Virtual Machine Placement and Workload Assignment for Mobile Edge Computing

Wei Wang[1, 2], Yongli Zhao[1, 2], Massimo Tornatore[2, 3], Abhishek Gupta[2], Jie Zhang[1], Biswanath Mukherjee[2]
[1] Beijing University of Posts and Telecommunications, China;
[2] University of California, Davis, USA; [3] Politecnico di Milano, Italy.
{weiw, yonglizhao, lgr24}@bupt.edu.cn; massimo.tornatore@polimi.it; {abgupta, bmukherjee}@ucdavis.edu

***Abstract*—As an extension to cloud computing, Mobile Edge Computing (MEC) provides computing capability at close proximity to mobile users to reduce service latency and improve users' quality of experience. In MEC, tiny datacenters, equipped with computing and storage capabilities, are located at the edge of the mobile network. Services in these tiny datacenters are supported by application-specific software instances, which are packaged in Virtual Machines (VM). We study the problem of VM placement and workload assignment for mobile cloud applications in MEC. We formulate a mathematical model to minimize the hardware consumption required by VMs for supporting given workloads in a multi-application scenario, while meeting heterogeneous latency requirements of different applications. Numerical results show that applications' latency requirement, MEC servers' hardware capacity, and users' request load have significant influences on overall hardware consumption, and MEC server utilization can be optimized by leveraging remote VM placement and workload aggregation.**

Keywords—Mobile Edge Computing; VM placement; Workload assignment; Latency; Hardware consumption.

I. INTRODUCTION

Cloud computing is a popular form for Internet-service provisioning, and Datacenters (DCs) are major infrastructures to provide required computing and storage capabilities. DCs used in cloud computing are usually centralized and deployed at a few locations, which can be far away from most users. Therefore, users may experience long service latency, because their traffic needs to travel a long distance to get served. Recently, new applications (APPs), which require ultra-low service latency (e.g., augmented reality, self-driving cars, etc.), are emerging, and cloud computing may not be able to serve these APPs due to the excessive latency incurred to reach distant DCs. Even for APPs with looser latency constraints, yet having intensive bandwidth requirement, cloud computing might not be the most effective solution. Thus, Mobile Edge Computing (MEC) has been introduced to reduce service latency by deploying computing resource at close proximity to mobile users [1].

Computing and storage resources for MEC are usually deployed within the footprint of access-aggregation networks in form of tiny DCs, namely, MEC servers [2]. In the context of 5G mobile networks, it is a popular idea to deploy MEC servers along with Base Stations (BSs) within Radio Access Networks (RAN) [3]. In this case, traffic from user devices to each BS can be serviced by a local MEC server, and thus avoid long-distance network transmission. But, even in a MEC scenario, it does not mean that all traffic would be processed locally. On the one hand, MEC servers' capacity is usually limited, and they might not be able to host all traffic from various APPs. On the other hand, APPs' latency requirement are heterogeneous, and some traffic can still be served in regional MEC servers or cloud DCs as long as their latency requirements are met [4]. Hence, distributed MEC servers usually do not work independently; instead, they work collaboratively with one another and with centralized cloud DCs [5]. An issue for service providers operating such collaborative systems (e.g., a content provider) is to determine exact service destination DCs for each APP request from each edge. Ref. [6] studied the tradeoff between low latency of edge computing and high power efficiency of cloud computing, and formulated a power-minimization model under constrained service delay. It is defined at hardware level, where computing power is taken as the general service resource for all APPs. However, in principle, single hardware resources cannot serve heterogeneous APP requests unless specific software instances are employed to provide service logics and data.

Virtual Machine (VM) is an attractive concept for DC resource management. A VM is a logical container, which can host software instances with data, and can run using only a subset of a server's hardware resources. VMs have been introduced in MEC to provide application-specific capabilities in MEC servers [7]; thus specific APP requests can only be hosted at the MEC servers, which have corresponding VMs.

In the context of VM-based MEC, several VM-related problems have been studied. Due to user mobility, MEC systems require service migrations as user locations change over time. To decide on service migrations optimally, a sequential decision making problem has been formulated for service migration using a Markov decision process [8]. To reduce the performance degradation during service migration, a VM handoff approach has been proposed in [9] to adaptively reduce data transferred when users move. To maximize the revenue of infrastructure owner in a multi-operator network-sharing environment with time-critical Service Layer Agreements (SLAs), a VM scheduling model is formulated in [10] to dynamically provide VMs according to predicted workload. However, a more generic problem - VM placement and workload assignment - which aims to allocate DCs' hardware resources to VMs for each APP at

978-1-5090-4026-1/17/$31.00 ©2017 IEEE

each MEC server, and to assign given workloads to the placed VMs, has not been studied yet.

VM placement and workload assignment has a strong impact on service latency. The location of VMs decides the distance from source node to destination VM, and number of VMs decides the service capabilities at host MEC servers. Also, service capabilities of each VM is finite, and workload assigned to each VM can affect VM's average response time significantly. VM management problems and studies exist in cloud computing system [11-15], but these approaches cannot be applied to MEC directly, as new features of MEC (e.g., tiny and highly distributed MEC servers, latency-sensitive APPs, etc.) have not been addressed in the context of cloud computing.

In this work, we focus on the VM placement and workload assignment problem. We discuss the impact of VM placement and workload assignment on MEC service latency. A Mixed Integer Linear Program (MILP) model is formulated to minimize hardware consumption for deploying VMs, while meeting heterogeneous latency requirements of different APPs. Numerical results show that APPs' latency requirement, MEC servers' hardware capacity, and users' request load have significant influences on overall hardware consumption, and MEC server utilization can be optimized by leveraging remote VM placement and workload aggregation.

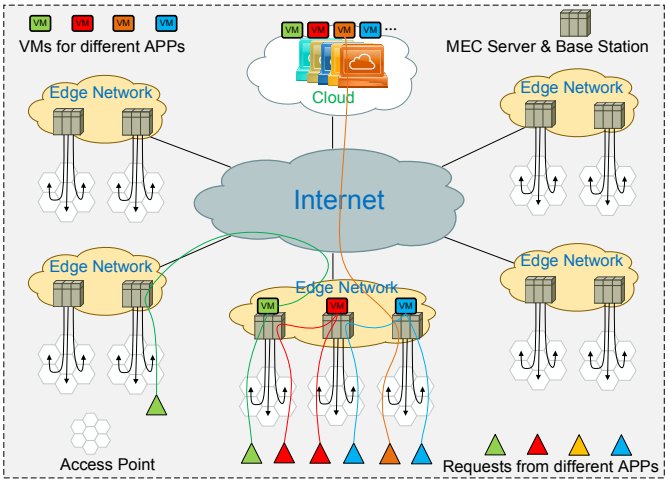

Fig. 1. Overview of a VM-based MEC system.

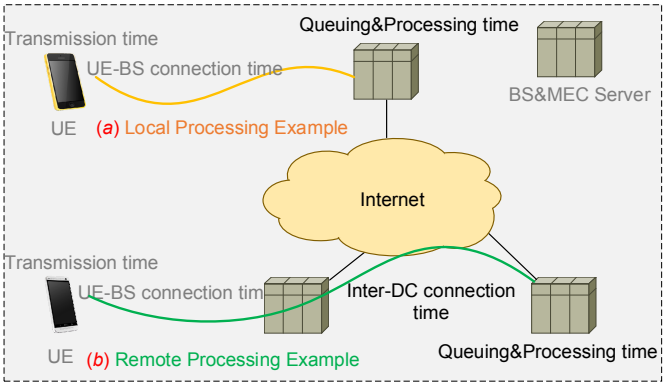

Fig. 2. Latency components for local and remote DC procesing.

II. VM Placement and Workload Assignment

In this section, we discuss the latency components for MEC service, and clarify the parts relevant to our work. We discuss the VM-related parameters in MEC system and analyze their impacts on service latency. Finally, we model the problem of VM placement and workload assignment in MEC system.

A. Overview of MEC System

MEC servers can be deployed at any network node from access networks to core networks. The nearer a MEC server is to the core network, the more users can reach it, but the longer is distance from end users. A popular option is to deploy MEC servers along with mobile BSs, so there would be lots of MEC servers distributed geographically with BSs. On each MEC server, there might be several kinds of VMs for different APPs. Fig. 1 shows an overview of a VM-based MEC system, where there are several distributed MEC servers, and one cloud DC. (For convenience of presentation, we will use DC to represent both MEC server and cloud DC in the rest part of this work.) Even though DCs are highly distributed, VMs on them are not independent, and they can work collaboratively via networking when necessary.

B. Latency Analysis

Before delving into details of VM placement and workload assignment, let us clarify the term "service latency" relevant to this work. In this work, service latency refers to the round-trip time from sending a request to getting a response back. It has four parts: 1) transmission time, 2) propagation time, 3) queueing time, and 4) processing time.

In the context of MEC, users' requests might be processed by local or remote DCs. Fig. 2 shows the latency components arising in local and remote processing scenarios. Originating from User Equipment (UE), the first hop of each APP request is the connection between UE and local BS, and local BS becomes the common access node for requests from its covered UEs. In this hop, each request will experience a transmission delay at UE and a network delay on UE-BS connection. Fig. 2(a) shows the local processing scenario, where requests from BS are handled by local DC directly, and queueing and processing occur at local DC. If local DC is unable to serve a request, the request will be re-directed to another remote DC for collaborative processing (see Fig. 2(b)). Now, the inter-DC connection will cause network delay, and queueing and processing delay occurs at remote DCs. Since UE and UE-BS connection part cannot be affected by VM placement and workload assignment, transmission time and network propagation time of UE-BS connection are out of our control. Thus, we focus on the queueing and processing time at local or remote DCs and network propagation latency on inter-DC connections.

Let set V denote all DCs in a collaborative MEC and cloud computing system. Two parameters are important for service latency: 1) H_v, hardware capacity of DC v, where $v \in V$, and 2) $t_{v1,v2}$, network propagation latency from DC $v1$ to DC $v2$, where $v1, v2 \in V$. Since the number of users covered by each BS is not very large, the capacity of each DC does not need to be very large to manage infrastructure cost. Hardware in a DC is composed of CPU, memory, and storage. Our work considers a

generalized parameter H_v as the hardware capacity of DC v. Hardware capacity of a DC decides the number of VMs it can host, which in turn affects queueing and processing delays. The network connection between each DC pair may go through local access network, regional aggregation network, or even backbone transport networks, depending on specific locations of DCs, so inter-DC latency can assume very diverse values.

VMs are containers used to serve APP requests. Since different APPs generate different requests, VMs are not general, but application-specific. Hence, in a multi-application scenario (as studied here), there will be multiple kinds of VMs. VMs are virtualized from DC infrastructures, and need certain amount of hardware resource to support self-running and service provisioning. Excluding the case where VMs' occupied resource can be expanded/shrinked, the amounts of hardware resource required by each kind of VM are assumed to be fixed.

Let A denote a set of APPs in a MEC system, and each APP a, where $a \in A$, corresponds to a unique kind of VM. For each APP a, two parameters of its corresponding VM have key impacts on service latency: 1) C_a, hardware capacity required for deploying one VM for APP a, and 2) u_a, VM's service capability (service rate) for requests of APP a. C_a and H_v set the number of VMs for APP a that DC v can host. u_a sets the queueing and processing latency under certain workloads.

C. VM Placement and Workload Assignment

To study VM placement and workload assignment, let us first clarify the term "request flow" and "workload". Excluding the scenario where one mobile user has access to multiple BSs, we can assume that each APP request is associated only to one BS. Thus, the DCs, which are co-located with BSs, can be taken as the source nodes of APP requests, and we can remove the UE-BS connection part from our modeling. Note that the original DC can also be the destination node of a request if it is processed locally. At each DC, requests of each APP arrive as per users' actions. We call the requests of same APP originating from same DC as a *request flow*. To estimate the queuing and processing latency of each request flow, we assume the individual requests in one request flow arrive as a Poisson process, and r_v^a denotes the arrival rate (workload) of requests for APP a at DC v.

With the above parameters, the basic problem for MEC is how to provision all the request flows, while meeting their heterogeneous latency requirements. Two parameters need to be decided to provision each request flow: 1) exact destination DC for each flow, and 2) exact service capability allocated from destination nodes for each flow. Since service capabilities are allocated from the VMs at destination nodes, we should first decide the number of VMs for each APPs at each DC (VM placement) before selecting destination and allocating service capabilities (service rate) from the placed VMs for each request flow (workload assignment).

VM placement has to decide the number n_v^a of VMs for APP a at DC v. n_v^a could be 0 if there is no VMs for APP a at DC v. For each DC, all the VMs it hosts share its hardware capacity, and the total hardware capacity occupied by all placed VMs cannot exceed its hardware capacity. Since the service capability of each VM is known, n_v^a decides the potential service capability for all requests of APP a to DC v.

Workload assignment has to decide the destination and allocate service capability for each request flow. Since each request flow is aggregated from a certain amount of users, some flows might be too large for local existing VMs to process. In this case, one request flow will be split into several sub-flows for collaborative processing. The sub-flows from one original request flow are assigned independently to either local or remote VMs, and thus their destinations could be same or not. Note that sub-flows can only be assigned to DCs, which have sufficient service capability to satisfy latency bounds. Based on sub-flow, workload assignment has to decide the amount of sub-flows for each original request flow, the destinations of each sub-flow, and the service capabilities for each sub-flow.

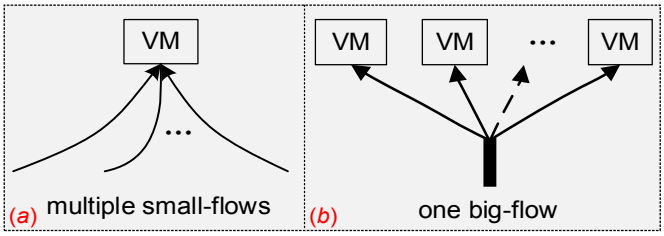

Fig. 3. Sub-flow assignment scenarios.

One sub-flow can be assigned to VM(s) according to two scenarios: 1) when a sub-flow is small enough and does not need one entire VM, it might be assigned to a VM, which is shared with some other sub-flows from different, source DCs as illustrated in Fig. 3(a). In this case, we name it as *small flow*, and $sr_{s,d}^a$ denotes the small flow of APP a from source DC s to destination DC d, where $s, d \in V$, and the service capability assigned to it from d is expressed as $u_{s,d}^a$. The handling process of this case can be modeled as a M/M/1 queueing system, and the queueing and processing latency is calculated as in Eqn. (1); 2) if a sub-flow is large and exceeds the service capability of one entire VM, it will be assigned to one or more entire VMs (see Fig. 3(b)). In this case, we call it as *big flow*, and $br_{s,d}^a$ denotes the big flow of APP a from DC s to DC d, and the number of VMs assigned to it from d is expressed as $m_{s,d}^a$. This case can also been modeled as a M/M/1 queue with the assumption that the requests in this sub-flow are dispatched to all allocated VMs evenly. The queueing and processing latency of this case is calculated as in Eqn. (2).

$$t_{s,d}^{pq} = \frac{1}{u_{s,d}^a - sr_{s,d}^a} \tag{1}$$

$$t_{s,d}^{pq} = \frac{1}{u_a - br_{s,d}^a / m_{s,d}^a} \tag{2}$$

$$t_{s,d}^a = t_{s,d}^{pq} + t_{s,d} \tag{3}$$

Results of VM placement for each APP at each DC are given by the value of variable n_v^a. Results of workload assignment are given by the values of variables $sr_{s,d}^a$, $u_{s,d}^a$, $br_{s,d}^a$, and $m_{s,d}^a$. For each sub-flow, queueing and processing latency is calculated as per Eqn. (1) or (2), while $t_{s,d}$ is a known parameter representing the network propagation time between source s and destination d. In summary, service latency for each sub-flow is expressed as Eqn. (3). Note that $t_{s,d} = 0$ when $s = d$.

III. Mixed Integer Linear Program model for hardware consumption minimization

In a multi-application scenario, many solutions exist to place VMs and assign workloads to provide latency-satisfied service. The basic solution is to deploy VMs and process all request flows at local DCs. But this solution may not be optimal in terms of hardware consumption. First, a small sub-flow, which does not require one entire VM, cannot go to a remote spare VM to avoid deploying a new VM. Second, this solution is infeasible when congestion occurs at some DCs. To provide latency-satisfied service to certain known request flows optimally, this section formulates a MILP model to place VMs and assign request flows for each APP at each DC, and the objective is to minimize hardware consumption for deploying VMs, which is a common objective for both service providers and infrastructure operators. Hardware consumption means the cost of leasing space to place VMs for service providers, and it also means resource efficiency for infrastructure operators.

A. Problem Statement:

Given a set of DC, hardware capacity of each DC, network propagation latency between each DC pair, a set of APP, hardware capacity required for deploying a VM for each APP, service rate of a VM for each APP, average latency threshold for each APP, and workload of each request flow, we determine the number of VMs for each APP at each DC, request load and destination of each sub-flow, and service rate for each sub-flow.

B. Input Parameters:

V: Set of DC
A: Set of APP
H_v: Hardware capacity of DC v, $v \in V$
$t_{s,d}$: Network propagation latency from DC s to DC d, $s, d \in V$
C_a: Hardware required by a VM for APP a, $a \in A$
T_a: Average latency threshold for APP a
u_a: Service rate of one VM for APP a
r_v^a: Arrive rate of request for APP a from v

C. Variables:

n_v^a: integer, number of VMs for APP a placed at DC v
$br_{s,d}^a$: float, load of big sub-flow of APP a, from DC s to d
$m_{s,d}^a$: integer, number of VMs allocated for $br_{s,d}^a$
$sr_{s,d}^a$: float, load of small sub-flow of APP a, from DC s to d
$sf_{s,d}^a$: binary, whether $sr_{s,d}^a$ is zero or not
$u_{s,d}^a$: float, allocated service rate to $\mathrm{sr}_{s,d}^a$

D. Objective: Minimize Hardware Consumption:

$$\text{Min} \sum_{v \in V} \sum_{a \in A} n_v^a * C_a \tag{4}$$

E. Constraints:

$$\sum_{a \in A} n_v^a * C_a \le H_v \qquad v \in V \tag{5}$$

$$\sum_{d \in V} br_{s,d}^a + sr_{s,d}^a = r_s^a \qquad s \in V \;\; a \in A \tag{6}$$

$$\sum_{s \in V} u_{s,d}^a + u_a * m_{s,d}^a = u_a * n_d^a \qquad d \in V \;\; a \in A \tag{7}$$

$$br_{s,d}^a \le (u_a - \frac{1}{T_a - t_{s,d}}) * m_{s,d}^a \qquad s \in V \; d \in V \; a \in A \tag{8}$$

$$sr_{s,d}^a \le u_{s,d}^a - \frac{sf_{s,d}^a}{T_a - t_{s,d}} \qquad s \in V \; d \in V \; a \in A \tag{9}$$

$$u_a * m_{s,d}^a - br_{s,d}^a \ge 0 \qquad s \in V \; d \in V \; a \in A \tag{10}$$

$$u_{s,d}^a - sr_{s,d}^a \ge 0 \qquad s \in V \; d \in V \; a \in A \tag{11}$$

$$(T_a - t_{s,d}) * m_{s,d}^a \ge 0 \qquad s \in V \; d \in V \; a \in A \tag{12}$$

$$(T_a - t_{s,d}) * sf_{s,d}^a \ge 0 \qquad s \in V \; d \in V \; a \in A \tag{13}$$

$$sf_{s,d}^a = 0 \Leftrightarrow u_{s,d}^a = 0 \qquad s \in V \; d \in V \; a \in A \tag{14}$$

Constraint (5) guarantees that all the VMs placed at each DC cannot exceed the host's hardware capacity. Equation (6) enforces that each request flow is totally assigned in terms of small sub-flows and big sub-flows. Eqn. (7) enforces that the allocated service capabilities for each APP at each DC is from the placed application-specific VMs at target DC. Constraints (8) and (9) are transformed from Eqn. (3), and they guarantee that the service latency of both big sub-flows and small sub-flows are within their latency threshold. Constraints (10) and (11) guarantee that the queueing system of both big sub-flow and small sub-flow are stable. Constraints (12) and (13) guarantee that both the big sub-flows and small sub-flows are not routed to the DC to which the network propagation latency is over their latency threshold. Constraints (14) guarantees that no service capability is allocated to an empty small sub-flow, whose workload is zero. Note that this model will be infeasible when there is no DC can provide sufficient computing resource to meet the required latency threshold of any request flow.

IV. Illustrative Numerical Examples

In this section, we study the performance of VM placement and workload assignment in several conditions. This work models the relationship between DC's hardware capacity, VM features, and APPs' latency requirement. It can be applied not only to MEC system, but also to other collaborative computing systems. To get realistic data as input parameters (e.g., inter-DC network propagation latency), we choose seven cities from south-west United States (US) as edge nodes, which host tiny DCs, and one city from northern-west US as the cloud DC node. We define a parameter – Scale Factor - to expand/shrink the basic eight-city topology, and to see the performance under different scales (see Fig. 7). The hardware capacity of each edge DC is set according to the proportion of populations on each city. The details of edge nodes and cloud node are listed in Table I. Network latencies between each node are obtained from a website [16], and they are used as inter-DC network propagation latency. We consider seven APPs as examples for MEC system, and their latency requirements [17] are listed in Table II. The VM-related parameters for each APP can also be found in Table II. At each edge city, there is a certain amount of request load for each APP, and the load proportion among all APPs at each

edge are not the same. Due to page limitation, the values of inter-DC latency and request load are not listed here. We define three parameters to make changes to the base parameters in Tables I and II, and they are Latency requirement Ratio (LR), HW capacity Ratio (HR), and Request load Ratio (RR). We vary LR and HR in Figs. 4, 5 and 6, respectively, to see their influence on hardware consumption and inter-DC traffic.

TABLE I. PRAMETERS ABOUT EDGE.

City	HW Capacity (unit)	Type
San Francisco	225	Edge
Los Angeles	1000	Edge
Phoenix	375	Edge
Sacramento	125	Edge
San Jose	250	Edge
Las Vegas	150	Edge
San Diego	350	Edge
Portland	10000	Cloud

TABLE II. PRAMETERS ABOUT APPLICATIONS.

App	Latency threshold (ms)	HW requirement per VM (unit)	Service rate per VM (#/sec)
Self-Driving	1	1	1500
Augmented Reality	2	1	800
Healthcare	5	1	500
Accelerated Video	10	1	400
Virtual Reality	20	1	300
Web Game	30	1	200
Broadcast	50	1	100

Figs. 4, 5, and 6 show hardware consumption (see curves, measured by left-Y) and inter-DC traffic (see bars, measured by right-Y) under three scenarios. We observe that, as expected, overall hardware consumption for placing VMs and inter-DC traffic increase significantly when 1) APPs' latency requirement decreases (Fig. 4), 2) DCs' hardware capacity decreases (Fig. 5), and 3) request load increases (Fig. 6).

In Fig. 4, overall hardware consumption increases because more VMs need to be placed to process given workloads faster when APPs require lower latency. In addition, the increasing hardware consumption may cause congestion at edge DCs, and some requests need to go to remote DCs. Now, inter-DC communication used for request relocation introduces network propagation latency, and makes the time left for queueing and processing even lower. Thus, congestion also increases hardware consumption. In Fig. 5, computing resources are more easily congested when edge DCs' hardware capacity decreases, and congestion increases hardware consumption as explained above. In Fig. 6, hardware consumption increases when request load increases. Theoretically, it should increases proportionally with workload. However, the increment is nonlinear because the equipped hardware capacity is not infinite, and congestion occurs when request load become high.

Among seven example APPs, Figs. 4, 5, and 6 also show that the inter-DC traffic of the most latency-insensitive APP is much higher than other APPs, especially under congestion. When local DCs cannot handle all the requests, the APPs whose latency requirements are loose can reach more remote DCs; hence they will more likely to be assigned to remote DCs to make room for latency-sensitive APPs at local DC.

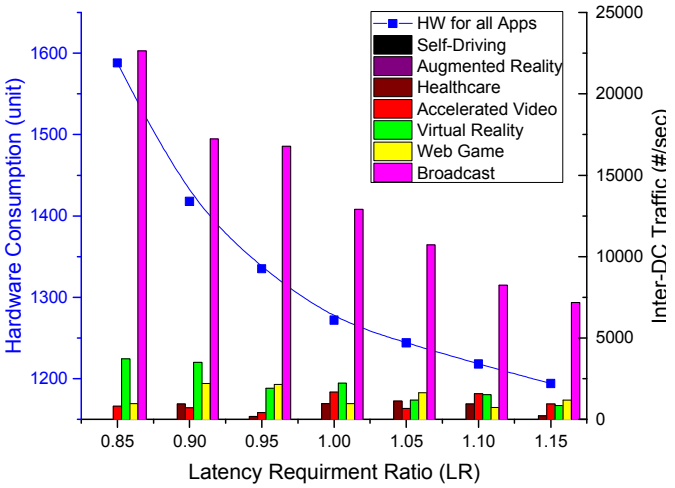

Fig. 4. Hardware consumption and inter-DC traffic for different LRs.

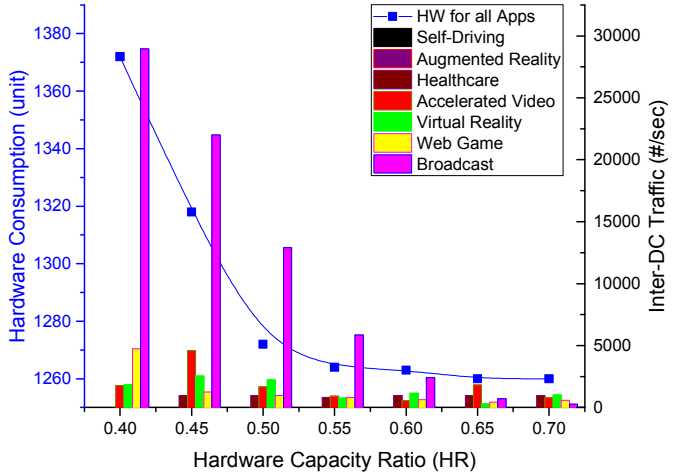

Fig. 5. Hardware consumption and inter-DC traffic for different HRs.

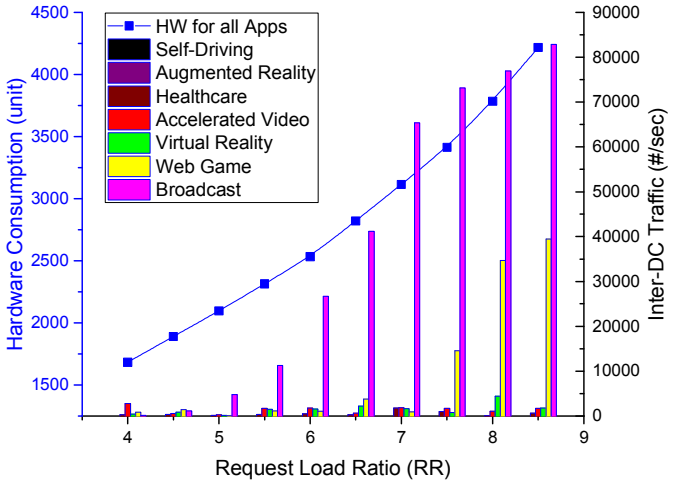

Fig. 6. Hardware consumption and inter-DC traffic for different RRs.

Fig. 7 shows hardware consumption under different topology scales. When the topology shrinks, edge DCs are nearer to each other and APPs pay less time on inter-DC communications. Thus, more time is left for queueing and

processing, and less VMs and hardware capacity can satisfy the given latency requirement. Another reason is that the collaboration between DCs is more flexible in a small-scale topology, because more APPs have chances to get served at remote DCs. We also observe that the proportion of inter-DC traffic among all APPs become unstable when the test topology shrinks significantly, as more APPs have chances to go farther.

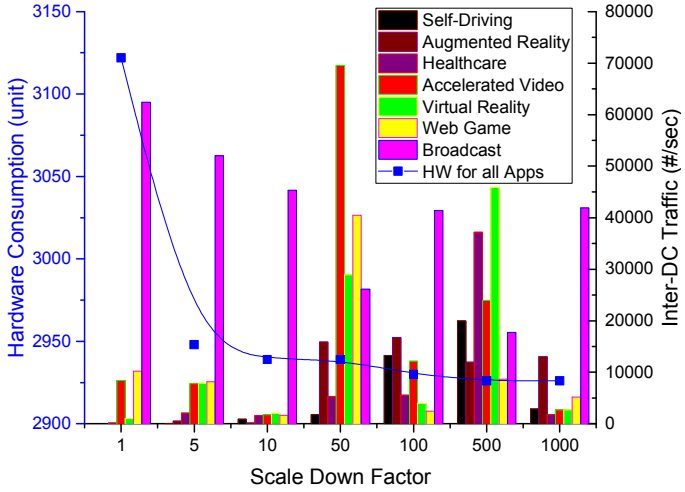

Fig. 7. Hardware consumption and inter-DC traffic for different topo scales.

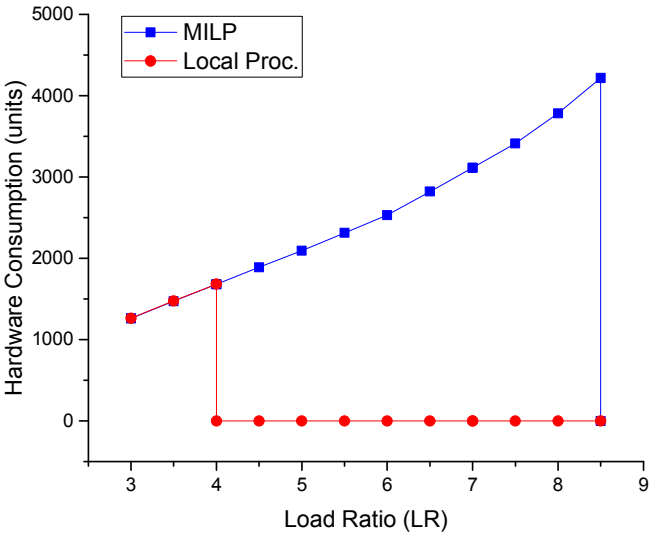

Fig. 8. Hardware consumption of different stategies.

Fig. 8 shows the benefits of proposed model in terms of hardware consumption. Note that the point where lines go down to zero means blocking happens. Compared with pure local processing, MILP model can accommodate more workloads without blocking, and it indicates that DC utilization can be optimized by leveraging remote VM placement and workload aggregation.

V. CONCLUSION

This work studied Virtual Machine (VM) placement and workload assignment for a collaborative Mobile Edge Computing (MEC) system. The relationship between MEC service latency and VM-related MEC resource were discussed. A Mixed Integer Linear Program (MILP) model was formulated to optimally place VMs and assign workloads while meeting applications' latency requirements. Numerical results showed that applications' latency requirement, MEC servers' hardware capacity, and users' request load have significant influences on hardware consumption and inter-datacenter traffic, and MEC server utilization can be optimized by leveraging remote VM placement and workload aggregation.

ACKNOWLEDGEMENT

This work is sponsored in parts by NSFC (61571058, 61601052), and BUPT Excellent Ph.D. Students Foundation (CX2016310). We also thank China Scholarship Council for supporting Wei Wang to study at UC Davis.

REFERENCES

[1] Y. Hu, et al., "Mobile edge computing—A key technology towards 5G," ETSI White Paper (2015).

[2] G. Brown. "Mobile Edge Computing Use Cases & Deployment Options," Juniper White Paper (2016).

[3] A. Ahmed and E. Ahmed, "A survey on mobile edge computing," 10th International Conference on Intelligent Systems and Control (ISCO), Coimbatore, 2016.

[4] L. Gupta, et al., "Mobile Edge Computing – An Important Ingredient of 5G Networks," IEEE Software Defined Networks Newsletter, 2016.

[5] T. X. Tran, et al., "Collaborative Mobile Edge Computing in 5G Networks: New Paradigms, Scenarios, and Challenges," IEEE Communications Magazine, vol. 55, no. 4, pp. 54-61, April 2017.

[6] R. Deng, et al., "Optimal Workload Allocation in Fog-Cloud Computing Toward Balanced Delay and Power Consumption," IEEE Internet of Things Journal, vol. 3, no. 6, pp. 1171-1181, Dec. 2016.

[7] L. F. Bittencourt, et al., "Towards Virtual Machine Migration in Fog Computing," 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), Krakow, 2015.

[8] S. Wang, et al., "Dynamic service migration in mobile edge-clouds," IFIP Networking Conference, Toulouse, 2015.

[9] K. Ha, et al., "Adaptive vm handoff across cloudlets," Technical Report CMU-CS-15-113, CMU School of Computer Science, 2015.

[10] K. Katsalis, et al., "SLA-Driven VM Scheduling in Mobile Edge Computing," IEEE 9th International Conference on Cloud Computing (CLOUD), San Francisco, 2016.

[11] F. Larumbe and B. Sansò, "Green Cloud Broker: On-line Dynamic Virtual Machine Placement Across Multiple Cloud Providers," IEEE 5th International Conference on Cloud Networking (Cloudnet), Pisa, 2016.

[12] K. Chen, et. al., "Intelligent virtual machine placement for cost efficiency in geo-distributed cloud systems," IEEE International Conference on Communications (ICC), Budapest, 2013.

[13] Z. Xiao, et al., "Dynamic Resource Allocation Using Virtual Machines for Cloud Computing Environment," IEEE Transactions on Parallel and Distributed Systems, vol. 24, no. 6, pp. 1107-1117, June 2013.

[14] S. Rahman, et al., "Dynamic Workload Migration over Optical Backbone Network to Minimize Data Center Electricity Cost," IEEE International Conference on Communications (ICC), France, 2017.

[15] Y. Wu, et al, "Green Data Center Placement in Optical Cloud Networks," in IEEE Transactions on Green Communications and Networking, doi: 10.1109/TGCN.2017.2709327

[16] Wondernetwork, [online] Available: https://wondernetwork.com/pings

[17] "5G White Paper," [online] Available: ww.ngmn.org/uploads/media/NGMN_5G_White_Paper_V1_0.pdf