Scalable HDBSCAN* for Large-Scale Data

Karthick T. Sharma
Department of Computer Engineering
Faculty of Engineering
University of Sri Jayewardenepura
Colombo, Sri Lanka
en93899@sjp.ac.lk

***Abstract*—Clustering algorithms play a pivotal role in unsupervised machine learning, revealing inherent patterns and structures within datasets. Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN*) has emerged as a robust technique for clustering complex datasets. However, as dataset sizes continue to grow, the computational demands of HDBSCAN* become a bottleneck. We propose a high-performance implementation of HDBSCAN*, empowered by CUDA and OpenMP parallelization. This methodology yields remarkable speed improvements, achieving a 7x acceleration in dense data scenarios and a 3x enhancement in sparse data situations. Extensive experiments were conducted on diverse datasets, including sparse and dense large-scale datasets with varying dimensions. The results highlight substantial improvement in computational efficiency without compromising the quality of clustering outcomes. Additionally, we compare our implementation with the state-of-the-art scikit-learn's HDBSCAN* implementation, demonstrating consistent performance gains across a range of scenarios. The achieved acceleration makes it particularly well-suited for handling big data clustering tasks, opening avenues for further exploration of hierarchical density-based clustering in large-scale applications.**

***Index Terms*—HDBSCAN, Clustering Algorithms, Unsupervised Machine Learning, Parallel Computing**

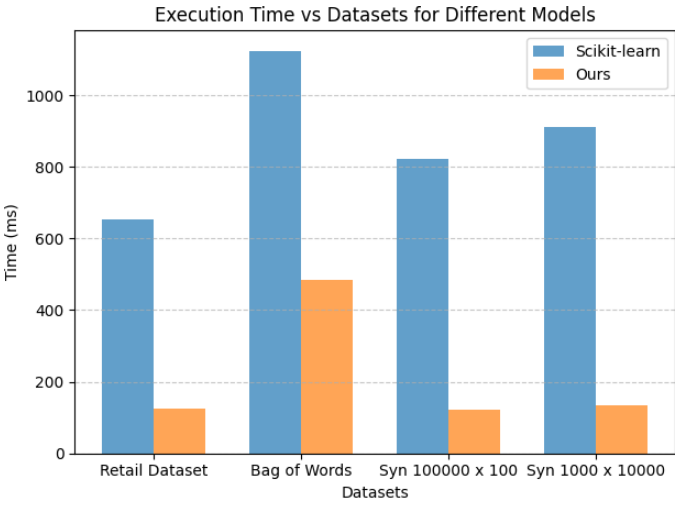

Fig. 1. Performance comparison with reference implementation

I. Introduction

Data analysis serves as a prevalent approach in contemporary scientific research, spanning disciplines such as communication science, computer science, and biological science. Clustering, as a fundamental component of data analysis, holds considerable importance. While numerous tools for cluster analysis have emerged in response to growing data volumes, each clustering algorithm exhibits distinct advantages and limitations, reflecting the intricate nature of the data landscape. Cluster analysis, a fundamental task in unsupervised machine learning, involves categorizing similar data points into groups without relying on predefined labels, revealing underlying patterns within datasets.

Clustering presents greater challenges compared to supervised classification, primarily due to the absence of labels associated with data patterns. In supervised classification, the provided labels guide the categorization of data objects. However, in clustering, determining the appropriate group for a pattern becomes more complex in the absence of such labels.

As the primary objective is to reveal underlying structures, patterns, or relationships within the data, making it an essential tool for exploratory data analysis and knowledge discovery. This technique is widely used in various domains such as pattern recognition, data mining, anomaly detection and information retrieval [1]. There are many different algorithms proposed for cluster analysis [1]. While widely-used algorithms like K-Means offer speed and simplicity, they necessitate prior knowledge of the number of clusters and operate as partitioning rather than true clustering methods. Other clustering algorithms, such as affinity propagation, spectral clustering, and agglomerative clustering, encounter similar challenges in partitioning all data points into groups, including outliers and noise. HDBSCAN*, a hierarchical density-based spatial clustering algorithm [2], stands out by identifying dense clusters without obliging every data point to be assigned, treating unassigned points as outliers or noise.

While HDBSCAN* is robust and effective algorithm for density-based clustering, has shown limitations in terms of scalability. As the size and complexity of dataset grow, the computational demands of HDBSCAN can become a hindrance, impeding its applicability to truly large-scale data clustering. In response to this challenge, high-performance implementation of HDBSCAN* is introduced here. This aims to address the limitations posed by the computational complexity of traditional HDBSCAN* by optimizing the underlying algorithms, parallelizing computations, and leveraging advanced computational architectures.

A comprehensive set of experiments on synthetic and real-world datasets, employing varying parameters, evaluates our implementations against optimized existing parallel scikit-lean's implementation[1]. As shown in Figure 1, our fastest version achieves a notable 7x speedup on dense synthetic data and 3x speedup in sparse real-world data compared to reference implementation.

The subsequent sections provide an extensive background and literature review (Section 2) to contextualize the significance of clustering algorithms and the challenges they face. Section 3 outlines the methodology employed in this study, including the proposed optimizations for the HDBSCAN* algorithm. Experimental setup, dataset setup and results from a series of rigorous experiments comparing our optimized implementation with reference method is presented in Section 4. Finally, Section 5 offers a summary discussion of the findings and outlines potential avenues for future research in HDBSCAN*.

II. Literature Review

A. Clustering

The diversity of clustering methods across different techniques arises from the lack of a definitive definition for the concept of a "cluster" [3]. [4] proposed categorizing clustering methods into two distinct groups: hierarchical and partitioning techniques. Meanwhile, [5] introduced three additional categories: density-based methods, model-based methods, and grid-based methods. [6] offered an alternative classification based on the underlying principles of various clustering approaches. Figure 2 shows the taxonomy of clustering approaches [4].

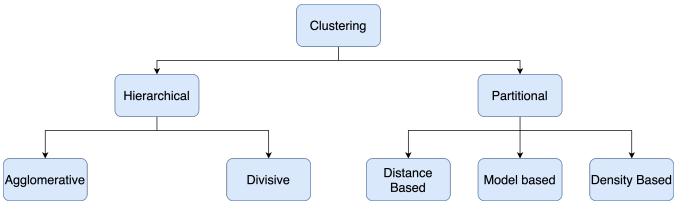

Fig. 2. Taxonomy of clustering approaches

Each clustering method possesses its strengths and weaknesses. Among partitioning-based algorithms, K-means and K-medoids are renowned for their low time complexity and computational efficiency; however, they struggle with non-convex data and are sensitive to outliers. Additionally, these methods require domain expertise to set the initial K value. In contrast, hierarchical algorithms address these challenges by establishing hierarchical relationships among data, enabling them to cluster data with arbitrary shapes and sizes. While generally scalable, hierarchical methods tend to be more time-consuming and complex compared to K-means [1].

[1]https://hdbscan.readthedocs.io/en/latest/index.html

B. HDBSCAN*

HDBSCAN* algorithm was introduced as an extension to the well-known DBSCAN algorithm, denoted as Hierarchical DBSCAN* [2]. This extension, detailed by the same authors in a subsequent work [7], presents a comprehensive framework for cluster analysis and outlier detection. HDBSCAN* improves upon DBSCAN by introducing a hierarchical representation of clusters, enabling effective cluster extraction and outlier detection. Notably, it addresses the limitations of DBSCAN by accommodating clusters of varying densities. Despite the merits of HDBSCAN*, it is crucial to note a drawback associated with its overall asymptotic complexity of $O(n^2)$. Moreover, the algorithm comprises multiple substeps, each with a complexity of $O(n^2)$, potentially impacting the execution time, especially for large datasets.

III. Methodology

Our implementation of the HDBSCAN* algorithm unfolds through four pivotal stages, elucidated as follows.

A. Core distance calculation

The first and cruicial step of HDBSCAN* algorithm is core distance calculation. Diverging from traditional methodologies anchored in K-NN, ball-tree, and KD-Tree approaches [2], [7], we embrace the Locality-Sensitive Hashing (LSH) technique, a variant of the approximate K-NN algorithm. This is utilized as a scalable solution capable of handling high-dimensional spaces. To expedite the process, exact K-NN operations were executed on curated buckets derived from LSH, leveraging the multi-threading capabilities of OpenMP. And finally Mutual Reachability Distance (MRD) was calculated for each data points from core distances.

B. Minimum Spanning Tree Construction

The subsequent phase entails the construction of a Minimum Spanning Tree (MST) derived from the MRD. Adopting the parallel Borůvka algorithm for this task facilitated hierarchical clustering via a bottom-up methodology. Given the inherent parallelizability of the Borůvka algorithm, CUDA technology was employed to optimize performance. Each thread was used to identifying the nearest edge for individual vertices, and a reduction mechanism was employed to generate the final MST.

C. Condense Tree

Following the MST generation, the data points underwent hierarchical clustering in a bottom-up fashion. Initially, each data point was construed as an individual cluster. As the MST processing unfolded, the number of clusters progressively decreased until a singular cluster encapsulating all data points remained. Given the memory-intensive nature of this process, OpenMP was leveraged to for clustering operations at each hierarchical level.

TABLE I
COMPARISON OF TIME AND ARI METRICS FOR DIFFERENT DATASETS. OPTIMAL VALUE IS DENOTED WITH BOLD.

	Retail Dataset		Bag of Words		Syn 100000 x 100		Syn 1000 x 10000	
	Time (ms)	ARI	Time (ms)	ARI	Time (ms)	ARI	Time (ms)	ARI
SK [7]	652	**1.0000**	1124	**0.2353**	821	**0.8365**	911	0.7934
Ours	**124**	0.9932	**483**	0.2215	**122**	**0.8365**	**134**	**0.8219**

D. Cluster Extraction

Final step is to filter out stable clusters. Given the propensity for overlapping clusters and the presence of outliers at each hierarchical level, this phase assumes paramount importance in determining the quality and purity of the resultant clusters. Since it is not a high computational task, single-thread is deployed to do this work. Stable clusters were defined based on stablilty score of each clusters [7].

IV. EXPERIMENTS

A. Experimental Setup

The computational experiments detailed in this paper were executed with a strategic selection of compiler flags to enhance the efficiency and performance of the underlying implementation. The optimization process leveraged the -O3 flag for aggressive optimization, aimed at accelerating the execution speed of the code. Additionally, the -march=native flag was employed to tailor the generated code to the specific architecture of the host machine, optimizing its performance for the available hardware. Concurrent execution and the harnessing of multiple cores or processors were facilitated by the -fopenmp flag, which enabled support for the OpenMP parallel programming API. Finally, the use of the -fPIC flag contributed to the creation of position-independent code, enhancing the adaptability of the compiled code for use in shared libraries. These compiler flags collectively played a pivotal role in ensuring the computational efficiency and robustness of the high-performance implementation.

Furthermore, these experiments were conducted using a system equipped with Intel i7-13700KS 48-core processor, 64 GB RAM and NVIDIA A6000 GPU.

B. Dataset

1) Retail Dataset [8]: The Online Retail dataset, with 541,909 instances and 6 attributes, records transactions from a UK-based online retail business between December 2010 and December 2011. It is a valuable resource for studying e-commerce trends, market segmentation, and predictive modeling.
2) Bag of Words [9]: The Bag of Words dataset comprises five text collections represented as bags-of-words, totaling 8,000,000 instances and consisting of 100,000 features. This dataset serves as a valuable resource for text analysis and natural language processing tasks, providing researchers with a diverse set of textual data for exploration and modeling.
3) Syn 100000 x 100: The Synthetic Dataset for dense representation in clustering, generated with 2 cluster centers, a cluster standard deviation of 2.5, and a total of 100000 instances featuring 100 attributes.
4) Syn 1000 x 10000: The Synthetic Dataset for dense representation in clustering, generated with 5 cluster centers, a cluster standard deviation of 1.7, and a total of 1000 instances featuring 10000 attributes.

C. Results

We employed two key metrics to assess the performance and effectiveness implementations. The first metric, Time (ms) focused on the computational efficiency, specifically the time taken for the clustering process. This metric provided insights into the algorithm's speed and scalability. The second metric, Adjusted Rand Index (ARI) compares the clustering results of our implementation with a reference implementation or ground truth, it quantifies the agreement between the two clustering outcomes. This metric served as a robust indicator of the algorithm's accuracy and ability to produce clustering results that align with a known or established reference.

Table I shows comparison of time taken for clustering and ARI score of implementations. For the 'Retail Dataset', ground truth is not given therefore results of reference implementation taken as ground truth and ARI score for reference is marked as '1'. For all scenario such as dense synthetic data and sparse real-world data, our implementation constantly outperforms reference implementation by a large margin in terms of clustering time and closely matches the reference accuracy.

V. CONCLUSION

Proposed implementation of a high-performance HDBSCAN* implementation, empowered by CUDA and OpenMP parallelization, marks a significant leap forward. Witnessing a remarkable 7x acceleration in dense data and a noteworthy 3x improvement in sparse data scenarios. Rigorous experimentation across diverse datasets, including large-scale instances with varying dimensions, underscores substantial gains in computational efficiency without compromising clustering quality.

A critical aspect for future exploration involves addressing accuracy concerns in the HDBSCAN* algorithm. Despite the accelerated performance, ensuring the preservation of high-quality cluster assignments remains paramount. Additionally, the integration of distributed computing for handling extremely large datasets, addressing the challenges posed by datasets exceeding the capacity of a single machine will be explored, further enhancing the scalability of our accelerated HDBSCAN implementation.

REFERENCES

[1] D. Xu and Y. jie Tian, "A comprehensive survey of clustering algorithms," *Annals of Data Science*, vol. 2, pp. 165 – 193, 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:54134680

[2] R. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," vol. 7819, 04 2013, pp. 160–172.

[3] L. Rokach and O. Maimon, *Clustering Methods*, 01 2005, pp. 321–352.

[4] C. Fraley and A. E. Raftery, "How many clusters? which clustering method? answers via model-based cluster analysis," *Comput. J.*, vol. 41, pp. 578–588, 1998. [Online]. Available: https://api.semanticscholar.org/CorpusID:9775342

[5] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011.

[6] V. Estivill-Castro and J. Yang, "Fast and robust general purpose clustering algorithms," *Data Mining and Knowledge Discovery*, vol. 8, pp. 127–150, 03 2004.

[7] R. J. G. B. Campello, D. Moulavi, A. Zimek, and J. Sander, "Hierarchical density estimates for data clustering, visualization, and outlier detection," *ACM Trans. Knowl. Discov. Data*, vol. 10, no. 1, jul 2015. [Online]. Available: https://doi.org/10.1145/2733381

[8] "Online Retail," UCI Machine Learning Repository, 2015, DOI: https://doi.org/10.24432/C5BW33.

[9] D. Newman, "Bag of Words," UCI Machine Learning Repository, 2008, DOI: https://doi.org/10.24432/C5ZG6P.