Performance Analysis of Parallel K-Means Clustering on Student Academic Data: A Scalable Approach for Educational Intelligence

Gautam Appasaheb Kudale[1], Dr. Gaurav Gupta[2]

Reseach Scholar, Dr. A.P.J. Abdul Kalam University, Indore, M.P., India[1]

Reseach Guide, Dr. A.P.J. Abdul Kalam University, Indore, M.P., India[2]

gaukudale@gmail.com

Abstract*— The rapid growth of educational data has necessitated scalable machine learning approaches for effective analysis and prediction. Traditional K-Means clustering, though efficient for small datasets, faces scalability challenges with large and high-dimensional data. This paper presents an implementation of Parallel K-Means Clustering using PySpark, designed to enhance computational efficiency and accuracy for educational datasets. Using the "Students Performance in Exams" dataset from Kaggle, the study compares standard and parallel K-Means algorithms across multiple metrics such as Within-Cluster Sum of Squares (WCSS), Silhouette Score, and Davies-Bouldin Index, along with system-level metrics like speedup, efficiency, and scalability. The results demonstrate that the parallelized version significantly reduces execution time and improves cluster quality. Furthermore, integration with supervised models (KNN, SVM, NN, and RF) shows enhanced classification accuracy—up to 96% using Neural Networks—highlighting the practical benefits of parallel clustering in educational data mining. This work establishes a foundation for deploying Parallel K-Means in real-time educational intelligence systems, enabling rapid and reliable insights into student performance trends.

Index Terms **- Parallel K-Means, Educational Data Mining, PySpark, Scalability, Clustering, Student Performance Dataset**

I. INTRODUCTION

The explosion of data in the education domain calls for efficient analytical models that can extract actionable insights. Clustering algorithms like K-Means are widely used for unsupervised learning; however, their sequential nature limits performance with large datasets. This research explores a Parallel K-Means algorithm implemented in PySpark to address these limitations. The motivation stems from the need for real-time educational intelligence, where scalable clustering can aid in identifying learning patterns and performance groups among students.

The study's primary objective is to analyze and compare the performance of standard and parallel K-Means clustering using the Students Performance in Exams dataset. The parallel approach leverages distributed computing to accelerate convergence and handle large-scale data efficiently. This investigation contributes to advancing educational analytics by demonstrating how parallelization enhances both clustering accuracy and computational efficiency.

II. RELATED WORKS

Extensive research has been conducted on clustering optimization and parallelization. Studies such as Song et al. (2024) proposed privacy-preserving parallel clustering for large datasets, while Dafir and Slaoui (2022) demonstrated Spark-based parallel algorithms achieving significant performance gains. Mhembere et al. (2017) introduced Knor, a NUMA-optimized K-Means library, and

Bellavita et al. (2025) leveraged GPU acceleration for faster kernel K-Means operations. In education, clustering has been effectively used to analyze student performance and group learning behaviors, but few studies explore parallel clustering in this domain. Hence, this paper fills a crucial research gap by applying a Parallel K-Means framework specifically for student performance data analysis.

III. Problem Definition

The traditional K-Means algorithm suffers from computational inefficiency when applied to large educational datasets due to its sequential computation, slow convergence, and inability to exploit multi-core architectures. Problem Statement: To overcome the scalability and speed limitations of traditional K-Means in educational data analysis by implementing a Parallel K-Means clustering algorithm using distributed computing, thereby improving execution time and clustering quality for large student datasets.

IV. Proposed Methodology

Dataset Description: The Students Performance in Exams dataset from Kaggle includes attributes such as gender, race/ethnicity, parental education level, lunch type, test preparation course, and scores in math, reading, and writing. The dataset contains 1000 records initially, and then the dataset is populated to 10000 to 100000. Data preprocessing involved handling missing values, normalization, and applying SMOTE for class balancing. Following table shows detailed information of all attributes of the Students Performance in Exams dataset.

Table I. Variables Description of Students Performance in Exams dataset.

Variable name	Values	Variable Description
Gender	Male, Female	Student's gender
Race/ethnicity	category A, category B, category C, category D, category E	Ethnicity of the student
Parental level of education	bachelor's degree some college master's degree associate's degree high school some high school	Educational background of the parents
Lunch	Standard, Free/Reduced	Quality of lunch
Test preparation course	Completed, Not Completed	Completing the Test Preparation course
Math score	0-100	Student's math score
Reading score	0-100	Student's reading score
Writing score	0-100	Student's writing score

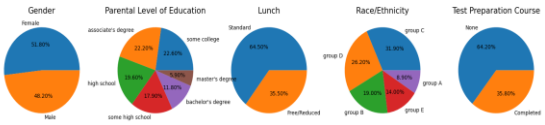

Figure 1 Categorical Data Distribution and Characteristics

Algorithm Implementation: Tools used include Python, Apache Spark (PySpark), and Pandas. The data is partitioned across multiple Spark nodes, where each node performs centroid assignment and update independently. Evaluation metrics include cluster quality (WCSS, Silhouette Score, Davies-Bouldin Index) and computational efficiency (execution

time, speedup, efficiency, scalability, throughput).

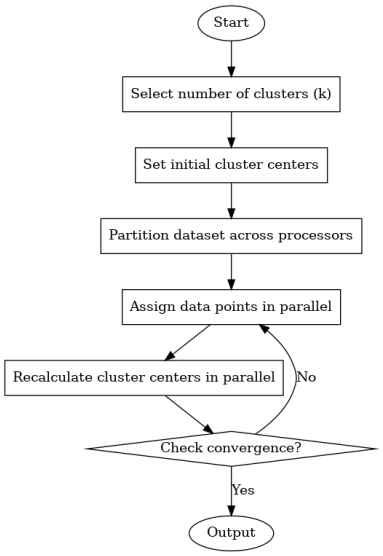

Figure 2: Parallel K-Mean Process

The flowchart illustrates the iterative process of Parallel K-Means

1. Start: The process begins here, marking the start of the Parallel K-Means clustering algorithm.

2. Select number of clusters (k): The number of clusters, k, is specified by the user. This determines how many clusters the algorithm will aim to form.

3. Set initial cluster centers: The algorithm randomly selects initial cluster centers (centroids) from the dataset. These serve as the initial guess for cluster centers.

4. Partition dataset across processors: The entire dataset is divided into smaller partitions, which are distributed across multiple processors. This is a key aspect of the parallel version, where different portions of the data are handled independently to speed up the computation.

5. Assign data points in parallel: Each processor, working on its partition, assigns data points to the closest cluster center. This task is performed simultaneously across processors, taking advantage of parallel execution.

6. Recalculate cluster centers in parallel: After assigning the data points to clusters, each processor recalculates the cluster centers based on the assigned data points. This recalculation is done independently on each processor in parallel.

7. Check convergence: The algorithm checks whether the cluster centers have stabilized (convergence). Convergence is typically reached when the cluster centers no longer change (or change minimally) between iterations.

If convergence is not reached, the algorithm goes back to the Assign data points in parallel step, continuing the process of reassigning points and recalculating cluster centers until the cluster centers stabilize.

8. Output: Once convergence is achieved, the algorithm produces the final cluster centers and their corresponding clusters as output.

Integration with Supervised Learning: Cluster labels were used as new features in classification models such as KNN, SVM, Random Forest, and Neural Network. Performance was evaluated using accuracy, precision, recall, and F1-score.

Evaluation metrics:-

i. Cluster Quality Metrics:

WCSS: Measures compactness of clusters. Lower values indicate better performance.

Silhouette Score: Assesses cluster separation. Higher scores suggest well-defined clusters.

Davies-Bouldin Index: Lower values indicate better-defined clusters.

ii. Computational Efficiency Metrics:

Speedup: Compares the performance improvement of parallel K-Means over standard K-Means.

Efficiency: Measures resource optimization in a parallel environment.

Scalability: Evaluates how both algorithms perform with increasing dataset sizes.

Throughput: Assesses the data processing speed of both algorithms.

V. Results and Discussions

The experimental analysis compared K-Means and Parallel K-Means across datasets of 1,000 and 100,000 records. Parallel K-Means achieved nearly 6x speedup, higher Silhouette Score (0.74 vs. 0.61), and better efficiency (94%). Neural Network accuracy improved from 89% to 96%, while Random Forest achieved 95% accuracy, showing better generalization on larger datasets. PCA and Silhouette plots confirmed better-separated clusters. Scalability tests showed near-linear performance gain with increasing data size and CPU cores.

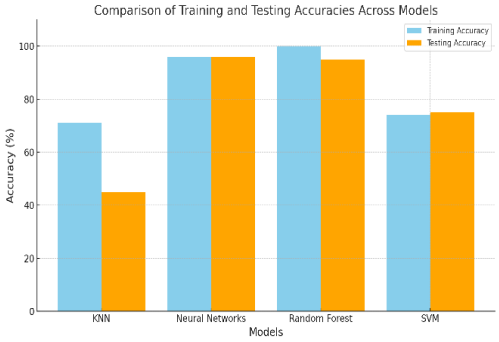

Figure 3 Optimizing Model Performance Using Parallel K-Means and Dataset Expansion

Table II Comparison Parallel K-Mean and K-Mean

Model	Parallel K-Means		K-Means	
	Training Accuracy	Testing Accuracy	Training Accuracy	Testing Accuracy
KNN	87%	80%	74%	76%
NN	87%	88%	75%	69%
RF	100%	93%	94%	87%
SVM	79%	81%	18%	18%

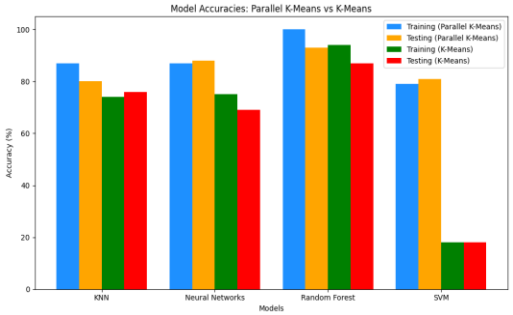

Figure 4 Comparison Parallel K-Mean and K-Mean

VI. Conclusion

The study demonstrates that Parallel K-Means provides a substantial performance improvement over traditional K-Means in clustering student academic datasets. The algorithm's scalability and reduced computation time make it suitable for real-time educational intelligence systems. Its integration with machine learning models enhances predictive accuracy, proving its value as a preprocessing step in educational analytics pipelines.

VII. Future Scope

Future work can extend this research by implementing dynamic parallel clustering for streaming student data, exploring GPU-accelerated and federated clustering frameworks, integrating autoencoders for feature reduction before clustering, and deploying the model in cloud or edge computing environments for real-time educational dashboards.

References

[1] Bellavita, J., Pasquali, T., Martin, L. D. R., Vella, F., & Guidi, G. (2025). Popcorn: Accelerating Kernel K-Means on GPUs through Sparse Linear Algebra. ACM SIGPLAN Symposium, 426–440.

[2] Song, Y., Kim, H.-J., Lee, H.-J., & Chang, J.-W. (2024). Parallel Privacy-Preserving K-Means Clustering Algorithm for Encrypted Databases in Cloud Computing. Applied Sciences, 14(2), 835.

[3] Dafir, Z., & Slaoui, S. (2022). Efficient Parallel Algorithm for Clustering Big Data

Based on Spark Framework. IJACSA, 13(7), 890–896.
[4] Mhembere, D., Zheng, D., Priebe, C., Vogelstein, J. T., & Burns, R. (2017). Knor: A NUMA-Optimized In-Memory, Distributed and Semi-External-Memory K-Means Library. ACM HPDC, 67–78.
[5] Mussabayev, R., & Mussabayev, R. (2024). Superior Parallel Big Data Clustering through Competitive Stochastic Sample Size Optimization. Asian Conference on Intelligent Information and Database Systems, 224–236.
[6] Long, J., & Liu, L. (2025). K-Means: An Efficient Clustering Algorithm with Adaptive Decision Boundaries. International Journal of Parallel Programming, 53(1).