PERFORMANCE COMPARISON OF K-MEANS, PARALLEL K-MEANS AND K-MEANS++

Ramiz Aliguliyev[1], Shalala F. Tahirzada[1]

•

[1]Ministry of Science and Education Republic of Azerbaijan, Information Technology Institute, Baku, Azerbaijan
r.aliguliyev@gmail.com, fmv.shalala@gmail.com

Abstract

K-means clustering is a fundamental unsupervised machine learning technique widely applied in various domains such as data analysis, pattern recognition, and clustering-based tasks. However, its efficiency and scalability can be challenged, particularly when dealing with large-scale datasets and complex data structures. This thesis explores strategies to improve the performance of the K-means clustering algorithm through parallelism and iterative techniques. Parallelism leverages modern parallel computing architectures, including multi-core processors and distributed frameworks like Apache Spark, to enhance computational efficiency and scalability. On the other hand, an iterative approach involves refining clustering results through multiple iterations, adjusting cluster centroids, and optimizing convergence criteria. It delves into the design frameworks of these approaches, highlighting their respective advantages and limitations.

Comparative analyses are conducted to evaluate the effectiveness of parallelism and iterative techniques in terms of execution time, scalability, clustering accuracy, and convergence speed. The findings contribute to advancing the understanding of how parallelism and iterative strategies can significantly improve K-means clustering performance, especially in the context of big data and complex datasets. By comparatively analyzing parallelism and iterative approaches, this paper aims to contribute to the development of more efficient and scalable clustering algorithms in the Big Data context.

Keywords: k-means clustering, big data, parallel computing, iterative techniques, computational efficiency, performance enhancement.

I. Introduction

As a result of high development trends, big data has become an integral part of our lives. The widespread use of Information and Communication Technology (ICT) and the application of new technological concepts such as the Internet of Things (IoT) has created a global revolution across various fields.

Often referred to as the "new oil," the exponential increase in big data presents significant challenges such as preprocessing, analysis, and real-world business applications. According to the recent report from Edge Delta [1], considering that large-scale data is being generated at every moment “Figure 1”, it is projected that the volume of big data will reach 181 zettabytes by 2025 “Figure 2”.

Time	Amount of data created
Daily	0.33 zettabytes
Weekly	2.31 zettabytes
Monthly	10 zettabytes
Annually	120 zettabytes

Figure 1: *The generated data volume over various intervals*

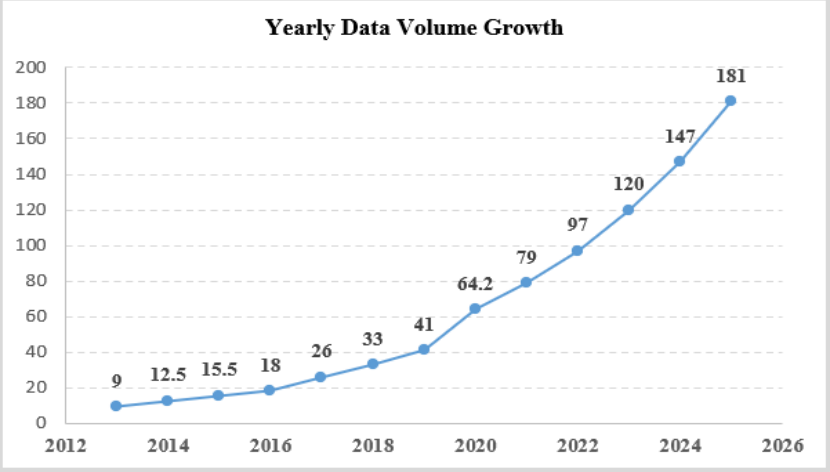

Figure 2: *Trends in global data volume (In Zettabytes)*

On the other hand, the revenues generated from big data have reached significant figures, highlighting its value and the potential insights it can provide "Figure 3". This realization has led to an increased demand for managing and analyzing big data. As the big data market continues to expand rapidly, this trend is expected to persist. Therefore, the implementation of new technologies to automate processes by increasing efficiency in the various fields is considered as one of the necessary requirements.

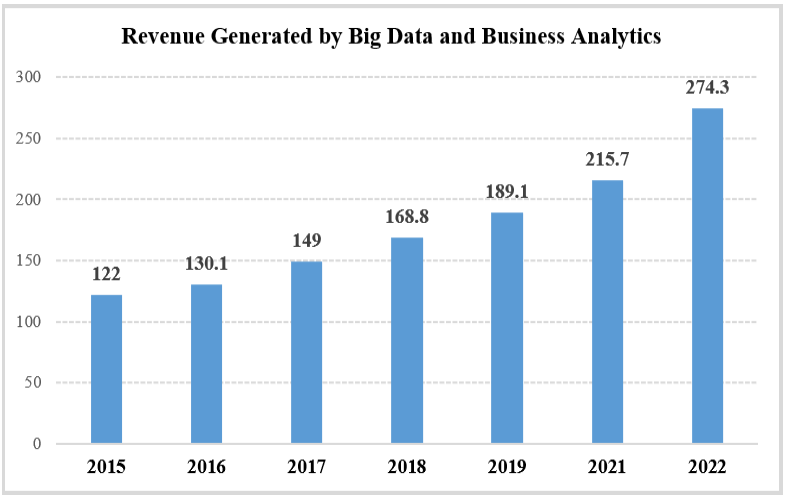

Figure 3: *Trends in Data Market Growth (In Billion)*

In this context, developing big data processing methods at a pace that matches technological advancement and improving existing methods is crucial. Hybrid processing of traditional methods with new approaches for the analysis of big data will increase the effectiveness and mitigate the conflicts of these methods. Given the urgency of the problem, this article investigates the application of iterative and parallel approach methods to traditional clustering techniques through experimental analysis.

The article is structured as follows. In Section 2, K-means clustering and brief overview, in Section 3 information about parallel and iterative approaches and its implementation to K-means in literature is given. Section 4 deals with experimental results and discussion, Section 5 concludes our article.

II. K-means Clustering: A Brief Overview

A.The K-means Algorithm and Applications

K-means clustering is widely used in fields such as data mining and pattern recognition. The K-means algorithm forms clusters by utilizing the mean value of objects within each cluster. In its standard form, the algorithm requires the number of clusters to be specified by the user, which is then used to randomly select initial cluster centers from the dataset and works by assigning data points to clusters near a predetermined centroid and then recalculating new centroids for each cluster. This iterative process continues until the centroids stabilize. To achieve the best cluster results, it is necessary to run the algorithm multiple times with different initial cluster centers for a given number of clusters. Additionally, the standard K-means algorithm primarily identifies spherical or ball-shaped clusters because it relies on the Euclidean distance metric.

B.Advantages of Standart K-means

Its simplicity and speed make it suitable for large datasets.

It often performs well on low-dimensional data and can generalize to high-dimensional data.

Specifying the number of clusters beforehand allows for better management of analyses.

C.Limitations of Standart K-means

- Due to its greedy nature, the K-means algorithm may converge to a local minimum, requiring multiple runs with different initial cluster centers to find the optimal result.
- Determining the number of clusters beforehand is challenging, as it requires knowledge of the data's inherent structure.
- The algorithm primarily identifies spherical clusters due to its reliance on the Euclidean distance metric.

Outliers and non-spherical cluster structures can affect the accuracy of K-means; in such cases, alternative clustering techniques may be more appropriate.

III. Parallel and Iterative Approaches in Literature

A.Parallel Processing Methods

Given the ease of data generation and the proliferation of technologies like IoT, parallel processing is a vital component of any microservices delivery. Simply put, parallel processing is a computational method in which multiple calculation or data processing tasks occur simultaneously, utilizing several central processing units (CPUs) working concurrently.

This approach can significantly reduce the execution time of a program by distributing multiple components of the task across various processors or CPUs. Multi-core processors, frequently found in modern computers, and any system with more than one CPU can perform parallel processing. The various varieties of parallel processing exist such as MPP, SIMD, MISD, SISD, and MIMD:

1)MPP (Massively Parallel Processing): involves a large number of processors working in parallel on a single task or set of tasks. It's often used in supercomputing environments. MPP can be applied in clustering tasks to handle massive datasets and complex algorithms. For example, in hierarchical clustering, ***MPP*** can accelerate the computation of pairwise distances between data points in large datasets.

2)SIMD (Single Instruction, Multiple Data)*:* executes the same instruction on multiple data points simultaneously. It's commonly used in vector processing and GPU computations. SIMD can be utilized in clustering algorithms like K-means to perform simultaneous calculations on multiple data points, improving the algorithm's efficiency, especially in high-dimensional spaces.

3)MISD (Multiple Instruction, Single Data): involves multiple processors executing different instructions on the same data stream. It's a less common form of parallel processing. MISD is not commonly applied in clustering tasks due to its rarity and complexity. Clustering algorithms typically do not require different instructions on the same data stream.

4)SISD (Single Instruction, Single Data): is the traditional sequential processing where a single processor executes a single instruction on a single data stream at a time. SISD is not directly applicable to clustering tasks as it lacks parallelism, which is crucial for efficient clustering algorithms.

5)MIMD (Multiple Instruction, Multiple Data): involves multiple processors executing different instructions on different data streams concurrently. It's commonly used in distributed computing and multi-core processors. MIMD is highly applicable in clustering tasks, especially for parallelizing computations in algorithms like DBSCAN, where multiple processors can handle different regions of the dataset concurrently, improving scalability and performance.

MPP, SIMD, and MIMD are the most relevant types of parallel processing for clustering tasks, with MIMD being particularly effective in parallelizing computations and improving scalability in clustering algorithms. There are several implementations of parallel K-means algorithms, each utilizing different parallel computing frameworks and techniques. Parallel K-means based on MapReduce, Parallel K-means using MPI can be example algorithms mentioned:

1)Parallel K-means using MPI: is a standard established by the Message Passing Interface Forum [2]. It is a premier tool for implementing parallel computing in distributed memory environments due to its comprehensive standard libraries. The first version of MPI was released in 1994, followed by MPI-2 in 1997 [3]. MPI supports both collective and point-to-point communication, offering a wide array of functions that facilitate communication between different clusters in a parallel development environment [4].

2)Parallel K-means based on MapReduce: As stated in [5] The MapReduce programming model divides the initial center point selection process into two jobs for parallel computation. In Job1, the nearest and farthest points are calculated, and these results are passed to Job2, which assigns values to clusters based on the smallest distance center point. This iterative process continues until the specified number of clusters is reached, calculating the average of clusters to determine the initial center point. The k-means algorithm's good locality allows it to be effectively parallelized. In the first stage, input files are segmented and processed by Mapper nodes. In the second stage, clustering operations are performed by Reduce nodes, partitioning intermediate results and generating final cluster outputs. The overall process involves executing map and reduce functions to calculate and update cluster points iteratively until the desired clustering is achieved “Fig. 4”.

B.Iterative Approachs

Iterative approaches for clustering involve refining the clustering results through a series of iterations. These approaches are used to improve the quality of clustering results, handle complex datasets, and address challenges such as cluster size imbalance, sensitivity to initialization, and the presence of outliers or noise. Some of the most popular iterative approaches for k means are listed below:

1)Iterative K-means minus–plus (I-k-means–+): [6] introduces an iterative strategy to enhance the quality of results generated by the k-means algorithm. Known as iterative k-means minus–plus (I-k-means–+), this approach aims to iteratively refine the solution by removing one cluster, splitting another cluster, and then re-clustering the data in each iteration. To expedite this process, the I-k-means–+ method employs techniques to determine which cluster to remove, which one to split, and how to accelerate the re-clustering step. Experimental results indicate that I-k-means–+ can surpass k-means++, a well-regarded variant of k-means, in terms of minimizing Sum of Squared Euclidean Distances (SSEDM). Additionally, I-k-means–+ exhibits reasonable runtime compared to k-means, making it a promising advancement in optimizing k-means clustering.

2)Mini Batch K-means*:* forms a minibatch consisting of a collection of small randomized data with a constant size enable to be stored in a memory. The mechanism is that the sample is taken randomly from the dataset to form a minibatch, and then it is assigned to the nearby centroid. In the second step, the centroid is updated and so on [7].

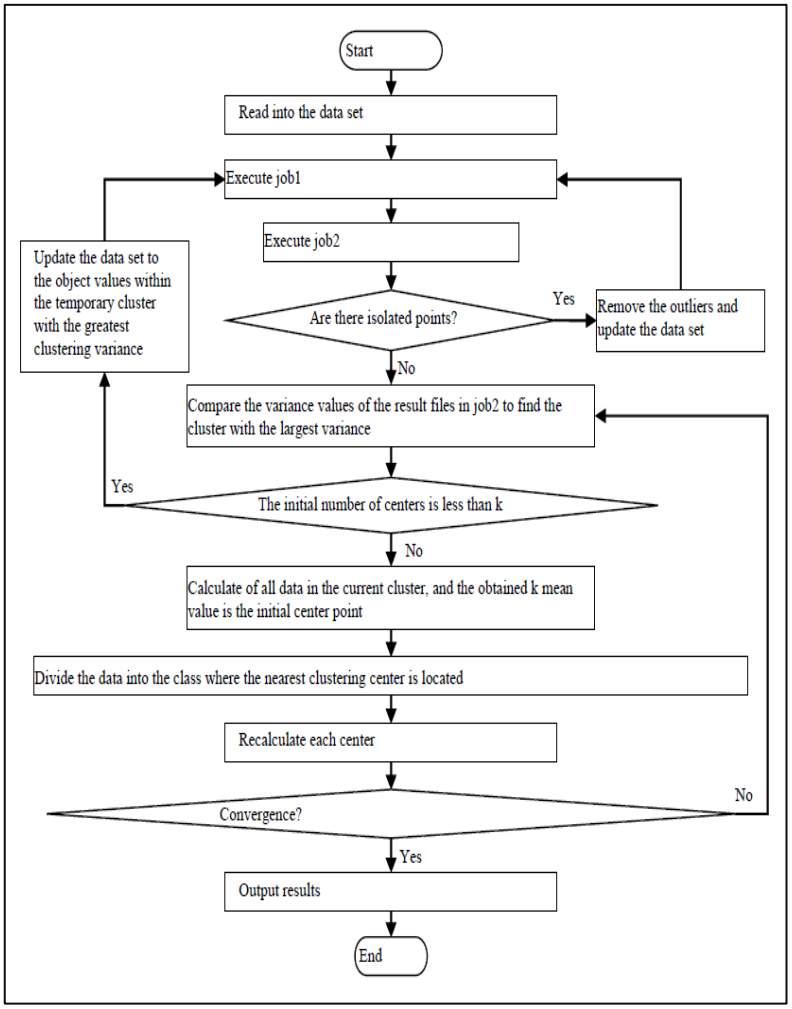

Figure 4: *K-means clustering process based on MapReduce distributed programming framework*

IV. Experimental Results

In this paragraph, we will delve into a comprehensive exploration of the implementation outcomes of parallel and iterative methodologies, both based on the standard k-means clustering algorithm. The parallel version of the algorithm focus on concurrently assigning data points to

clusters and computing new centroids during each iteration. Iterative K-means++ iteratively selects initial cluster centers in a way that spreads them out, which can help the algorithm converge more quickly and avoid poor local minima. It is an initialization algorithm for the K-means clustering algorithm that improves the quality of the final clusters. The results will be analyzed in more detailed based on a few evaluation metrics (F1 Score, Precision, Recall, Accuracy and Execution Time).

A. Dataset Description

Each record in the Poker Hand dataset represents a five-card hand drawn from a standard 52-card deck [8]. Each card is described by two attributes: suit and rank, resulting in 10 predictive attributes per hand. Additionally, there is a class attribute that indicates the type of "Poker Hand." Since the order of cards is significant, there are 480 possible Royal Flush hands compared to just 4 if the order were not considered. Each hand is represented by 10 predictive attributes and one class attribute that categorizes the hand into one of ten possible poker hands. Each row in the dataset corresponds to a single poker hand, using both categorical and ordinal attributes to describe the hand and predict its classification.

B. Implementation Setup

The study utilizes a system featuring 4 cores and 8 logical processors, powered by an 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz processor. It includes 16.0 GB of RAM and operates on Python version 3.11.4.

Table 1: *Experimental Results for K-means, Parallel K-means, and K-means++ on the Poker Hand dataset*

Cluster number	Algorithms	Precision	Recall	Accuracy	F1 score
	parallel	0.0974	0.0983	0.3303	0.0864
k=3	***kmeans++***	0.0971	0.0991	0.3256	0.0856
	kmeans	0.1012	0.1052	0.3228	0.0870
	parallel	0.1000	0.1048	0.2016	0.0687
k=5	***kmeans++***	0.0994	0.1024	0.1980	0.0678
	kmeans	0.1032	0.1038	0.2072	0.0710
	parallel	0.1011	0.0946	0.1416	0.0554
k=7	***kmeans++***	0.1004	0.0981	0.1435	0.0559
	kmeans	0.1011	0.1018	0.1473	0.0561
	parallel	0.0989	0.0954	0.1107	0.0467
k=9	***kmeans++***	0.1008	0.0967	0.1128	0.0485
	kmeans	0.1094	0.0979	0.1196	0.0508
	parallel	0.0927	0.0765	0.0935	0.0382
k=11	***kmeans++***	0.0891	0.0737	0.0889	0.0373
	kmeans	0.0919	0.1296	0.0901	0.0374

C. Result and Discussion

Due to its limitations, the K-means algorithm is almost impossible to apply to very large-scale data. If we pay attention to the experimental result in Table I, we can see here that the results of parallel and iterative algorithms are not behind K-means, and even show the same or better results in different numbers of k in some metrics. This means that we can apply these alternative approaches for the larger data that we want to apply the K-means algorithm to, and we can find an answer to the question of how K-means can be applied to large data sets. Looking at the results of the algorithms with parallel and iterative approaches, it is possible to see that although the F1 value is low, the Recall and Precision values are higher, that is, the algorithms predict the classes better.

In terms of execution time in Table II, although the iterative algorithm shows better results for smaller numbers of k, these results are almost the same as the parallel one for larger scale data or larger numbers of k. By combining the advantages of both approaches in a hybrid model, significantly better clustering results can be achieved. This suggests that the hybrid approach will deliver more effective outcomes, depending on the statistical properties and scalability of the data.

Table 2: *Model Evaluation based on Execution Time*

Cluster number	Parallel K-means	K-means++
k=3	22.36	11.09
k=5	40.409	21.5901
k=7	43.8295	22.0569
k=9	46.8376	38.1536
k=11	47.6508	42.7741

V. Conclusion and future work

Since the paper investigates the feasibility of applying alternative k-means modifications in scenarios where standard k-means cannot be implemented, the primary focus is on the qualitative comparison of the experimental results (Table I). K-means++ seems to be a better choice for a lower number of clusters, offering significantly better performance compared to Parallel K-means. However, as the number of clusters increases, the performance of both algorithms converges. Therefore, it is possible to obtain much better results by using these algorithms in a hybrid manner for very large-scale data. Depending on the characteristics of the data, it may be necessary to benefit from these two types of approaches and, if necessary, use hybrid methods. At the same time, using other modification methods based on these approaches for K-means, or preparing a structure specific to the data and model, allows us to obtain clusters with much better final results.

In the future, we plan to consider the problem for different modification algorithms based on K-means and other methods, where we apply a parallel iterative approach using a hybrid method. Additionally, the effective selection of initial centroids and the best number of clusters, which remains an important clustering issue, will help improve the accuracy of the parallel and iterative clustering algorithm.

This work was supported by the Azerbaijan Science Foundation - **Grant No. AEF-MCG-2023-1(43)-13/04/1-M-04.**

References

[1] Edge Delta. (2024, March 22). *Insightful Statistics on Data Market Size and Forecast 2024*. [Online]. Available at: https://edgedelta.com/company/blog/data-market-size-and-forecast/

[2] M. Subramaniam, S.M. Jai Sakthi, C. Aravindan, Strategies for Parallelizing K-Means Data Clustering Algorithm, in: *Communications in Computer and Information Science*, vol. 147, Springer, 2011, pp. 427-430.

[3] W. Gropp, E. Lusk, *Implementing MPI: The 1994 MPI Implementors' Workshop*, IEEE, 1994, pp. 55-59.

[4] J. Bhimani, M. Leeser, N. Mi, *Accelerating K-Means Clustering with Parallel Implementations and GPU Computing*, in: *Proceedings of the IEEE High Performance Extreme Computing Conference (HPEC)*, 2015, pp. 1-6.

[5] L. Zhang, *Research on K-means Clustering Algorithm Based on MapReduce Distributed Programming Framework*, *Procedia Computer Science*, vol. 228, 2023, pp. 262-270.

[6] H. Ismkhan, *I-k-means+: An Iterative Clustering Algorithm Based on an Enhanced Version of the k-means*, *Pattern Recognition*, vol. 79, 2018, pp. 402-413.

[7] F. Rachman, H. Santoso, A. Djajadi, *Machine Learning Mini Batch K-means and Business Intelligence Utilization for Credit Card Customer Segmentation*, *International Journal of Advanced Computer Science and Applications*, vol. 12, 2021, pp. 218-227.

[8] Cattral, R., & Oppacher, F. (2002). *Poker Hand* [Dataset]. UCI Machine Learning Repository. https://doi.org/10.24432/C5KW38.