Edge-CoCaCo: Toward Joint Optimization of Computation, Caching, and Communication on Edge Cloud

Min Chen, Yixue Hao, Long Hu, M. Shamim Hossain, and Ahmed Ghoneim

Abstract

With the development of recent innovative applications (e.g., augmented reality, natural language processing, and various cognitive applications), more and more computation-intensive and rich-media tasks are delay-sensitive. Edge cloud computing is expected to be an effective solution to meet the demand for low latency. By the use of content offloading and/or computation offloading, users' quality of experience is improved with shorter delay. Compared to existing edge computing solutions, this article introduces a new concept of computing task caching and gives the optimal computing task caching policy. Furthermore, joint optimization of computation, caching, and communication on the edge cloud, dubbed Edge-CoCaCo, is proposed. Then we give the solution to that optimization problem. Finally, the simulation experimental results show that compared to the other schemes, Edge-CoCaCo has shorter delay.

Introduction

Due to the rapid development of wireless technology and the Internet of Things (IoT), more and more mobile devices (smartphones, wearable devices, etc.) access wireless networks with various requirements in terms of bandwidth and computation. In the future, mobile devices will become more intelligent, while the applications deployed on them will require extensive computation power and persistent data access [1]. However, the development of these emerging applications and services is restricted by the limited computational capacity and battery life of those devices. If computation-intensive and rich-media tasks could be offloaded to the cloud for task execution, it would be able to overcome the limitation of insufficient computing ability of mobile devices. However, relatively long delay appears when mobile devices are connecting to the cloud through wireless networks [2], which is not suitable for delay-sensitive tasks such as transcoding for virtual reality (VR) streaming and image processing for augmented reality (AR) games.

Recently, edge cloud computing has provided computing services with short delay and high performance to users through computing nodes or servers deployed on the network edge in order to meet the computing requirements of delay-sensitive tasks [3, 4]. There are two major advantages of using the edge cloud:

- In contrast to local computing [5], edge cloud computing can overcome the restrictions of limited computation capacity on mobile devices.
- Compared to computation offloading toward the remote cloud [6], edge cloud computing can avoid the high latency that may be caused by offloading the task contents to the remote cloud.

Thus, edge cloud computing typically exhibits a better trade-off for delay-sensitive and computation-intensive tasks.

To the best of our knowledge, all previous work on edge cloud computing focuses on the following three aspects:

Content offloading, or edge caching. On this topic, various caching policies were proposed to reduce the latency and energy cost of users getting the requested content [7, 8].

Computation offloading. Its main design issues are when, what, and how to offload users' task from their device to the edge cloud in order to reduce the computation latency with energy saving [9]. For instance, Chen *et al*. proposed a scheme to schedule computation tasks on the edge cloud under the circumstance of user mobility [10].

Mobile edge computing. Its main concern is to deploy edge clouds near the base station [11]. By considering both communication and computing resources, optimal solutions were designed for reducing energy cost and latency [12]. However, there is no work considering computing task caching. In this article, we first introduce the new concept of computing task caching. Then joint optimization of computation, caching, and communication on the edge cloud, dubbed Edge-CoCaCo, is proposed.

In order to explain "computing task caching," and illustrate the difference between edge cloud computing and Edge-CoCaCo, we use the scenario of a typical video processing task, as shown in Fig. 1.

Edge cloud computing: As shown in Fig. 1a, a mobile device (e.g., smartphone, wearable device, vehicle, and cognitive device) offloads video decoding tasks T1, T2, T3, and T4 to the edge cloud through a cellular network or WiFi.

Min Chen, Yixue Hao, and Long Hu are with Huazhong University of Science and Technology; M. Shamim Hossain is with King Saud University; Ahmed Ghoniem is with King Saud University and Menoufia University.

Digital Object Identifier: 10.1109/MWC.2018.1700308

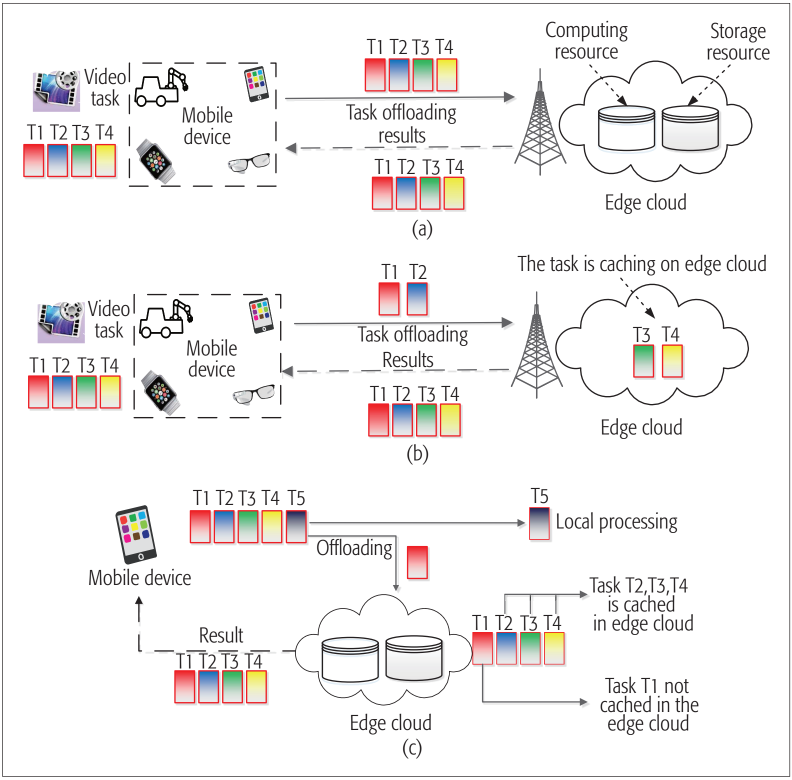

FIGURE 1. Illustration of: a) edge cloud computing; b) computing task caching; c) joint optimization of computation, caching, and communication on edge cloud (Edge-CoCaCo).

After edge cloud finishes the tasks, the computing results are fed back to the mobile device.

Computing task caching: As shown in Fig. 1b, among the requested video decoding tasks T1, T2, T3, and T4, let us assume that tasks T3 and T4 are already cached on the edge cloud. This is because the videos corresponding to T3 and T4 happen to be cached due to their high popularity. Therefore, the mobile device only needs to offload T1 and T2 to the edge cloud. The processing results of T1 and T2 are fed back to the mobile device after the edge cloud finishes the tasks, while the cached results of T3 and T4 are sent back to the mobile device immediately. Since there is no need to offload tasks T3 and T4, their communication cost is saved, while the task duration is shorter.

Edge-CoCaCo: In Fig. 1c, a mobile device has five delay-sensitivity tasks that need to be processed, namely T1, T2, T3, T4, and T5. In order to finish the required processing of tasks as soon as possible, the scheme of Edge-CoCaCo is designed as follows: Since T2, T3, and T4 are relatively popular tasks, they are already cached on the edge cloud, and the mobile user does not need to offload them again. Task T5 has a larger data size and needs a lot of data transmissions, but its requirement for computing resources is relatively small. Therefore, it is processed locally. Task T1 needs fewer data transmissions but is computation-intensive with higher need for computing resources. Therefore, it can be offloaded to the edge cloud for processing.

Hence, the computing task caching and Edge-CoCaCo need to be considered in order to achieve the lowest latency of the task execution at the edge cloud. However, there are three challenges:

- **How the task is cached:** Considering the computing task diversity (i.e., task popularity, data size, and the required computation capacity of the task), caching of computing tasks is still challenging.
- **How computation offloading works with computing task caching:** During the execution of a task, a computing task can be processed not only on the edge cloud, but also locally. However, when the task is cached in the edge cloud, it may not need to be processed locally. Therefore, it is challenging to make a computation offloading decision with computing task caching.
- **How to solve the joint optimization problem at Edge-CoCaCo:** Edge-CoCaCo includes a computing task caching problem and a task offloading problem that are hard to solve.

In this article, we first analyze the influence factors on caching policy and give the optimal computing task caching policy. Then we solve the optimization problem of Edge-CoCaCo using an alternating iterative algorithm. The simulation experiment shows that processing delay of tasks can be decreased significantly by deploying the computing task caching and computing offloading strategy reasonably. In summary, the main contributions of this article include:

- Considering task diversity and whether tasks can be cached on the edge cloud, there are three possibilities:
  –The computing task is not cached.
  –The computing task is cached.
  –The computing task result is cached.
  In terms of the deployment scheme of computing task caching, the experiment shows that caching of computing tasks relates to the popularity and size of task contents, as well as the required computation capacity of tasks.
- We propose Edge-CoCaCo to achieve the lowest delay of task processing, which includes the computing task caching placement and task offloading decision optimization problems.
- Through joint optimization of communication, caching, and computing on the edge cloud, we develop innovative caching and offloading schemes of computing tasks. The experimental result shows that the delay of computing tasks in the Edge-CoCaCo scheme is the shortest.

The article is organized as follows. In the next section, the architecture and problem formulation of computing task caching are described. Then the Edge-CoCaCo model and problem formulation are presented. After that, the obtained experimental results and related discussions are given. Finally,we conclude the article.

Architecture and Problem Formulation of Computing Task Caching

In this section, we introduce the architecture of computing task caching and propose a computing task caching strategy.

From Content Caching to Computing Task Caching

In recent years, content caching has been studied extensively, including caching policy, the distribution of caching contents, and so on [13, 14]. In content caching, a content provider can cache popular content on the edge cloud to reduce the delay of user requesting content.

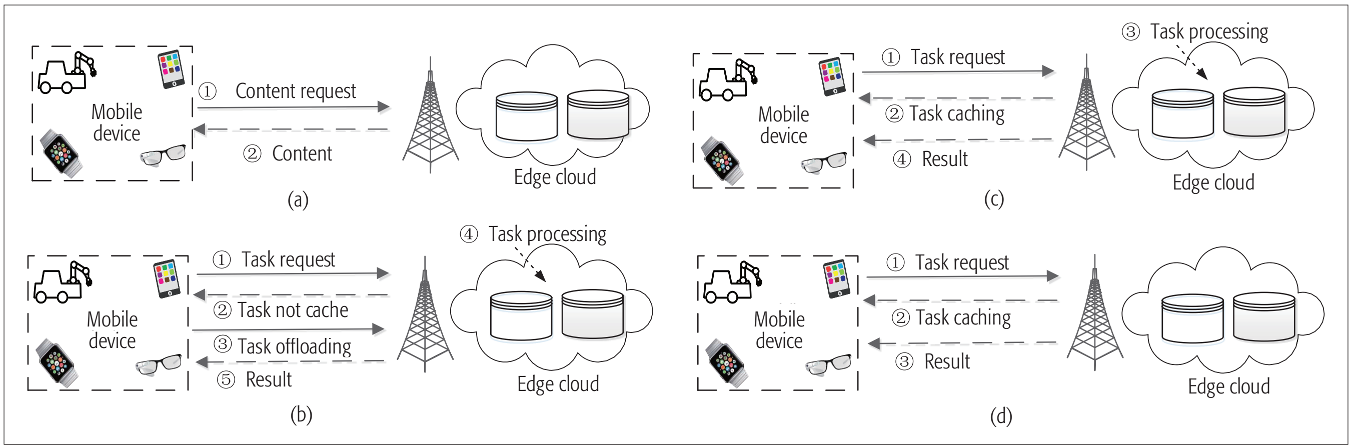

FIGURE 2. Illustration of content and task request process: a) content caching; b) computing task not cached; c) computing task cached; d) computing task result cached.

The specific process is explained below. When a mobile device requests a content, the request will go to the edge cloud. If the edge cloud has cached the content, it will transmit the content to the user who is requesting the content, which is shown in Fig. 2a. In computing task caching, by considering task diversity, we divide the computing task processing on the edge cloud in three situations.

Computing task not cached: The tasks of a mobile device that need to be processed are not cached on the edge cloud. The specific process is shown in Fig. 2b. Namely, the mobile device requests a computing task that needs to be offloaded, and the edge cloud discovers that it does not cache the task. In that case, the mobile device needs to offload the computing task to the edge cloud first, and when the edge cloud finishes the task, it transmits the result back to the mobile device.

Computing task cached: The tasks of a mobile device that need to be processed are cached on the edge cloud. Alternatively, if other different cached tasks can be transformed to the requested ones, we also count the tasks as being indirectly cached. The specific process is shown in Fig. 2c, where the mobile device first requests the computing task that needs to be offloaded; then the edge cloud informs the user that the task exists on the edge cloud, and the mobile device does not have to offload the computing task to the edge cloud. Finally, when the edge cloud finishes task processing, it transmits the result to the mobile device.

Computing task result cached: The result of the task of a mobile device that needs to be processed is cached on the edge cloud. In this case, the control flow is as shown in Fig. 2d. Namely, the mobile device does not need to offload the task to the edge cloud, and the edge cloud does not need to process the task. The edge cloud just needs to transmit the task result to the mobile devices directly. This situation is similar to content caching.

In the case of computation-intensive and rich-media computing tasks, offloading and computing of tasks typically cause various delays. According to the above discussion, the delay of task offloading and computing can be reduced by caching the task that needs to be processed on the edge cloud or caching the task result after being processed on the edge cloud, so as to meet the requirement for short delay of the task.

Key Design Problem of Computing Task Caching

In this subsection, we consider the factors that affect computing task caching. Figure 1b shows an example of computing task caching. For instance, mobile devices have four video tasks that need to be decoded, T1, T2, T3, and T4. Among these tasks, T2 and T4 are relatively popular videos, while T1 and T3 are not popular videos. In this article, the number of requests is used to represent the popularity of a task (i.e., the number of requests for a task). Let us assume the popularity of T1, T2, T3, and T4 be 1, 4, 3, and 2, respectively. Let the size of data associated with the four tasks (i.e., T1, T2, T3, and T4) be 20 Mb, 5 Mb, 30 Mb, and 10 Mb, respectively. The required computation resources for the executions of the four tasks are 6 gigacycles, 2 gigacycles, 4 gigacycles, and 10 gigacycles. Let the data rate for delivering task contents be 20 Mb/s. Let the computation capacity of the edge cloud be 10 GHz.

There is a following question: Given the capacity of the edge cache as 40 Mb, what is the best allocation scheme of computing task caching? Based on the aforementioned parameter configurations, it can be calculated that 1.6 s will be reduced for T1 by use of computing task caching. Likewise, computing task caching for T2, T3, and T4 leads to a delay reduction of 1.8 s, 3.8 s, and 4.5 s, respectively. Thus, caching T3 and T4 is the optimal solution to achieve the lowest latency.

Thus, in the computing task caching strategy, there are still challenges related to computing task caching:

- Although the caching capacity and computing capability of the edge cloud are better than those of a mobile device, the edge cloud is still not able to cache and support all types of computing tasks.
- Compared to content caching, caching of a computing task needs to consider not only task popularity, but also data size and computing resource required for the task.

Therefore, the design of computing task caching strategy is a challenging issue.

The delay of task offloading and computing can be reduced by caching the task that needs to be processed on edge cloud or caching the task result after being processed on edge cloud, so as to meet the requirement for short delay of task computing.

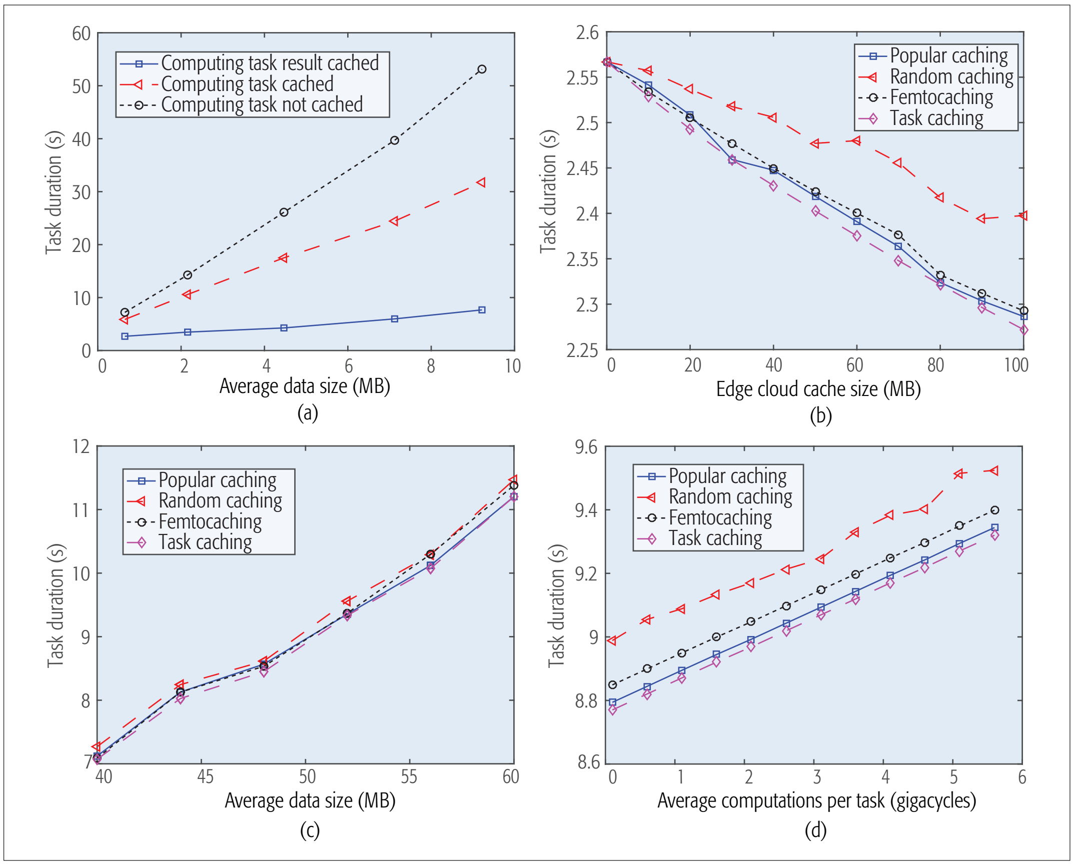

FIGURE 3. Computing task caching evaluation: a) task duration with different sizes in three cases: computing task not cached, computing task cached, and computing task result cached. Task duration achieved by popular caching, random caching, femtocaching, and task caching for various value of b) the edge cloud cache capacity; c) the average data size of per task; d) the average computation cacpacity per task. The default setting is n = 100, c_e = 100 MB, λ = 0.2; ω follows normal distribution with an average of 2 gigacycles per task; s follows uniform distribution with an average of 50 MB.

Computing Task Caching Model

In this subsection, we give the computing task caching strategy. We consider an edge computing ecosystem that includes multiple mobile devices and an edge cloud. Mobile devices can communicate with the edge cloud through wireless channel. The edge cloud is a small data center with computing and storage resources, where the computing resources provide task processing for mobile devices and storage resource provides for task content, processing code, and computation result caching.

We assume that mobile devices have n computing tasks that need to be processed, and the set of tasks is denoted as $\mathcal{Q} = \{Q_1, Q_2, ..., Q_n\}$. Since some computing tasks have higher popularity, they may be processed many times. For computing task Q_i, $Q_i = \{\omega_i, s_i, o_i\}$, where ω_i is the amount of computing resource required for task Q_i (i.e., the total number of CPU cycles needed to complete the task), and s_i is the data size of computation task Q_i, that is, the amount of data content (e.g., the processing code and parameter(s)) to be delivered toward the edge cloud. Finally, o_i represents the data size of the task result. For instance, in the video decoding case, ω_i is the computing resource needed for video decoding, s_i is the video data size, and o_i is the data size of the decoded video. Since the computing and capacity of the edge cloud is limited, we assume that the cache size and computing capacity of the edge cloud are c_e and c_s, respectively.

In this article, for the sake of simplicity, we divide tasks into two categories: computing tasks cached on the edge cloud and computing tasks not cached on the edge cloud. We define the integer decision variable, $x_i \in \{0, 1\}$ that indicates whether task Q_i is cached at the edge cloud (x_i = 1) or not (x_i = 0). We also define the respective task caching placement strategy: $\mathbf{x} = (x_1, x_2, ..., x_n)$. In addition, when the computing task is cached on the edge cloud, we define T_i^c as the task duration of task Q_i being processed on the edge cloud, and T_i^{nc} as the task duration of task Q_i being processed on the edge cloud when the computing task is not cached. We give a more detailed discussion of task duration later.

Consequently, the problem of determining the computing task caching placement strategy that minimizes the task duration can be defined as follows:

$$\begin{aligned} &\underset{\mathbf{x}}{\text{minimize}} \sum_{i=1}^{n} p_i \left[x_i T_i^c + (1 - x_i) T_i^{nc} \right] \\ &\text{subject to: } \sum_{i=1}^{n} x_i s_i \le c_e \end{aligned} \quad (1)$$

where p_i denotes the probability of requests for computing task Q_i. The objective function computes the minimal processing delay of the task through deployment of task caching. The constraint condition is that the data size of a cached computing task cannot exceed the largest caching capacity.

For solving the optimization problem, since the objective function and constraint are linear, the optimal problem is a 0 – 1 integer linear optimization problem. By the use of a branch and bound algorithm [15], the optimal solution can be obtained. The optimal result represents the computing task caching strategy of the edge cloud.

Edge-CoCaCo Model

In this section, we propose the Edge-CoCaCo model in order to finish task computing more quickly. The Edge-CoCaCo model deals with the following two problems:

- Computing task caching placement problem: It refers to the decision of whether to cache the computing tasks on the edge cloud or not.
- Task offloading problem: It refers to the decision of which task should be processed locally and which tasks should be processed on the edge cloud.

Communication Model

We first introduce the communication model and give the uplink data rate when a mobile device offloads a task on the edge cloud. Let h and p denote channel power gain and transmission power of a mobile device, respectively. Then the uplink data rate of task Q_i can be obtained as follows:

$$r = B\log_2\left(1+\frac{ph^2}{\sigma^2}\right),$$

where σ^2 denotes the noise power, and B represents the channel bandwidth.

Computation Model

Now we introduce the computation offloading model. In this article, we assume that a computing task is divisible, which means that a task can be divided into two or more parts. Therefore, it can be processed locally or on the edge cloud. For task Q_i, we define the decision variable $\alpha_i \in [0, 1]$. When $\alpha_i = 1$, task Q_i is processed locally; when $\alpha_i = 0$, task Q_i is offloaded to the edge cloud; when $\alpha_i \in (0, 1)$, part α_i of task Q_i is processed locally, and part $1 - \alpha_i$ is offloaded to the edge cloud. We define the respective task offloading policy: $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_n\}$. The specific computing delay is shown as follows.

Local Computing: For local task computing, we define f_l as the CPU computing capability of a mobile device. Thus, the local execution time of task Q_i can be expressed as $T_i^l = \omega_i/f_l$.

Edge Cloud Computing: For task computing on the edge cloud, we define f_c as a CPU computing capability of the edge cloud. In this case, the task duration consists of time consumed by three procedures:

- Time consumed when the mobile device offloads the task
- Time consumed when the computation task is processed on the edge cloud
- Time consumed to feed back the computing results to the mobile device

Since data size after task processing is generally smaller than before processing [12], and the downlink rate from the edge cloud to the mobile device is higher than the uplink rate from the mobile device to the edge cloud, we denote the time consumed to feed back the computing result to the mobile device as a variable $\xi_i(o_i)$, and it relates to the data size of the computing result. Therefore, we can obtain the task duration of task Q_i on the edge cloud as follows:

$$T_i^e = \frac{\omega_i}{f_c} + \frac{s_i}{r} + \xi_i(o_i).$$

Thus, when the task is cached in the edge cloud, the task duration can be expressed as

$$T_i^c = \frac{\omega_i}{f_c} + \xi_i(o_i).$$

In comparison, when the task is not cached, the task duration can be expressed as: $T_i^{nc} = T_i^e$.

According to the above discussion, for task Q_i, considering computing task caching, and local and edge cloud computing, the total task duration of Q_i can be obtained as follows:

$$T_i = x_i\left(\frac{\omega_i}{f_c} + \xi_i(o_i)\right) + (1-x_i)\left[\alpha_i T_i^l + (1-\alpha_i)T_i^c\right] \quad (2)$$

Edge-CoCaCo Model

Our objective is to minimize the task duration of all tasks under the caching capacity constraints, which include the computing task caching placement problem and task offloading problem. The problem can be expressed as follows:

$$\underset{x,\alpha}{\text{minimize}} \sum_{i=1}^{n} p_i T_i$$
$$\text{subject to} \sum_{i=1}^{n} x_i s_i \le c_e \quad (3)$$

where the objective function computes the minimal task duration through deployment of computing task caching and task offloading. The constraint condition is that the data size of a cached computing task cannot exceed the edge cloud caching capacity.

For solving the optimization problem, since the objective function and constraint are linear, the optimal problem is a mixed integer linear optimization problem. Since the objective function with respect to α is a linear optimization function, the optimal solution can be obtained based on the Karush-Kuhn-Tucker (KKT) condition [15]. Then the objective function is transformed into a 0 – 1 linear programming problem with respect to $\mathbf{x}$. By the use of the branch and bound algorithm, the solution can be calculated. Thus, the linear iterative algorithm can be utilized and obtain the approximate optimal solution. The optimal result represents the computing task caching and task offloading strategy of the edge cloud.

Performance Evaluation

In this section, we evaluate the performance of computing task caching and the Edge-CoCaCo model. We assume that system bandwidth B and transmitting power p of the mobile device are 1 MHz and 0.2 W, respectively. The corresponding

Mobile devices can communicate with the edge cloud through wireless channel. The edge cloud is a small data center with computing and storage resources, whereas computing resources provide task processing for mobile devices and storage resource provides task content, processing code, and computation result caching.

The task caching strategy proposed in this article is optimal, and the random caching strategy is relatively poor. This is because the random cache fails to consider the number of task requests and also fails to consider the task computation amount and data size of computing task in the case of the computing task caching.

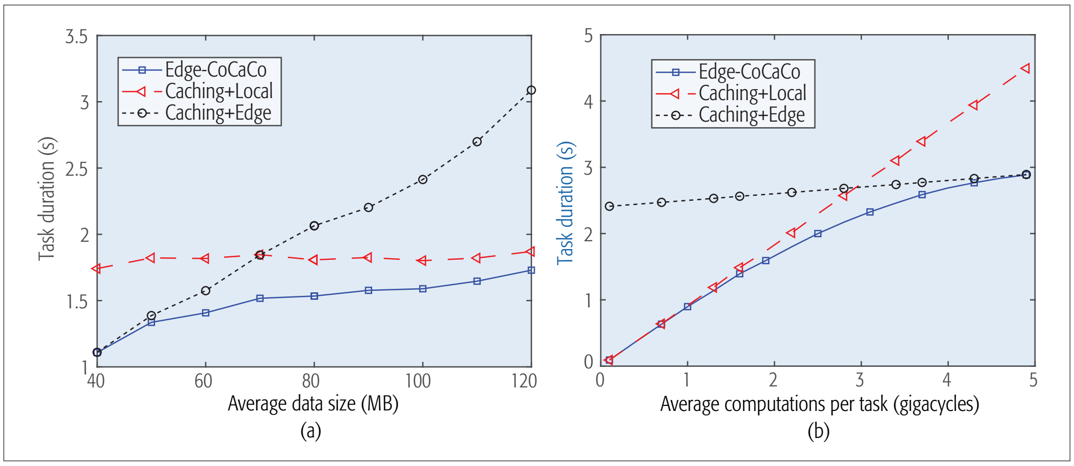

FIGURE 4. Edge-CoCaCo model evaluation: a) task duration over different data sizes of computation task; b) task duration over different required computation capacity of tasks. The default setting is $n = 100$, $c_e = 100$ MB, $\lambda = 0.2$; ω follows normal distribution with an average of 2 gigacycles per task; s follows uniform distribution with an average of 10 MB.

Gaussian channel noise σ^2 and channel power gain h are 10^{-9} W and 10^{-5}, respectively. For task Q_i, we assume that required computing capacity ω_i and data size s_i are generated by a probability distribution [4]. For the task popularity, we assume that the number of task requests follows the Zipf distribution with parameter λ. Furthermore, we assume that the computing capabilities of the edge cloud and mobile device are 10 GHz and 1 GHz, respectively.

Computing Task Caching Evaluation

We first describe the comparison of task duration in terms of three cases: computing task not cached, computing task cached, and computing task result cached. As can be seen in Fig. 3a, when the computing task result is cached, the computation delay is the minimum. When the task is not cached, the computation delay is the maximum. Thus, computing task caching can reduce the delay of the task. From Fig. 3a, we can also see that the bigger the computing task data size, the longer the delay. This is because when the task transmission rate is constant, the larger the task size, the longer the delay.

To evaluate the computing task caching strategy, we compare the task caching strategy proposed in this article with the following caching strategies.

- Popular caching scheme: The edge cloud caches the computing task with the maximum number of requests until reaching the caching capacity of the edge cloud.
- Random caching scheme: The edge cloud caches the computing task randomly until reaching the caching capacity of the edge cloud.
- Femtocaching scheme: The caching capacity of the computing task is set as being empty at the start. Iteratively, add a task to a cache that minimizes the total delay of computing task processing (i.e., Eq. 1) until reaching the caching capacity of the edge cloud.

From Figs. 3b–3d, the task caching strategy proposed in this article is optimal, and the random caching strategy is relatively poor. This is because the random cache fails to consider the number of task requests as well as the task computation amount and data size of the computing task in the case of computing task caching. The popular caching strategy only considers the number of task requests, rather than considering the data size and computation amount of task comprehensively. Femtocaching considers the computation amount, data size, and request of the task to a certain extent. From Fig. 3b, we can also see that the larger the cache capacity of the edge cloud, the smaller the task delay. This is because the capacity of the cache becomes larger, which leads to caching more tasks; thus, task duration can be reduced. From Figs. 3c and 3d, we can see that the impact of task data size on the algorithm is less than the impact of computing capacity on the algorithm.

Edge-CoCaCo Model Evaluation

To evaluate the Edge-CoCaCo model, we compare the Edge-CoCaCo with the following caching and computing strategy.

- Caching+Local: First, the computation tasks are cached according to the caching policy proposed in this article, and second, the tasks that are not cached will only be handled locally rather than being offloaded to the edge cloud for processing.
- Caching+Edge: The not cached tasks are offloaded to the edge cloud for processing relative to Caching+Local. From Fig. 4 we can see that the task duration of the proposed Edge-CoCaCo model is lowest, that is, reasonably deploying computation task caching placement and task offloading can effectively reduce the computation latency of a task.

From Fig. 4a, we can also see that the difference of task duration between Edge-CoCaCo and Caching+Edge is little when the data size of tasks is small. Also, the difference between Edge-CoCaCo and Caching+Local is little when the data size is large. Thus, we can conclude that under the same required computation capacity of tasks, the tasks should be processed at the edge cloud when the data size of tasks is small; conversely, if the data size is large, the tasks should be handled locally. Similarly, we can conclude from Fig. 4b that under the same data size of tasks, the tasks should be handled locally when the required computation capacity is relatively small, and processed at the edge cloud when the computation capacity is large.

Conclusion

In this article, we first propose computing task caching on the edge cloud, analyze the influence factors of task popularity and size of task content, as well as the required computation capacity on caching strategy, and provide the optimal caching strategy of a computing task. To the best of our knowledge, this is the first study of computing task caching in the edge cloud. Furthermore, we propose Edge-CoCaCo to meet the low delay demands of computation-intensive and rich-media tasks. Simulation results have shown that our proposed scheme has less delay compared to other schemes. For future work, we will consider multiple edge cloud computing task caching strategies and use real traces to do experiments.

Acknowledgement

The authors extend their appreciation to the Deanship of Scientific Research at King Saud University, Riyadh, Saudi Arabia, for funding this work through the research group Project no. RGP-229. Long Hu is the corresponding author.

References

[1] M. Chen and Y. Hao, "Task Offloading for Mobile Edge Computing in Software Defined Ultra-dense Network", *IEEE JSAC*, vol. 36, no. 3, 2018, pp. 587–97.

[2] X. Hou *et al.*, "Vehicular Fog Computing: A Viewpoint of Vehicles as the Infrastructures," *IEEE Trans. Vehic. Tech.*, vol. 65, no. 6, 2016, pp. 3860–3873.

[3] M. Patel *et al.*, "Mobile-Edge Computing Introductory Technical White Paper," Sept. 2014; http://www.etsi.org/technologies-clusters/technologies/mobile-edge-computing.

[4] L. Tong, Y. Li, and W. Gao, "A Hierarchical Edge Cloud Architecture for Mobile Computing," *Proc. IEEE INFOCOM*, San Francisco, CA, 2016, pp. 399–400.

[5] M. Chen *et al.*, "On the Computation Offloading at Ad Hoc Cloudlet: Architecture and Service Modes," *IEEE Commun. Mag.*, vol. 53, no. 6, June 2015, pp. 18–24.

[6] M. Barbera *et al.*, "To Offload or Not to Offload? The Bandwidth and Energy Costs of Mobile Cloud Computing," *Proc. IEEE INFOCOM*, Turin, Italy, 2013, pp. 1285–93.

[7] X. Li *et al.*, "Collaborative Multi-tier Caching in Heterogeneous Networks: Modeling, Analysis, and Design," *IEEE Trans. Wireless Commun.*, vol. 16, no. 10, 2017, pp. 6926–39.

[8] M. Chen *et al.*, "Green and Mobility-Aware Caching in 5G Networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 12, 2017, pp. 8347–61.

[9] W. Shi, *et al.*, "Edge Computing: Vision and Challenges," *IEEE Internet of Things J.*, vol. 3, no. 5, 2016, pp. 637–46.

[10] M. Chen *et al.*, "Mobility-Aware Caching and Computation Offloading in 5G Ultra-Dense Cellular Networks," *Sensors*, vol. 16, no. 7, 2016, pp. 974–87.

[11] P. Mach and Z. Becvar, "Mobile Edge Computing: A Survey on Architecture and Computation Offloading," *IEEE Commun. Surveys & Tutorials*, vol. 19, no. 3, 2017, pp. 1628–56.

[12] X. Chen, *et al.*, "Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing," *IEEE-ACM Trans. Net.*, vol. 24, no. 5, 2016, pp. 2795–2808.

[13] X. Wang *et al.*, "Serendipity of Sharing: Large-Scale Measurement and Analytics for Device-to-Device (D2D) Content Sharing in Mobile Social Networks," *Proc. IEEE SECON*, San Diego, CA , 2017. DOI: 10.1109/SAHCN.2017.7964925.

[14] X. Wang *et al.*, "D2D Big Data: Content Deliveries over Wireless Device-to-Device Sharing in Realistic Large-Scale Mobile Networks," *IEEE Wireless Commun.*, vol. 25, no. 1, 2018, pp. 1–10.

[15] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge Univ. Press, 2004.

Biographies

Min Chen [SM'09] (minchen2012@hust.edu.cn) has been a full professor in the School of Computer Science and Technology at HUST since February 2012. He is Chair of the IEEE Computer Society STC on Big Data. His Google Scholars Citations reached 12,800+ with an h-index of 55. He received the IEEE Communications Society Fred W. Ellersick Prize in 2017. His research focuses on cyber physical systems, IoT sensing, 5G networks, healthcare big data, and cognitive computing.

Yixue Hao (yixuehao@hust.edu.cn) received his B.E. degree from Henan University, China, and his Ph.D degree in computer science from HUST in 2017. He is currently working as a postdoctoral scholar in the School of Computer Science and Technology at HUST. His research includes 5G networks, the Internet of Things, and mobile cloud computing.

Long Hu (longhu@hust.edu.cn) has been a lecturer in the School of Computer Science and Technology, HUST, since 2017. He was a visiting student at the Department of Electrical and Computer Engineering, University of British Columbia from August 2015 to April 2017. His research includes the Internet of Things, software defined networking, caching, 5G, body area networks, body sensor networks, and mobile cloud computing.

M. Shamim Hossain [SM'09] (mshossain@ksu.edu.sa) is a professor with the Department of Software Engineering, College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia. He has authored or co-authored more than 175 publications. He is the recipient the ACM TOMM Nicolas D. Georganas Best Paper Award. He currently serves on the Editorial Board of *IEEE Multimedia*. His research focuses on social media, the Internet of Things, cloud and multimedia for healthcare, and smart health.

Ahmed Ghoneim (ghoneim@ksu.edu.sa) received his Ph.D. degree in software engineering from the University of Magdeburg, Germany, in 2007. He is currently an assistant professor with the Department of Software Engineering, King Saud University. His current research interests include address software evolution, service oriented engineering, software development methodologies, quality of service, net-centric computing, and human-computer interaction.

Recently, the edge cloud computing has provided computing services with short delay and high performance to users through the computing nodes or servers deployed on the network edge in order to meet the computing requirements of delay-sensitive tasks.