IJSES
Volume 6, Issue 3, pp. 15-19, 2022.
International Journal of Scientific Engineering and Science
ISSN (Online): 2456-7361

Task Offloading Strategy Based on Improving Edge Server Computing Resources

Jie Fang[1], Chengyu Wen[1], Xiulan Sun[1], Juchao Zeng[1], Hantao Liu[2], Wenzao Li[1,2]
[1]College of Communication Engineering, Chengdu University of Information Technology, Chengdu, China
[2]Educational Informationization and Big Data Center, Education Department of Education
Correspondence should be addressed to Chengyu Wen: 308282616@qq.com

***Abstract*—** *The popularity of edge computing makes up for the lack of computing power of cloud computing in the era of massive data. Besides, computation task offload to mobile edge severs which can effectively reduce the energy consumption and transmission delay. Plenty of research shows that a suitable task offloading schedule can optimize the performance of an application system. However, Computing resource utilization of edge devices is a major factor affecting task offloading. It is a key factor to the system load balancing. Making full use of edge computing resources is beneficial to save computation resource, energy efficiency and load balancing. Furthermore, it helps unlock the potential performance of the system. Aim to this, we proposed a balance offloading strategy in this paper. We not only ensure the success rate of task offloading but also improves the utilization of computing resources on edge devices. The success rate of task offloading is nearly the greedy strategy. And the utilization of edge computing resources is better than other strategies proposed in this paper.*

***Keywords*—** *Edge computing, offloading strategy, computing resources, balance strategy.*

I. INTRODUCTION

The emerging applications like AR and VR can generate massive data, which increases the computing burden on the mobile devices. Despite the computing ability of mobile devices have good performance now, they are unable to achieve real-time and efficient execution due to their limited computing resources and ever-demanding applications. The emerging of cloud computing is a promising solution. With the excellent computing capacity, it can rapidly and efficiently process the massive uploaded data[1]. In a computing paradigm, users can rely on extremely rich storage and computing resources of a cloud computing center to expand the computing and storage power of devices, and achieve the rapid processing of computing intensive tasks. Yet there are some disadvantages in the cloud computing, such as incurring high transmission delay[2] and pushing network bandwidth requirement to the limit[3].In order to solve this problem, edge computing [4] is proposed to provide nearer services for users. Edge computing is a distributed computing scheme that allows tasks on the mobile devices uploading to the edge servers for processing. In contrast to the cloud computing[5], the edge computing infrastructure can provide increased bandwidth and reduced latency, which significantly improving the Quality of Service. Traditional scheduling strategies of edge computing tasks are to offload all computing intensive tasks of edge device to an edge server for processing[5]. However, it may result in the waste of computing and storage sources in edge devices and cloud computing centers. In addition, many devices accessing an edge server at the same time can result a long queue of tasks in edge server, which increases the processing burden of edge server[6].This increases the completion time of all queued tasks, even causing the processing delay of tasks in the edge server to exceed that at the edge devices. On the other hand, many edge devices may be idle, resulting in a waste of their computing resources and resource-rich cloud centers may be underutilized[7]. To solve this problem, we combine the edge server and cloud serve to improve the computing resource utilization and reduce the failure rate. To further reduce task failure rate and improve edge server computing resource utilization, we propose a balance scheduling strategy. Compared with other scheduling strategies, it can effectively reduce the failure rate and improve the utilization of edge server resources. In failure rate, the balance strategy is nearly the greedy strategy and averaged 10% lower than random strategy. In edge server utilization, the balance strategy has the best performance.

The organization of this paper is as follows: In Section II, the background of the edge computing task offloading strategies and related works are detailed. And the simulation environment and system model are presented in Section III. In Section IV, we will give the results of our balance scheduling strategies. Finally, we will conclude our study and provides possible directions for future search in Section V.

II. RELATED WORKS

The appearance of edge computing makes up for the problem of insufficient computing resources and limited power of terminal equipment. Task offloading is one of the hot topics of edge computing. Li et al. [8], they formulate the sum cost of delay and energy consumptions for all edge devices as their optimization objective. Wu et al.[9][9], they utilize the cost of energy in edge computing. Some scholars focus on optimizing bandwidth. Wang et al.[10], they optimize the transmission bandwidth allocation and data compression rate to reduce the energy consumption. While they do not consider about the utilization of edge server. It may lead to the waste of computing resources of edge server and increase the spend of cloud computing server. Cagatay et al. [11], they based on fuzzy logic to consider the load of edge server. While they do not farther increase the utilization of edge server. Our work mainly focuses on increasing the utilization of edge server. This is of great significance for saving computing cost and making full use of computing resources.

International Journal of Scientific Engineering and Science
Volume 6, Issue 3, pp. 15-19, 2022. ISSN (Online): 2456-7361

III. SIMULATION ENVIRONMENT AND SYSTEM MODEL

3.1 Simulation environment

In this part, we mainly introduce our simulation environment. EdgeCloudSim [12] is a useful tool for edge computing. It provides a simulation environment specific to edge Computing scenarios where it is possible to conduct experiments that considers both computational and networking resources. We use it in our experimentation. We modeled the experimentation phase with the focus on university campus scenario. In our experiment, as shown in Figure 1, in the campus, each student uses some edge devices, such as mobile phones, VR and health sensors. Students move around the campus randomly, randomly in all corners of the campus, such as libraries, buildings, cafes. At each building, we put a wireless access point (AP)and edge server. Student access wireless access points via WLAN to edge servers, which connect to cloud servers via WAN. The edge servers are connected to each other through LAN. In the experiment, the unloading task generated by edge devices can be executed by edge servers or cloud servers.

Fig. 1: Proposed scenario of a campus

3.2 System model

3.2.1 Application

For each edge server, Some users are randomly assigned. And a user every time generate only one task. It is denoted by a set tasks=(task1,task2,task3… taskn). AS for each task, we simulated three applications. AR (Augmented Reality), Health App and Information App. The detailed configuration of each application is as follows:

TABLE 1: Application details

Application type	AR	Health App	Information App
Use percentage(%)	60	20	20
Active period(sec)	45	10	40
Idle period(sec)	15	20	60
Data upload(kb)	1500	40	20
Data download(kb)	25	20	1000
VM utilization on edge (%)	20	5	10
VM utilization on edge (%)	2	0.5	3

We used three different applications to simulate the real world as much as possible in this chart, User Percentage means the probability of a user using the application. Active period and Idea period means when the application run and sleep. We use these three parameters to randomly and dynamically generate one application at a time for each use. Data upload represents the amount of data uploaded to an edge server or cloud server through a wireless access point. Data download refers to the amount of data returned to the terminal through the wireless access point after the application is processed by the server. In order to simulate a really world scenario as closely as possible, we chose three typical applications. When user use Information App, such as watching a video, the mobile will download much data from the Internet. Opposite, VR will upload bigger data. Health App do not need download and upload much data.VM utilization means the usage of computing resources.AR requires the server to execute and calculate heavy data. It has the higher utilization.

The VM utilization is our important optimization goal. Improving the utilization of computing resources helps to save computing costs. And it is beneficial to improve the stability of the system. There are resources reserved for future application. For each edge server, there is a queue of tasks waiting to be executed.

3.2.2 System architectures

In this part of the experiment, we mainly focus on the utilization of computing resources in two architectures.

In one-tie, Terminal tasks can only be unloaded to the edge server near the terminal and executed by the edge server. While in two-tier, Terminal tasks can be offloaded to the edge server near the terminal and executed in collaboration with the cloud. As shows in pictures:

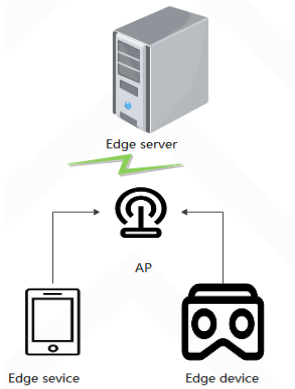

Fig. 2: 0ne-Tier data offloading structure

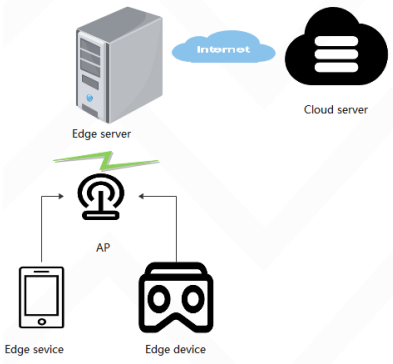

Fig. 3: Two-Tier data offloading structure

International Journal of Scientific Engineering and Science
Volume 6, Issue 3, pp. 15-19, 2022. ISSN (Online): 2456-7361

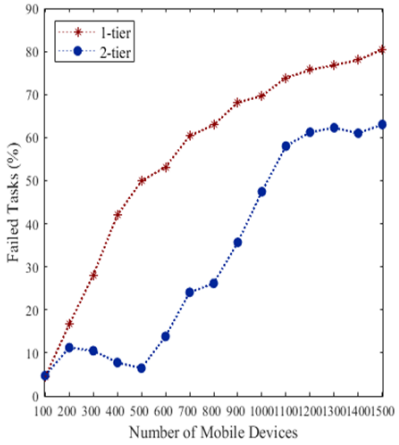

Fig. 4: Task failure ratio

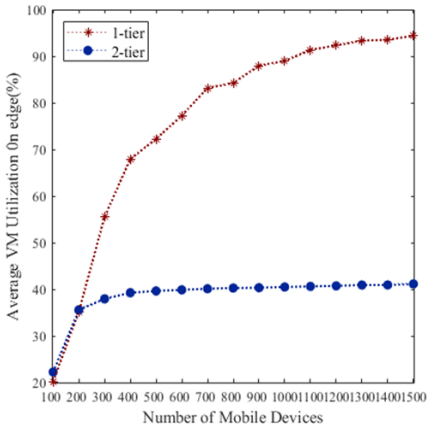

Fig. 5: Average VM utilization on edge serve

The experimental results show that in the one-tier architecture, the number of failure tasks increases with the increase of the number of terminals in fig 5. In fig 6, edge devices are fully utilized and computing resource utilization increases. However, edge terminals are overloaded. And tasks cannot be processed timely. As a result, the task failure rate increases. In the two-tier architecture, some tasks are offloaded to cloud servers, which relieves the load on edge server VMS and reduces the task failure rate. Meanwhile, tasks are offloaded to cloud servers, which reduces the utilization rate of edge computing resources. Meanwhile, transferring tasks to cloud servers increases the cost of task procession. Then the reasons for the failure of the task are further analyzed. The main failure caused by the following three reasons. First, the mobility. In our experiments, each around the edge server has a wireless access point. Edge server through the wireless access point to send and receive tasks. When the terminal leaves the wireless point which sends task to the edge server, the task processing results cannot be returned to the terminal. So, the task failure duo to mobility. Second, VM capacity. In our experiment, each edge server and cloud server have multiple virtual machines (VMS) that can concurrently process multiple tasks. However, when the VM load is heavy, the VM cannot complete the uninstalled tasks. And the tasks fail duo to VM capacity. Third, Network congestion. When a task is uninstalled to the cloud server, the task accesses the WAN through the wireless access point. However, due to limited bandwidth resources, a great number of tasks are queued. Some tasks cannot be offloaded timely. It causes the task failure. The task failure rate caused by the three reasons are shown in the figures below:

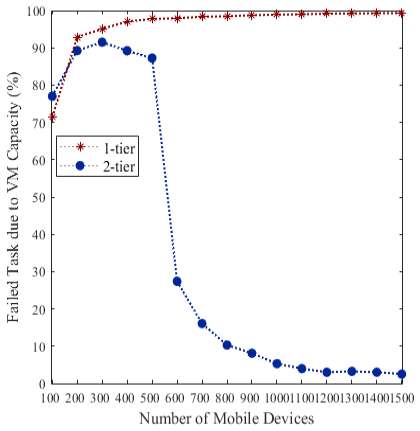

Fig. 6: Failed task due to VM Capacity

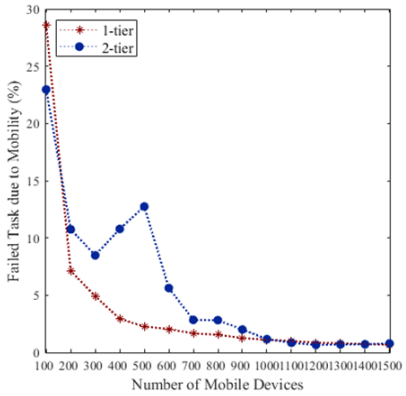

Fig. 7: Failed task due to Mobility

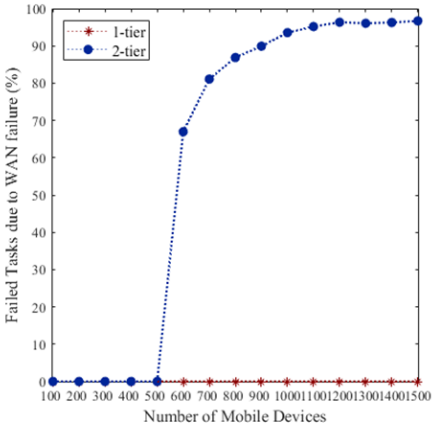

Fig. 8: Failed task due to Network congestion

As shown in the above picture, the main reason of the task failure is that the limited VM capacity in one-tier architecture.

International Journal of Scientific Engineering and Science
Volume 6, Issue 3, pp. 15-19, 2022. ISSN (Online): 2456-7361

The edge server has a heavy load, when all tasks are executed by the edge server. The VM of edge server is overloaded. The tasks cannot be processed in timely and the task fails. In two-tier architecture, when the VM capacity of edge servers reaches 40%, tasks which are unloaded to cloud services. It reduces the task load on edge servers and reducing the task failure rate. The main change of two-tier is that the Network congestion. The tasks queue needs to be unloaded to the cloud through the wireless access point. However, the network bandwidth of the wireless access point is limited and the task queue causes network congestion, resulting in task failure.

In the following experiments, we further optimized the unloading strategy to ensure the success rate of task unloading and make full use of computing resources of edge devices. In actual scenarios, a great number of terminals may gather in some places. On the one hand, the wireless access point in these places is heavily loaded. If all devices access the WAN through this point, network congestion may occur and network delay may be increased. On the other hand, edge servers with dense terminal distribution also have the problem of heavy task processing pressure. While servers with loose terminal distribution have low task unloading pressure, resulting in low computing resource utilization. In the experiment, all the wireless access points were in the same LAN. Based on the edge 0rchestrator module provided by the simulator, we can distribute the task to other wireless access points through the wireless access point. Meanwhile, the task can also be performed by the edge server near the new wireless access point, which relies the processing pressure of the terminal dense edge server. First, we randomly distribute tasks to other edge servers for execution. The results are shown below:

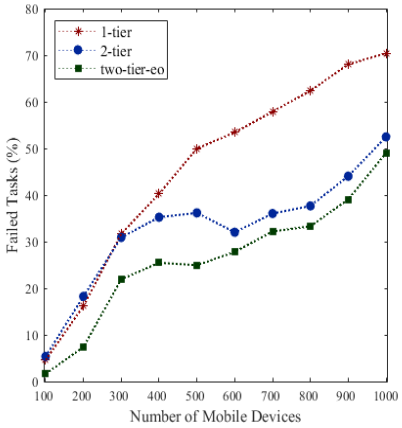

Fig. 9: The ratio of task failure

As shown in fig 9, The 0rchestrator strategy is adopted to reduce the task offloading failure rate. Then, the causes of task unloading failure are analyzed in detail. In fig 11, by randomly distributing tasks to other edge servers, the VM load is reduced and the task failure rate caused by VM load is reduced. The random 0rchestrator strategy is adopted to reduce the load of VM and improve the success rate of task offloading. In fig 11, it may cause unbalanced offloading with random strategy. Computing resource usage of edge devices is low. Next, further research is carried out to further improve the utilization rate of edge equipment. We propose the balance strategy.

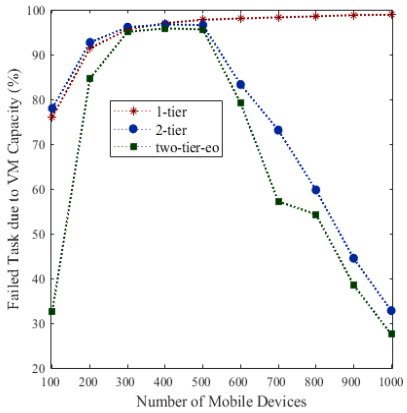

Fig. 10: Failed task duo to VM capacity

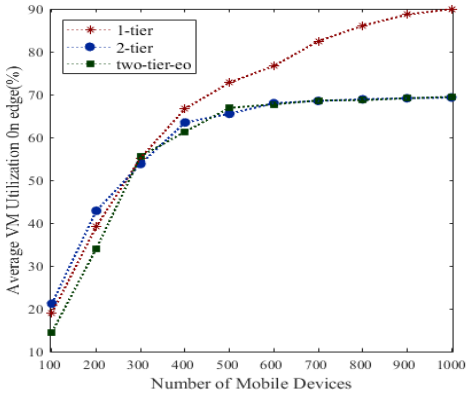

Fig. 11: Average VM utilization on edge serve

IV. BALANCE STRATEGY

When the terminal accesses the edge server from the access point, if the load of the edge server at this time is less than a certain value, we consider that the edge server is idle at this time and the load rate is low, then the task will be processed by the edge server. When the edge server load is greater than this value, we choose to offload tasks to other edge servers for execution. In the process of selecting edge server, the iterative algorithm is adopted to continuously find edge server lower than this value, and the task is unloaded to the edge server meeting the conditions for execution. If no such edge server can be found, the task is offloaded to the cloud server for execution.

In this part of the experiment, we compare three task offloading strategies: Random, Greed and Balance. When grave strategy is used, the edge server with the lowest load is selected for cooperation. And the idle edge server is selected for execution each time. As for the Balance strategy, when a task is in the edge server load capacity has not yet reached a certain value, we can think of at this moment the edge server load is small. The task is still executed by the edge server. But when the task reaches this threshold, we will assign tasks to other edge server. We try to make integral edge server load to achieve

the effect of a more balanced. The experimental results are shown below:

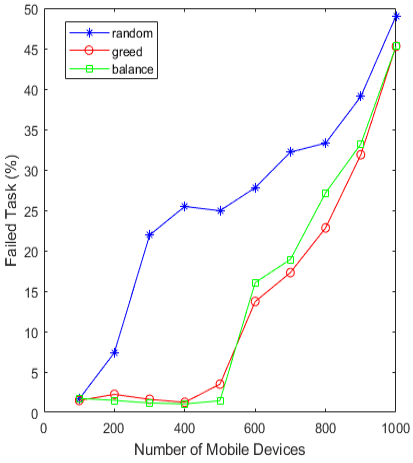

Fig. 12: The ratio of task failure

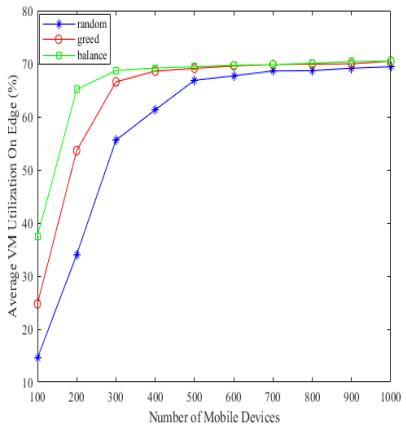

Fig. 13: Average VM utilization on edge serve

In terms of task failure rate, the failure rate of the balance offloading strategy is smaller than the random strategy, approximately equal to greed strategy. But, when it comes to the VM utilization, the balance strategy has the best utilization. In conclusion, the balance strategy improves the utilization of computing resources of edge services. And its task failure rate is close to greedy strategy.

V. CONCLUSION

In this paper, we focus on improve the utilization of edge server computing resources. We propose a balance strategy that not only improve the utilization of edge server computing resource, but also decrease the rate of failure. In contrast to greed strategy and random strategy, the greedy strategy has the higher rate of task offloading success. While the balance strategy has the higher utilization of edge server. As for the rate of success of task offloading, the balance strategy is very approach to the greedy strategy. The balance strategy is a better strategy for task offloading. In the future work, we can combine the network resources with computing resources to further decrease the rate of failure of task offloading.

ACKNOWLEDGMENTS

We thank all the reviewers and editors who have contributed to the quality of this paper. At the same time, we also appreciate the support by the fund from the Network and Data Security Key Laboratory of Sichuan Province, University of Electronic Science and Technology of China (UESTC) (NO.NDS2021-7). This work is also supported by College Students' Innovative Entrepreneurial Training Project, Chengdu University of Information Technology (202110621147).

REFERENCES

[1] Dinh H T, Lee C, Niyato D, et al. A survey of mobile cloud computing: architecture, applications, and approaches[J]. Wireless communications and mobile computing, 2013, 13(18): 1587-1611.

[2] Yang X, Fei Z, Zheng J, et al. Joint multi-user computation offloading and data caching for hybrid mobile cloud/edge computing[J]. IEEE Transactions on Vehicular Technology, 2019, 68(11): 11018-11030.

[3] Ren J, Zhang D, He S, et al. A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet[J]. ACM Computing Surveys (CSUR), 2019, 52(6): 1-36.

[4] Yu W, Liang F, He X, et al. A survey on the edge computing for the Internet of Things[J]. IEEE access, 2017, 6: 6900-6919.

[5] Imagane K, Kanai K, Katto J, et al. Performance evaluations of multimedia service function chaining in edge clouds[C]//2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC). IEEE, 2018: 1-4.

[6] Mach P, Becvar Z. Mobile edge computing: A survey on architecture and computation offloading[J]. IEEE Communications Surveys & Tutorials, 2017, 19(3): 1628-1656.

[7] Ahmed E, Ahmed A, Yaqoob I, et al. Bringing computation closer toward the user network: Is edge computing the solution?[J]. IEEE Communications Magazine, 2017, 55(11): 138-144.

[8] Li J, Gao H, Lv. T, et al. Deep reinforcement learning based computation offloading and resource allocation for MEC[C]//2018 IEEE Wireless Communications and Networking Conference (WCNC). IEEE, 2018: 1-6.

[9] Wu G, Miao Y, Zhang Y, et al. Energy efficient for UAV-enabled mobile edge computing networks: Intelligent task prediction and offloading[J]. Computer Communications, 2020, 150: 556-562.

[10] Wang J B, Zhang J, Ding C, et al. Joint Optimization of Transmission Bandwidth Allocation and Data Compression for Mobile-Edge Computing Systems[J]. IEEE Communications Letters, 2020, 24(10): 2245-2249.

[11] Sonmez C, Ozgovde A, Ersoy C. Fuzzy workload orchestration for edge computing[J]. IEEE Transactions on Network and Service Management, 2019, 16(2): 769-782.

[12] Sonmez C, Ozgovde A, Ersoy C. Edgecloudsim: An environment for performance evaluation of edge computing systems[J]. Transactions on Emerging Telecommunications Technologies, 2018, 29(11): e3493.