https://doi.org/10.48048/tis.2021.413

A Multi Agent Based Dynamic Resource Allocation in Fog-Cloud Computing Environment

Ismail Zaharaddeen Yakubu[*], Lele Muhammed, Zainab Aliyu Musa, Zakari Idris Matinja and Ilya Musa Adamu

Department of Computer Science, Federal Polytechnic Bauchi, Bauchi740102, Nigeria

([*]Corresponding author's e-mail: ysbfamily2010@gmail.com)

Received: 16 May 2020, Revised: 21 May 2021, Accepted: 26 May 2021

Abstract

Cloud high latency limitation has necessitated the introduction of Fog computing paradigm that extends computing infrastructures in the cloud data centers to the edge network. Extended cloud resources provide processing, storage and network services to time sensitive request associated to the Internet of Things (IoT) services in network edge. The rapid increase in adoption of IoT devices, variations in user requirements, limited processing and storage capacity of fog resources and problem of fog resources over saturation has made provisioning and allotment of computing resources in fog environment a formidable task. Satisfying application and request deadline is the most substantial challenge compared to other dynamic variations in parameters of client requirements. To curtail these issues, the integrated fog-cloud computing environment and efficient resource selection method is highly required. This paper proposed an agent based dynamic resource allocation that employs the use of host agent to analyze the QoSrequirements of application and request and select a suitable execution layer. The host agent forwards the application request to a layer agent which is responsible for the allocation of best resource that satisfies the requirement of the application request. Host agent and layers agents maintains resource information tables for matching of task and computing resources. CloudSim toolkit functionalities were extended to simulate a realistic fog environment where the proposed method is evaluated. The experimental results proved that the proposed method performs better in terms of processing time, latency and percentage QoS delivery.

Keywords: Agent, Fog computing, IoT, Latency, Processing time, Resource allocation

Introduction

Cloud computing is one of the emerging, promising and pay on demand computing model [1-3, 29-31]that provide configurable computing infrastructures to end users in an on-demand fashion that comes up with various benefits, such as ubiquitous and flexible access to data, resource scaling and availability [4-6]. Conventionally, IoT devices like smart phones, home automation devices, medical infrastructures, and other IoT devices remotely communicate with cloud for task processing. This incurs large latency and network bandwidth overload that results to adversity in real time interactions. To curtail the defects of cloud computing, fog computing was introduced to facilitate applications like smart transportation [7] and health care solutions [8] to perform application processing on the edge network proximate to the client using the computing power of the diverse end and edge nodes [9-11].

Fogmodel is a3-layer architecture that resides between cloud and edge devices [12,13]. In fog computing each resource with a processing power or a free resource serve as a fog node.Contrary to the cloud paradigm that relies on a stationary data centremanipulated by the owner, [14,15] fog nodes are manipulated by diverse owners or manipulating body. Both fog and cloud computing paradigm are distributed computing systems with cloud being more centralized and fog being more decentralized [16].

The rapid increase in the adoption of IoT devices has resulted to high demand of fog resources for processing of time sensitive applications and clients request. The behaviour of the client request varies with respect to various parameters like response time, cost, deadline, latency, memory and more. The major challenge is that the fog system has a decentralized and definite resource. Thus, in order to reduce the burden and over saturation of fog resources and satisfy the processing and storage needs of

applications and client request simultaneously, both applications and client request cannot be processed only on fog without the help of cloud.

This paper proposed an agent based dynamic resource allocation (AB-DRA) on the hybrid Fog-Cloud computing environment. The objectives of this work are as follows:

- To reduce the burden and oversaturation on fog resources and enable prompt response to time sensitive applications and client request.
- To propose an agent based method for examining the behaviour and requirements of applications and client request for selection of appropriate layer for the execution of such application and request.
- To propose a resource ranking algorithm that ease the decision process of agent during layer selection.
- To propose a resource selection method that allocates the best resource capable of satisfying the application and request processing and storage need.

The organization of this paper is as follows: Section 2 focuses on work related to resource allocation on hybrid fog-cloud environment, section 3 describes the architecture of the fog-cloud environment assumed in this work, section 4 provides the detail description of the agent based resource allocation method, section 5 presents the steps involved in the experiment and the results obtained, and lastly, section 6 concludes our work and presents the future research trend.

Literature review

In [19], authors proposed a priority task scheduling method in fog computing environment that tends to execute tasks at the fog layer according to their levels of priority. The proposed scheduling scheme propagatesthe remaining task to cloud for processing upon saturation of the fog nodes.

In [20], authors explore the technology behind fog computing and their applications in the field of IoT. The configurations of the fog nodes, fog computing platform and the key technological innovations in fog computing like Software Defined Networking (SDN), Network Function Virtualizations (NFV) and 5G technologies were explored.

In [21], authors inspect the IoT-Fog-Cloud ecosystem and supply a literature review from various aspects of the system. The organizational structure and the management aspect of the ecosystem were also explored. Authors also explain how applications can benefit from the ecosystem and the challenging concerns to be treated in the IoT-Fog-Cloud infrastructures.

In [22], authors proposed a tool called iFogSim for the modelling and simulation of resource manipulation methods in IoT, edge and Fog computing environment. iFogSim can be used to evaluate the influence of resource management methods in delay, network congestion, cost as well as the resource energy consumption. Two case studies were depicted to prove modelling of IoT environment and examination of resource management policies.

In [23], a workload balancing and cloud resource optimization approach was proposed. The proposed method deploys the dynamic cloud resource allotment approach based on reinforcement learning and genetic algorithm to ceaselessly monitor network traffic, gather information about the load on every server, manage incoming requests, and equally distribute the request to available servers using dynamic resource allocation strategy.

In [24], authors try to reduce the amount of energy consumed by nodes at the edge network and reduce the delay in the processing of workload. To achieve this, authors proposed 2 methods based on XCS learning classifier system called XCS and BCM-XCS. The proposed method distributes request across the resources such that processing time and communication latency between cloud and fog are minimal.

In [25], authors proposed a profit aware application placement policy for the hybrid Fog-Cloud system. The policy is articulated using constraint integer linear programming model that concurrently improves profit and guarantees QoS delivery throughout application placement on computing resource instances. The method also set the price of resource instance in decreasing order of their service delivery time. Upon violation of the Service Level Agreement (SLA), the method gives compensation to users.

In [26], authors proposed a fog based IoT broker that decides a capable notification degree to access the IoT resource while maximizing the required application QoS and preventing congestion in the network. First, the degree selection is standardized as an optimization problem and then a practical algorithm that provides transmission dependability to vigorously select notification periods is proposed.

In [27], authors proposed a framework for secure computing resource allocation in fog computing environment. In this method the fog servers are reserved for computing request execution and resource assignments whereas the cloud audit centre is reserved for fog server and fog devices actions scrutiny.

The proposed scheme is capable of withstanding an attack of single vicious node and the conspiracy attack of fog server and computing resource.

In [28], authorstry to reduce the delay and service overhead and increase the cloud ability and dependability. Service requests are assigned to the most sufficient node in the hybrid Fog-Cloud computing environment. The method grouped request issuing devices into time sensitive, time tolerant and core. Time sensitive requests are assigned to edge nodes using a prepared executive device list. Time tolerant requests are assigned to either cloud edge devices or cloud core and core requests are assigned to cloud core.

In [32], authors minimized cloud latency and ensure QoS delivery by proposing2 schedulers to take decision on the cloud-fog layer to execute an application. The schedulers were based on linear integer programming that schedule task on fog resources or on cloud resources. The schedulers consider the class of services while choosing the processing elements to process the client task.

In [33], a trust-enabled communication structure was built in cloud-fog-edge environment using block chain technology. Authors introduced Berger's approach and concept of service justice to ensure workload balance and matching of task and computing resource for QoS delivery.

In [34], authors try to maximize the number of service request from the users on the three-tier fog-cloud computing environment considering their deadlines. To ensure QoS delivery, deadline misses were minimized using mixed integer programming optimization model.

In [35], attributes associated to computing resources are standardized and normalized and resources are partitioned based on fuzzy clustering combined with particle swamp optimization to reduce the resource search scale. To curtail the high service request delay due to cloud, computing resources are scheduled based on optimized fuzzy clustering algorithm.

Materials and methods

In this section we present the system architecture used in our work as shown in **Figure 1** and the proposed resource allocation and scheduling model in the fog-cloud computing environment.

System architecture

The fog-cloud architecture consists of fog layer, high capacity fog layer and cloud layer as depicted in **Figure 1**. Fog layer consists of fog nodes connected to fog server responsible for managing the resources. The fog agent in the fog layer ranks the fog resources according to their computing capacity and store the ranking information in the fog agent dynamic resource table. The highest ranked resource is updated in the agent information table for agent selection by the host agent. A central scheduler residing in the fog layer receives and schedule client request based on priority.

The configuration of high capacity fog resources is greater compared to the fog layer resources but the fog layer resources are more proximate to the client devices. These resources have higher memory, number of cores, network and storage capacity compared to the fog resources. Fog layer, high capacity fog layer and cloud layer are connected to form a large network. The high capacity fog agent in the high capacity fog layer ranks the layer resources according to their computing capacity and store the ranking information in the high capacity fog agent dynamic resource table. The highest ranked resource is updated in the agent information table for agent selection by the host agent.

The cloud layer consists of more resources with higher configuration than the fog layer and high capacity fog layer resources but far away from the client. The cloud agent in the cloud layer ranks the layer resources according to their computing capacity and store the ranking information in the cloud agent dynamic resource table. The highest ranked resource is updated in the agent information table for agent selection by the host agent.

Host agent receives client tasks from the scheduler, examine each task and make decision on which agent to allocate a resource for the execution of such task based on specified QoS requirements like processing capacity, tolerable latency, bandwidth, and response time. The host agent uses an agent information table to decide whether a task will be executed at the fog layer, high capacity fog layer or cloud layer. Agent information table store agent's identifiers, processing capacity, latency, and storage capacity, bandwidth, and response time of highest ranked resource in each dynamic resource table of the agents. Host agent first check the possibility of executing the task on cloud then high capacity fog layer and finally on fog layer.

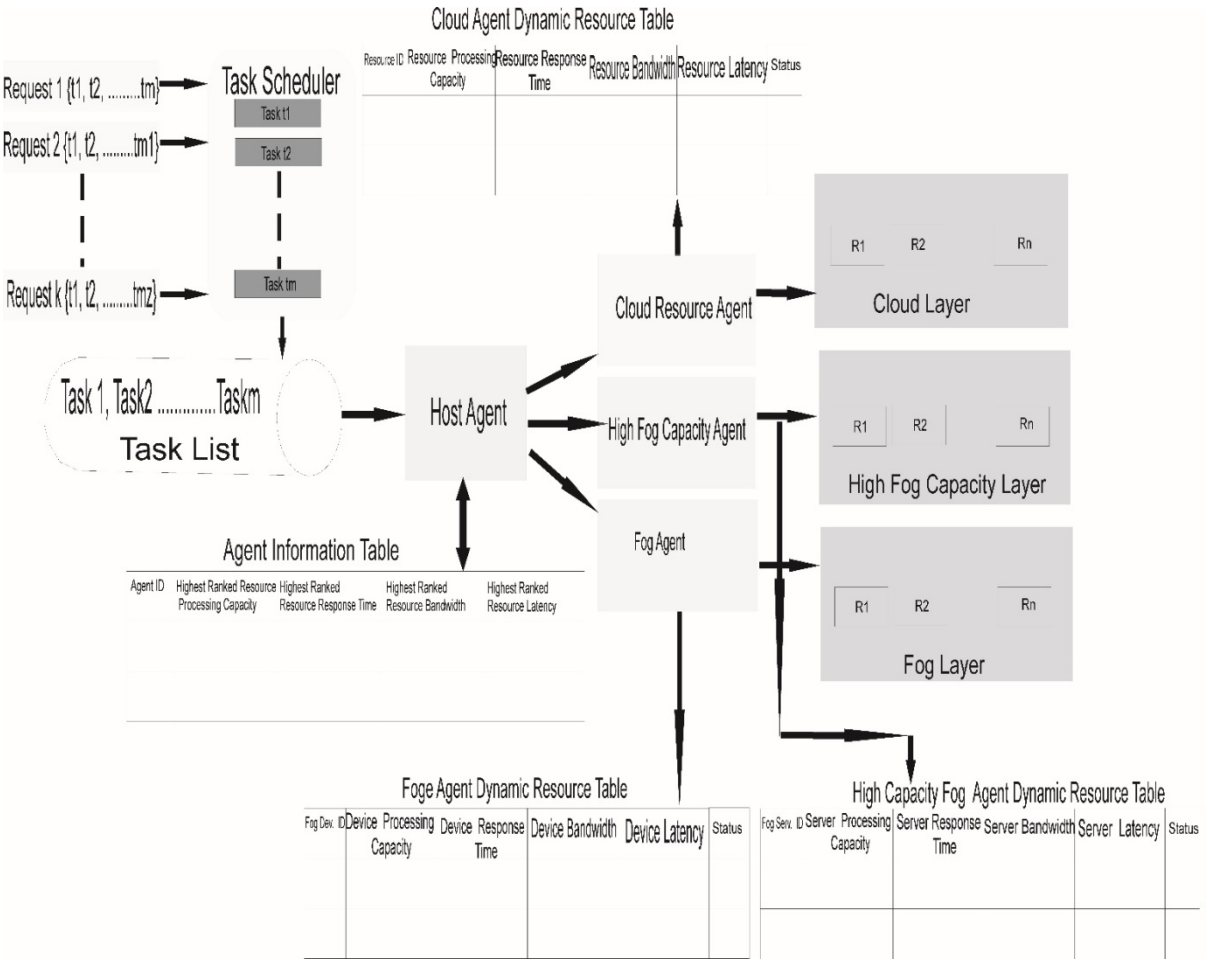

Figure 1 System architecture.

Resource allocation model

The proposed resource allocation model consists of task scheduling which is carried out by the central scheduler, layer agent selection by the host agent, resource ranking, dynamic resource table update and resource selection by layer agent.

Task scheduling

The scheduler receives client request consisting of m tasks, computes the deadline for each task based on the request deadline and submits the task to the host agent for layer selection.

Consider a client request R consisting of m task, represented as follows:

$$R= \{t_1, t_2, ., ., t_m\} \quad (1)$$

where $1 \leq m < \infty$

The length of each task within a request may vary from one another. The scheduler computes the deadline of each task within a request by dividing the request deadline with the average length of all tasks in each request as follows:

$$t^d = R^d * \frac{M}{\sum_{i=1}^{m} l_i} \quad (2)$$

where $1 \leq m < \infty$

Variables t^d, R^d, M and l_i represent task deadline, request deadline, number of tasks within a request and the length of a task. Each task within a request is allocated a deadline t^d by the scheduler such that:

$$R^d = \sum_{i=1}^{m} t_i \quad ^d \tag{3}$$

The scheduler finally submits the request to the host agent maintaining the task sequence.

Agent module

An agent establishes agent to agent connection for client to fog resource, client to cloud resource or client to high capacity fog resource matching based on resource availability and resource capability. A connection is made when

$$C_n = \begin{cases} 0, & t_m > 0 \\ 1, & Otherwise \end{cases} \tag{4}$$

where C_n represents connectivity made at runtime between client request and fog resource, high capacity fog resource or cloud resource and t_m refers to tasks of a request.

Let A represent the set of agents in the fog-cloud environment. The elements of set A are represented as follows:

$$A = \{a_h, a_f, a_c, a_s\} \tag{5}$$

Where za_h, a_f, a_c, a_s represent host agent, fog layer agent, cloud resource agent and high capacity fog agent, respectively.

The fog layer agent, high capacity fog agent and cloud agent ranks the fog resource's, high capacity fog resource's and cloud resource's based on their processing capacity, bandwidth, latency and response time. The ranking by each agent is stored in the corresponding agent dynamic resource table and the highest ranked resource is updated in the agent information table used by host agent for agent selection.

Resource ranking

Let R_{pc}, R_{bw}, R_{rt}, R_{lc} represent the processing capacity, bandwidth, response time and latency of a resource.

Let C_r, R_c, R_k, R[],C_p, C_b, C_{rt} C_l, A_t represent the current resource to be ranked, resource counter, resource rank, resource list and initial value of processing, bandwidth, response time, latency of a resource, and Dynamic resource table

The resource ranking algorithm ranks the resources based on their processing capacity, bandwidth, response time and latency.

Algorithm 1 Resource ranking algorithm.

Input: List of resource R_L
Output: Ranked resource list

$R_c = 1$ * initializing the resource counter*\
$R_k = 1$ * initializing resource rank*\
$C_p = R[R_c] . R_{pc}$ * initializing the processing capacity of current resource to capacity of resource in the counter index*\
$C_{bw} = R[R_c] . R_{bw}$ * initializing the bandwidth of current resource to bandwidth of resource in the counter index*\
$C_{rt} = R[R_c] . R_{rt}$ * initializing the response time of current resource to response time of resource in the counter index*\
$C_l = R[R_c] . R_{lc}$ * initializing the latency of current resource to latency of resource in the counter index*\
For ∀ R[] **do** * repeat for all resources in the resource list*\
    For i=1:R[].size
        If R[i]. $R_{pc} \geq C_p$& R[i].$R_{bw} \geq C_{bw}$&R[i]. $R_{rt} \geq C_{rt}$
            &R[i]. $R_{lc} \geq C_l$ * compare the capacity of the current resource and the resource in the counter index*\
        $C_r = R[i]$ * set current resource to resource in the counter index if above condition is true*\
    End for
    Rank = R_k * rank the resource with the value in the rank variable*\
    Update C_r Rank in A_t * update current resource in dynamic resource table*\
    Remove(R[].C_c) *remove current resource from resource list*\
    $R_k = R_k + 1$ * increment resource rank*\
    $R_c = R_c + 1$ * increment resource counter*\
End for

The capacity of the highest ranked resource is updated in the agent information table for agent allocation.

The host agent examines the requirements of the tasks in the task list and selects a suitable layer for the execution of such tasks. The host agent establishes a connection with the agent of the selected layer and forwards the task to the layer agent.

Algorithm for agent selection

Let B_{max}, P_{max}, L_{cmax}, S_{max}, R_{tmax} represent bandwidth, processing capacity, latency, storage, and response time of the highest ranked resource in fog, high capacity fog and cloud layer.

Let B_r, P_r, L_{ct}, S_r, R_{tr} represent task required bandwidth, processing capacity, task tolerable latency, task required storage, and response time.

The host agent selects an agent to allocate a resource for the execution of task by comparing the QoS requirement of the task and the capacity of the highest ranked resource in each host dynamic resource table.

Algorithm 2 Agent selection algorithm.

Input: Task list from a request R, Agent list and list of highest ranked resources in fog, high capacity fog and cloud layer.
Output: Task to agent map
For ∀ t ε R **do** * repeat for all task in a request*\
If $t^d \leq a_c.\ L_{cmax} + a_c\ .R_{tmax} \& S_r \leq a_c.S_{max}$ * compare task deadline with sum of latency & response time of cloud *\
Submit task to cloud agent * submit task to cloud agent if above condition is true*\
Else if $t^d \leq a_s.\ L_{cmax} + a_s\ .R_{tmax} \& S_r \leq a_s.S_{max}$ * compare task deadline with sum of latency & response time of high capacity fog *\
Submit task to high capacity fog agent * submit task to high capacity fog agent if above condition is true*\
Else if $t^d \leq a_f.\ L_{cmax} + a_f\ .R_{tmax} \& S_r \leq a_f.S_{max}$ * compare task deadline with sum of latency & response time fog layer*\
Submit task to fog agent * submit task to fog agent if above condition is true*\
Else
Return (Resource unavailability message) * return failure due to resource unavailability*\
End if
End for.

An agent allocate a capable resource for task received from host agent by taking into account the deadline of the task, required processing capacity, bandwidth, response time and tolerable latency. Any available resource that meets the requirement of the task is allocated for execution of such task.

Algorithm for resource allocation

Let B_w, P_c, L_c, S, R_t represent bandwidth, processing capacity, latency, storage, and response time of resource in fog, high capacity fog or cloud layer.

Let B_r, P_r, L_{ct}, S_r, R_{tr}, t^d represent task required bandwidth, processing capacity, task tolerable latency, task required storage, response time, and task deadline.

The resource allocation algorithm selects the best available resource for execution of task that satisfies the QoS requirement of the task.

Algorithm 3 Resource allocation algorithm.

Input: List of Resource R [] and List of task T []
Output: Resource Allocation
For ∀ t ε T[] * repeat for all task in task list*\
For ∀ r ε R[] * repeat for all resource in resource list*\
If $t^d \leq L_c + R_t \&\ P_c \geq P_r \&$ r.status = free * compare task deadline with sum of latency & response time of resource *\
Allocate resource to task * allocate resource to task if above condition is true *\
r.status = allocated. * change resource status to allocated *\
t.status = allocated * change task status to allocated *\
End if
End for
If t.status=!Allocated * change task status not allocated *\
Return (Resource unavailability message) * return failure due to resource unavailability*\
End for

Dynamic resource table update

Dynamic resource table update monitors the resources in fog-cloud environment and updates the status of each resource. It checks if a resource has completed the execution of task and changed the status of the resource for further allocation by agents.

Dynamic resource table update algorithm

Let R [], A_t, C_t represent resource list, corresponding agent table, and current task executed by resource.

The dynamic resource table updates algorithm updates the status of a resource after task execution is completed.

Algorithm 4 Dynamic resource table update algorithm.

Input: Resource list R[], and agent table A_t
Output: Updated resource table
For ∀ r ε A_t. R[] * repeat for all resource in dynamic resource update table*\
t_c = R[].Get (C_t) * get the current task executed by a resource*\
Ift_c = NULL * if task not found*\
r.status = free * change the status of the resource to free*\
End If
End for

Dynamic resource table update algorithm updates the status of an allocated resource to enable the agent to reallocate the resource after execution of the task is completed.

Performance evaluation

In this section, we present the techniques used to evaluate the method in this paper, metrics used for the evaluation, the environment used and the environmental setup and finally we present the experimental results obtained.

Experimental setup

CloudSim environment is used to conduct the experiment. The environment functionalities is adopted and extended to create the scalable cloud data center with series of resources (Virtual Machines) running on physical machines, 3 resource layers and layers agents, a central scheduler that schedules task for layer and resource allocation. The layers are initialized with resources and the dynamic resource table and information table are initialized by the layer agents.

Tables 1 and **2** below present the hardware requirements and environment specification for the implementation of the method in this paper.

Table 1 Hardware requirement.

Required	Hardware and software specification
HDD	500GB
Processor	Intel(R) Core (TM) i5-3120M CPU@2.50GHz
RAM	4GB
Operating System	Windows 7 (x64) 64-bit OS
System	64 Bit OS System

Table 2 Simulation specification.

Components	Specifications	Values
Resources	Cloud Host	2
	Fog Host	2
	High capacity fog Host	2
Cloudlets	Number of task	100
	Task length	1000-5000MIPS
Physical Machines	Memory	2048GB
	Bandwidth	10GHz
	Storage	500GB
	Processing Elements	10

Performance metrics

In this paper, we used processing time, latency, and percentage QoS expectations delivery with respect to the computing capacity allocated to user task as metrics to evaluate the performance of our method. Fog computing environment was introduced to specifically serve delay sensitive applications; as such the major factors to guarantee are latency and processing time. Latency can be calculated as follows:

$$l_t = t_{est} - t_{st} \tag{6}$$

where l, t_{st}, t_{est}, t_{st} represent latency, task execution time and task submission time.

Sine 3 layered fog architecture is used the overall latency due to fog resources, high capacity fog resources and cloud resources can be calculated as follows:

$$l_t = l^{cl} + l^{fd} + l^{fs} \tag{7}$$

where l^{cl}, l^{fd}, l^{fs} represent cloud latency, fog latency and high capacity fog latency.

The processing time of the system is calculated as follows:

$$P_t = E_{et} - E_{st} \tag{8}$$

where P_t, E_{et}, E_{st} represent processing time, task execution end time and task execution start time.

Also, since 3 layered fog architecture is used the overall processing time due to fog layer resources, high capacity fog layer resources and cloud layer resources can be calculated as follows:

$$P_t = E^{cl} + E^{fd} + E^{fs} \tag{9}$$

where $E^{cl} + E^{fd} + E^{fs}$ represent cloud execution time, fog execution time and high capacity fog execution time.

Evaluation technique

Resource-aware and latency-aware methods are the 2 baselines used to evaluate the method in this paper as they try to improve the performance of the system considering latency and resource availability. Processing power, bandwidth, latency and bandwidth are the QoS parameters used in allocating a resource to a task in the Fog-Cloud computing environment. Latency aware algorithm extends resource aware algorithm taking into consideration both the internal and overall latency. In our proposed solution, processing capacity, bandwidth, latency and response time are considered for ranking of provisioned resources. The resources are provisioned before the arrival of client request to avoid delay due to resource provisioning.

Results and discussion

Tables 3 through **4** show the average latency, average processing time and percentage QoS delivery for AB-DRA, latency-aware and resource-aware methods

Table 3 Average latency (ms) for AB-DRA, Resource-aware and Latency-aware method.

	Average latency (ms)		
No of Task	**AB-DRA**	**Resource-Aware**	**Latency-Aware**
20	201	223	224
40	342	355	357
60	432	480	482
80	578	583	586
100	680	713	723

Table 4 Average processing time (ms) for AB-DRA, Resource-aware and Latency-aware method.

	Average processing time (ms)		
No of Task	**AB-DRA**	**Resource-Aware**	**Latency-Aware**
20	28	33	34.2
40	28.3	34.1	34.2
60	29.5	34.2	34.4
80	30.1	34.3	34.5
100	30.2	34.3	34.7

Table 5 Percentage QoS delivery (%) for AB-DRA, Resource-aware and Latency-aware method.

	Percentage QoS delivery (%)		
No of task	**AB-DRA**	**Resource-aware**	**Latency-aware**
20	99.9	89.7	91.9
40	99.9	86.9	89.6
60	99.9	81.5	86
80	99.9	75.9	82.7
100	99.9	70.2	78.1

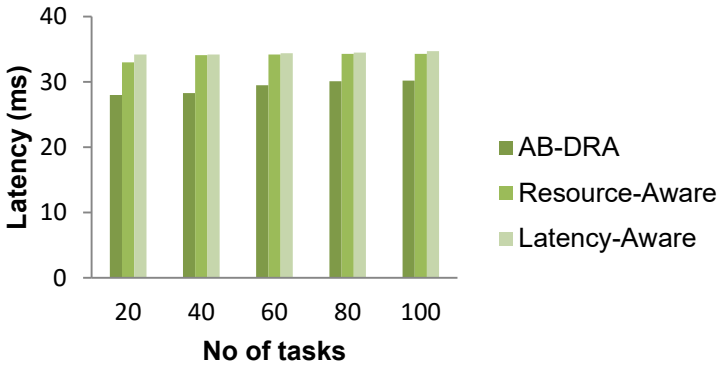

Figure 2 Average latency in fog-cloud environment.

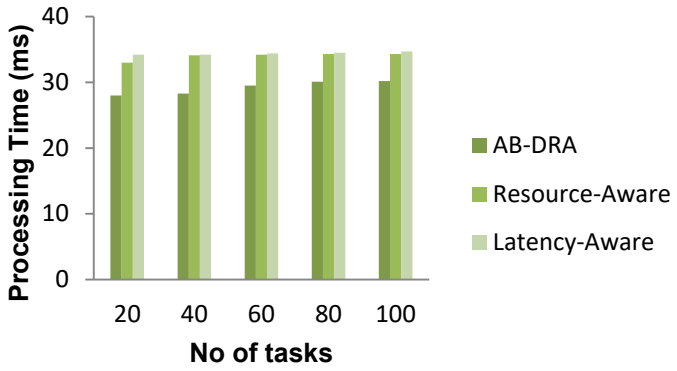

Figure 3 Average processing time in fog-cloud environment.

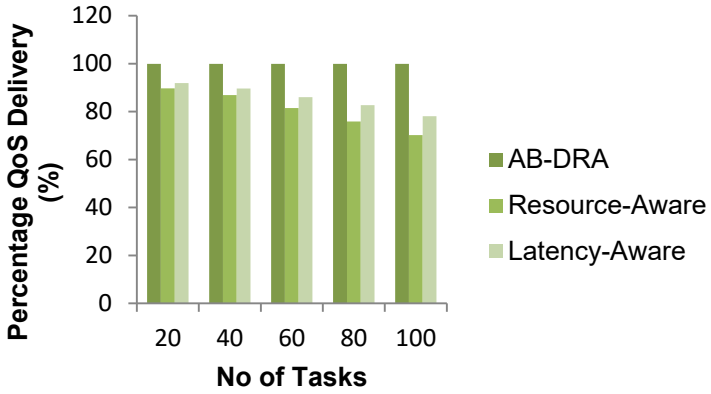

Figure 4 Percentage QoS delivery in fog-cloud environment.

Figures 2 through **4**show the simulation results in different experimental settings. **Figures 2** and **3** show the comparison of average latency and processing time between resource-aware and latency-aware allocation methods where AB-DRA recorded the minimum latency and processing time. **Figure 2** represent the average processing time of different set of tasks submitted to the system at different time interval. The number of tasks submitted to the fog-cloud system is represented on x-axis and the average time taken to process the tasks at the 3 layers is presented on y-axis. Also, in **Figure 3**, the number of tasks submitted to the system at different time interval is presented on x-axis and the average latency experienced by the task on the 3 layers is presented on y-axis. **Figure 4** shows the comparison of percentage QoS expectations delivered to client where AB-DRA recorded the best QoS expectations delivered to client's task. The number of task submitted to the fog-cloud system is presented on x-axis and the average QoS requirements delivered by the fog-cloud resources is presented on y-axis.

Conclusions

This paper proposed an agent based dynamic resource scheduling in Fog-Cloud computing system to cut down the over saturation of the Fog devices due to huge number of request generated by IoT devices. Agents were used to examine the resource and QoS requirements of user task, select appropriate environment and resource that can satisfy the task requirements. Based on the simulation results in the result and discussion section of this paper, it is proved that Multi Agent Based Dynamic Resource Scheduling recorded the least processing time, latency and highest percentage QoS.

References

[1] RK Naha, S Garg, A Chan and SK Battula. Deadline-based dynamic resource allocation and provisioning algorithms in fog-cloud environment. *Future Generat. Comput. Syst*. 2020; **104**, 131-41.

[2] ND Vahed, M Ghobaei-Arani and A Souri. Multiobjective virtual machine placement mechanisms using nature-inspired metaheuristic algorithms in cloud environments: A comprehensive review. *Int. J. Comm. Syst*. 2019; **32**, e4068.

[3] M Gamal, R Rizk, H Mahdi and BE Elnaghi. Osmotic bio-inspired load balancing algorithm in cloud computing. *IEEE Access* 2019; **7**, 42735-44.

[4] P Azad and NJ Navimipour. Scheduling in the cloud an energy-aware task computing using a hybrid cultural and ant colony optimization algorithm. *Int. J. Cloud Appl. Comput*. 2017; **7**, 20-40.

[5] HA Bouarara, RM Hamou, A Rahmani and A Amine. Application of meta-heuristics methods on PIR protocols over cloud storage services. *Int. J. Cloud Appl. Comput*. 2014; **4**, 1-19.

[6] PS Kumar, MS Ashok and R Subramanian. A publicly verifiable dynamic secret sharing protocol for secure and dependable data storage in cloud computing. *Int. J. Cloud Appl. Comput*. 2012; **2**, 1-25.

[7] S Yangui, P Ravindran, O Bibani, RH Glitho, NB Hadj-Alouane, MJ Morrow and PR Polakos. A platform as-a-service for hybrid cloud/fog environments. *In*: Proceedings of IEEE International Symposium on Local and Metropolitan Area Networks, Rome, Italy. 2016, p. 1-7.

[8] R Mahmud, FL Koch and R Buyya. Cloud-fog interoperability in IoT-enabled healthcare solutions. *In*: Proceedings of the 19th International Conference on Distributed Computing and Networking, Varanasi, India. 2018, p. 1-10.

[9] FG Bonomi, RA Milito, J Zhu and SK Addepalli. Fog computing and its role in the internet of things. *In*: Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, Helsinki, Finland. 2012, p. 13-6.

[10] D Puthal, MS Obaidat, P Nanda, M Prasad, SP Mohanty and AY Zomaya. Secure and sustainable load balancing of edge data centers in fog computing. *IEEE Comm. Mag.* 2018; **56**, 60-5.

[11] RK Naha, S Garg, D Georgakopoulos, PP Jayaraman, L Gao and Y Xiang. Fog computing: Survey of trends, architectures, requirements, and research directions. *IEEE Access* 2018; **6**, 47980-8009.

[12] P Hu, S Dhelim, H Ning and T Qiu. Survey on fog computing: Architecture, key technologies, applications and open issues. *J. Netw. Comput. Appl*. 2017; **98**, 27-42.

[13] D Deepa and KR Jothi. Survey on fog computing: Issues and challenges. *J. Adv. Res. Dyn. Contr. Syst.* 2020; **12**, 1301-14.

[14] SK Mishra, D Puthal, B Sahoo, PP Jayaraman, S Jun, AY Zomaya and R Ranjan. Energy-efficient VM-placement in cloud datacenter. *Sustain.Comput. Informat. Syst*. 2018; **20**, 48-55.

[15] D Puthal, B Sahoo, S Mishra and S Swain. Cloud computing features, issues, and challenges: A big picture. *In:* Proceedings of the International Conference on Computational Intelligence and Networks, Odisha, India. 2015, p. 116-23.

[16] D Puthal, R Ranjan, A Nanda, P Nanda, PP Jayaraman and AY Zomaya. Secure authentication and load balancing of distributed edge data centers. *J. Parallel Distr. Comput*. 2019; **124**, 60-9.

[17] J Biswas, F Naumann and Q Qiu. Assessing the completeness of sensor data. *In*: Proceedings of International Conference on Database Systems for Advanced Applications, Dallas, USA. 2016, p. 717-32.

[18] M Kocakulak and I Butun. An overview of wireless sensor networks towards internet of things. *In*: Proceedings of the IEEE 7th Annual Computing and Communication Workshop and Conference, Las Vegas, Nevada, USA. 2017, p. 1-3.

[19] T Choudhari, M Moh and TS Moh. Prioritized task scheduling in fog computing. *In*: Proceedings of the ACMSE 2018 Conference, Richmond, Kentucky, USA. 2018, p. 1-8.

[20] M Ketel. Fog-cloud services for IoT. *In*: Proceedings of the South East Conference, New York, USA. 2017, p. 262-4.

[21] L Bittencourt, R Immich, R Sakellariou, N Fonseca, E Madeira, M Curado, L Villas, LD Silva, C Lee and O Rana. The internet of things, fog and cloud continuum: Integration and challenges. *Internet Things* 2018; **3-4**, 134-55.

[22] H Gupta, AV Dastjerdi, SK Ghosh and R Buyya. iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments. *J. Software Pract. Ex*. 2017; **47**, 1275-96.

[23] FM Talaat, MS Saraya, AI Saleh, HA Ali and SH Ali. A load balancing and optimization strategy (LBOS) using reinforcement learning in fog computing environment. *J. Ambient Intell. Humanized Comp*. 2020; **11**, 4951-66.
[24] M Abbasi, M Yaghoobikia, M Rafiee, A Jolfaei and MR Khosravi. Efficient resource management and workload allocation in fog-cloud computing paradigm in IoT using learning classifier systems. *Comput. Comm*. 2020; **153,** 217-28.
[25] R Mahmud, SN Srirama, K Ramamohanarao and R Buyya. Profit-aware application placement for integrated fog-cloud computing environments. *J. Parallel. Distr. Com*. 2020; **135**, 177-90.
[26] S Bolettieri and R Bruno. Edge-centric resource allocation for heterogeneous IoT applications using a CoAP-based broker.*Int. J. Cloud Comput*. 2020; **9**, 28-54.
[27] J Jiang, L Tang, K Gu and W Jia. Secure computing resource allocation framework for open fog computing. *Comput. J*. 2020; **63**, 567-92.
[28] AAlarifi, F Abdelsamie and M Amoon. A fault-tolerant aware scheduling method for fog-cloud environments.*Plos One* 2019; **14**, e0223902.
[29] IZ Yakubu and C Malathy. Priority based delay time scheduling for quality of service in cloud computing networks. *In*: Proceedings of the International Conference on Emerging Trends in Information Technology and Engineering, Vellore, India. 2020, p. 1-5.
[30] IZ Yakubu, ZA Musa, L Muhammed, B Ja'afaru, F Shittu and ZI Matinja. Service level agreement violation preventive task scheduling for quality of service delivery in cloud computing environment. *Proc. Comput. Sci*. 2020; **178**, 375-85.
[31] IZ Yakubu, M Aliyu, ZA Musa, ZI Matinja and IM Adamu. Enhancing cloud performance using task scheduling strategy based on resource ranking and resource partitioning. *Int. J. Inf. Tech*. 2020; **13**, 759-66.
[32] JC Guevara and NLSD Fonseca. Task scheduling in cloud-fog computing systems. *Peer Peer Netw. Appl*. 2020; **19**, 962-77.
[33] W Li, S Cao, K Hu, J Cao and R Buyya. Blockchain-Enhanced fair task scheduling for cloud-fog-edge coordination environments: Model and algorithm. *Secur. Comm.Netw*.2021; **2021**, 5563312.
[34] RO Aburukba, T Landolsi and D Omer. A heuristic scheduling approach for fog-cloud computing environment with stationary IoT devices. *J. Netw. Comput. Appl*. 2021; **180**, 102994.
[35] G Li, Y Liu, J Wu, D Lin and S Zhao. Methods of resource scheduling based on optimized fuzzy clustering in fog computing. *Sensors* 2019; **19**, 2122.