AITA
ARTIFICIAL INTELLIGENCE
THEORY AND APPLICATIONS

■ RESEARCH ARTICLE

The Comparative Effects of Clustering Algorithms on CPU and GPU

Pınar Ersoy [a †], **Mustafa Erşahin** [b], **Buket Erşahin** [c]

[a] Department of Technology, Dataroid, İstanbul, Turkey
[b] Department of Software Development, Commencis, İstanbul, Turkey
[c] Department of Computer Engineering, İzmir Institute of Technology, İzmir, Turkey
† pinar.ersoy@dataroid.com, corresponding author

RECEIVED JULY 6, 2022
ACCEPTED SEPTEMBER 14, 2022

CITATION Ersoy, P., Erşahin, M., Erşahin, B. 2022. The Comparative Effects of Clustering Algorithms on CPU and GPU. *Artificial Intelligence Theory and Applications, 2*(2), 19-27.

Abstract

The algorithm clustering can be defined as the operation of separating the populace or pieces of information into various groups. This article aims to construct a performance comparison for Partitional Clustering by using random, k-means++ algorithms implemented with Scikit-Learn and k-means++, Tunnel k-means algorithms implemented with TensorFlow-GPU by means of their execution times. As a final output, a related comparison table will be printed by supplying their framework specifications. Since the article does not focus on the context of data, the necessary data sets will be produced in a random manner.

Keywords: clustering, random k-means, k-means++, tunnel k-means, algorithm performance, TensorFlow-GPU, Scikit-Learn.

1. Introduction

To gain insightful comprehension of a subject, a common methodology can be placing them in separate pieces of groups to make it simpler to understand. For instance, you should see motion pictures by title, while someone else could see them by classification. How you choose to bunch them guides you to become familiar with them.

In Artificial Intelligence field, partitioning the data of interest into a specific number of gatherings is called bunching. These information focuses don't have starting names. Hence, clustering methodology is the gathering of unlabeled items of information into chunks to sort them out in a more significant manner.

K-means intends to parcel information into k groups such that data of interest in a similar group are comparable and data of interest in the various bunches are farther separated. The similitude of the two not set in stone by the distance between them. There are numerous techniques to gauge the distance.

K-implies computation is an iterative estimation that endeavors to portion the dataset into specific non-covering subgroups where each data point has a spot with only a solitary

Artificial Intelligence Theory and Applications, ISSN: 2757-9778. ISBN: 978-605-69730-2-4 © 2022 University of Bakırçay

occasion. It endeavors to make the intra-pack snippets of data as near as possible while similarly keeping the gatherings as different as could truly be anticipated. It consigns data concentration to a gathering with the ultimate objective that how much the squared distance between the information of interest and the bundle's centroid is at the base.

The partitioning the data items into a specific number of groupings can be referred as clustering. The clustering methodology is the gathering of unlabeled items of information into chunks to sort them out in a more significant manner.

This paper is coordinated in the upcoming pattern: Section 2 presents some ideal approaches on clustering calculations. Section 3 provides the additional information of the datasets and AI models that we explored different avenues regarding. At last, in Section 4, we present our experimental discoveries and examine them.

2. Related Work

The idea driving the k-means algorithm mainly is to scatter the data into centroids. The K-means algorithm [1] is an iterative grouping methodology. In the technique, k items are chosen in a contingent way as the underlying grouping community, the distance between each item and the underlying group origin is determined, and it is allowed to the closest center. Clustering origins and the items appointed to them address a group. For each data point allotted, the cluster origin will be recalculated by the current items in the group [2].

2.1. Hierarchical Clustering

The hierarchical clustering can be defined as a group investigation that searches for generating an order of groups. Primarily, the cluster merges and separations do not entirely settle in a covetous way [3]. The consequences of various leveled grouping are typically introduced in a dendrogram. The primary trait of this technique is examining and gathering the data, at the same time over a diversity of scales.

Hierarchical clustering contains a group of strategies that looks to assemble a ranking of clusters [4]. There exists two wide ways to deal with this methodology: the agglomerative hierarchical clustering, as indicated by which every elective begins in its own cluster, and sets of groups are shaped as one action up the ordered progression until a last, single bunch is in the long run framed; and divisive hierarchical clustering, where each of the choices starts in one single bunch, which is additionally divided in a recursive manner [5].

Divisive Hierarchical Clustering

The first one, divisive hierarchical clustering, has been referred by [6] which utilizes the basic thought of disruptive choices with positive and negative streams in various groups. That specific approach can be accepted as an insightful method with regard to this stream, yet it is straightforward in its presumptions since it did not depend on any advancement work connected to the component.

Agglomerative Hierarchical Clustering

The second category, agglomerative hierarchical clustering, represents significant and deeply grounded procedure in machine learning field [7]. This methodology is easy to implement and can give exceptionally useful depictions and perceptions of the potential

information clustering structure. In view of its computational dependability, the AHC method has been one of the essential grouping calculations [8].

2.2. Partitional Clustering

The partitional clustering algorithm is the most generally utilized bunching calculation. Among different executions of the partitional clustering approach, K-means is one of the most famous in actuality in view of its effortlessness and viability [9]. There are three wide ways to deal with this methodology as centroid, model-based, graph-theoretic, and spectral.

Centroid Partitional Clustering

The first category, centroid partitional clustering, contains a classification to depict the data dissemination of a cluster group. Thus, the ambiguity during grouping can be kept for long as conceivable before the genuine choices are made [10].

Model-Based Partitional Clustering

The model-based clustering method is committed to multivariate partial positioning information. This is an expansion of the Insertion Sorting Rank model [11] for positioning information, which has the double property to be a significant model through its area and scale parameters depiction [12].

Graph-Theoretic Partitional Clustering

The second category, graph-theoretic partial clustering, tackles the multi-view grouping issue [13] via coordinating the diagram designs of various perspectives, which can completely take advantage of the mathematical property of the essential information structure. Accordingly, it is a main point of contention for information investigation to accumulate data from numerous perspectives and investigate the chart based models [14].

Spectral Partitional Clustering

The third category, spectral partial clustering, is achieved by developing a Laplacian chart from relating data of interest with edges between them addressing the likenesses, so it can assure the general construction of data [15].

2.3. Bayesian Clustering

In Bayesian model-based clustering, data points which are driven to have similar model-explicit boundaries are viewed as from a similar group [16]. There are two ways to deal with this methodology as decision-based, and non-parametric Bayesian clustering.

3. Materials and Methods

In this section, we give the details of the data generated and the method applied in this study.

3.1. Dataset

The dataset is composed of randomly generated integers since the effect of the different content of the data has not taken into consideration by means of performance of the comparison of the algorithms.

3.2. Data Preprocessing

In this study, we examined the different types of clustering algorithms from both Scikit-learn Python package and TensorFlow-GPU based packages that relates to the presented algorithms.

As the data is manually produced, no additional data processing step is applied. The created data points are directly used in the clustering algorithms.

3.3. Experimental Setup

There exist several approaches while comparing clustering approaches in a dataset. Being able to find the correct metrics to compare them in appropriate way is a tough task to accomplish. In this paper, as a first step, we generate datasets that includes randomly generated integers.

The CPU can be accepted as the crucial hardware where the majority of the calculations actualizes inside. The Graphical Processing Unit (GPU) powered TensorFlow library was created to be utilized for huge datasets of mathematical processes.

Table 1. System Specifications of CPU and GPU-Powered Systems

System Specs	First CPU System	Second CPU System	GPU System
Processor Type	Intel(R) Core i7	Intel(R) Core i9	Intel(R) Core i9
Processor GHz	2.60 GHz	3.70 GHz	3.70 GHz
P-Cores	6	10	10
L-Cores	12	20	20
Random Access Memory	64 GB	256 GB	256 GB
Graphics Processors (GPU)			NVIDIA GeForce® RTX 2080Ti

3.4. Model Evaluation Metric

The execution times of each algorithm were calculated to be able to use it while evaluating each algorithm in an additional metric. To achieve this, timers were added at the beginning and ending of each algorithm as a starting and ending time values in seconds. After the execution of each algorithm, the time between the end time and the start time was subtracted to create the duration value in seconds.

4. Results and Discussion

In this section, We trained two clustering algorithms and tested them on three different system by using the corresponding clustering functions in Scikit-Learn and TensorFlow Python packages. We present system specifications in Table 1, and time metrics for each dataset in Table 2, Table 3 and Table 4, respectively.

4.1. CPU-Based K-Means Clustering

In the context of our experiments, we chose the following algorithms of centroid partitional k-means algorithm on both two CPU-powered system with the Scikit-learn, kmeans++ algorithm, and GPU-powered system with tensorflow-gpu package installed tunnel kmeans-random algorithm for comparison by their existing clustering performances.

After assigning cluster parameters, random numbers are generated with 5 columns and 500 K rows in a contingent manner.

K-Means (random) Clustering with Scikit-Learn

K-means algorithm of the Python's machine learning package selects the initially the center cluster in an arbitrary fashion.

Table 2. Experiment Results For K-Means (random) Clustering with Scikit-Learn

Metrics	First CPU System	Second CPU System
Algorithm Run Time	2506.26 seconds	1395.10 seconds
Cluster Depiction	Blue area represents the item sets, while Red are can be accepted as the groups generated	Blue area represents the item sets, while Red are can be accepted as the groups generated

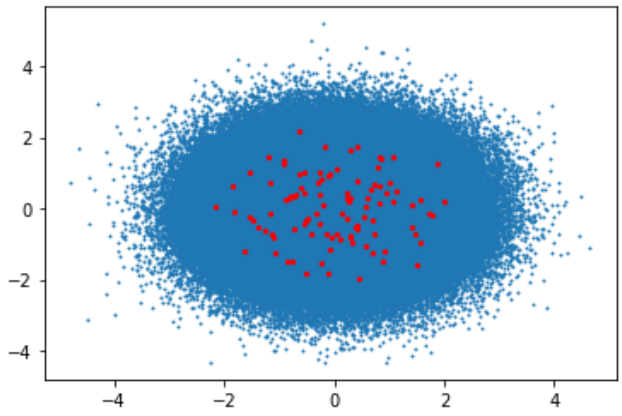

Figure 1. Experiment results for First CPU-Based K-Means (random) Clustering with Scikit-Learn

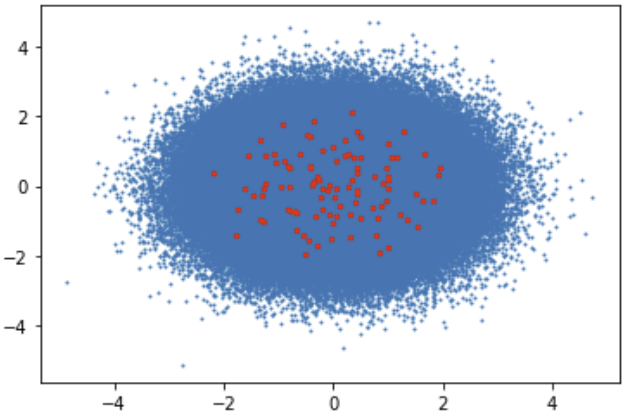

Figure 2. Experiment results for Second CPU-Based K-Means (random) Clustering with Scikit-Learn

Table 3. Experiment Results For kmeans++ Clustering with Scikit-Learn

Metrics	**First CPU System**	**Second CPU System**
Algorithm Run Time	2603.75 seconds	1384.73 seconds
Cluster Depiction	Blue area represents the item sets, while Red are can be accepted as the groups generated	Blue area represents the item sets, while Red are can be accepted as the groups generated

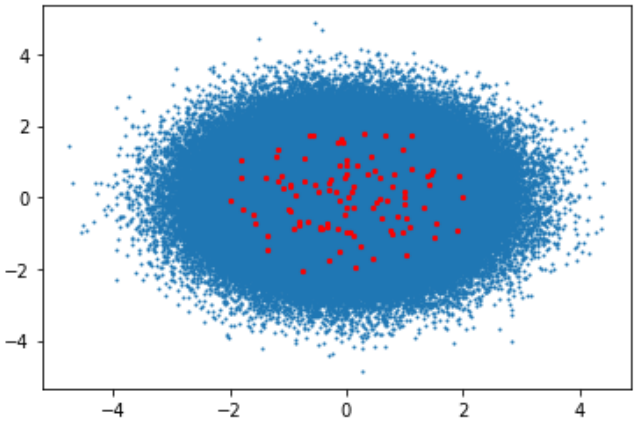

Figure 3. Experiment results for First CPU-Based K-Means (kmeans++) Clustering with Scikit-Learn

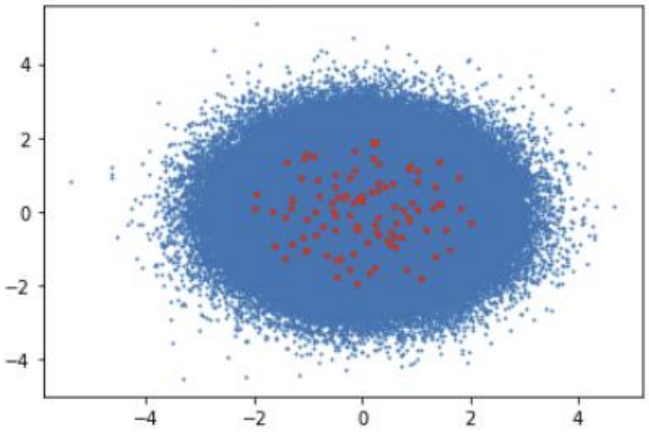

Figure 4. Experiment results for Second CPU-Based K-Means (kmeans++) Clustering with Scikit-Learn

4.2. GPU-Based K-Means Clustering

Tensorflow library can be applied for greater amount of numerical calculations.

K-Means (kmeans++) Clustering

TensorFlow k-means methodology refracts the unwanted effects of causally derived initial origin centroid. To enable this feature, kmeanstf package can be implemented in the existing environment.

Table 4. Experiment Results For kmeans++ Clustering with TensorFlow-GPU

Metrics	**First CPU System**	**Second CPU System**
Algorithm Run Time	219.18 seconds	107.38 seconds
Cluster Depiction	Blue area represents the item sets, while Red are can be accepted as the groups generated	Blue area represents the item sets, while Red are can be accepted as the groups generated

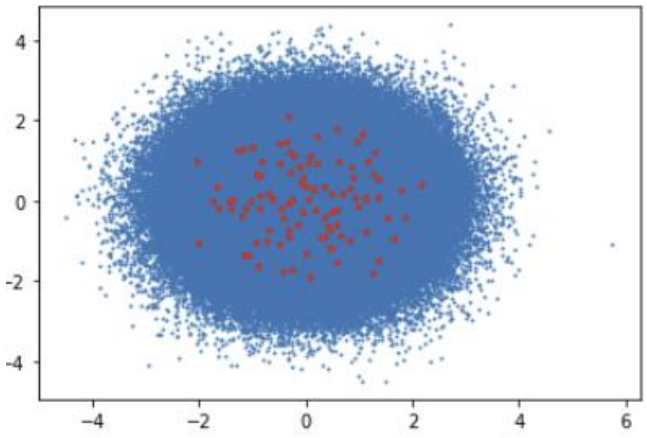

Figure 5. Experiment results for GPU-Based kmeans++ Clustering with TensorFlow-GPU

K-Means (Tunnel-K-means)

The tunnel k-means method actualizes shifts between clusters to find the most optimum centroids for the items in the chunk.

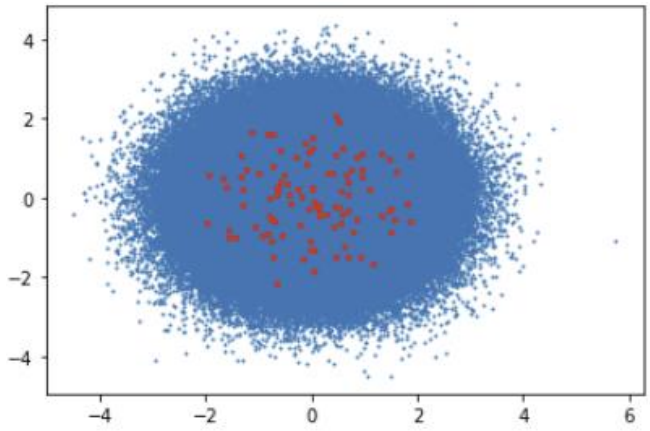

Figure 6. Experiment results for GPU-Based K-Means (Tunnel K-means) Clustering with TensorFlow-GPU

5. Conclusion

As the brief depiction, matrix table is created by listing each algorithm tested with their matching execution times.

Execution time of an algorithm indicates an insight about its time complexity. The runtime of an algorithm might affect computational systems in a negative way. To reduce its resource consuming side effects, it is critical to monitor the execution period of each algorithm with model performance metrics.

In parallel with system capabilities, higher CPU-powered system practices the lowest algorithm runtime compared to that of lower one. By evaluating this result, we can

conclude that higher CPU capability have promising effects on run-time performances of clustering algorithms.

In addition, GPU system contains a higher performance by means of run times. By considering this information, GPU boosts execution times more than CPU-based systems.

References

[1] Julia, S., Oliver, S. (2016). Multi-objective three stage design optimization for island microgrids. Appl Energy 2016;165:789–800.
[2] Zhang Z. et al. (2021). Clustering analysis of typical scenarios of island power supply system by using cohesive hierarchical clustering based K-Means clustering method. Energy Reports 7, 250–256
[3] Garza-Ulloa, J. (2018). Chapter 6 - Application of mathematical models in biomechatronics: artificial intelligence and time-frequency analysis, Applied Biomechatronics using Mathematical Models.
[4] Murtagh, F. Contreras, P., (2019). Algorithms for hierarchical clustering: an overview, II. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 7 (6) (2017), p. e1219.
[5] Ishizaka, A. (2021). A Stochastic Multi-criteria divisive hierarchical clustering algorithm. B. Lokman and M. Tasiou, Omega, vol. 103.
[6] De Smet, Y. (2014). An extension of PROMETHEE to divisive hierarchical multicriteria clustering", 2014 IEEE International Conference on Industrial Engineering and Engineering Management, IEEE, pp. 555-558.
[7] D. Müllner, (2021). Modern hierarchical, agglomerative clustering algorithms. Comput. Sci., pp. 1-29.
[8] Zhou, S., Xu, Z. Liu, F., (2017). Method for determining the optimal number of clusters based on agglomerative hierarchical clustering. IEEE Trans. Neural Networks Learn. Syst., 28 (12), pp. 3007-3017.
[9] Zhu, E., Ma, R. (2018). An effective partitional clustering algorithm based on new clustering validity index. Applied Soft Computing 71, 608–621.
[10] Zhu, S., Xu, L. (2018). Many-objective fuzzy centroids clustering algorithm for categorical data", Expert Systems With Applications 96, 230–248.
[11] Knuth, D., (1973). The Art of Computer Programming, Sorting and Searching. Vol. 3, Addison-Wesley, Massachusetts.
[12] Jacques, J., Biernacki, C. (2020). Model-based clustering for rank data based on an insertion sorting algorithm. In: 17th Rencontres de la Société Francophone de Classification, La Réunion.
[13] Zhan, K., Niu, C., Chen, C., Nie, F., Zhang, C., Yang, Y. (2018). "Graph structure fusion for multiview clustering", IEEE Transactions on Knowledge and Data Engineering, 31 (10), pp. 1984-1993
[14] Tang, C., Zhu, X., Liu, X., Li, M., Wang, P., Zhang, C., Wang, L. (2018). Learning a joint affinity graph for multiview subspace clustering", IEEE Transactions on Multimedia, 21 (7), pp. 1724-1736.
[15] Kumar, A., Iii, H.D. (2011). A co-training approach for multi-view spectral clustering", International Conference on International Conference on Machine Learning, Omnipress, pp. 393-400.
[16] Wang, K., Porter, M.D. (2018). Optimal Bayesian clustering using non-negative matrix factorization", Computational Statistics and Data Analysis 128. 395–411
[17] Randles, B. M., Pasquetto, I. V., Golshan, M. S., and Borgman, C. L. (2017). Using the jupyter notebook as a tool for open science: An empirical study," in ACM/IEEE Joint Conference on Digital Libraries (JCDL).
[18] McKinney, W. (2011). Pandas: A foundational python library for data analysis and statistics. Python for High Performance and Scientific Computing, vol. 14.
[19] Greenfield, P., Miller, J. T., Hsu, J. & White, R. L (2003). Numarray: a new scientific array package for python. In PyCon DC.
[20] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, É. (2011). Scikit-learn: Machine learning in python. Journal of Machine Learning Research, vol. 12, pp. 2825-2830.