International Journal of Computer Networks and Applications (IJCNA)
DOI: 10.22247/ijcna/2021/209987 Volume 8, Issue 5, September – October (2021)

RESEARCH ARTICLE

Learning Based Task Placement Algorithm in the IoT Fog-Cloud Environment

Shifa Manihar
Department of Computer Science and Engineering, University Institute of Technology, RGPV, Bhopal, Madhya Pradesh, India
shifa27manihar@gmail.com

Ravindra Patel
Department of Computer Applications, University Institute of Technology, RGPV, Bhopal, Madhya Pradesh, India
ravindra@rgtu.net

Sanjay Agrawal
Department of Computer Engineering and Applications, NITTTR, Bhopal, Madhya Pradesh, India
sagrawal@nitttrbpl.ac.in

Received: 03 August 2021 / Revised: 21 August 2021 / Accepted: 03 September 2021 / Published: 27 October 2021

Abstract – Task scheduling means allocating resources to the tasks in such a way that processing can be accomplished in the most optimal way possible. Here the optimal strategy means processing all the tasks in such a way that it incur the least delay, hence the least response time can be achieved by all the tasks. This becomes a major concern when dealing with the Fog computing environment. Fog have limitations on storage capacity and processing power. So all the real time applications cannot be scheduled at the Fog environment. Also it is required to allocate these resources in the most optimal way possible. So it is best suggested to schedule latency critical applications on the fog and other applications to the cloud. This paper proposes a learning based task placement algorithm (LBTP) which used supervised feed forward neural network to recognize the latency critical applications. This algorithm executes in two phases. In the first phase, the features of the tasks serve as the input to this machine learning based framework for decision making regarding whether to schedule task at the fog environment or forward it to the cloud for execution. In the second phase if the tasks scheduled at fog, then tasks are rearranged in the fog queue based on the priority to achieve the most optimal resource utilization. The simulation results were evaluated using the Matlab 8.0 and Aneka 5.0 platform. The results revealed that the proposed method LBTP recorded the best response time, waiting time and resource utilization when compared with the task scheduling at the fog only and task scheduling at the Cloud only environment. LBTP also recorded better results on horizontal scaling by raising the number of virtual machines at the fog environment.

Index Terms – Task Scheduling, Resource Allocation, Fog, Edge, Cloud, Latency, Internet of Things, Machine Learning.

1. INTRODUCTION

The basic idea of the IoT comes from the word ‘‘smartness’’ – ‘‘the capability of the device to independently acquire and relate knowledge’’ [1, 2]. Thus, we call IoT as the ‘‘things or devices and sensors’’ those are smart, uniquely addressable based on their communication protocols, and are adjustable and autonomous with in-built security mechanism [2, 3]. In the world of sensors, gadgets and devices, the computing is not restricted to the single workstation. With the advent of cloud computing paradigm, the computing is distributed over the data centers available throughout the world. With unlimited number of resources e.g. data centers and servers, the computing can be performed uninterruptedly. Cloud computing facilitates us with powerful and reliable infrastructures with the property scalability and accessibility through flexible pay-as-you-go models [4].

The bottleneck associated with cloud computing is that it has several kinds of delays associated with it such as transmission delays, networks issues like network congestion etc. which makes it unsuitable for delay sensitive applications. Although computation sensitive tasks can be scheduled at cloud easily, but for latency sensitive applications, its required to have some computing device at the vicinity of the Internet of things (IoT) devices. This gave the need for the Fog computing. Compared to the cloud, fog computing offers proximate, small-scale resources that can be instantiated dynamically [4].

Fog environments are just similar to cloud but with smaller processing and storage capabilities. Fog infrastructures find their place in between the mobile devices and the cloud in an intermediate layer [4]. Fog computing is just an extension of the cloud at the network edge. This addition to the cloud supports IoT applications to be used in the proximity of sensors, adding on to the newer benefits like fast response time and better security and privacy [5].

RESEARCH ARTICLE

Many number of fog cloud architectures has been proposed by different researchers, but mostly used architecture is the one with the three layers [6, 7, 8]. The fog layer act as the intermediate layer in between the cloud and the end devices. Fog network expands cloud services to the network edge [6]. The fog cloud architecture can be illustrated below:

1.1. Cloud Layer

It is the layer with very high computing capabilities. It consists of very large number of data centers and servers distributed throughout the world. It also act as the permanent storage for large amount of information. It is the final destination for the tasks if we do not find any computational resource for its execution locally.

1.2. Fog Layer

It is the intermediate layer which consist of collection of limited number of computing resources (access points, routers, switches, gateways, etc.) having small storage and processing capabilities. It is providing cloud computing services but at the network edge. Fog nodes interact and cooperate with the cloud. They are capable of storing, processing and transmitting the sensed data from the end devices [6].

1.3. End Devices

This layer is composed of different IoT devices such as sensors, cellphones, smart automobiles, cards, and readers [6]. These act as smart sensing devices [6].

Thus according to this architecture, user level smart devices are connected to the fog layer using wired or wireless connection technologies such as 3G, 4G, wireless LAN, ZigBee, Bluetooth, and Wi-Fi [6]. The Fog Cloud architecture can be shown in Figure 1.

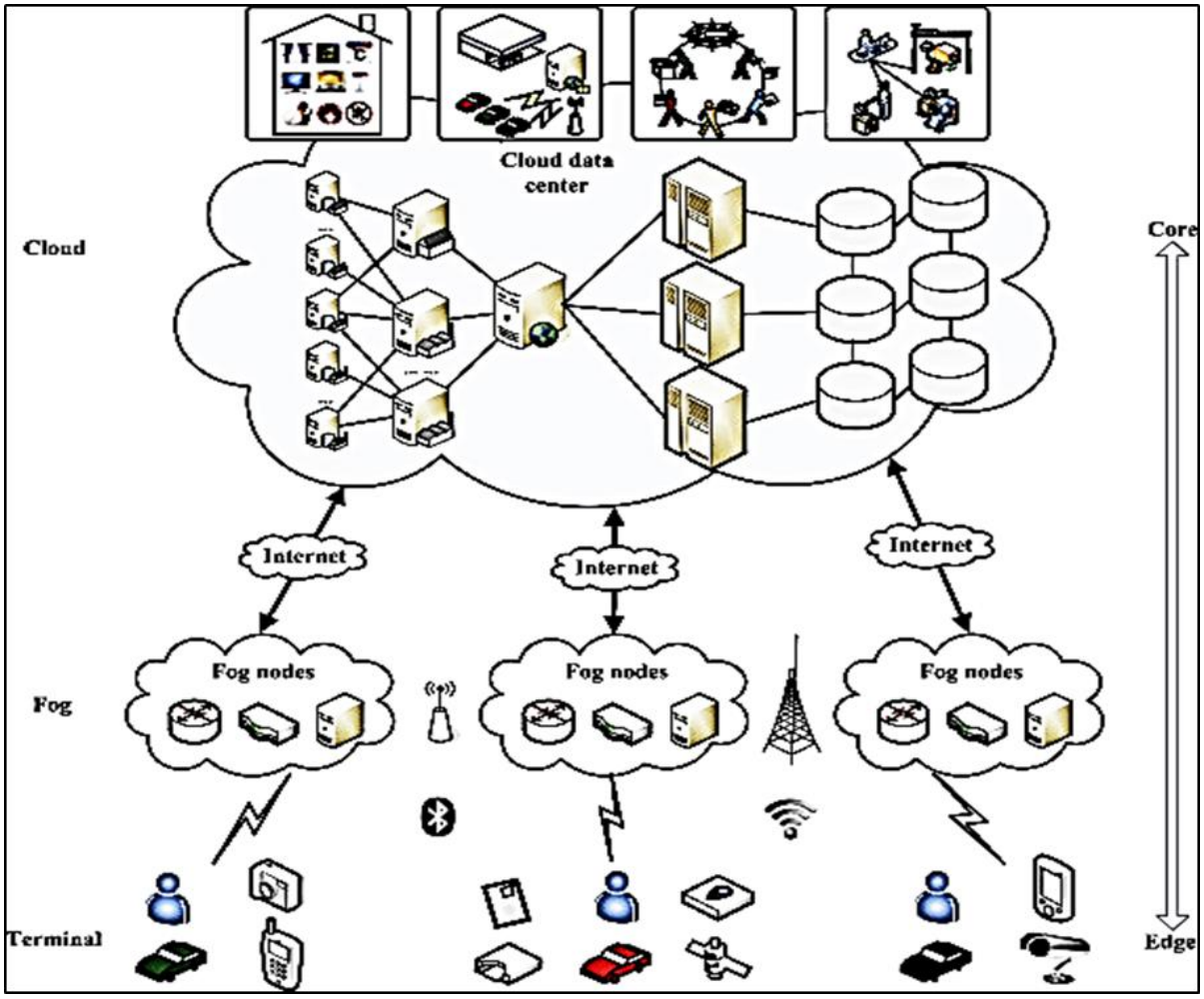

Figure 1 the Fog-Cloud Architecture [9]

RESEARCH ARTICLE

Using computations at fog level, one can save its task processing time that may incur due to propagation delay while transmission to data centers at different geographical locations. The computation is performed locally and responded without any extra delays. With the existing limitations with fog, all the applications cannot be processed at the fog. We are in need of some mechanisms so that proper collaboration can be performed in between the fog and the cloud to obtain the optimal results, basically in terms of low latency and energy consumption. The problem statement here is how to distribute the tasks among the fog and cloud efficiently and also among different virtual machines available at both the fog and cloud end. It has been discovered that there is a need to adopt a context aware automated support to decide where to store data and perform computation either at edge or at cloud, to enhance the smooth performance of the cloud edge system and reduce latency for the mission critical applications [10]. Recent researches are being carried out to develop computer based solutions to address this problem. Researchers also came up with several machine learning and meta-heuristic algorithms, hence artificial intelligence to evaluate these contexts on the basis of which the task scheduling or distribution can be performed easily and optimally. Task offloading can be carried in two scenarios:

(i) Independent tasks: The number of tasks can be offloaded to the different computing resources simultaneously and can processed in parallel [11].

(ii) Dependent tasks: The task is made up of different sub-task modules and each subtask may need data and input from some other sub-tasks and parallel offloading may not be applicable [11].

The motivation behind carrying out this research is to define the appropriate context on the basis of which the tasks can be placed between the fog and the cloud environment. The research objective of this paper is to present a learning based task scheduling algorithm which considers the features of the tasks (i.e. location, type of sensor from which data coming, etc.) and distributes tasks among the cloud and the fog using machine learning algorithm.

The rest of the paper is organized as follows: Section 2 gives short description of the similar related works carried out in the field of task scheduling. Section 3 discusses the paper contribution (LBTP method) in detail along with its mathematical formulations and the proposed algorithm. Section 4 gives detail of the simulation environment adopted for this research. Section 5 shows the observed results by implementing several scenarios mentioned in the proposed algorithm. Section 6 discusses reasons behind the better obtained results from this research. Section 7 compares the proposed algorithm (LBTP) with the related works discussed in this paper. Section 8 concludes the paper and proposes the future scope and extensions of this research.

2. RELATED WORK

Tahani Aladwani [12] has proposed a task scheduling algorithm to reduce the transfer of data between sensors and cloud by introducing fog computing at the middle. He has assigned the priorities of the healthcare data based on their importance: the high importance task, medium importance tasks and low importance tasks according to the status of the patient health conditions [12]. He used Task classifications and Virtual machines categorizations (TCVC) for the same. The MIN MAX algorithm has been used to evaluate the performance of these methods.

L. Lin et al. [13] in their work has proposed a distributed and application aware task scheduling framework called Petrel. Petrel not only used for load balancing but also ensures adaptive scheduling policy according to the type of tasks. Minh-Quang Tran et al. [14] proposed a decentralized context aware 3-tier framework for task scheduling in the fog-cloud environment. The author defined the context as location, compute and storage capacities of fog devices, and expected deadline of an application [14] and made use of these parameters to place the tasks so that maximum utilization of the virtual resources available could be accomplished at the fog orchestration node, fog neighboring nodes and the cloud. Here quality of service is considered to be the response time of the tasks.

Tejaswini Choudhari [15] in her work, presented a task scheduling algorithm which schedules tasks in the fog environment based on the priorities determined by the deadlines set the requests. F. Fellir et al. [16] has proposed a task scheduling model in the fog cloud environment which takes into account multiple agents or features of the tasks such as its priority, waiting time, its status and the number of resources required by the tasks, etc. to determine the importance of the task, and schedule it in the fog environment. This model schedules both independent and dependent tasks.

J. U. Arshed et al. [17] in his work proposed RACE (Resource Aware Cost Efficient) task scheduler which classified the applications based on their computational power requirement and the available virtual machines at the Fog cloud environment. He made use of priority mechanism in order to arrange the tasks so that minimum bandwidth can be expended and considered different scenarios. L. Yin et al. [18] has proposed container based task scheduling and resource optimization and distribution framework in the fog environment. Elarbi Badidi [19] proposed quality of service aware task placement strategies on the fog cluster. The author gave the concept of Fog broker which is a collection of several components such as the fog resource manager, task

RESEARCH ARTICLE

scheduler, fog services registration manager. M. Breitbach et al. [20] considered the features of the edge and applied data placement strategy even places data on the most suitable fog resources before the actual execution of the tasks.

Shudong Wang et al. [21] made use of the disaster genetic algorithm for adopting task scheduling algorithm in the edge-cloud scenario. The author defined an objective function as the execution time based on the time delay and termed it as punish or penalty factor or the fitness function. M. K. Hussein et al. [22] has made used of the three architecture of the cloud fog environment and deployed nature inspired Meta heuristic task scheduling namely, Ant Colony Optimization and Particle Swarm Optimization. T. Qayyum et al. [23] has proposed multilevel resource sharing frameworks for the fog cloud environments. This framework make use of end devices, underutilized end devices, regional fog nodes and cloud data centres for scheduling incoming requests and carrying out computations in order to avoid delays in response times. The authors used Ant Colony Optimization and Earliest Deadline First to achieve the necessary quality of service.

In M. Goudarzi et al. [24], an Application Placement Memetic Algorithm based on Genetic Algorithm has been proposed to reduce the weighted cost of the IoT devices. T. S. Nikoui et al. [25] in their work used cost aware task scheduling which is based on the genetic algorithm. Y. Sahni et al. [26] has proposed a multi-stage greedy adjustment (MSGA) algorithm which considers the task placement and the network flow for the purpose of scheduling the tasks. The author used the optimization function to be the completion time. M. Abbasi et al. [27] has considered a scenario of 5 fog nodes where tasks are distributed equally among them. The author used NSGA II algorithm to optimize the latency and the energy function in the fog cloud scenario.

Lindong Liu et al. [28] integrated classification algorithm with task scheduling in the fog environment. The authors deployed I-Apriori and Task Scheduling in Fog Computing to accomplish this. Mohammad Khalid Pandit et al. [29] proposed a real time task scheduling algorithm using neural network with two levels. The first level is the decision layer which decides whether the incoming task will be executed at the fog or at the cloud. If the output of the level one (feed-forward) network classifies to the fog, then reinforcement learning is deployed at level two, which assigns the tasks to the nodes or resources available at the fog layer.

Xuejing Li et al. [30] has used an intelligent adaptive task scheduling and server balancing algorithm in the mobile fog environment. The author has formulated load balancing scenario into the combinatorial problem and has used the Deep Neural Network and Reinforcement learning for scheduling the tasks at device, fog or cloud server. N. Mostafaz [31] has made use of artificial neural network with three modules namely task scheduler module, resource selector module and History analyzer module for task placement in the fog environment. M. Bhatia et al. [32] has made use of QCI based neural network approach for balancing the load and achieve minimal latency in the real time scenario in the fog environment. Fatma M. Talaat et al. [33] has proposed Effective load balancing technique which used fuzzy logic to identify the priorities of the incoming tasks. The author has made use of fuzzy rules based on the inputs such as predefined priority, deadline time and the size of the task. Based on the priority it is decided whether the task will be executed on the device, dew layer, fog layer or the cloud layer. Then load balancing is applied at these levels using the probabilistic neural network. He Li et al. [34] has proposed a task scheduling algorithm based on the reinforcement learning in which the task scheduler placed at cloud maintain a task queue and collects the state of all the tasks, which serve as input to the reinforcement neural network, which output the task to the cloud or to the best fog node manager. V. P. Kafle et al. [35] made used of offline supervised training and online unsupervised training and then applied multiple regression model in order to predict the dynamic demand adjustment and satisfy the delay constraint of the latency sensitive applications of the tasks in the IoT edge cloud environment. Y. Dong et al. [36] has proposed a joint 'cloud-edge' aware task placement algorithm which make use of exploration exploitation property of the deep reinforcement learning to find out most appropriate cloud edge set(based on their attributes) for scheduling tasks on the long term basis.

Abbreviations	Description
TCVC	Task classifications and Virtual machines categorizations
RACE	Resource Aware Cost Efficient
DGA	Disaster Genetic Algorithm
ACO	Ant Colony Optimization
PSO	Particle Swarm Intelligence
EDF	Earliest Deadline First
DAG	Directed Acyclic Graph
CGA	Cost Aware Genetic Algorithm
MSGA	Multi stage Genetic Algorithm
NSGA II	Non-dominated Sorting Genetic Algorithm II
TSFC	Task Scheduling in Fog Computing
GDP	Gradient Descent Policy
RL	Reinforcement Algorithm
DNN	Deep Neural Network
ANN	Artificial Neural Network
QCINN	Quantum computing-inspired Neural Network
PNN	Probabilistic Neural Network
DRL	Deep Reinforcement Learning

Table 1 Notations of the Classification of Task Scheduling Algorithms

RESEARCH ARTICLE

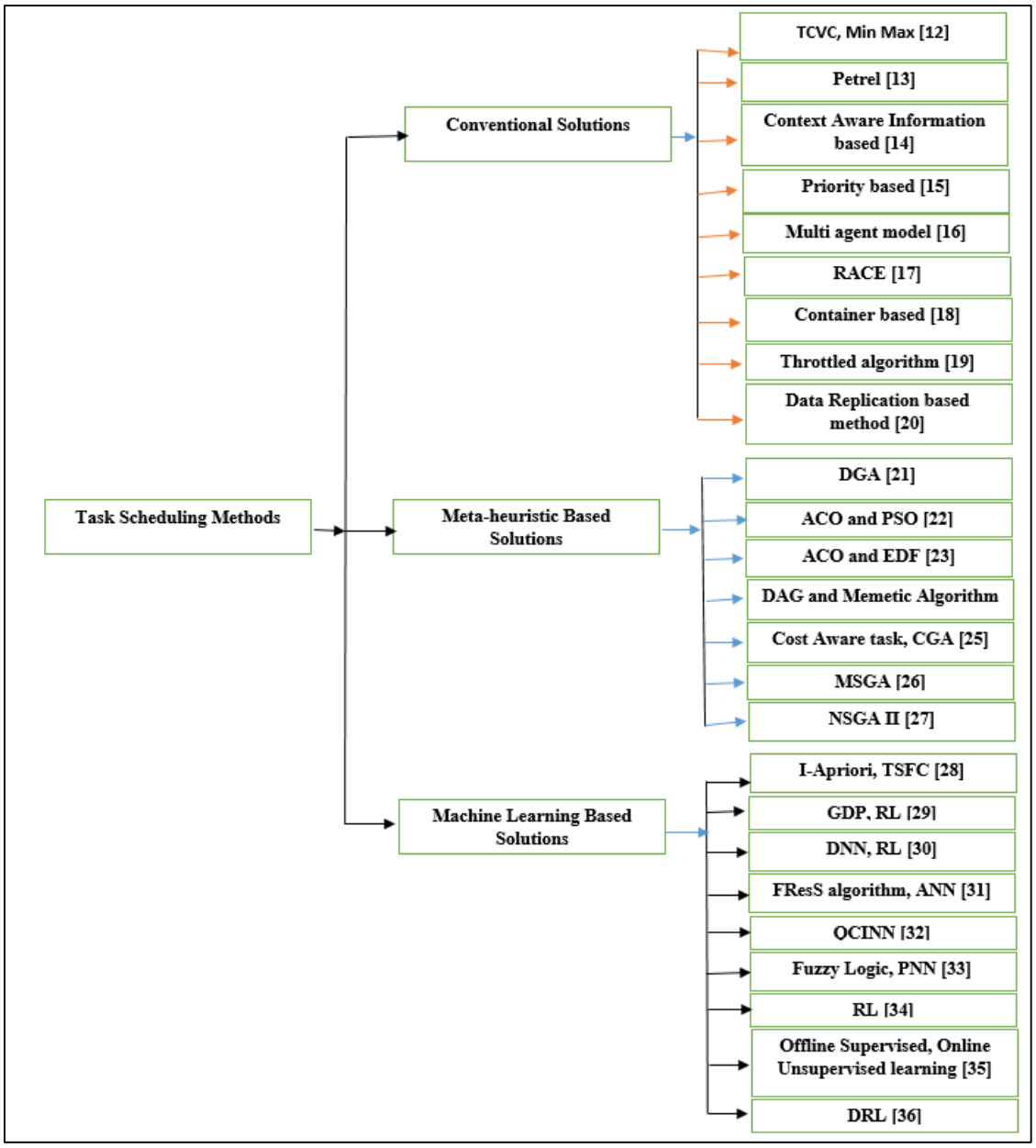

Figure 2 Classification of the Related Work Pertaining to Task Scheduling

In the related work discussed in this section, the task scheduling methods which deployed either only the conventional methods or only meta-heuristic based solutions or only the machine learning based solutions are discussed. In an approach to encourage multidisciplinary fields to obtain better results, in this research paper, we made use of machine learning (for classifying the tasks at the cloud and the fog) along with the conventional method (for arranging the task queue at fog, we used shortest job first method). This research is being carried out to define the appropriate context on the basis of which task placement and resource allocation can be done. Figure 2 gives the classification of task scheduling methods discussed in the related work above. Table 1 gives the description of the notations used in the Figure 2.

3. THE PROPOSED METHOD (LEARNING BASED TASK PLACEMENT METHOD)

The proposed model Learning based task placement (LBTP) makes use of machine learning algorithm in order to classify the mission critical applications and delay insensitive applications. Here in this paper, we have made use of supervised training at the Intelligent Fog master node (IFMN)

RESEARCH ARTICLE

to do this. The input to the feed forward neural network is the attributes of the tasks defined in terms of its location and its size. The task definition here is four tuples <lat, long, alt, tsize> as described above. At this level, the tasks are classified based on these attributes as per the training given to the neural network, hence it is decided by the IFMN that whether the task to be placed at the fog nodes or to be forwarded to the cloud for execution. If the tasks are scheduled at the fog level, then the next responsibility of the IFMN is to make decision regarding the task placement at the fog nodes (virtual machines) available under its responsibility, i.e. its worker fog nodes (WFN). The IFMN maintains the list of all the WFNs. All the information such as MIPS or available bandwidths of all WFNs are maintained. IFMN rearranges these tasks in the order of their priority on the basis of their task size and allocates tasks to these virtual machines based on these information and schedules the tasks to the most appropriate virtual machine with fast computing power and best fit available bandwidth available at the fog layer. Hence the system accomplishes two tasks: Firstly the identification of the mission critical applications and secondly to schedule these applications on the most appropriate fog node in order to achieve the minimum response time. The tasks scheduled at the fog are rearranged in the fog queue according to their size. IFMN allots the tasks to the most suitable virtual machine of WFN according to the availability of the virtual machines and the task size and sensitivity of the task. The overall architecture and the flow diagram of the proposed method (LBTP) can be shown in the Figure 3.

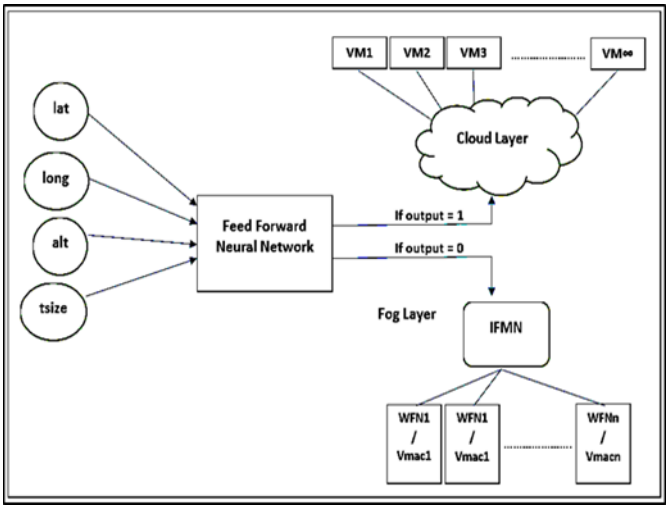

Figure 3 Architecture and Flow Diagram of LBTP

3.1. Mathematical Formulation of the Problem

The objective of this research is to design a context aware middleware in between cloud and the IoT devices, hence to apply machine learning based algorithm to make the fog intelligent. Intelligent master fog places the incoming tasks either at the fog nodes (in case of mission critical application) or at the cloud (in case of non-mission critical applications) for execution. As has been mentioned in the related work above, the different authors have tried to optimize different performance metrics. The Intelligent master fog also schedules tasks among the worker fog nodes based on the availability of the virtual machines and their capacity. Here the decision maker is the intelligent master fog node.

3.1.1. Definition 1

The $task_i$ is i^{th} incoming task and its attributes are defined as to be four tuple $<lat_i, long_i, alt_i, tsize_i>$ which serves as the context for scheduling task.Where lat_i denotes the latitudinal position of the i^{th} IoT device from which data is coming, $long_i$ denotes the longitudinal position of the i^{th} IoT device, alt_i is the altitude of the i^{th} device, and $tsize_i$ is the i^{th} task size (it is evaluated at the Intelligent master fog).

3.1.2. Assumptions

(a) This research did not consider any kind of dependency among tasks or its subtasks. Hence, only independent tasks considered.

(b) The research assumed static geographic condition with IoT devices having predefined location. The research limited to certain environment.

(c) Here fog nodes and virtual machines are used interchangeably. They refer to computing resources available at the fog layer.

(d) Running tasks cannot be pre-empted on the fog node.

(e) The terms mission critical tasks and delay sensitive tasks are used interchangeably.

3.1.3. Definition 2

The number of virtual machine (resources) at the Fog Node be m i.e. {Vmac1, Vmac2, Vmac3,……Vmacm}. The number of tasks be n, where each task has got its reference from its context of location. Hence, number of tasks can be represented as {t1, t2, …..tn}.

3.1.4. Definition 3

The bandwidth information and computing capacity of all the virtual machines (worker fog nodes) is maintained by the Intelligent Fog master node in the form of the list as shown below:

Vmac1	Vmac2	…… …… …	Vmacj	…… …… …	Vmacm
$B_{1,fog}$	$B_{2,fog}$	…… …… …	$B_{j,fog}$	…… …… …	$B_{m,fog}$
$MIPS_{1,fog}$	$MIPS_{2,fog}$	…… …… …	$MIPS_{j,fog}$	…… …… …	$MIPS_{m,fog}$

RESEARCH ARTICLE

Where MIPS stands for Million instructions per second.

3.1.5. Definition 4: Priority Decision

Each incoming task which scheduled to the fog, is assigned the priority based on the task size (tsizei) feature of the task. The tasks scheduled for execution in the fog queue are rearranged in the increasing order of their task size. The task having the smallest size has the highest priority and is executed first and can be mathematically stated in the the equation (1) as below.

i.e. $\text{Priority(task}i) \propto \frac{1}{\text{sizeof(task}_i)}$ (1)

3.1.6. Definition 5: Response time

At Fog nodes:

$$Res_i^{fog} = \mu_i^{wait} + \mu_{i,j}^{exec} + \mu_j^{netdel}$$
$$= \mu_{i,j}^{wait} + (tsize_i/B_{j,fog}) + \mu_j^{netdel} \quad (2)$$

Where, the Res_i^{fog} denotes the response times of the task ti when it is placed at the jth fog for execution. μ_i^{wait} is the waiting time spent by the task on the fog queue before its assignment to some virtual machine at fog layer. It includes the time taken for decision making by the intelligent fog master whether to schedule tasks at fog or at cloud. $\mu_{i,j}^{exec}$ is the execution time of the i^{th} task to the fog node j. μ_j^{netdel} denotes the network latency at the fog layer. $B_{j,fog}$ is the bandwidth necessary for scheduling task at the fog node j.

At Cloud:

$$Res_i^{cloud} = \mu_i^{wait} + \mu_j^{netdel} + \mu_{i,c}^{exec} + \mu^{netdelc}$$
$$= \mu_i^{wait} + \mu_j^{netdel} + (tsize_i/B_{j,c}) + \mu^{netdelc} \quad (3)$$

Where, Res_i^{cloud} denotes the response times of the task ti when it is forwarded to the cloud for execution. $B_{j,c}$ is the bandwidth necessary for scheduling the task at the cloud. $\mu^{netdelc}$ denotes the network latency at the cloud.

Total Response Time

i.e. $RT(total) = \sum_{i=1}^{n}(\text{Resfog(i)} + \text{Rescloud(i)})$ (4)

Where n = number of delay sensitive tasks.

3.1.7. Definition 6: Waiting Time

Waiting Time,

$WT(task_i) = (Res_i^{fog} + Res_i^{cloud}) - (\mu_{i,j}^{exec} + \mu_{i,c}^{exec})$ (5)

Total Waiting Time

i.e. $WT(total) = \sum_{i=1}^{n}(\text{WT(taski)})$ (6)

The summary of the notations which used in this paper are listed in the table 2.

Notations	Description
LBTP	Leaning Based Task Placement Algorithm
IFMN	Intelligent Fog master node
WFN	Worker Fog node
Vmac	Virtual Machine
t_i	i^{th} task
MIPS	Million Instructions per second
Res_i^{fog}	Response times of the task ti when it is placed at the jth fog for execution.
Res_i^{cloud}	Response times of the task ti when it is forwarded to the cloud for execution
$B_{j,fog}$	bandwidth necessary for scheduling task at the fog node j
$B_{j,c}$	bandwidth necessary for scheduling the task at the cloud
μ_i^{wait}	waiting time spent by the task on the fog queue before its assignment to some virtual machine at fog layer
$\mu_{i,j}^{exec}$	execution time of the i^{th} task to the fog node j
μ_j^{netdel}	network latency at the fog layer
$\mu^{netdelc}$	network latency at the cloud
$tsize_i$	Size of task i.
lat_i	latitudinal position of the i^{th} IoT device
$long_i$	longitudinal position of the i^{th} IoT device
alt_i	altitude of the i^{th} device
RT(total)	Total response time of all the delay sensitive tasks.
Resfog(i)	Total response time of all those mission critical tasks which executed in fog environment
Rescloud(i)	Total response time of all those mission critical tasks which executed in cloud environment
$WT(task_i)$	Waiting time for task i.
WT(total)	Total Waiting time of all mission critical tasks.

Table 2 Notations and their Description

3.2. LBTP Algorithm

Assumption: All the WFNs have the same MIPS or computing power.

1. Read the input task set (t1, t2, t3,…..tn) (n=number of incoming tasks)
2. For each task ti, do
   a. IFMN extracts the four tuple $<lat_k,long_k,alt_k,tsize_k>$ (k=1…….,n)
   b. Input < lati, longi, alti > as input to the supervised feed forward net.

RESEARCH ARTICLE

c. Train the feed-forward neural network in the supervised fashion at the IFMN which outputs either 0 or 1.

d. If (Output == 0) then

   i. Schedule the task at Fog.

   ii. Rearrange the fog queue at IFMN as follows:

   for (int i=1; i<n; i++)

   for (j=2;j<n-i;j++)

   if (sizeof($task_i$) > sizeof($task_j$)

   swap ($task_i$, $task_j$);

   iii. IFMN maintains the resource utilization chart of each worker virtual machine amd allocates WFNs based on its availability (Least busy Vmac) and MIPS.

   iv. Calculate the response time as using equation no. (2)

e. Else if (Output == 1) then

   i. Schedule the task at Cloud

   ii. Calculate the response time as mentioned in equation (3).

f. Calculate the total response time using equation (4).

g. Waiting time of tasks calculated using equation (5) and (6).

3. IFMN maintains record of resource utilization of all worker fog nodes. It is calculated from the CPU utilization graph of each WFN.

4. SIMULATION

This research was carried out in the simulation environment of Aneka platform, which is a software platform for developing cloud computing applications [37]. It is also known as Pure PaaS solution for cloud computing. We made to install Aneka 5.0 on our system which configured as per the user requirements and agreed upon SLA. Aneka is interfaced with Matlab 8.0 which contains the code for task placement algorithm of the mission critical sensitive application and delay insensitive application. We created one Aneka IFMN and 12 Aneka WFN which served as the fog environment, and have installed Aneka on the virtual machine created on the AWS cloud. This setup served as the environment of the fog-cloud.

Description of WFN: MIPS = {133,740}, BW= 3.9 GHz, Number of CPU=1, intel i7.

Description of IFMN: Storage= 32 GB, BW= 3.9 GHz, MIPS=133740.

The rest of the simulation parameters are mentioned in the Table 3 as below:

Simulation Parameters	Values
Duration of the simulation	180 minutes
The no. of iterations carried out	15
The total number of IoT devices used	150
Total number of Master fog nodes (IFMN) used	01
Total number of Worker fog nodes (WFNs) used in this research	4, 8, 12
The range of task size	2000 to 25000 MI
Probability of selecting WFN for processing the task	Equal
The number of virtual machines used in the Cloud	Unlimited

Table 3 Simulation Parameters

This proposed LBTP algorithm used the following applications: Hospital, Surveillance, Organization, Inventory, and Smart Home. And we used the following Sensor: Smoke, Temperature, Proximity, Thumb, etc. The importance of each sensor depends upon the location and task size, e.g. the task generated by smoke sensors is of latency sensitive task and has to be processed immediately; the thumb sensors of the Organization doesn't come under mission critical applications; the temperature sensor is latency critical when found in the location such as Hospital and Inventory, etc.

For determining the location attributes of the tasks coming from the devices, GPS camera has been used to obtain the latitude, longitude and altitude position of each device. We have designed a feed forward network with input as the features of the tasks and one output (decision on where to schedule tasks on the fog or to the cloud).The priority of each device has been fixed manually. For example, the data coming from healthcare devices has been assigned the highest priority. So for this task, the desired output of the feed-forward network is set to 0(meaning task to be scheduled at fog node for quick response). Whereas the data coming from the biometric sensors has been assigned low priority and the desired output is set to 1 (meaning task to be scheduled to the cloud for execution purposes). For training the feed forward network with learning rate 0.5, a test set of 200 is taken by varying the features within interval -5^o to $+5^o$ for example, the values of lat-long-alt may change due to some geographical disturbances. So we have trained the net with the all the possible dataset within range <lat+5º, lat-5º, long+5º, long-5º, alt+5º, alt-5º>. The number of tasks in the experiment starts from 30 tasks/second, increasing 15 tasks/second each time, until the number of tasks reaches 90 tasks/second. With the increase in the number of incoming tasks, the neural net trains itself better.

RESEARCH ARTICLE

For simulating the proposed model and evaluating the effectiveness of this algorithm, two different scenarios were created and tested. In the Scenario 1, which is without applying machine learning, further sub-scenarios were created regarding task scheduling at cloud only and at fog only. Scenario 2 used the machine learning to schedule tasks at both cloud and fog. We also varied the number of WFNs in order to see the performance of the proposed algorithm (LBTP). In the following sub-sections, we discuss these scenarios and their impact on the response time and resource utilization of the incoming tasks.

Scenario 1: Task Scheduling without Machine Learning

a. At Cloud only

1. Here in this scenario, all the tasks were scheduled at the AWS cloud. We do not evaluate the resource utilization in this scenario.

b. At Fog only

2. In this scenario, we have fixed to one IFMN, and varied the number of WFNs with the help of one Aneka master node and a number of worker nodes. And evaluated the response time and resource utilization of each case. The number of nodes created were 4 WFNs, 8 WFNs and 12 WFNs.

Scenario 2: Task Scheduling with Machine Learning (at both fog and cloud layers)

In this scenario, a classification algorithm (LBTP) as mentioned in Section above is run using machine learning algorithm, in order to schedule tasks between the fog and the cloud. This also considered the number of WFNs to be 4, 8, and 12 for evaluating the performance metrics such as response time and utilization of the resources.

5. SIMULATION RESULTS

The performance metrics such as response time, waiting time and resource utilization were evaluated for the mission critical applications only. The response time of each task was evaluated and summed up for this research. These parameters were not evaluated for the delay insensitive applications. The scenarios were simulated and following observations were noted:

5.1. Classification Rate

Number of tasks	Delay sensitive tasks	Delay insensitive tasks
30	13	17
45	24	21
60	37	23
75	48	27
90	51	39

Table 4 Classification of Delay Sensitive and Delay Insensitive Tasks

The classification percentage was found to be 100% with LBTP. Since the scenario was geographically limited. But this may vary when applied to geographically unstable situations.

The following was the proportion of the mission critical tasks and the mission non critical tasks considered for this research, which were classified correctly, as mentioned in Table 4.

5.2. Response Time

The response time of each task was evaluated using equation (2) and (3). For number of WFN = 4, the response times when number of tasks is equal to 30 are calculated as 24.8 s (when task scheduled at cloud only without any machine learning), 17.56 s (when tasks scheduled at fog only without any machine learning), and 3.72 s (when tasks scheduled at LBTP with machine learning). All such reading varies as observed by raising the number of tasks and the number of WFNs. The response time (in seconds) of these scenarios when WFN = 4, WFN = 8, and WFN =12 were observed and recorded as shown in the table no. 5, 6 and 7 respectively.

WFN = 4			
Number of tasks	RT@Cloudonly	RT@Fogonly	RT@LBTP
30	24.8	17.56	3.72
45	29.2	18.22	3.99
60	29.38	19.31	5.3
75	32.35	22.29	6.08
90	34.98	23.86	7.1

Table 5 Recorded Response Time when Number of WFN = 4

WFN = 8			
Number of tasks	RT@Cloudonly	RT@Fogonly	RT@LBTP
30	24.8	12.35	3.72
45	29.2	13.01	3.99
60	29.38	14.1	5.3
75	32.35	17.08	6.08
90	34.98	18.65	7.1

Table 6 Recorded Response Time when Number of WFN = 8

WFN = 12			
Number of tasks	RT@Cloudonly	RT@Fogonly	RT@LBTP
30	24.8	7.38	3.72
45	29.2	8.04	3.99
60	29.38	9.13	5.3
75	32.35	12.11	6.08
90	34.98	13.68	7.1

Table 7 Recorded Response Time when number of WFN = 12

RESEARCH ARTICLE

5.3. Waiting Time

The waiting time of each task was evaluated using equation (5) and (6). The waiting times when number of tasks is equal to 30 are calculated as 3.15 s (when task scheduled at cloud only without any machine learning), 4.7 s (when tasks scheduled at fog only without any machine learning), and 1.22 s (when tasks scheduled at LBTP with machine learning). All such reading varies as observed by raising the number of tasks and the number of WFNs. The waiting time (in seconds) of these scenarios when WFN = 4, WFN = 8, and WFN =12 were observed and recorded as shown in the table no. 8, 9 and 10 respectively.

WFN = 4			
Number of tasks	WT@Cloudonly	WT@Fogonly	WT@LBTP
30	3.15	4.7	1.22
45	9.55	7.36	1.49
60	11.73	10.45	2.8
75	16.7	15.43	3.58
90	21.33	19	4.6

Table 8 Recorded Waiting Time when Number of WFN = 4

WFN = 8			
Number of tasks	WT@Cloudonly	WT@Fogonly	WT@LBTP
30	3.15	3.03	1.22
45	9.55	5.69	1.49
60	11.73	8.78	2.8
75	16.7	13.76	3.58
90	21.33	17.33	4.6

Table 9 Recorded Waiting Time when Number of WFN = 8

WFN = 12			
Number of tasks	WT@Cloudonly	WT@Fogonly	WT@LBTP
30	3.15	1.74	1.22
45	9.55	4.4	1.49
60	11.73	7.49	2.8
75	16.7	12.47	3.58
90	21.33	16.04	4.6

Table 10 Recorded Waiting Time when Number of WFN = 12

5.4. Resource Utilization

Resource utilization was calculated by counting the number of WFNs or worker virtual machines busy for executing the incoming tasks at the fog layer. For WFN = 8 and tasks = 30, the resource utilization was found to be 63% and 50% for fog only environment and LBTP respectively. All such reading varies as observed by raising the number of tasks and the number of WFNs. The resource utilization of these scenarios when WFN = 4, WFN = 8, and WFN =12 were observed and recorded as shown in the table no. 11, 12 and 13 respectively.

WFN = 4		
Number of tasks	RU@Fogonly	RU@LBTP
30	100%	50%
45	100%	50%
60	100%	75%
75	100%	100%
90	100%	100%

Table 11 Recorded Resource Utilization when Number of WFN = 4

WFN = 8		
Number of tasks	RU@Fogonly	RU@LBTP
30	63%	50%
45	63%	63%
60	75%	63%
75	88%	63%
90	100%	88%

Table 12 Recorded Resource Utilization when Number of WFN = 8

WFN = 12		
Number of tasks	RU@Fogonly	RU@LBTP
30	42%	33%
45	42%	42%
60	50%	42%
75	58%	42%
90	67%	58%

Table 13 Recorded Resource Utilization when number of WFN = 12

In the table no. 5,6,7 and figure no.4, the terms RT@Cloudonly, RT@Fogonly and RT@LBTP stands for response times when task scheduling at cloud only, when task scheduling at fog only without any machine learning and task scheduling using LBTP with machine learning respectively. In the table no. 8, 9, 10 and figure no. 5, the terms WT@Cloudonly, WT@Fogonly and WT@LBTP stands for waiting times when task scheduling at cloud only, when task scheduling at fog only without any machine learning and task scheduling using LBTP with machine learning respectively. In the table no. 11, 12, 13 and figure no. 6, the terms RU@Cloudonly, RU@Fogonly and RU@LBTP stands for resource utilization when task scheduling at cloud only, when task scheduling at fog only without any machine learning and

RESEARCH ARTICLE

task scheduling using LBTP with machine learning respectively.

6. DISCUSSION

For the results observed and mentioned in the Section 5, the following inferences has been derived and discussed in the subsections as below:

6.1. Response Time

It was observed that when the tasks scheduled at cloud only, then mission critical tasks took the considerable amount of time. When these tasks were scheduled at fog only, then also there was no as such difference in the response time as tasks spent more time competing for the allocation of the WFNs when WFNs = 4. As the number of tasks increased, the response time increased. Increasing the number of WFNs to 8 and 12, considerably improved the response time when tasks scheduled at fog only. But the best response time was observed when the tasks used LBTP algorithm. The reason for this is that only mission critical applications executed over fog, rest forwarded to the cloud. Increasing the number of WFNs further improved the response time. The response time (in seconds) of these scenarios when WFN = 4, WFN = 8, and WFN =12 can be compared in the Figure no. 4.

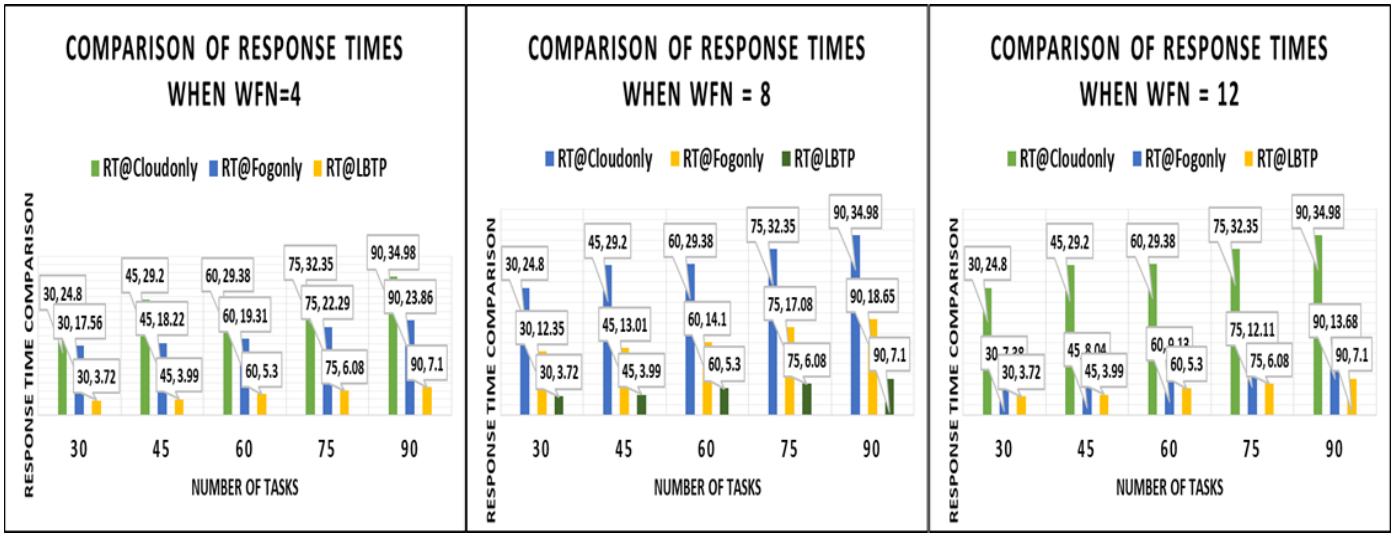

Figure 4 Comparison of Response times when WFN = 4, 8 and 12

6.2. Waiting time

The waiting time of the tasks increased as the number of tasks scheduled at fog only environment increased and when number of WFN is less because the queue waiting time for allocation of virtual machine increased. But on raising the number of WFN, this waiting time reduces. But with LBTP algorithm, the minimum waiting time was recorded as the workload is less as compared with fog only scenario. The waiting time (in seconds) of these scenarios when WFN = 4, WFN = 8, and WFN =12 can be shown and compared in figure no. 5.

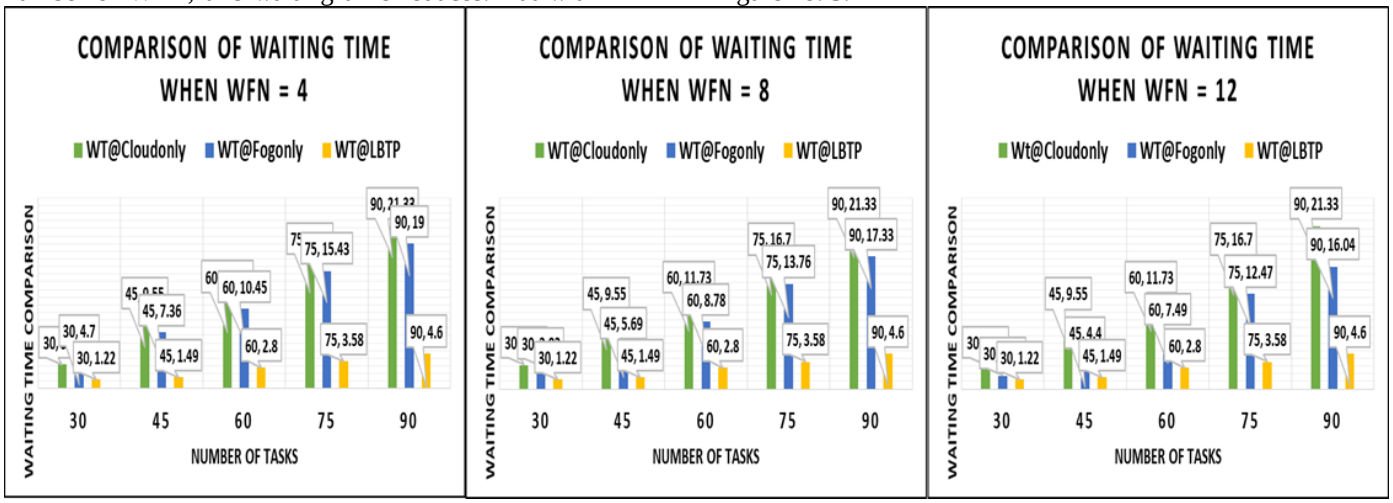

Figure 5 Comparison of Waiting Times when WFN = 4, 8 and 12

RESEARCH ARTICLE

6.3. Resource Utilization

It was observed as the number of WFN increases, the resource utilization decreases because number of free WFNs increases. The resource utilization of these scenarios when WFN = 4, WFN = 8, and WFN =12 can be shown and compared in figure no. 6.

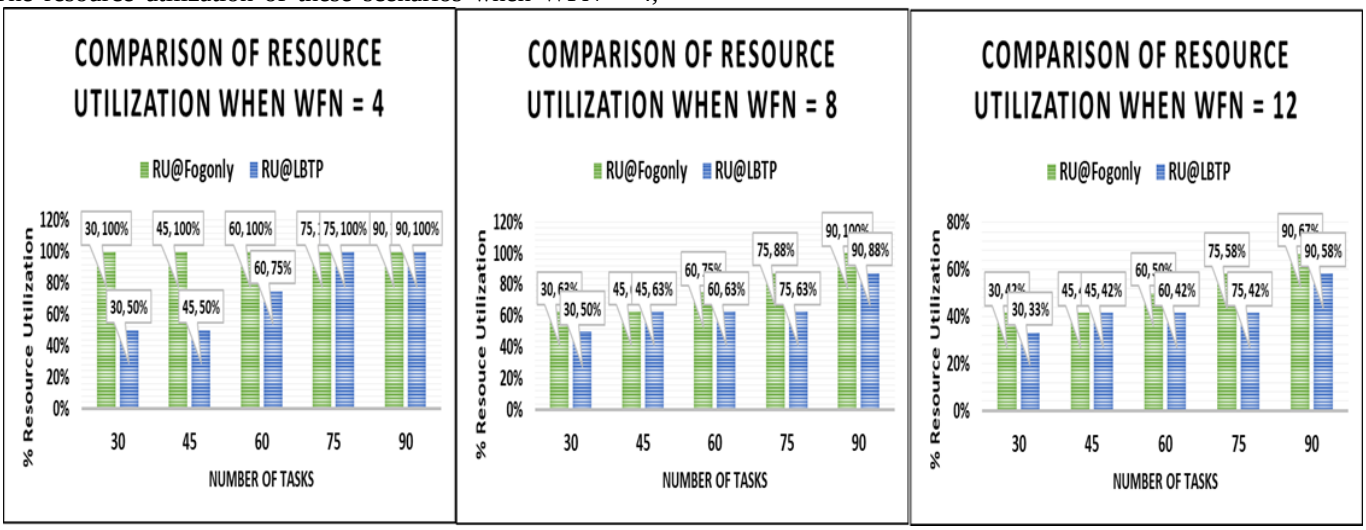

Figure 6 Comparison of Resource Utilization when WFN = 4, 8 and 12

7. COMPARISON OF LBTP WITH OTHER RELATED WORK

With reference to table 14 as shown below, it has been observed that different task scheduling algorithms different kinds of performance metrics such as makespan, energy, cost, etc. LBTP considered the performance metrics to be response time, waiting time and resource utilization.

Also different authors proposed different approaches for scheduling the tasks in the IoT fog-cloud environment. Some authors used network based solutions so others used machine learning and meta-heuristic algorithms. This work used machine learning method. The LBTP used optimization parameter to be response time, while some other task scheduling methods used different time equations, comparison of which is given in table 15. Description of the notations is given in table 16.

Task Scheduling Algorithm	Simulator	Deadline Satisfaction	Latency	Makespan	Waiting Time	Execution Time	Response Time	Energy	Resource Utilization	Packet drop	Cost	Efficiency	Throughput	Convergence Speed	Network Congestion	Dependent tasks	Independent tasks	Algorithm Complexity
[12]	Cloudsim	✗	✗	✗	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓
[13]	Petrel	✗	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗
[14]	Real world ITS	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓
[15]	CloudAnalyst	✓	✗	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✓	✗
[16]	iFogSim	✗	✓	✗	✓	✗	✗	✓	✓	✗	✓	✓	✓	✗	✗	✓	✓	✗
[17]	iFogSim	✗	✗	✗	✗	✓	✗	✗	✓	✗	✓	✗	✗	✗	✗	✗	✓	✗
[18]	QEMU, Libvirt	✓	✓	✗	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗
[19]	CloudAnalyst	✗	✓	✗	✓	✓	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗
[20]	Tasklet	✗	✓	✗	✓	✓	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗

RESEARCH ARTICLE

[21]	CloudAnalyst, Matlab	✓	✓	✗	✓	✓	✓	✗	✗	✗	✗	✓	✗	✓	✗	✗	✓	✗
[22]	Matlab	✗	✓	✗	✗	✗	✓	✗	✓	✗	✓	✓	✗	✗	✗	✗	✓	✗
[23]	OMNeT++	✓	✓	✗	✗	✗	✗	✓	✓	✓	✓	✓	✗	✗	✓	✗	✓	✗
[24]	iFogSim	✗	✗	✗	✗	✓	✗	✓	✓	✗	✓	✗	✓	✓	✗	✓	✓	✗
[25]	iFogSim	✓	✓	✗	✗	✗	✓	✗	✗	✗	✓	✓	✓	✗	✓	✗	✓	✗
[26]	Matlab	✗	✗	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✓	✓	✓	✓
[27]	Matlab	✗	✓	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗
[28]	SimGrid	✗	✗	✓	✓	✓	✓	✗	✗	✗	✗	✓	✗	✗	✗	✓	✓	✗
[29]	Matlab	✗	✓	✓	✗	✓	✗	✗	✓	✗	✓	✗	✗	✗	✗	✓	✓	✗
[30]	Python	✗	✓	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✓	✓	✓	✗
[31]	Cloudsim	✗	✓	✓	✓	✓	✓	✗	✓	✗	✓	✓	✓	✗	✗	✗	✓	✓
[32]	iFogSim	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✓	✗
[33]	iFogSim	✓	✗	✓	✗	✗	✓	✗	✓	✓	✗	✗	✗	✗	✗	✗	✓	✗
[34]	Python	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗
[35]	VirtualBox	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✓	✓	✗	✗	✗	✓	✗
[36]	Python	✗	✗	✓	✗	✓	✓	✗	✓	✗	✗	✗	✓	✗	✗	✗	✓	✗
LBTP	Aneka, Matlab	✗	✗	✗	✓	✗	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗

Table 14 Metrics Used by Different Task Scheduling Algorithms

Task Scheduling Algorithm	Time Equation
[13]	$T^{i}_{cloud} = R^{i}_{cloud} + (D_i / B_{cloud}) + RTT_{cloud}$ $T_i^{cloudlet} = R_i^{vd} + W_i^{vd} + (Di /B_{vd}) + RTT_{vd}$
[14]	$r_{\square i} = w_{\square i} + m_{\square i} + d_{\square i}$
[15]	$delay_i^T = (DL_i^T - C_i^T)$
[17]	$EFT(M_k) = \sum_{i=1}^{n}(ET(M_i) \rightarrow FD_j) + (ET(M_k)) \rightarrow FD_j)$
[18]	$Ex^j_t = \tau^{j}_{t,com} + \tau^{j}_{t,data} + \tau^{j}_{t,img}$. $Ex^c_t = \tau^{c}_{t,com} + \tau^{j}_{t,data} + \tau^{j,c}_{t,data}$,
[21]	$ECT_{i,k} = \frac{data_i}{MIPS_k}$ (k = 1, 2, . . . , Nvm; i = 1, 2, . . . , Ntsk)
[22]	$Rij = CommCost_{ij} + ST_{jc}$ $= L_{jc} + Dsize_i /BWc + L_{ij} + Dsize_i /BW_l + 1/ (\mu jc - \sum_{i \epsilon c} \lambda ji)$
[23]	$D_{xi} = D^{F}_{xi} + D_n$.
[24]	$\Gamma(Xn) = \Gamma^{exe}_{Xn} + \Gamma^{lat}_{Xn} + \Gamma^{tra}_{Xn}$
[26]	$T\ s_{i,j} = \max(avail_j , \max_{1 \le r \le Ri} (Tf_r + T\ task_{r,i}))$ $Tf_{i,j} = T\ s_{i,j} + T\ comp_{i,j}$
[29]	$\sum_{i=1,k=k}^{n,K} (\lambda_k^i + C_k + Q_k)$
[31]	$T_{TRT} = T_{SIT} + T_{QWT} + T_{RT} + T_{SOT}$
Proposed Algorithm (LBTP)	$Res_i^{fog} = \mu_i^{wait} + \mu_{i,j}^{exec} + \mu_j^{netdel}$ $= \mu_{i,j}^{wait} + (tsize_i/B_{j,fog}) + \mu_j^{netdel}$ $Res_i^{cloud} = \mu_i^{wait} + \mu_j^{netdel} + \mu_{i,c}^{exec} + \mu^{netdelc}$ $= \mu_i^{wait} + \mu_j^{netdel} + (tsize_i/B_{j,c}) + \mu^{netdelc}$ $RT(total) = \sum_{i=1}^{n} (Resfog(i) + Rescloud(i))$

Table 15 Time Equations Used for Optimization

RESEARCH ARTICLE

Notations	Description
T^{i}_{cloud}	Completion time when task scheduled at mobile device.
R^{i}_{cloud}	Time taken by task to execute in the cloud environment.
D_i	Total Data volume (both uploading + downloading)
B_{cloud}	Bandwidth requires for task scheduling at the cloud.
RTT_{cloud}	Network delay at cloud.
$T_i^{cloudlet}$	Completion time when task scheduled at cloudlet.
R_i^{vd}	Task execution time on the cloudlet.
W_i^{vd}	Waiting time at the cloudlet.
B_{vd}	Bandwidth requires for task scheduling at the cloudlet.
RTT_{vd}	Network delay at cloudlet.
$r_{\square i}$	Response time of a task.
$w_{\square i}$	deployment time in which data and compute resources needed by the task are prepared
$m_{\square i}$	Execution time (or makespan time) in which the task actually utilizes resources on the deploying node for execution.
$d_{\square i}$	Time taken during communication.
$delay_i^T$	Maximum allowed (tolerated) delay of user request i.
DL_i^T	Deadline given by request i.
C_i^T	Current time.
$EFT(M_k)$	Expected Finish time at fog node module k.
$ET(M_i)$	Execution time of i^{th} module at fog node.
FD_j	Task scheduled at j^{th} fog device.
Ex^j_t	execution time in fog node j of task t.
$\tau^{j}_{t,com}$	computation time of task t in the fog node j.
$\tau^{j}_{t,data}$	data transmission time of task t.
$\tau^{j}_{t,img}$.	Image transmission time of the task t.
Ex^c_t	execution time of a task t that runs on the cloud
$\tau^{c}_{t,com}$	Computation time of task t in the cloud.
$\tau^{j,c}_{t,data}$	Data transmission time from fog node j to cloud.
$ECT_{i,k}$	Execution time required for each task to run on a computing resource (virtual machine).
$data_i$	Length of the task.
$MIPS_k$	Million instructions per second.
Nvm	Number of virtual machines
Ntsk	Number of tasks
Rij	Overall response time of the sensor Si workload as a result of task offloading.
$CommCost_{ij}$	Total communication cost for sensor Si offloading.
ST_{jc}	Service time of task offloading of sensor Si to fog nodes fgj.
L_{jc}	Fog latency between fog node j and the cloud.
$Dsize_i$	Data Size generated from Sensor i.
BWc	Cloud network bandwidth.
L_{ij}	Network Latency between fog node j and sensor Si.
BW_l	Local network bandwidth.
μjc	Service rate of fog node j for application class c.
D_{xi}	Computational delay for a request x_i.
D^F_{xi}	Task execution delay which is a combination of both queuing delay and the service delay.
D_n	Network delay.
Γ(Xn)	overall execution time of each candidate configuration
$\Gamma^{exe}{}_{Xn}$	Computing time of workflow's tasks based on their assigned servers.
$\Gamma^{lat}{}_{Xn}$	Latency in task offloading.
$\Gamma^{tra}{}_{Xn}$	Data transmission time between each pair of dependent tasks in each workflow.

RESEARCH ARTICLE

T $s_{i,j}$	start time of a task i executed at device j.
$avail_j$	The time when device j finishes executing any previously scheduled task.
Tf_r	Completion time for task r.
$Ttask_{r,i}$	time taken to transfer data from predecessor task r to current task i.
$Tf_{i,j}$	finish time of a task i executed at device j.
T $comp_{i,j}$	the time to compute task i
λ_k^i	Compute latency of task I at resource k.
C_k	Communication costs.
Q_k	Queuing delay at resource R_k.
T_{TRT}	Total Run Time.
T_{SIT}	Stage In Time.
T_{QWT}	Queue Wait Time.
T_{RT}	Run Time.
T_{SOT}	Stage Out Time.

Table 16 Notations Describing the Time Equation

8. CONCLUSION AND FUTURE WORK

The proposed algorithm (LBTP) made use of machine learning and the results revealed that this is an effective algorithm for task scheduling. The performance metrics were evaluated and LBTP showed better results in different scenarios when compared with task scheduling without machine learning. The performance metrics considered were total response time, waiting time of the mission critical tasks and resource utilization of the fog environment. For the number of WFN = 4,8 and 12, the LBTP showed better results in terms of response time and waiting time, when compared with task scheduling at the fog only and the task scheduling at the cloud only scenarios. For WFN = 4, the resource utilization was observed to be better when compared with the resource utilization when WFN = 8 and WFN = 12 using LBTP. Hence, LBTP performance degrades in terms of resource utilization on increasing the number of WFNs. Also, task scheduling at Fog only scenario without machine learning, showed better resource utilization than the proposed method LBTP. Hence it was observed that there need to have trade-off between the response time, waiting time and resource utilization. Resource utilization is an important factor because idle virtual machines also incurs energy expenditure and contributes to the cost. For attaining the better response time, the resource utilization cannot be compromised. So there should be limited WFNs, and LBTP performs better even with small number of WFNs.

This research is carried out only for limited number of parameters, whereas the other related works measured and tried to optimize other parameters as well, which can be enlisted in the table 14. So for the future work, we will explore the following areas:

(a) We try to optimize the other parameters also as mentioned in the table 14.

(b) Also, this research was carried out in the limited geographical environment with limited IoT devices with predefined known location.

(c) We will try to evaluate this algorithm for remote uncertain geographic locations and considering some other features of the task and test results with different machine learning algorithms.

(d) In this research, we have assumed uniform computing capacity of all worker virtual machines, in the next research, we will try to simulate task scheduling algorithms on worker virtual machines with varying computing power.

REFERENCES

[1] E. Ahmed, I. Yaqoob, A. Gani, M. Imran, and M. Guizani, ‘‘Internet-ofThings-based smart environments: State of the art, taxonomy, and open research challenges,’’ IEEE Wireless Commun., vol. 23, no. 5, pp. 10–16, Oct. 2016.

[2] K. Shafique, B. A. Khawaja, F. Sabir, S. Qazi and M. Mustaqim, "Internet of Things (IoT) for Next-Generation Smart Systems: A Review of Current Challenges, Future Trends and Prospects for Emerging 5G-IoT Scenarios," in IEEE Access, vol. 8, pp. 23022-23040, 2020, doi: 10.1109/ACCESS.2020.2970118.

[3] Co-Operation With the Working Group RFID of the ETP EPOSS, Internet of Things in 2020, Roadmap for the Future, Version 1.1, INFSO D.4 Networked Enterprise RFID INFSO G.2 Micro Nanosystems, May 2008.

[4] Gedeon, J., Jens Heuschkel, L. Wang and M. Mühlhäuser. “Fog Computing: Current Research and Future Challenges.” (2018).

[5] S. Dustdar, C. Avasalcai and I. Murturi, "Invited Paper: Edge and Fog Computing: Vision and Research Challenges," 2019 IEEE International Conference on Service-Oriented System Engineering (SOSE), San Francisco, CA, USA, 2019, pp. 96-9609, doi: 10.1109/SOSE.2019.00023.

[6] Kashani, M. H., Ahmad Ahmadzadeh and Ebrahim Mahdipour. “Load balancing mechanisms in fog computing: A systematic review.” ArXiv abs/2011.14706 (2020): n. pag.

[7] M. Rahimi, M. Songhorabadi, and M. H. Kashani, "Fog-based smart homes: A systematic review," Journal of Network and Computer Applications, vol. 153, p. 102531, 2020/03/01/ 2020.

[8] O. C. A. W. Group, "OpenFog reference architecture for fog computing," OPFRA001, vol. 20817, p. 162, 2017.

RESEARCH ARTICLE

[9] Pengfei Hu, Sahraoui Dhelim, Huansheng Ning, Tie Qiu, "Survey on fog computing: architecture, key technologies, applications and open issues",Journal of Network and Computer Applications, Volume 98, 2017, Pages 27-42, ISSN 1084-8045, https://doi.org/10.1016/j.jnca.2017.09.002.

[10] L. I. Carvalho, D. M. A. da Silva and R. C. Sofia, "Leveraging Context-awareness to Better Support the IoT Cloud-Edge Continuum," 2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC), Paris, France, 2020, pp. 356-359, doi: 10.1109/FMEC49853.2020.9144760.

[11] Xuan-Qui Pham, Nguyen Doan Man, Nguyen Dao Tan Tri, Ngo Quang Thai, and Eui-Nam Huh. "A Cost- and Performance-Effective Approach for Task Scheduling Based on Collaboration between Cloud and Fog Computing." International Journal of Distributed Sensor Networks, (November 2017). https://doi.org/10.1177/1550147717742073.

[12] Tahani Aladwani, "Scheduling IoT Healthcare Tasks in Fog Computing Based on their Importance", Procedia Computer Science, Volume 163, 2019, Pages 560-569, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2019.12.138.

[13] L. Lin, P. Li, J. Xiong and M. Lin, "Distributed and Application-Aware Task Scheduling in Edge-Clouds," 2018 14th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN), Shenyang, China, 2018, pp. 165-170, doi: 10.1109/MSN.2018.000-1.

[14] Minh-Quang Tran, Duy Tai Nguyen, Van An Le, Duc Hai Nguyen, Tran Vu Pham, "Task Placement on Fog Computing Made Efficient for IoT Application Provision", Wireless Communications and Mobile Computing, vol. 2019, Article ID 6215454, 17 pages, 2019. https://doi.org/10.1155/2019/6215454.

[15] Choudhari, Tejaswini, "Prioritized Task Scheduling In Fog Computing" (2018). Master's Projects.581, DOI: https://doi.org/10.31979/etd.shqa-fdp6, https://scholarworks.sjsu.edu/etd_projects/581.

[16] F. Fellir, A. El Attar, K. Nafil and L. Chung, "A multi-Agent based model for task scheduling in cloud-fog computing platform," 2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT), Doha, Qatar, 2020, pp. 377-382, doi: 10.1109/ICIoT48696.2020.9089625.

[17] J. U. Arshed and M. Ahmed, "RACE: Resource Aware Cost-Efficient Scheduler for Cloud Fog Environment," in IEEE Access, doi: 10.1109/ACCESS.2021.3068817.

[18] L. Yin, J. Luo and H. Luo, "Tasks Scheduling and Resource Allocation in Fog Computing Based on Containers for Smart Manufacturing," in IEEE Transactions on Industrial Informatics, vol. 14, no. 10, pp. 4712-4721, Oct. 2018, doi: 10.1109/TII.2018.2851241.

[19] Elarbi Badidi, "QoS-Aware Placement of Tasks on a Fog Cluster in an Edge Computing Environment", Journal of Ubiquitous Systems & Pervasive Networks, Volume 13, No. 1 (2020) pp. 11-19. doi: 10.5383/JUSPN.13.01.002.

[20] M. Breitbach, D. Schäfer, J. Edinger and C. Becker, "Context-Aware Data and Task Placement in Edge Computing Environments," 2019 IEEE International Conference on Pervasive Computing and Communications (PerCom, Kyoto, Japan, 2019, pp. 1-10, doi: 10.1109/PERCOM.2019.8767386.

[21] Shudong Wang, Yanqing Li, Shanchen Pang, Qinghua Lu, Shuyu Wang, Jianli Zhao, "A Task Scheduling Strategy in Edge-Cloud Collaborative Scenario Based on Deadline", Scientific Programming, vol. 2020, Article ID 3967847, 9 pages, 2020. https://doi.org/10.1155/2020/3967847.

[22] M. K. Hussein and M. H. Mousa, "Efficient Task Offloading for IoT-Based Applications in Fog Computing Using Ant Colony Optimization," in IEEE Access, vol. 8, pp. 37191-37201, 2020, doi: 10.1109/ACCESS.2020.2975741.

[23] T. Qayyum, Z. Trabelsi, A. W. Malik and K. Hayawi, "Multi-Level Resource Sharing Framework Using Collaborative Fog Environment for Smart Cities," in IEEE Access, vol. 9, pp. 21859-21869, 2021, doi: 10.1109/ACCESS.2021.3054420.

[24] M. Goudarzi, H. Wu, M. Palaniswami and R. Buyya, "An Application Placement Technique for Concurrent IoT Applications in Edge and Fog Computing Environments," in IEEE Transactions on Mobile Computing, vol. 20, no. 4, pp. 1298-1311, 1 April 2021, doi: 10.1109/TMC.2020.2967041.

[25] T. S. Nikoui, A. Balador, A. M. Rahmani and Z. Bakhshi, "Cost-Aware Task Scheduling in Fog-Cloud Environment," 2020 CSI/CPSSI International Symposium on Real-Time and Embedded Systems and Technologies (RTEST), Tehran, Iran, 2020, pp. 1-8, doi: 10.1109/RTEST49666.2020.9140118.

[26] Y. Sahni, J. Cao and L. Yang, "Data-Aware Task Allocation for Achieving Low Latency in Collaborative Edge Computing," in IEEE Internet of Things Journal, vol. 6, no. 2, pp. 3512-3524, April 2019, doi: 10.1109/JIOT.2018.2886757.

[27] Abbasi, M., Mohammadi Pasand, E. & Khosravi, M.R., "Workload Allocation in IoT-Fog-Cloud Architecture Using a Multi-Objective Genetic Algorithm" J Grid Computing 18, 43–56 (2020). https://doi.org/10.1007/s10723-020-09507-1.

[28] Lindong Liu, Deyu Qi, Naqin Zhou, Yilin Wu, "A Task Scheduling Algorithm Based on Classification Mining in Fog Computing Environment", Wireless Communications and Mobile Computing, vol. 2018, Article ID 2102348, 11 pages, 2018. https://doi.org/10.1155/2018/2102348.

[29] Mohammad Khalid Pandit, Roohie Naaz Mir, Mohammad Ahsan Chishti, "Adaptive task scheduling in IoT using reinforcement learning", International Journal of Intelligent Computing and Cybernetics, Vol. 13 No. 3, pp. 261-282, 2020. https://doi.org/10.1108/IJICC-03-2020-0021.

[30] Xuejing Li, Yajuan Qin, Huachun Zhou, Du Chen, Shujie Yang, Zhewei Zhang, "An Intelligent Adaptive Algorithm for Servers Balancing and Tasks Scheduling over Mobile Fog Computing Networks", Wireless Communications and Mobile Computing, vol. 2020, Article ID 8863865, 16 pages, 2020. https://doi.org/10.1155/2020/8863865.

[31] N. Mostafaz "Resource Selection Service Based on Neural Network in Fog Environment", Advances in Science, Technology and Engineering Systems Journal, vol. 5, no. 1, pp. 408-417 (2020).

[32] Bhatia, M., Sood, S.K. & Kaur, S. Quantumized approach of load scheduling in fog computing environment for IoT applications. Computing 102, 1097–1115 (2020). https://doi.org/10.1007/s00607-019-00786-5.

[33] Fatma M. Talaat, Shereen H. Ali, Ahmed I. Saleh, Hesham A. Ali, "Effective Load Balancing Strategy (ELBS) for Real-Time Fog Computing Environment Using Fuzzy and Probabilistic Neural Networks", Journal of Network and Systems Management (IF 2.250) Pub Date : 2019-02-06 , DOI: 10.1007/s10922-019-09490-3.

[34] He Li, Kaoru Ota, and Mianxiong Dong, "Deep Reinforcement Scheduling for Mobile Crowd sensing in Fog Computing", ACM Trans. Internet Technol. 19, 2, Article 21 (April 2019), 18 pages. DOI: https://doi.org/10.1145/3234463.

[35] V. P. Kafle and A. H. A. Muktadir, "Intelligent and Agile Control of Edge Resources for Latency-Sensitive IoT Services," in IEEE Access, vol. 8, pp. 207991-208002, 2020, doi: 10.1109/ACCESS.2020.3038439.

[36] Y. Dong, G. Xu, M. Zhang and X. Meng, "A High-Efficient Joint 'Cloud-Edge' Aware Strategy for Task Deployment and Load Balancing," in IEEE Access, vol. 9, pp. 12791-12802, 2021, doi: 10.1109/ACCESS.2021.3051672.

[37] Shifa Manihar, Tasneem Bano Rehman, Ravindra Patel and Sanjay Agrawal, "Intelligent and Scalable IoT Edge-Cloud System" International Journal of Advanced Computer Science and Applications (IJACSA), 11(8), 2020. http://dx.doi.org/10.14569/IJACSA.2020.0110846.

RESEARCH ARTICLE

Authors

Shifa Manihar is pursuing PhD in the faculty of Computer Science and Engineering from UIT RGPV Bhopal. She holds MTech in Computer Technology Application Engineering from SoIT RGPV, Bhopal. She received Gold Medal for being University topper in MTech. She Holds BE degree in Computer Science Engineering from JNCT, Bhopal. She has publication in SCI, SCOPUS and UGC care listed journals and international patents. She is UGC NET qualified faculty and has qualified the GATE examination 4 times till date. She has about 9 years of teaching and research experience. Her field of research includes Internet of Things, Cloud Computing, Fog Computing, Machine Learning, etc.

Dr. Ravindra Patel has put in an illustrious academic career spanning 16 years ever since he joined service in 2005 as lecturer at SATI Vidisha, Madhya Pradesh, India. Subsequently he joined RGPV, Bhopal, and Madhya Pradesh as Reader in the year 2007, became Associate Professor in the year 2010 and Professor of Computer Applications in the year 2013 and ever since acquired rich and diversified experience in teaching, research and administration. His educational qualifications include M.Sc. Mathematics, from Model Science College Rewa, MP. Master in Computer Applications from SATI, Vidisha, MP and PhD from Rani Durgavati University, Jabalpur, MP, India. His academic experience spans the responsibilities as Head of the Department of Computer Applications since 2008 to till date and dean faculty Computer Science and Information Technology in RGPV, Bhopal, for two years. He has successfully supervised 10 Ph.D. and 06 MTech thesis in the area of Software Engineering, Data Mining, and Computer Network. Presently, 08 Ph.D. research scholars are pursuing Ph.D. under his supervision. He reviewed several research papers and was the examiner for PhD thesis of other Universities too. He has presented and published 12 research papers in International conferences, published 38 research papers in peer reviewed international journals and 02 book chapters.

Dr. Sanjay Agrawal is working as a Professor in the Department of Computer Engineering & Applications in National Institute of Technical Teachers Training & Research (NITTTR), Bhopal, M.P., India. He is having over 28 years of Technical Teachers Training along with PG Students & had done PhD under the faculty of Computer science and Information Technology of RGPV, Bhopal. He has published more than 60 Research Papers in International Journals and Conferences. He had successfully developed NPTEL/SWAYAM Course on "Accreditation for Undergraduate Engineering Programs". He has delivered several Training Sessions, Expert/Guest Lectures, Attended Seminars and Chaired Sessions in various IEEE, Springer International Conferences. He has been acting as a Dean Research and Development. Dr. Agrawal acts as an Expert Members to Committee, i.e., Expert Member of the NBA Evaluation in various Indian institutes. Dr. Agrawal is guiding many PhD & PG students and having an open offer for the research community to work collaboratively on various research projects in India and abroad.

How to cite this article:

Shifa Manihar, Ravindra Patel, Sanjay Agrawal, "Learning Based Task Placement Algorithm in the IoT Fog-Cloud Environment", International Journal of Computer Networks and Applications (IJCNA), 8(5), PP: 549-565, 2021, DOI: 10.22247/ijcna/2021/209987.