Modelling the offload of AI Tasks in Mobile Clouds

Zhiyan Chen
Department of Computer Science
University of Warwick
United Kingdom
z.chen.8@warwick.ac.uk

Ligang He
Department of Computer Science
University of Warwick
United Kingdom
Ligang.he@warwick.ac.uk

Abstract**—**In this paper, we systematically model the performance of running AI inference tasks on Mobile Cloud (MC) systems. Mobile devices collect the monitoring data and perform the model inference tasks. When the mobile devices cannot respond timely, the mobile device offloads a portion of inference tasks to the cloud server. We aim to model the performance of tasks running in such a MC system and also the resource capacities that the cloud server must have to achieve the required task performance.

Keywords—AI tasks; mobile cloud computing; task performance;

I. INTRODUCTION

AI applications have become increasingly popular and been widely deployed in various scenarios. In an AI application, an AI model is typically first trained, and is then deployed in systems to perform model inference tasks based on the input data.

In this paper, we systematically model the performance of running inference AI tasks on Mobile Cloud (MC) systems. In such a MC architecture, a collection of mobile devices are connected to a cloud server [2][3]. The model inference service is deployed in both mobile devices and the cloud server. Mobile devices collect the monitoring data (such as the occurring events captured by the sensor-enabled surveillance cameras in a factory or a farm) and perform the model inference tasks based on the input data and react with the corresponding actions based on the inference outcome. However, when the arrival rate of the incoming data becomes too big, the mobile devices may not be able to respond timely. When this happens, the mobile device offloads a portion of inference tasks to the cloud server by the way of uploading the incoming data (such as the photos taken by the surveillance camera) to the cloud. The model inference service is invoked in cloud by taking the uploaded data as input.

Given the arrival rate of the tasks (e.g., the arrival rate of the incoming data), we aim to model the performance of tasks running in a MC system and also the resource capacities such as processing speed that the cloud server has to have, so that the tasks can achieve the required performance (i.e., required average response time of the tasks).

II. RELATED WORKS

Many studies have been conducted on mobile cloud computing [1][2][3][6][9][12][13]. Some focus on the infrastructure of the system. MAUI [3] implemented the cloud computing system with VM migration and code partitioning for saving energy. In [4], device clones are used in the CloneCloud for keeping mobile applications unmodified to reduce the cost. Moreover, MobiCloud [5] transforms the traditional Mobile Ad Hoc Networks to a service oriented architecture by deploying a service on each mobile node that has sufficient computing capacity.

In addition, different methods have been developed to optimize the time and energy cost in mobile clouds [10][11]. In [7] and [8], the NP-hard property is proven for the centralized optimization problem in the multi-user cloud system. Game-theory methods is used to find the Nash Equilibrium in a distributed manner. In [14], a heuristic offloading decision algorithm is presented to achieve jointly optimization in terms of offloading decision, communication and computation resources.

III. MODELLING THE OFFLOAD-ENABLED MOBILE CLOUD SYSTEM

The architecture of a mobile cloud is illustrated in Figure 1. Assume that the arrival rate of the tasks at mobile device m_i is λ_i and u_i is the processing rate of m_i (i.e., the number of tasks that can be completed by m_i). If λ_i is equal or greater than u_i, the average response time of the tasks arriving at m_i, denoted by T_i, will be infinitely big. If λ_i is less than u_i, T_i can be calculated by Equation (1) according to the queuing theory.

$$T_i = \frac{1}{u_i - \lambda_i} \tag{1}$$

u_i in Equation (1) can be calculated by Equation (2), where w_i^D is the average computation workload (e.g., the number of instructions) of the tasks arriving at m_i while f_i is the performance of m_i, i.e., the workload that m_i can compute in a time unit.

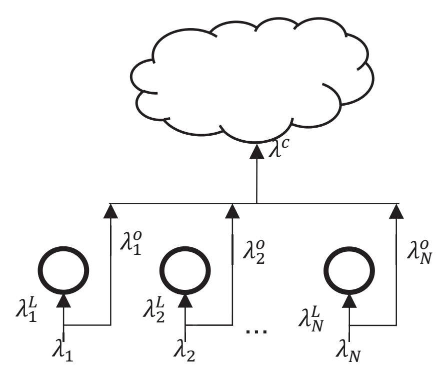

Fig. 1. The architecture of the offload-enabled mobile cloud system

$$u_i = \frac{f_i}{w_i^d} \quad (2)$$

When T_i is greater than the required response time of the tasks, denoted by T_i^L, the tasks arriving at m_i needs to be offloaded. Let λ_i^o and λ_i^L denote the rate of the tasks offloaded to the cloud and the rate of the tasks remaining in m_i, respectively. We have $\lambda_i^o = \lambda_i - \lambda_i^L$. In order to satisfy the required average response time T_i^L, λ_i^L can be calculated by Equation (3) by transforming Equation (1) and combining Equation (2).

$$\lambda_i^L = \frac{f_i}{w_i^d} - \frac{1}{T_i^L} \quad (3)$$

We also have $\lambda_i^o = \lambda_i - \lambda_i^L$. Based on the above discussions, λ_i^o can be calculated by Equation (4).

$$\lambda_i^o = \begin{cases} \lambda_i - \frac{f_i}{w_i^d} + \frac{1}{T_i^L} & \frac{1}{\frac{f_i}{w_i^d} - \lambda_i} > T_i^L \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

We can calculate λ_i^o for every mobile device. Then the total arrival rate of the tasks at the cloud, denoted by λ^c, is: $\lambda^c = \sum_{i=1}^{N} \lambda_i^o$.

The average response time of a task offloaded to the cloud (denoted by T^c) can be calculated by Equation (5), where u^c is the processing rate of the cloud.

$$T^c = \frac{1}{u^c - \lambda^c} \quad (5)$$

The average computation workload of the tasks offloaded to the cloud (denoted by $\overline{w^{dc}}$) can be calculated by Equation (6).

$$\overline{w^{dc}} = \sum_{i=1}^{N} \frac{\lambda_i^o}{\lambda^c} \times w_i^d \quad (6)$$

Then, u^c in Equation (5) can be calculated by Equation (7), where f^c is the performance of the cloud, i.e., the workload that the cloud can process in a time unit.

$$u^c = \frac{f^c}{\overline{w^{dc}}} \quad (7)$$

When a task is offloaded from mobile device m_i to the cloud, the input data of the task (e.g., the data that is needed for an AI model to infer the outcome) need to be transmitted to the cloud, which incurs the communication time from m_i to the cloud (denoted by T_i^{mc}). Let r_i^{mc} denote the bandwidth between m_i and the cloud and W_i^b denote the average size of the message that has to be communicated from m_i to the cloud when the tasks in m_i are offloaded. We can treat the network between a mobile device and the cloud as a processing system (i.e., the network processes the messages). Then $\frac{r_i^{mc}}{w_i^b}$ is the processing rate of the system (i.e., the number of messages that can be processed by the network in a time unit). Therefore, T_i^{mc} can be calculated by Equation (8) also based on the queuing theory.

$$T_i^{mc} = \frac{1}{\frac{r_i^{mc}}{w_i^b} - \lambda_i^o} \quad (8)$$

The average response time of an offloaded task in mobile device m_i (denoted by T_i^o), which is the time duration between the time point when m_i starts offloading for the task to the time when the task is completed in the cloud, is given by Equation (9). We neglect the time for the output data to be send back to m_i due to the output data size is in general much smaller than input data.

$$T_i^o = T_i^{mc} + T^c \quad (9)$$

In order to meet the required response time of T_i^L, the following inequality should hold.

$$T_i^o \le T_i^L \quad (10)$$

Combining Equations (5)-(10), we can calculate the minimal performance of the cloud server (denoted by f_i^c) to meet the required response time T_i^L for the tasks arriving at m_i.

$$f_i^c = \left(\lambda^c + \frac{1}{T_i^L - \frac{1}{\frac{r_i^{mc}}{w_i^b} - \lambda_i^o}} \right) \times \overline{w^{dc}} \quad (11)$$

Finally, the minimal performance the cloud server to meet the required response time for the tasks in all mobile devices should be:

$$f^c = \max_{1 \leqslant i \leqslant N} \{f_i^c\} \quad (12)$$

IV. EXPERIMENTS

In this section, we presents the experimental results based on the models presented in Section 3. The default values of the parameters in the experiments are listed in Table 1 unless otherwise stated.

Tab. 1. The default values of the parameters in the experiments on mobile cloud system

w_i^d	[125*0.8, 125*1.2]	Workload of the task in mobile device i
w_i^b	[16*0.2, 16*1.2]	Communication data of the task in mobile device i
f_i	[200*0.8, 200*1.2]	The processing speed of mobile device i
r_{ij}^{me}	[50*0.8, 50*1.2]	The network bandwidth between mobile device j and edge device i
λ_i	[2.0*0.8, 2.0*1.2]	The arrival rate of the tasks at mobile device i.
T_i^L	[1.2*0.9, 1.2*1.1]	The required response time of the tasks arriving at mobile i.
r_i^{mc}	[50*0.8, 50*1.2]	The network bandwidth between mobiles and the cloud

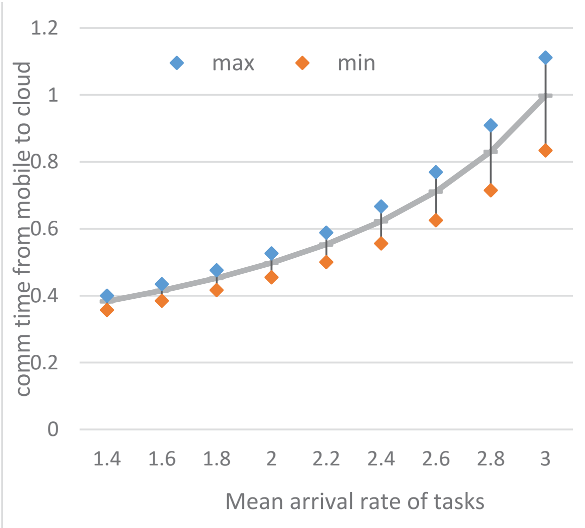

Fig. 2 The change in T_i^{mc} (communication time from a mobile device to the cloud server) as λ_i increases. The legends - max, min and mean – represent the maximum, minimum and average of T_i^{mc} (1⩽j⩽N).

Fig. 2 shows T_i^{mc} increases as λ_i increases. This is because more tasks have to be offloaded to meet the performance requirement, which leads to the increase in T_i^{mc}. Note that when λ_i is higher than 3, the performance requirement (T_i^L=1.2) cannot be met for the tasks in at least one mobile device since the maximum T_i^{mc} (shown by the blue diamond legend in the figure) will be over 1.2.

Fig. 3 shows T^c decreases as λ_i increases. This can be explained as follows. When a task is offloaded to the cloud server. The turnaround time for the offloaded task equals to $T_i^{mc} + T^c$. Since we need to maintain the task's required performance (i.e., T_i^L=1.2), T^c must be reduced by increasing the computation capacity of the cloud server to compensate for the increase in T_i^{mc}.

Tab. 2 shows the experimental results over λ_i. It can be seen from the table that as λ_i increases, λ^c, u^c and f^c increase. This result is to be expected since more tasks have to be offloaded from the mobile devices to the cloud server as $\lambda_i$increases.

Tab. 2 The experimental results over λ_i

λ_i	λ^c	u^c	f^c
1.400	6.332	7.582	946.219
1.600	8.332	9.639	1202.915
1.800	10.332	11.713	1461.852
2.000	12.332	13.816	1724.294
2.200	14.332	15.967	1992.666
2.400	16.332	18.208	2272.310
2.600	18.332	20.655	2577.738
2.800	20.332	23.775	2967.084
3.000	22.332	33.670	4202.001

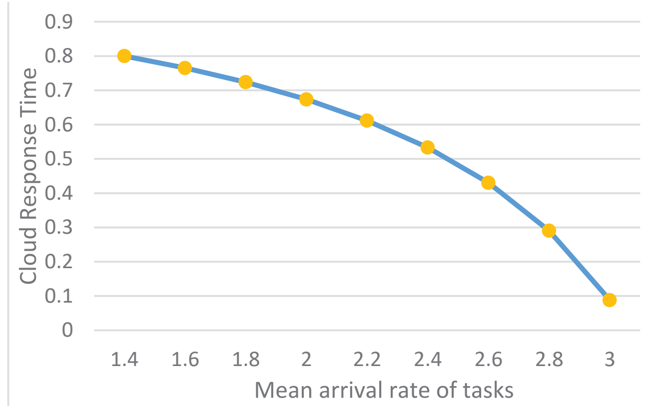

Fig. 3 The change in T^c as λ_i increases.

Table 3 shows the experimental results over the number of mobile devices (N) in the MC. As can be seen from the table, λ_i^o and T_i^{mc} remain unchanged as N increases. This is to be expected because the values of other parameters, including the required performance of tasks (T_i^L), the tasks' arrival rate (λ_i) and the processing speed of the mobile devices (f_i), are fixed in the experiments. Further, λ^c increase when N increases. This is because more mobile devices offload their tasks to the cloud. This in turn demands the more powerful cloud server in order to meet the required tasks' performance, and hence the increase in f^c and u^c. It can be seen that the increased capacity of the cloud server enables T^c to stay constant as N increases. This result indicates that our models can effectively capture the increasing demand for the cloud server as the number of the mobile devices in the MC increases.

Table 4 shows the experimental results as the mean computation workload (i.e., meanwd in the table, which the average of w_i^d) increases. It can be seen from this table that as the mean computation workload increases up to 225, λ^c, u^c and f^c all increase, which are to be expected while T^c decreases. The reason why T^c increases is because as w_i^d increases, the tasks' arrival rate (i.e., λ_i^L) that mobile devices can cope with will decrease. This results in the increase in the arrival rate of the offloaded tasks. Consequently, the arrival rate of the communication tasks (i.e., the messages that the network between a mobile device and the cloud has to transmit in a time unit) between the mobile devices and the cloud will increase.

Tab. 3 The experimental results over the number of mobile devices

N	λ_i^o	T_i^{mc}	f^c	u^c	λ^c	T^c
10	1.233	0.529	1727.854	13.823	12.333	0.671
20	1.233	0.529	3269.521	26.156	24.667	0.671
30	1.233	0.529	4811.187	38.49	37	0.671
40	1.233	0.529	6352.855	50.823	49.333	0.671
50	1.233	0.529	7894.521	63.156	61.667	0.671

Tab. 4 The impact of the mean computation workload of tasks (N=30, meanwb=16, meanlmd=1.5)

meanwd	λ^c	u^c	f^c	T^c
125	20.68	22.551	2860.452	0.534
150	27.927	30.293	4506.138	0.423
175	34.611	38.278	6743.68	0.273
200	38.623	46.477	9260.412	0.127
225	41.708	48.906	10784.428	0.139
250	Nan	Nan	Nan	Nan

Tab. 5 The impact of the communication volume

w_i^b	λ^c	u^c	f^c	T^c	maxtmc	mintmc	meantmc
14	20.68	22.208	2816.919	0.655	0.545	0.225	0.335
16	20.68	22.618	2868.92	0.516	0.684	0.263	0.413
18	20.68	22.567	2862.519	0.53	0.67	0.28	0.461
20	20.68	25.692	3258.875	0.2	1	0.33	0.564
22	20.68	38.945	4939.985	0.055	1.145	0.365	0.633

This in turn increases the communication that the offloaded tasks have to experience. Eventually, the cloud has to compensate by reducing its response time for the offloaded tasks in order to meet the tasks' performance requirement (i.e., T_i^L). This is also the reason why f^c has to increase at a higher rate than λ^c. For example, when meanwd increases from 125 to 150, λ^c increases by around 35%, but f^c increases by 58%. Note that when meanwd increases to 250, the cloud will not be able to meet the tasks' performance requirement, no matter how much resource capacity is allocated to the cloud server. The reason is because with this value of meanwd, the communication time alone between at least one mobile device and the cloud server is greater than the tasks' performance requirement, which means that the offloaded tasks will not meet the performance requirement even if the cloud server takes zero second to complete the tasks.

Table 5 shows the experimental results as the communication volume of the tasks (i.e., w_i^b) increases. It can be observed from Table 5 that as the average of communication volume (w_i^b) increases, the communication time between the mobile devices and the cloud, including maxtmc (i.e., the max of $T_i^{mc}, 1 \leqslant i \leqslant N$), mintmc (min of T_i^{mc}) and meantmc (mean of T_i^{mc}), increase. This is to be expected. Moreover, as w_i^b increases, u^c and f^c also increase T^c while decreases. This is because when w_i^b increases, a task's communication time, which is one part of the total turnaround time of an offloaded task, increases. Consequently, the cloud has to reduce the other part of the total turnaround time (i.e., T^c - the cloud response time) in order to meet the task's performance requirement, which can only be achieved by increasing f^c (i.e., the processing speed of the cloud server). Since the computation workload of the tasks remain unchanged in the experiments, the processing rate of the cloud (i.e., u^c) also increases as the result of the increase in f^c.

V. CONCLUSIONS

In this paper, we consider a mobile cloud system where the AI inference services are deployed in mobile devices and the cloud server. The data arrive at the mobile devices and then the inference services deployed in the mobile devices need to process the AI inference tasks and meet the required response time. If a mobile device cannot meet the tasks' performance requirement, it can offload the tasks to run on the cloud server. We present an approach to modelling the task performance in such a scenario and also model the minimal resource capacity that the cloud server has to be equipped with in order to meet the performance requirement. The experimental results show that the proposed modelling approach can accurately capture the task performance and the resource demand in order to meet the performance requirement.

REFERENCES

[1] Kremer U, Hicks J, Rehg J M. 2000. Compiler-directed remote task execution for power management. Proceedings of the Workshop on Compilers and Operating Systems for Low Power.

[2] Kumar K, Lu Y H. 2010. Cloud computing for mobile users: Can offloading computation save energy? Computer 43, 4 (April 2010), 51-56

[3] Cuervo E, Balasubramanian A, Cho D, et al. 2010. Maui: making smartphones last longer with code offload. Proceedings of the 8th international conference on Mobile systems, applications, and services. (June 2010): 49-62.

[4] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, A. Patti, Clonecloud: elastic execution between mobile device and cloud, in: Proceedings of the Sixth Conference on Computer Systems, EuroSys'11, ACM, New York, NY, USA, 2011, pp. 301–314.

[5] D. Huang, X. Zhang, M. Kang, J. Luo, Mobicloud: building secure cloud framework for mobile computing and communication, in: Proceedings of the Fifth IEEE International Symposium on Service Oriented System Engineering, SOSE, pp. 27–34.

[6] Barbera M V, Kosta S, Mei A, et al. 2013. To offload or not to offload? the bandwidth and energy costs of mobile cloud computing. Proceedings Ieee Infocom. IEEE, 1285-1293.

[7] Chen, Xu. "Decentralized computation offloading game for mobile cloud computing." IEEE Transactions on Parallel and Distributed Systems 26.4 (2014): 974-983.

[8] Chen X, Jiao L, Li W, et al. 2016. Efficient multi-user computation offloading for mobile-edge cloud computing. IEEE/ACM Transactions on Networking. 24, 5 (October 2016), 2795-2808.

[9] Gao B, He L, Jarvis S A. 2015. Offload decision models and the price of anarchy in mobile cloud application ecosystems. IEEE Access. 3, 3125-3137.

[10] Zhang W, Wen Y, Wu D O. 2014. Collaborative task execution in mobile cloud computing under a stochastic wireless channel. IEEE Transactions on Wireless Communications. 14, 1(Jan. 2015), 81-93.

[11] Wang, Xinghan, et al. "PSOGT: PSO and game theoretic based task allocation in mobile edge computing." 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS). IEEE, 2019.

[12] Chatzopoulos, Dimitris, et al. "Flopcoin: A cryptocurrency for computation offloading." IEEE transactions on Mobile Computing 17.5 (2017): 1062-1075.

[13] Soyata, Tolga, et al. "Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture." 2012 IEEE symposium on computers and communications (ISCC). IEEE, 2012.

[14] Lyu X, Tian H, Sengul C, et al. 2016. Multiuser joint task offloading and resource optimization in proximate clouds. IEEE Transactions on Vehicular Technology. 66, 4(April 2017), 3435-3447.