Optimization of virtual machine placement for balancing network and server load in edge computing environments

Shota Nangu*, Tomotaka Kimura† and Kouji Hirata ‡
* Graduate School of Science and Engineering, Kansai University, Osaka, Japan
E-mail: k998653@kansai-u.ac.jp
† Faculty of Science and Engineering, Doshisha University, Kyoto, Japan
E-mail: tomkimur@mail.doshisha.ac.jp
‡ Faculty of Engineering Science, Kansai University, Osaka, Japan
E-mail: hirata@kansai-u.ac.jp

***Abstract*—In edge computing environments, edge servers are deployed in networks in addition to cloud data centers. Designing an edge computing system involves several issues such as edge server deployment, virtual machine (VM) placement, and traffic routing. This paper focuses on the VM placement problem, which determines the location of VMs in edge servers, and proposes a new VM placement method considering network and server load. The proposed method formulates the VM placement problem as mixed integer programming (MIP) problems. By solving the MIP, the proposed method determines the optimal location of VMs so as to minimize the network load generated by the communication between users and VMs, and balance the load on edge servers. Since it takes a large amount of computation time to solve the MIP, the proposed method also introduces a heuristic algorithm for finding an approximate solution. Numerical experiments show the effectiveness of the proposed method.**

I. Introduction

Recently, based on the concept of the Internet of Things (IoT), the research and development of technologies aiming to solve various problems in the real world have been actively conducted. In IoT environments, a huge amount of information with wide versatility will be generated by a wide variety of interconnected devices [2]. For example, to date, main devices connected to the Internet are smartphones and computers. With the spread of the IoT environments, many kinds of devices such as home appliances, automobiles, and sensors will be connected to the Internet. Thus, the number of devices will increase considerably, and the traffic load on the Internet will increase accordingly.

IoT services are supported by cloud computing platforms. Cloud computing processes data transmitted from IoT devices in cloud data centers located away from the devices and provides services to users through the Internet. Therefore, various information is transmitted among IoT devices, cloud data centers, and users. If the cloud data centers are located far from the IoT devices, the network load becomes large because data traffic pass through many network links. As the number of IoT devices increases, the network load further increases, which causes network congestion. As a technology resolving the problem, edge computing now attracts much attention [3], [6].

In edge computing environments, edge servers are deployed at the edge of networks where there exist users and IoT devices while cloud data centers are located far from them. Instead of the cloud data centers, the edge servers process data transmitted from the IoT devices. By doing so, network load and communication delay are expected to be reduced. In general, data processing in the edge servers is performed by virtual machines (VMs) [5]. VMs are duplicated into multiple edge servers in order to distribute the load for the VMs. Each data sent from the IoT devices are processed by one of the VMs as necessary. It is important to consider the VM placement problem that determines which VMs are allocated to edge servers [4], [7], [9]. The location of the VMs affects the performance of the edge computing environments such as the load on network links and edge servers.

In this paper, we propose a VM placement method that aims to smooth the load on network links and edge servers in edge computing environments. The proposed method first formulates the VM placement problem of smoothing the load as a mixed integer programming (MIP) problem. Furthermore, the proposed method provides a heuristic algorithm for obtaining approximation solution in relatively short time. Through numerical experiments based on a realistic network model, we show the effectiveness of the proposed method.

II. System model

Fig. 1 shows the system model assumed in this paper. Let $G = (\mathcal{V}, \mathcal{E})$ denote a given network, where $\mathcal{V}$ denotes a set of nodes and $\mathcal{E}$ denotes a set of links. Let $\mathcal{S} \subset \mathcal{V}$ and $\mathcal{H} \subset \mathcal{V}$ denote a set of edge servers and a set of host (i.e., user or IoT device). VMs are duplicated into edge servers and each host communicates with some VMs. Let $\mathcal{N}$ denote a set of services provided by the VMs, assuming that VM n provides service n. Each edge server can have one or more VM unless its capacity is full and provide different services according to the VMs. By duplicating VMs providing common

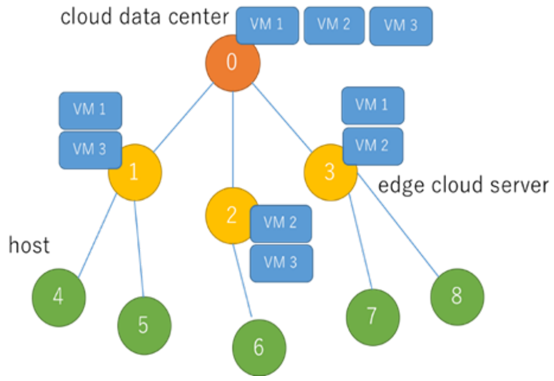

Fig. 1. System model.

TABLE I
LIST OF SYMBOLS.

Symbol	Meaning
$G = (\mathcal{V}, \mathcal{E})$	Network consisting of the set $\mathcal{V}$ of nodes and the set $\mathcal{E}$ of network links
$\mathcal{S} \subseteq \mathcal{V}$	Set of edge servers
$\mathcal{H} \subseteq \mathcal{V}$	Set of hosts
$\mathcal{N}$	Set of VMs
$\mathcal{N}_i$	Set of VMs communicated with host $i \in \mathcal{H}$
$\mathcal{P}_{i,j}$	Set of links on the shortest path from host $i \in \mathcal{H}$ to edge server $j \in \mathcal{S}$
$\lambda_{i,n}$	Traffic demands between host $i \in \mathcal{H}$ and VM $n \in \mathcal{N}_i$
C_j	Capacity of edge server $j \in \mathcal{S}$
S_n	Size of VM $n \in \mathcal{N}$
$x_{i,j,n}$	Binary variable that is equal to 1 if host $i \in \mathcal{V}$ communicates VM $n \in \mathcal{N}_i$ in edge server $j \in \mathcal{S}$; otherwise, 0
$\delta_{n,j}$	Binary variable that is equal to 1 if VM $n \in \mathcal{N}$ is allocated to edge server $j \in \mathcal{S}$; otherwise, 0
A, B	Weight parameters
α	Real variable that indicates the maximum link load
β	Real variable that indicates the maximum server load

services into multiple edge servers, we can balance the load on network links and edge servers. When requiring a service, each host communicates with a corresponding VM in one of edge servers.

In this system, it is very important to consider the location of the VMs. When allocating a popular VM into a small number of edge servers, a lot of load are imposed on not only the edge servers but also links around the edge servers. On the other hand, duplicating a non-popular VM into many edge servers could waste the capacity of the edge servers. Since the number of VMs that the edge servers can have is limited, VMs should be duplicated strategically according to their popularity and traffic demands.

III. PROPOSED METHOD

The symbols used in this paper are listed in Table I. The shortest path $\mathcal{P}_{i,j}$ from each host $i \in \mathcal{H}$ to each edge server $j \in \mathcal{U}$ is prepared in advance. The host communicates with the edge server along the path. In this section, we first provide MIP that minimizes the maximum link load, i.e., smoothing the load on the network links. We then provide MIP that minimizes the maximum edge server load, i.e., smoothing the load on the edge servers. Finally, we provide the heuristic algorithm that determines location of VMs so as to smooth load on both the network links and the edge servers. In the proposed method, we aim to minimizing the maximum link load α and/or the maximum server load β by coordinating the variables $\delta_{n,j}$ and $x_{i,j,n}$.

A. Network link load

We first provide the MIP that smooths the load on the network links. The MIP determines edge servers to which respective VMs are allocated. It also selects VMs with which respective hosts communicate. The MIP is given as follows:

Minimize

$$\alpha, \tag{1}$$

Subject to

$$\forall l \in \mathcal{E}; \quad \sum_{i \in \mathcal{H}} \sum_{n \in \mathcal{N}_i} \sum_{j \in \mathcal{S}: l \in \mathcal{P}_{i,j}} \lambda_{i,n} x_{i,j,n} \leq \alpha, \tag{2}$$

$$\forall j \in \mathcal{S}; \quad \sum_{n \in \mathcal{N}} \delta_{n,j} S_n \leq C_j, \tag{3}$$

$$\forall i \in \mathcal{H}, n \in \mathcal{N}_i; \quad \sum_{j \in \mathcal{S}} x_{i,j,n} = 1, \tag{4}$$

$$\forall n \in \mathcal{N}; \quad \sum_{j \in \mathcal{S}} \delta_{n,j} \geq 1, \tag{5}$$

$$\forall i \in \mathcal{H}, j \in \mathcal{S}, n \in \mathcal{N}_i; \quad \delta_{n,j} \geq x_{i,j,n}. \tag{6}$$

(1) is the objective function, which aims to minimize the maximum link load. (2) is a constraint ensuring that the load imposed on each link is equal to or less than α. (3) is a capacity constraint of each edge server, which represents that the total size of VMs allocated to the edge server is equal to or less than its capacity. (4) indicates that when requesting a service, each host communicates with only one of VMs providing the service. (5) ensures that each VM is duplicated into at least one edge server. (6) is a constraint that each host accesses only edge servers having target VMs.

B. Edge server load

We then provide the MIP that determines the VM location and the VM selection so as to smooth the load imposed on the edge servers, which is given as follows:

Minimize

$$\beta, \tag{7}$$

Subject to

$$\forall j \in \mathcal{S}; \quad \sum_{i \in \mathcal{H}} \sum_{n \in \mathcal{N}_i} \lambda_{i,n} x_{i,j,n} \leq \beta, \tag{8}$$

$$\forall j \in \mathcal{S}; \quad \sum_{n \in \mathcal{N}} \delta_{n,j} S_n \leq C_j, \tag{9}$$

Input: Set $\mathcal{S}$ of edge servers and set $\mathcal{N}$ of VMs
Output: VM placement
1: For each edge server $j \in \mathcal{S}, \mathcal{M}_j \leftarrow \mathcal{N}$, $C'_i \leftarrow C_i$
2: $j \leftarrow 0$
3: **while** $\mathcal{M}_j \neq \emptyset$ in each edge server $j \in \mathcal{S}$ **do**
4: **if** $\mathcal{M}_j \neq \emptyset$ **then**
5: Select VM n with the largest $\lambda_{j.n}$ among $\mathcal{M}_j$
6: $\mathcal{M}_j \leftarrow \mathcal{M}_j - \{n\}$
7: **if** there are no neighbors of server j having VM n && $C'_j \geq S_n$ **then**
8: Place VM n on server j
9: $C'_j \leftarrow C'_j - S_n$
10: $j \leftarrow (j+1) \mod |\mathcal{S}|$
11: **end if**
12: **else**
13: $j \leftarrow (j+1) \mod |\mathcal{S}|$
14: **end if**
15: **end while**

Fig. 2. VM placement algorithm.

$$\forall i \in \mathcal{H}, n \in \mathcal{N}_i; \quad \sum_{j \in \mathcal{S}} x_{i,j,n} = 1, \tag{10}$$

$$\forall n \in \mathcal{N}; \quad \sum_{j \in \mathcal{S}} \delta_{n,j} \geq 1, \tag{11}$$

$$\forall i \in \mathcal{H}, j \in \mathcal{S}, n \in \mathcal{N}_i; \quad \delta_{n,j} \geq x_{i,j,n}. \tag{12}$$

(7) is the objective function, which aims to minimize the maximum edge server load. In this paper we define the traffic volume imposed on an edge server as the edge server load. (8) is a constraint ensuring that the load imposed on each edge server is equal to or less than β. (9)-(12) are the same constraints as (3)-(6).

By solving these MIP, we can determine edge servers to be allocated respective VMs (i.e., $\delta_{n,j}$). At the same time, we can select VMs to communicate with respective hosts (i.e., $x_{i,j,n}$).

C. Network link and edge server load

We now consider how to smooth both the link load and the edge server load. We formulate the problem as the following MIP.

Minimize

$$A\alpha + B\beta \tag{13}$$

Subject to

$$\forall l \in \mathcal{E}; \quad \sum_{i \in \mathcal{H}} \sum_{n \in \mathcal{N}_i} \sum_{j \in \mathcal{S}: l \in \mathcal{P}_{i,j}} \lambda_{i,n} x_{i,j,n} \leq \alpha, \tag{14}$$

$$\forall j \in \mathcal{S}; \quad \sum_{i \in \mathcal{H}} \sum_{n \in \mathcal{N}_i} \lambda_{i,n} x_{i,j,n} \leq \beta, \tag{15}$$

$$\forall j \in \mathcal{S}; \quad \sum_{n \in \mathcal{N}} \delta_{n,j} S_n \leq C_j, \tag{16}$$

$$\forall i \in \mathcal{H}, n \in \mathcal{N}_i; \quad \sum_{j \in \mathcal{S}} x_{i,j,n} = 1, \tag{17}$$

$$\forall n \in \mathcal{N}; \quad \sum_{j \in \mathcal{S}} \delta_{n,j} \geq 1, \tag{18}$$

$$\forall i \in \mathcal{H}, j \in \mathcal{S}, n \in \mathcal{N}_i; \quad \delta_{n,j} \geq x_{i,j,n}. \tag{19}$$

(13) is the objective function, which aims to minimize the weighted sum of the maximum link load and the maximum edge server load. The maximum link load and the maximum edge server load are given by (14) and (15), respectively. (16)-(19) are the same constraints as (3)-(6).

Here, in order to reduce the computational complexity, we provide another solution instead of solving the MIP directly. We divide the problem into sub-problems: VM placement and VM selection. The proposed method first solves the VM placement sub-problem to determine which edge servers should be allocated respective VMs to. To do so, the proposed method introduces the heuristic algorithm, which is shown in Fig. 2, where edge servers are randomly indexed by j ($j = 0, 1, \ldots, |\mathcal{J}| - 1$). In the heuristic algorithm, VMs are duplicated into edge servers in such a way that VMs providing the same service are not allocated to edge servers adjacent to each other. By doing so, the heuristic algorithm distributes the load on edge servers.

The proposed method then solves the VM selection sub-problem. Specifically, it determines which VMs communicate with respective hosts by solving MIP based on the VM location (i.e., $\delta_{n,j}$) decided by the heuristic algorithm. The MIP smooths the load on network links, which is given as follows.

Minimize

$$\alpha, \tag{20}$$

Subject to

$$\forall l \in \mathcal{E}; \quad \sum_{i \in \mathcal{H}} \sum_{n \in \mathcal{N}_i} \sum_{j \in \mathcal{S}: l \in \mathcal{P}_{i,j}} \lambda_{i,n} x_{i,j,n} \leq \alpha, \tag{21}$$

$$\forall i \in \mathcal{H}, n \in \mathcal{N}_i; \quad \sum_{j \in \mathcal{S}} x_{i,j,n} = 1, \tag{22}$$

$$\forall i \in \mathcal{H}, j \in \mathcal{S}, n \in \mathcal{N}_i; \quad \delta_{n,j} \geq x_{i,j,n}. \tag{23}$$

The objective function (20) minimizes the maximum link load. (21)-(23) correspond to (14), (17), and (19), respectively.

IV. Performance Evaluation

A. Model

In this paper, we evaluate the performance of the proposed method through numerical experiments using networks created based on the Watts-Strogatz (WS) model [8]. The WS model can make networks having the small-world property that many real networks have. Each node fills the role of a host, an edge server, and an intermediate node. The amount $\lambda_{i,j}$ of traffic

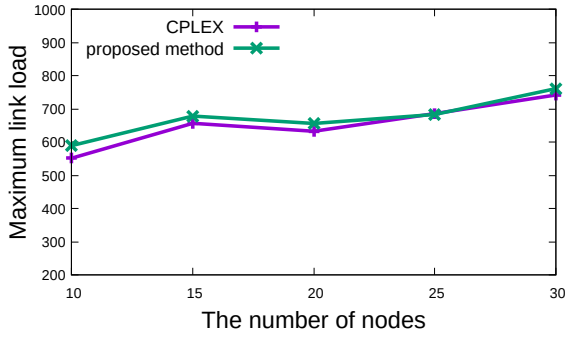

Fig. 3. Maximum link load against the number of nodes in small-size networks.

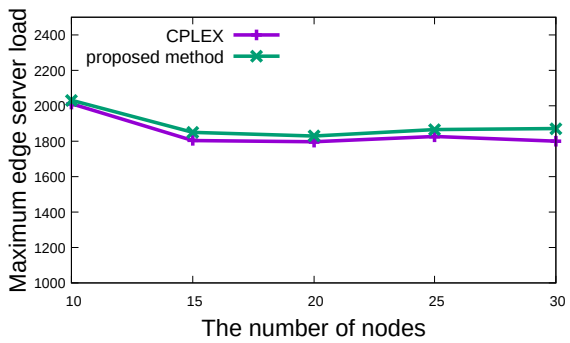

Fig. 4. Maximum edge server load against the number of nodes in small-size networks.

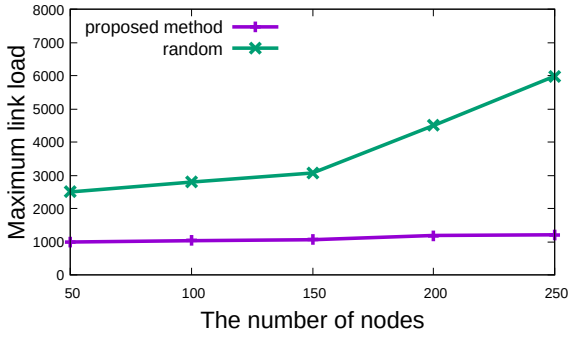

Fig. 5. Maximum link load against the number of nodes in large-size networks.

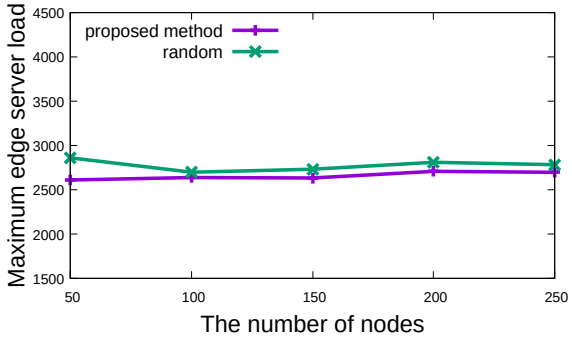

Fig. 6. Maximum edge server load against the number of nodes in large-size networks.

demands of hosts are randomly selected in such a way that the average traffic demand is equal to Λ. There exist $|\mathcal{N}| = N$ types of VMs in the network. The capacity C_j of each edge server j is equal to $\sum_{n\in\mathcal{N}} S_n/2$, where the size S_n of each VM n is randomly selected from [7,10]. We use IBM ILOG CPLEX [1] to solve the optimization problems formulated by MIP.

B. Results

We first evaluate the performance of the proposed method considering both the maximum link load and the maximum edge server load, which is discussed in Section III-C. We here use small-size networks where the number $|\mathcal{V}| = V$ of nodes are [10,30]. Figs. 3 and 4 show the maximum link load and the maximum edge server load as a function of the number V of nodes, where the average traffic demand Λ is 250 and the number N of VM types is 10. We plot the result (labeled with CPLEX) obtained by the MIP given by (13)-(19), where $A = B = 1$. We also the result (labeled with proposed method) obtained by the heuristic algorithm shown in Fig. 2 and the MIP given by (20)-(23). As we can see from these figures, their performances are almost the same, which means that the heuristic algorithm works well while reducing the computation time.

We then evaluate the performance of the heuristic algorithm in large-size networks where the number V of nodes are [50,250]. Figs. 5 and 6 show the maximum link load and the maximum edge server load as a function of the number V of nodes, where the average traffic demand Λ is 250 and the number N of VM types is 10. For the sake of comparison, we plot the result (labeled with random) of the method which places VMs on randomly selected edge servers and hosts communicate with the nearest VMs. From Fig. 5, we observe that the proposed method can efficiently improve the maximum link load, compared with the random method. On the other hand, the maximum edge server loads of the proposed method is slightly smaller than that of the random method. These results indicate that the proposed method can suppress both the maximum link load and the maximum edge server load.

Next, we compare the MIP formulations of the proposed method. Figs. 7 and 8 show the maximum link load and the maximum edge server load as a function of the number N of VM types, where the average traffic demand Λ is 250 and the number V of nodes is 50. In these figures, we plot the results labeled with Link-load-only and Sever-load-only of the MIP formulation given by Sections III-A and III-B, respectively. The result labeled with Link-and-Sever-load is obtained by the heuristic algorithm shown in Fig. 2 and the MIP formulation given by (20)-(23). As we can see from Fig. 7, the maximum link load of the Server-load-only method is very high because it does not take the load on network links into account. On

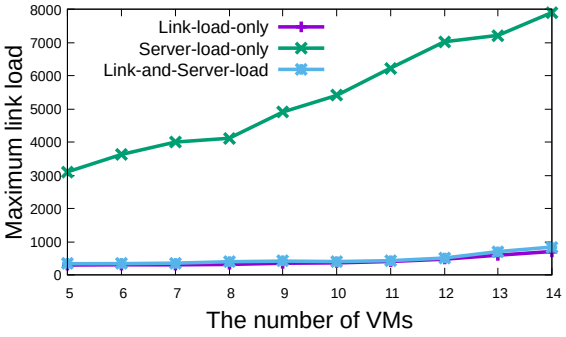

Fig. 7. Maximum link load against the number of VM types.

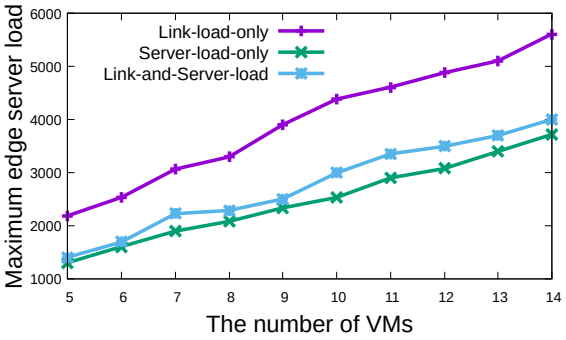

Fig. 8. Maximum edge server load against the number of VM types.

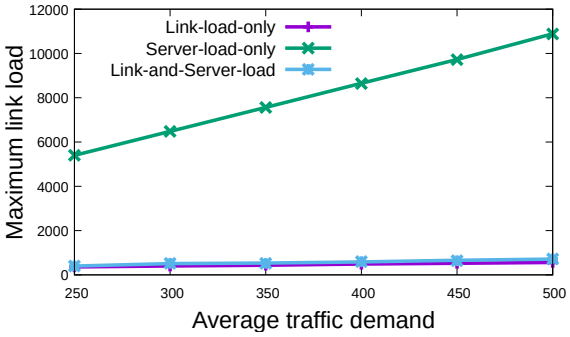

Fig. 9. Maximum link load against the average traffic demand.

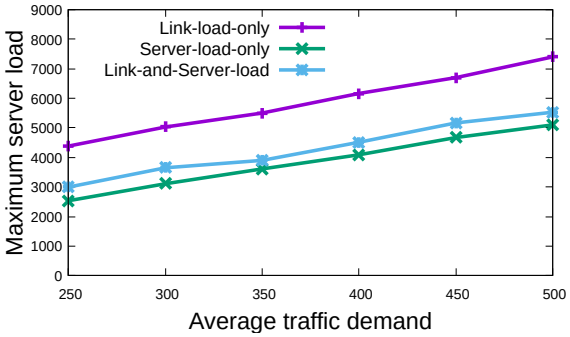

Fig. 10. Maximum edge server load against the average traffic demand.

the other hand, the Link-load-only method and the Link-and-Server-load method efficiently reduce the maximum link load. On the other hand, as shown in Fig. 8, the maximum edge server load of the Link-load-only method is very high. The maximum edge server load of the Server-load-only method is the smallest. We also observe that the Link-and-Server-load method relatively decreases the maximum edge server load, which is close to that of the Sever-load-only method.

Figs. 9 and 10 show the maximum link load and the maximum edge server load as a function of the average traffic demand Λ, where the number N of VM types is 10 and the number V of nodes is 50. From these figures, we observe that the Link-and-Server-load method can efficiently suppress both the maximum link load and the maximum server load.

V. Conclusion

In this paper, we proposed a VM placement method that smooths the load on network links and edge servers in edge computing environments. The proposed method first formulates the VM placement problem of smoothing the load as a mixed integer programming (MIP) problem. Furthermore, the proposed method provides a heuristic algorithm for obtaining approximation solution in relatively short time. Through numerical experiments based on a realistic network model, we showed the effectiveness of the proposed method.

Acknowledgement

This research was partially supported by Grant-in-Aid for Scientific Research (C) of the Japan Society for the Promotion of Science under Grant No. 18K11282.

References

[1] IBM ILOG CPLEX. https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer.

[2] A. Biswas and R. Giaffreda, "IoT and cloud convergence: Opportunities and challenges," in *Proc. 2014 IEEE World Forum on Internet of Things (WF-IoT)*, Mar. 2014.

[3] P. Corcoran and S. Datta, "Mobile-edge computing and the Internet of Things for consumers: extending cloud computing and services to the edge of the network," *IEEE Consumer Electronics Magazine*, vol. 5, no. 4, pp. 73–74, 2016.

[4] G. Gao, M. Xiao, J. Wu, H. Huang, S. Wang and G. Chen, "Auction-based VM allocation for deadline-sensitive tasks in distributed edge cloud," *IEEE Transactions on Services Computing*, 2019 (DOI: 10.1109/TSC.2019.2902549).

[5] L. Li, Y. Shi, J. Wang and Z. He, "A VM-friendly NIC architecture for cloud computing," in *Proc. 2017 IEEE 2nd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*, Apr. 2017.

[6] W. Shi, J. Cao, Q. Zhang, Y. Li and L. Xu, "Edge computing: vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[7] D. Tan, F. Liu, N. Xiao and Y. Xie, "Optimizing virtual machine live migration in distributed edge servers based on hybrid memory," in *Proc 2019 International Conference on High Performance Big Data and Intelligent Systems*, May 2019.

[8] D. Watts and S. Strogatz, "Collective dynamics of ' small world ' networks," *Nature*, vol. 393, pp. 440–442, 1998.

[9] L. Yang, D. Yang, J. Cao, Y. Sahni and X. Xu, "QoS guaranteed resource allocation for live virtual machine migration in edge clouds," *IEEE Access*, vol. 8, pp. 78441-78451, 2020.