Optimizing K-Means Clustering Using CUDA and OpenMP for Enhanced Computational Efficiency

Guozheng Wu
School of Information, Renmin University of China
Beijing, China
2021201543@ruc.edu.cn

Abstract—This thesis investigates the optimization of the k-means clustering algorithm using CUDA (Compute Unified Device Architecture) and OpenMP (Open Multi-Processing), targeting improvements in computational efficiency and scalability for large dataset analysis in data mining and machine learning. This thesis integrate CUDA to harness GPU computing and OpenMP for CPU multi-threading, creating a hybrid approach that enhances k-means performance. Through comprehensive experiments, this thesis assess key performance metrics like execution time, scalability, and accuracy. Our findings show notable improvements in processing speed and efficiency, demonstrating the advantage of combining CUDA and OpenMP in k-means optimization. This work provides valuable insights into the practical application of parallel computing techniques in data analysis and contributes significantly to the field by offering an optimized k-means solution for real-world data challenges.

I. Introduction

Facing the challenges of big data, this thesis optimizes the k-means clustering algorithm, a cornerstone in data mining and machine learning, using CUDA (Compute Unified Device Architecture) and OpenMP (Open Multi-Processing). This research primarily investigates the performance differences between CPU and GPU programming, leveraging CUDA and OpenMP to enhance k-means across various platforms.

The study provides a comparative analysis of k-means under these parallel computing frameworks, emphasizing optimization strategies tailored to the unique architectures of CPUs and GPUs. Beyond performance gains, the research delves into scalability and accuracy improvements, aiming to contribute novel insights to parallel computing in data algorithms.

The experimental scope of this thesis is comprehensive yet concise, evaluating optimizations on diverse datasets and computational settings. The results are expected to offer significant academic and practical advancements in data processing and machine learning efficiency.

II. Related Work

In the fields of data mining and machine learning, the optimization of the k-means clustering algorithm has garnered widespread attention, especially in the context of parallel computing frameworks like CUDA and OpenMP. Daoudi et al. (2019) compared different implementations of the k-means algorithm on CPU and GPU platforms, finding that GPU parallel algorithms performed better with large datasets, whereas OpenMP implementations were more suitable for smaller datasets [1]. Kruliš and Kratochvíl (2020) explored performance optimizations of the CUDA k-means algorithm on modern GPU accelerators, proposing several enhancements that significantly improved algorithm efficiency [2]. Yang et al. (2020) achieved significant speed improvements in the k-Means algorithm through CUDA GPU programming [3]. These studies provide crucial technical insights for processing large-scale data.

III. Methodology

The k-means algorithm, recognized for its efficacy in unsupervised classification, presents numerous opportunities for parallel processing, making it a prime candidate for acceleration techniques. This research explores the acceleration of k-means using two distinct parallel computing methodologies: OpenMP and CUDA. Each method harnesses different aspects of parallelism inherent in the k-means algorithm.

A. OpenMP Acceleration

Leveraging the multi-threading capabilities of OpenMP, our approach focuses on parallelizing the critical steps of the k-means algorithm on a CPU. This involves distributing the task of calculating distances between data points and their nearest cluster centers across multiple threads, significantly reducing computational time. Additionally, the re-calculation of cluster centroids after each iteration is also parallelized, allowing simultaneous updates that enhance overall efficiency. This approach capitalizes on the CPU's ability to handle multiple threads efficiently, thereby improving the execution speed of the k-means algorithm.

B. CUDA Implementation

The CUDA-based acceleration targets the GPU's parallel processing strengths. Given the high degree of parallelism in GPUs, this thesis map the computationally

intensive elements of the k-means algorithm, such as distance calculations and cluster reassignment, to the GPU's cores. This methodology is particularly effective in handling large datasets where the sheer volume of data can be processed more rapidly in parallel. The CUDA implementation aims to utilize the extensive computational power of GPUs to significantly reduce the time required for clustering large datasets.

C. Experimentation and Performance Metrics

Both OpenMP and CUDA implementations are subjected to rigorous testing. This thesis conduct experiments across a variety of datasets, assessing performance based on execution time, scalability, and the accuracy of clustering. These metrics provide insights into the efficiency gains from each parallel computing technique, highlighting their respective strengths in optimizing the k-means algorithm.

IV. experiment

A. Data Preparation

For my experiments, I generated various sizes of two-dimensional datasets randomly. This approach allowed us to systematically evaluate the performance impact of our optimizations across a consistent range of controlled scenarios.

B. Experimental Results

The experimental outcomes, as illustrated in the accompanying graph, indicate that both OpenMP and GPU (CUDA) implementations significantly accelerate the computation of the k-means algorithm. Specifically, the performance enhancements achieved by OpenMP and GPU are quantified by orders of magnitude, with OpenMP reaching an order of magnitude of 10^1–10 and GPU scaling up to 10^2–10^3

in terms of acceleration ratio.

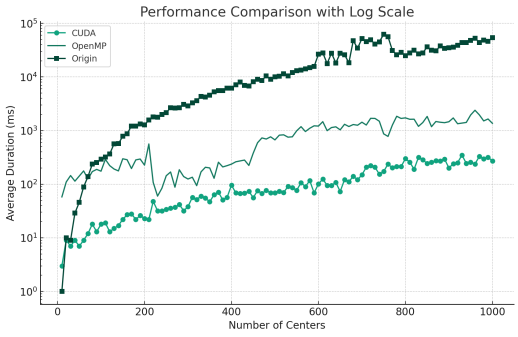

Fig. 1. Performance comparison with Log scale

The graph, which employs a logarithmic scale for clarity, shows that as the number of centers increases, the average duration of computation for the original algorithm scales steeply, whereas the OpenMP and GPU enhanced versions maintain a more gradual increase in computation time. Notably, the GPU version demonstrates a superior performance, maintaining lower average computation times across all numbers of centers. This is indicative of the massive parallel processing capabilities of GPUs that are well-exploited by the CUDA framework.

Furthermore, the OpenMP implementation also shows a marked improvement over the original algorithm, with the multi-threading capability effectively reducing computation times, particularly noticeable as the number of centers increases.

These results underscore the potential for utilizing parallel computing techniques to optimize the efficiency of the k-means algorithm. They provide compelling evidence that both OpenMP and GPU acceleration can yield significant performance benefits, particularly in scenarios with large numbers of clusters where the computational burden is greatest.

V. Conclusion

This study has thoroughly examined the acceleration of the k-means clustering algorithm using OpenMP and CUDA, two key parallel computing technologies. The results show that both significantly enhance computational efficiency and scalability, with CUDA's GPU acceleration being particularly effective for larger cluster numbers.

OpenMP's multi-threading capabilities effectively reduce CPU execution times, demonstrating its value in multi-core processing. Similarly, CUDA leverages GPU cores to significantly decrease computation times for large datasets, highlighting the advantages of parallelism in data-intensive tasks.

These findings validate the use of parallel computing in optimizing traditional algorithms and suggest substantial performance gains in machine learning workflows. The insights from this research are crucial for future advancements in large-scale data processing, illustrating the practical benefits of OpenMP and CUDA in high-performance computing for data analysis and machine learning.

References

[1] S. Daoudi, C. M. A. Zouaoui, M. C. El-Mezouar, and N. Taleb, "A Comparative study of parallel CPU/GPU implementations of the K-Means Algorithm," in 2019 International Conference on Advanced Electrical Engineering (ICAEE), 2019.

[2] M. Kruliš and M. Kratochvíl, "Detailed Analysis and Optimization of CUDA K-means Algorithm," in Proceedings of the 49th International Conference on Parallel Processing, 2020.

[3] C. Yang, Y. Li, and F. Cheng, "Accelerating k-Means on GPU with CUDA Programming," IOP Conference Series: Materials Science and Engineering, vol. 790, 2020, IOP Publishing.