様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

令和　２ 年　 ７ 月　 ７ 日現在

機関番号：６２６１５
研究種目：研究活動スタート支援
研究期間：2018～2019
課題番号：１８Ｈ０６４７１・１９Ｋ２１５３９

研究課題名（和文）Cotask-Aware Offloading and Scheduling in Mobile-Edge Computing Systems

研究課題名（英文）Cotask-Aware Offloading and Scheduling in Mobile-Edge Computing Systems

研究代表者
江　易翰（Chiang, Yi-Han）

国立情報学研究所・アーキテクチャ科学研究系・特任助教

研究者番号：１０８２４１９６

交付決定額（研究期間全体）：（直接経費）　　2,300,000円

研究成果の概要（和文）：モバイルエッジコンピューティングシステムにおけるコタスク機能は、複数のサブタスクで構成され、すべてのサブタスクの実行が完了し、最終的な結果が返された場合にのみ完了となります。この研究では、平均コタスク完了時間を最小化することを目的とする混合整数非線形計画を作成しました。この問題を解決するために、線形計画の丸め手法に基づいてコタスクをオフロードし、最先のコタスク到着優先の規則に従ってスケジュールを設計しました。この二つのアルゴリズムによって、共同で達成される近似比を求めることができます。設計したアルゴリズムの効果はシミュレーションとテストベッドで検証しました。

研究成果の学術的意義や社会的意義
コタスクの存在は、計算資源を効果的に活用するために、サブタスクのオフロードとスケジューリングの共同設計を必要とします。この研究では、混合整数非線形計画として定式化し、その問題のNP困難も証明しました。そして、設計したコタスクのオフロードとスケジューリングのアルゴリズムに基づいて、到達した近似率はシステムの計算能力と無線遅延によって決まります。テストベッドとシミュレーションの結果は、設計されたアルゴリズムがコタスクをバランスよくオフロードして、ほぼ同じタイミングでスケジュールできるので、ローカル計算とエッジ計算のトレードオフが改善されることを示し、ネットワークの計算資源を効率的に使用できました。

研究成果の概要（英文）：Mobile edge computing (MEC) systems provide mobile devices (MDs) with low-latency cloud services by deploying edge servers (ESs) in the vicinity. In fact, various mobile applications may generate cotasks, each of which is completed only if all its constituent tasks are finished. In this research, we investigate the problem of joint cotask-aware offloading and scheduling in MEC systems (Cool-Edge), and we formulate it as a mixed integer non-linear program (MINLP) to minimize average cotask completion time (ACCT). To cope with the Cool-Edge problem, we propose two low-complexity algorithms to offload cotasks based on an LP rounding technique and schedule them according to an earliest-cotask-arrival-first rule, respectively, and we further prove the approximation factor jointly achieved by the two algorithms. Finally, we conduct testbed experiments and simulations to demonstrate the effectiveness of our proposed solution, and we also show how ACCT varies with the network environment.

研究分野：情報科学、情報工学およびその関連分野

キーワード：モバイルエッジコンピューティング　オフロード　スケジューリング　コタスク　混合整数非線形計画

様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９（共通）

１．研究開始当初の背景

Recent advances show that various emerging delay-sensitive applications have proliferated and become parts of mobile devices (MDs) in recent years. However, resource-constrained MDs (due to their limited battery lives and computing resources) may have difficulties in performing computation-intensive jobs while meeting stringent delay requirements. Therefore, mobile edge computing (MEC) systems [1]–[5] came into the world to resolve this problem by deploying edge servers (ESs) on the network edge, thereby providing low-latency cloud services to MDs.

To fully grasp the potentials of MEC systems, leveraging parallel processing that admits jobs to be processed on independent machines is an effective way to exploit the networked computing resources. In [6], Kosta *et al.* proposed a framework to facilitate on-demand resource allocation and parallelism for managing virtual machines in the cloud. In [7], Jia *et al.* designed an online task offloading algorithm for concurrent tasks, where the edge and cloud computation can be performed in parallel. In [8], Yang *et al.* investigated the problem of joint computation partitioning and resource allocation, where each user's application is divisible into modules that can be processed concurrently. In fact, mobile applications may generate data-parallel jobs as *cotasks*: each cotask is completed only if all its constituent sub-tasks are finished and returned to the hosting MD, and its completion time is determined by the latest completion time among all of the constituent sub-tasks. Despite the above works that guide us to leverage parallel processing to accelerate computation for MDs, none of them can tell us how to properly allocate computing resources to cotasks in MEC systems.

２．研究の目的

The purpose of this research is to investigate the problem of cotask-aware offloading and scheduling in MEC systems and show how to better utilize the networked computing resources. To this end, we plan to take the following actions for the success of this research.

- To study how the cotask feature matters in the design of MEC offloading and scheduling.
- To formulate the Cool-Edge problem as an MINLP to minimize ACCT and show its NP-hardness.
- To propose approximation algorithms to offload cotasks based on the LPR technique and employ the ECAF rule for scheduling, respectively.
- To prove that the approximation factor of $4\kappa(1+\epsilon)+2\rho$ is achievable for any $\epsilon > 0$, where κ and ρ refer to the ratios of machine computability and radio access delays, respectively.
- To conduct testbed experiments and simulations to show the effectiveness of our proposed solution.

３．研究の方法

In this research, we investigate the problem of cotask-aware offloading and scheduling in MEC systems (Cool-Edge). The Cool-Edge problem can be characterized as a mixed integer non-linear program (MINLP), the objective of which is to minimize average cotask completion time (ACCT). Due to the NP-hardness of the Cool-Edge problem, we are motivated to design approximation algorithms with a provable performance guarantee. Our proposed solution consists of two consecutive phases. First, we propose the cotask-aware offloading algorithm (denoted by CoOFLD) to minimize the makespan for each cotask, and then apply an LP rounding (LPR) technique that explores a perfect matching from a constructed bipartite graph. Next, we design the cotask-aware scheduling algorithm (denoted by CoSKED) to schedule cotasks according to the earliest-cotask-arrival-first (ECAF) rule in the way that the earliest cotask arriving at the network edge will be processed first by all ESs. We conduct testbed experiments to assert the practicability of our proposed solution. In larger network scales, we run simulations to demonstrate the achieved ACCT as well as the impacts of the constitution of cotasks and the distribution of computing resources.

４．研究成果

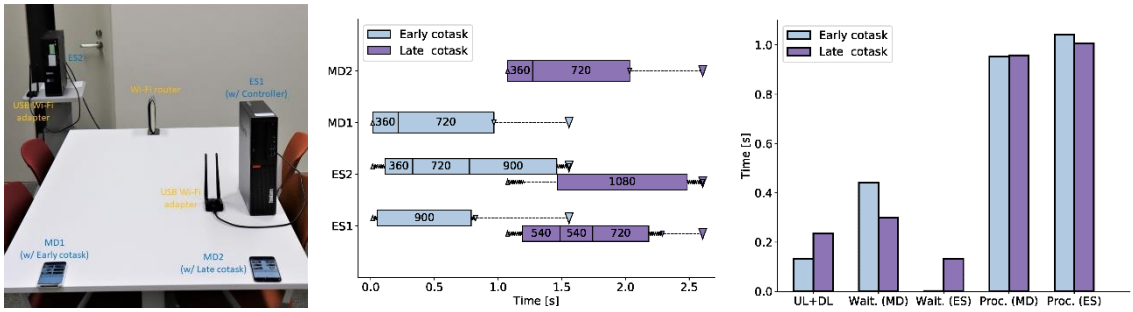
To demonstrate the ACCT performance of our proposed solution, we consider various offloading and scheduling schemes for comparison (see TABLE I), including CoCo (the proposed CoOFLD and CoSKED), MC/RC (existing offloading schemes combined with CoSKED), CS/CL (CoOFLD combined with existing scheduling schemes), and NN (the baseline scheme with merely local computation).

・*Testbed Experiments*

We construct an MEC system (see FIGURE 1(a)) of 2 ESs (desktop computers with Intel Core i76700K processors) and 2 MDs (smartphones with the Qualcomm Snapdragon 835 mobile PC platform), where each ES is equipped with a USB Wi-Fi adapter (Alfa Network AWUS036ACH). The controller is implemented in one of the ESs, and it connects ESs through a Wi-Fi router (Aterm WG2600HP2).

A simple application is made to capture the cotask feature in the MEC system: the mission of each cotask is to count the detected frontal faces, and the constituent sub-tasks represent a set of images for counting. To this end, we collect 10^3 images from Pexels [9] and use Dlib [10] to detect frontal faces. The set of collected images consists of five different image qualities. We measure the elapsed processing time (see TABLE II), which is shown to grow with image qualities. These measurement results will further be used to set the processing time of tasks as well as the ratio of machine computability in our simulations.

In our testbed (see FIGURE 1(a)), we randomly choose 6 images to form a cotask. FIGURE 1(b) illustrates the spatial-temporal dynamics of the two cotasks. It can be seen that CoOFLD produces a balanced offloading while CoSKED enables the early cotask to be scheduled first, which conforms to our designed solution. FIGURE 1(c) indicates the corresponding elapsed time on average, from which we see that all PUs are efficiently utilized and no much waiting time on ESs, which affirm the practicability of our proposed solution. Moreover, we observe that the effect of shared network bandwidth (in terms of uplink and downlink delays) is not pronounced, since the effect of shared computing resources (in terms of processing and waiting time) plays a more dominant role in the resulting CCT.



(a) (b) (c)

FIGURE 1: Testbed environment and the measurement results under CoCo.
(a) The testbed setup. (b) The proc. time of the two cotasks. (c) The elapsed time profile.

TABLE I: The offloading and scheduling comparison schemes.

Acronyms	MC	RC	CoCo	CS	CL	NN
Offloading schemes	MT[1]	RR[2]	CoOFLD	CoOFLD	CoOFLD	∅[5]
Scheduling schemes	CoSKED	CoSKED	CoSKED	SPT[3]	LPT[4]	∅[5]

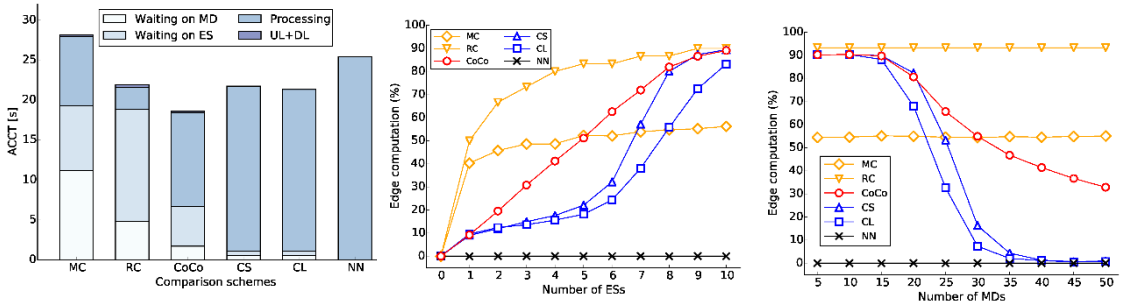
[1]MC: Each task is processed by the PU that minimizes its TCT.
[2]RR: Each MD offloads tasks in a round robin fashion.
[3]SPT: Each ES schedules the shortest-processing-time task first.
[4]LPT: Each ES schedules the longest-processing-time task first.
[5]∅: No MDs offload tasks and hence ESs have nothing to schedule.

TABLE II: The elapsed time for processing.

Image quality	640×360	960×540	1280×720	1600×900	1920×1080
ES's Processing time [s]	0.1145	0.2556	0.4446	0.6966	0.9889
MD's Processing time [s]	0.1893	0.4319	0.7572	1.1863	1.7087
Machine computability ratio	1.653	1.689	1.703	1.703	1.728

・*Simulations*

Here, we consider an MEC system of 5 ESs and 15 MDs. Each cotask is formed by randomly selected 30 images, and the release time of cotasks is determined by a Poisson process with the mean of 0.1 seconds. The uplink and downlink delays between an MD and an ES are both uniformly distributed in the range of [0.1,0.3] seconds. The error-tolerant parameter is set to $1/10^8$. According to the measurement results in TABLE II, we set the ratio of machine computability to 1.7 unless stated otherwise. All results are averaged over 100 iterations.

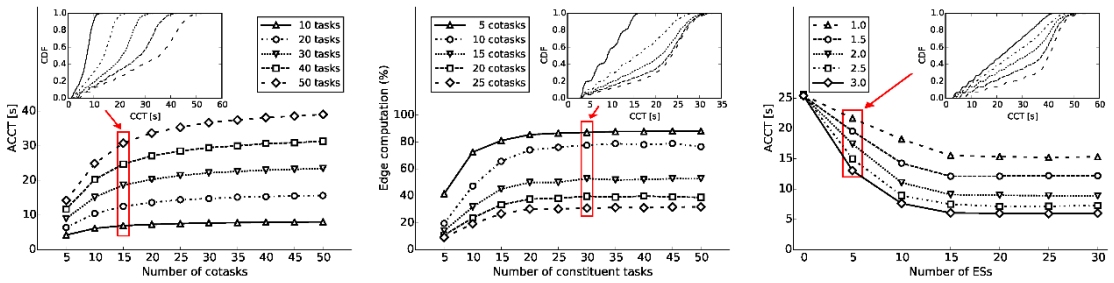


(a) (b) (c)

FIGURE 2: The achieved ACCT and the exploitation of edge computation. (a) The elapsed time profile. (b) The impact of the number of ESs. (c) The impact of the number of MDs.

Our proposed solution is shown to outperform the comparison schemes in terms of ACCT. In FIGURE 2(a), we see that CoCo strikes a good balance between processing time and waiting time. Specifically, CoCo gains from lower waiting time as compared with MC and RC, since it intends to offload tasks to PUs in a balanced way to avoid stacking tasks on a processing unit (PU) (since both MDs and ESs are capable of processing sub-tasks, we call them PUs for brevity). On the other hand, CoCo enables cotasks to be scheduled in the same order on all ESs, and hence it prevents the early completed tasks of a cotask from waiting for the completion of other constituent sub-tasks, thereby achieving lower ACCT with respect to CL, CS and NN. Altogether, CoCo offloads tasks in a balanced way and schedule cotasks synchronously, and therefore it performs better than the comparison schemes.

The exploitation of edge computation provides evidences to the achieved ACCT gain. FIGURE 2(b) and 2(c) show that the exploitation of edge computation increases with the number of ESs and decreases with that of MDs, respectively. This is because the benefit of edge computation is pronounced if there are not too many tasks queued on ESs. If the number of MDs are relatively larger than that of ESs, the benefit of edge computation diminishes since the offloaded tasks will encounter large waiting time, and hence less edge computation is preferable. Another observation is that the exploitation of edge computation essentially does not approach to 100%. The reason is that each hosting MD needs to wait for the last returned task. If the hosting MD can process a small portion of its tasks locally while waiting for the offloaded tasks to return, its CCT can be further reduced as compared with doing all computation on the edge.



(a) (b) (c)

FIGURE 3: The achieved ACCT and the exploitation of edge computation under CoCo. (a) The impact of the number of cotasks. (b) The impact of the number of sub- tasks. (c) The impact of the number of ESs.

The numbers of constituent sub-tasks, cotasks and ESs have prompt effects on ACCT performance. In FIGURE 3(a) and 3(b), we see that ACCT increases with the numbers of constituent sub-tasks and cotasks. This is because there are more tasks to be processed, and hence longer elapsed time for processing can be observed. In addition, as the number of tasks per cotask increases, the exploitation of edge computation first increases and then saturates. The saturation happens since plenty of tasks are waiting for processing on ESs, from which we see that a balanced local and edge computation attains better ACCT.

The more the number of ESs or the higher ratio of machine computability, the lower the achieved ACCT. In FIGURE 3(c), it can be seen that ACCT is lower whenever there are abundant computing resources. Another notable fact is the distribution of computing resources. Given a fixed amount of computing resources, it is better to deploy fewer but more powerful ESs. This is because the offloaded tasks may not be perfectly uniform among PUs since they cannot be divided into fractional pieces. Therefore, from the perspective of PUs, fewer PUs gives rise to more uniform offloading and lower ACCT.

REFERENCES

[1] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing – a key technology towards 5G," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, Sept. 2015.

[2] D. Sabella, A. Vaillant, P. Kuure, U. Rauschenbach, and F. Giust, "Mobile-edge computing architecture: The role of MEC in the Internet of Things," *IEEE Consum. Electron Mag.*, vol. 5, no. 4, pp. 84–91, Oct. 2016.

[3] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, 3rd Quart. 2017.

[4] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1657–1681, 3rd Quart. 2017.

[5] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart. 2017.

[6] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Proc. IEEE INFOCOM*, Mar. 2012.

[7] M. Jia, J. Cao, and L. Yang, "Heuristic offloading of concurrent tasks for computation-intensive applications in mobile cloud computing," in *Proc. IEEE INFOCOM Wkshps*, Apr. 2014.

[8] L. Yang, B. Liu, J. Cao, Y. Sahni, and Z. Wang, "Joint computation partitioning and resource allocation for latency sensitive applications in mobile edge clouds," in *Proc. IEEE CLOUD*, Jun. 2017.

[9] (2018, Jul.) Pexels. [Online]. Available: http://www.pexels.com/

[10] (2018, Jul.) Dlib. [Online]. Available: http://dlib.net/

５．主な発表論文等

〔雑誌論文〕　計2件（うち査読付論文　2件／うち国際共著　1件／うちオープンアクセス　1件）

１．著者名 Chao Zhu, Yi-Han Chiang, Abbas Mehrabi, Yu Xiao, Antti Yla-Jaaski, and Yusheng Ji	４．巻 68
２．論文標題 Chameleon: Latency and Resolution Aware Task Offloading for Visual-Based Assisted Driving	５．発行年 2019年
３．雑誌名 IEEE Transactions on Vehicular Technology	６．最初と最後の頁 9038 - 9048
掲載論文のDOI（デジタルオブジェクト識別子） 10.1109/TVT.2019.2924911	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 該当する

１．著者名 Yi-Han Chiang, Tianyu Zhang, and Yusheng Ji	４．巻 7
２．論文標題 Joint Cotask-Aware Offloading and Scheduling in Mobile Edge Computing Systems	５．発行年 2019年
３．雑誌名 IEEE Access	６．最初と最後の頁 105008 - 105018
掲載論文のDOI（デジタルオブジェクト識別子） 10.1109/ACCESS.2019.2931336	査読の有無 有
オープンアクセス オープンアクセスとしている（また、その予定である）	国際共著 －

〔学会発表〕　計2件（うち招待講演　0件／うち国際学会　2件）

１．発表者名 Jian-Jyun Hung, Wanjiun Liao, and Yi-Han Chiang
２．発表標題 Resource Allocation for Multi-access Edge Computing with Coordinated Multi-Point Reception
３．学会等名 IEEE Wireless Communications and Networking Conference (WCNC)（国際学会）
４．発表年 2020年

１．発表者名 Tianyu Zhangy, Yi-Han Chiang, Cristian Borcea, and Yusheng Ji
２．発表標題 Learning-based Offloading of Tasks with Diverse Delay Sensitivities for Mobile Edge Computing
３．学会等名 IEEE Global Communications Conference (GLOBECOM)（国際学会）
４．発表年 2019年

〔図書〕　計0件

〔産業財産権〕

〔その他〕

―

６．研究組織

	氏名 （ローマ字氏名） （研究者番号）	所属研究機関・部局・職 （機関番号）	備考
研究協力者	張　天宇 (Zhang Tianyu)		
研究協力者	朱　超 (Zhu Chao)		