Communications on Applied Nonlinear Analysis
ISSN: 1074-133X
Vol 32 No. 3 (2025)

Improving QoS Parameters in Fog Computing Environment using QCSO Framework

Meena Rani[1], Kalpna Guleria[1*] and Surya Narayan Panda[1]
Chitkara University Institute of Engineering and Technology, Chitkara University, Punjab, India[1]
E-mail: meena.rani@chitkara.edu.in
Corresponding Author: guleria.kalpna@gmail.com, kalpna@chitkara.edu.in
Email: snpanda@chitkara.edu.in

Article History:

Received: *27-07-2024*

Revised: *15-09-2024*

Accepted: *30-09-2024*

Abstract:

Cloud computing is a complex infrastructure because heterogeneous devices are involved to share the resources on it. In recent years, the Internet of Things (IoT) is being used by a huge number of different devices which create large amounts of data. Due to their remote location, IoT devices create delays in processing the data. To ensure the Quality of Service (QoS) of the network, the real time applications need low latency and high bandwidth which in turn improves the quality of life. To solve this problem, Fog Computing (FC) is introduced which plays a vital role in computing paradise. Fog computing processes the request locally and uses available resources to reduce the latency and enhance the technological innovation and capabilities. To allocate and manage the resources is the primary goal of fog computing. To achieve this, Queuing theory method is used in which tasks are assigned to the resources. Proposed study presents an algorithm named Queue-Cuckoo Search Optimization Framework (QCSOF) to improve the quality of service through resource optimization. The aim of the proposed framework is to enhance resource allocation, energy utilization.

Keywords: Fog computing, Energy efficiency, Queuing theory, Cuckoo Search, Quality of Service, Resource allocation, Resource use efficiency

1. Introduction

Fog computing is a leading technology which efficiently provides computing resources to the real time applications for processing their data. It leverages Virtual Machine (VM) integration and environmental disruptions [1]. This form of computing is employed due to the impracticality of utilizing cloud computing for numerous IoT applications [2]. Fog computing is a distributed method that meets the needs of the Internet of Things and industrial IoT. Processing and analysing the data on the cloud is more expensive and time consuming as compared to fog computing because FC efficiently manages large amounts of data generated by end devices [3, 4]. Fog computing minimizes data transmission to the cloud, leading to a reduction in bandwidth usage and the corresponding expenses.

Currently, the allocation of resources in virtualized data cores has become a significant focus, as it greatly impacts the energy efficiency of the data core [5].

The utilization of IT resources has been changed by fog computing. Now, instead of obtaining their exact IT resources, they utilize the services offered at a reasonable cost based on a pay-per-use model [6]. FC expands the availability of pay-per-use services such as IaaS, PaaS, & SaaS by utilizing various full-service providers [7].

The key difference between conventional approaches and fog computing approaches is the utilization of services. While the conventional approaches depend on a fixed infrastructure, fog computing leverages all available online resources. Under the conventional approaches, organizations initially

allocate a specific budget towards the acquisition of hardware and software in order to offer services according to their customers' needs [8]. However, accurately predicting the service demands or dynamic workload of customers within a limited time-frame poses a significant challenge. Because of low QoS, organizations could turn away potential clients [9]. Numerous challenges must be overcome in order to deploy fog computing, such as those related to scalability, security, effective load balancing, lowering energy consumption in data centres, guaranteeing service availability, preventing data lock-in, scheduling resource management, and upholding quality of service. Effective management of fog resources can provide several advantages, including improved energy efficiency, reduced network load, increased profit, balanced workload distribution, and minimized violations of Service Level Agreements (SLAs). The scheduling approaches aim to minimize the likelihood of SLA violations and maximize revenue by efficiently allocating resources [10]. The highest level of optimal QoS requirements is guaranteed to clients by the adaptive agent-based SLA monitoring approach [11-12].

FC is crucial for managing the data flow in large and complex networks. Reducing energy consumption and improving overall efficiency are two benefits of using effective and dynamic load balancing devices [13]. Load balancing technology enables the transfer or exchange of workloads between computer nodes [14-16]. Resources are groups of finite virtual or physical parts that are part of a computer system. All devices that are connected and all internal components of a system are considered to be resources. Because fog computing adds many QoS metrics, like CPU memory, processing speed, and system stability, to traditional distributed computing systems, the way resources are allocated is different. In order to demonstrate how throughput and time delay vary between single and numerous server systems in fog computing, a mathematical model based on queuing systems is required [17, 18]. Strong resource constraints necessitate efficient resource allocation in queueing systems.

1.1 Motivation

Allocating resources and dealing with delays are the primary challenges in cloud computing. By creating a load balancing system that prioritizes edge nodes, fog computing overcomes these difficulties. This innovative method addresses energy usage, resource availability, and time allocation, among other QoS metrics, by applying an entropy-based queuing theory optimization methodology.

1.2 Contributions

Resources are assigned to the applications running in the virtual environment. Effective resource allocation depends significantly on timing and sequence. Below is a summary of the contributions made in this work:

- The proposed queuing theory framework enhances resource management in fog computing services by prioritizing resources according to the user's needs and requirements.
- The QoS metrics in fog computing are improved by the proposed method, particularly in the areas of resource allocation, classification, and energy utilization.
- The proposed approach ensures equitable resource distribution among users in the environment.

The main objective of the work is to efficiently view the distribution of resources in a fog environment. Therefore, a fog computing QCSO framework is proposed that effectively controls resource distribution in queue with the task priority. This approach reduces overall cost and improves QoS metrics leading to a significant improvement in effectiveness of the fog services.

The outline of the remaining sections of the paper is as follows: The literature survey is presented in Section 2. The proposed resource allocation QCSO framework is further discussed in Section 3.

Section 4 implies the experimental findings and analysis with details and performance evaluation. Section 5 explains the significant contribution of the proposed work. Lastly, the conclusion and future directions are covered under Section 6.

2. Literature Survey

The literature on quality of service and resource allocation in FC is thoroughly reviewed in this section. Busuyi et al. [19] faced the difficulties in assigning the tasks and deploying virtual machines in a fog computing situation. For effective virtual machine implementation, the authors proposed a genetic algorithm and a unique task assignment method. The focus of the study is assignment of cost, a QoS metric that has been greatly enhanced through the implementation of the algorithm [20, 21]. Fog computing technology has improved the QoS for resource management with its infrastructure-as-a-service. Li et al. [22] developed a fog queuing system which is achieved by assigning a unique queue to each category of task by using the parallel virtual queue technique. Offloading and buffering are examples of parallel algorithms that can be used to optimize resource allocation and increase task completion ratios.

The data cannot be computed in the cloud due to processing delays and scalability issues. The fog computing scheduling mechanisms control the process of resource allocation. The support and confidence method were introduced by Battula et al. [23] as a new way to maximize resource usage in fog computing. Real-time applications were used to test the suggested method, and the outcomes show that resource consumption is reduced when compared to conventional approaches. Due to their internet-based deployment, most cloud services are susceptible to network delay. In Swain et al. [24], authors perform real-time operations at the edge nodes. An innovative method for allocating tasks within the IoT fog network, called Matching-theory-based Efficient Task Offloading (METO), is also included in the architecture. This lowers the rate of task failures and the amount of battery used. Fog computing nodes use edge network deployment to enable effective communication. In this study, Huang et al. [25] examined the problem of allocating resources in an energy-efficient manner in fog computing networks, specifically focusing on the fog nodes. They also developed a method for load balancing that takes into account the limitations of network resources. Seth et al. [26] explored machine learning algorithms to enhance resource management, QoS optimization, and energy efficiency. In a fog environment, authors address the job scheduling issue. This research aims to discover a fair balance between the two performance measures, namely execution time and memory allocation. The authors suggested a bee life algorithm as a swarm-based optimization method which was helpful to attain the objective [27, 28].

Ahmed et al. [29] proposed a resource provisioning scheme that takes into account acceptable levels of intensity. The virtual machine-physical mapping is used to resolve the current issues. The authors were able to achieve a result by implementing an approach to reduce the power consumption up to 32%. The method aligns the resource requirements of the VMs to create a single, unified entity. The outcomes obtained from this approach were better in comparison to the current applications. An approach to improve the fog resource provisioning capacity in local clusters is proposed in this paper. In order to accomplish this, scheduling, resource brokering, and fog sequence dispatching are taken into consideration [30, 31]. In this article authors explained how to maximize quality of service (QoS) delivery, which improves performance and resource utilization in the fog environment. Fog computing required a variety of service-based task scheduling strategies [32, 33].

Task scheduling is essential in fog computing settings. Due to this, each task exists in fog computing has a different cost of resources. Task scheduling is controlled by multiple criteria outlined in the contract between users and FSPs, rather than a single criterion. Ensuring high-quality service to users in accordance with the agreement is a crucial responsibility [34, 35]. In [36], authors represented a

genetic-based method to cut down the time and expense of computation, storage, and resource communication. In [37, 38] authors discussed the core concepts of smart computing, focusing on the IoT, examining how smart computing enables interconnected devices to reduce the total service request latency.

Three processes are involved in allocating resources: resource mapping [39], resource execution as well as resource monitoring [40-42]. An efficient scheduling mechanism is crucial for optimizing server performance and resource utilization [43-46]. Efficient task scheduling became especially critical when multiple tasks simultaneously request fog resources [47,48]. In fog computing, workflow scheduling involves assigning each task to the most appropriate fog node and utilizing multiple resources to provide the desired quality of service [49,50]. Currently, the green fog computing technique is being employed to increase efficiency in resource consumption. It decreases the amount of energy used and as a result, it enhances the power efficiency. Table 1 represents the summary of literature surveys of various significant works.

3. Proposed Resource Allocation QCSO

Framework

The proposed framework collects tasks into a task pool. Computational resources have been distributed and the assignments will be reviewed in the fog computing. The resources distribution is dependent upon their relative weights. This resource distribution makes use of the QCSO framework in FC. In order to help the fog administrators to allocate the resources efficiently through QCSO framework, queuing theory is employed to sort the tasks according to their priority. To optimize the resource selection cuckoo search algorithm plays an important role. Four stages of the suggested framework are: Evaluation of the task matric values, Entropy analysis, Prioritized entropy sorting, and Entropy-driven QCSO framework, illustrated in Figure 1. A thorough explanation of these stages is given in the following subsections.

Table 1: Review of Related Work

Authors/Citation	Approach	Algorithm Adopted	Key Aim	Outcomes Achieved
Mani SK et al., [20]	Genetic algorithm approach	iFogSim simulation in fog infrastructure	Strengthening QoS-aware scheduling	Enhancing latency, bandwidth while decreasing the energy usage
Li L. et al., [22]	Grey Wolf optimization technique	CloudSim and cloud-fog environment	Optimize task scheduling	Scale down energy consumption and delay
Singh P et al., [31]	Bio-inspired hybrid optimization	MPSO and MCSO	Handle resources and plan tasks for fog devices	Shorter response time, optimize resource management, and superior performance relative to other algorithms
Wadhwa and Aron [33]	Expectation Maximization (EM)	Expectation Maximization (EM) algorithm	Jobs are assigned and scheduled in fog network	Decrease in the time required for resource allocation, lower energy usage,

				advancement in job allocation
Abd Elaziz et al., [35]	Artificial Ecosystem-based Optimization (AEO)	Salp Swarm Algorithm (SSA)	The task scheduling process is optimized	With respect to the time rate and productivity, decreases the response time, greater efficacy in comparison to other heuristic techniques
Jamil et al., [38]	Streamlined job scheduling	Shortest Job First (SJF) algorithm	Boosted the response time for internet-reliant applications, cut down the energy usage, and optimize bandwidth usage	Compare to the other algorithms, the average response time and energy utilization are enhanced by 32% and 16% respectively
Yin et al., [27]	Augmented ant colony optimization (ACO)	Ant Colony Optimization (ACO) algorithm	Refine task performance to reduce the overall cost	A reduction in both total and economic costs and time needed for completion has led to enhance the overall QoS
Zhou et al., [42]	Pricing driven resource allocation for vehicular fog computing	Pricing oriented stable algorithms	Streamlined Resource efficiency	Significant outcomes in resource management and comparable results in other efficient search methods with lower complexity
Member and Luo [28]	Container oriented task scheduling	Container-based task-scheduling algorithm	Shorten execution time and refined task distribution	There was a 5% improvement in task capacity and time required for execution was cut by 10%
Zhang and Li [48]	Privacy centric allocation of resources	Contributory public key searchable encryption algorithm	Strengthen the robustness and security of fog node infrastructure	Resolving the security concerns and maintaining the user privacy

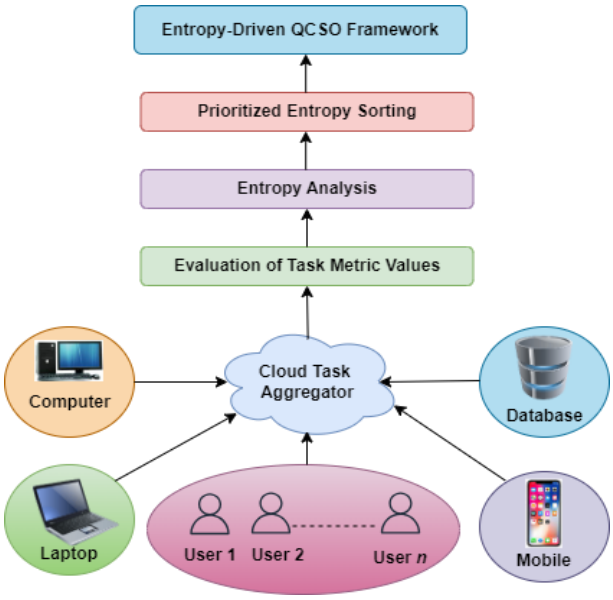

Figure 1: Architectural Design of the QCSO Framework

3.1Evaluation of Task Matric Values

In fog computing, task scheduling and resource allocation optimization is required for real time use-cases. The effectiveness of the resource allocation is directly influenced by the capabilities of fog computing. To prevent this, it is advisable to calculate the task measure values like task loss, delay, utilization, and reliability, before performing the optimization of QCSO framework.

3.1.1 Task Loss

Here, considering the values of R_A and T_P, where,

$R_A = \{R_1, R_2, R_3, \ldots\ldots\ldots\ldots R_m\}$ (1)

$T_P = \{T_1, T_2, T_3, \ldots\ldots\ldots\ldots\ldots T_m\}$ (2)

Here, both Equations (1) and (2) are the finite set. In context to the origin of the problem, R_A represents the resources and T_P represents the tasks.

Let Z represents the assignment of the tasks to resources within a set of all possible sequential assignments, where each task is assigned exactly one resource. The elements $z \in Z$ are represented as NxM matrix, following the First in First Out (FIFO) principle. The order of tasks for resource R_A is listed in column 'i' of this matrix, where $i = 1,2,3,\ldots\ldots,n$. According to Equation (3), resource R_1 will carry out tasks in the order of T_1, T_2, T_3, while resource R_2 will carry out tasks in the order of T_2, T_3, T_1.

$$z = \begin{bmatrix} 1 & 2 \\ 2 & 3 \\ 3 & 1 \end{bmatrix} \quad (3)$$

As demonstrated in Equation (4), assuming that the resource allocation process involves a cost function (C^{fun}),

$C^{fun}: Z \rightarrow (0,\infty)$ (4)

The relation between the cost function and the total processing time is shown in Equation (4). The values are varying from 0 to ∞ ranges. Equation (5) indicates that a NxM matrix is the cost function matrix.

C_{ij} : N x M → (0,∞) (5)

The objective is to assign job 'z', where $z \in Z$, to the resources R_A such that the cost function $C^{fun}(z)$ is minimized.

Here, the total number of tasks available before the scheduling process is denoted by $T_{(n)}^{tot}$ and the total scheduled tasks are denoted by $T_{(m)}^{tot}$. Task loss (T^{Loss}) is calculated by using the Equation (6).

$$T^{Loss} = \frac{T_{(n)}^{tot} - T_{(m)}^{tot}}{T_{(n)}^{tot}} \times C^{fun}(z) \quad (6)$$

3.1.2 Delay

The length of a task is proportional to task delay. The surplus time needed for task completion is represented by $P(s)$ and time needed to communicate the task within the given time window is represented by $Q(s)$. The formula to calculate the *Delay Time* (T^{delay}) is shown in Equation (7).

$$\textit{Delay Time } (T^{delay}) = P(s) + Q(s) \quad (7)$$

Here, the formulas shown in Equation (8) and Equation (9) are used to calculate *Transfer Time* (T^{Time}) and *BANDWIDTH* respectively.

$$\textit{Transfer Time } (T^{Time}) = \frac{T^{delay}}{BANDWIDTH} \quad (8)$$

$$BANDWIDTH = \frac{bandwidth}{N^{request}} \quad (9)$$

where, number of requests are represented by $N^{request}$ and available bandwidths are represented by *BANDWIDTH*.

3.1.3 Utilization

Here, $(TC)^{time}$ represents the total completion time, *VMs* refer to the current virtual machines, T^{Loss} indicates the loss of a task, and *T* represents the task. The utilization formula is shown in Equation (10).

$$Utilization = \frac{\Sigma_{JVM} T^{Loss}\, T}{(TC)^{time} + Number\ of\ VMs} \quad (10)$$

3.1.4 Task Reliability

The reliability of the task is determined using a task reliability coefficient $(TR)^{coff}$, which calculates the correlation between the two groups values. Here, the number of existing tasks is represented by *N*, total variance is denoted by $S(v)$, and sum of the individual variances is denoted by $T(v)$. In this instance, Equation (11) can be used to express the reliability coefficient.

$$(TR)^{coff} = \frac{N}{(N-1)} \left(\frac{S(v) - T(v)}{S(v)} \right) \quad (11)$$

Here, the interval of time is represented by *f*. The formulas shown in Equation (12) and Equation (13) are used to calculate *Reliability* and *Failure Time* respectively.

$$Reliability = 1 - T(f) \quad (12)$$

$$Failure\ Time\ T(f) = 1 - e^{-wt} \quad (13)$$

3.2 Evaluation of Entropy

The term probability distribution used when entropy is an estimate of the expected quantity of information or uncertainty. It is also described as the degree of uncertainty surrounding a split or the disorder in a system. Here, a task consisting of a set of inputs is denoted by T_i and P_k denotes the probability of that task. Assume that n tasks, designated as $T_1, T_2, T_3,.....,T_n$ are present and that each task has probabilities, $P_1, P_2, P_3,.....,P_n$ that add up to 1. Because tasks with lower probability are less predictable, their occurrence produces more information. Information computation is a decreasing function of P_k, symbolized by entropy. As it ranges from 0 to 1, the value of P_k decreases from ∞ to 0. This function demonstrates a reduced capacity for task selection. The entropy, denoted as E, is calculated using the Equation (14) by multiplying each task value by its corresponding probability.

$$E = -\sum_{i=1}^{n} pi \log_2 pi + T^{delay} + Utilization + (TR)^{coff} + T_{ij} \qquad (14)$$

All the values in the above equations have been calculated. T^{delay} represents delay time, *Utilization* represents utilization time, $(TR)^{coff}$ for task reliability, and T_{ij} for trust value. In this case, if e ≥ 0, then $pilog_2pi \geq 0$, that indicates that pi lies between $0 \leq pi \leq 1$, where e = 0 if one $pi = 1$ and all other $pi = 0$.

3.3 Prioritized Entropy Sorting

In contrast to an ordinary queue, a priority queue does not follow the principle of First Come First Serve (FCFS). Let's consider the entropy to be represented by the set $\{E_1, E_2, E_3,…....,E_n\}$ for the tasks $\{T_1, T_2, T_3,.....,T_n\}$. Using queuing techniques and prioritized based on the maximum entropy value the tasks are organised. The tasks are arranged in arrival time sequence, and with highest entropy according to queuing theory which maximize the entropy. Entropy values in algorithms 1 and 2 are prioritized according to queuing theory, with the highest values being ranked first.

Algorithm 1: Resource Allocation
Input: Request by User, Resource allocation
Output: Priority based allocation of resource
Start
if (R_i = = *Valid*) then
if (R_i Ɛ R) then
    ***call* Manage Priority**
USER$_i$*gets* R_i
Status (R_i) = allocated
else
Request is not approved
else
User Request is invalid
End if
End if
End
Algorithm 2: Manage Priority
Input: Resource Priority
Output: Set Time
Start
if (Priority of Resource = = High)
Set time = low; /* Resources with high priority */

else
Set time = normal; /* Resources with low priority */
End if
End

3.4 Entropy-Driven QCSO Framework

The cuckoo search (CS) algorithm uses entropy measurements driven from the queuing theory as input. Task allocation for the queuing theory driven optimization approach is considered as a result that is in accordance with the precedence of these entropy measures. To check the information regarding the accessibility of the resources, the algorithm gets that information from resource allocation. After the allocation process, it checks the information regarding the accessibility of the resources. The entropy value is updated through the CS optimization algorithm which is useful for scheduling the resources in a better way.

1. Initialization: A random population Y_i is assigned where Y_i = (3, 2, 1, ……., n) which is shown in Equation (15) of n host nests.
2. Behaviour of Levy Flight: It acquires a cuckoo through a Levy flight as explained in Equation (16).

$$Y_i\,(u+1) = Y_i\,(u) + \infty + \text{Levy}\,(\lambda), \quad a > 0 \qquad (15)$$

$$\text{Levy}\,(\lambda) = u\,(-\lambda), \quad 1 < \lambda < 3 \qquad (16)$$

3.4.1 Fitness Calculation

Using the fitness function (F_i), fitness is measured to achieve optimal solution through the formula shown in Equation (17). Select a random nest called j. Once the fitness function is calculated, the randomly selected nest j is replaced with the new solution. If current fitness reading of nest (F_i) is below fitness value of a selected random nest (F_j) (i.e. $F_i < F_j$) then, a new solution replaces the randomly chosen nest j. In order to attain the most favourable outcome, the fitness functions are computed as

$$F_i = (best)^{current} - (best)^{previous} \qquad (17)$$

where, $(best)^{current}$ represents the current best solution at that moment, and $(best)^{previous}$ represents the previous best solution for choosing resources allocation. After the data is accessible, the system examines the workload for each task if it requires a significant amount of CPU or memory. The utilization of cuckoo search algorithms facilitates the identification of the most optimal resources. Subsequently, the obtained optimal resources are utilized to compute the quality score that is shown in algorithm 3.

3.4.2 Working of Queue-Cuckoo Search Optimization Framework (QCSOF)

Algorithm 3: Queue-Cuckoo Search Optimization Framework (QCSOF)
Input: Performance Function $f(x)$, ($x_1, x_2, x_3, \ldots\ldots\ldots, xq$)
Output: Optimal Resource
Start
Initial population of m host is generated
Y_i = (1, 2, 3,……n)
While (u_i < Maximum Generation)
Random Cuckoo Search with Lévy Flights
Compute F_z = quality
Select a random nest from m nests

if ($f_x < f_y$) then
replace *y* with a new solution
End if
Discard the faulty nest
Identify the optimal nest
End while
Return Optimal Resource

Here, energy consumption, resource allocation time, and resource availability are the various QoS parameters that are taken into consideration. Equation (18) is used to calculate the quality score (*QS*) for the resources that are matched.

$$QS(i) = (QS_i)^{time} + (QS_i)^{energy} + (QS_i)^{availability} \quad (18)$$

where, $i = 1, 2,\ldots..,n$. *QS* represents the Quality Score, $QS(i)$ indicates the quality score assigned to the i^{th} resource. The best possible quality score would be achieved by utilizing Equation (19).

$$(QS)^{maximum} = \max (QS(i)) \quad (19)$$

where $i = 1, 2, 3\ldots\ldots n$

The peak score labelled as $(QS)^{maximum}$ signifies the superior quality across the resources. The quality found with the highest score is chosen after evaluating the factors like resource availability, quick response times, and efficient energy use.

4. Experimental Findings and Analysis

The suggested framework is simulated using the CloudAnalyst simulator, which is the top most simulator of CloudSim. A 50-node heterogeneous physical data centre is established. Every node has an 8-GB RAM, 1 TB of storage, and the CPU is capable of processing more than 2000 million instructions per second. When a request is submitted by the user, the processing is begun and the resource's capacity is checked and simulated within the data centre. This can serve as a framework for randomly adjusting CPU utilization based on specific requirements. The trial is repeated 10 times and the results are analysed. For the simulation process, a virtual server XEN is used. The arrival rate of 1000 requests per hour during peak hours and 50 requests per hour during non-peak hours are assumed in this simulation. The execution time of every request is 1 hour. The simulator manages the traffic and generates a set of requests on a regular interval of time. As the user requests increase, the resource selection becomes more significant to enhance the quality score. The quantity of matching resources varies across all the iterations. A quality of a resource is calculated by analysing its allocation time, energy utilization, and availability. This query system enables the selection of the most suitable resource that can deliver best performance.

The outcome of the framework is showing the direct relation between count of user requests and the time needed for allocate resources. Resource assignment is entirely dependent on their availability and the distance between their location and the client's location. Figure 2 shows the execution range from 6, 20, 25, 31, and 34 seconds of user requests of 50, 100, 150, 200, and 250.

The total number of requests assigned to the corresponding resources that were matched is shown in Figure 3. There is a direct relation between energy consumption and the increase in the number of requests. In fog computing, which is helpful in resource distribution, a specific amount of energy is required. When a resource is examined, it is important for it to be engaged and actively participating in the task. Figure 4 clearly illustrates a direct relation between the increase in resource requests and total time required for resource allocation.

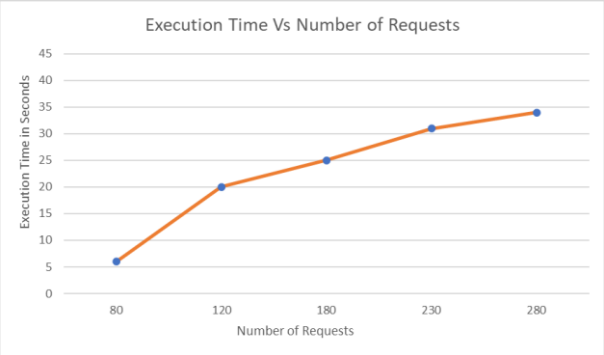

Figure 2: Execution Time in Relation to Number of Requests

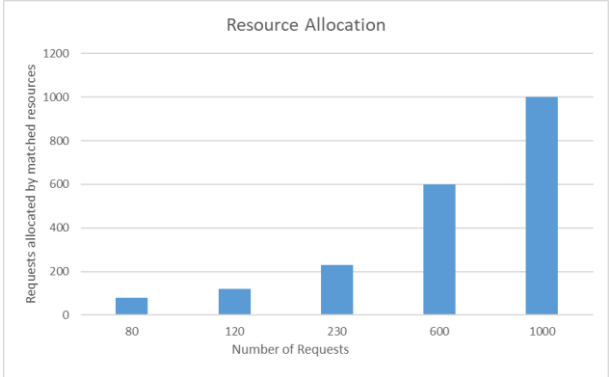

Figure 3: Matched Resources in Relation to Number of Requests

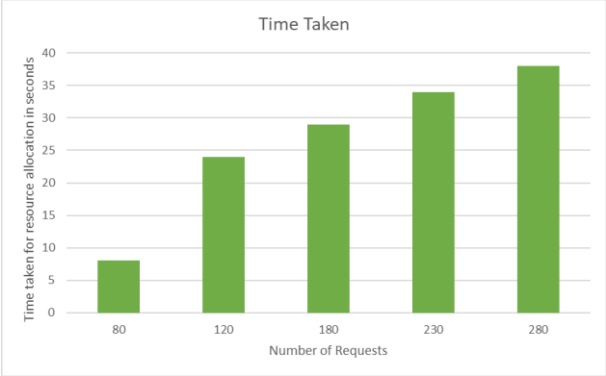

Figure 4: Time Taken in Relation to Number of Requests

Table 2: A Comparative Analysis of Energy Utilization with various scheduling techniques

Request Count	**150**	**250**	**350**	**450**	**550**
FCFS	22	28	34	38	40
Delay Priority	20	26	31	35	37
QoS Aware	19	25	30	34	35
EBKHRA	18	23	26	31	33
QCSOF	16	18	20	23	26

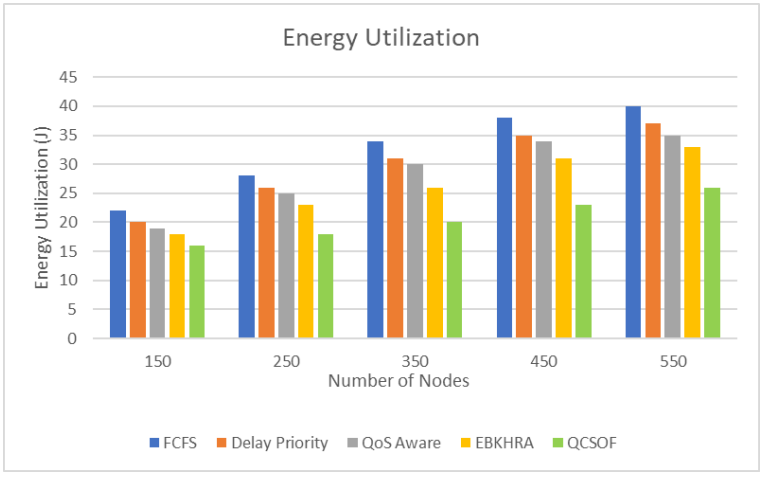

Figure 6: Energy Utilization in joules for different scheduling techniques

Based on the simulation experiments, the results demonstrates that the energy is conserved through the proposed QCSO framework by an overall average of 5.55%, 4.93%, 4.47%, and 1.79% in comparison to the existing scheduling techniques like FCFS (First Come First Serve), Delay Priority, QoS Aware and EBKHRA respectively [11, 51]. For the large data centres in the fog environment, the proposed method is very efficient, successful and potentially work that is shown in the outcome. The analysis of energy utilization between various scheduling techniques and QCSO framework is depicted in Table 2 and represented in Figure 6.

5. Significant Contribution of the Proposed Framework

To assess the effectiveness of the proposed work, a comparison between QCSO framework and various scheduling techniques is conducted.

- QCSO framework efficiently allocates resources by utilizing a priority-based approach.
- Energy is conserved using priority-based QCSO framework.
- Overall duration of time is reduced for allocating the resources to the tasks using the QCSO framework.

6.Conclusion and Future Directions

In order to allocate resources effectively and dynamically, the QCSO framework is built in a real-time fog environment. Based on a comprehensive survey, it is determined that QoS parameters require enhancement when allocating resources. To achieve the QoS metrics like efficient energy usage, response time, and resource allocation, different simulations are run in the proposed framework. The experimental findings demonstrate that even with a large number of user requests, the proposed framework efficiently distributes resources dynamically. Vital QoS parameters like response time and energy usage are improved by the QCSO framework. However, the major challenges are efficient resource allocation to edge nodes and large-scale application deployment in the fog environment. To further optimize resource allocation, the incorporation of a Markovian model into a fog node can enhance this work.

References

[1] Singh S, Chana I. A survey on resource scheduling in cloud computing: Issues and challenges. *J grid Comput*. 2016;14:217-264.
[2] Maddikunta PKR, Gadekallu TR, Kaluri R, Srivastava G, Parizi RM, Khan MS. Green communication in IoT networks using a hybrid optimization algorithm. *Comput Commun*. 2020;159:97-107.
[3] Alazab M, Khan S, Krishnan SSR, Pham QV, Reddy MPK, Gadekallu TR. A multidirectional LSTM model for predicting the stability of a smart grid. *IEEE Access*. 2020;8:85454-85463.
[4] Numan M, Subhan F, Khan WZ, et al. A systematic review on clone node detection in static wireless sensor networks. *IEEE Access*. 2020;8:65450-65461.
[5] Wolke A, Bichler M, Chirigati F, Steeves V. Reproducible experiments on dynamic resource allocation in cloud data centers. *Inf Syst*. 2016;59:98-101.
[6] Naha RK, Garg S, Georgakopoulos D, et al. Fog computing: Survey of trends, architectures, requirements, and research directions. *IEEE Access*. 2018;6:47980-48009.
[7] Yousefpour A, Fung C, Nguyen T, et al. All one needs to know about fog computing and related edge computing paradigms: A complete survey. *J Syst Archit*. 2019;98:289-330.
[8] Moura J, Hutchison D. Fog computing systems: State of the art, research issues and future trends, with a focus on resilience. *J Netw Comput Appl*. 2020;169:102784-102798.
[9] Murtaza F, Akhunzada A, Boudjadar J. QoS-aware service provisioning in fog computing. *J Netw Comput Appl*. 2020;165:102674-102689.
[10] Qu Z, Wang Y, Sun L, Peng D, Li Z. Study QoS optimization and energy saving techniques in cloud, fog, edge, and IoT. *Complexity*. 2020;2020(1):8964165-8964181.
[11] RM SP, Bhattacharya S, Maddikunta PKR, et al. Load balancing of energy cloud using wind driven and firefly algorithms in internet of everything. *J Parallel Distrib Comput*. 2020;142:16-26.

[12] RM SP, Maddikunta PKR, Parimala M, et al. An effective feature engineering for DNN using hybrid PCA-GWO for intrusion detection in IoMT architecture. *Comput Commun*. 2020;160:139-149.
[13] Mustafa S, Bilal K, Malik SUR, Madani SA. SLA-aware energy efficient resource management for cloud environments. *IEEE Access*. 2018;6:15004-15020.
[14] Xiao Y, Krunz M. Dynamic network slicing for scalable fog computing systems with energy harvesting. *IEEE J Sel Areas Commun*. 2018;36(12):2640-2654.
[15] Reddy T, RM SP, Parimala M, et al. A deep neural networks based model for uninterrupted marine environment monitoring. *Comput Commun*. 2020;157:64-75.
[16] Bhattacharya S, Maddikunta PKR, Kaluri R, et al. A novel PCA-firefly based XGBoost classification model for intrusion detection in networks using GPU. *Electronics*. 2020;9(2):219-234.
[17] Shone R, Glazebrook K, Zografos KG. Resource allocation in congested queueing systems with time-varying demand: An application to airport operations. *Eur J Oper Res*. 2019;276(2):566-581.
[18] Rani M, Guleria K, Panda SN. Enhancing latency performance in fog computing through intelligent resource allocation and Cuckoo search optimization. In: *Applied Data Science and Smart Systems*;CRC Press:256-263.
[19] Akintoye SB, Bagula A. Improving quality-of-service in cloud/fog computing through efficient resource allocation. *Sensors*. 2019;19(6):1267-1285.
[20] Mani SK, Meenakshisundaram I. Improving quality-of-service in fog computing through efficient resource allocation. *Comput Intell*. 2020;36(4):1527-1547.
[21] Ul Islam MS, Kumar A, Hu YC. Context-aware scheduling in Fog computing: A survey, taxonomy, challenges and future directions. *J Netw Comput Appl*.; 2021; 6(2):103008-103023.
[22] Li L, Guan Q, Jin L, Guo M. Resource allocation and task offloading for heterogeneous real-time tasks with uncertain duration time in a fog queueing system. *IEEE Access*. 2019;7:9912-9925.
[23] Battula SK, Garg S, Montgomery J, Kang B. An efficient resource monitoring service for fog computing environments. *IEEE Trans Serv Comput*. 2020;13(4):709-722.
[24] Swain C, Sahoo MN, Satpathy A, et al. METO: Matching-theory-based efficient task offloading in IoT-fog interconnection networks. *IEEE Internet Things J*. 2020;8(16):12705-12715.
[25] Huang X, Fan W, Chen Q, Zhang J. Energy-efficient resource allocation in fog computing networks with the candidate mechanism. *IEEE Internet Things J*. 2020;7(9):8502-8512.
[26] Seth I, Guleria K, Panda SN. Introducing intelligence in vehicular ad hoc networks using machine learning algorithms. *ECS Trans*. 2022;107(1):8395-8412.
[27] Yin C, Li T, Qu X, Yuan S. An improved ant colony optimization job scheduling algorithm in fog computing. In: *International Symposium on Artificial Intelligence and Robotics*. 2020:132-141.
[28] Member S, Luo J. Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing. *IEEE Trans Ind Informatics*. 2018;14(10):4712-4721.
[29] Ahmed-Nacer M, Suri K, Sellami M, Gaaloul W. Simulation of configurable resource allocation for cloud-based business processes. In: *IEEE International Conference on Services Computing (SCC)*. ; 2017:305-313.
[30] Kaur PD, Chana I. Cloud based intelligent system for delivering health care as a service. *Compute methods programs biomed*. 2014;113(1):346-359.
[31] Singh P, Singh R. Energy-efficient delay-aware task offloading in fog-cloud computing system for IoT sensor applications. *J Netw Syst Manag*. 2022;30(1):1-25.
[32] Singh A, Juneja D, Malhotra M. A novel agent based autonomous and service composition framework for cost optimization of resource provisioning in cloud computing. *J King Saud Univ Inf Sci*. 2017;29(1):19-28.
[33] Wadhwa H, Aron R. TRAM: Technique for resource allocation and management in fog computing environment. *J Supercomput*. 2022;78(1):667-690.
[34] Khattar N, Sidhu J, Singh J. Toward energy-efficient cloud computing: A survey of dynamic power management and heuristics-based optimization techniques. *J Supercomput*. 2019;175-193.
[35] Abd Elaziz M, Abualigah L, Attiya I. Advanced optimization technique for scheduling IoT tasks in cloud-fog computing environments. *Futur Gener Comput Syst*. 2021;124:142-154.
[36] Rani M, Guleria K, Panda SN. Unleashing the power of QoS: A comprehensive study and evaluation of services-based scheduling techniques for fog computing. *Int J Intell Syst Appl Eng*. 2023;12(4s):388–405.
[37] Seth I, Panda SN, Guleria K. The essence of smart computing: Internet of things, architecture, protocols, and challenges. In: *9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*. ; 2021:1-6.
[38] Jamil B, Shojafar M, Ahmed I, Ullah A, Munir K, Ijaz H. A job scheduling algorithm for delay and performance optimization in fog computing. *Concurr Comput Pract Exp*. 2020;32(7):5581-5598.
[39] Espadas J, Molina A, Jiménez G, Molina M, Ramirez R, Concha D. A tenant-based resource allocation model for scaling software-as-a-service applications over cloud computing infrastructures. *Futur Gener Comput Syst*. 2013;29(1):273-286.

[40] Wang WYC, Rashid A, Chuang HM. Toward the trend of cloud computing. *J Electron Commer Res*. 2011;12(4):238-253.
[41] Karthick A V, Ramaraj E, Subramanian RG. An efficient multi queue job scheduling for cloud computing. In: *World Congress on Computing and Communication Technologies*. ; 2014:164-171.
[42] Zhou Z, Member S, Liu P, Feng J, Zhang Y, Member S. Computation resource allocation and task assignment optimization in vehicular fog computing : A contract-matching approach. 2019;68(4):3113-3125.
[43] Singh P, Walia NK. A review: Cloud computing using various task scheduling algorithms. *Int J Comput Appl*. 2016;142(7):30-43.
[44] Rani M, Guleria K, Panda SN. Blockchain technology novel prospective for cloud security. In: *10th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)*; 2022:1-6.
[45] Tripathy AK, Patra MR, Khan MA, Fatima H, Swain P. Dynamic web service composition with QoS clustering. In: *2014 IEEE International Conference on Web Services*. ; 2014:678-685.
[46] Shimpy E, Sidhu MJ. Different scheduling algorithms in different cloud environment. *Int J Adv Res Comput Commun Eng*. 2014;3(9):8003-8016.
[47] Patel M, Kadian R. A review on ACO based scheduling algorithm in cloud computing. *Int J Comput Sci Mob Comput*. 2016;5(5):489-504.
[48] Zhang L, Li J. Enabling robust and privacy-preserving resource allocation in fog computing. *IEEE Access*. 2018;6:50384-50397.
[49] Alkhanak EN, Lee SP, Khan SUR. Cost-aware challenges for workflow scheduling approaches in cloud computing environments: Taxonomy and opportunities. *Futur Gener Comput Syst*. 2015;50:3-21.
[50] Gadekallu TR, Khare N. Cuckoo search optimized reduction and fuzzy logic classifier for heart disease and diabetes prediction. *Int J Fuzzy Syst Appl*. 2017;6(2):25-42.
[51] Cardellini V, Grassi V, Presti FL, Nardelli M. On qos-aware scheduling of data stream applications over fogcomputing infrastructures. Proceedings of the 2015 IEEE Symposium on Computers and Communication (ISCC); 2015;271-276.