Journal of Information Systems Engineering and Management
2025, 10(32s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Adaptive Task Scheduling and Resource Management for Managing Flash Crowds in Cloud Environment

[1*]S.Prathiba, [2] Dr.Sharmila Sankar
[1] Research Scholar, Department of Information Technology, B. S. Abdur Rahman Crescent Institute of Science and Technology, Chennai, India. ORCID Id: https://orcid.org/0000-0002-1257-2589, prathibasakthivel@gmail.com*
2Professor, Department of Computer Science and Engineering, B. S. Abdur Rahman Crescent Institute of Science and Technology, Chennai, India. sharmilasankar@crescent.education

ARTICLE INFO

Received: 29 Dec 2024

Revised: 15 Feb 2025

Accepted: 24 Feb 2025

ABSTRACT

Cloud computing being a master in controlling wide variety of virtual resources, incorporated scheduling which imprinted its footsteps deeply. For every job, multiple such virtual resources from cloud will be utilized. Manual scheduling of resources for each job results in complexity and as such remains an impractical solution. During Resource Allocation (RA), a node's failure could cause interruption of cloud service. Present RA techniques struggle to achieve high throughput in less execution time. To handle flash crowds thereby serving the aforementioned challenge, an appropriate task scheduling algorithm in addition to RA techniques is essential. This work focuses on job scheduling and effective resource allocation for flash crowds that used online streams of multiple user requests as input. During RA, the exact split of data related to a specific user request was recognized and preprocessed. The Closed Frequent Itemset (CFI)from the keywords related to the corresponding query were obtained followed by computation of their corresponding entropy values. Then, the scheduling process is done using Normalized K- means Algorithm (NKMA) and firefly algorithm with respect to the obtained entropy values. Finally, Using Genetic algorithm based on Cauchy Mutations (CMGA) appropriate resources were allocated to the scheduled tasks. The experimental evaluation of the proposed work demonstrates that efficient cloud-based resource allocation and job scheduling can be accomplished, resulting maximum throughput and reduced execution time.

Keywords: Task Scheduling, Resource allocation, Genetic algorithm based on Cauchy Mutations (CMGA)Normalized KMA (NKMA).

INTRODUCTION

Cloud computing provides users easy as well as data access on-demand. It is cost-effective and provides IT services [1,2]. Through cloud computing, resources like storage, servers, software, and networking are hosted and rented via the Internet [3,4]. As cloud infrastructures expand, managing resources in large, diverse, and distributed environments has become a challenging task [5]. Resource management consists of two essential phases: one is resource provisioning and the another one is resource scheduling

Cloud services aim to provide computing, storage, and networking resources to address the requirements of users in remote locations [7]. To enhance both quality of service and resource efficiency [8], Task Scheduling (TS) is utilized to assign tasks to Virtual Machines (VMs), which are a core technology in cloud computing [9]. The fundamental technology in cloud computing is Load balancing (LB) which is centred around the allocation and scheduling of virtual machines (VMs) by assigning them to appropriate servers and distributing resources across server based on demand [10].

The cloud environment presents several challenges. Researchers from academia, industry, and other sectors are exploring key issues such as scheduling and load balancing (LB) that cloud users encounter [11]. A scheduling algorithm typically runs in two modes: batcch mode and real-time mode. The timing of task dispatch is the primary difference between these two modes [12]. Load balancing (LB) refers to distributing the workload evenly across two or more different servers to ensure faster execution and efficient resource usage [13]

As a result, Load Balancing (LB) and Task Scheduling (TS) have become critical issues, receiving significant attention in recent years. In cloud computing, scheduling resources to attain LB in cloud computing and enhance resource utilization is the key research focus [14]. For example, Amazon Elastic Compute Cloud (EC2), an established Infrastructure as a Service (IaaS) provider, uses elastic load balancing to distribute customer requests. In a cloud environment, Recognizing is important for Task Scheduling that incorporates balancing the load is classified in to different categories as an NP-hard problem [15-18]. To efficiently allocate task to computing resources, LB algorithms have been proposed by many researchers.

LITERATURE REVIEW

Martin Bichler et al. [19] discussed about the optimal allocation of virtual servers using various capacity planning challenges in virtualized IT infrastructures and introduced decision models. The way that IT service providers must determine how to partition servers based on predefined objective functions that align with user requirements. The system addressed both the Static Servers Allocation issue with variable workloads and the allocation problem involving a fixed number of servers. Additionally, it provided evaluations in their first experiments based on traces of workloads from an industry partner.

A Genetic Algorithm (GA) scheduling technique was presented by Jinhua Hu et al. [20] to balance the workload of Virtual Machine (VM) resources. By contrasting the existing scheduling approach with the ideal one, the system first computes a cost gene. The scheduling strategy is then decided upon based on this cost gene. To minimize the impact on the system's load after scheduling, the scheduling option through the minimal rate is selected as the ultimate decision. This method minimizes dynamic migration while achieving load balancing at the lowest possible cost. The outcomes show that the approach also improves resource use.
A Genetic Algorithm (GA) was created by Chandrasekaran K. and Usha Divakarla [21] to schedule Virtual Machine (VM) resources on a cloud computing environment. Tasks on virtual machines were scheduled using the GA, which produced efficient load balancing. VM migrations were less necessary by intelligently assigning virtual machines (VMs) to real machines based on a fitness function. The system determined the node's load once the virtual machines were installed and came up with a plan that offered the best load balancing. This method's effectiveness was assessed by analyse with Eucalyptus's Greedy and Round-Robin algorithm
Using heterogeneous multi-cloud environment, Mohanty and Tamanna Jena [22] proposed a Task Scheduling (TS) approach called GA-based Customer-Conscious Scheduling task and Allocation Resources (RA). The main two stages of the algorithm were designated as RA and TS. Resource allocation was done by GA and the shortest task first method was employed for scheduling. The trial findings showed that the model performed better than the current algorithms regarding the multi-cloud platform makespan time and customer satisfaction rate.
An adaptive task allocation approach was presented by Sambit Kumar Mishra et al. [23] for cloud work scheduling. User-provided tasks are initially routed to the Cloud Service Providers alongside other submitted jobs in the work queue. These tasks are then directed to appropriate task queries. Critical jobs are prioritized by SCHEDULER1 for urgent and SCHEDULER2 for urgent IO-bound tasks.The results indicate that this methodology exhibited energy efficiency in the cloud environment when compared to alternative alternatives.
For resource allocation (RA), Raed and Muamer [24] suggested combining an efficient load balancing (LB) strategy using adapted dynamic energy efficient cloudlets framework. The Krill Herd optimization method used was that determined by variables such task cost, speed, and weight, to improve LB. In addition, to lower service times and energy usage for mobile devices, enhanced dynamic energy efficient mobile cloud computing cloudlets-based architecture was applied. The purpose of effective resource allocation and energy cost awareness were attained with this strategy. The Round Robin, Honey Bees Behaviour-LB, and Krill Herd algorithms were used to compare the outcome of the suggested Krill-LB algorithm.

PROPOSED METHODOLOGY

One common computing is Cloud computing (CC) model in the modern information economy. The cloud must regularly distribute jobs among servers and manage computing resources flexibly to satisfy user demands because it is a complicated system with many servers and users. This article suggests a method for allocating resources and scheduling tasks to manage cloud flash crowds. The first stage of the suggested method uses online streaming data that is seasonal. The cloud receives seasonal requests or inquiries from different users, and each request is broken down into a series of tasks overseen by a task manager. Each user's request data is analyzed to extract keywords and eliminate stop words. There are two preprocessing stages for this divided data: Closed Frequent Itemsets (CFI) are initially identified. The proposed structural design is illustrated in the following Figure 1.

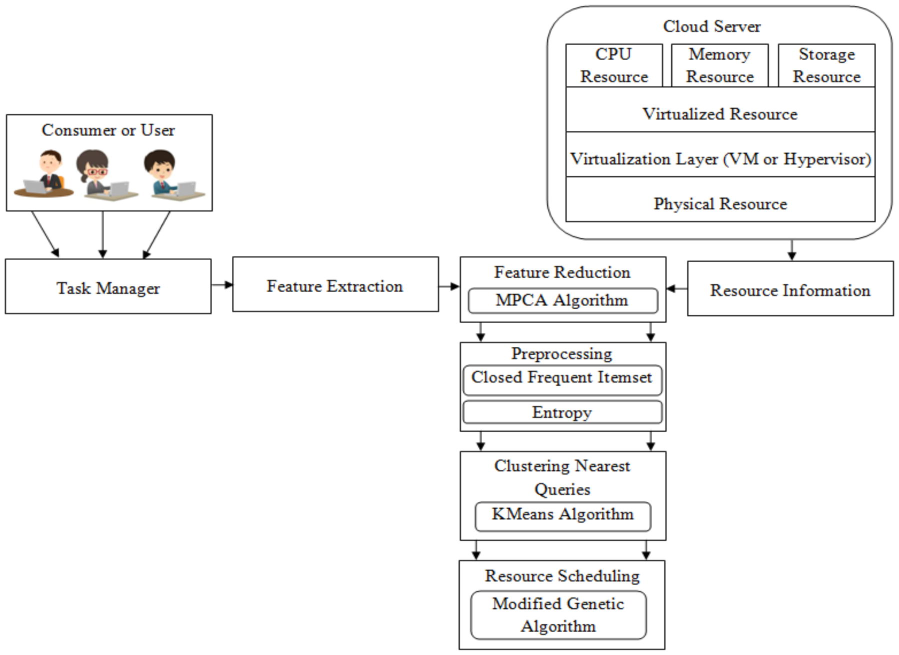
Management of Flash Crowd

Data that is continuously created from multiple sources is referred to as streaming data. Stream processing processes this data piecemeal, avoiding the need to access the full dataset all at once. Another way to define data streaming technology is as a means by which consumers can instantly access content from the internet on their devices without having to wait for it to download. Seasonal online streaming data is examined in this paper, with a particular emphasis on seasonal requests from various consumers.

Task Manager

The Task Managers oversee a series of jobs that are created from the seasonal requests from various users. Process ID, Waiting Time (WT), Task Cost, Turnaround Time (TAT), Speed, Weight, and Size are just a few of the capabilities that are available in the Task Manager. Below is a description of a few of these features.

Figure 1. The Structure of the proposed Model



Speed of the task*:* The speed of the task is refers to the ratio between Turnaround Time (TAT) required for completing a request and the waiting time of the request. It is stated as:

$$S = \frac{T}{W} \quad (1)$$

Where, S represents the speed of the task, T denotes the Turnaround Time (TAT) for fulfilling the request and W is the waiting time for the request.

Task Cost: This refers to the prepayment required before a request can be fulfilled. The cost of the task is computed using the formula:

$$C = R \times T \quad (2)$$

Here C stands for cost of the task. R signifies the rate of the data for the task requirements along with T representing time needed for the task.

Task Weight*:* The weight assigned to a task which is based on both the cost and speed of the request is defined by the following expression:

$$W = k \times \frac{C}{S} \quad (3)$$

Where W represents the task weight, C represent the task cost, S stands for the task speed and K remains a constant value.

Data Size: The datasize for a request is calculated by using a formula:

$$D = \frac{S}{P} \quad (4)$$

Where D represents data size, S denotes the overall proportion of consumer's request along with P denoting the allowable error probability when choosing a small demonstrative subset of the users request.

Preprocessing

Here, users' requests are managed as individual items. Itemset is a group of distinct items. Then, the Closed Frequent Itemset (CFI) for all user requests is identified. The CFI helps to find similar queries from different users. Entropy values are calculated, based on the CFI. Frequent itemsets (FI) are first identified. and then the CFI is computed from the FI, as detailed as follows:

Frequent Itemsets

Take in to account the quantity of items in an itemset as,

$$N = \text{number of items in the itemset} \tag{5}$$

Where N indicates the quantity of items, The Frequent Itemsets (F1) is calculated by the complete number of times for a specific item appears in the itemset. To calculate F1:

$$\mathrm{FI}(i) = \sum_{j=1}^{N} \text{occurrences of item } i \text{ in itemset } j \tag{6}$$

The CFI is then calculated to assess the frequent items with in dataset. This process involves identifying a subset of frequent itemsets known as CFI. It helps in recognizing the large number of frequent itemsets. Once the CFI is determined, all frequent itemsets (F1) can be directly obtained from the CFI collection without needing to re-scan the original data.

$$\mathrm{CFI} = \{\text{itemsets} \mid \text{itemsets are frequent}\} \tag{7}$$

where, CFI represents the subset of frequent itemsets derived from the original dataset.

Entropy of CFI

Empirical entropy is the mean value of an information characteristic. Each attribute information generates a random variable that may reflect the anticipated or average value, termed as

$$\text{Entropy}(\mathrm{CFI}) = -\sum_{i=1}^{n} P(i)\log_2 P(i) \tag{8}$$

In CFI, where P(i) denotes probability of itemset I, n indicates the complete number of itemsets. This entropy formula measures the uncertainty or randomness in the distribution of itemsets with in the CFI.

NORMALIZED K-MEANS CLUSTERING ALGORITHM

In this context, requests from the users are scheduled using NKMA based on entropy values. First, the randomness value of the CFI, which includes both minimum and maximum values, is considered. Normalization is then performed to obtain accurate and useful information about the CFI's entropy value. The normalization is expressed as:

$$N_r = \frac{A - A_{min}}{A_{max} - A_{min}} \tag{9}$$

In this context, N_r represents the normalized value, $A_{\max}$ denotes the maximum value, and $A_{\min}$ indicates the minimum value of the attributes A. Using K-Means clustering Algorithm (KMA), the quantity of clusters and initial centroids are initially established. This methodology has been widely used clustering methodology that operates as an unsupervised learning approach. This method divides the dataset into k predetermined, separate, and non-overlapping clusters in an looping behaviour, allocating each data point to a single cluster precisely. The ideal number of clusters K that achieves the highest level of separation among the clusters is not pre-established and must be derived directly from the data.

The goal of KMA is to minimize the total variance within clusters commonly measured by the su, of squared errors. The procedure for KMA is outlined as follows:

- Start by initializing the number of entropy values $E = \{e_1, e_2, \ldots. e_n\}$ and a set of cluster centers where $c = \{c_1, c_2, \ldots. c_n\}$
- Select k centers of the cluster as (C_c) to partition the selected features in a random manner into n clusters.

- Measure the interval among each data point b and C_c all cluster center, assigning each point to the adjacent center based on the minimum distance.

- Update the cluster centers C_c by determining the average of all data points b assigned to each cluster.

- Repeat step 3 with the new cluster centers. If the cluster assignments b change, continue iterating; otherwise, terminate the process.

The Euclidean distance (ED) is the interval among two points A(x1,y1) and B(x2,y2) is given by:

$$ED = \sqrt{\sum_{i=1}^{n} (x_{i2} - x_{i1})^2} \tag{10}$$

To determine the ideal quantity of clusters k in Normalized K-Means algorithm, the Silhouette Method is used. For centroid initialization, the k++ method is employed. After clustering the queries using entropy values, the clusters are organized using the Firefly algorithm.

CAUCHY MUTATION-BASED GA

Through CMGA, the requests from the user are distributed among the cloud servers. Multiple servers are available in which each server has different information pertaining to these requests. A cluster of requests is sorted using an optimization technique to determine which requests are the most appropriate. The chosen requests or resources are then distributed among several cloud servers, each of which has data pertinent to the request that was selected. Which request, with all the necessary information, gets assigned to a cloud server is decided by the system. This ensures that the tasksare allotted to the active cloud server that is in use. Following allocation, the relevant data pertaining to the user's request is given. Resources are allocated in an economically efficient way among various workloads.

CMGA

Genetic Algorithms (GAs) are a method designed to address both constrained and unconstrained optimization problems. They mimic the natural selection process observed in biological evolution. By repeatedly modifying a population of individual solutions, GAs drive a series of points towards the optimal solution.

In Genetic Algorithms (GAs), solutions are represented as chromosomes, and a collection of these solutions is referred to as a population. The algorithm begins with an initial population of solutions. These solutions are designated based on their suitability, and a new population is created with the expectation that it will be better than the previous one [27] This process continues iteratively until an improved solution is found or a specified condition is met [28]

A random mutation process is applied in Genetic Algorithms (GAs), based on the mutation rate is mentioned in equation (25). However, one common matter with GAs is the occurrence of local optima in various optimization problems. To tackle this issue, the integration of Cauchy Mutation technique has led to development of Cauchy Mutation Genetic Algorithm (CMGA). CMGA process includes the following phases:

Step 1: Assign the positions of the chromosomes with in the population as $P = \{p1, p2, ...pn\}$.

Step 2: By applying the formula, determine the fitness value for each chromosome in p

$$F_t = \frac{zT_f}{\sqrt{z + z(z+1)T_b}} \tag{11}$$

whereby T_f , denotes the backward and forward time delay, implies a constant value, and shows the threshold value. If so, the algorithm will be ended.
Step 3: Continue doing so until a new population is created.
By applying the equation below, choose "2" parent chromosomes according to the fitness value of the population.

$$H = \alpha F_t + \frac{\left(e - e^{z/z}\max\right)}{\left(e + e^{z/z}\max\right)} \tag{12}$$

where, H denotes next generation's the fitness value. The chromosomes are selected using the equation (22)

A crossover probability is used to cross over the parents to produce a new offspring [29]. If the cross-over happens, children will be identical copies of their parents. The adaptive cross-over probability may be determined in one of the following ways:

$$P_{cr} = \left\{ \begin{matrix} P_{cr1}, F_{ag} > F', P_{cr2} \leq 1 \\ P_{cr1}\left(F_{mx} - F'\right) \Big/ \left(F_{mx} - F_{ag}\right) F_{ag \leq} F', P_{cr1} \leq 1 \end{matrix} \right\} \tag{13}$$

where F' indicates the fitness of a single chromosome that is greater than another during the crossover phase. F suggests the level of fitness of the individual which will be a crossover. $P_{cr1}, and P_{cr2}$ are the metrics, F_{mx} denote the ultimate generation's maximum fitness and F_{ag} average fitness, respectively.

Next, Cauchy mutation is employed to carry out the mutation. Such a mutation operator is used to increase the likelihood of escaping the local optimum. The Cauchy density function, with its center at the origin, is represented by the equation below

$$f\left(P_{cr}\right) = \frac{1}{\pi} \frac{t}{t^2 + \left(P_{cr}\right)^2}, \quad -\infty < P_{cr} < +\infty \tag{14}$$

Here t > 0 represents the proportional parameter and its corresponding distribution function is illustrated below:

$$f\left(P_{cr}\right) = \frac{1}{2} + \frac{1}{\pi} arctan\left(\frac{P_{cr}}{t}\right) \tag{15}$$

In order to prevent local optima, this mutation increases population variety and facilitates large-scale searches by producing random numbers over a broad range. Therefore, in comparison to conventional Genetic Algorithms, the Cauchy Mutation-Genetic Algorithm (CMGA) converges more quickly.

Using a Java implementation, we examine the suggested Resource Allocation (RA) scheme's performance for flash crowds in this phase. This work addresses seasonal user requests that pertain to specific fields during particular times of the year, such as election-related queries during elections or cricket-related queries during the World Cup. For this evaluation, we use user requests related to the National Football League (NFL) and process these NFL-related requests. Process ID, Waiting Time (WT), and Turnaround Time (TAT) are included with every user request. First, Evaluate the proposed NKMA algorithm performs by comparing other mythologies like KMA and fuzzy C-means (FCM), which are used in the clustering process. Next, we assess how well the suggested CMGA technique for RA performs in comparison to other approaches, such as PSO and GA.

Figure 2. Pseudo code for CMGA

```
Input: Entropy values
Output: Optimized Entropy values
Begin
Initialize the position of n chromosomes in the population as P={p1,p2,....pn}
Compute the fitness value for each chromosome using equation (22)
While iteration ≤ max iteration do
    Iteration=iteration+1
    Select two parent chromosomes according to their fitness
    Perform crossover with the probability $P_{cr}$
    Perform Cauchy mutation using equation (25)
    Update the population for next generation
End while
End
```

RESULTS AND DISCUSSION

Performance analysis of NKMA

As indicated in Table 1, the effectiveness of the suggested NKMA methodology is assessed by contrasting it with other techniques, such as KMA and FCM, in terms of precision (Prc), f-measure (Fms), accuracy (Acr), recall (Rca), and clustering time (Ct). The evaluation of these factors is based on the values of "true positive," "true negative," "false positive," and "false negative."

Table 1. Performance Analysis of proposed CMGA and existing Methods

(a)

Number of tasks	Proposed NKMA		KMA		FCM	
	P_{rc}	R_{ca}	P_{rc}	R_{ca}	P_{rc}	R_{ca}
5000	87.6588	91.0235	86.47856	90.5689	84.6332	90.2345
6000	90.9874	92.2658	90.67885	91.3256	90.1244	90.1245
7000	91.8856	92.3258	91.67885	91.4578	91.1248	90.7415
8000	93.9856	93.8978	92.85742	92.5645	92.1475	91.2356
9000	96.7856	96.2356	95.87445	95.6321	94.6352	94.5645

(b)

Number of tasks	Proposed NKMA		KMA		FCM	
	F_{ms}	A_{cr}	F_{ms}	A_{cr}	F_{ms}	A_{cr}
5000	90.98567	91.8645	90.4578	90.3256	90.1247	85.6245
6000	91.65784	92.8547	90.7454	91.7845	90.1523	87.6545
7000	93.42564	94.8965	92.1245	93.5678	91.4578	89.8475
8000	94.23554	95.6587	93.4578	93.7548	92.1247	91.8475
9000	95.12457	95.8741	94.2547	94.6589	93.4578	93.5645

Therefore, the proposed NKMA yields superior results compared to other techniques. The precision (Prc), recall (Rca), accuracy (Acr), and f-measure (Fms) performance metrics for the proposed NKMA strategy are shown in Table 1 along with comparisons to the current KMA and FCM approaches over a range of task counts (5000–9000). The NKMA shows the greatest values for 5000 tasks: Acr (91.8645), Fms (90.9856), Rca (91.0235), and Prc (87.6588). Lower values are found using the KMA and FCM methods, respectively: Acr (90.3256 and 85.6245), Fms (90.4578 and 90.1247), Rca (90.5689 and 90.2345), and Prc (86.4785 and 84.6332). The results show that the NKMA continues to beat the current KMA and FCM techniques for task counts between 6000 and 9000. Consequently, when compared to alternative methods, the suggested NKMA produces better outcomes.

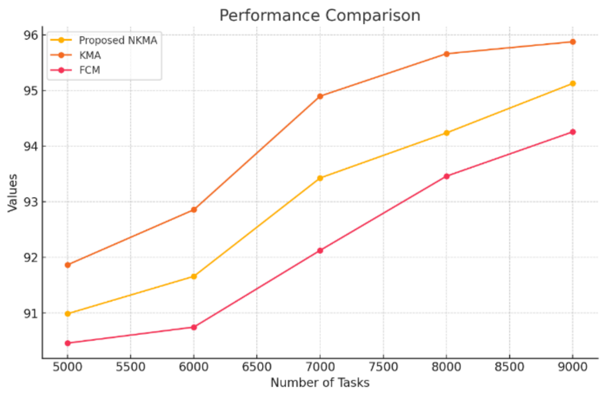


Figure 3. Performance Analysis of clustering time

The above Figure 3 compares the clustering time values of proposed NKMA approach with existing KMA and FCM methods across different task numbers. For all task sizes ranging from 5000 to 9000, the NKMA consistently shows shorter clustering times compared to KMA and FCM.

By contrasting it with current GA and PSO approaches in a number of measures, including response time, average waiting times (WTs), process time, latency, throughput, and average turnaround times (TATs), the effectiveness of the CMGA technique is assessed. The ensuing sections contain thorough explanations of each metric as well as performance analyses of both suggested and current methods.

Response Time and Process Time

The technique's response time R_t is the amount of time it takes to reply to a user request, and it is assessed in the manner described below:

$$R_t = \frac{N_{rs}}{N_{rq} - Av_{tt}} \tag{16}$$

Here, the number of resources represented as N_{rs}, Number of requests denoted as N_{rq} and AV_{tt} refers average travel time

The processing time P_t shows the variation between the total requests time to travel and the time taken to respond which is evaluated as,

$$P_t = T_{tr} - T_{rs} \tag{17}$$

Table 2. Performance metric of Response and Processing time

Number of tasks	Proposed CMGA		Existing GA		Existing PSO	
	R_t	P_t	R_t	P_t	R_t	P_t
5000	4374	6012	5544	7122	6321	8896
6000	5784	9224	6384	10556	7124	11785
7000	6021	15177	6847	17554	7952	19553
8000	6328	21661	7132	23557	8472	25778
9000	7124	29885	7966	30885	8989	32556

Table 2 presents a comparison of response time (Rt) and process time (Pt) between the CMGA approach, GA, and PSO. While GA and PSO take longer, recording 5544 ms for Rt and 7122 ms for Pt, and PSO recording 6321 ms for Rt and 8896 ms for Pt, CMGA completes 5000 tasks in 4374 ms for Rt and 6012 ms for Pt. In comparison to CMGA, the current approaches continue to exhibit greater Rt and Pt for task sizes ranging from 6000 to 9000. Based on this data, it can be summarized that the suggested CMGA outperforms than the current methods.

Average waiting time

Waiting Time of the task (WT) is the duration of a task which remains in the line sequence (queue) of each assigned virtual machine before it can be executed. The mean waiting time A_{wt} is determined using the following formula:

$$A_{wt} = \frac{\sum T_{wt}}{n} \tag{18}$$

Table 3. Average waiting time in Milliseconds

Number of tasks	Existing PSO	Existing GA	Proposed CMGA
5000	580	400	300
6000	620	600	400
7000	980	800	550
8000	1200	980	800
9000	1500	1400	1180

Table 3 presents the performance metrics of the proposed CMGA methodology in terms of average waiting time over a variety of task counts, spanning from 5000 to 9000, in contrast to GA and PSO. GA typically takes 400 ms to complete 5000 jobs, PSO takes 580 ms, and CMGA takes 300 ms. The CMGA algorithm generates an average waiting time of 1180 ms for 9000 actions, while the PSO and GA algorithms need 1500 ms and 1400 ms, respectively. This study shows that the suggested CMGA system provides similar performance, but with quite extended waiting time for job assigned in comparison with current systems.

The average waiting time (WT) for a range of work counts, from 5000 to 9000, is compared between the suggested CMGA and the current GA and PSO in Figure 4. The CMGA has a WT of 292 ms on average for 5000 tasks, but the average WTs of PSO and GA are higher at 548 ms and 412 ms, respectively. Likewise, when it comes to task numbers between 2000 and 5000, CMGA maintains its minimum average WT when compared with GA and PSO. When compared to other solutions, this investigation shows that the suggested CMGA delivers noticeably shorter waiting times for work allocation in a cloud server.

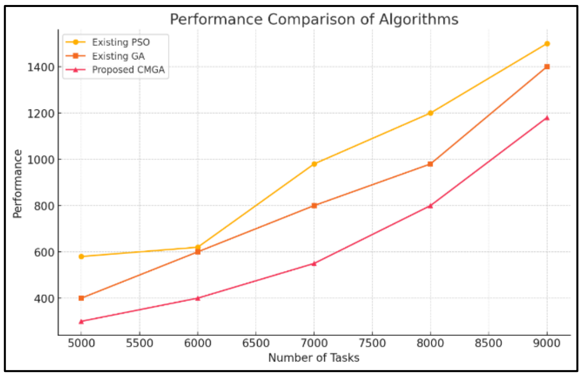


Figure. 4 Performance metric of Average Waiting Time

Average turnaround time

Total Active Time (TAT) is the cumulative duration needed for a process to transition from its starting stage of preparedness to its final state of completion. The sum of the execution time and WT is determined by mathematical calculation.

$$A_{tat} = \frac{\sum [E_{time} + W_{time}]}{N_{tasks}} \qquad (19)$$

where, W_{time} and E_{time} - waiting and execution time of the network,

N_{tasks} - Number of jobs

Table 4. Average Turnaround time in milliseconds

Number of tasks	Existing PSO	Existing GA	Proposed CMGA
5000	580	400	250
6000	750	600	400
7000	900	800	700
8000	1190	980	880
9000	1600	1380	1210

Table 4 presents an average turnaround time (TAT) comparison between the CMGA approach and the current GA and PSO methodologies. The suggested CMGA obtains an average TAT of 250 ms for 5000 tasks, while the TATs of GA and PSO are higher, at 400 ms and 580 ms, respectively. CMGA has average TATs of 400 ms, 700 ms, 880 ms, and 1210 ms for task sizes between 6000 and 9000, whereas GA and PSO show higher TATs: GA with 600 ms, 800 ms, 980 ms, and 1380 ms, and PSO with 750 ms, 900 ms, 1190 ms, and 1600 ms. This comparison shows that, in comparison to CMGA, the TATs of both the current GA and PSO approaches are higher. As a result, the suggested CMGA shows a task scheduling system that is more efficient.

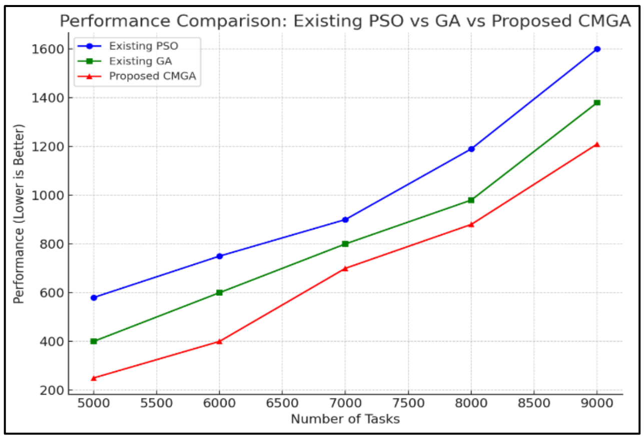


Figure. 5 Average Turnaround Time Performance metric

Table 5. Latency Comparison in ms

Number of tasks	Existing PSO	Existing GA	Proposed CMGA
5000	2100	1500	1000
6000	3500	2800	2200
7000	4900	3900	3000
8000	5900	4800	4000
9000	7200	6200	5200

The time taken to move a data or request from its source to its destination is known as latency (Lcy). It is a crucial component in determining how effectively a network or execution plan operates and is measured in milliseconds. Since lower latency enables more throughput and more efficient connections, it suggests a more successful approach. Equation (30) is used to calculate the latency

$$L_{cy} = A_{FT} + T_{CD} \tag{20}$$

whereas, A_{FT} is the finishing time taken for tasks, and T_{CD} indicates task completion delay.

Table 5 presents a comparison of the latency (Lcy) metric performance of CMGA, GA, and PSO. CMGA obtains a latency of 1000 ms for 5000 tasks, while GA and PSO achieve greater latencies of 1500 ms and 2100 ms, respectively. Similarly, when compared to GA and PSO, CMGA consistently shows the lowest latency for task numbers of 6000, 7000, 8000, and 9000. This reduced latency for CMGA suggests that its efficiency surpasses that of the current systems.

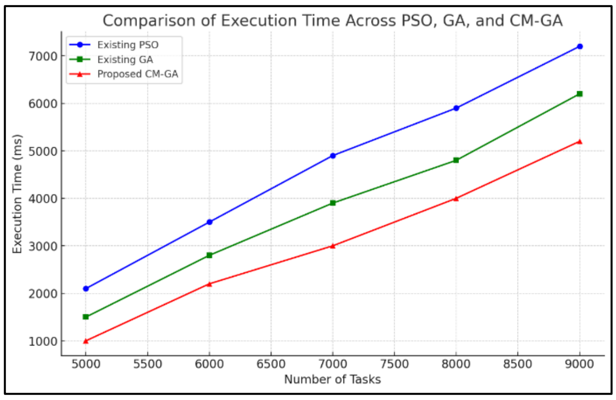


Figure 6. Latency Performance metric for the proposed methodology

Throughput

Throughput (Tp) is a key parameter for evaluaing the network performance as it measures the amount of data transferred over time. It reflects how efficiently tasks are completed. As the proportion of successfully completed jobs rises, throughput also increases. The throughput is calculated using the formula:

$$T_p = \frac{R_{tasks} * \beta}{A_{tst}} \tag{21}$$

wherein, β denotes the quantity of the data, A_{tst} denotes thetotal task simulation time, and R_{tasks} denotes the number of allocated or arrived jobs data at the destination time.

Table 7. Throughput Performance metric for the proposed methodology

Number of tasks	Existing PSO	Existing GA	Proposed CMGA
5000	1000	1100	1150
6000	790	900	1000
7000	590	700	800
8000	400	500	600
9000	200	220	300

In terms of throughput (Tp), Table 6 compares the performance of the suggested CMGA with GA and PSO. Reducing the duration required to finish a task is the main objective of Tp optimization. While GA and PSO achieve lesser throughputs of 1100 and 1000, respectively, for 5000 tasks, CMGA gets a throughput of 1150. In a similar vein, CMGA achieves the maximum throughput of 300 for 9000 tasks, whilst PSO and GA reach throughputs of 200 and 220, respectively. While throughput figures do decline when the overall number of jobs grow, CMGA consistently achieves higher throughput than both GA and PSO. This suggests that the proposed CMGA outperforms than the current systems. See Figure 7 for an illustration of throughput performance.

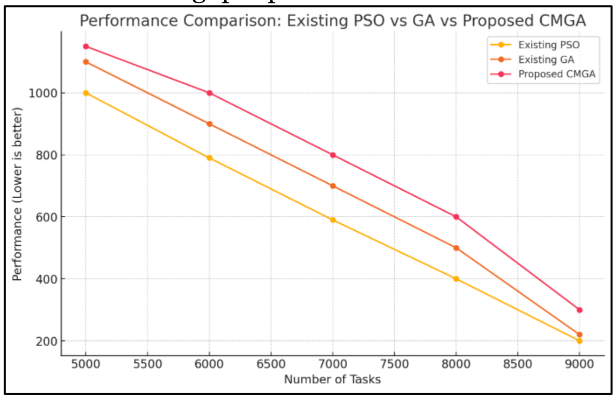


Figure 7. Throughput Performance metric for the proposed methodology

CONCLUSION

This paper proposes a seasonal requests RA approach based on the CMGA technique. Through comparison with the most recent approaches, the outcome of the proposed CMGA methods is evaluated. In specific, the precision (Prc), recall (Rca), accuracy (Acr), f-measure (Fms), and clustering time (Ct) spanning up to 9000 tasks are used to compare the NKMA with KMA and FCM. The NKMA consistently attains the shortest clustering time (Ct) and yields the greatest values for Prc, Rca, Fms, and Acr. Regarding response time (Rt), process time (Pt), average waiting time (Awt), latency (Lcy), and throughput (Tp), average turnaround time (Atat), the CMGA approach is likewise contrasted with GA and PSO. CMGA shows the better performance in terms of the lowest values for Rt, Pt, Awt, and other metrics across all task sizes.

REFERENCES

[1] Sukhpal Singh Gill., Rajkumar Buyya, "Resource provisioning-based scheduling framework for execution of Heterogeneous and Clustered Workloads in Clouds: From Fundamental to Autonomic offering", J of Grid Computing, 17(3):385-417,2019.

[2] Samaresh Bera., Sudip Misra, and Joel JPC Rodrigues, "Cloud computing Applications for Smart Grid: A survey", in 2014 IEEE Transactions on Parallel and Distributed Systems.26(5):1477-1494,2014

[3] Sanjaya Panda K., and Prasanta K. Jana, "An Efficient Resource Allocation Algorithm for Iaas Cloud", in the International Conference on Distributed Computing and Internet Technology Springer Cham 351-355,2015.

[4] Edington Alex., M,Kishore R, "Forensics Framework for Cloud computing in Computers & Electrical Engineering 60", 193-205,2017.

[5] Gill,S.S.Chana,I.,Singh,M.et.al, "CHOPPER: An Intelligent QoS-aware Autonomic Resource Management Approach for Cloud Computing", Cluster Comput 21, 1203-1241,2018.

[6] Sukhpal Singh & Inderveer Chana, "A Survey on Resource Scheduling in Cloud Computing: Issues and challenges" in the Journal of Grid Computing,14, 217-264,2016.

[7] Rosy Aoun., Elias A., Doumith., & Maurice Gagnaire,"Resource Provisioning for Enriched Services in Cloud Environment" in the IEEE Second International Conference on Cloud Computing Technology and Science IEEE,296-303,2010

[8] Prassanna J., Neelanarayanan Venkataraman, "Adaptive regressive holt–winters workload prediction and firefly optimized lottery scheduling for load balancing in Cloud", Wireless Networks 1(1) 1-19,2019.

[9] Jing Xiao., Zhiyuan Wang , "A priority based Scheduling Strategy for Virtual Machine Allocations in Cloud Computing Environment",in International Conference on Cloud and Service Computing IEEE, 50-55,2012.

[10] Keng-Mao Cho Pang-Wei Tsai., Chun-Wei Tsai., Chu-Sing Yang, "A Hybrid Meta-Heuristic Algorithm for VM Scheduling with Load Balancing in Cloud Computing", Neural Computing and Applications,6(6) 1297-1309,2015.

[11] Amer Al-Rahayfeh, Saleh Atiewi, Abdullah Abuhussein., & Muder Almiani, "Novel Approach to Task Scheduling and Load Balancing using the Dominant Sequence Clustering and Mean Shift Clustering Algorithms", Future Internet,11(5) 109,2019.

[12] Mao-Lun Chiang, Hui-Ching Hsieh., Wen-Chung Tsai., Ming-Ching Ke, " An Improved Task Scheduling and Load Balancing Algorithm under the Heterogeneous Cloud Computing Network", in IEEE 8th International Conference on Awareness Science and Technology IEEE. 290-295,2017.

[13] Pradeep K, Prem Jacob, "Comparative Analysis of Scheduling and Load Balancing Algorithms in Cloud Environment", in International Conference on Control Instrumentation Communication and Computational Technologies IEEE. 526-531,2016.

[14] Mala Kalra, Sarbjeet Singh, " A Review of Metaheuristic Scheduling Techniques in Cloud Computing", Egyptian Informatics Journal. 16(3) 275-295,2015.

[15] Jiangtao Zhang, Hejiao Huang, Xuan Wang , " Resource provision algorithms in Cloud Computing",A survey Journal of Network and Computer Applications.64 23-42,2016.

[16] Sanjaya Panda K., Prasanta K. Jana, " Normalization-based Task Scheduling Algorithms for Heterogeneous Multi-Cloud Environment",Information Systems Frontiers,20(2) 373-399,2018.

[17] Jiayin Li, Meikang Qiu, Zhong Ming, Gang Quan, Xiao Qin., Zonghua Gu, " Online Optimization for Scheduling Preemptable tasks on IaaS Cloud Systems",Journal of Parallel and Distributed Computing, 72(5), 666-677,2012

[18] Zhi-Hui Zhan, Xiao-Fang Liu, Yue-Jiao Gong, Jun Zhang, Henry Shu-Hung Chung, & Yun Li, " Cloud Computing Resource Scheduling and a Survey of its Evolutionary Approaches",ACM Computing Surveys, (CSUR), 47(4),63,2015.

[19] Bichler, M., Setzer, T., & Speitkamp, B, " Capacity Planning for Virtualized Servers", 1.Corpus ID: 67783611,2008.

[20] Jinhua Hu., Jianhua Gu., Guofei Sun., Tianhai Zhao, " A Scheduling Strategy on Load Balancing of Virtual Machine Resources in Cloud computing environment" in 3rd International Symposium on Parallel Architectures, Algorithms and Programming IEEE 89-96,2010.

[21] Chandrasekaran K., & Usha Divakarla, " Load balancing of Virtual Machine Resources in Cloud using Genetic Algorithm"in the Proceedings of ICCN 2013.156-168,2013.

[22] Tamanna Jena., &Mohanty J R, " GA-based Customer-Conscious Resource Allocation and Task Scheduling in Multi-Cloud Computing", in the Arabian Journal for Science and Engineering 43(8) 4115-4130,2018.

[23] Sambit Kumar Mishra., Deepak Puthal., Bibhudatta Sahoo., Sajay Kumar Jena., & Mohammad S. Obaidat, " An Adaptive Task Allocation Technique for Green Cloud Computing",in the Journal of Supercomputing, 74(1)370-385,2018.

[24] Raed Abdulkareem Hasan., & Muamer N Mohammed, "A Krill Herd Behaviour Inspired Load Bbalancing of tasks in Cloud Computing", Studies in Informatics and Control 26(4) 413-424,2017.

[25] 25.Mohit Agarwal., & Gur Mauj Saran Srivastava, "A Genetic Algorithm Inspired Task Scheduling in Cloud Computing",in the International Conference on Computing, Communication and Automation (ICCCA), IEEE, pp. 364-367,2016.

[26] 26.Mehran Ashouraie.,& Nima Jafari Navimipour, “Priority-based task scheduling on heterogeneous resources in the Expert Cloud”, Kybernetes 44(10):1455-1471,2015. DOI: 10.1108/K-12-2014-02931.

[27] 27.A. Jayanthiladevi,Mohamed Uvaze Ahamed Ayoobkhan,”Implementation of multicloud strategies for healthcare organisations to avoid cloud sprawl”, International Journal of Cloud ComputingVol. 11, No. 5-6,January 27, 2023pp 529-536https://doi.org/10.1504/IJCC.2022.128699

[28] 28.Mahendra Bhatu Gawali., & Subhash K. Shinde, “Task Scheduling and resource Allocation in Cloud computing using a Heuristic Aapproach”, Journal of Cloud Computing, vol. 7(1) 4,2018.

[29] 29.Bahman Keshanchi, Alireza Souri, and Nima Jafari Navimipour, “An improved genetic

[30] algorithm for task scheduling in the cloud environments using the priority queues: formal verification, simulation, and statistical testing”, Journal of Systems and Software, vol. 124, pp. 1-21, 2017.