FUZZY LOGIC-BASED DYNAMIC K-MEANS CLUSTERING TECHNIQUE FOR ML-DRIVEN JOB SCHEDULING IN CLOUD ENVIRONMENT

Zeenath Sultana*

Asra Sarwath[1]

Assistant Professor*,Department of CSE,Khaja Bandanawaz University, Kalaburagi, Karnataka, India.

Assistant Professor[1],Department of CSE,Khaja Bandanawaz University, Kalaburagi, Karnataka, India.

profzeenathcse@gmail.com

***CorrespondingAuthor**

Abstract: Cloud computing has become a key technology for providing context-aware, low-latency, and effective computing at the network edge. Efficient task scheduling is essential for optimizing cloud computing systems' efficiency. Traditional task scheduling algorithms, largely developed for centralized cloud environments, often fail to meet the needs of cloud nodes due to their dynamic, diverse, and resource-constrained nature. In order to get over these restrictions, we provide an advanced machine learning-based approach that adjusts work distribution to the constantly changing circumstances of the cloud environment. Our method combines fuzzy logic, a strong unsupervised learning methodology, with the K-Means clustering algorithm to quietly classify cloud nodes according to their workload patterns and resource attributes. The suggested technique assigns jobs to cloud nodes dynamically by fusing fuzzy logic's flexibility with K-means' grouping powers. We demonstrate how machine learning methods can intelligently distribute workloads to cloud nodes, thereby reducing execution time, response time, and network utilization. We demonstrate the efficacy and flexibility of our suggested method in dynamic cloud systems with a series of tests. Clustering is a time-efficient technique that effectively identifies groupings of work per virtual machine (VM). We have modelled and assessed our suggested method using CloudSim. The outcomes of the simulation show how successful our scheduling strategy is, demonstrating notable improvements in response time, reduced execution time, and reduced network utilization in comparison to the CloudSim framework's existing machine learning as well as non-machine learning-driven scheduling methods.

Keywords: Cloud Computing (CC), Machine Learning, Internet of Things (IoT), task scheduling, fuzzy logic, K-mean clustering algorithm, Cloud nodes, symbiotic organism search algorithm(SOS).

1. Introduction

The Internet of Things (IoT) has ushered in an era of ubiquitous data creation, enabling a wide range of applications including smart cities, industrial automation, healthcare monitoring, and more [1]. Nevertheless, a number of challenges stand in the way of efficiently handling and evaluating these massive volumes of data on the network. Cloud computing does this by relocating processing and data storage closer to the data sources. An extension of cloud computing [2] provides a possible solution by lowering latency and bandwidth utilization and facilitating real-time decision-making. The scheduling of activities on cloud nodes is an essential component of this paradigm that requires careful consideration, as it plays a crucial role in guaranteeing timely execution and resource optimization [3]. Cloud computing creates a dynamic and heterogeneous environment in which resources may differ greatly across cloud nodes in terms of processing power and network bandwidth. Furthermore, completing activities often involves a variety of features, including different execution durations and real-time demands [4]. The difficulty lies in effectively coordinating these resources to maximize resource utilization and meet strict job execution time frames. Cloud nodes function in the periphery, where resources are limited and environmental circumstances are unstable, in contrast to centralized cloud environments, which have abundant resources and stable conditions. A precise estimation of task execution durations is a critical component of efficient task scheduling in cloud computing. Cloud nodes are vulnerable to resource variations because of things like network congestion, device malfunctions, and environmental conditions, in contrast to cloud data centres [5], where resource performance is often constant. As a result, work completion times might vary significantly. Between end users and cloud data centres, cloud computing acts as a middle layer. Its usefulness is most noticeable in applications that need fast responses and low latency, depending on where the data originates from. Within this tier, it is easy to establish numerous virtual servers to handle incoming requests. According to , resource allocation[6] is a rigorous procedure that involves distributing resources that are available to cloud clients based on the internet. It is crucial to determine the exact order of these allotments to maximize the benefits of using virtual servers. This deliberate distribution may improve system throughput without charging users excessively. Furthermore, ensuring the availability of resources for high-priority jobs is crucial to prevent them from falling to the bottom of the task queue. Ignoring this could lead to an inefficient use of a company's virtual servers, potentially resulting in financial losses. As a result, a crucial and interesting study subject is the prioritized distribution of resources for profit maximization. The crucial area of machine learning (ML) [7] has advanced significantly in a number of academic fields. Researchers have conducted numerous studies to explore the application of machine learning in addressing cloud computing problems. In recent years, machine learning (ML) has been developed to advance cloud computing programs and provide various cloud-related services. Improved security, effective resource management, decreased latency, traffic modelling, and energy efficiency are just a few of these services. Processing vast volumes of data generated by a multitude of objects, sensors, and devices is necessary in the heterogeneous world of cloud computing.

Not only may real-time processing improve operational effectiveness, but in some cases, it becomes necessary. In order to guarantee the best possible use of resources, sensors, devices, and objects often interact with resources intensively. As a result, cloud computing resource management requires careful thought and execution. This part of the research explores studies that use machine learning algorithms to manage resources in the context of cloud computing. In this work, we provide a new scheduling algorithm specifically for cloud computing job allocation. Our method works well by allocating jobs to virtual machines (VMs), which are in charge of carrying out the response and request models in the cloud computing environment. To do this work, we use fuzzy logic in conjunction with the K-Means clustering algorithm [8] to schedule cloud devices. The following are our work's main contributions:

- We are introducing fuzzy logic-enhanced K-means clustering scheduling into the cloud computing paradigm.
- We are putting the suggested method into practice using the CloudSim simulator [9-10].
- The system has significantly reduced execution, response, and network consumption times, indicating increased operational efficiency.

  The document's organizational structure divides the following sections: Section 2 examines the relevant literature, Section 3 discusses the problem statement and suggested solution, Section 4 covers the application and model in detail, Section 5 elaborates on the suggested workflow, and Section 6 presents experimental data along with a comparison analysis. Section 7 concludes the work.

2. Literature Survey

We can classify the many scheduling techniques currently in use into three main categories: hybrid [11], deterministic [12], and stochastic [13]. Stochastic algorithms, sometimes referred to as non-deterministic algorithms, generate a result based on an objective function and a degree of randomness. Stochastic algorithms are also known as evolutionary algorithms. This category also includes heuristic and metaheuristic algorithms [14-15]. Heuristic algorithms use trial and error to discover solutions, but they are unable to guarantee that the output will be the best one. Cloud computing is an area that is now seeing rapid growth. Although there isn't much research on the topic, the academic community is gradually beginning to focus more on cloud computing job scheduling. Distributed systems [16-17] often use the metaheuristic approach to manage resources and plan tasks. However, the use of scheduling techniques in cloud computing architecture has received little research. Researchers conducted and described in have examined a wide range of methods for placing services in the context of edge and cloud computing. Researchers have tested the atomization concept for service location, as well as sequential and simultaneous processing. We have also tested designs for cloud computers and other functional tools. We have received more information on resource allocation and organization in the cloud environment, along with the various modelling tools available for cloud computing. The scientific community faced the work scheduling issue, prompting the implementation of the Bee Life Algorithm [18-19]. They employed both the CPU's speed of execution and the total quantity of RAM as parameters.

To show how far things had progressed, they contrasted their effort with the work on GA and PSO. The allocation process uses a static approach to job scheduling instead of a dynamic one. We also developed and described a method for work scheduling based on cloud infrastructure and genetic algorithm (GA) mechanisms [20-21]. The suggested approach is better than the Bee Life algorithm, according to the comparison. The authors use a small collection of data to show effectiveness in their article. In [22], they developed a method for deploying cloud device-based services in cloud colonies. After determining a suitable service deployment sequence, this technique uses the GA method to optimize it. The method's limitation is that it only compares the created method to the GA and first-fit methods. We developed the resource scheduling technique known as EPSO specifically for use in cloud computing networks. It merged the proximal gradient method with the PSO strategy to make the convex difficulties more manageable. The developed method employs additional gradient parameters to reduce the difference between the local and global solutions. Comparisons of makespan and total time measures demonstrate the utility of the established technique. Though it costs more energy, the recently developed method reduces the time required to complete a large number of tasks. The study by [23] used the pyrotechnics approach to plan tasks in a setting with a wide range of resources. It uses elements of the fireworks evolutionary technique to get the best scheduling sequences. When the researchers assessed the methods using the genetic algorithm, they saw a quick convergence. To pick the pyrotechnics, the tournament-based selection method was used; however, the algorithm lacked a cloud or an environment specifically designed for clouds. The work [24] introduced the fireworks algorithm, a job scheduling methodology. They have created a technique to identify fireworks-related explosions. Furthermore, after implementing the task clustering mechanism, the authors went on to build the approach. They have included memory and CPU utilization indications in their load utilization model; however, they have not included the time aspect. The study identified energy and time consumption as the most critical factors. The pyrotechnics approach has allowed the writers to effectively address the problems related to commitment and schedule. A study [25] suggests that mobile edge computing has effectively implemented cloudlets. We completed the placement with a focus on efficiency, taking into account the entire ownership period. Their contribution is the process for distributing tasks via cloud computing. To reduce the number of instances of improper work allocation and increase performance, this technique employed two different strategies. We have enhanced both the scheduling and the distribution of load across cloud-based resources. The authors developed a unique paradigm that simultaneously considers three different criteria for work scheduling and resource allocation[26]. These components are referred to as the throughput, job completion time, and resource balancing factor. The Lyapunov drift approach [27] has optimized and reduced the waiting time for jobs, but it lacks a contemporary optimization technique. This method integrates resource allocation and selection in a cloud environment, enabling scheduling to operate on multiple criteria. This method necessitates scheduling to take into account a variety of factors. In order to reduce the task loading factor, the proposed strategy chooses a methodology for quickly finishing jobs; nevertheless, it excludes an optimization technique, which would assist in improving the method's presently subpar performance. Several scholars have attempted to address the

multi-objective optimization problem in workflow applications, utilizing varying numbers of objectives. The present research proposes the GA-PSO hybrid meta-heuristic approach for allocating workflow jobs among the available resources. The suggested approach takes advantage of the characteristics of diverse tasks and nodes; however, it is unable to carry out multi-objective optimization. We contrast the suggested approach with the k-means and Symbiotic Organism Search (SOS) [28] algorithms. The symbiotic interactions among species in an ecosystem serve as the inspiration for the metaheuristic algorithm known as Symbiotic Organism Search (SOS). When it comes to work scheduling in cloud computing, symbiotic organism search (SOS) may be used to identify the best task assignments for cloud nodes while minimizing particular goals, including makespan, cost, and energy usage. SOS may be sensitive to the starting population and difficult to converge on, particularly for complex issues. The K-means algorithm can group tasks together based on attributes such as job duration and resource needs. After grouping, we allocate the jobs to cloud nodes based on the resources available at each node. For instance, cloud nodes with more resources may allocate resource-intensive jobs, and cloud nodes nearest to the data source may allocate jobs with a short turnaround time. When there is little uncertainty in cluster assignments and data points naturally fall into discrete clusters, K-means is a good fit.

3. Problem Statement

The k-mean approach is only useful for task scheduling when every data point is allocated to a single cluster. In order to determine whether, when cluster allocations are somewhat definite and data points naturally fall into discrete clusters, K-means is a good fit.

3.1. Proposed Solution

When job schedules and assignments are unclear, unpredictable, and ambiguous, cloud computing situations in which K-means clustering employs fuzzy logic are useful to get the best outcomes. It takes advantage of probabilistic, pliable membership allocations. It optimizes resource utilization and satisfies a variety of Quality of Service (QoS) criteria.

4. Methodology

In situations where dispersed data and low latency requirements are required, cloud computing leverages the advantages of both cloud and edge computing. IoT sensors are essential to this architecture's base layer because they allow data receipt and transfer via gateways to higher levels. Actuators simultaneously handle system control responsibilities. In practice, cloud computing gives edge devices the ability to pre-process and analyze data, which improves system performance. It is noteworthy that every cloud network application has a unique topology. The CloudSim simulator provides an intuitive Graphical User Interface (GUI) module to aid in the construction of customized and preset topologies. With this GUI, users can add a variety of components to their topological designs, including sensors, actuators, the cloud, cloud components, and connection elements. We demonstrate the practical application of these ideas in a parking case study using CloudSim. Specifically, we show how to create and use a novel topology for this particular

situation. Other simulator modules may interpret and execute these customized topologies. Cloud computing is a key enabler for several important IoT application areas, including smart parking systems for cars. When it comes to cloud computing, creative solutions emerge. One of the main components of cloud computing is smart traffic lights, which are essentially cloud nodes that talk to each other without any problems and efficiently provide warning signals to other cars. This collaboration leverages several communication technologies, such as 4G, zigbee, roadside infrastructure, and smart traffic signals, to enhance connectivity between cloud computing companies and vehicle access points. The distributed video data at the center of this case study comes from many different areas, such as manufacturing, security, transportation, and healthcare. In this case, the application model consists of five separate roles:

Data processing: To recognize particular objects and identify motions in front of the cameras, the system examines raw video inputs.

Object Tracking: We use an object tracking module to compute the settings necessary for efficient monitoring.

Configuration: By changing the camera's location, it improves its surveillance capabilities. Actuators and real cameras are both involved in this process.
Figure 1 depicts the DCNS data application architecture, which symbolizes the physical topology of "case study-A." This architectural arrangement considers every element within the dotted line essential, and houses cloud devices within a specific box. The architecture designates Module 1 as "M1".

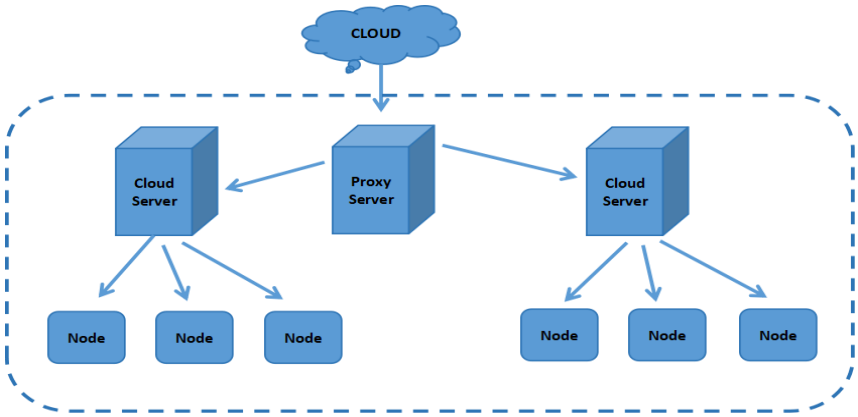

Figure 1. Case study A data source to cloud.

5. Proposed Model

5.1. The determination for the number of clusters

If all you wanted to do was send tasks with output and input file sizes as job parameters or photos as data, varying numbers of instructions, and varying CPU lengths (510, 2,100) in MIPS, RAM (510, 2,100) in MB, as well as bandwidth as cloud node parameters, then there would be nine clusters. RAM, bandwidth, and CPU length are the three factors affecting the task's reaction time, execution time, and network use.
Cluster 1: Cloud nodes with CPU lengths between 510 and 1,100 MIPS, RAM between 210

and 1,100 MB, and bandwidth between 10 and 100 Mbps.
Cluster 2: Cloud nodes with (10, 100 Mbps) of bandwidth, RAM (10, 1, 500 MB), and CPU lengths (1,000, 1, 500 MIPS).
Cluster 3: Cloud nodes with high CPU, high RAM, low bandwidth, and CPU lengths between 1,500 and 2,000 MIPS, 1,500 and 2,000 MB of RAM, and 10 to 100 Mbps of bandwidth.
Cluster 4: Cloud nodes with (500, 1,000) MIPS CPU lengths, (200, 1,000) MB RAM, and (100, 1,000) Mbps bandwidth.
Cluster 5: Cloud nodes with (1,000, 1,500) MIPS CPU lengths, (1,000, 1,500) MB RAM, and (100, 1,000) Mbps bandwidth.

5.2 Task allocation using fuzzy logic

Cloud computing task allocation uses fuzzy logic to distribute jobs to cloud nodes with different membership levels. This method allows for more flexibility and adaptability in selecting the most appropriate cloud node for a specific operation. Here's an extended, equation-based explanation:
Step 1: membership functions and fuzzy sets Fuzzy logic uses fuzzy sets to express qualities, and a membership function defines each set. These membership functions determine the degree to which an element belongs to a set. You can use fuzzy sets to represent any cloud node and task attribute, execution time (including response time), and bandwidth for job assignment. The membership functions give each element x, represented by μ(x), a membership degree ranging from 0 to 1. If a task has an execution time attribute, it is possible to create a fuzzy set called "LowExecutionTime" with an equation for membership called μ(LowExecutionTime), where μ(LowExecutionTime)(x) reflects the job's execution time as measured in terms of how "low" it is.
Step 2: Using the properties of each work allocated to a cloud node, we first determine the membership degree of each task. To distribute tasks across cloud nodes, determine the degree of membership. This degree indicates a cloud node's suitability for a certain job. There are several techniques to determine the association degree μ(A)(x) of an element x that is a member of the fuzzy set A. These approaches include the use of trapezoidal functions and the triangular Gaussian. A general equation using a Gaussian membership function is as follows:

$$\mu(A)(x) = \exp(-0.5*((x-c)/\sigma)^2)$$

A represents the fuzzy list ("less execution of time").
X represents the attribute's value, such as the execution time of a task.
The fuzzy set's center, denoted by c, represents a special value for the list.
Sigma controls the breadth or spread of the membership function.
Step 3: Fuzzy rules and several properties Many factors, such as bandwidth, execution time, and reaction time, are often considered when allocating tasks. Each attribute may have its own membership functions and fuzzy sets.
Step 4: Membership degree totals combined: After computing membership degrees for distinct qualities, we combine these degrees to determine the overall membership level of the job in relation to a cloud node. The aggregation may be carried out utilizing functions like the minimum or maximum (AND) as well as minimum or maximum (OR) operators, based on the inference approach used.

5.3 The updated version of K-means

The suggested method improves on the conventional K-means algorithm by including fuzzy logic for work scheduling in cloud computing settings. The following are the main phases in this improved K-means algorithm:

Step 1: Initialization: Set the fuzziness parameter and initialize cluster centroids, which stand in for cloud nodes.

Step 2: Membership calculation: Using factors like resource availability, past performance, and task proximity to each cloud node, determine the membership degrees for each job.

Step 3: Using the membership degrees and the weighted mean of the job qualities, update the cloud node centroids.

Step 4: Verify convergence. Continue to repeat steps 2 and 3 until the convergence criteria are satisfied.

5.4 Coud Simulation Process of workflow

CloudSim is a Python-based simulator designed to create and assess topologies, cloud computing scenarios, and applications. The CloudSim process in action. Initially, we construct the cloud computing environment using the CloudBroker class. Next, we use the create application approach to set up an application or case study that evaluates the performance of sensors, cloud nodes, and actuators in a cloud context. Use the construct CloudDevice function to generate each with distinct characteristics, a specific number of cloud devices, and capabilities. These features include important hardware information such as the node's name, MIPS (million instructions per second), level, RAM (random access memory), ratePerMips, busyPower, and idlePower for both the uplink and downlink. We activate our suggested scheduling technique, named "K-Means Clustering and Fuzzy Logic," after the application submission. This technique clusters tasks or tuples using the strategy from Algorithm 1. The appAllocationPolicy class then schedules the allocation of virtual machines (VMs) to the formed clusters. The simulation contains updates about execution durations, response times, and network use after the scheduling step.

Algorithm 1

Input:

CloudNodes: A collection of cloud nodes, each with a set of properties.

tasks: A collection of tasks, each possessing a unique set of properties

numClusters: The total number of cloud clusters using K-means clustering

Fuzziness factor: greater than 1 fuzzy logic fuzzy factor

maxIterations: K-means and fuzzy logic provide the maximum number of iterations.

Output:

taskAssignments: An array enumerating the cloud nodes assigned to each task.

1.Give the cloud nodes a K-means clustering application with the given number of clusters (numClusters). This procedure results in cloud centroids and clusters.

2.Create a blank array called taskAssignments, whose sizes correspond to the total number of

tasks.
3.Repeat these steps until you reach convergence or maxIterations, the maximum number of cumulative iterations:
4.Assign tasks to each cloud cluster's designated cloud node (centroid).
5.Return the taskAssignments array, which allocates each task to a different cloud node using fuzzy logic and K-means clustering.
End

First, we use K-means clustering to create clusters with centroids from the cloud nodes. Subsequently, we compute membership degrees of tasks for every cloud cluster using fuzzy logic, make updates to the cluster centroids according to the membership degrees of tasks, and assign jobs to cloud clusters at the end. Next, we assign each cloud cluster's tasks to its unique cloud node, also known as the centroid. In cloud computing settings, this hybrid technique uses fuzzy logic and K-means clustering to make well-informed recommendations about job distribution. We start the process of establishing the application and the cloud broker. Every camera and location-specific region generates a cloud device. Once we create cloud devices, we present these apps to the cloud broker. During the creation stage, the cloud broker integrates the application into itself. Module mapping occurs, and then CloudSim starts up. This includes module scheduling for virtual machine allocation via clustering.

6. Experiments and Results

This study's simulations used the CloudSim library. CloudSim, a Java-based library, houses classes and modules specifically designed to simulate cloud computing scenarios. In order to do their job, the CloudSim package and its associated classes are essential for users who are already familiar with the Cloudsim library. The machine has to have an Intel Core i5 CPU, 4GB of RAM, and Windows 11 installed in order to run the program properly. As part of the testing phase, we will use two different case studies to build the new scheduling method.

TABLE 1. Features of task input

Parameters	Units	Value
No. of instructions	Instructions	$(1000)*10^5$
Size of input file	mb	(11,000)
Size of output file	mb	(11,000)

TABLE 2. Features of cloud nodes

Parameters	Units	Cloud node
CPU Length	MIPS	(510,2100)
Bandwidth	mbps	(11,11,000)
RAM	mb	(510,2100)

Table 1 summarizes the experiment parameters. It demonstrates that we created eight scenarios for experiment 1, increasing the number of data inputs in each scenario by a factor of 100. Table 2 summarizes features of cloud nodes using the parameters CPU length, bandwidth and RAM.

We conducted a total of eight simulation runs for the data and three methods: K-means clustering, symbiotic organism search (SOS), and K-means clustering with fuzzy in the DCNS framework. The primary goal of our comparison is to identify the optimal results for every case study under identical circumstances. Table 3 shows the response time and clearly shows that DCNS K-means clustering with fuzzy logic and our proposed scheduling method work better than DCNS K-means clustering and DCNS (SOS), the two existing scheduling methods.

Table 3. Real-time execution timings, including network use

Data	DCNS SOS Response time	DCNS K-means Response time	DCNS K-means Fuzzy Response time
100	1567.25	828.48	410
300	10759.82	8533.93	4025
500	37530.17	23607.65	17520
700	55212.74	33541.56	27120
900	54136.25	33895.25	25201

The DCNS case study data were gathered from five CloudSim simulation runs. These findings include network usage in kilobytes (KB) and response time in milliseconds (ms). This is done to facilitate a comparison between the proposed K-means with fuzzy scheduling approach and the existing DCNS K-means clustering and DCNS (SOS) system.

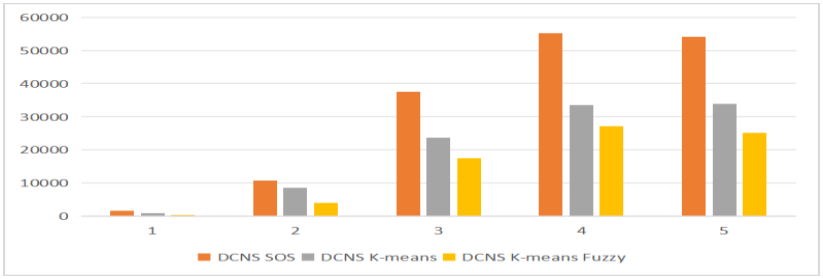

FIGURE 2 : The DCNS K-means, DCNS (SOS), and the suggested DCNS K-means fuzzy are compared for execution duration.

7. Conclusion

In this study, we proposed a unique way to work with scheduling in cloud computing settings using fuzzy logic to improve the K-means clustering technique. Fuzzy logic's capacity to adapt to uncertainty and unpredictability, in addition to its clustering characteristics, has proven useful in increasing task scheduling efficiency, decreasing execution and response times, and using less network bandwidth while also optimizing resource consumption. Our method, which makes use of machine learning methods, shows impressive reductions in execution time, response time, and network utilization in dynamic cloud settings. In order to find more appropriate virtual machines for allocation, this method sticks to the task clustering technique and then distributes these ideal clusters to cloud devices. We conduct an experimental examination of this suggested model using the CloudSim toolbox. It is clear from the results that the suggested clustering method works better in CloudSim than k-means

and the current scheduling algorithms (SOS), especially when it comes to reducing execution time, response time, and network usage. In cloud computing, task scheduling continues to be a major difficulty. An intriguing new direction for research and development is the combination of fuzzy logic with K-means clustering. Subsequent investigations might concentrate on augmenting the scalability and resilience of our suggested methodology, along with its suitability for diverse cloud computing situations. All things considered, the suggested task scheduling technique shows promise for use in cloud computing applications. We haven't tested the suggested technique in every cloud computing scenario yet; it's still in the development stage. Designing fuzzy logic rules that take into account the capabilities of cloud nodes, job requirements, etc. is a highly difficult undertaking, as failing to do so would restrict the suggested algorithm's ability to find the best answer.

References

[1] Kumar, S., Tiwari, P., and Zymbler, M. (2019), "Internet of Things is a revolutionary approach for future technology enhancement: a review*", J. Big Data 6*, 111, doi: 10.1186/s40537-019-0268-2.

[2] Golightly, L., Chang, V., Xu, Q. A., Gao, X., & Liu, B. S. C. (2022). Adoption of cloud computing as innovation in the organization. *International Journal of Engineering Business Management*, *14*. https://doi.org/10.1177/18479790221093992.

[3] Ghani Alyouzbaki, Y. A., & Al-Rawi, M. F, "Novel load balancing approach based on ant colony optimization technique in cloud computing", *Bulletin of Electrical Engineering and Informatics*, *10*(4), 2320–2326,2021, https://doi.org/10.11591/EEI.V10I4.2947.

[4] Guo, C., and Chen, J. (2023),"Big data analytics in healthcare," in Knowledge Technology and Systems: Toward Establishing Knowledge Systems Science, eds Y. Nakamori and H. Nomiya (Singapore: Springer Nature Singapore), 27–70.

[5] Sultana, Z., Gulmeher, R., & Sarwath, A. (2024), "Methods for optimizing the assignment of cloud computing resources and the scheduling of related tasks", *Indonesian Journal of Electrical Engineering and Computer Science*, *33*(2), 1092–1099. https://doi.org/10.11591/ijeecs.v33.i2.pp1092-1099.

[6] Senthilkumar, G., Tamilarasi, K., Velmurugan, N., & Periasamy, J. K. (2023), "Resource Allocation in Cloud Computing", *Journal of Advances in Information Technology*, *14*(5), 1063–1072. https://doi.org/10.12720/jait.14.5.1063-1072.

[7] Asra Sarwath *et al* (2024), "Spark Mllib intrusion detection mechanism using machine learning models ", *IJEECS*, Vol 33,No 2, Feb 2024, http://dx.doi.org/10.11591/ijeecs.v33.i2.pp1235-1242.

[8] Anam, S., Fitriah, Z., Hidayat, N., & Maulana, M. H. A. A. (2023), "Classification Model for Diabetes Mellitus Diagnosis based on K-Means Clustering Algorithm Optimized with Bat Algorithm", *International Journal of Advanced Computer Science and Applications*, *14*(1), 653–659. https://doi.org/10.14569/IJACSA.2023.0140172.

[9] Hicham, G. T., & Chaker, E. A. (2016), "Cloud computing CPU allocation and scheduling algorithms using cloudsim simulator", *International Journal of Electrical and Computer Engineering*. Institute of Advanced Engineering and Science. https://doi.org/10.11591/ijece.v6i4.10144.

[10] Al Rostom, G. (2019), "Simulating fault tolerance in cloud computin by migration using cloudsim simulator X", *Journal of Theoretical and Applied Information Technology*, *97*(12), 3425–3435.
[11] Bhaumik, H., Bhattacharyya, S., Nath, M. D., and Chakraborty, S. (2016), "Hybrid soft computing approaches to content based video retrieval: a brief review", *Appl. Soft Comput*. 46, 1008–1029. doi: 10.1016/j.asoc.2016.03.022.
[12] Hosseinzadeh, M., Azhir, E., Lansky, J., Mildeova, S., Ahmed, O. H., Malik, M. H., et al. (2023), "Task scheduling mechanisms for fog computing: a systematic survey", IEEE Access. doi: 10.1109/ACCESS.2023.3277826.
[13] Saif, F. A., Latip, R., Hanapi, Z. M., and Shafinah, K. (2023), "Multi-objective grey wolf optimizer algorithm for task scheduling in cloud-fog computing", *IEEE Access 11*, 20635–20646. doi: 10.1109/ACCESS.2023.3241240.
[14] Pan, J. S., Hu, P., Snášel, V., & Chu, S. C. (2023), "A survey on binary metaheuristic algorithms and their engineering applications", *Artificial Intelligence Review*, *56*(7), 6101–6167. https://doi.org/10.1007/s10462-022-10328-9.
[15] M. Almufti, S., Ahmad Shaban, A., Arif Ali, Z., Ismael Ali, R., & A. Dela Fuente, J. (2023), "Overview of Metaheuristic Algorithms", *Polaris Global Journal of Scholarly Research and Trends*, *2*(2), 10–32. https://doi.org/10.58429/pgjsrt.v2n2a144.
[16] van Steen, M., & Tanenbaum, A. S. (2016), "A brief introduction to distributed systems", *Computing*, *98*(10), 967–1009. https://doi.org/10.1007/s00607-016-0508-7.
[17] Shabani, I., Mëziu, E., Berisha, B., & Biba, T. (2021), "esign of Modern Distributed Systems based on Microservices Architecture", *International Journal of Advanced Computer Science and Applications*, *12*(2), 153–159. https://doi.org/10.14569/IJACSA.2021.0120220.
[18] Bitam, S. (2012), "Bees Life Algorithm for Job Scheduling in Cloud Computing", *Second International Conference on Communications and Information Technology*, 186–191.
[19] Bitam, S., Zeadally, S., & Mellouk, A. (2018), "Fog computing job scheduling optimization based on bees swarm", *Enterprise Information Systems*, *12*(4), 373–397. https://doi.org/10.1080/17517575.2017.1304579.
[20] George, N., Kadan, A. B., & Vijayan, V. P. (2023), "Multi-objective load balancing in cloud infrastructure through fuzzy based decision making and genetic algorithm based optimization",*International Journal of Artificial Intelligence*, *12*(2), 678–685. https://doi.org/10.11591/ijai.v12.i2.pp678-685.
[21] McCall, J. (2005), "Genetic algorithms for modelling and optimization", *Journal of Computational and Applied Mathematics*, *184*(1), 205–222. https://doi.org/10.1016/j.cam.2004.07.034.
[22] Liu, H. (2022), "Research on cloud computing adaptive task scheduling based on ant colony algorithm", *Optik*, *258*. https://doi.org/10.1016/j.ijleo.2022.168677
[23] YUE, H., YANG, Y., LU, Y., YANG, F., WU, J., RUAN, Q., & DENG, Z. (2022), "Research progress of space non-pyrotechnic low-shock connection and separation technology (SNLT): A review",*Chinese Journal of Aeronautics*,Elsevier B.V. https://doi.org/10.1016/j.cja.2021.07.001.
[24] Yue, Z. H., Zhang, S., & Xiao, W. D. (2020), "A novel hybrid algorithm based on grey wolf optimizer and fireworks algorithm", *Sensors (Switzerland)*, *20*(7).

https://doi.org/10.3390/s20072147.
[25] Abbas, N., Zhang, Y., Taherkordi, A., & Skeie, T. (2018, February 1). Mobile Edge Computing: A Survey. *IEEE Internet of Things Journal*. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/JIOT.2017.2750180.
[26] Mohamed, A., Hamdan, M., Khan, S., Abdelaziz, A., Babiker, S. F., Imran, M., & Marsono, M. N. (2021), "Software-defined networks for resource allocation in cloud computing: A survey",*Computer Networks*, *195*. https://doi.org/10.1016/j.comnet.2021.108151.
[27] Cordeiro, J. D., Kharoufeh, J. P., & Oxley, M. E. (2019), "On the ergodicity of a class of level-dependent quasi-birth-and-death processes", *Advances in Applied Probability*, *51*(4), 1109–1128. https://doi.org/10.1017/apr.2019.43.
[28] Cheng, M. Y., & Prayogo, D. (2014) "Symbiotic Organisms Search: A new metaheuristic optimization algorithm", *Computers and Structures*, *139*, 98–112. https://doi.org/10.1016/j.compstruc.2014.03.007.