ADAPTIVE OFFLOADING MECHANISM IN HIERARCHICAL MULTI-CLOUD FRAMEWORK

Naga Lakshmi Somu[1], Dr Prasadu Peddi[2]

[1]*Research Scholar, Department of Computer Science, Shri Jagdishprasad Jhabarmal Tibrewala University, Rajasthan*
[2]*Professor, Dep of CSE & IT, Shri Jagdishprasad Jhabarmal Tibrewala University, Rajasthan.*

ABSTRACT

This research effort performs adaptive offloading across context-aware, capability-aware, and energy-aware models in order to meet the needs and provide context adaptability. Demand-based offloading satisfies a variety of application needs and provides efficient resource allocation. These offloading strategies are only effective when they are used in conjunction with the appropriate cloud resource selection. Applications perform worse when a single kind of cloud resource has problems with availability, dependability, backhaul network congestion, and unusual connection loss during offloading. As a result, the single cloud architecture is less flexible to meet application requirements. By creating a multi-cloud framework made up of heterogeneous cloud resources most suited for the suggested offloading models, this study introduces a change in cloud architecture. To achieve context adaptability, a game-theoretic method is suggested for choosing the appropriate offloading model and cloud resource. Using this method, the interactions between the players and other entities—like user gear, the root server, and the app vendor—are studied. These interactions aid in determining the best offload option that benefits each entity individually. Based on their computation and communication impacts with other players, such entities' usefulness is made available to the participants.

Keywords: *Cloud computing, offloading, edge computing and Virtual Machines.*

1. INTRODUCTION

The growth of an intelligent society and the ongoing enhancement of people's requirements have impacted a variety of sectors and people's day-to-day social interactions. A wide range of applications for edge devices have emerged in society, including cameras, intelligent production robots in intelligent manufacturing, intelligent homes as well as autonomous cars within the field of transport. Therefore, there is significantly more devices that are connected to the Internet. Based on Cisco's Global Cloud Index [1] it was 17.1 billion connected devices to the internet in the year 2016. In 2019, 10.4 Zettabytes (ZB) of data will be transmitted across global data centers and 45 percent of the data will be stored, processed and analyzed on the edge of the network; and by 2020, there'll more than 50 billion devices wirelessly connected to the internet. Globally, device data generation has also grown, rising from 218 ZB in 2016 to 847 ZB in 2021. According to figures from the international data provider Internet figures Centre (IDC), there will be more than 50 billion terminals and devices linked to the network by 2020, and there will be more than 40 ZB of data globally [2]. Cloud-based big data processing has shown several drawbacks due to the constant and enormous expansion of data volume and diverse data processing needs. Among them are:

Real-time: Adding a lot of edge devices will still result in a lot of terminal data being sent to the cloud for processing, which will significantly increase the volume of intermediate data transmission and decrease the performance of data transmission. It will place an enormous stress on the network's transmission bandwidth. This can cause delays in the transmission of data. Cloud computing is unable to satisfy business needs for real-time responses in a variety of situations that call for instantaneous input, such traffic and monitoring.

Security and privacy: For instance, utilizing a variety of smartphone apps may need providing user data, including private data. This data is very vulnerable to privacy breaches or attacks after it is uploaded to

the cloud centre. Energy consumption: China's data centers are using a lot more electricity as a result of the country's growing smart device population. Improving cloud computing energy consumption efficiency is insufficient to meet the growing need for data energy consumption [3]. The demands for energy-intensive cloud computing will increase in the rapidly developing intelligent society.

Section 1 defines the introduction, section 2 works on literature survey, section 3 with methodology section 4 defines results and finally conclusion

2. STUDY OF EXISTING RESEARCH

On the other hand, Fernandoet'al in [4] suggested a way to leverage all kinds of nearby assets (computers, PADs, and smartphones) that may work together to build the nearby cloud in order to gain a not uncommon goal. Their strategy is to overcome the issues with limited connection, battery consumption, and scant assistance that come with typical mobile cloud computing. Workload sharing is proactive, dynamic, and price-based in order to benefit everyone. An assistance handler, an activity handler, and a value handler are specifically included in the structure. The resource handler locates the collocated assets; the price handler then computes the values to determine how jobs should be distributed to maximize benefits; the job handler then assigns the subtasks, performs the responsibilities, and retrieves the assets from the sender. Lastly, a number of the participating devices' micropayments are handled by the cost handler.

In [5], the SPACCE concept is put out to give PCs with calculating capability in order to enable dispersed cooperation. A SPACCE is a sophisticated adhoc cloud computing environment that may be dynamically moved a server for utility sharing to a different computer based on the demands that change over time on a collection of private, or non-dedicated, computers. Through the process of server migration, redundant computer capacity may be used to construct a SPACCE, which advances the software's shared response time for consumers. A SPACCE offers the characteristics that a laptop's calculating capacity may have since it serves as a collaborative server for various PCs that lack applications or have insufficient processing power to function as a server when needed[6].

One effective way to accomplish a shared goal is for cell devices to cooperate [6] and function as a single unit in a networked environment. Sometimes, however, work cannot be divided across mobile devices and must be transferred to a platform with plenty of helpful resources. To do it, an executable block migration must be carried out[7].

The stack-on-demand asynchronous exception (SOADA) execution technique was suggested in [8] to offload work to a neighboring cloud. Only the current execution country, which is located on top of the runtime stack, may be relocated on this technique, which maintains a stack for storing execution state. Therefore, SOD migrates the necessary portion of the data to the vacation location website the best using this approach, regardless of the size of the picture. Asynchronous exception handling has been used to carry out the portable picture-taking process on mobile devices. Dual method hierarchy methodology has been used to decrease overhead. However, latency is a new parameter when offloading to the cloud[9].

A cloudlet [10] design that employs M. Satyanarayan to lower latencies suggests a two-tier strategy. According to the proposed design, we can probably use a local, resource-rich cloudlet to handle the resource shortage of the mobile device instead of depending on a distant "cloud." A decentralized addition to internet infrastructure, cloudlets are widely dispersed and may be utilized to acquire processing power and storage by neighboring cell computing systems. Cloudlet access may be facilitated by wifi, which is more energy-efficient and offers more bandwidth than conventional internet services[11].

3. Methodology:

Offloading is the prominent solution that speeds up the execution of the application. The application contains complex tasks which may take long processing time when executed at the user equipment itself. Fragile devices with such complex tasks prefer cloud for running their tasks. With the help of cloud servers offered by cloud providers, the heavier tasks of UE can be migrated by offloading them. This reduces the conductance cost or completion time of the application. It assures a quality of service delivered to the users. The average speedup of the application is also improved by decreasing the conductance cost of the application. Though conductance cost is very much reduced through offloading, it is important that the pre-offload process does not append additional overhead to the application execution[12].

One of the main challenges in task offloading is transmission overhead. This overhead includes transmission latency and transmission energy spent for offloading process. The geographical distance between user and cloud creates a huge latency and delays the application execution[13]. The communication latency will have great impact on the overall completion time of the application. If the input and output data to be transmitted during offloading is large, then the data transportation cost will be high. Also, more bandwidth is consumed by user to complete its task. The decision of whether to execute the task at remote server or locally at UE has to be made carefully. Suppose if the decision is to offload, then time for preoffload process in finding appropriate cloud resource is also added to the remote execution time of application. This overall pre-offload processing time must be lesser than local execution time for profitable offloading.

Ultra-Reliable Low-Latency Communication (URLLC) applications like AR/VR capture real-time images in mobile camera and process it to obtain its information. This task of faster rendering of localized contents is crucial in those applications. The speed and latency are the major factors which needs improvement for such applications. Since these applications involve multimedia processing, it may drain the battery of user equipment quickly. Hence, the energy consumption will be very high for it. Quality of experience including high bandwidth requirements is also mandatory for continuous delivery of services and seamless rendering to UE without feeling dizzy and nauseous. Mobile Edge Computing (MEC) is the optimal cloud solution for these applications which can provide latency lesser than 13 ms. 5G with MEC has the objective of reducing latency and energy consumption of mobile equipment. Collectively, it ensures scalability, availability and user experience properties of an ideal cloud[14].

The arrival of offload requests from UE is irregular and also contains divergent requirements. The offload decision must contemplate these changes to offer an optimal solution to user equipment. On the other hand, the computing capacities of heterogeneous cloud servers also vary. Hence, the resource allocation technique should give an optimal benefit for the users as well as the service providers. It has to ensure that the servers are utilized properly with no overloading or under utilization. Our proposed adaptive offloading using game-theoretic approach satisfies both UE and app vendor by considering the real-time environmental changes in the network and servers. Moreover, it develops a multi-cloud framework with heterogeneous cloud resources that can deliver services to dissimilar user requirements[15].

For latency-critical tasks require higher processing power for their execution. Hence, latency-aware and energy-sensitive models are used for handling such tasks. However, the other two cloud resources such as Fog, MEC considered in this proposed work use different context-aware information. Initially, in the framework of Alternatives, the device monitor checks the CPU, memory and bandwidth details of the UE. In addition, the communication cost between UE and Fog nodes is considered in the Fog network. The real-time context information such as geospatial and bandwidth information of the UE are collected from Radio Access Networks (RAN) in MEC framework to provide QoE services to UE.

Calculation offloading is basically a disseminated figuring worldview, in which cell phones influence remote servers to accelerate processing and spare vitality. In calculation offloading, cell phones need to transmit information over the system. It is detailed as

Tlocal > Toffloading = Tcomm + Tremote

Tcomm demonstrates the correspondence time, and Tlocal and Tremote allude to the execution time at nearby and remote, individually. Then, the vitality advantage is

Elocal > Ecomm

Accordingly, offloading is useful if performing a lot of calculation with transmitting a moderately little amount of information [37]. At that point, we play out a further investigation on the Toffloading with the cloud and the cloudlet. Edge gadgets have the correspondence time with the cloud Tcomm cloud bigger than Tcomm cloudlet for cloudlets. In any case, the cloud has a few sets of extent figuring speed quicker than cloudlets, at that point

Tcloud < Tcloudlet.

This basic investigation gives an understanding that applications which are data transmission hungry and idleness touchy yet decently figure concentrated can accomplish significantly more advantage. Despite what might be expected, applications requiring enormous scale figuring are still cloud accommodating, for instance, web search.

Algorithm 1 Updated Migration Operator.

1) do // butterflies in SP1 for i = 1 to NP1
2) where D is the length of an individual butterfly and k = 1 to do
3) Choose a random rand number generator.
4) rand peri = r.
5) In the event if r p
6) Choose a butterfly at random from subpopulation 1 (let's say r1);

7) use Equation (1) to generate xt+1 i,k;
8) else
9) Choose a butterfly at random from subpopulation 2 (let's say r2);
10) Use Equation (3) to generate xt+1 i,k;
11) End if;
12) End for;
13) Generate xt+1 i,new using Equation (10), which illustrates the greedy technique.
14) Conclude for

Algorithm 2 SPMBO
Algorithm.
1) First things first. In addition to setting the maximum generation tm, NP1, NP2, BAR, peri, and the lower (pmin) and higher (pmax) constraints of parameter p, set the generation counter t = 1.
2) Population assessment. Determine the fitness based on the goal function.
3) while t < tm execute
4) Arrange the population of butterflies.
5) Use Equation (7) to compute parameter p.
6) Ascertain how many butterflies are present in each of land 1 (NP1) and land 2 (NP2).
7) Separate individual butterflies into two subpopulations.
8) do // butterflies in SP1 for i = 1 to NP1
9) Use Algorithm 1 to execute the revised migration operator.
10) finish for 11) do // butterflies in SP2 for j = 1 to NP2.
12) Execute the fundamental MBO algorithm using the butterfly adjusting operator.
13) Come to an end for
14) Determine the fitness of the freshly created butterfly individuals.
15) t is equal to t plus 1.
16) come to an end, and
17) print the finished answer.

Algorithm:
Future state host Prediction
Input : Workload_monitor
Output: Host Predicted
Construct Matrix of Probability Transimition
Extract the Overload protocol O
end
O Overload avg
if O Overload avg ≥ Threshold_Upper then
    Host has been overloaded
    Migration = False
    Call transmission Algorithm again
end
else if O Overload avg ≤ Threshold_Lower then
    Host is under loaded
    Migration =True
end
End

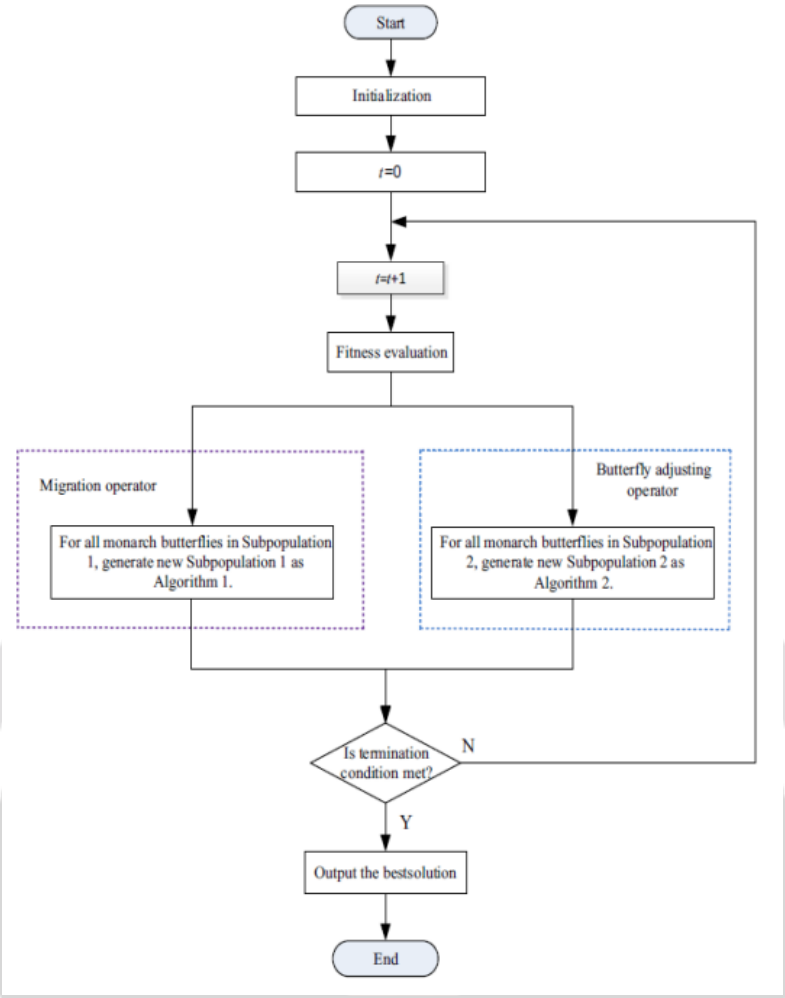

Fig. 1 Flowchart of MBO method

4. Experiments & Results

Implementing the Selective Offloading Scheme Due to the heterogeneity in gadgets and IoT administrations, offloading is more useful for some gadgets while nearby execution becomes increasingly advantageous for others. Offloading can be beneficial for a gadget that has a limited amount of assets, such as if it is unable to provide specialized assistance. We then introduce the following condition to crisis management assignments that can be offloaded. Conditions 1. If Til> Tireq, then the confirmation controller will choose gadget I to be offloaded. Asset-limited gadgets that have delay-sensitive undertakings and meet Condition 1 will be arranged for offloading because their local processing capacity can't fulfill the dormancy requirements (i.e. Tit > Tyrq). The edge server then pre-dispenses the fimin assets for these gadgets. It also determines the outstanding assets and examines the condition that prevents some gadgets from being offloaded.

Table 1: CPU Consumption for offloading

CPU Consumption	Data size (in Bytes)
0.0112	10
0.312	20
0.00357	30

0.012	40
0.00396	50

Table 2: CPU consumption for local computation

CPU Consumption	Data size (in Bytes)
0.0199	10
0.0512	20
0.0152	30
0.0059	40
0.00470	50

Table 3 Workload Data

Workload	NumberofHosts	NumberofVMs	TimePeriod
Real	3	3	Day1
			Day2
			Day3
			Day4
			Day5
			Day6
			Day7
			Day8
			Day9
Random	3	30	24hrs

Each host has a storage capacity of 1TB and a bandwidth of 1GB/second. There are four VM kinds that are used that resemble Amazon EC2 instances. High compute instances that have capacities in 2500 MIPS and 0.85 GB, larger instances that have a capacity of 2500 MIPS along with 3.75 GB, small instances that have capacities that is 1000 MIPS and 1.7 1 GB and micro instances that have a capacity in excess of 500 MIPS and 0.633 gigabytes make up the configuration of VMs.. Each kind of VM has a 100 MB bandwidth and a 2.5 GB storage capacity.

With a total of 30 VMs, we have mimicked 3 hosts. Various workloads are used in the trials. The functions carried out by VMs on hosts need a lot of resources and the CPU. The VMs are migrated to other hosts in order to fulfill the CPU demand for the VM and guarantee that the work is executed continuously and without violating SLAs. Occasionally, the recently relocated virtual machines may not need more CPU, but this might result in the host's present virtual machines running low on resources. Since VM migrations have an impact on the application's performance, they should only be performed when necessary. There are 90% fewer migrations when comparing an initial Markov to a second-order Markov. The number of migrations for the second order Markov is equal to the number of migrant events for the first order Markov for certain labour loads. For certain labour loads, the number of migration events in the first order Markov model is equal to the number of migrations in the second-order Markov model on day 6. The frequency of migrations across the first and second order is the same for 1% of the load on days 7 and 8, however for 95% of the load, there are fewer migratory incidents in the second order. On day nine, no migrations occur, based on the second third order Markov model. The second-

order Markov model exhibited fewer movements than the first order in 95% of the work completed on day ten.

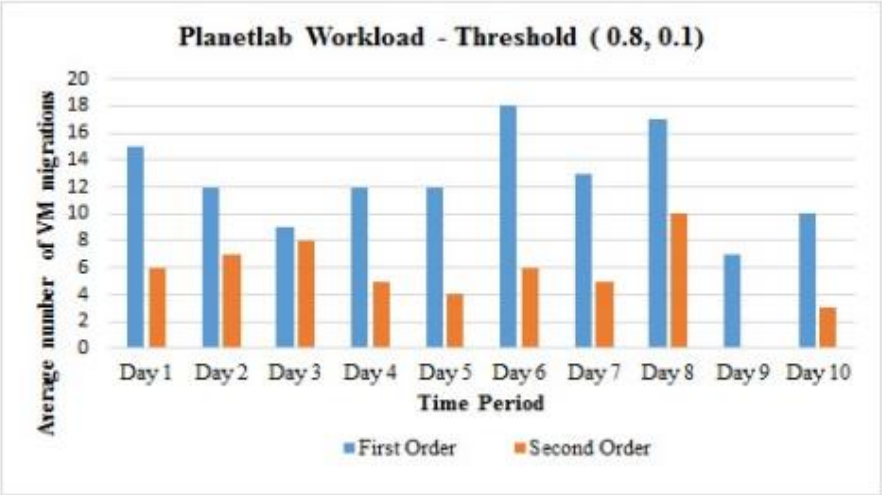

Fig 2: Planetlab Workload: Average VM Migrations-for-82%-CPU-Utilization

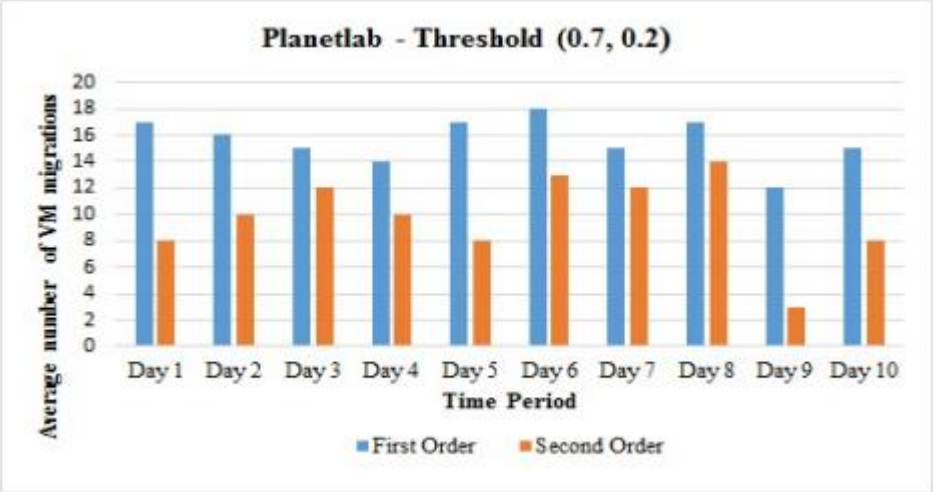

Fig 3: Workload on Planetlab: Mean Count of VM Migrations for 72% CPU Usage

If you use the Markov model of second order It has been shown that when the threshold is set at 0.8, 0.1 for each of Planetlab and random workloads, the average number of VM migrations drops by more than 60%. Similarly, using this second ordered Markov model and selecting thresholds of 0.7 and 0.2 for random workloads results in a reduction of around 80% in the number of virtual machine migrations. The number of migration in the Planetlab workloads with thresholds set to 0.7 and 0.2 drops by 40% when using the second-order Markov model.

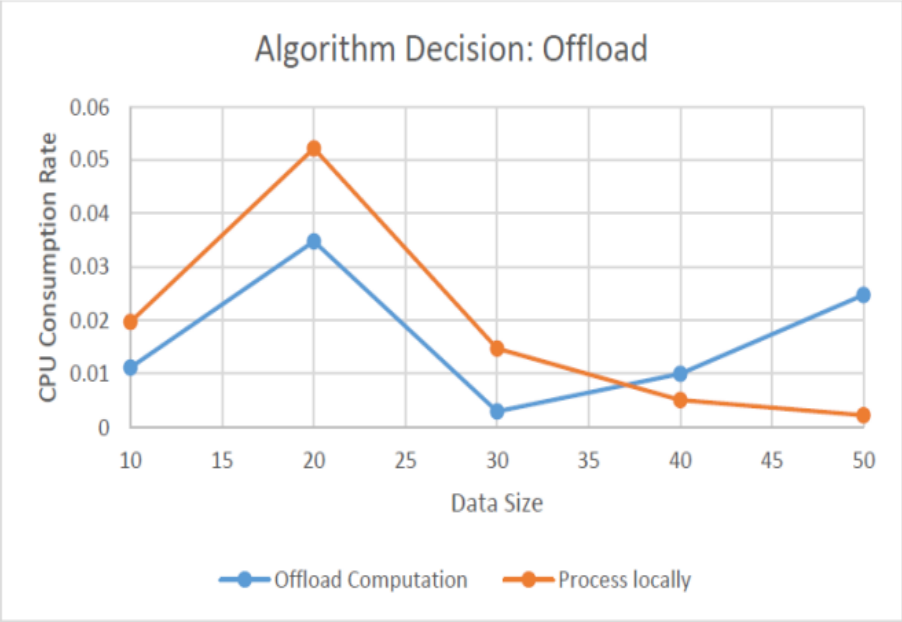

Fig 4: CPU consumption rate for an Offload Decision

We can find in the figure over that our calculation predicts precisely when managing little information as offloading really expends lower CPU. Be that as it may, when managing genuinely bigger information, the calculation still proposes an offload despite the fact that as appeared over, that devours more CPU. In this area, we actualize our proposed way to deal with check the exhibition, and we assess our proposed methodology contrasted and a few delegate situation approaches as far as remaining task at hand adjusting, get to defer under different edge server outstanding burdens, and the position of various quantities of edge servers. In this way, the vital situation of edge servers to offload the remaining task at hand of some overburden base stations ought to be direness performed to upgrade the exhibition of versatile applications.

5. CONCLUSIONS

The proposed resource allocation scheme is designed as a three-stage sequential game which in turn formulated as optimization problems corresponding to three tiers. In each stage of the game, backward induction method is applied to find sub-optimal solution. From the sub-optimal solutions, an overall Stackelberg equilibrium for the game is obtained. It ensures that the allocation is successful and all the players yield a fair profit from which no player deviates. When an offload request is generated at the UE, it is sent to the root server for processing. The root server does a Contextaware offloading to search for competent alternatives within the local network of users.

REFERENCES

1 K. Kaur, S. Garg, G. S. Aujla, N. Kumar, J. J. P. C. Rodrigues and M. Guizani, “Edge Computing in the Industrial Internet of Things Environment: Software-Defined-Networks-Based Edge-Cloud Interplay,” in IEEE Communications Magazine, vol. 56, no. 2, pp. 44-51, Feb. 2018.
2 S. Deng, L. Huang, J. Taheri and A. Y. Zomaya, “Computation Offloading for Service Workflow in Mobile Cloud Computing,” in IEEE Transactions on Parallel and Distributed Systems, vol. 26, no. 12, pp. 3317-3329, 1 Dec. 2015.
3 F. Guo, H. Zhang, H. Ji, X. Li and V. C. M. Leung, “Joint Trajectory and Computation Offloading Optimization for UAV-assisted MEC with NOMA,” IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Paris, France, 2019, pp. 1-6.
4 S. Singh, Y. Chiu, Y. Tsai and J. Yang, “Mobile Edge Fog Computing in 5G Era: Architecture and Implementation,” 2016 International Computer Symposium (ICS), Chiayi, 2016, pp. 731-735.
5 Y. Zhang, J. He and S. Guo, “Energy-Efficient Dynamic Task Offloading for Energy Harvesting Mobile Cloud Computing,” 2018 IEEE International Conference on Networking, Architecture and Storage (NAS), Chongqing, 2018, pp. 1-4.

6 S. Azodolmolky, P. Wieder and R. Yahyapour, "Cloud computing networking: challenges and opportunities for innovations," in IEEE Communications Magazine, vol. 51, no. 7, pp. 54-62, July 2013.
7 W. Shi, J. Cao, Q. Zhang, Y. Li and L. Xu, "Edge Computing: Vision and Challenges," in IEEE Internet of Things Journal, vol. 3, no. 5, pp. 637-646, Oct. 2016, doi: 10.1109/JIOT.2016.2579198.
8 L. Lei, H. Xu, X. Xiong, K. Zheng and W. Xiang, "Joint Computation Offloading and Multiuser Scheduling Using Approximate Dynamic Programming in NB-IoT Edge Computing System," in IEEE Internet of Things Journal, vol. 6, no. 3, pp. 5345-5362, June 2019.
9 Z. Kuang, Y. Shi, S. Guo, J. Dan and B. Xiao, "Multi-User Offloading Game Strategy in OFDMA Mobile Cloud Computing System," in IEEE Transactions on Vehicular Technology, vol. 68, no. 12, pp. 12190-12201, Dec. 2019.
10. Prasadu Peddi, & Akash Saxena. (2014) "Exploring The Impact Of Data Mining And Machine Learning On Student Performance", ISSN-2349-5162, Vol 1, Issue 6, pp:314-318.
11. K. R. R. Babu, A. A. Joy, P. Samuel, in 2015 Fifth International Conference on Advances in Computing and Communications (ICACC). Load balancing of tasks in cloud computing environment based on bee colony algorithm, (2015), pp. 89–93. https://doi.org/10.1109/ICACC.2015.47
12. X. He, Z. Ren, C. Shi, F. Jian, A novel load balancing strategy of software-defined cloud/fog networking in the internet of vehicles. Chin. Commun. 13(S2), 145–154 (2016)
13. Prasadu Peddi, & Dr. Akash Saxena. (2016). STUDYING DATA MINING TOOLS AND TECHNIQUES FOR PREDICTING STUDENT PERFORMANCE. International Journal Of Advance Research And Innovative Ideas In Education, 2(2), 1959-1967.
14. X. Wang, Z. Ning, L. Wang, Offloading in internet of vehicles: A fog-enabled real-time traffic management system. IEEE Trans. Ind. Inf. (2018). https://doi.org/10.1109/tii.2018.2816590
15. Naga Lakshmi Somu, Prasadu Peddi (2021), "An Analysis Of Edge-Cloud Computing Networks For Computation Offloading", Webology (ISSN: 1735-188X), Volume 18, Number 6, pp 7983-7994.