering [13];
2. MapReduce-based clustering: This technique works with large volumes of data by partitioning tasks to be distributed and executed on a large number of servers. Such a technique decomposes the tasks (Map) into smaller tasks that can be dispatched. Finally, it collects and consolidates the results (Reduce) [14]. Figure 2 describes the function of the MapReduce technique.

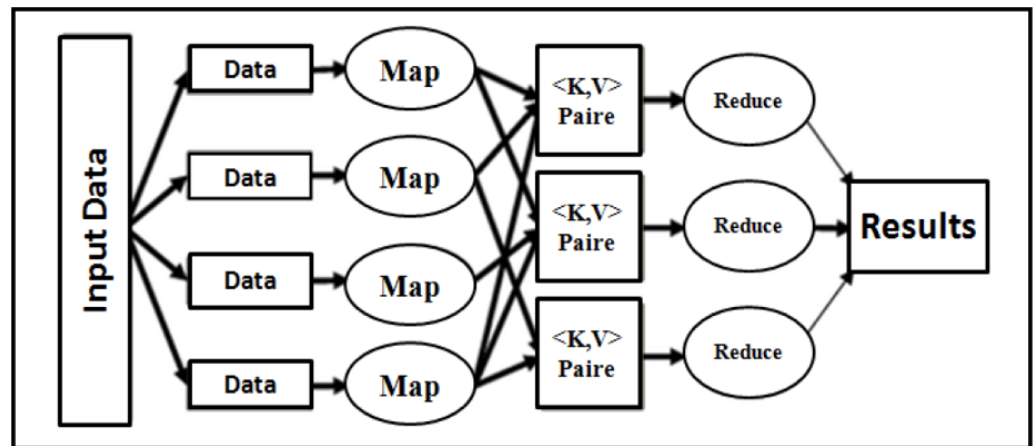


Figure 2. MapReduce function.

In this work, special attention was given to partitioned-based clustering algorithms. This technique family has been proven in terms of efficiency and clustering performance, and this is due to its ability to detect noisy objects and find clusters of unusual shapes.

2.2. *k*-Means Clustering

k-means clustering seeks to group n objects into k clusters based on the nearest mean value group, serving as a prototype of the cluster. The iterative refinement technique is the most common technique used with the standard *k*-means algorithm. However, this technique is the so-called native *k*-means to differentiate from the much faster alternative solutions that have already been developed. Each clustered object should be compared and calculated based on the nearest centroid points, which are random initialized points to represent the cluster center.

k-means clustering has three main issues that most of the recent solutions have been trying to tackle, which are [15,16]: (P1) pre-clustering requirements, which represent the number of clusters and answer the following question “Do the data have clusters? If yes, how many (k)?”; (P2) finding the k clusters of the data; (P3) post-clustering validation, which represents the validity and accuracy of the clustering process.

3. Big Data Platform

A summary of the most-used platforms for big data is given in this section. In general, there are two types of platforms, which are: horizontal scaling and vertical scaling, as shown in Figure 3.

The horizontal scaling platforms work based on scaling the processing performance by increasing the number of devices in the clustering system. This technique includes MapReduce, peer-to-peer networks, and Spark. On the other hand, the vertical scaling principle is to scale the processing power for clustering of the device itself (vertically). This technique includes a graphics processing unit, field programmable gate array (FPGA), and multi-core CPU. Each platform has several specific design clustering algorithms, and some related techniques are discussed in this paper.

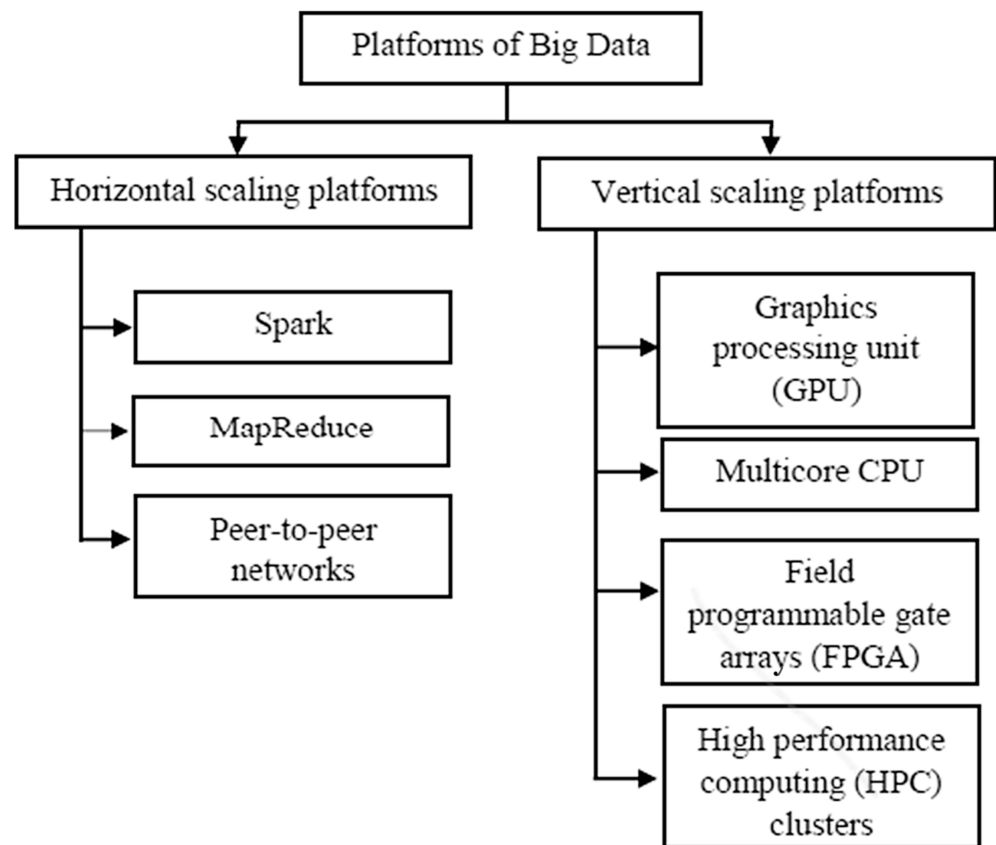


Figure 3. Big data platform.

Spark Platform

Apache Spark is a data-intensive application framework that is designed to process big data and can be executed on commodity clusters [17]. The main difference between the Spark framework and the competition such as MapReduce is that it loads only the useful dataset into the memory, which enables iterative jobs to run queries on big datasets. Such a technique can reduce the execution time significantly.

There are three fundamental aspects presented in Spark, which are [18]: resilient distributed datasets (RDDs), parallel operations, and shared variables. RDDs are a shared-machine-based collection of objects that can be restored in the case of loss. They are reusable in multiple parallel MapReduce jobs, as they can be stored in memory. Secondly, the tasks can be executed in parallel with RDDs by reducing and collecting for each operation. The last aspect consists of broadcast variables and accumulators.

Practically, the main advantages of Spark are its flexibility, ease of use, and that the program does not need any abstraction. The high performance of Spark enables it to deal with data in a real-time streaming module and by using distributed workers, which is caching the result partially in memory. Moreover, it is very efficient such that it outperforms the Hadoop MapReduce framework [19] while preserving the fault tolerance and scalability of MapReduce, which is up to 10-times faster for interactive machine learning workloads [12].

4. Related Work

A variety of scalable machine learning algorithms have been developed in recent years to address various difficulties in big data analytics, with the majority of them designed to cluster data before processing.

In 2017, Reference [20] used a data science and engineering solution based for fast k-means clustering of big data. The article addressed the k-means problems with the risks

associated with the random selection of the k centroid, the equal circular cluster, and the complexity of k -means, which is very high. The solution used a heuristic prototype-based algorithm together with k -means clustering to improve the efficiency and scalability of the clustering process.

In 2018, Reference [21] studied the performance of the k -nearest neighbor classifier (KNN) and k -means clustering for predicting diagnostic accuracy. The proposed solution aimed to improve the performance of the diagnosis and classification process for the machine learning repository prepared by the University of Wisconsin Hospital. The performance of KNN was compared with k -means clustering using the same dataset from the hospital.

In 2018, Reference [22] worked on the improvement of k -means clustering using the density canopy. The density canopy technique has been used for preprocessing procedures before running the k -means clustering as canopy results in the initial clustering centers. The proposed solution was tested and simulated with the well-known dataset from the UCI machine learning repository and a simulated dataset with noise. The simulation results showed an improvement in the performance of the k -means algorithm by up to 30.7% in terms of accuracy on the UCI dataset and up to 44.3% with the simulated data compared with the traditional k -means algorithm. However, the accuracy improvement went down to only 6% in some cases.

In 2019, Reference [23] proposed a novel and efficient clustering technique for big data clustering in Hadoop. The proposed solution tried to solve the efficiency of the k -means clustering algorithm by using a new hybrid algorithm on the basis of the precision, recall, F-measure. The performance of the solution was evaluated regarding the execution time and accuracy using the National Climatic Data Center (NCDC) dataset, which contains data from over 20,000 stations around the world. However, the analysis of the data result was not clear, and the execution time was not tested for a multi- k value.

In 2019, Reference [24] worked on the importance of the mixed data approach by addressing the limitations of most clustering algorithm solutions, which are focused on either numerical or categorical data only. The authors implemented the clustering technique on a real-world mixed-type dataset to validate the clustering with such data types. The results showed that using clustering techniques for mixed data in general via a discretization procedure may result in the loss of essential information.

In 2020, Reference [25] proposed an improvement of the k -means clustering algorithm for big data. The proposed solution aimed to improve the efficiency and accuracy of the data clustering. However, the experiment used a private artificial dataset with up to 50,000 records to measure the clustering performance compared with simple k -means. The results showed an improvement in the clustering speed of up to 50%.

In 2020, Reference [26] proposed a new solution to improve the k -means clustering for big data using the Hadoop parallel framework. The improvement was in the clustering process by dividing and merging clusters according to the distance between them. The points that did not belong to a cluster were divided to be clustered with the nearest clusters. Such a method should improve the efficiency of k -means clustering. The KDD99 dataset was used to evaluate the performance of the solution with shared memory space and a parallel Hadoop platform. The simulation results showed an improvement in the accuracy by up to 10%.

In 2021, Reference [27] worked on the improvement of k -means clustering for big data. The improved version of the k -means algorithm was applied with acceptable precision rates to big data with lower processing loads. Using this technique, the distance between points was used along with their variations with the last two iterations. Furthermore, those points were compared together with the research index cluster radius. The results showed an improvement in clustering by up to 41.8% in the base case. Many real datasets were used in the experiment, such as the Concrete, Abalone, Facebook, and CASP datasets. Moreover, the processing power requirement of the solution was high, which was performed with 12 GB of RAM and a multi-core Core i7 processor.

In 2021, Reference [28] analyzed the performance of the simple k-means algorithm and improved it with the parallel k-means technique to optimize the clustering of a dataset. The solution was designed to take advantage of multi-core machines to improve the clustering performance. The experiment used an education sector dataset with up to 10 times of evaluation to calculate the time elapsed and the number of iterations for the clustering. The overall performance was improved up to three-times faster than the simple k-means algorithm.

Table 1 summarizes the related works that made an improvement of k-means clustering to work with big datasets.

Table 1. Related works.

Proposed Solution	Focus	Drawbacks	Dataset
Fast k-means clustering [20]	Efficiency and scalability	Not tested with a real dataset	Private
KNN and k-means algorithms for predicting the diagnostic accuracy [21]	Performance and accuracy	Solution verified with only one dataset	University of Wisconsin Hospital dataset
k-means improvement with canopy [22]	Accuracy	Accuracy improvement not stable	UCI dataset and simulated data
Novel hybrid clustering in Hadoop [23]	F-measure and execution time	Experiments not applied for different k values	National Climatic Data Center (NCDC) dataset
Real mixed-type application [24]	Performance and efficiency	-	Real-world mixed dataset
Fast k-means clustering algorithm [25]	Accuracy and performance	Using image dataset only	Artificial and actual datasets AT&T, Yale, COIL-20, CMD)
k-means clustering improvement with Hadoop [26]	Efficiency and execution time	-	KDD99 dataset
k-means improvement [27]	Execution time	Requires a high-performance processing machine	Multiple datasets (Facebook, CASP, etc.)
Parallel k-means clustering [28]	Number of iterations and time elapsed	-	Education sector dataset

5. Proposed Solution

The rapid growth of the data volume has led to the expansion of the use of clustering algorithms. However, the effectiveness of the traditional techniques is not always high because of the high computational load. The basic idea of the proposed algorithm is to take advantage of the high performance of smart mobile devices in machine learning through an artificial intelligence processor dedicated to machine learning algorithms. The main goal is to speed up the performance of the k-means algorithm to work better and give higher performance by taking advantage of the technology mentioned above. Due to the simplicity, efficiency, and adaptability of the k-means algorithm, it is considered the most popular clustering method. The cluster efficiency should be always maintained when involving a large dataset, while the intercluster and intracluster distances between data objects in a dataset should be considered. A new distributed clustering based on the k-means algorithm is suggested.

The proposed algorithm distributes a dataset equally to the smartphone device, which is connected to a cloud server. Then, it runs the clustering algorithm, taking advantage of the machine-learning-engine-dedicated processors in modern mobile devices. The goal is to preserve the dataset characteristics while improving the clustering performance. The

new solution can handle the three issues in the k-means algorithm, which are: the number of k, optimal centers, and best clustering performance. The centers are calculated for each cluster on each device and compared later with the other clustering results that come from the devices in the network.

Algorithm 1 shows the main steps of the processing on the server’s side. The Spark framework starts preparing the dataset to be sent to the clients, then begins receiving the requests from the clients. The server keeps listening to the clients, which request the content or send the results. The best result from the received results is selected. Each dataset is marked with the best random centroid points.

Algorithm 1 Proposed k-means distribution.

Require: Big dataset
Input: Prepare dataset for clients
Start: Init database to receive results
while $file \neq \text{IncomeHTTP} - \text{Stopped}$ **do**
 Save: *centroidpoints* from clients
end while
Spark: select best result
Save: best centroid points
Output: centroid points for the dataset

Figure 4 describes the complete layout for the network of the proposed solution. First, the big dataset is in the cloud server and managed by the cloud management server, which is responsible for the data flow, results, and optimal solution selection.

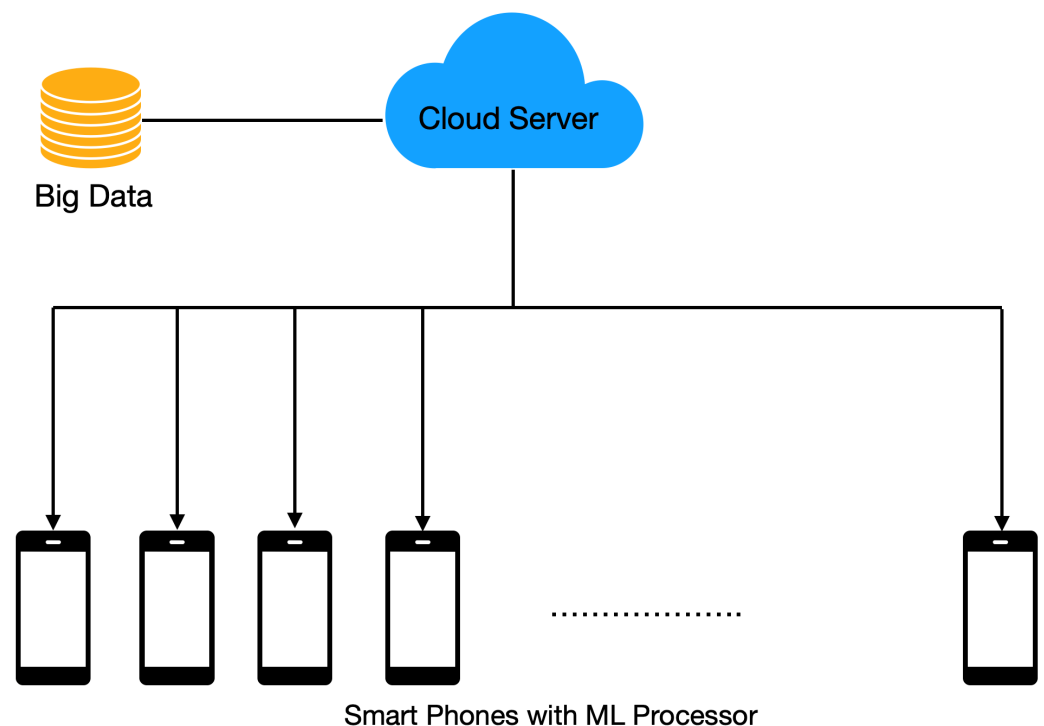


Figure 4. Proposed solution network.

Besides that, the Spark server system also works with the cloud management system to select the optimal solutions that come from the mobile devices in the network. Finally, the system requires a mobile device with a dedicated ML engine connected to the cloud server. The mobile device starts downloading the dataset from the server, then processes the data with the random centered point of the k-means algorithm. With the high performance of

dedicated ML processors and taking advantage of the multi-core system of ML processors, mobile devices can cluster data at a high speed. Each mobile device in the network returns the random center point and clustering accuracy to the server, as shown in Figure 5.

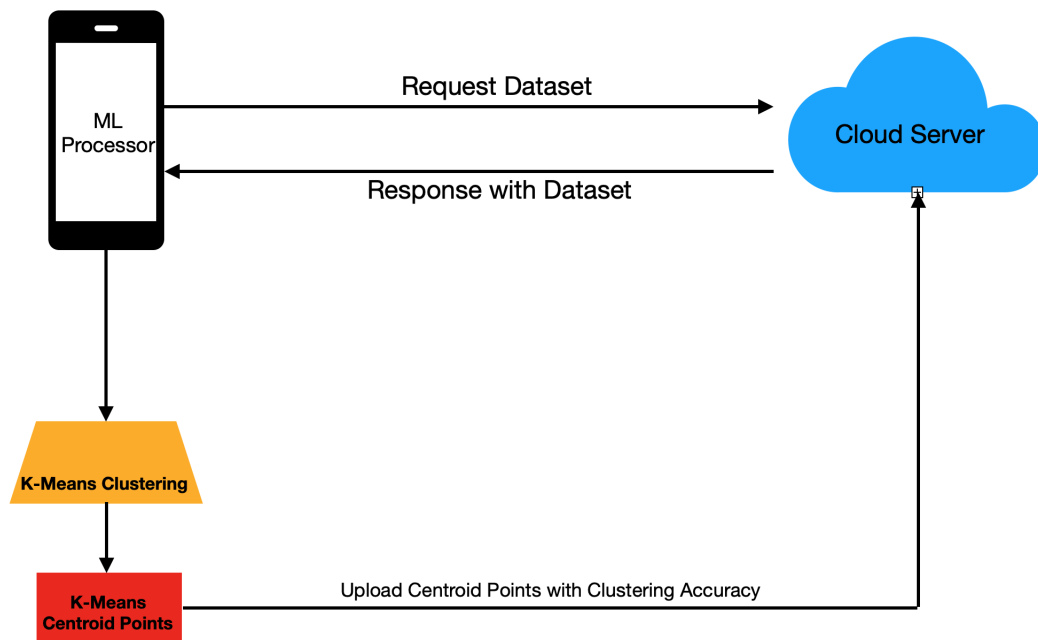


Figure 5. Proposed solution network.

5.1. Proposed Solution Processing

Parallel neural engine k-means clustering consists of three main steps, which are:

- Partition;
- Computation;
- Compilation.

In the first step, the mobile device’s main processor divides the dataset into sub-datasets. All neural processors within the mobile device system process these sub-datasets with an initial centroid and a specific number of clusters. Each processor calculates clusters, and then, the main processor collects the results to be prepared and sent to the server. This process continues until there is no change in the clusters. Each partition of the dataset has its centroid points; therefore:

$$C = C1 + C2 + C3 + C4 + Cn \tag{1}$$

Send initial centroids: $C1, C2, C3, C4, \dots, Cn$ and sub-datasets to all connected processors receive clusters and centroids from all the processors. The server recalculates the distance amongst all these centroid points later to detect the best performance from all results that are collected.

The number of incoming results to the server depends on the number of connected devices. Therefore and due to the vast amount of modern mobile devices (phones, tablets, etc.), many results are collected by the server. Such an amount of results requires a selection algorithm to select the best results of clustering, and as a result, the optimal center points can be expected in the future without requiring sending the data again to the mobile devices.

As described in Algorithm 2, the mobile device sends a request to the server to retrieve the dataset, then it initializes the centroid points with a random location.

Algorithm 2 k-means clustering on mobile devices.

Require: Request dataset file from the server**Input:** random centroid points**Start:** clustering points**while** *file* \neq *end* **do** **Calculate:** *meanvalue***end while****Output:** clustered points

The cluster initialization requires partitioning all the dataset into k number of sub-datasets where the initial clusters can be calculated with the following equation:

$$\text{initial cluster} = \text{floor}(n/k) \quad (2)$$

where n is the total number of sub-data partitioning and k is the number of clusters. The mobile device starts clustering the objects with a loop until the end of the dataset file. The output of the process is the clustered points with the initial centroid points.

The distance between centroid points is calculated on the server's side to determine the best result from the collected results, and this is achieved by applying the following equation:

$$\text{Euclidean Distance}(C_i, D_i) = |C_i - D_i| \quad (3)$$

where C_i is the initial centroid points and D is the data item, which should be less than the (n) value.

Figure 6 summarizes the data processing on the mobile device. The data are requested from the server by the mobile device, then they are partitioned to be processed by the neural engine and general-purpose processor. The general-purpose processor is responsible for the general calculation, such as the initialization of the centroid points and calculating the variant of each cluster. The neural engine processor calculates the rest of the operations and tasks with the k-means algorithm. All these tasks run in parallel for both the neural and general-purpose processors.

The proposed solution does not directly affect the clustering precision because it does not edit the clustering precision part in the standard k-means clustering. The solution focuses on improving the performance of the standard k-means clustering by editing the way that k-means runs and clusters the objects as shown in Figure 6. It splits the clustering processes for both the neural and general-purpose processors; therefore, the core of the k-means clustering itself is not affected. However, the precision is improved in different ways. Running k-means clustering with high performance and very low consumption regarding both the energy and system resources enables running the clustering process many times to obtain different results. The results collected from the mobile devices connected to the server are evaluated to select the best amongst them.

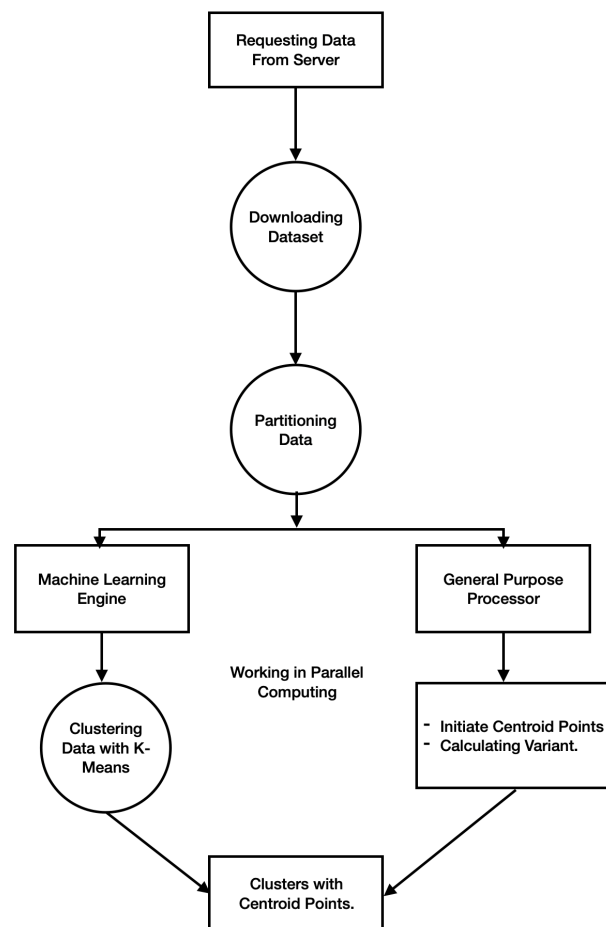


Figure 6. Proposed solution processing flowchart.

5.2. Complexity Analysis

k-means algorithms are comprised of two phases, which are computing the initial clusters after dividing the dataset into k equal parts and then calculating the arithmetic means where the second phase is to assign the item to an appropriate cluster. $O(n)$ is the time complexity of the first phase, as it requires partitioning and finding a fixed period. On the other hand, the second phase has $O(nkt)$ complexity, where n is the number of items in the data, k is the number of clusters, and t is the number of iterations that are performed to cluster all items.

Thus, the total time complexity of the k-means clustering algorithm is equal to adding the maximum of $O(n)$ to $O(nkt)$, that is:

$$\text{Max}(O(n)) + O(nkt) = O(nkt) \quad (4)$$

so the overall time complexity is $O(nkt)$.

6. Analysis of the Experiment Results

The comparison between the proposed solution and the recent k-means algorithm improvements was based on the following terms:

- Number of iterations, which represents the number of iteration processes that are required to cluster the dataset into (k), the number of clusters;
- Cluster quality, which represents the performance of the clustering in terms of the accuracy of the clustering;
- Elapsed time: this term is calculating the time that is required to cluster the dataset into (k) clusters.

Each of the above terms was calculated for each algorithm and compared with the proposed solution to present the quality and validity of the proposed solution.

The main results of the proposed solution are shown with all the above performance measurements and with two main big datasets, which are: the Google Play Store dataset and the KDD99 dataset.

The results obtained were compared with the k-means algorithm. The experiments showed the performance of the proposed algorithm for big data. The proposed solution was developed with ARM-based code with a dedicated machine learning code for the smartphone ML engines. For the iOS device, the Swift programming language was used, whereas the Kotlin programming language was used for the Android smartphone. For the server's side, the Python programming language with the Spark server was used with the Fedora Linux operating system. The k-means algorithm was run on a desktop-based processor (Intel Core i5 3.5 GHz with four cores) and the Unix-based operating system without an ML engine, then with mobile devices with ML engines. The mobile devices that were used in the experiment were the iPhone 11 Pro Max with an ML engine, which contains 16 cores dedicated to the machine learning process, and the Samsung S22, which includes a neural processing unit (NPU), as the Android smartphone with a 16 bit floating-point number (FP16). The experiments were divided into two categories: single-core neural engine performance and multi-core neural engine performance with mobile ARM cores.

Several datasets were used in the experiments to validate the proposed solution in many situations and dataset properties. All the dataset characteristics are given in Table 2.

Table 2. Dataset characteristics.

Dataset	Size in MB	Number of Records
Education Sector Dataset [28]	30 MB	15,000
Google Play Store Dataset	300 MB	11,000,000
KDD99 Dataset [26]	71 MB	9,000,000

6.1. Neural Engine Performance

First, the k-means algorithm was evaluated with a single-core neural engine processor without using the mobile ARM processor. This means all the operations were examined with only neural cores. Table 1 shows the result of the experiment, which clearly shows the high performance of the mobile processors when running a machine learning algorithm compared with the non-ML processor.

The performance as shown in Table 3 was about twice as fast with a dedicated ML processor when processing the 11 million records of the Google Play Store dataset. The next experiment was performed on the education sector dataset, and the mobile processors showed a performance up to 10-times faster than the desktop processor. The second experiment ran the k-means algorithm on many devices connected to the server. This part of the experiment required testing with the real-world server; therefore, a testing cloud server was built. The result as shown in Figure 3 describes the effects of the number of mobile devices on the number of results in minutes by taking the time that was required to send and receive the requests from the devices.

Table 3. Clustering performance.

Dataset	Desktop Processor	Mobile Processor
Google Play Store Dataset	90 min	46.1 min
Education Sector Dataset	24.3 ms	2.3 ms

6.1.1. Number of Iterations

To validate the proposed solution compared with the recently advanced k-means, the number of iterations was calculated with different numbers of (k).

As shown in Table 4, the number of processors used to perform the computations for each series run was verified. The number of clusters in the k-means algorithm was specified to generate efficient results for different variations of the clusters. Clustering is an issue that mainly depends on the data used and the problems considered. The proposed algorithms showed a significant increment in the efficiency of clustering in terms of execution.

Table 4. Number of results per hour.

Dataset	Number of Devices	Results Received per Hour
Google Play Store Dataset	5	4
	10	9
	15	12
	25	22

In Table 5, several iterations are fixed in the case of the parallel neural k-means algorithm using the education sector dataset, i.e., for $k = 4, 5, 6, 7$, but this kept changing from one run to another in the case of the parallel k-means clustering algorithm with multiple running times.

Table 5. Number of iterations for k clustering.

Runs/Executions	Number of k	Simple k-Means Clustering	Parallel k-Means [28] (2021)	Proposed Solution
1	3	12	3	3
2		9	3	2
3		12	3	2
4		9	3	3
5		7	3	2
6		14	3	3
7		9	3	3
8		12	3	2
9		10	3	3
10		15	3	2
1	4	15	4	3
2		18	4	3
3		18	4	3
4		15	4	3
5		15	4	3
6		15	4	3
7		16	4	3
8		14	4	3
9		13	4	3
10		13	4	3
1	5	14	6	5
2		18	6	5
3		17	6	3
4		22	6	5
5		23	6	3
6		18	6	5
7		26	6	3
8		24	6	5
9		28	6	3
10		26	6	5

Table 5. Cont.

Runs/Executions	Number of k	Simple k-Means Clustering	Parallel k-Means [28] (2021)	Proposed Solution
1	6	14	7	3
2		13	7	3
3		17	7	3
4		20	7	3
5		16	7	3
6		18	7	3
7		17	7	3
8		21	7	3
9		19	7	3
10		23	7	3
1	7	10	8	4
2		12	8	4
3		19	8	5
4		20	8	5
5		15	8	4
6		17	8	5
7		17	8	4
8		20	8	4
9		22	8	5
10		21	8	4

Figure 7 shows the overall performance of the proposed solution compared with simple k-means clustering and parallel k-means clustering using the education sector dataset. It is noticeable that the proposed solution outperformed the compared algorithms in terms of the number of iterations. The number of iterations decreased significantly due to the efficiency and performance of the dedicated neural engine processor. The neural engine processor was designed with ARM-v8-based technology, and this enabled executing each task with a single 64 bit instruction with up to 16 cores [29]. Furthermore, with the minimal energy consumption using the single-instruction technique, it enabled running all processor cores to execute tasks in parallel at the same time efficiently [30]. Such a technique lets the tasks in the clustering run on dedicated neural cores and as a result decreases the iterations number, which is required to run over all objects.

In Table 6, the number of iterations is fixed in the case of the parallel neural k-means algorithm using the Google Play Store dataset, i.e., for $k = 4, 5, 6, 7$, but kept changing from one run to another in the case of the parallel k-means clustering algorithm with multiple running times.

Figure 8 shows the overall performance of the proposed solution compared with simple k-means clustering and parallel k-means clustering using the Google Play Store dataset. It is noticeable that the proposed solution outperformed the compared algorithms in terms of the number of iterations by up to four-times better and up to ten-times better than simple k-means clustering.

Table 6. Number of iterations for k clustering.

Runs/Executions	Number of k	Simple k-Means Clustering	Parallel k-Means [28] (2021)	Proposed Solution
1	3	33	5	3
2		23	5	2
3		40	5	2
4		35	5	3
5		38	5	2
6		39	5	3
7		32	5	3
8		33	5	2
9		35	5	3
10		23	5	2
1	4	34	7	4
2		44	7	4
3		47	7	4
4		49	7	4
5		34	7	4
6		42	7	4
7		41	7	4
8		42	7	4
9		41	7	4
10		38	7	4
1	5	44	9	5
2		34	9	5
3		44	9	3
4		47	9	5
5		47	9	3
6		49	9	5
7		50	9	3
8		43	9	5
9		42	9	3
10		40	9	5
1	6	45	11	3
2		48	11	3
3		51	11	3
4		52	11	3
5		48	11	3
6		47	11	3
7		49	11	3
8		53	11	3
9		55	11	3
10		54	11	3
1	7	44	12	4
2		43	12	4
3		46	12	5
4		53	12	5
5		52	12	4
6		54	12	5
7		51	12	4
8		50	12	4
9		48	12	5
10		49	12	4

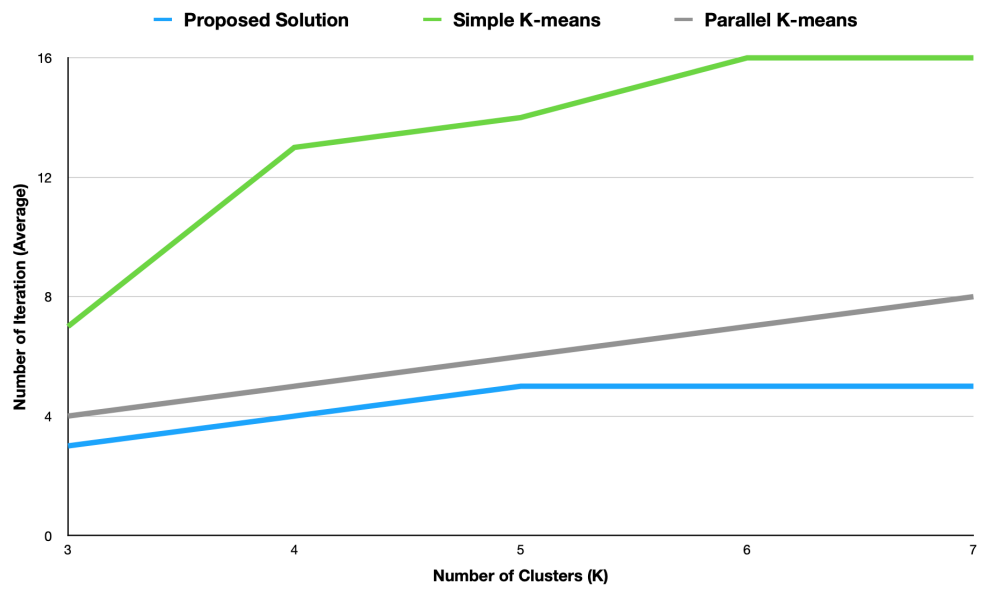


Figure 7. Number of iterations for the education sector dataset.

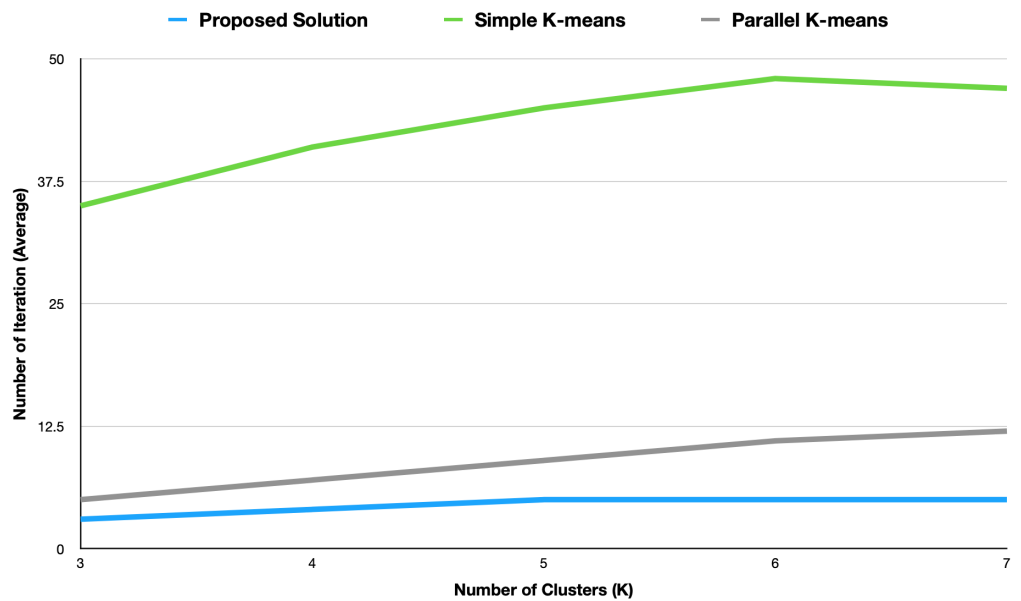


Figure 8. Number of iterations for the Google Play Store dataset.

6.1.2. Time Elapsed Performance

Calculating the time elapsed for the proposed solution is an important term as it presents the overall performance in terms of time. The performance of the solution compared with standard k-means (simple k-means) and parallel k-means was calculated, and the results are shown in Table 7 for the education sector dataset.

Table 7. Time elapsed for k clustering in milliseconds.

Runs/Executions	Number of k	Simple k-Means Clustering	Parallel k-Means [28] (2021)	Proposed Solution
1	3	18.7	7.8	2.3
2		14.1	7.8	2.1
3		20.3	9.3	3.1
4		18.7	9.2	2.3
5		14.6	7.5	2.2
1	4	18.7	7.8	2.1
2		14.1	7.8	1.9
3		20.3	9.3	2.8
4		18.7	9.2	2.1
5		14.6	7.5	2.5
1	5	18.7	10.3	4.2
2		18.8	10.3	4.2
3		21.2	9.3	5.1
4		18.9	9.8	4.8
5		24.1	11.1	3.9
1	6	18.7	11.2	6.1
2		25.7	12.4	6.1
3		24.1	14.1	7.2
4		22.3	14.3	7.2
5		27.6	15.1	6.1
1	7	23.5	10.9	7.3
2		22.5	15.1	7.3
3		21.4	13.7	6.5
4		25.1	14.2	6.8
5		20.3	15.1	7.3

To measure the performance of the solution compared with standard k-means (simple k-means) and parallel k-means with a big dataset, the elapsed time was calculated, and the results are shown in Table 8 for the Google Play Store dataset.

Table 8. Time elapsed for k clustering in minutes.

Runs/Executions	Number of k	Simple k-Means Clustering	Parallel k-Means [28] (2021)	Proposed Solution
1	3	210	150	46
2		220	155	60
3		205	160	55
4		213	157	57
5		216	152	52
1	4	223	150	49
2		230	152	58
3		235	164	62
4		223	153	63
5		228	152	68
1	5	240	160	64
2		245	165	69
3		243	168	72
4		251	170	71
5		246	162	75

Table 8. Cont.

Runs/Executions	Number of k	Simple k-Means Clustering	Parallel k-Means [28] (2021)	Proposed Solution
1	6	260	171	71
2		264	177	70
3		280	181	73
4		275	179	75
5		279	179	74
1	7	275	175	75
2		280	178	78
3		281	185	77
4		277	187	79
5		284	188	74

The next experiments were performed over another big dataset and compared with an advanced k-means clustering algorithm [26]. To achieve a fair comparison, the KDD99 data that was used in [26] was used in the experiment with the same settings and row count for clustering. The results are shown in Table 9, giving the comparison amongst the proposed solution, single-machine clustering, and Hadoop platform clustering. It describes clearly the efficiency and high performance of the proposed solution compared with the advanced k-means clustering. The time consumption for the proposed solution was up to four-times faster than the single-machine and Hadoop platform clustering in the overall comparison.

Table 9. Time elapsed for k clusters in seconds.

Number of Data	Number of Clusters	Single-Machine Clustering [26] (2020)	Hadoop Platform Clustering [26] (2020)	Proposed Solution
15,000	3	10.5	9.2	2.3
30,000		23.4	12.4	4.3
45,000		48.1	27.8	6.1
15,000	4	12.3	11.1	3.1
30,000		25.2	13.2	5.3
45,000		50.1	29.2	7.3
15,000	5	13.1	12.3	3.6
30,000		26.3	14.1	5.6
45,000		51.2	29.9	7.7
15,000	6	13.9	13.1	4.2
30,000		27.5	14.9	6.3
45,000		52.3	30.7	8.1
15,000	7	14.3	15.2	5.8
30,000		29.8	17.1	7.1
45,000		53.8	32.8	9.2

6.2. Multiple Cores and Multiple Processors

In the second part of the experiment, the clustering algorithm was divided into two sections. First, the normal operations and tasks were run on mobile ARM processors and cores to take advantage of the high performance of the cores and preserve the energy of the device at the same time. Second, the k-means algorithm itself was run specifically on the neural engine cores of the mobile processor.

Figure 9 shows that increasing the number of cores besides the neural engine cores significantly increased the performance of the overall clustering by up to two-times faster than using only neural engine cores for the overall system.

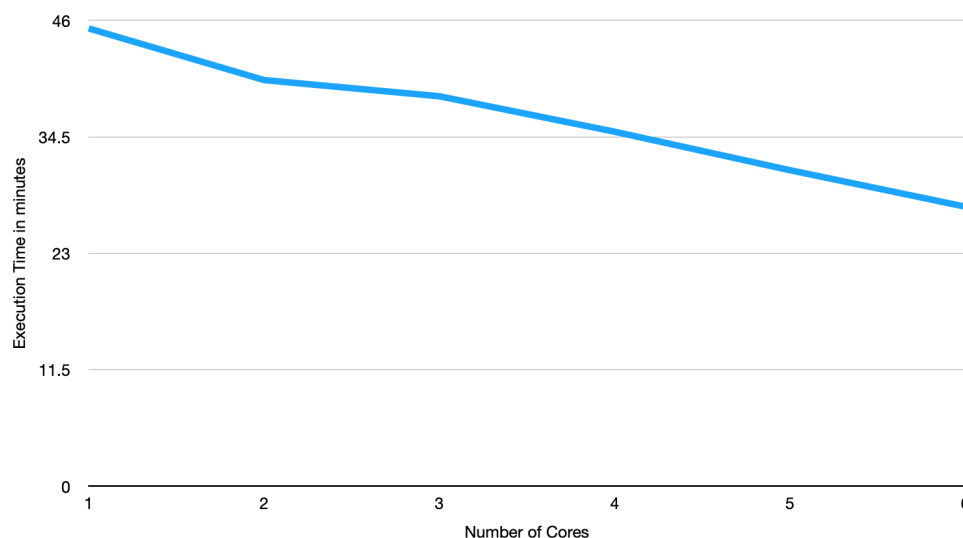


Figure 9. Multi-core performance.

7. Conclusions

Processing big data efficiently requires using mobile-processor-based clustering. Many related clustering techniques were discussed in this paper to provide an overview of the clustering technique. k-means, which is a clustering-based algorithm, was used in this paper to implement the proposed technique. It can work efficiently with numerical data better than categorical data. Running k-means clustering on a machine-learning-based processor can significantly improve the performance and efficiency of the clustering. Using a distributed system can effectively handle the problem of running k-means clustering to find the optimal centroid points. Moreover, the number of iterations of clustering can be increased without affecting the speed of the overall system.

Author Contributions: Conceptualization, F.H.A. and M.M.H.; methodology, F.H.A. and M.M.H.; software, F.H.A. and M.M.H.; validation, F.H.A. and M.M.H.; formal analysis, F.H.A. and M.M.H.; investigation, F.H.A. and M.M.H.; resources, F.H.A. and M.M.H.; data curation, F.H.A. and M.M.H.; Writing and original draft preparation, F.H.A. and M.M.H.; writing a review editing, F.H.A. and M.M.H.; visualization, F.H.A. and M.M.H.; supervision, F.H.A. and M.M.H.; project administration, F.H.A. and M.M.H.; funding acquisition, F.H.A. and M.M.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: We would like to thank all who gave us support to complete this paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

- Zhang, Q.; Yang, L.T.; Chen, Z.; Li, P. A survey on deep learning for big data. *Inf. Fusion* **2018**, *42*, 146–157. [[CrossRef](#)]
- Baum, J.; Laroque, C.; Oeser, B.; Skoogh, A.; Subramanian, M. Applications of big data analytics and related technologies in maintenance—Literature-based research. *Machines* **2018**, *6*, 54. [[CrossRef](#)]
- Nguyen, B.; De Baets, B. Kernel-based distance metric learning for supervised k-means clustering. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 3084–3095. [[CrossRef](#)] [[PubMed](#)]
- Tsai, C.W.; Lai, C.F.; Chao, H.C.; Vasilakos, A.V. Big data analytics: A survey. *J. Big Data* **2015**, *2*, 1–32. [[CrossRef](#)]
- Mahdi, M.A.; Hosny, K.M.; Elhenawy, I. Scalable clustering algorithms for big data: A review. *IEEE Access* **2021**, *9*, 80015–80027. [[CrossRef](#)]
- Cai, H.; Xu, B.; Jiang, L.; Vasilakos, A.V. IoT-based big data storage systems in cloud computing: Perspectives and challenges. *IEEE Internet Things J.* **2016**, *4*, 75–87. [[CrossRef](#)]
- Ahmed, E.; Yaqoob, I.; Hashem, I.A.T.; Khan, I.; Ahmed, A.I.A.; Imran, M.; Vasilakos, A.V. The role of big data analytics in Internet of Things. *Comput. Netw.* **2017**, *129*, 459–471. [[CrossRef](#)]

8. Fu, L.; Lin, P.; Vasilakos, A.V.; Wang, S. An overview of recent multi-view clustering. *Neurocomputing* **2020**, *402*, 148–161. [[CrossRef](#)]
9. Zhang, Y.; Cao, T.; Li, S.; Tian, X.; Yuan, L.; Jia, H.; Vasilakos, A.V. Parallel processing systems for big data: A survey. *Proc. IEEE* **2016**, *104*, 2114–2136. [[CrossRef](#)]
10. Ohadi, N.; Kamandi, A.; Shabankhah, M.; Fatemi, S.M.; Hosseini, S.M.; Mahmoudi, A. Sw-dbscan: A grid-based dbscan algorithm for large datasets. In Proceedings of the 2020 6th International Conference on Web Research (ICWR), Tehran, Iran, 22–23 April 2020; pp. 139–145.
11. Jane, E.M.; Raj, E. SBKMMA: Sorting based K means and median based clustering algorithm using multi machine technique for big data. *Int. J. Comput. (IJC)* **2018**, *28*, 1–7.
12. Dafir, Z.; Lamari, Y.; Slaoui, S.C. A survey on parallel clustering algorithms for big data. *Artif. Intell. Rev.* **2021**, *54*, 2411–2443. [[CrossRef](#)]
13. Ibrahim Hayatu, H.; Mohammed, A.; Barroon Isma'eel, A. Big Data Clustering Techniques: Recent Advances and Survey. In *Machine Learning and Data Mining for Emerging Trend in Cyber Dynamics*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 57–79.
14. Heidari, S.; Alborzi, M.; Radfar, R.; Afsharkazemi, M.A.; Rajabzadeh Ghatari, A. Big data clustering with varied density based on MapReduce. *J. Big Data* **2019**, *6*, 1–16. [[CrossRef](#)]
15. Azhir, E.; Navimipour, N.J.; Hosseinzadeh, M.; Sharifi, A.; Darwesh, A. An efficient automated incremental density-based algorithm for clustering and classification. *Future Gener. Comput. Syst.* **2021**, *114*, 665–678. [[CrossRef](#)]
16. Li, Y.; Yang, Z.; Han, K. k-Means Parallel Algorithm of Big Data Clustering Based on Mapreduce PCAM Method. *Int. J. Eng. Intell. Syst.* **2021**, *29*, 674–679.
17. Hosseini, B.; Kiani, K. A robust distributed big data clustering-based on adaptive density partitioning using apache Spark. *Symmetry* **2018**, *10*, 342. [[CrossRef](#)]
18. Wang, D.; Zhou, F.; Li, J. Cloud-based parallel power flow calculation using resilient distributed datasets and directed acyclic graph. *J. Mod. Power Syst. Clean Energy* **2019**, *7*, 65–77. [[CrossRef](#)]
19. Daghistani, T.; AlGhamdi, H.; Alshammari, R.; AlHazme, R.H. Predictors of outpatients' no-show: Big data analytics using Apache Spark. *J. Big Data* **2020**, *7*, 1–15. [[CrossRef](#)]
20. Dierckens, K.E.; Harrison, A.B.; Leung, C.K.; Pind, A.V. A data science and engineering solution for fast k-means clustering of big data. In Proceedings of the 2017 IEEE Trustcom/BigDataSE/ICSS, Sydney, NSW, Australia, 1–4 April 2017; pp. 925–932.
21. Mittal, K.; Aggarwal, G.; Mahajan, P. Performance study of K-nearest neighbor classifier and K-means clustering for predicting the diagnostic accuracy. *Int. J. Inf. Technol.* **2019**, *11*, 535–540. [[CrossRef](#)]
22. Zhang, G.; Zhang, C.; Zhang, H. Improved K-means algorithm based on density Canopy. *Knowl.-Based Syst.* **2018**, *145*, 289–297. [[CrossRef](#)]
23. Kumar, S.; Singh, M. A novel clustering technique for efficient clustering of big data in Hadoop Ecosystem. *Big Data Min. Anal.* **2019**, *2*, 240–247. [[CrossRef](#)]
24. Caruso, G.; Gattone, S.A.; Balzanella, A.; Battista, T.D. Cluster analysis: An application to a real mixed-type dataset. In *Models and Theories in Social Systems*; Springer: Berlin/Heidelberg, Germany, 2019; Volume 179, pp. 525–533.
25. Xie, T.; Liu, R.; Wei, Z. Improvement of the Fast Clustering Algorithm Improved by-Means in the Big Data. *Appl. Math. Nonlinear Sci.* **2020**, *5*, 1–10. [[CrossRef](#)]
26. Lu, W. Improved K-means clustering algorithm for big data mining under Hadoop parallel framework. *J. Grid Comput.* **2020**, *18*, 239–250. [[CrossRef](#)]
27. Moodi, F.; Saadatfar, H. An improved K-means algorithm for big data. *IET Softw.* **2021**, *16*, 48–59. [[CrossRef](#)]
28. Shang, R.; Ara, B.; Zada, I.; Nazir, S.; Ullah, Z.; Khan, S.U. Analysis of simple K-mean and parallel K-mean clustering for software products and organizational performance using education sector dataset. *Sci. Program.* **2021**, *2021*, 9988318. [[CrossRef](#)]
29. Fojtik, R. New Processor Architecture and Its Use in Mobile Application Development. In Proceedings of the 2018 International Conference on Digital Science, Budva, Montenegro, 19–21 October 2018; Springer: Berlin/Heidelberg, Germany, 2021; pp. 545–556.
30. Goodacre, J.; Sloss, A.N. Parallelism and the ARM instruction set architecture. *Computer* **2005**, *38*, 42–50. [[CrossRef](#)]