Computers, Materials & Continua
DOI:10.32604/cmc.2022.015707
Article

Tech Science Press

Efficient Resource Allocation in Fog Computing Using QTCS Model

M. Iyapparaja[1], Naif Khalaf Alshammari[2,*], M. Sathish Kumar[1], S. Siva Rama Krishnan[1] and Chiranji Lal Chowdhary[1]

[1]School of Information Technology and Engineering, Vellore Institute of Technology, Vellore, 632014, India
[2]Mechanical Engineering, University of Hail, Saudi Arabia
[*]Corresponding Author: Naif Khalaf Alshammari. Email: naif.alshammari@uoh.edu.sa
Received: 03 December 2020; Accepted: 09 January 2021

Abstract: Infrastructure of fog is a complex system due to the large number of heterogeneous resources that need to be shared. The embedded devices deployed with the Internet of Things (IoT) technology have increased since the past few years, and these devices generate huge amount of data. The devices in IoT can be remotely connected and might be placed in different locations which add to the network delay. Real time applications require high bandwidth with reduced latency to ensure Quality of Service (QoS). To achieve this, fog computing plays a vital role in processing the request locally with the nearest available resources by reduced latency. One of the major issues to focus on in a fog service is managing and allocating resources. Queuing theory is one of the most popular mechanisms for task allocation. In this work, an efficient model is designed to improve QoS with the efficacy of resource allocation based on a Queuing Theory based Cuckoo Search (QTCS) model which will optimize the overall resource management process.

Keywords: Queuing theory; Cuckoo search; QoS; resource allocation; energy efficiency

1 Introduction

The fog computing is a modern yardstick for the vibrant provision of the amenities of calculating recent data cores which naturally exercises the technology of virtual machine (VM) amalgamation and environment severance [1]. This type of computing is used because cloud computing is not feasible for many Internet of Things (IoT) applications [2]. The distributed approach of fog computing addresses IoT and industrial IoT (IIoT) needs, as well as the enormous amount of data produced by smart sensors and the IoT devices, which otherwise would be expensive and time-consuming for processing and analyzing in the cloud [3,4]. Fog computing decreases the amount of data sent to the cloud, thereby reducing the use of the bandwidth and its associated costs. Recently, allocating the resources in the virtualized data cores have reached substantial consideration, which hold notable influence on the expertise of energy in the data core [5]. Fog computing has altered the manner in which people utilize the IT resources at present. Now, as a replacement of acquiring their precise IT resources, they utilize the facilities proffered by a sensible

price reliant on a "pay-per-use" model [6]. Fog computing extends pay per employ services of IaaS, PaaS and SaaS through diverse full-service provider (FSP) [7].

The basic difference between the fog computing and classical approach is that fog uses every service on the web whereas classical approach uses fixed infrastructure. In the classical approach, organizations first invest certain amount on hardware and software for providing services based on their customer requirements [8]. However, it is a challenging task to predict the customers' service or dynamic workload in a short period of time. The organizations may discourage potential clients because of the low QoS [9]. Various challenges like scaling, security, competent load balancing, data-center energy consumption, service availability, data lock-in, resource scheduling, and QoS management are encountered during fog computing deployment. Efficient fog resource management can offer various benefits namely, energy efficiency, network load reduction, profit maximization, load balancing, and minimization of Service Level Agreement (SLA) violations. The scheduling approaches reduce SLA violation risks and optimize revenues in resource allocation [10]. It can lead to Green Fog Computing by providing workload consolidation and reduction in carbon emissions [11,12]. Adaptive agent-based SLA monitoring model assures the customers of maximum optimized QoS requirements [13].

Fog computing has a greater role in controlling the flow of data from large and complex networks. To boost the overall efficiency and minimize energy consumption, dynamic and efficient load balancing mechanisms can be implemented. Using load balance technology, loads can be transferred or exchanged between computer nodes. The key emphasis has therefore been the design of energy-efficient load balancing solutions for edge and fog environments [14–16]. Resources are collections of physical or virtual components within a computer structure of limited availability. Any connected device, and any internal system components are regarded as a resource. Resource allocation in fog computing varies from the traditional to distributed computing environment due to the presence of various QoS metrics such as CPU memory, speed, and stability. In dealing with resource allocation, multiple requests will be in the queue, waiting to be served at various stages. To solve this type of situation, queuing methods are therefore required. A mathematical formulation to demonstrate how a system's throughput and time delay will differ between a single server system and a multiple server system in a fog computing environment using queuing system technique is the need of the hour [17]. Also, an optimal resource allocation is required in highly congested queueing systems.

1.1 Motivation

Major challenge in the cloud computing are the network delay and the resource allocation. Fog computing solves these challenges by deploying the load balancing mechanism concentrated on edge nodes. This has motivated us to implement a novel method using entropy based queuing theory optimization approach for addressing specific QoS metrics such as resource availability, allocation time, and energy consumption.

1.2 Contributions

The resources are allocated to the applications that run on the virtual environment. The order and time play major role in the efficient allocation of resources. The contributions of this work are as follows:

- The proposed queuing theory model helps in managing the resources based on priority or requirement of the fog computing services.

- The proposed model improvises QoS metrics of fog computing such as resources classification, allocation, and energy consumption.
- The suggested model provides the distribution of resources in a uniform manner for the users in the environment.

Our primary motivation is to efficiently manage resource allocation operations in fog environment. Hence, we propose a QTCS model in fog computing which efficiently manages resource allocation with respect to the task schedule. This would improve the QoS metrics such as efficiency of the fog services and also reduce the overall cost.

1.3 Organization of the Paper

The rest of the paper is organized as follows: Section 2 presents a survey on existing works. Section 3 discusses the proposed work and addresses open issues in resource allocation based on QTCS model. Section 4 includes experimental results and discussions. Section 5 describes the significance of our proposed model. The paper concludes in Section 6 with future directions.

2 Literature Survey

In this section, the summary of existing works on QoS and resource allocation in fog computing are discussed. Assigning the virtual resources to physical resources in a cloud environment is a major challenge as the QoS needs to be effectively managed. Akintoye, Samson Busuyi et. al. [18] analyzed the task assignment and virtual machine deployment issues for a fog computing scenario. The authors proposed an improvised task assigning method and a genetic algorithm for efficiently implementing the virtual machines. The QoS metric considered in this work is the assignment cost which has been improved significantly by implementing the proposed method [19].

The fog computing technology through its infrastructure-as-a-service has improvised QoS for resource management. Li et al. [20] implemented a fog queuing system with a constrained infrastructure for deploying real time applications with variations in the categories of the task and run time. In order to achieve this, a parallel virtual queue method is deployed to store every category of task in specific queue. The resource allocation is balanced and the task completion ratio is improvised by implementing the parallel algorithms such as offloading and buffering.

The emergence of IoT in all spheres of life has increased the data being generated at a fast rate. The processing delay and scalability issues restrict this data to be computed in the cloud. The scheduling mechanisms in fog computing service take care of resource allocation. Battula et al. [21] implemented a novel technique for efficient resource utilization in fog computing namely Support and Confidence (SC) technique. The proposed technique was deployed with real time traffic and it is inferred that resource consumption is lesser than the traditional methods. Majority of the cloud services are deployed over the Internet and are thus prone to network delay. Swain et al. [22] implemented a fog computing architecture to execute real time IoT tasks at the edge nodes. The devices in the IoT network can increase their battery life by distributing the tasks to nearby fog nodes. The architecture also implements a novel method called METO, which is an effective strategy for distributing the tasks in the IoT fog network. This lessens the battery consumed and task failure rate. Fog computing nodes use edge network deployment for efficient communication, especially in an IoT scenario. Huang et al. [23] discussed the energy-efficient resource allocation issue in the fog computing networks with respect to the fog nodes and implemented an efficient method for load balancing keeping in mind the network resource limitations. Fog computing technology is used for deployment of real time applications in IoT.

Suri et al. [24] introduced the resource provisioning scheme by considering acceptable intensity levels. The existing problems are rectified through Virtual Machine-Physical Mapping. The authors achieved up to 32% of power consumption under the host of 1000 VMs that automatically turn off the unnecessary servers in the fog environment. This proposed approach compact the VMs into a single entity by correlating resource requirements and the results achieved are better when compared to the existing applications. An approach for extending the capacity of fog resource provisioning to local clusters by considering scheduling, resource brokering, and dispatching the sequences in the fog is proposed in this work [25].

Task scheduling expresses a central role in fog computing environments. Subsequently cost of every single task in the fog resources is diverse amongst themselves. Scheduling of client tasks in the fog is not similar considering conventional scheduling methods. Tasks scheduling is not done under single criteria but under several rules and regulations, which are in the contract between users and FSPs. And also providing good QoS to the users in conferring to the agreement is a decisive task [26].

Resource allocation comprises of three functions: resource mapping [27], resources execution, and resources monitoring [28,29]. Scheduling mechanism is extremely necessary to enrich the server and usage of resources to boost the functioning of servers [30–32]. When the fog resources are requested by numerous tasks, a proficient task scheduling procedure is specifically significant [33]. Workflow scheduling in fog computing assigns every task to a pertinent fog service by gathering the implementation of several resources to attain suitable QoS [34]. The successive phases of a cuckoo basically form a random walk procedure which pursues a control law step length allotment with a hefty tail [35]. In the recent trend, Green Fog (GC) computing approach is used to improve the resources utilization. It reduces energy consumption and thereby improves power efficiency [36]. The summary of some of the significant works are listed in Tab. 1.

Table 1: Summary of existing works

Ref. No.	Methods	Evaluation metrics	Research gaps
[11]	Task allocation using genetic algorithm	QoS—Allocation cost	Network delay in cloud services
[14]	Support and Confidence method for resource monitoring	Resource consumption	Energy consumption in case of IoT devices, deployment of VM
[15]	Matching theory based efficient task offloading strategy	Total system energy and number of outages	Priority task execution, task allocation
[21]	Clustering the jobs based on burst time	Reduced fragmentation and starvation	Processing delay and latency in cloud
[26]	Cost-aware workflow scheduling approaches	Performance and system functionality	Processing delay in cloud

3 Proposed Resource Allocation Model—QTCS

Tasks are collected as a task pool in the proposed framework. Those will be graded, and compute resources have been disseminated in fog computing (FC). The available resources are

allotted based on the relative weights of the tasks. The QTCS is employed in the allocation of resources in FC. The queuing theory is used to prioritize the task and helps fog admins to allocate the resources available optimally using QTCS. The cuckoo search is used to optimize the resource selection. As illustrated in Fig. 1, the proposed model comprises of four phases such as assessment of task measure values, entropy computation, priority wise entropy assortment, and entropy based QTCS optimization. These phases are discussed in detailed in the following sub-sections.

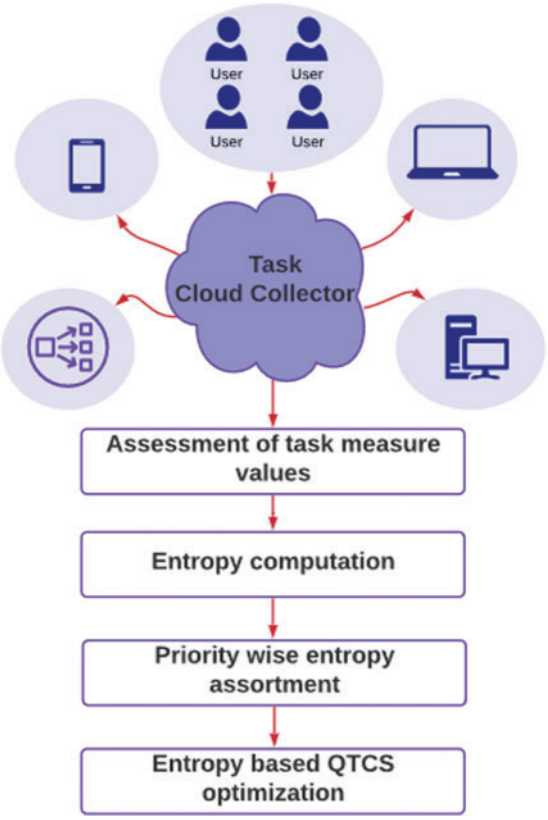

Figure 1: Architecture of the QTCS method

3.1 Assessment of Task Measure Values

Task programming and resource allocation optimization are vital, exigent and core components in fog application services and systems. The proficiency of the allocation openly effects the functioning of the entire FC services. To evade this situation, it is better to compute the task loss, delay, utilization, and reliability which are considered as the task measure values before applying the QTCS optimization.

Task Loss—Let us consider values R and T, where

$$R = \{R1, R2 \ldots Rm\} \tag{1}$$

$$T = \{T1, T2 \ldots Tm\} \tag{2}$$

The above Eqs. (1) and (2) are two finite sets. Grounded on the industrial genesis of the issue, 'R' is known as resource and 'T' denotes as task.

Let X denote the set of every sequential assignment of tasks to the resources, likewise every task is carried out by every resource precisely once; where the elements $x \in X$ are written as NxM matrix which follows First in First Out (FIFO), in which column 'i' (where i= 1,2,3,. . .,n) lists the tasks order of that resource R. Here, resource R1 will perform three tasks T1, T2, T3 in the sequence T2, T3, T1 whereas the resource R2 will perform in the sequence T1, T2, T3 (Eq. (3)).

$$X = \begin{vmatrix} 1 & 2 \\ 2 & 3 \\ 3 & 1 \end{vmatrix} \tag{3}$$

Let us assume there is some cost function(C) involved in resource allocation process which is depicted in Eq. (4),

$$C : X \rightarrow (0, \infty) \tag{4}$$

The Eq. (4) relates cost function to the total processing time where the value lies between 0 to ∞. The cost function matrix correlates to NxM which is given in Eq. (5),

$$C_{ij} : N \times M \rightarrow (0, \infty) \tag{5}$$

The task-loss is to accomplish an assignment of job 'x' where $x \in X$, to the resources R which means the cost function of 'x' i.e., $C(x)$ is minimum.

Let the total task available before scheduling be $T_{(n)}$ and the total tasks that are scheduled be $T_{(m)}$, then the task loss TL is computed in Eq. (6).

$$\text{Task Loss (TL)} = \frac{T_{(n)} - T_{(m)}}{T_{(n)}} \times C(x) \tag{6}$$

Delay—The task delay is proportionate to the task length. L(t) is extra time taken to finish the task and P(t) is the time taken to transfer the task within the allocated period. Then the formula is given in Eq. (7),

$$\text{Delay Time } D(t) = L(t) + P(t) \tag{7}$$

where, the transfer time is calculated by using the formula in Eq. (8) and bandwidth (BW) is calculated in Eq. (9),

$$\text{Transfer time} = \frac{D}{BW + user} \tag{8}$$

$$BW + user = \frac{BW}{N} \tag{9}$$

where, N signifies the number of requests and total BW represents available bandwidths.

Utilization—Let m(t) denote the task completion time, VM's denotes current virtual machine, TL signifies the task loss and 'T' denotes the task. Then the formula for utilization is given in Eq. (10),

$$\text{Utilization} = \frac{\sum_{jVM} TLT}{m(t) \times No\ of\ VM} \tag{10}$$

Reliability—The reliability of the task is computed by means of reliability coefficient that is used to calculate the correlation of two group values. Assume N as the number of existing tasks, S(v) as the individual variance's sum and V(t) as the total variance, then the expression for the reliability coefficient is expressed as (Eq. (11)),

$$Reliability\ coefficient\ (R) = \frac{N}{(N-1)}\left(\frac{\mathrm{V}(\mathrm{t}) - S(\mathrm{v})}{\mathrm{V}(\mathrm{t})}\right) \tag{11}$$

Let 't' be the time period interval then the reliability is calculated using the following formula in Eq. (12) and failure time F(t) is given in Eq. (13),

$$R = 1 - F(t) \tag{12}$$

$$F(t) = 1 - e^{-\omega t} \tag{13}$$

3.2 Entropy Computation

Entropy is a degree of the anticipated information matter or uncertainty of a probability distribution [37]. Moreover, it is defined as the grade of chaos in a system or the ambiguity regarding a partition. Let T_i stands for a task that is a set of inputs and let p_k be the probability of task T_i. Let there are n tasks such as T_1, ..., T_n with probabilities $p_1,\ldots p_n$ adding up to 1. As the incidence of tasks with lesser probability generates more information as they are less predicted, a computation of information 'E' which represents the entropy is a declining function of p_k. This decreases from infinity to 0, for p_k ranges from 0–1. This function indicates the ability of task selection is lower. From these n task values, the anticipated information content 'E' termed as entropy is given in Eq. (14) and is obtained by weighting the task values existing by means of their relevant probabilities.

$$\mathrm{E} = -\sum_{i=1}^{n} p_i \log_2 p_i + D(t) + U + R + T_{ij} \tag{14}$$

Let TL denotes the task loss, D(t) denotes the delay time, U refers the utilization time, R specifies the reliability moreover $\mathrm{T_{ij}}$ gives the trust value which are already calculated in the above equations. Here, $p_i \log_2 pi \geq 0$ describes p_i lies between $0 \leq p_i \leq 1$, if $\mathrm{e} \geq 0$, where $\mathrm{e} = 0$ if one of $p_i = 1$ and remaining $p_i = 0$.

3.3 Priority Wise Entropy Assortment

A priority queue is a deviation on the queue without FIFO, as it is not precisely evocative of the behavior of the data structure. The queuing theory follows many standard queuing structures; one among them is FIFO. Assume entropy as {E1, E2, and E3, ..., En} for the tasks {T1, T2, and T3, ..., Tn}. The tasks are arranged on the basis of queuing techniques and the maximum entropy value. As a result, with the queuing theory the tasks are arranged based upon the time arrival which classifies the equivalent entropy values based on the task with maximum of the entropy too. As depicted in Algorithms 1 and 2 the entropy values are prearranged, subject to the priority of queuing theory with the maximum values attained.

Algorithm 1: Resource allocation

Input: User request, Resources
Output: Allocate resource based on priority
Begin
if (R_i == valid) then
if (R_i ε R) then
call Manage Priority **//Algorithm 2**
$User_i$gets R_i
status (Ri) = allocated
else
Request is rejected
else
User Request is invalid
End if
End if
End

Algorithm 2: Manage priority

Input: Resources Priority
Output: Set Time
Begin
if (Resource priority == High)
set time = less; // for High_Priority resource
else
set time = normal; // for Less_Priority
End If
End

3.4 Entropy Based QTCS Optimization

The assorted entropy values based on the queuing theory is taken as input to the cuckoo search (CS) algorithm. Therefore, the intended queuing theory-based optimization technique tasks are allotted based on the precedence of entropy values. After this allocation, it checks if the information concerning the resources accessibility is present or not. The CS optimization algorithm updates the entropy values reliant on which the resources are scheduled in a proficient manner.

(a) Initialization—Assign a random population X_i (where $X_i = 3, 2, 1, \ldots, n$) of n host nests which is depicted in Eq. (15),

(b) Levy Flight Behavior- Acquire a cuckoo by Levy flight depicted in Eq. (16),

$$X_i(t+1) = X_i(t) + \alpha + \text{Levy}(\lambda), \quad a > 0 \tag{15}$$

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

3.4.1 Fitness Calculation

The fitness is calculated by means of the fitness function (Fi) in Eq. (17) to attain an optimum solution. Choose an arbitrary nest 'j'. While calculating the fitness function, if the current nest

fitness value (F_i) is lesser than the fitness value of arbitrarily picked nest (F_j) (that means $Fi < Fj$) then the arbitrarily picked nest (j) is supplanted by the new solution. On the off chance that the value of the fitness function of the cuckoo egg is not exactly or equivalent to the fitness function value of the arbitrarily picked nest (that means $Fi < Fj$) then the arbitrarily picked nest (j) is supplanted by the new solution. Here, to achieve the optimal result fitness functions is calculated as,

$$F_i = Cbest - Pbest \tag{17}$$

where, Cbest represents current best solution and Pbest represents previous best solution for selecting the number of resources. When the information is extant then the system assesses the sort of workload of tasks, whether it is CPU or memory intensive with their necessities. Algorithm 3 helps to find the optimum resources using cuckoo search and with the obtained optimum resources, the quality score will be calculated.

3.4.2 Queuing Theory Based Cuckoo Search (QTCS) Working

Algorithm 3: Queuing theory based Cuckoo search

Input: Objective Function f(x), (x_1, x_2, x_3, …, x_q)
Output: Optimum Resource
Begin
Initial population of m host is generated
$X_i = 1, 2, 3, \ldots, n)$
While (ti < Maximum Generation)
Read cuckoo randomly using Levy flights
Compute F_k = quality/fitness
Select a random between m nest
If (Fp < Fq) Then
replace q by new solution
End IF
A defective nest is rejected
Compute optimum nests
End While
Return Optimum resource

Here, the considered QoS parameters are resource availability, resource allocation time, and energy. The quality score (QS) is calculated among matched resources as in Eq. (18),

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

where, i = 1, 2, …, n, QS denotes the Quality Score, QS(i) is quality score for i_{th} resource. The highest quality score can be obtained using the Eq. (19).

$$Maximum\_QS = \max(QS(i)) \quad \text{where } i = 1, 2, \ldots, n \tag{19}$$

Maximum_QS is the highest score that shows the best quality among various resources. The highest QS value is selected in term of high resource availability, less response time, and energy for resource allocation.

4 Experimental Results and Discussion

CloudAnalyst simulator, which is developed on top of the CloudSim simulator, is used for simulating the proposed model. A data center with 50 heterogeneous physical nodes having CPU with performance more than 2000 million instructions per second, 8-GB RAM, and 1 TB of storage is created. When the user submits a request, it is processed and the capacity of the resource is simulated under the data center. This can be used as a model for creating the utilization of CPU according to the requirement randomly. Each experiment is simulated more than 10 times and the results are analyzed. The XEN server is the virtual server used to simulate the processing. In the simulation, 1000 requests/hour are taken as arrival rate of the requests in peak hours and 50 requests/hour are taken as arrival rate of the requests in non-peak hours with execution times of 60 min duration. At a regular interval of time, the simulator generates a set of requests and manages the traffic.

Since the numbers of users' requests are more at the same time, choosing the resources is also more in terms of QS. The number of matching (selected) resources is not constant for all the iterations. It depends on the availability of the resources and QS. The QS of the resource can be calculated by investigating the time taken to allocate the resource, energy consumption, resource availability etc. Using this QS, the appropriate resource can be chosen, which can provide better performance in the fog environment.

From the obtained results, it is clear that when request from user increases, the time taken for resource allocation also increases. The time taken to allocate the resources depends on the resource availability, distance between the resources and the client (request place). Fig. 2 describes the execution time ranges from 8, 23, 29, 34, and 38 s of user requests of 50, 100, 150, 200, and 250 respectively. Fig. 3 describes total number of requests allocated based on the matched resources.

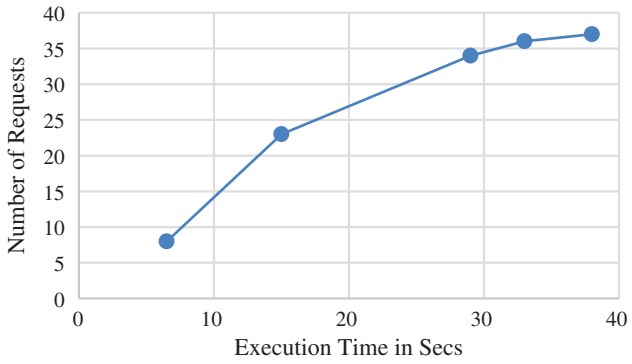

Figure 2: Execution time comparison of the requests

Energy consumption is directly proportional to the increasing number of requests. To allocate the resources, fog entity needs some amount of energy. A resource needs certain amount of energy for each state like active, service, and live etc. When a resource is investigated, it should be active and involved in the job. Fig. 4 clearly depicts that while the number of resource requests increases, simultaneously the time taken for resource allocation also increases.

Figure 3: Number of requests allocated by matched resources

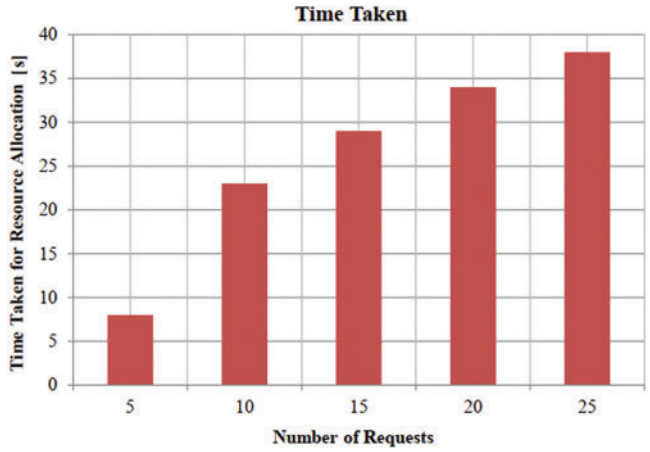

Figure 4: Number of Requests *vs.* Time

It is also simulated for large sized problems such as 50, 100, 200, 500, and 1000 requests and the performance is evaluated. The energy savings are in a range from 96% to 99% when the user requests are among 50–1000. When the numbder of requests increase, the number of resources that are matched for the requests also increases. The resources availability can be verified dynamically while evaluating the fog user request by comparing the request-based resources by the multiple fog services. Since multiple services are working simultaneously, the time complexity does not affecting the fog computing-based resource allocation. Here, Fig. 5., describes the number of requests arrived and allocated by matched resources in the fog environment.

From the experimental simulation results, the proposed QTCS model improves the response time and energy consumption by 20.39% and 12.55% respectively when compared to the existing energy based K-means hybrid method resource allocation (EBKHRA) model [11]. The simulation results illustrate that the proposed approach is efficient and has a good potential for the large data center in the fog.

Here, Tab. 2. and Fig. 6., describe the comparison of energy consumption of existing EBKHRA model and proposed QTCS model.

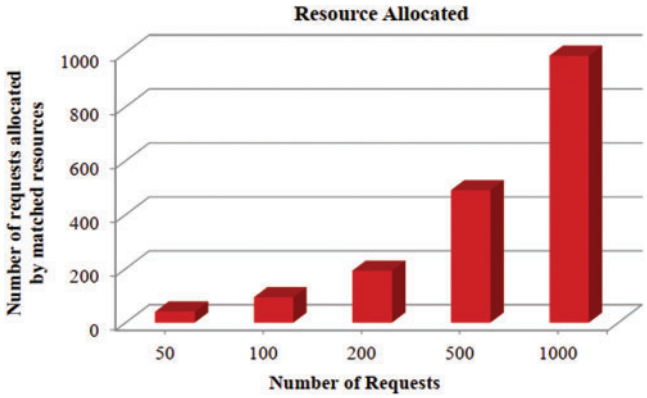

Figure 5: Number of requests *vs.* resource allocated

Table 2: Comparison table of EBKHRA vs QTCS

Energy consumption					
Number of request	100	200	300	400	500
EBKHRA	20	25	30	35	40
QTCS	20	15	20	25	30

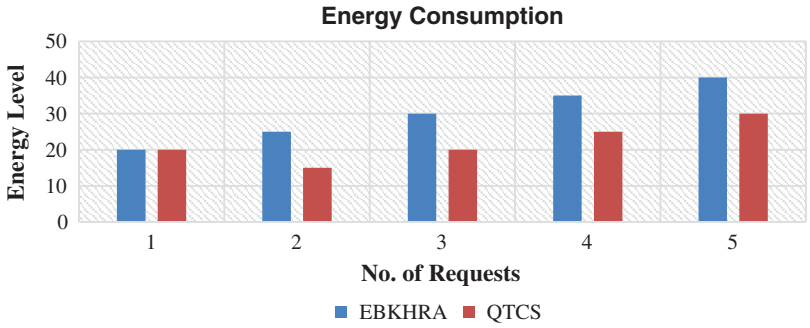

Figure 6: Comparison chart of EBKHRA *vs.* QTCS

5 Significance of the Proposed Work

The proposed QTCS is compared with the EBKHRA model to evaluate the performance of the proposed work.

- The proposed QTCS model helps in assigning the resources efficiently using priority-based resource allocation.
- The priority-based QTCS helps to reduce the energy consumption.
- The proposed QTCS model reduces the overall time for the resource allocation process.

6 Conclusion and Future Direction

The proposed QTCS model was implemented in real time fog environment for allocating the resources quickly and dynamically. After a thorough survey, we infered that the QoS parameters

needs to be improved during resource allocation. In our proposed model simulation results, the evaluation of QoS metrics such as resource allocation, response time, and the energy consumption were performed. By using the derived results, it is evident that the average response time has increased by 20.39% and average of energy consumption got reduced by 12.55%, when compared with existing approaches. Based on the experimentation results, it is infered that the proposed method is efficient for resource allocations dynamically even when the numbers of user requests are high. The QTCS approach also improves QoS parameters in terms of response time and energy consumption. Deploying a large application in a fog environment and managing the resource allocation with respect to the edge nodes still has major challenges. This work can be enhanced further by incorporating Markovin model into a fog node in order to produce an optimal solution during resource allocation.

Funding Statement: The authors revived no funding for this work.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] S. Singh and I. Chana, "A survey on resource scheduling in cloud computing: Issues and challenges," *Journal of Grid Computing*, vol. 14, no. 2, pp. 217–264, 2016.

[2] P. K. R. Maddikunta, T. R. Gadekallu, R. Kaluri, G. Srivastava, R. M. Parizi *et al.,* "Green communication in IoT networks using a hybrid optimization algorithm," *Computer Communications*, vol. 159, no. 1, pp. 97–107, 2020.

[3] M. Alazab, S. Khan, S. S. R. Krishnan, Q. V. Pham, M. Reddy *et al.,* "A Multidirectional LSTM model for predicting the stability of a smart grid," *IEEE Access*, vol. 8, pp. 85454–85463, 2020.

[4] M. Numan, F. Subhan, W. Z. Khan, S. Hakak, S. Haider *et al.,* "A systematic review on clone node detection in static wireless sensor networks," *IEEE Access*, vol. 8, pp. 65450–65461, 2020.

[5] A. Wolke, M. Bichler, F. Chirigati and V. Steeves, "Reproducible experiments on dynamic resource allocation in cloud data centers," *Information Systems*, vol. 59, pp. 98–101, 2016.

[6] R. K. Naha, S. Garg, D. Georgakopoulos, P. P. Jayaraman, L. Gao *et al.,* "Fog computing: Survey of trends, architectures, requirements, and research directions," *IEEE access*, vol. 6, pp. 47980–48009, 2018.

[7] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali *et al.,* "All one needs to know about fog computing and related edge computing paradigms: A complete survey," *Journal of Systems Architecture*, vol. 98, pp. 289–330, 2019.

[8] J. Moura and D. Hutchison, "Fog computing systems: State of the art, research issues and future trends, with a focus on resilience," *Journal of Network and Computer Applications*, vol. 169, no. 1, pp. 102784, 2020.

[9] F. Murtaza, A. Akhunzada, S. ul Islam , J. Boudjadar and R. Buyya, "QoS-aware service provisioning in fog computing," *Journal of Network and Computer Applications*, vol. 165, no. 1, pp. 102674, 2020.

[10] Z. Qu, Y. Wang, L. Sun, D. Peng and Z. Li, "Study QoS optimization and energy saving techniques in cloud fog, edge, and IoT," *Complexity*, vol. 2020, pp. 2022–2038, 2020.

[11] RM. Swarna Priya, S. Bhattacharya, P. K. R. Maddikunta, S. R. K. Somayaji, K. Lakshmanna *et al.,* "Load balancing of energy cloud using wind driven and firefly algorithms in internet of everything," *Journal of Parallel and Distributed Computing*, vol. 142, pp. 16–26, 2020.

[12] RM. Swarna Priya, P. K. R. Maddikunta, M. Parimala, S. Koppu, T. Reddy *et al.,* "An effective feature engineering for DNN using hybrid PCA-GWO for intrusion detection in IoMT architecture," *Computer Communications*, vol. 160, no. 1, pp. 139–149, 2020.

[13] S. Mustafa, K. Bilal, S. U. R. Malik and S. A. Madani, "SLA-aware energy efficient resource management for cloud environments," *IEEE Access*, vol. 6, pp. 15004–15020, 2018.

[14] A. U. Rehman, Z. Ahmad, A. I. Jehangiri, M. A. Ala'Anzy, M. Othman *et al.,* "Dynamic energy efficient resource allocation strategy for load balancing in fog environment," *IEEE Access*, vol. 8, pp. 199829–199839, 2020.

[15] T. Reddy, RM. Swarna Priya, M. Parimala, C. L. Chowdhary, S. Hakak *et al.,* "A deep neural networks based model for uninterrupted marine environment monitoring," *Computer Communications*, vol. 164, no. 1, pp. 64–75, 2020.

[16] S. Bhattacharya, S. S. R. Krishnan, P. K. R. Maddikunta, R. Kaluri, S. Singh *et al.,* "A novel PCA-firefly based XGBoost classification model for intrusion detection in networks using GPU," *Electronics*, vol. 9, no. 2, pp. 219, 2020.

[17] R. Shone, K. Glazebrook and K. G. Zografos, "Resource allocation in congested queueing systems with time-varying demand: An application to airport operations," *European Journal of Operational Research*, vol. 276, no. 2, pp. 566–581, 2019.

[18] A. Samson Busuyi and A. Bagula, "Improving quality-of-service in cloud/fog computing through efficient resource allocation," *Sensors*, vol. 19, no. 6, pp. 1267, 2019.

[19] M. Sathish Kumar and M. Iyapparaja, "Improving quality-of-service in fog computing through efficient resource allocation," *Computational Intelligence*, vol. 36, no. 4, pp. 1–21, 2020.

[20] L. Li, Q. Guan, L. Jin and M. Guo, "Resource allocation and task offloading for heterogeneous real-time tasks with uncertain duration time in a fog queueing system," *IEEE Access*, vol. 7, pp. 9912–9925, 2019.

[21] S. K. Battula, S. Garg, J. Montgomery and B. Kang, "An efficient resource monitoring service for fog computing environments," *IEEE Transactions on Services Computing*, vol. 13, no. 4, pp. 709–722, 2020.

[22] C. Swain, M. N. Sahoo, A. Satpathy, K. Muhammad, S. Bakshi *et al.,* "METO: Matching theory based efficient task offloading in IoT-fog interconnection networks," *IEEE Internet of Things Journal*, vol. 10, pp. 1–11, 2020.

[23] X. Huang, W. Fan, Q. Chen and J. Zhang, "Energy-efficient resource allocation in fog computing networks with the candidate mechanism," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8502–8512, 2020.

[24] M. Ahmed-Nacer, K. Suri, M. Sellami and W. Gaaloul, "Simulation of configurable resource allocation for cloud-based business processes," in *2017 IEEE Int. Conf. on Services Computing*, Honolulu, HI1, pp. 305–313, 2017.

[25] P. D. Kaur and I. Chana, "Cloud based intelligent system for delivering health care as a service," *Computer Methods and Programs in Biomedicine*, vol. 113, no. 1, pp. 346–359, 2014.

[26] A. Singh, D. Juneja and M. Malhotra, "A novel agent based autonomous and service composition framework for cost optimization of resource provisioning in cloud computing," *Journal of King Saud University—Computer and Information Sciences*, vol. 29, no. 1, pp. 19–28, 2017.

[27] J. Espadas, A. Molina, G. Jiménez, M. Molina, R. Ramírez *et al.,* "A tenant-based resource allocation model for scaling software-as-a-service applications over cloud computing infrastructures," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 273–286, 2013.

[28] W. Y. C. Wang, A. Rashid and H. M. Chuang, "Toward the trend of cloud computing," *Journal of Electronic Commerce Research*, vol. 12, no. 4, pp. 238, 2011.

[29] A. V. Karthick, E. Ramaraj and R. G. Subramanian, "An efficient multi queue job scheduling for cloud computing," *IEEE 2014 World Congress on Computing and Communication Technologies*, vol. 1, pp. 164–166, 2014.

[30] P. Singh and N. K. Walia, "A review: Cloud computing using various task scheduling algorithms," *International Journal of Computer Applications*, vol. 42, no. 7, pp. 30–32, 2016.

[31] A. K. Tripathy, M. R. Patra, M. A. Khan, H. Fatima and P. Swain, "Dynamic web service composition with QoS clustering," in *IEEE 2014 Int. Conf. on Web Services*, Anchorage, AK, USA, vol. 1, pp. 678–679, 2014.

[32] E. Shimpy and M. J. Sidhu, "Different scheduling algorithms in different cloud environment," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 3, no. 9, pp. 8003–8006, 2014.

[33] M. Patel and R. Kadian, "A review on ACO based scheduling algorithm in cloud computing," *International Journal of Computer Science and Mobile Computing*, vol. 5, no. 5, pp. 489–493, 2016.

[34] E. N. Alkhanak, S. P. Lee and S. U. R. Khan, "Cost-aware challenges for workflow scheduling approaches in cloud computing environments: Taxonomy and opportunities," *Future Generation Computer Systems*, vol. 50, pp. 3–21, 2015.

[35] T. R. Gadekallu and N. Khare, "Cuckoo search optimized reduction and fuzzy logic classifier for heart disease and diabetes prediction," *International Journal of Fuzzy System Applications*, vol. 6, no. 2, pp. 25–42, 2017.

[36] D. Maharana, B. Sahoo and S. Sethi, "Energy-efficient real-time tasks scheduling in cloud data centers," *International Journal of Science Engineering and Advance Technology*, vol. 4, no. 12, pp. 768–767, 2017.

[37] X. Kong, L. Yan, D. Wang, M. Yu and X. Liu, "Evaluation of education resources allocation in Beijing based on entropy-TOPSIS method," *Journal of Physics: Conference Series*, vol. 1670, no. 1, pp. 12042, 2020.