School of Informatics

Informatics Research Review
Edge Computing Offloading in Internet of Things: Experimental Designs and Configurations

s2517285
January 2024

Abstract

Based on the increasingly popular trend of edge computing, amounts of papers have concentrated on the optimization algorithms on the edge. However, due to the various configurations of the model, it is impossible to effectively compare between different algorithms. Hence, this review will focus on the difference between these configurations and the reasons for the design.

Date: Saturday 20$^{\text{th}}$ January, 2024

Supervisor: Zonglin Ji

1 Introduction

Internet of Things (IoT) means that the objects around people can communicate with each other and cooperate to achieve common goals, which has great potential for both private and business uses [1]. Most tasks handled by devices in IoT tend to be delay-sensitive, which also generate an amount of data of nearly 49 EB [2]. However, the IoT devices are usually limited in terms of memory, battery life, and computing power [3, 4]. Hence, it is impossible to process all the application tasks in local devices and meanwhile satisfy all the performance requirements [2]. As a consequence, computation offloading is applied to solve this problem. Computation offloading means the tasks on the user equipment (IoT devices) can be transmitted to the remote server while getting the result after the remote server has successfully processed the tasks [5]. Hence, the local devices are not required to be equipped with advanced hardware or software which could be expensive.

In tradition, cloud computation integrates with the Internet of Things since cloud server has great capability. Unlike IoT, the storage and computation power provided centrally by the cloud server is almost unlimited, which corresponds to the disadvantages of IoT [6, 7]. Hence, cloud computing can help IoT devices to complete their computation tasks with high performance. However, for IoT devices, it is an obstacle to obtaining stable and acceptable network performance to reach the cloud [8]. Additionally, the cloud server has to be challenged by reliability problems since the devices may fail or become inaccessible [8]. Unfortunately, the extensive scale of the resultant system, stemming from interactions with a significant number of devices, renders the rising requirements for storage capacity and computational power in subsequent processing progressively difficult to meet [8]. Therefore, edge computing is introduced to address the issues of the IoT and the cloud.

Edge Computing (EC), also named Mobile Edge Computing (MEC), provides cloud computing capabilities within the Radio Access Network close to mobile users [9]. Compared with cloud computing, edge computing is more likely to compute in real-time because the edge servers are located at places closer to the users [10]. Moreover, edge computing doesn't need the users to upload the data to the cloud computing center and reduce the load on the network bandwidth, which lowers the cost and the network bandwidth pressure [10]. Many algorithms have been designed to optimize the task offloading problem in IoT applications based on edge computing. Nevertheless, these algorithms shown high performance have not been systematically compared to come to a conclusion about the best algorithm. One of the reasons is the experimental designs and configurations for each algorithm are extremely different. For example, one configuration includes the cloud component while another only considers the edge servers. Consequently, it is difficult to get an objective comparison.

As edge computing plays a more and more significant role in coordinating the work between IoT devices, it is necessary to design optimization algorithms to enhance the functionality of IoT through the utilization of Edge Computing characteristics. Quantities of optimization algorithms have been proposed and implemented, however, it is impossible to compare their performance due to the difference between system models (edge computing models). There are several important components that can be used to build different kinds of edge computing models for IoT devices, which may have a great impact on the performance of the algorithms on edge computing. Hence, this review will summarize the designs or configurations for the IoT application based on edge computing. It is noted that the pattern mentioned only includes the cloud server, edge server and objective function.

To address the problem of differences in edge computing system models that result in incomparability, this article will attempt to answer the following questions:

1. What are the main differences between different system models? How do those affect the performance of the algorithms?

2. Why did the designers choose such configurations? What are the pros and cons?

3. Based on questions 1 and 2, what designs or configurations should be considered when applying optimization algorithms?

Section 2 will focus on the cloud computing component in the EC designs. Additionally, in section 3 the full offloading and partial offloading will be discussed. Last but not least, section 4 will study the different objective functions chosen by the EC system models.

2 Literature Review

2.1 Cloud Computing components

Though the main conception discussed in this review is edge computing, it doesn't mean that the cloud components should be excluded from the edge computing models. Because edge computing and cloud computing are not mutually exclusive, instead they are complementary. However, there exist some papers which don't agree to include the cloud components for some reasons.

Zhang et.al [11] have constructed a model which doesn't consider the cloud server in the system model. They suggest that transmission failure probability can be largely reduced if the tasks aren't offloaded to the cloud server. Additionally, they believe that the model has to suffer from significant delay if introducing the cloud server, while the quality of experience can be guaranteed by their designed cooperative network. It should be noted that each edge server only belongs to one cooperative network based on the physical distance and the cluster of the edge servers will be divided into K cooperative networks. Furthermore, the IoT device will offload the tasks to the closest edge server. Nevertheless, since the model also adds constraints to the edge servers on the maximum throughput of processing the tasks, the cooperative will assist in redirecting the offloaded task to other nearby edge servers. Additionally, Ali et.al [12] attempt to concentrate on the smart offloading of IoT devices for edge computing, hence the cloud computing component is unnecessary for the algorithm research.

Nevertheless, When the number of tasks doesn't exceed a threshold based on the number of edge servers, the edge servers can provide better service due to the shorter transfer time. However, if the number of tasks grows beyond the threshold, the shortage of the computation resources of the edge servers compared to the cloud server will show the impact on performance [4]. Hence, it is reasonable to introduce the cloud server, with powerful computing resources and computing power, to help take the burden of the edge servers by processing an excessive number of tasks. Nonetheless, for those models that have included the cloud server, there is a problem with how to coordinate the edge servers and the cloud server's tasks.

Ning et.al [4] have proposed a computation offloading model that the IoT equipment that sends offloading requests to the small evolved NodeBs (SeNBs), while SeNBs take the responsibility of offloading tasks to the edge server or the cloud server according to schedule algorithms. This model also supposes that the transmission delay between SeNBs and edge servers can be ignored, which means sending tasks to the edge server or cloud server via SeNBs takes the same amount of time as sending them directly to the edge or cloud server. To better simulate the characteristics of the edge servers and cloud servers, the model adds constraints to the number of tasks to be processed for every edge server, but cloud servers have no such constraints. Additionally, a similar model designed by Jiang et.al [13] sets a manager in edge computing servers to decide where to process the tasks according to the result of the optimization algorithm. The role of the manager is similar to the SeNBs, but the manager is responsible for the whole assignment of tasks, rather than the assignment of offloaded tasks. These models set a special model to gather the edge servers and cloud server real-time information [4]. Hence, the extra transmission delay can be avoided since the IoT devices are not required to get the information of servers' available resources.

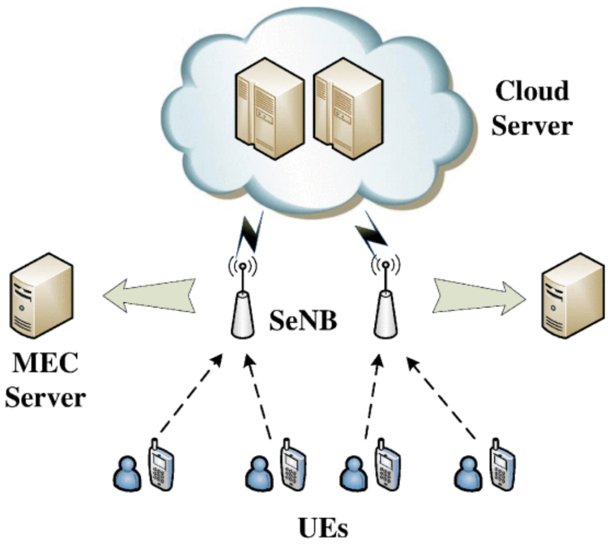

Figure 1: Special server to schedule [4]

Another computation model created by Chen et.al [14] also includes cloud computing components, however, the architecture differentiates from Ning's model. The SeNBs are used to schedule whether the task offloading request is sent to the edge server or cloud server in Ning's model. However, in Chen's model, all offloaded tasks are sent to the edge server at first, if the edge server can't process more tasks, some tasks will be offloaded to the cloud server to reduce the burden of the edge server. Hence, this model has to judge by itself whether it can process the coming tasks

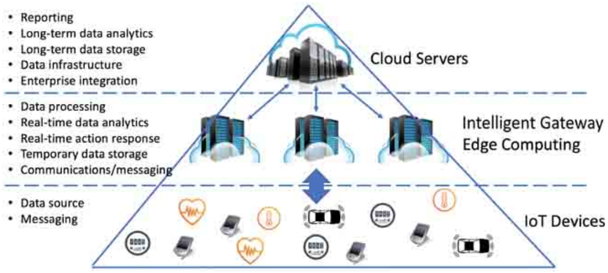

Figure 2: Offloading tasks directly from edge server to cloud server [15]

Hence, whether the cloud computing component is included can have a great impact on the system model since there are pros and cons to using cloud servers. The cloud server can provide a huge amount of computing resources. However, due to the distance from IoT devices to the cloud, the delay or the energy consumption has to be considered which could increase the complexity of the model and has the potential to lower the performance. Furthermore, the role of deciding the direction of the offloading tasks is significant, in light of the fact that different roles would select different strategies.

2.2 Offloading Strategies

Since IoT devices have limited computation and energy resources, they can hardly satisfy the complicated tasks required by the IoT service. Therefore, the goal of the task offloading is to gain computation capability without using more energy-cost devices [15]. The offloading strategies can be separated into two categories: full offloading and partial offloading strategies. A full offloading strategy means offloading the task all to the edge computing server or cloud server. On the contrary, the partial offloading strategy is aimed at dividing tasks into several parts, one part is executed on the local machine while the other parts are offloaded to the edge server or cloud server [16].

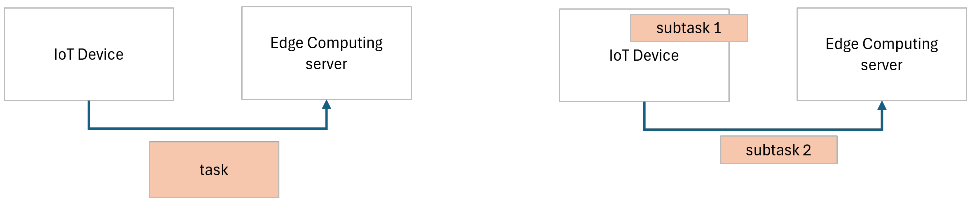

Figure 3: Full offloading strategy VS partial offloading strategy

Zhang et.al applies a full offloading strategy to process the tasks, unlike the partial offloading strategy, which treats the task as the smallest unit [11]. Another research proposed by Chen et.al also chooses full offloading strategies [17]. The reason that they both make such a choice is that from their point of view, due to the heterogeneity of the IoT devices, it is unlikely to gather the prior statistical information such as the size of the coming tasks [11, 17]. Hence, from this perspective, the better option is the full offloading strategy since it makes the model more general and realistic.

However, Ning et.al [4] supports the partial offloading strategy rather than the full offloading strategy. They suggest that one factor that affects the choice of different strategies is the type of applications. For example, if the input data of the application is private information, the tasks should be partially offloaded. Nevertheless, the complex relationships between different modules in each task can make the model become more complicated. Therefore, to simplify the model, the complicated module dependency system (the module refers to the part of the tasks) has been simplified into a linear sequence processing module which means the output of the last module is the input of the next module. Moreover, the paper emphasizes that the computation offloading model can be applied to the tasks that are not allowed to be offloaded, since the module has a flag to indicate whether the module is executed locally or remotely in the edge or cloud server. As a result, their offloading model becomes more general since it not only can capture the situation of full and partial offloading but also can capture the case where the task is executed locally on IoT devices.

There are many other factors that affect the offload strategy, such as the quality of communication links, the computing resource availability of the cloud and edge server, and IoT devices's abilities [4]. Furthermore, based on the condition that the task is allowed to be divided, for the reason of optimizing the user's energy conservation, partial offloading strategy has a higher priority [18]. However, the offloading becomes more complicated when the partial offloading strategy is considered, for the reason that the task relevance, characteristics and segmentation have to be concentrated [19].

It is worth mentioning that even though those experiments choose partial offloading strategies, there is a big difference in the level of granularity they use when dividing tasks into subtasks. Huang et.al's partial offloading strategy considers the tasks that can be partitioned at any granularity. Consequently, the optimization problem of this model becomes non-linear and non-convex, which is more challenging to solve [2]. By contrast, since Ning's model only considers the linear sequence relationship, the model is simpler to solve as a mixed integer linear problem [4]. Nevertheless, Ali et.al [12] have proposed a more systematic way to partition the tasks. Instead of fixing the number of modules or allowing partition of arbitrary granularity, they apply deep neural networks (DNN) to find the optimal components and the optimal way to partition. The dataset, consisting of the size of the task, the minimum size of the module or partition, the largest possible number of components and other information, will be generated by a comprehensive cost function. As a consequence, after training the model, it can generate a partition policy for the coming tasks.

As discussed above, the heterogeneity of the edge server, the unknown upcoming tasks, and the difficulty of reasonably dividing the tasks make the full offloading strategy more appealing. However, some specific information hidden in the tasks, the energy-saving requirements and other factors give support to the application of the partial offloading strategy.

2.3 Objective Functions

The objective function can be composed of one or several objectives. Hence, if one objective function focuses on delay, and another objective function focuses on energy, it is hard to compare the performance of the optimization algorithms since they optimize in different directions. Hence, it is essential to review different objective functions in previous literatures.

The objective function chosen by Ning et.al focused on computation offloading delay which consists of process time and transmission time [4]. It should be noted that the subject of the objective function is the module (the partition of the tasks). Hence, whether the module is executed locally or remotely on an edge server or cloud server can have an impact on the process and transmission delay. To conclude, the objective function only focuses on the total delay of the tasks. Additionally, the objective function of another model also concentrated on the delay. However, it is significant that for both of the objective functions, different edge server has different computing capabilities. [11]. For example, the edge server in Ning's model can only process one offloading request at a time, but another objective function uses the maximum size of total tasks to constrain the objective function, which means the edge server can process multiple tasks at the same time. Moreover, the subjects involved in the model are distinct, one pertains to the module, while the other corresponds to the task itself.

Instead of concentrating on the delay, the model created by Chen et.al [17] has used a cost function to replace the total delay. This cost function consists of network cost and transmission cost, which can be influenced by the number of offloaded tasks, the traffic and other factors. Hence, this objective function is based on the actual cost of using the network, edge server and cloud server. Although it takes into account delay, it is merely considered as a constraint rather than an optimization objective. As a consequence, comparing the optimal solution of the algorithm with the optimal solution of the previous models' solution is meaningless.

Except for the delay and the cost, objective functions can also be dominated by energy. According to the study conducted by Fu et.al [20], the objective function cares about the energy consumption caused by task processing and task offloading. Moreover, the delay and the limitation of the computation resources are constraints on the objective function. Hence, the definition of the optimal solution could be slightly different from the delay or the cost objective functions.

Lu et.al's model [21] is different from others since IoT devices are equipped with energy storage equipment and energy collector. As a consequence, the objective function is restricted by the power stored in the storage equipment. Furthermore, the function consists of three different objectives: service delay, energy consumption and task success rate based on coding error probability which hasn't been discussed in other objective functions. The reason for considering the task success rate is that it is likely that when the downloading process results from the remote server, errors could happen during this process. Therefore, this objective function places a great emphasis on the quality of the service instead of only the delay or the cost. An objective function that is slightly different from the previous one consists of three objectives: delay, energy consumption and price. The aim of applying this objective function is to guarantee the quality of the service while lowering the energy consumption and the cost [19].

In conclusion, the objective functions encompass various types, and each type of objective function has numerous variants. The emphasis of those objective functions is different, which makes the comparison between different types of optimization algorithms impossible. For example, the objective function composed of multiple objectives is more comprehensive and realistic at the cost of simplicity.

3 Summary & Conclusion

This review has studied many kinds of literature on different edge computing optimization designs and configurations. One of the conclusions that can be drawn is that the main difference between different designs or configurations is the cloud computing component, offloading strategies and the objective function.

For the cloud computing component, the advantages and disadvantages of including a cloud server have been discussed. Although the delay and the transmission failure probability caused by offloading tasks to the cloud could affect the IoT device's service quality, with the great computation power and resources, it is reasonable to introduce the cloud server into the computation model to avoid bottlenecks of the edge servers and quantities of experiments have considered the cloud. Additionally, the role that determines when to offload tasks to the cloud server also has a significant impact on the model. For example, the manager server will the optimization solution of the scheduling algorithm as the decision basis, while the edge server may offload the tasks when the edge can't handle more tasks.

Apart from the cloud component, the offloading strategies have an impact on the models. If the partial offloading strategy is chosen, it is important to consider the methods to deal with complicated dependency relationships between the modules in the tasks. Additionally, the way to partition the tasks into different modules is essential to the model structure. If it is supposed that the tasks can be arbitrarily partitioned, it would become a non-line and non-convex problem which increases the difficulty of the algorithm. However, if another algorithm is introduced to get the optimal partition, a dataset has to be generated to train the algorithm. However, if the full offloading strategy is selected, the model will become more realistic because it is hard to get prior information about the tasks. Additionally, the type of the applications and other factors also support the application of partial offloading strategies.

After reviewing many papers, the objective function is the most varied difference between different experiments and configurations. The objective function can only focus on one objective, moreover, it can be the combination of different objectives using weights. The most common single objective functions focus on delay, energy or cost, which is the different aspect of measuring the performance of edge computing. Moreover, the multi-objective functions are more comprehensive and they care more about the overall quality of the service of the users.

To conclude, the cloud component is hard to decide whether it should be configured in the computation model since either of the decisions has pros and cons. However, for the offloading strategies, partial offloading is recommended due to the possible requirements of the applications or the users and the coverage of the situation of full offloading. On the contrary, it is difficult to judge which type of objective function should be considered because the only difference between them is their emphasis on the model.

4 Future Work

Many differences between the configurations and designs of the experiments have been discussed. However, other differences such as the computing ability of the IoT devices can influence the models and the application of the optimization algorithms should be considered. Additionally, a more general edge computing model should be proposed to make the comparison between different models easier. Finally, based on the general model, the previous optimization algorithms should be reimplemented according to this model if they attempt to prove their performance on the problem.

References

[1] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer Networks*, 54(15):2787–2805, 2010.

[2] Jiwei Huang, Ming Wang, Yuan Wu, Ying Chen, and Xuemin Shen. Distributed offloading in overlapping areas of mobile-edge computing for internet of things. *IEEE Internet of Things Journal*, 9(15):13837–13847, 2022.

[3] Arash Heidari, Mohammad Ali Jabraeil Jamali, Nima Jafari Navimipour, and Shahin Akbarpour. Internet of things offloading: Ongoing issues, opportunities, and future challenges. *International Journal of Communication Systems*, 33(14):e4474, 2020. e4474 IJCS-19-0207.R3.

[4] Zhaolong Ning, Peiran Dong, Xiangjie Kong, and Feng Xia. A cooperative partial computation offloading scheme for mobile edge computing enabled internet of things. *IEEE Internet of Things Journal*, 6(3):4804–4814, 2019.

[5] Li Lin, Xiaofei Liao, Hai Jin, and Peng Li. Computation offloading toward edge computing. *Proceedings of the IEEE*, 107(8):1584–1607, 2019.

[6] Manuel Díaz, Cristian Martín, and Bartolomé Rubio. State-of-the-art, challenges, and open issues in the integration of internet of things and cloud computing. *Journal of Network and Computer Applications*, 67:99–117, 2016.

[7] Muhammad Asim, Yong Wang, Kezhi Wang, and Pei-Qiu Huang. A review on computational intelligence techniques in cloud and edge computing. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 4(6):742–763, 2020.

[8] Alessio Botta, Walter de Donato, Valerio Persico, and Antonio Pescapé. Integration of cloud computing and internet of things: A survey. *Future Generation Computer Systems*, 56:684–700, 2016.

[9] Milan Patel, Brian Naughton, Caroline Chan, Nurit Sprecher, Sadayuki Abeta, Adrian Neal, et al. Mobile-edge computing introductory technical white paper. *White paper, mobile-edge computing (MEC) industry initiative*, 29:854–864, 2014.

[10] Keyan Cao, Yefan Liu, Gongjie Meng, and Qimeng Sun. An overview on edge computing research. *IEEE Access*, 8:85714–85728, 2020.

[11] Cheng Zhang, Hailiang Zhao, and Shuiguang Deng. A density-based offloading strategy for iot devices in edge computing systems. *IEEE Access*, 6:73520–73530, 2018.

[12] Zaiwar Ali, Ziaul Haq Abbas, Ghulam Abbas, Abdullah Numani, and Muhammad Bilal. Smart computational offloading for mobile edge computing in next-generation internet of things networks. *Computer Networks*, 198:108356, 2021.

[13] Congshi Jiang, Yihong Li, Junlong Su, and Quan Chen. Research on new edge computing network architecture and task offloading strategy for Internet of Things. *Wireless Networks*, 2021.

[14] Ying Chen, Wei Gu, and Kaixin Li. Dynamic task offloading for internet of things in mobile edge computing via deep reinforcement learning. *International Journal of Communication Systems*, n/a(n/a):e5154.

[15] Wei Yu, Fan Liang, Xiaofei He, William Grant Hatcher, Chao Lu, Jie Lin, and Xinyu Yang. A survey on the edge computing for the internet of things. *IEEE Access*, 6:6900–6919, 2018.

[16] Chunli Ren, Guoan Zhang, Xiaohui Gu, and Yue Li. Computing offloading in vehicular edge computing networks: Full or partial offloading? In *2022 IEEE 6th Information Technology and Mechatronics Engineering Conference (ITOEC)*, volume 6, pages 693–698, 2022.

[17] Ying Chen, Ning Zhang, Yongchao Zhang, Xin Chen, Wen Wu, and Xuemin Shen. Energy efficient dynamic offloading in mobile edge computing for internet of things. *IEEE Transactions on Cloud Computing*, 9(3):1050–1060, 2021.

[18] Congfeng Jiang, Xiaolan Cheng, Honghao Gao, Xin Zhou, and Jian Wan. Toward computation offloading in edge computing: A survey. *IEEE Access*, 7:131543–131558, 2019.

[19] Xiaoheng Deng, Zihui Sun, Deng Li, Jie Luo, and Shaohua Wan. User-centric computation offloading for edge computing. *IEEE Internet of Things Journal*, 8(16):12559–12568, 2021.

[20] Yaru Fu, Xiaolong Yang, Peng Yang, Angus K. Y. Wong, Zheng Shi, Hong Wang, and Tony Q. S. Quek. Energy-efficient offloading and resource allocation for mobile edge computing enabled mission-critical internet-of-things systems. *EURASIP Journal on Wireless Communications and Networking*, 2021(1):26, 2021.

[21] Haodong Lu, Xiaoming He, Miao Du, Xiukai Ruan, Yanfei Sun, and Kun Wang. Edge qoe: Computation offloading with deep reinforcement learning for internet of things. *IEEE Internet of Things Journal*, 7(10):9255–9265, 2020.