AbdElSamea and Saif
Journal of Electrical Systems and Inf Technol (2024) 11:35
https://doi.org/10.1186/s43067-024-00161-1

Journal of Electrical Systems and Information Technology

RESEARCH Open Access

K-medoid clustering containerized allocation algorithm for cloud computing environment

Amany AbdElSamea[1*] and Sherif M. Saif[1]

*Correspondence:
amany@eri.sci.eg

[1] Computers and Systems Department, Electronics Research Institute, Cairo, Egypt

Abstract

Load balancing is critical for container-based cloud computing environments for several reasons. A lack of appropriate load balancing techniques could result in a decrease in performance and possible service interruptions as some nodes get overloaded, while others are left underutilized. Cloud service providers can reduce latency and boost system performance by strategically placing containers using clustering algorithms. These techniques aid in efficiently using resources and load balancing by clustering related containers together according to their shared attributes. Clustering strategies are effective in allocating and controlling resources to meet the demands of a changing workload. Algorithms for clustering combine related workloads or containers into clusters, improving performance isolation and maximizing resource usage. One popular methodology for data clustering is the K-Medoid Clustering Algorithm. It is especially helpful when working with categorical data or when the dataset contains outliers. K-medoids is an unsupervised clustering approach where the core of the cluster is made up of data points known as "medoids." A medoid is a location in the cluster whose total distance to every object in the cluster—also known as its dissimilarity—is as small as possible. Any appropriate distance function may be used, such as the Manhattan distance, the Euclidean distance, or another one. Thus, by choosing K medoids from our data sample, the K-medoids method splits the data into K clusters. This work presents the K-Medoid clustering technique for containers, which can enhance load balancing, decrease resource execution times, and increase resource utilization rates all at the same time. The results of the experiment show that the proposed method outperforms MACO and FCFS in terms of throughput by about 70% when number of cloudlets increased. The relative improvement of execution time of the proposed K-medoid algorithm w.r.t FCFS is about 50%.

Keywords: Clustering, Scheduling, Load balancing, Containers, Cloud computing

Introduction

In cloud computing, virtualization [1] is the process of the abstraction of computer resources from their actual hardware to allow for the simultaneous operation of several virtual instances or environments on a single physical machine over a network of devices. This technology creates virtual versions of servers, storage, networks, and operating systems, enabling effective resource management and utilization by using a tiny layer of software known as a hypervisor. There are two types of hypervisors, type 1 (bare-metal or Native) and type 2 (Hosted) [2]. Type 1 is hardware virtualization that runs directly on the system with VMs running on them since it has direct access to hardware it provides better scalability, speed, and security. Type 2 is OS virtualization that should run on an operating system [3].

Containers [4] are an OS virtualization technique which has an indispensable part of the cloud computing system because of their ability to bundle and isolate applications together with their dependencies. One of the main benefits of adopting containers in cloud computing is their lightweight nature, which enables rapid application scalability and deployment. The growing acceptance of containerization technology has increased the demand for efficient scheduling systems. Clustering techniques [5] are important for container allocation in cloud computing environments since they efficiently allocate resources to meet varying demands of workload. Clustering algorithms combine containers with similar features together to improve communication and provide low latency. One can group the clustering algorithms into density-based [6, 7], hierarchical [8], and partition-based clustering [9]. One popular methodology for partition-based clustering is the K-Medoid Clustering Algorithm [10]. It is especially helpful when working with categorical data or when the dataset contains outliers. The primary points of distinction between them k-medoid and k-mean [11] is the way they identify cluster centers and manage noise and outliers. Table 1 shows the difference between k-medoid and k-means clustering algorithms. In k-medoid clustering, one of the actual data points in the cluster acts as the cluster center, or medoid. K-medoids represents the center by an actual data point unlike k-means, which calculates the mean. When using k-means clustering, the mean of all the data points in a cluster serves as a representation of the cluster center, or centroid. The objective of the technique is to minimize the total squared distances across every point and the specified cluster centroid [12].

Since K-medoids use real-data points (medoids) as cluster centers, it is less vulnerable to outliers. Using actual data points as cluster representatives directly increases the robustness of k-medoids against noise and outliers. Due to its reliance on the mean of all points allocated to the cluster for computing cluster centers, K-means are susceptible to anomalies. Outliers can dramatically affect the centroids' positions, sometimes resulting in less-than-ideal clustering. K-medoids can handle clusters of various sizes, densities, and shapes more easily since it doesn't make any assumptions about the shape of the clusters. K-means

Table 1 The difference between k-means and k-medoid clustering algorithms

Feature	K-means	K-medoids
Cluster center type	Centroid (Mean of points)	Medoids (Actual data point)
Cluster Shape Assumption	Assumes spherical cluster shape and equal variance	Does not assume certain cluster shape
Data Type	Only numerical data can be used with it	It is applicable to both categorical and numerical data
Objective	Minimize the SSE (Sum of Squared Errors), or the sum of squared distances	Reduce the dissimilarities between clusters of data from the dataset
Outlier sensitivity	Not sensitive to outliers within the data	It is outlier-resistant and can reduce the effect of outliers
Computational cost	Less costly to implement	More costly to implement
Speed	Faster	Slower
Noise Handling	It does not cater to noise in the data	Effectively reduces the noise in the data

presupposes that clusters are spherical and have equal variance. It may have trouble with varying sizes and densities of clusters. This may not always be the case in real data. To identify the optimal medoids, pairwise dissimilarity calculations between data points are required in k-medoids, which makes it more computationally demanding than k-means. With bigger data sets, this might be more difficult. K-means is computationally efficient and often faster than k-medoids, especially for large data sets.

Although both algorithms are widely used for grouping data into clusters, k-means is more scalable and faster, but it is more susceptible to outliers and cluster shape assumptions. In contrast, K-medoids (PAM) do not require spherical clusters and are more resilient to outliers, although they come at a higher processing cost. The unique properties of your data and the objectives of your clustering task will determine which of these algorithms is best for you.

To enhance load balancing, shorten resource execution times, and boost resource utilization rates for containers and virtual machines (VMs), the K-medoid clustering technique for containers is presented in this paper. The reason for choosing the K-medoid clustering algorithm is that the K-medoid algorithm's resilience to noise and outliers is one of its main benefits. K-medoid is hence more suited for datasets with skewed or noisy distributions since the medoid is less impacted by extreme values because they are actual data points from the dataset. The interpretability of K-Medoid clustering is an additional advantage. Medoids can offer valuable insights into the features of each cluster because they are real data points. This can be especially helpful in practical applications when it's imperative to comprehend the underlying patterns inside clusters. The execution time of the suggested solution is compared to Modified Ant Colony Optimization (MACO) for containers [13] and First Come First Serve (FCFS) algorithm and keeps the resource utilization of virtual and physical machines much improved. More precisely, the major contribution of this paper might be summed up as follows:

- Present a detailed comparison between k-medoid and k-mean clustering algorithms.
- Introduce and implement a K-medoid clustering algorithm for containers.
- Comparing the outcome of the suggested method against that of modified MACO and FCFS algorithms.

This paper has been structured as follows: "Related work" section explains the related work. "Basic K-medoid clustering algorithm" section identifies the basic k-medoid clustering technique. "K-medoid clustering algorithm for containerized cloud computing environment" section presents the proposed k-medoid clustering technique for containers. "Implementation and Simulation Results section covers the implementation and simulation results. Finally, future work and the conclusion are discussed in "Conclusion" section.

Related work

The most recent research on the K-medoid clustering technique is reviewed in this section. Yaobin Jiang et al. [14] proposed a Hadoop-based parallel K-Medoids technique known as HK-Medoids. Although the K-Medoids clustering technique resolves the K-Means algorithm's issue with handling outlier samples, its temporal complexity prevents it from processing large amounts of data. Map Reduce is a huge data processing paradigm for parallel programming that has been integrated into Hadoop. To surpass the big-data constraints, the parallel K-Medoids algorithm HK-Medoids is proposed. Several iterative MapReduce algorithms are present in every submitted job: The map phase involves assigning each sample to a cluster whose center most closely matches the sample; the combine phase involves calculating an intermediate center for every cluster; and the reduce phase involves calculating the new center. As soon as the new center resembles the previous one, the iterator ends. The experimental findings demonstrated the linear speedup and good clustering performance of the HK-Medoids algorithm for big data. The paper's drawback is that the work didn't consider platforms other than Hadoop.

Algorithm 1 K-medoid clustering containerized allocation algorithm for cloud computing

```
Input: VM list, Container list, Number of Medoids (k).
Output: Allocation of Containers on VMs
for each container in the container-list do
        Compute the values of MIPS and RAM size
end for
Choose randomly k-medoids
for each container in the container-list do
        Assign each container to the closest medoid based on the established
distance metric
        Compute the total cost.
        for each cluster do
                Choose a new medoid that decrease the total distance to all other
        objects in its cluster

                Swap new medoid with the previous medoid
                Calculate the total cost of the new medoid
                If the total cost (New medoid) < Previous medoid then
                        Make the new medoid permanent
                else
                        Undo the swap and choose a new medoid
        end for
        Iterate until convergence
        return Container-Clusters
end for

/ Clustering the VMs using K-medoid
for each VM in VM-list do
        Get the values of MIPS and RAM size
        Apply the K-medoid clustering algorithm on the VMs
        return VM-Clusters
end for

for each container in the container-list do
        Assign container to appropriate VM
end for
send now (VM ID, Container ID)
```

Md. Touhidul Islam et al. [15] provide a Hybrid Genetic Algorithm (HGA), in which n data points are divided into k clusters. The methods employed in the k-means and k-medoids algorithms of data clustering are utilized to determine the cluster centers of the created clusters. The sum of the Euclidean distances among every data point and the respective cluster centers is then utilized to determine the clustering's fitness. We have conducted experiments using the UCI Machine Learning Repository's Iris, Seeds, and Ionosphere datasets. According to experimental data, the clustering accuracies produced by the suggested HGA were between 2.67 and 28.68% greater than those previously reported in the literature. The drawback is that this paper didn't consider other clustering algorithms such as density-based and Hierarchal clustering algorithms.

Manila Gupta et al. [16] present a strategy that takes Quality of Service (QoS) considerations into account by proposing Modified Fire Hawks Gazelle Optimization (MFHGO) algorithm. Partitioning around K-medoids (PAKM) clustering is used to attain an optimal resource allocation to do this. K-means clustering is extended by the proposed paradigm. The JAVA program is used for exploratory analysis, while the GWA-T-12 Bitbrains dataset is used for simulation. By contrasting the suggested resource allocation and clustering technique with the current systems, its efficacy is shown. The drawback of the proposed algorithm is that research on energy-conscious resource allocation and energy-efficient computing techniques are not considered.

Heidari et al. [17] suggested proposing three techniques to enhance K-means and K-medoids algorithms. These methods can automatically calculate the right number of clusters by taking into account the overlap space among clusters and the density of each cluster. They use a novel technique to identify outlier data when choosing initial centers. To lessen the adverse effects of outlier data on the clustering process, distinct clusters are taken into consideration. Clusters with non-spherical shapes can also be found using the NSK-means and NSK-medoids methods that have been used on many data sets. Additionally, they compared the simulation results with other algorithms to determine the ideal number of clusters, and the results were good. The data can be more precisely clustered into the overlap space using the CSK-means and NSK-means techniques. Data sets could be clustered using non-spherical clusters using the NSK-means and NSK-medoids methods. Furthermore, the simulated results on the data set with various combinations of outliers demonstrate that the algorithms can identify the outliers and take into account different clusters for them. The suggested techniques drawback is that they need a lot of steps to get to the final cluster number when the input data set has a lot of clusters.

The previous related work neglected to take into account using the K-medoid clustering algorithm for container resource allocation in cloud computing environments, but this paper provides a K-medoid clustering algorithm for efficient container allocation in cloud computing environments.

Basic K-medoid clustering algorithm

K-medoids [18] is an unsupervised technique for clustering unlabeled data. K-medoids, that is often referred to as Partitioning Around Medoids (PAM), is a commonly utilized clustering algorithm which selects k representative objects from a dataset to group k data points into clusters. K-medoid is a widely used clustering algorithm that divides a dataset into clusters according to how similar the data points are to one another.

K-medoid clustering uses the actual data points as representatives for every cluster, in contrast to K-means clustering, which uses data points mean to determine the centroid of each cluster.

The k-medoids algorithm seeks to cluster data points into k clusters with every data point allocated to a medoid. A medoid is a point in a dataset's cluster where the total distances to all other points are as small as possible. It is the data point in a cluster characterized by the lowest dissimilarity with other data points and the total distance between each data point and its designated medoid is kept to a minimum. Each data point is successively assigned to the nearest medoid by the algorithm, which then switches the medoid of each cluster until convergence occurs. The flowchart of the basic k-medoid algorithm is shown in Fig. 1.

The technique starts by choosing k randomly chosen points as the starting medoids from the dataset. The chosen medoids define the first k clusters, and data points are thereafter allocated to the nearest medoids cluster. For each medoid, determine the total sum of the data points' distances from their designated medoids. The cost, or the total sum of the differences or distances across the data points and the designated medoid, is next computed. Swap a non-medoid point with a medoid point, and then compute the cost again. If the new medoids recalculated cost is higher than the old one, undo the swap, and repetition is required until new medoids are used to classify data points without causing any changes so the algorithm converges.

K-medoid clustering algorithm for containerized cloud computing environment

Multi-objectives-based container scheduling algorithms must map containers to VMS in an efficient way in order to optimize resource use. One method for improving the usage of cloud computing resources is clustering. Task scheduling is an essential first step in improving the overall performance of cloud computing. Clustering-based task scheduling techniques could be a promising solution to the previously described issue because they allow for the efficient scheduling of many containers based on container multi-criteria while reducing processing time. A clustering algorithm called K-medoid is used by the virtual machines and containers. The k-medoid clustering algorithm is used to group the virtual machines and containers based on MIPS, RAM, and BW capacity. In order to enhance load balancing and decrease resource execution times while concurrently raising resource utilization rates, this research suggests using a k-Medoid clustering technique for containers. For cloud computing containers, Algorithm1 presents the pseudocode of the proposed K-Medoid clustering scheduling algorithm.

Implementation and simulation results

In this section, the implementation environment, performance metrics, and experimental results are presented. The proposed k-medoid clustering containerized allocation algorithm is compared w.r.t Modified Ant Colony Optimization (MACO) [12] and FCFS algorithms using the Container CloudSim simulator.

Implementation environment

ContainerCloudSim [19] is the first simulator program that supports Container as a Service and it is an extension of CloudSim. It is mostly used to test container scheduling and provisioning policies, as well as overbooking approaches and container consolidation. It offers a setting for assessing resource management strategies such as container placement, scheduling, and consolidation. Researchers may examine resource management strategies for both virtualization types OS level virtualization, or containers, and system level virtualization, or VMs simultaneously using ContainerCloudSim.

Parameter setting

Using the ContainerCloudSim platform, we evaluate and compare our proposed K-medoid clustering scheduling method for cloud computing containers against other job scheduling algorithms. Table 2 shows us how to set the parameters of the suggested algorithm to get the greatest performance. We initialize the number of clusters

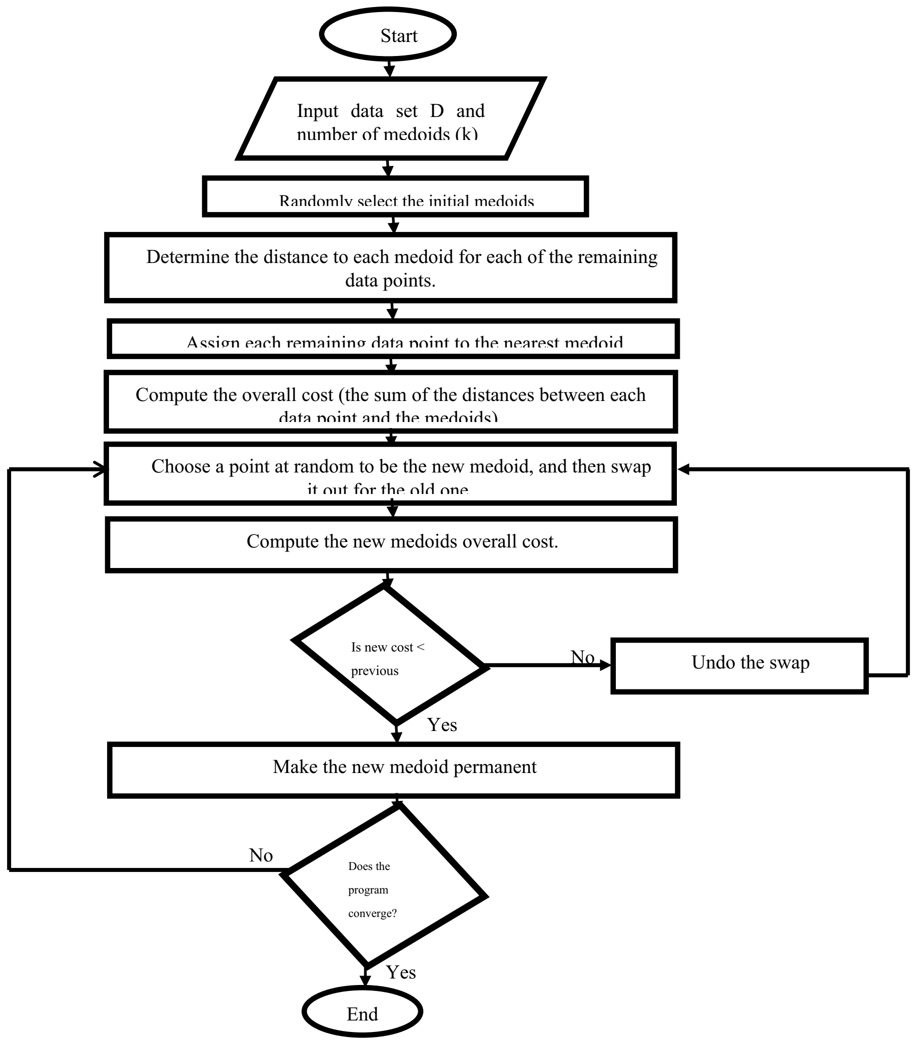
Start
Input data set D and number of medoids (k)
Randomly select the initial medoids
Determine the distance to each medoid for each of the remaining data points.
Assign each remaining data point to the nearest medoid
Compute the overall cost (the sum of the distances between each data point and the medoids)
Choose a point at random to be the new medoid, and then swap it out for the old one
Compute the new medoids overall cost.
Is new cost < previous
No
Undo the swap
Yes
Make the new medoid permanent
No
Does the program converge?
Yes
End

Fig. 1 Basic K-medoid clustering algorithm flowchart

k to be 3 for both containers and VMs. We use random workload traces presented in ContainerCloudSim to compare between the literatures algorithms and the proposed algorithms.

Performance metrics

Several experiments were conducted and analyzed using the following metrics in order to provide a thorough examination of the recommended algorithm to compare and assess the performance of the algorithm [20, 21]:

Total execution time: It is the length of time, expressed in seconds, needed to complete an experiment. It's the total amount of time required to complete a group of tasks.

Throughput: It is the quantity of tasks it can finish in a predetermined amount of time. A system's throughput is used to evaluate its overall performance.

Silhouette score: The silhouette score quantifies the degree to which an item is more cohesive (similar to) within its own cluster than it is with other clusters (separation). A high number suggests that the object is well matched to its own cluster and poorly matched to neighboring clusters. The silhouette goes from -1 to $+1$. The clustering setup is useful if the majority of the objects have a high value. The clustering arrangement may have too many or too few clusters if a large number of the points have low or negative values. Silhouette score is calculated using Eq. 1

$$\text{Silhouette score} = (\mathrm{b} - \mathrm{a})/\max(\mathrm{a}, \mathrm{b}) \tag{1}$$

where *a* is mean intra cluster distance, *b* is the mean nearest cluster distance.

Experimental results

To test the functionality and validity of the basic k-Medoid clustering algorithm, the Iris dataset [22] is used in the first experiment. The clustering visualization of the basic K-medoid clustering algorithm on the iris dataset is shown in Fig. 2. Although K-medoid is expected to perform exclusive clustering but in real datasets, clusters can overlap and there are often outliers that do not belong to any cluster. Since iris dataset is a real dataset so there is an overlap between cluster 0 and cluster 2. It is shown that the cluster shape is not necessarily to be in a spherical shape and this algorithm is sensitive to outliers and noise. Then we apply the basic k-Medoid to cluster the containers in the cloud computing environment based on the reading of the MIPS and RAM utilization into three clusters also we use k-Medoid to cluster the VMs then we assign the containers to the suitable VM.

The relative improvement of execution time is calculated w.r.t FCFS algorithm is displayed in Fig. 3. As a result, a positive value of relative improvement denotes poorer performance and a negative value denotes a better performance. It is demonstrated that the suggested approach minimizes the needed time while achieving a good balance of system loads compared to MACO and FCFS algorithm. When number of cloudlets increased (700, 800, 900, 1000) the relative improvement of the proposed K-medoid algorithm w.r.t FCFS is about 50%, but the relative improvement of execution time of MACO w.r.t FCFs is about 30% when number of cloudlets are increased than 400.

Table 2 ContainerCloudSim parameter setting

Type	Parameters	Value
Containers	*TYPES*	3
	MIPS	4658, 9320, 18,636
	PES	1
	RAM	128, 256, 512
	BW	2500
Virtual MACHINE	*TYPES*	4
	PES	2, 4, 1, 8
	RAM	1024, 2048, 4096, 8192
	BW	100,000
	SIZE	2500
Hosts	*TYPES*	3
	MIPS	37,274
	PES	4, 8, 16
	RAM	65,536, 131,072, 262,144
	BW	1,000,000
	STORAGE	1,000,000

Figure 4 illustrates that the suggested technique outperforms MACO and FCFS in terms of throughput across all cloudlets since it divide the containers into clusters and assign each container to the suitable VM so improve the throughput. Also it is sensitive to outliers and noise while MACO is computationally expensive when handling large problem instances. MACO technique becomes difficult to handle real-time applications as the number of ants and iterations increases, increasing the algorithm's execution time and memory requirements. Also in FCFS the average waiting times are frequently rather long causing lower efficiency and CPU utilization. The suggested technique performs best when there are 1000 cloudlets, which allows for the optimal placement of containers on virtual machines (VMs) with lower response times and higher throughput. When there are only 100 cloudlets, the algorithm performs worst.

Figure 5 shows the mean silhouette score of clusters A silhouette score of one indicates that the clusters are well-separated and extremely dense. Overlapping clusters are indicated by a silhouette score of 0. A score of less than zero suggests that there may be errors in the data. As shown in the figure cluster 1 has the highest mean silhouette score (0.8) then cluster 0 (0.6) then cluster 2 (0.54).

Table 3 presents the silhouette score significance when silhouette score is near 1 as the case of cluster 1 so this indicates that the data points are well separated and extremely dense. When the silhouette score becomes nearer to 0, this means that data points of the clusters may be overlapped.

Conclusion

In a cloud computing system that uses containers, load balancing is essential for a number of reasons. Primarily, it guarantees that the resources dispersed among several containers are employed effectively, averting any one container from experiencing excessive traffic while others stay underutilized. Clustering-based task scheduling algorithms

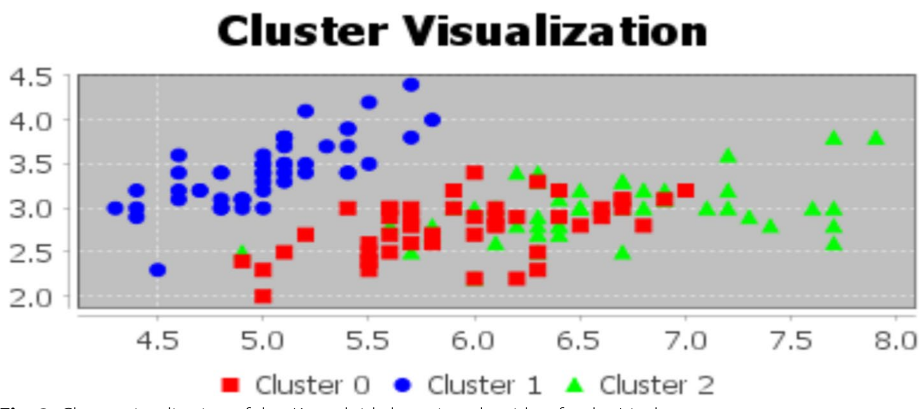

Fig. 2 Cluster visualization of the K-medoid clustering algorithm for the Iris dataset

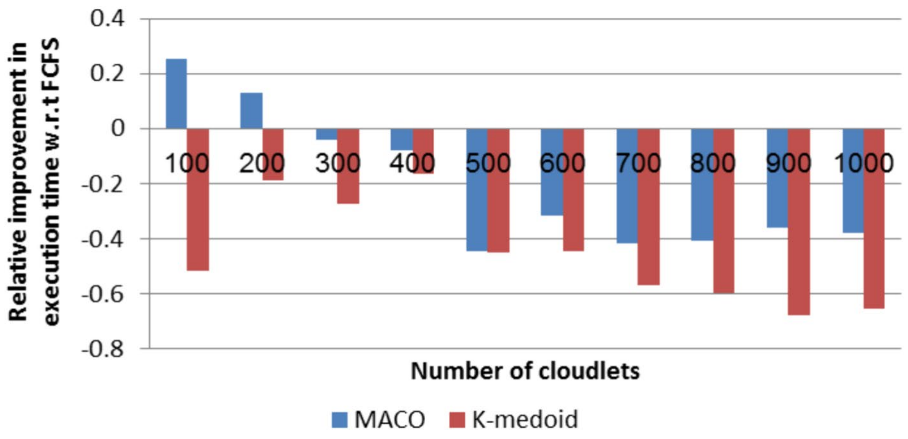

Fig. 3 Relative improvement in execution time w.r.t FCFS algorithm

are essential for container allocation in cloud computing. These techniques aid in the efficient use of resources and load balancing by clustering related containers together according to their shared attributes. In order to improve load balancing, shorten resource execution times, and boost resource utilization rates all at once, this paper proposes a

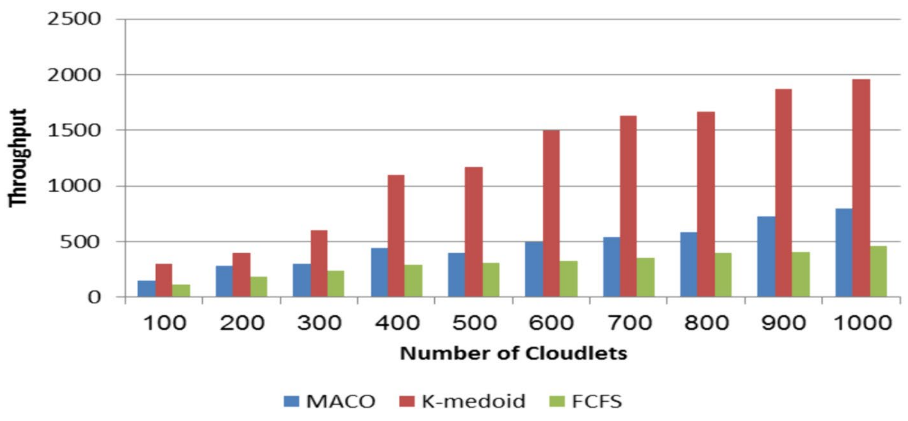

Fig. 4 Throughput

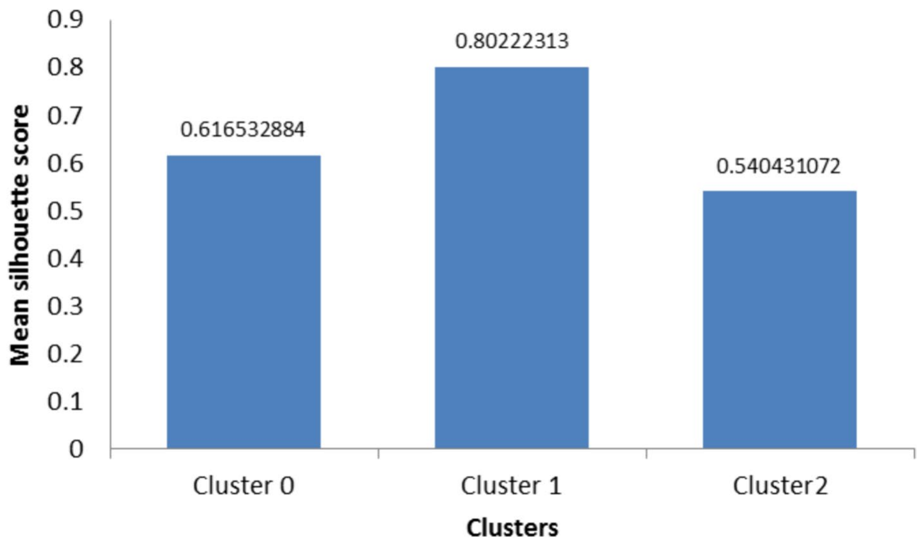

Fig. 5 Mean silhouette score of clusters

Table 3 Mean silhouette score significance

Cluster number	Silhouette score	Significance
Cluster 0	*0.6165*	*Clusters can be overlapped (Medium Clustering)*
Cluster 1	*0.8022*	*Clusters are well-separated and extremely dense (Strong Clustering)*
Cluster 2	0.5404	*Clusters can be overlapped (Medium Clustering)*

Medium clustering significance means that the silhouette score value becomes near zero so the data points of the clusters overlap

Strong clustering significance means that the silhouette score value is near one so data points are well-separated

k-Medoid clustering technique for containers. The experimental results show that the proposed approach performs better than the MACO and FCFS algorithm in terms of throughput and execution time. Although K-medoid clustering algorithm resolves the issues with K-mean clustering algorithm since it is sensitive to noise and outliers but its time complexity compared to k-mean prevent it from processing large amount of data so as a future work the proposed algorithm could be parallelized. In the future, we also can apply the proposed algorithm on a real platform instead of utilizing simulation. We may also try different machine-learning techniques to optimize the placement of containers on virtual machines. Also we can test the proposed model on real workload as a future work. Also we can calculate the mean silhouette score of clusters values for the real workload experiment as a future work.

Abbreviations

SSE	Sum of squared errors
MACO	Modified ant colony optimization
FCFS	First come first serve
HGA	Hybrid genetic algorithm
QoS	Quality of service
PAM	Partitioning around medoids

Acknowledgements
Not applicable.

Author contributions

Amany AbdElSamea (A.A.) is the corresponding author, who was involved in the design of the proposed algorithm, implemented the proposed algorithm, compared the proposed with state of the art. Sherif M. Saif (S. M. S.) has collected the data, encouraged A.A. to study K-medoid clustering algorithm for containers in cloud computing environment, and verified the design method. All authors discussed the results, and A.A. wrote the final manuscript with support from S. M. S. All authors read and approved the final manuscript.

Funding

The authors declare that they have no funding.

Availability of data and materials

The datasets analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Competing interests

The authors declare that they have no competing interests.

Received: 14 July 2024 Accepted: 27 August 2024
Published online: 03 September 2024

References

1. Tian S, Xu K, Ding W, Li Y, Zeng D (2024) An offloading and pricing mechanism based on virtualization in edge–cloud computing. Comput Netw. https://doi.org/10.1016/j.comnet.2024.110468
2. García-Valls M, Cucinotta T, Lu C (2014) Challenges in real-time virtualization and predictable cloud computing. J Syst Architect 60(9):726–740. https://doi.org/10.1016/j.sysarc.2014.07.004
3. Abraham S, Paul AK, Khan RIS, Butt AR (2020) On the use of containers in high performance computing environments. In: 2020 IEEE 13th international conference on cloud computing (CLOUD), Beijing, China, pp 284–293. https://doi.org/10.1109/CLOUD49709.2020.00048
4. Hussein MK, Mousa MH, Alqarni MA (2019) A placement architecture for a container as a service (CaaS) in a cloud environment. J Cloud Comput. https://doi.org/10.1186/s13677-019-0131-1
5. Shrikant K, Gupta V, Khandare A, Furia P (2022) A comparative study of clustering algorithm. Intelligent computing and networking, vol 301. Springer, Singapore. https://doi.org/10.1007/978-981-16-4863-2_19
6. Abdelsamea A (2024) DBSCAN clustering algorithm for efficient container allocation in cloud computing environment. Int J Eng Res Appl 14(6):109–117
7. Fahim A (2023) A varied density-based clustering algorithm. J Comput Sci. https://doi.org/10.1016/j.jocs.2022.101925
8. Radovanović A, Li J, Milanović JV, Milosavljević N, Storchi R (2020) Application of agglomerative hierarchical clustering for clustering of time series data. In: 2020 IEEE PES innovative smart grid technologies Europe (ISGT-Europe), The Hague, Netherlands, pp 640–644. https://doi.org/10.1109/ISGT-Europe47291.2020.9248759
9. Alom R, Mazumdar A, Kr Prasad R, Basumatary G, Baruah B (2022) Analysis of seismic data using partition-based clustering techniques. In: 2022 IEEE global conference on computing, power and communication technologies (GlobConPT), New Delhi, India, pp 1–6. https://doi.org/10.1109/GlobConPT57482.2022.9938362
10. Park H, Jun C (2009) A simple and fast algorithm for K-medoids clustering. Expert Syst Appl 36(2):3336–3341. https://doi.org/10.1016/j.eswa.2008.01.039
11. Muthusamy G, Chandran SR (2021) Cluster-based task scheduling using K-means clustering for load balancing in cloud datacenters. J Internet Technol 22(1):121–130
12. Arora P, Deepali S, Varshney S (2016) Analysis of K-means and K-medoids algorithm for big data. Proc Comput Sci 78:507–512. https://doi.org/10.1016/j.procs.2016.02.095
13. Hafez AM, Abdelsamea A, El-Moursy AA, Nassar SM, Fayek MBE (2020) Modified ant colony placement algorithm for containers. In: 2020 15th International conference on computer engineering and systems (ICCES), Cairo, Egypt, pp 1–6. https://doi.org/10.1109/ICCES51560.2020.9334671
14. Jiang Y, Zhang J (2014) Parallel K-medoids clustering algorithm based on hadoop. In: 2014 IEEE 5th international conference on software engineering and service science, Beijing, China, pp 649–652. https://doi.org/10.1109/ICSESS.2014.6933652
15. Islam MT, Kumar Basak P, Bhowmik P, Khan M (2019) Data clustering using hybrid genetic algorithm with k-means and k-medoids algorithms. In: 2019 23rd International computer science and engineering conference (ICSEC), Phuket, Thailand, pp 123–128. https://doi.org/10.1109/ICSEC47112.2019.8974797
16. Gupta M, Singh D, Gupta B (2023) Modified fire hawks gazelle optimization (MFHGO) algorithm based optimized approach to improve the QoS provisioning in cloud computing environment. Int J Comput Netw Appl 10(3):383–400. https://doi.org/10.22247/ijcna/2023/221896
17. Heidari J, Daneshpour N, Zangeneh A (2024) A novel K-means and K-medoids algorithms for clustering non-spherical-shape clusters non-sensitive to outliers. Pattern Recogn. https://doi.org/10.1016/j.patcog.2024.110639
18. Peng L, Dong G, Dai F, Liu G (2014) A new clustering algorithm based on ACO and K-medoids optimization methods. IFAC Proc 47(3):9727–9731. https://doi.org/10.3182/20140824-6-ZA-1003.01501
19. Piraghaj SF, Dastjerdi AV, Calheiros RN, Buyya R (2017) ContainerCloudSim: an environment for modeling and simulation of containers in cloud data centers. Softw Pract Exp 47:505–521

20. Misso A (2024) Enhanced resource allocation strategies to improve the spectral efficiency in massive MIMO systems. J Electr Syst Inf Technol 11(1):1–16. https://doi.org/10.1186/s43067-023-00132-y
21. Islam R, Sultana A, Islam MR (2024) A comprehensive review for chronic disease prediction using machine learning algorithms. J Electr Syst Inf Technol 11(1):1–28. https://doi.org/10.1186/s43067-024-00150-4
22. Iris dataset. https://gist.github.com/myui/143fa9d05bd6e7db0114