Task Scheduling Scheme using Resource Allocation for Edge Computing Model

S.Premkumar and AN.Sigappi
Department of Computer Science & Engineering,
Faculty of Engineering & Technology,
Annamalai University, Tamilnadu, India

Abstract **- IoT has seen rapid development over the past decade in many applications that consume considerable memory, energy, computing resources and mainly impose strict time requirements. This can be fulfilled by the concept of edge computing by bringing the intelligence nearer to the data source itself for task computation and other necessary executions. The requirement of smart control and decision at each level depends on the time sensitivity of the IoT application. There is a need of mechanism in task and device management for assigning, offloading and scheduling the task among the edges. This paper proposes a scheme of computation offloading termed as Task scheduling with CPU allocation offloading Scheme (TSCAOS) and formulated for providing all-inclusive processes to derive the optimal computation offloading decision for tasks between the edge nodes and edge servers. In this work, primarily, tasks are partially offloaded with the calculation of the energy consumption by the task from the edge node to edge servers. Edge server's tasks scheduling algorithm is developed in which tasks received from the edge nodes are scheduled according to the priority level of tasks arranged in the queue with high, low and medium levels. Finally, the processes are combined with the resource allocation scheme where after the task arrangement, edge CPU utilization is calculated and servers are arranged with their utilization power. Then the resources are allocated according to CPU levels and task priorities to complete the computation. Here the proposed scheme is experimented using virtual reality application in EdgeCloudSim simulator comparing with three other existing offloading algorithms. The evaluation and analysis show that TSCAOS performs better in terms of response time, processing delay, low failed task rate and execution time.**

Index Terms - Edge computing, Edge server, IoT, Offloading computation, Partial offloading, Resource allocation, Task scheduling.

INTRODUCTION

The huge expansion of Internet-of-Things (IoT) devices has increased the demands for effective processing of low latency stream data generated near the edge of the network. By 2030, IHS Market has predicted that the number of IoT devices will proliferate to more than 125 billion [1]. The general fact is that applications developed using IoT put forth severe demands for low latency computations. For instance, Virtual Reality (VR) applications utilize head tracked systems which needs latencies less than 16 ms for achieving perceptual stableness [2]. Vehicle applications that are connected autonomously with features like traffic efficiency, warning of collisions, autonomous driving etc. possess latency requirements ranging from 10ms to 100ms [3]. This requirement of low latency computing is facilitated by edge computing with an efficient infrastructure for the IoT based applications by moving the computation near the end devices including smart home hubs (for instance., Amazon, Amazon Echo, Google Home), micro datacenters and road side units [4], [5] deployed at the edge of the network near the IoT devices. Edge computing [2] provides computations and storage devices at the edge of the network by utilizing edge servers close to users like Cloudlet and MEC. Moreover, it has some unique features like high bandwidth, low latency and security when compared to cloud computing mode whereas the problem lies in the way of making offloading decision influenced by several factors like the characteristics of tasks, network conditions and differences in platform. The potential gains from offloading are diminished by the complexity and versatility of the task is an instance of this. Also, the offloading benefits are affected by the instability in network and if the offloading decision fails to take account of variations in these factors, poor performance results. Therefore, the ways to make offloading decision in accordance with different factors and their unpredictable variations for achieving the desired objectives is still under research.

The process of computation offloading deals with offloading the computation tasks to Edge Computing using certain procedures for transmission, remote execution and sending back the results. The scheme of offloading involves the following key components – i) Task partition, ii) Placement of tasks, iii) Resource allocation. (i)Task partition: If a task can be divided, then it is divided optimally before the process of offloading commences. If not, then the entire task needs to be offloaded to an EC server. (ii) Task placement (offloading decision): Here, the process of deciding which EC server is to be executing the whole or divided task and it is also to be noted that the task can also be executed locally. (iii) Resource allocation: It deals with determining the number of resources (computing, communication and energy) required to be allocated to tasks. These key components have a key role in the modeling of computation offloading process. The purpose of offloading modeling lies in finding an optimal solution for these components (the way of partitioning the task, the type of decision, and the number of resources to be allocated) so as to optimize the offloading performance.

In our proposed work, for decreasing the total cost of the whole system with respect to energy consumption and network latency, an effective computation offloading model is constructed as an integer optimization problem for a Virtual reality (VR) based multi-level multi-player multi-task edge-cloud computing system. Additionally, the environment has concerned a single cloud computing connected with the multi edge computing server through an intelligent core network. And it is based on Software Defined Networking (SDN) technology for providing more resources if the number of VR devices expands and the resources of edge server becomes insufficient. An effective algorithm termed as Task scheduling with CPU allocation offloading Scheme (TSCAOS) is formulated for providing all-inclusive processes for deriving the optimal offloading decision in computation. This scheme is detailed in the upcoming sections. The proposed scheme is developed and implemented in VR application scenario that is simulated in Edgecloudsim and evaluated with edge metrices.

RELATED WORKS

Several existing works have focused on task scheduling, computation offloading problem and overloaded CPU in the edge computing environment with multi user or multi IoT devices combined in edge computing scenarios. The major problem of edge computing lies in the way offloading decisions are made and it is influenced by several factors like characteristics of task, network conditions and platform differences. A review on the offloading policies, task scheduling approaches and CPU allocation with energy, priority and server service availability constraints is given.

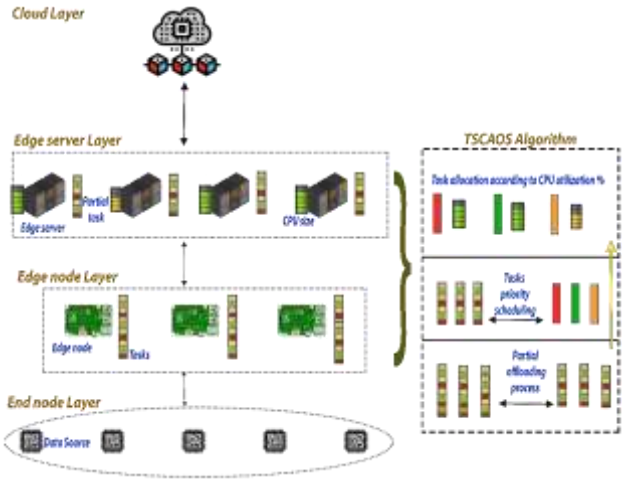


FIGURE 1
THE PROPOSED ALGORITHM WORKING ON THE EDGE COMPUTING SCENARIO.

- **Offloading Approaches:** Different locations such as IoT devices, edge server or cloud can be used for computation tasks. The offloading destination relies on the exchange between various objectives and impact factors [6]. The most crucial step in the offloading process is to find whether to offload. The offloading decision is influenced by various transient factors like the devices, network, services and user which tend to change often. The edge servers like Fog nodes [5], Cloudlets [7] or MEC servers are used by the IoT devices to offload the computation tasks and that gets further processing and hence, cost and latency can be reduced effectively. The authors in [8] utilized generic edge servers like MEC or cloudlets for offloading Deep Neural Network (DNN) computation tasks from IoT devices. Machine Learning (ML) algorithms are used by the embedded devices in offloading scenarios for the further improvement of the app performance. Two categories of processing such as parallel processing and program partitioning are capable of improving the offloading performance. However, some tasks cannot be partitioned into multiple tasks for the parallel execution as some programs would be highly interconnected or comparatively small. Therefore, local device is preferred for such execution or it is offloaded as a whole in the server. to sum up with, the offloading strategies can be categorized as full offloading and partial offloading.
  Partial offloading deals with partitioning the program into several tasks and offloading them to the server. The strategies involved in partial offloading is classified as single and multiple users. Authors in [9] introduced wireless parameter and utilized component dependency graphs (CDG) for optimizing the task scheduling and offloading decisions by proposing mobile application based joint scheduling and computation offloading (JSCO). This method completely reduced the time taken for execution by acquiring the benefits of mobile and cloud parallel processing. [10] described a general offloading scheduling problem for video applications in real time based on energy saving methodology. An adaptive scheduling method is proposed based on the dynamic wireless network conditions. [12] explained about single Edge and an Edge computing server and the edge decides on whether to offload a partial computation task to the EC server.[13], [14] extended the above mentioned scenario to multiple users computation offloading in the multi-channel wireless contention scenario.[11] and [14] discussed offloading with parallel components. where, an edge divides the task and offloads part of the components to an EC server. $\lambda(0 \leq \lambda \leq 1)$ was defined as the proportion of locally executed amount of bits to the total input data bits, i.e., $(1 - \lambda)$ is the ratio of offloaded bits to the total bits. Complete granularity is assumed in data partition for simplifying the analysis [16] and the task is divided into components of any size. The important factor to optimize the offloading performance is to determine the optimal λ.[17] performed offloading based on complete granularity in partitioning of data and simultaneous execution of components. Through this survey, it is concluded that energy consumption with task management has to be considered for the partial offloading in edge nodes to edge server level architecture.
- **Task Scheduling in Edge computing:** Internet services has attained a rapidity in its growth that led to versatility in computation intensive applications like virtual reality. There is a need for the users to decide about offloading their

applications to central/ edge clouds as high computing costs results when the applications are executed in end devices and high transmission cost incurs when the applications are executed in central/edge clouds. Task offloading involves mapping the users' tasks to appropriate resources in the form of Virtual Machines (VMs) for execution. In Cloud Computing (CC), users alone are considered to decide whether to offload their tasks or not as there is only one data center usually whereas in Edge Computing (EC), users have an option to consider where to offload their tasks as many edge clouds are present. For improving QoS in CC and EC, task scheduling becomes a critical issue [18].

Authors in [19] proposed a Genetic Algorithm (GA) oriented computation task offloading methodology for Mobile Cloud Computing (MCC). Energy consumption and execution time are optimized for arriving at robust offloading in mobile devices. The proposed approach produces near optimal solutions which is demonstrated by numerical results with near linear algorithmic complexity. GA based methodology was framed in [19] for reducing the communication and computation energy for elastic applications by determining the optimum offloading solution through augmented execution of mobile applications. The authors devised an optimization problem by lessening the cost function which is the combination of communication and computation energy. [20] states a GA for the placement of service in FC and surveyed about mapping data streams over fog nodes and given an optimized model. An extensible heuristic over GA is applied for tackling this model. The stability and performance of the proposed heuristic approach was verified using the experimental results. [21] satisfied the constraints in completion time and minimized the energy consumption by suggesting a task caching and migration mechanism in MEC based on GA. A fine-grained task partitioning migration model was applied for transforming the users' tasks into directed graphs with multiple subtasks. Further, the task caching is performed for reducing the energy consumption and delay. From the above survey, it is evident that the proposed methodology should be efficient specifically in dealing with task processing delay and energy consumption.

- **CPU Service Allocation:** The impact of computation offloading majorly relies in selecting the edge server nodes for providing service to process the tasks [22]. The edge server nodes could be edge servers or IoT based edge servers in the vicinity of cloud server [23]. A task is offloaded to any server node for execution where Edge device considers server nodes' computing capacity, available resources, distance and access technologies before arriving at the offloading decision. Authors in [24] studied reinforcement learning (RL) for allocating resources autonomously in CC, where proper initialization is framed at the early stages and convergence speedups are also applied in the learning stages. Authors [25] have chosen fog radio access networks by proposing a RL associated resource allocation algorithm. They framed the resource allocation problem as an Markov decision process (MDP) in two alternative formulations: infinite and finite-horizon MDP. Authors [26] infused greedy Q-learning for allocating the resources by proposing a RL approach in EC. Authors in [27] developed a GA for scheduling tasks in CC by distributing the loads among VMs efficiently for optimizing the complete response time using greedy based methods and First Come First Served approach. Authors [28] created a job scheduling model in CC based on multi objective GA for minimizing the consumption of energy and maximizing the profit of service.
- **Task management Edge computing Architecture:** The architecture has mobile edge nodes and edge servers as shown in Figure 1. The architecture considers many mobile edge nodes and edge servers which is connected in same wireless network with a task exchange. Each end device is the source generator of task and connected to the closer edge nodes where the task data is collected and processed, further offloaded to the edge server network for the task execution. If the task needs more computational power, then further sent to cloud for the execution of the task. This architecture can be considered as 4 tier architecture with end node layer, edge node layer, edge server layer and cloud server layer. Applications having computation intensive and time sensitive tasks can be handled well by four-tier architecture where the cloud and edge manage the former and latter respectively. The interface between the edge node and edge server and assigning of tasks are more focused in designing the four-tier architecture (i.e., assigning which tasks are executed at the edge node and at the edge server). The task allocation and offloading methodology between this layer is described in next section.

TASK SCHEDULING WITH CPU ALLOCATION OFFLOADING SCHEME (TSCAOS)

The Edge computing system's structure is depicted in Figure 1. The edge server is considered as a mini data center installed at a wireless access station and several edge node devices communicate with edge server. Edge node is appended to edge server that is closer and with the help of wireless channel, edge node sends the tasks. A similar MEC architecture was described by methods in [29]. Here in this working edge servers are benefited with improved computation performance and reduced energy consumption in edge devices using task offloading.

The proposed model possesses two layers: Mobile edge nodes and edge servers. Here, several task allocation methods are provided along with the specifications of how and where to handle tasks. Two models for allocation of tasks are framed and combined together in the proposed TSCAOS edge-based architecture that are (i) Partial offload process (ii) task allocation iii) CPU utilization server allocation. Scheduling and allocations in task management with respect to edge computing are discussed further in the following sections.

The decision for optimal offloading fraction of tasks for each mobile edge node is the crucial problem drawn up from an energy efficiency optimizing problem under the edge-based system. The partial offloading formula is formulated from the equation of β_m that is called as fraction of task offloading and it is detailed and derived in [29]. The derivation is based on the energy consumption of each mobile edge node with local execution and partial offloading decision mechanism. It is further optimized by the lambert function and adapted as a result β_m^* which is called as optimal fraction task offloading and it is derived in [29] to compute the optimal partial offloading equation whose equation is defined in Eq. (1) and its variable are detailed in the Table I.

$$\beta_m^* = \left\{ \frac{L_m g_m g_m^e - D_m g_m^e}{\frac{g_m^e g_m S_n}{t_{km}} + D_m g_m - D_m g_m^e} \right\} \quad (1)$$

TABLE I
PARAMETERS USED IN OFFLOAD PROCESS MODEL

Variables	Description
$T_k = \{t_{k1}, t_{k2}, t_{k3} \dots .\}$	Task
m	End edge node
t_{km}	Task modeled by each edge node
D_m	CPU cycle per bit of task
S_m	Task data size
L_m	Deadline requirement
X_m	Task Priority value
β_m	Task offloading fraction
$t_{km} = \{D_m, S_m, L_m, X_m\}$	Collection of task t by edge node m
O_m	Edge node offloading data size calculated using $O_m = S_m \beta_m$
P_{cm}	Edge node power consumption
β_m^*	Optimal fraction task offloading
g_m^e	Edge server computing capability
k_m	Transmission range of edge node
g_m	Edge node computing capability

The variables in Equation (1) are briefly explained in Table I. From the above discussion, it can be inferred that optimal computation offloading fraction is closely related to the task's data size and latency requirement.

Pseudocode for Module I: Mobile Edge Node Layer – Task offloading Algorithm

1: ***for*** *each edge node m*
2: ***for*** *each* S_x *in edge m*
3: *Calculate* β_m *;*
4: *Offload* $S_x * \beta_m$ *task to edge server;*
5: *Execute* $S_x * (1 - \beta_m)$ *task at local node*
6: ***end for***
7: ***end for***

The TSCAOS mechanism is composed of three modules: Module I in mobile edge node layer for partial offloading, Module II in edge layer with task allocation and Module III in edge layer with resource allocation The optimal offloading decision model calculates the offloading fraction with each node generating its task randomly within the mobile edge device as explained in the pseudo code for Module I. Initially, the optimal computation offloading fractions are calculated by the mobile edge devices for each batch of task in Module I. Fraction β_m^* and $(1 - \beta_m^*)$ of tasks are offloaded to the edge server and executed in local edge node within this step respectively.

TABLE II
PARAMETERS USED IN SCHEDULING ALGORITHM

Variables	Description
Z_{hg}	High priority queue
Z_{md}	Medium priority queue
Z_{lw}	Low priority queue
TH_1	Threshold 1
TH_2	Threshold 2
wt_x	Waiting time for each task *x*
x	Task *x*
LT_x	Latency requirement
U_x	Current time
$wt_x = (LT_x - U_x)$	Formula for waiting time for each task
$E_{s,x}$	Estimated service time for task
SUB_{CT}	Subscribed priority catalog value, Level High = 1 , Level Medium = 2, Level Low = 3

Pseudocode for Module II: Edge Layer - Priority offloading Algorithm

1: ***for*** *each task in edge server queue*
2: *Task manager in edge server check maximum waiting time* wt_x *by (2);*
3: ***if*** $wt_x \leq E_{s,x}$,***then***
4: *Place task* $_x$ *in* Z_{hg}*;*
5: ***else if*** $E_{s,x} + TH_1 \leq wt_x \leq E_{s,x} + TH_2$.,***then***
6: ***if*** SUB_{CT} *==1,* ***then***
7: *Place task* $_x$ *in* Z_{hg} *;*
8: ***if*** SUB_{CT} *==2,* ***then***
9: *Place task* $_x$ *in* Z_{md} *;*
10: ***if*** SUB_{CT} *==3,* ***then***
11: *Place task* $_x$ *in* Z_{lw} *;*
12: ***else if*** $wt_x > E_{s,x} + TH_2$.,***then***
13: ***if*** SUB_{CT} *==1, and is* Z_{hg} *not full,* ***then***
14: *Place* $E_{s,x}$ *in* Z_{hg}*;*
15: *else*
16: *Place* $E_{s,x}$ *in* Z_{md}*;*
17: ***else if*** SUB_{CT} *==2, and is* Z_{md} *not full,* ***then***
18: *Place* $E_{s,x}$ *in* Z_{md}*;*
19: *else*
20: *Place* $E_{s,x}$ *in* Z_{lw}*;*
21: ***else if*** SUB_{CT} *==3,* ***then***
22: *Place* $E_{s,x}$ *in* Z_{lw}*;*
23: ***end if***
24: ***end for***

The existing priority-based scheduling algorithm in edge server layer is enhanced as offloading tasks from mobile users with varying latency needs that required to be satisfied with respect to research in [30]. Table II presents the parameters and the formulas used in prioritized scheduling model. In Module II (edge server layer), all the offloading tasks from mobile edge node are processed and ordered by edge server in a priority queue depending on the subscription catalogues, latency requirement and priority levels.

With respect to latency requirement and subscription catalogues, task is placed in one of three queues. Thresholds are used for rearranging the tasks in accordance to their latency requirements and subscribed priority levels. Based on the experiments, the threshold values TH_1 and TH_2 are set such that TH_1 is less than TH_2 .Therefore, the maximum waiting time, wt_x is used to check against estimated service timeTH_2,TH_1 and $E_{s,x}$. The tasks having severe latency requirement and loose latency requirement will be inserted into high priority and medium or low priority queues respectively. Therefore, tasks having high priority are processed first.

Module II is executed for those tasks offloading to edge server. All offloading requests from mobile edge devices are processed here with maximum waiting time wt_x and compared against estimated execution time$E_{s,x}$. If $wt_x$is less than or equal to, $E_{s,x}$ the task is placed in to highest priority queue Z_{hg} irrespective of subscription catalogues. The value of withis added $E_{s,x}$ threshold value TH_1 and TH_2 individually. And if the value of wt_x lies between them, then the task is placed into the queue based on its subscription catalogues. Finally, wt_x plus $E_{s,x}$ is checked with threshold valueIf the corresponding queue . TH_2 has space, the task is placed in its original subscription catalogue, or else, downgrade one queue level. This model ensures that high priority tasks are executed first by the application of this algorithm.

Module III is final execution of the whole TSCAOS system where the CPU allocation according to the task of major concern. In this process module I and module II are executed at the step 3 and 4 with the calculation of the partial offloading that is executed for the task priority calculation. The tasks are placed in the Z_{hg}, Z_{md} Z_{lw} according to the priorities of the tasks categories. After the sorting of tasks in the edge server pool, $edge_{si}$ CPU utilization is calculated. Each $edge_{si}$ CPU utilization is divided into high and low utilization and if it is having same utilization then the steps of the partial offloading is repeated till the value changes. From the task queue the 3 categorical tasks are assigned where Z_{hg} is assigned to the low utilization CPU $edge_{si}$, Z_{md} is assigned to CPU $edge_{si} - 1$, Z_{lw} is assigned to CPU $edge_{si} - 2$. In this way the task is assigned to the edge server where the tasks are computed efficiently. Through this task failure is reduced, the processing speed increases and the overall delay performance is reduced.

Pseudo code for Module III: Task Allocation through edge server CPU utilization

1: ***for*** *each task* T_k *in edge server queue, where* $T_k = \{t_{k1}, t_{k2}, t_{k3} \ldots.\}$
2: *Calculate* β_m^* *by eq (1);*
3: *Partial_offload (); by module 1*
4: *Task_assignment (); by module 2*
5: ***end for***
6: Update & store task information;
7: ***for*** *all* $edge_{si}$ *in edge server pool where si = {1, 2…n}*
8: *calculate* $edge_{si}$ *CPU utilization;*
9: *Find high* $edge_{sn}$ *&* *low* $edge_{sn}$ *CPU utilization;*
10: ***if*** *the* $edge_{si}$ *with same CPU utilization*
11: *goto step 2;*
12: ***else***
13: ***for*** *each task* Z_{hg}, Z_{md}, Z_{lw} *in edge server queue;*
14: ***if*** Z_{hg} *task arrives*
15: *Find the* $edge_{si}$ *with low CPU utilization;*
16: *Assign* Z_{hg} *in* $edge_{si}$*;*
17: ***else if*** Z_{md} *task arrives*
18: *Find the* $(edge_{si} - 1)$ *with low CPU utilization;*
19: *Assign* Z_{md} *in* $edge_{si}$*;*
20: ***else if*** Z_{lw} *task arrives;*
21: *Find the* $(edge_{si} - 2)$ *with low CPU utilization;*
22: *Assign* Z_{lw} *in* $edge_{si}$*;*
23: ***end if***
24: ***end for***
25: ***end if***
26: ***end for***

TABLE III
COMPARISON OF DIFFERENT SCHEMES WITH PROPOSED SCHEME

Model used for Comparison	Schemes applied at Edge nodes	Schemes applied at Edge servers
All local processed	First Come First Served (FCFS)	Not Applicable
All Edge processed	Not Applicable	Earliest Deadline First (EDF)
Partial offload with Task allocation (POTA)	Partial offloading Scheme	Task Scheduling Scheme
TSCAOS	Partial offloading Scheme	Task Scheduling Scheme + CPU allocation Scheme

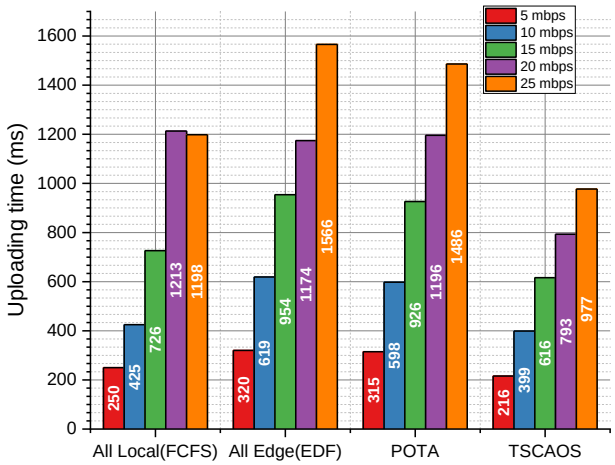
PERFORMANCE & EVALUATION

Edge computing has stepped into different applications like Virtual/Augmented Reality (VR/AR), online videos and games. In Cloud computing and Edge computing, metrics like privacy and security, cost, latency, throughput, resource utilization, energy consumption need to be evaluated for proving the effectiveness of the system and to provide Quality of Service (QoS). These metrics can be enhanced by considering issues like allocation of tasks, scheduling of jobs and offloading of tasks. For instance, designing of effective task offloading and accurate resource allocation improves privacy, security and energy consumption, latency respectively [31], [32]. Additionally, appropriate task offloading approaches enable the optimality in consumption of energy and latency [21], [33]. However, TSCAOS is proposed for increasing the efficiency of task completion and fulfilling the edge computing purpose that to compute the task near the edge nodes itself. In this work, four models as shown in Table III are developed in the simulator to check the efficiency of the proposed scheme between the edge servers.

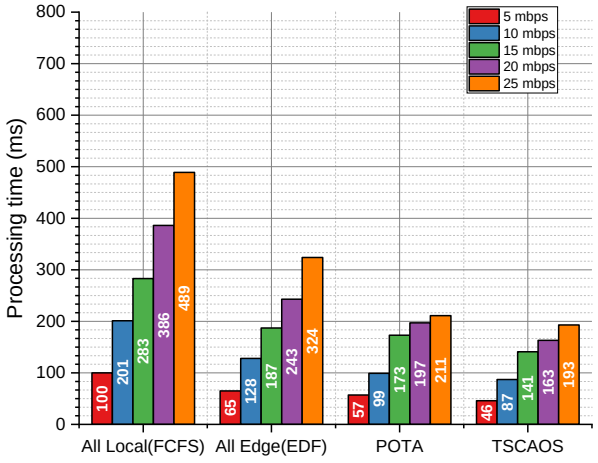
The EdgeCloudSim simulator is used for the experimental purpose to implement the four models and check the efficiency of the model. Although, EdgeCloudSim simulator simulates the mobility of devices and thereby investigates the failure rate of tasks,

virtual machine (VM) migration methods are not supported by it. Also, it considers the impact of network load and delays in transmission over the delay of service requests. The load generator facilitates the task generation dynamically with individual data size and length. For experimental purpose, the models are considered as shown in Table III that are (i) All local processed model runs locally between the edge nodes where the computation of task is locally processed and executed near the source itself and uses FCFS scheme in the edge nodes. (ii) All edge processed model, in which every task generated from the source of edge node is sent to the edge server network and executed in the edge server with the help of Earliest Deadline First (EDF) scheme. (iii) Partial offload with task allocation (POTA) model offloads the task partially from edge nodes to the edge servers. And these tasks are again scheduled based on priority using priority scheduling algorithm. (iv)TSCAOS is the proposed scheme which is developed by extending POTA model using CPU utilization scheme at the edge server network. Here, the task execution occurs in edge server network according to the CPU usage and availability. These four models are framed with the experimental parameters as shown in Table III where the scenarios are created accordingly with some application framework working with task execution.

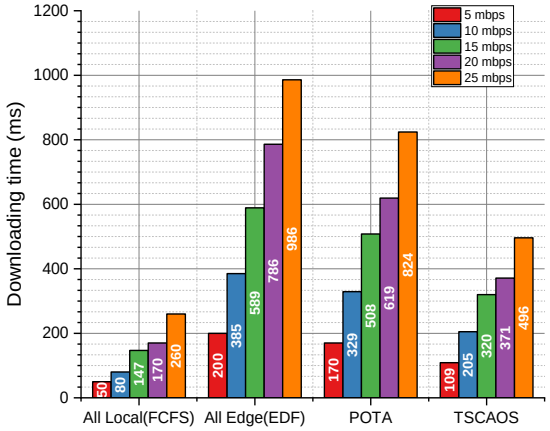
A virtual reality game application is modeled with some random simulation parameters in the EdgeCloudSim. The four models listed in Table III are deployed and evaluated through performance metrices such as response time, processing delay, total task failure rate and execution time.



(A)



(B)



(C)

FIGURE 2
(A) UPLOADING, (B)PROCESSING, (C) DOWNLOADING TIME COMPARISON WITH DIFFERENT TASK SIZES.

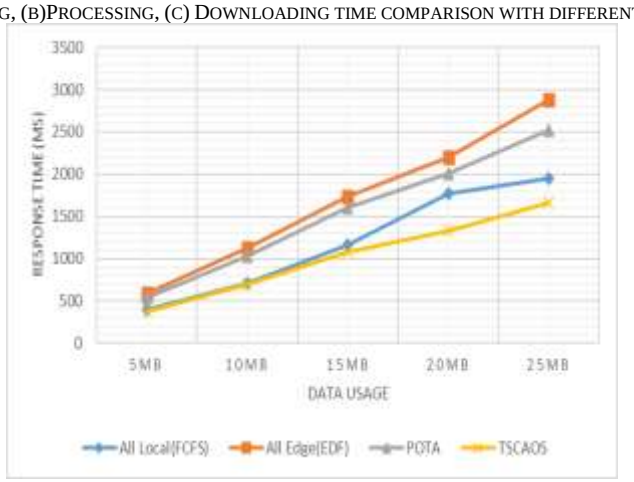


FIGURE 3
RESPONSE TIME AGAINST DATA USAGE UPLOADING

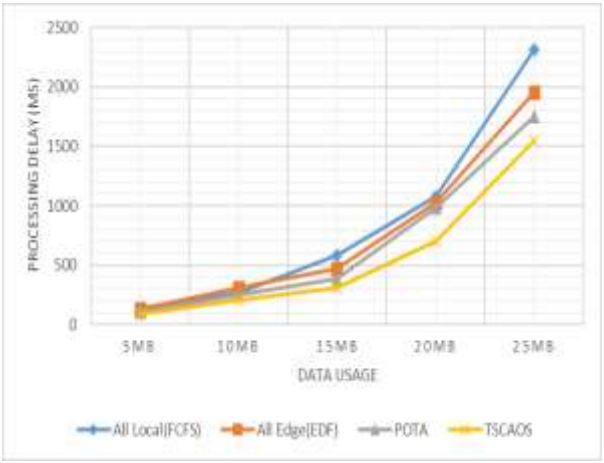


FIGURE 4
PROCESSING DELAY AGAINST DATA USAGE

The graphs given in Figure 2 (A), (B) and (C) shows the uploading time, processing time and downloading time respectively for all algorithms with respect to task size. TSCAOS performs better with all the tasks sizes. In the Figure 3 the graph represents comparison between the average response tie vs data usage of the algorithm's performance. The response time is the addition of the above parameters like uploading, processing and downloading in terms of milli second (ms) with the different data usage. In this comparison TSCAOS takes 35.02 % less response time for completing the task then the other algorithms. In the Figure 4 shows the graph of processing delay with respect to data usage, here the TSCAOS reduces the delay by 30.86% compared to other algorithms.

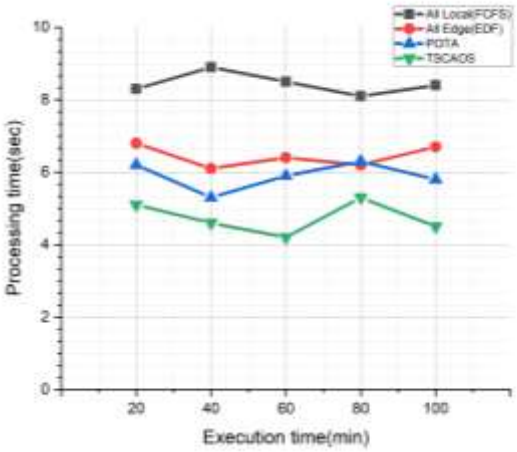


FIGURE 5
SIMULATION TIME INTERVALS FOR PROCESSING TIME FOR EACH TASK EXECUTION

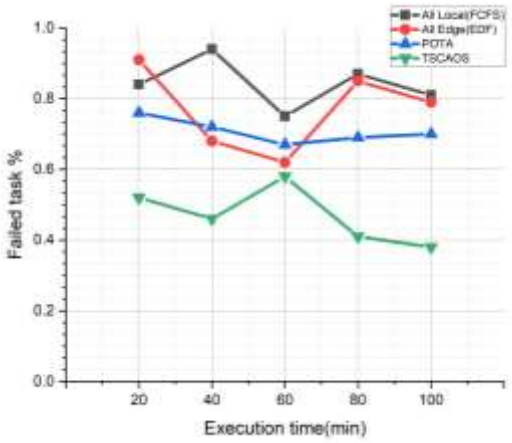


FIGURE 6
SIMULATION TIME INTERVALS FOR FAILED TASK RATE % OF EACH TASK EXECUTION

Here the TSCAOS algorithm is evaluated against different algorithms listed in Table III by means of failed task rate and processing time against the different simulation times. The simulation time is measured at increasing period of 20 min time interval for each task in EdgeCloudSim simulator. The five simulation tasks taken for experimental execution in above time intervals are 10, 20, 30, 40 and 50 Mbps sizes. The processing time against execution time (simulation time) for above mentioned tasks using different algorithms is depicted in Figure 5. In the same manner, task failure rate is also measured against execution time (simulation time) which is depicted in Figure 6. TSCAOS reduces the task failure rate and processing time by 48.49% and 36.11% respectively when compared to other methods.

CONCLUSION

IoT computation demands are increasing rapidly and therefore relies on edge computation mechanisms to complete the task executions. This paper proposes task scheduling with CPU allocation offloading Scheme that is TSCAOS implemented in edge server network and compared with the existing offloading schemes. From the analysis, the other offloading schemes are developed using the partial offloading and task scheduling schemes by combining both or individual implementation. The major shortcoming is not considering the CPU utilization. In this work, a novel scheme termed TSCAOS is proposed where task scheduling and partial offloading are performed from node to edge servers by considering the utilization of CPU. A virtual reality application is used for the simulation in EdgeCloudSim simulator. Based on the simulation results comparisons are done for FCFS in local node layer, EDF in edge node layer, POTA in edge node and edge server layer and TSCAOS in edge node to edge server layer from which TSCAOS outperforms in means of response time and processing delay. In this comparison TSCAOS takes 35.02 % less response time, reduces the delay by 30.86% compared to other algorithms. TSCAOS also reduces the task failure rate by 48.49% and execution time by 36.11% comparatively. In future, more applications can be evaluated using this algorithm and by increasing the task sizes and resource allocation levels. The proposed scheme can be further explored with more resource parameters which influence the efficiency of task accomplishment.

REFERENCES

[1] The internet of things: A movement, not a market. [Online]. Available: https://cdn.ihs.com/www/pdf/IoT_ebook.pdf

[2] M. Satyanarayanan, "The Emergence of Edge Computing," Computer, vol. 50, no. 1, pp. 30–39, Jan. 2017, doi: 10.1109/MC.2017.9.

[3] Z. Amjad, A. Sikora, B. Hilt, and J.-P. Lauffenburger, "Low Latency V2X Applications and Network Requirements: Performance Evaluation," in 2018 IEEE Intelligent Vehicles Symposium (IV), Changshu, Jun. 2018, pp. 220–225. doi: 10.1109/IVS.2018.8500531.

[4] S.-I. Sou and O. K. Tonguz, "Enhancing VANET Connectivity Through Roadside Units on Highways," IEEE Trans. Veh. Technol., vol. 60, no. 8, pp. 3586–3602, Oct. 2011, doi: 10.1109/TVT.2011.2165739.

[5] F. Bonomi, R. Milito, J. Zhu, and S. 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 - MCC '12, Helsinki, Finland, 2012, p. 13. doi: 10.1145/2342509.2342513.

[6] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge Computing: Vision and Challenges," IEEE Internet Things J., vol. 3, no. 5, pp. 637–646, Oct. 2016, doi: 10.1109/JIOT.2016.2579198.

[7] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The Case for VM-Based Cloudlets in Mobile Computing," IEEE Pervasive Comput., vol. 8, no. 4, pp. 14–23, Oct. 2009, doi: 10.1109/MPRV.2009.82.

[8] H.-J. Jeong, I. Jeong, H.-J. Lee, and S.-M. Moon, "Computation Offloading for Machine Learning Web Apps in the Edge Server Environment," in 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), Vienna, Jul. 2018, pp. 1492–1499. doi: 10.1109/ICDCS.2018.00154.

[9] S. E. Mahmoodi, R. N. Uma, and K. P. Subbalakshmi, "Optimal Joint Scheduling and Cloud Offloading for Mobile Applications," IEEE Trans. Cloud Comput., vol. 7, no. 2, pp. 301–313, Apr. 2019, doi: 10.1109/TCC.2016.2560808.

[10] L. Zhang, D. Fu, J. Liu, E. C.-H. Ngai, and W. Zhu, "On Energy-Efficient Offloading in Mobile Cloud for Real-Time Video Applications," IEEE Trans. Circuits Syst. Video Technol., vol. 27, no. 1, pp. 170–181, Jan. 2017, doi: 10.1109/TCSVT.2016.2539690.

[11] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in 2016 IEEE International Symposium on Information Theory (ISIT), Barcelona, Spain, Jul. 2016, pp. 1451–1455. doi: 10.1109/ISIT.2016.7541539.

[12] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-Edge Computing: Partial Computation Offloading Using Dynamic Voltage Scaling," IEEE Trans. Commun., pp. 1–1, 2016, doi: 10.1109/TCOMM.2016.2599530.

[13] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing," IEEE/ACM Trans. Networking, vol. 24, no. 5, pp. 2795–2808, Oct. 2016, doi: 10.1109/TNET.2015.2487344.

[14] G. Zhang, Y. Chen, Z. Shen, and L. Wang, "Energy Management for Multi-User Mobile-Edge Computing Systems with Energy Harvesting Devices and QoS Constraints," in 2018 27th International Conference on Computer Communication and Networks (ICCCN), Hangzhou, Jul. 2018, pp. 1–6. doi: 10.1109/ICCCN.2018.8487435.

[15] J. Ren, G. Yu, Y. Cai, Y. He, and F. Qu, "Partial Offloading for Latency Minimization in Mobile-Edge Computing," in GLOBECOM 2017 - 2017 IEEE Global Communications Conference, Singapore, Dec. 2017, pp. 1–6. doi: 10.1109/GLOCOM.2017.8254550.

[16] O. Munoz, A. Pascual-Iserte, and J. Vidal, "Optimization of Radio and Computational Resources for Energy Efficiency in Latency-Constrained Application Offloading," IEEE Trans. Veh. Technol., vol. 64, no. 10, pp. 4738–4755, Oct. 2015, doi: 10.1109/TVT.2014.2372852.

[17] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-Efficient Resource Allocation for Mobile-Edge Computation Offloading," IEEE Trans. Wireless Commun., vol. 16, no. 3, pp. 1397–1411, Mar. 2017, doi: 10.1109/TWC.2016.2633522.

[18] M. Asim, Y. Wang, K. Wang, and P.-Q. Huang, "A Review on Computational Intelligence Techniques in Cloud and Edge Computing," IEEE Trans. Emerg. Top. Comput. Intell., vol. 4, no. 6, pp. 742–763, Dec. 2020, doi: 10.1109/TETCI.2020.3007905.

[19] S. Deng, L. Huang, J. Taheri, and A. Y. Zomaya, "Computation Offloading for Service Workflow in Mobile Cloud Computing," IEEE Trans. Parallel Distrib. Syst., vol. 26, no. 12, pp. 3317–3329, Dec. 2015, doi: 10.1109/TPDS.2014.2381640.

[20] C. Canali and R. Lancellotti, "GASP: Genetic Algorithms for Service Placement in Fog Computing Systems," Algorithms, vol. 12, no. 10, p. 201, Sep. 2019, doi: 10.3390/a12100201.

[21] L. Tang, B. Tang, L. Kang, and L. Zhang, "A Novel Task Caching and Migration Strategy in Multi-Access Edge Computing Based on the Genetic Algorithm," Future Internet, vol. 11, no. 8, p. 181, Aug. 2019, doi: 10.3390/fi11080181.

[22] P. Zhao, H. Tian, C. Qin, and G. Nie, "Energy-Saving Offloading by Jointly Allocating Radio and Computational Resources for Mobile Edge Computing," IEEE Access, vol. 5, pp. 11255–11268, 2017, doi: 10.1109/ACCESS.2017.2710056.

[23] W. Wang, R. Lan, J. Gu, A. Huang, H. Shan, and Z. Zhang, "Edge Caching at Base Stations With Device-to-Device Offloading," IEEE Access, vol. 5, pp. 6399–6410, 2017, doi: 10.1109/ACCESS.2017.2679198.

[24] X. Dutreilh, S. Kirgizov, O. Melekhova, J. Malenfant, N. Rivierre, and I. Truck, "Using Reinforcement Learning for Autonomic Resource Allocation in Clouds: towards a fully automated workflow," p. 9, 2011.

[25] A. T. Nassar and Y. Yilmaz, "Reinforcement Learning-based Resource Allocation in Fog RAN for IoT with Heterogeneous Latency Requirements," arXiv:1806.04582 [cs], Jan. 2019, Accessed: Dec. 30, 2021. [Online]. Available: http://arxiv.org/abs/1806.04582

[26] X. Liu, Z. Qin, and Y. Gao, "Resource Allocation for Edge Computing in IoT Networks via Reinforcement Learning," arXiv:1903.01856 [eess], Mar. 2019, Accessed: Dec. 30, 2021. [Online]. Available: http://arxiv.org/abs/1903.01856
[27] M. Agarwal and G. M. S. Srivastava, "A genetic algorithm inspired task scheduling in cloud computing," in 2016 International Conference on Computing, Communication and Automation (ICCCA), Greater Noida, India, Apr. 2016, pp. 364–367. doi: 10.1109/CCAA.2016.7813746.
[28] J. Liu, X.-G. Luo, X.-M. Zhang, F. Zhang, and B.-N. Li, "Job Scheduling Model for Cloud Computing Based on Multi-Objective Genetic Algorithm," vol. 10, no. 1, p. 6, 2013.
[29] X. Tao, K. Ota, M. Dong, H. Qi, and K. Li, "Performance Guaranteed Computation Offloading for Mobile-Edge Cloud Computing," IEEE Wireless Commun. Lett., vol. 6, no. 6, pp. 774–777, Dec. 2017, doi: 10.1109/LWC.2017.2740927.
[30] D. M. Dakshayini, "An Optimal Model for Priority based Service Scheduling Policy for Cloud Computing Environment," International Journal of Computer Applications, vol. 32, p. 8.
[31] Y. Chen, Y. Zhang, and S. Maharjan, "Deep Learning for Secure Mobile Edge Computing," p. 7.
[32] Z. Luo, M. LiWang, Z. Lin, L. Huang, X. Du, and M. Guizani, "Energy-Efficient Caching for Mobile Edge Computing in 5G Networks," Applied Sciences, vol. 7, no. 6, p. 557, May 2017, doi: 10.3390/app7060557.
[33] Z. Cheng, P. Li, J. Wang, and S. Guo, "Just-in-Time Code Offloading for Wearable Computing," IEEE Trans. Emerg. Topics Comput., vol. 3, no. 1, pp. 74–83, Mar. 2015, doi: 10.1109/TETC.2014.2387688.

AUTHOR INFORMATION

S. Premkumar, Research Scholar, Computer Science and Engineering, Annamalai University, India. He has finished Master of Engineering (CSE) in Annamalai University. Currently he is also serving as a Project fellow (CSE) under UGC India granted DST-PURSE scheme at Annamalai University. His interested areas are Artificial Intelligence, Internet of Things, Edge computing and Cloud computing.

AN. Sigappi, Received her Ph.D in Computer Science and Engineering from Annamalai University in 2013. She did her Master Degree in Computer science and engineering from Anna University. Currently she is serving as a Professor in the Department of Computer Science and Engineering, Annamalai University, India. Her areas of interest include Image Processing, Machine Learning, Data Analytics and Internet of things. She has published more than 25 research articles in international journals and conferences.