Mobile Networks and Applications (2019) 24:643–652
https://doi.org/10.1007/s11036-018-1164-2

Power Efficient Clustering Scheme for 5G Mobile Edge Computing Environment

Jaewon Ahn[1] · **Joohyung Lee**[2] · **Sangdon Park**[3] · **Hong-Shik Park**[1]

Published online: 31 October 2018

Abstract
Mobile edge computing (MEC), which is an evolution of cloud computing, is acknowledged as a promising technology for meeting low latency and bandwidth efficiency required in fifth generation (5G) era. Accordingly, the enlargement of distributed MEC installments will be realized and their power consumption might be a significant problem in terms of operating costs for service providers. Thus, this paper proposes a theoretical framework for MEC server clustering to minimize power consumption of the MEC environment. To do this, considering power consumption behavior of MEC servers using CPUs with dynamic voltage frequency scaling, we propose a power-efficient clustering scheme (PECS) that minimizes power consumption of MEC servers by obtaining the optimal number of clusters through convex optimization. Numerical results reveal the proposed PECS reduces power consumption of the MEC environment by 12.32% relative to an existing scheme while sustaining average delay of inflows processed in MEC servers at the acceptable level without turning off MEC servers.

Keywords Virtual Network Functions (VNFs) · Mobile Edge Computing (MEC) · Power-efficient clustering

1 Introduction

Mobile edge computing (MEC), which sustains multiple distributed edge servers by moving computing points from centralized points to near end-user points, has emerged as a promising technology for pervasive fifth generation (5G) services such as internet of things (IoT), device-to-device (D2D), and low-latency video streaming (e.g., virtual reality / augmented reality video streaming) etc., to meet low latency and bandwidth efficiency [1–4]. Subsequently, the enlargement of distributed MEC installments will be realized and their power consumption might be a significant issue in terms of operating costs for service providers. Thus, with much attention being directed towards energy efficient data center management, power consumption is also expected to be an important criterion in MEC management along with delay.

According to iGR [5], the number of U.S. MEC installations is expected to reach 563,000 divided into 2,000 locations by 2026. They generate more than 2.66 TWh annually in the U.S. alone and the corresponding cost amounts to more than 217 million dollars when the average power consumption of each MEC installation is 300W and the power utilization effectiveness (PUE) is 1.8 [6]. Moreover, the PUE of MEC is getting worse since MEC is restricted to being located physically near the end-user point, whereas a traditional data center can reduce power consumption by more than 20% by locating the data center in a cool climate region with free cooling [7].

To date, researches on MEC has mostly focused on reducing delay [8, 9]. MEC is implemented as a virtualized platform instead of a custom hardware appliance, leveraging recent advances in network function virtualization (NFV) [1, 10, 11]. Correspondingly, it is possible to handle delay

✉ Joohyung Lee
j17.lee@gachon.ac.kr

Jaewon Ahn
anjwon@kaist.ac.kr

Sangdon Park
sangdon.park@kaist.ac.kr

Hong-Shik Park
park1507@kaist.ac.kr

[1] School of Electrical Engineering, Korea Advanced Institute of Science and Technology, Daejeon 34141, South Korea

[2] Department of Software, Gachon University, Seongnam 13120, South Korea

[3] Information and Electronics Research Institute, Korea Advanced Institute of Science and Technology, Daejeon 34141, South Korea

of MEC with optimal virtual network function (VNF) placement where a mobile edge application (ME app) is considered one kind of VNF [12]. It should be noted that, for convenience, we refer to both VNFs and ME apps by the single term "virtual element" (VE) throughout this paper. Recently, Nam et al. [9] proposed a clustering based VE placement scheme for MEC with the consideration of self-similar inflow traffic. The clustering technique links many edge servers together to act as a single node. With this clustering approach, the traffic flow is induced to be processed in the same cluster (i.e., neighbor edge servers) so that both delay and the volume of traffic through MEC are remarkably reduced. However, they focused solely on reducing delay through clustering, and thus did not consider power consumption in MEC. In this regard, there is still room for improving the work in [9], which inspired our scheme.

Approaches considering the power consumption issues in 5G MEC environment were proposed in [13–15]. Long D. Nguyen [15] proposed the novel hybrid resource allocation approaches for improving energy efficient performance in potential 5G network scenarios such as small cell, massive multiple-input and multiple-output (MIMO), and cell-free network. Ke et al. [13] studied energy efficient MEC computation offloading schemes for mobile devices in 5G heterogeneous networks. Mao et al. [14] investigated a MEC system with energy harvesting devices and developed an effective computation offloading strategy by utilizing a low-complexity online algorithm. While some studies related to the power consumption issues in MEC have been carried out, all of the previous works mainly concentrated on power consumption at mobile terminals rather from the perspective of the MEC server.

In this paper, we propose a theoretical framework for MEC clustering to minimize overall power consumption of the MEC environment without turning off any physical servers. This is due to fact that turning off the MEC server is likely to cause the degradation of quality of service in traffic arrival variations. The contributions of this paper are as follows.

- Considering the power consumption behavior of the MEC server using CPUs with dynamic voltage frequency scaling (DVFS) [16], we propose a power-efficient clustering scheme (PECS) for the MEC environment while sustaining average delay of inflows processed in MEC servers at the acceptable level. By analyzing the impact of the number of clusters on the CPU loads, the optimization problem for finding the optimum number of clusters to minimize the power consumption of MEC servers is formulated and solved.
- A rigorous analysis of the proposed model, under specified conditions that satisfy convexity of power consumption model, demonstrates that proposed PECS can efficiently determine the optimal number of clusters. On the other hand, under the other conditions, we also propose the exhaustive search algorithm to obtain the optimal number of clusters within a limited search space (e.g. 563,000 MEC installment in [5]).
- The proposed scheme reduces power consumption by 12.32% compared with an existing scheme while sustaining average delay of inflows processed in MEC servers at the acceptable level.

The rest of this paper is organized as follows. In Section 2, we describe the system model of the MEC environment including the resource provisioning scheme to each VM and the average delay of inflows processed in MEC servers. After that, we propose the power consumption model of MEC servers by using empirical power consumption equation according to the CPU load of the physical server in Section 3. In Section 4, we formulate the problem of PECS and we provide the method to obtain the optimal number of clusters. Numerical results are presented in Section 5. Finally, we conclude this paper in Section 6.

2 System model

In the MEC environment, VNFs such as a virtual firewall, virtual load balancer, and virtual switch, connect mobile edge host and ME apps internally within the data plane through service function chaining (SFC). In [12], the mobile edge host contains both of MEC apps and the data plane as Fig. 1. In this paper, mobile edge host is assumed to be set up on each physical server. Because the destination

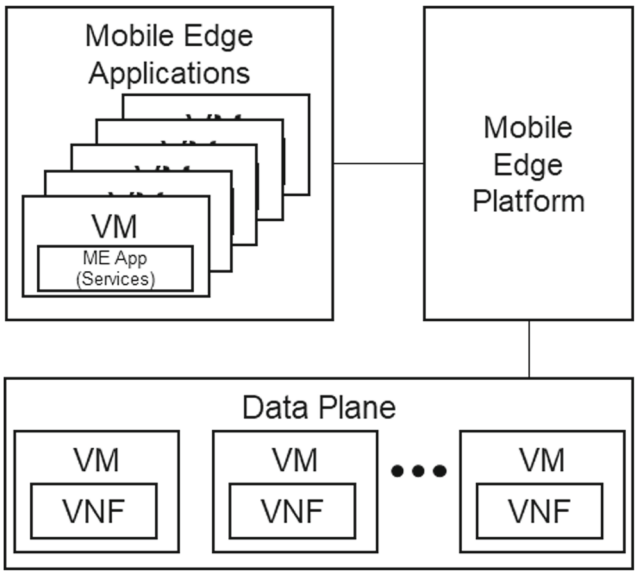

Fig. 1 Block diagram of mobile edge host

of the inflow in the MEC environment is an ME app, we assume the NFV service chain in MEC includes the chain of VNFs and ME apps as VEs. In the paper, we assume that VEs are instantiated entirely over the same virtualization infrastructure [11].

We develop a system model for power efficient clustering using the cluster based service chain model [9]. As in [9], MEC is composed of non-overlapped k clusters, and m physical MEC servers. MEC servers are divided into two types. In each cluster, there is one local clusterhead (LC) server, which has the VE distribution information and controls the inflow in the cluster, and other $m/k-1$ physical servers are classified as member servers that have different kinds of VEs. There are n VEs, which have their own popularity in this model, and the set of VEs is denoted as $\mathcal{N} = \{1, 2, ..., n\}$. It is assumed that the popularity of VEs follows Zipf's law, because it is used for modeling popularity based VE distributions [17]. The popularity of the jth popularly used VE for $j \in \mathcal{N}$ is given by

$$p_j = \frac{1-\alpha}{j^{\alpha}(n^{1-\alpha}-1)},$$

where α is a Zipf parameter.

In a physical server, there are c VMs and each VM can have one VE for the robustness of the MEC environment. When a VE needs to be changed to other VE on the VM, it needs some process to replace VE and it generates the power consumption because it needs CPU resource of the physical server. In this paper, we assume that the CPU load of each physical server is evenly provisioned to c VMs, that is, a VM can utilize CPU resource up to $1/c$. We also suppose that each VM fully operates when a request comes into the VE installed in each VM and each VM does not utilize CPU resource at all when there is no request to each VE installed in each VM. Every LC server contains the most popular c VEs redundantly. The VE installed in the LC server is denoted as LVE. Each member server contains exclusively unique c VEs excluding LVEs, which are denoted as MVE. The least popular MVEs can be uninstalled in member servers if there is no VM to occupy. If a request requires an uninstalled MVE, the least popular MVE among installed MVEs is replaced by the requested MVE. $\mathcal{N}$ can be divided into $\mathcal{N}_{\mathcal{L}} = \{1, 2, ..., c\}$ and $\mathcal{N}_{\mathcal{M}} = \{c+1, c+2, ..., n\}$, which are the set of LVEs and MVEs.

The number of flows that come into MEC servers during a average delay cycle, $T(k)$, is n_f. $T(k)$ means the expectation of the duration of processing n_f flows within the MEC environment. A flow comes into a MEC cluster and requires to process its requests to MEC servers in the MEC cluster. If a VE required by the flow is existed in the cluster, then the VE processes the request from the flow. However, there is no VE in the cluster, the flow goes to neighbor cluster which has required VE and returns to the cluster which it comes in initially. Each flow has requests to average n_{req} VEs, and thus $n_f n_{req}$ requests are processed in a period of $T(k)$. The inflow traffic of the MEC environment is assumed to be self-similar with an average data rate of r bps, because the MEC environment serves various 5G services that have different session inter-arrival times caused by having a different set of required VEs. We use a fractional Brownian motion (fBm) [18] for modeling self-similar traffic.

$T(k)$ [9] can be obtained by the fBm model and it is given by

$$T(k) = 2\mu n^{1-\alpha}k^{-1} - 2\nu(mc)^{1-\alpha}k^{\alpha-1} + 2\nu n^{1-\alpha}$$

$$\text{where } \mu = \frac{hn_{req}\omega^{-1}n_f\left(\frac{l_{cap}-r}{H}\right)^{\frac{2H}{H-2}}}{n^{1-\alpha}-1}, \nu = \frac{hn_{req}\beta}{n^{1-\alpha}-1},$$

$$\text{and } \omega = \left(\left(-2\sigma^2\ln\epsilon_a\right)^{\frac{1}{2H-2}}\right)/(1-H).$$

ϵ_a is the overflow probability of the fBm model, given by

$$\epsilon_a = \exp\left(-\frac{1}{2\sigma^2}\left(\left(l_{cap} - \max_{1\le i\le n_f}(r_i)\right)/H\right)^{2H} \times \left(\frac{b}{1-H}\right)^{2-2H}\right).$$

h is the average number of hop counts between servers in MEC, l_{cap} is the average capacity of a link, r_i is the inflow rate of the i-th flow and σ is the average standard deviation of the flow, β is the deterministic sum of propagation, transmission, and processing delays, b is the buffer size, and H is a Hurst parameter for the fBm model.

In this paper, we assume that there are at least one LC server and one member server in the cluster; that is, $1 \le k \le m/2$. There are $mc-n$ vacant VMs when there is one cluster in the MEC environment. As the number of cluster increases, there are $mc-(k-1)c-n$ vacant VMs in the MEC environment as described in Fig. 2, because the number of LC server also increases. If k is more than $m+1-n/c$, $(k-1)c$ VMs are occupied by LVEs, and thus n_U MVEs are uninstalled to MEC, where

$$n_U = \max(n-(m-k+1)c, 0).$$

$\mathcal{N}_{\mathcal{M}}$ can be divided into two set of MVEs, that are the set of deployed MVEs in member servers $\mathcal{N}_{\mathcal{D}}$ and the set of uninstalled MVEs in member servers $\mathcal{N}_{\mathcal{U}}$. We propose the power consumption model by analyzing the CPU load generated by LVEs, installed MVEs and uninstalled MVEs in next section.

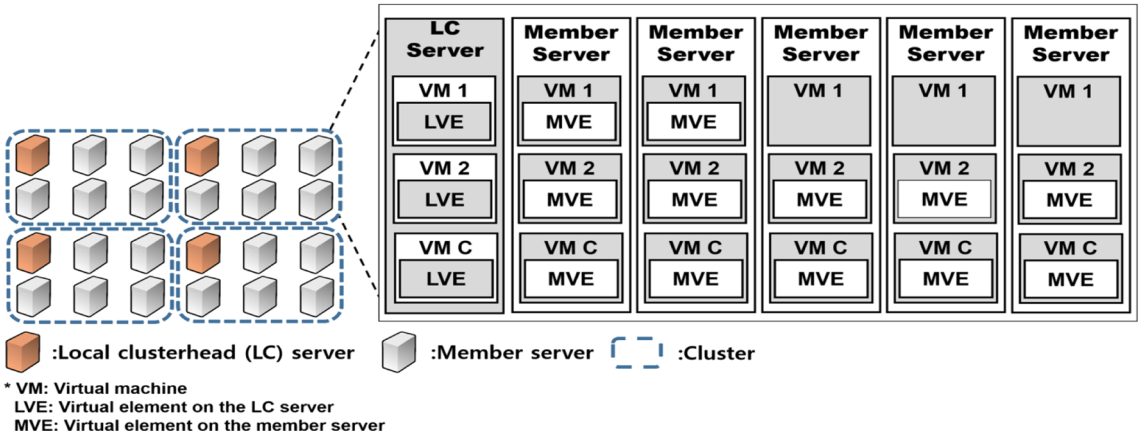

Fig. 2 System model of PECS

3 Proposed power consumption model

For the sake of minimizing the power consumption of MEC servers, it is necessary to estimate the amount of power consumption of MEC servers and formulate the power consumption model of MEC servers according to the number of clusters in the MEC environment. The first step to figure out the power consumption model of MEC servers is estimating the average CPU load of a MEC server.

Average CPU load of a MEC server can be measured by the average number of requests sent to a server in MEC during one average delay cycle, $T(k)$. If we know the amount of CPU load of each VE j for $j \in \mathcal{N}$ when a request is processed, it is possible to determine the average power consumption of the server with a power consumption equation according to the CPU load by aggregating the CPU load generated by each VE installed in the server. By aggregating the power consumption of each server, the power consumption of the MEC environment for PECS is provided by acquiring the average CPU load of LC servers and member servers. It can be shown as

$$P_{mec}(k) = kP(l_{LC}) + (m-k)P(l_{Member}),$$

where $P_{mec}(k)$ is the power consumption of MEC servers, $P(u)$ is the power consumption model of the MEC server according to its CPU load u, l_{LC} is the average CPU load of LC servers for one second, l_{Member} is the average CPU load of member servers for one second, m is the number of MEC servers in the MEC environment, and k is the number of LC servers within the MEC environment.

3.1 Average CPU load of a LC server

First, in order to obtain the average power consumption of a LC server during an average delay cycle, it is necessary to determine the average CPU load generated by each LVE j, which is the j-th popular VE and deployed on the LC server, where $j \in \mathcal{N}_{\mathcal{L}}$, during a cycle. To do this, it is necessary to figure out the probability that LVE j is busy. The average number of requests r_{lj} to one LVE j during $T(k)$ is

$$r_{lj} = \frac{n_f}{k} n_{req} p_j.$$

Thus, the probability that LVE j for $j \in \mathcal{N}_{\mathcal{L}}$ is busy during one second is given by

$$p_{lve_j} = \frac{r_{lj} t_j}{T(k)} = \frac{(\frac{n_f}{k}) n_{req} p_j t_j}{T(k)},$$

where p_{lve_j} is the probability of LVE j being busy for one second, and t_j is the processing time of LVE j when a request comes into LVE j. Because a flow traverses all n_{req} VEs during $T(k)$, t_j for $j \in \mathcal{N}$ can be obtained by

$$t_j = \frac{\eta_j T(k)}{n_{req}}$$

, where η_j is the portion of time dedicated to process a request in LVE j. When the CPU load generated by LVE j is $\hat{l}_j$ during LVE working, then a LVE generates CPU load during one second, l_j, is given by

$$l_j = p_{lve_j} \hat{l}_j = \frac{n_f p_j \hat{l}_j \eta_j}{k}.$$

Since we assume a VM with a LVE fully operates when a request comes into a LVE, $\hat{l}_j$ is equal to 1, consequently,

$$l_j = \frac{n_f p_j \eta_j}{k}.$$

The operation of each LVE in a LC server is mutually exclusive, and hence the probability of a LC server being busy, that is, the average CPU load of a LC server for one second, l_{LC} is given by

$$l_{LC} = \sum_{j=1}^{c} l_j, \text{ where } j \in \mathcal{N}_{\mathcal{L}}$$

If $\eta_j = \eta$ and we let $\sum_{j=1}^{c} p_j$ be denoted by p_L for $j \in \mathcal{N}_{\mathcal{L}}$, then the average CPU load of a LC server for one second is given by

$$l_{LC} = \frac{n_f p_L \eta}{k}. \tag{1}$$

3.2 Average CPU load of a member server

The average CPU load of a member server, l_{Member} can be obtained through different processes from that of a LC server because deployed MVEs are frequently changed to other MVEs according to demands of inflows. It is hard to designate the MVE which always stay on the member server, thus it is impossible to directly utilize the probability that the MVE is busy. To comprehensively figure out these factors, we divide the average load of a member server, l_{Member} to two parts, that is, average CPU load for replacing MVEs, l_{rep}, and average CPU load of a member server generated by operating deployed MVEs, l_M. We firstly figure out l_{rep} using the summation of popularity of uninstalled MVE j, where $j \in \mathcal{N}_{\mathcal{U}}$. After that, we obtain l_M using the average popularity of MVE j, where $j \in \mathcal{N}_{\mathcal{D}}$, which are installed to member servers by subtracting the popularity of LVEs and the popularity of uninstalled MVEs from 1.

3.2.1 Average CPU load for replacing MVEs

When an incoming request requires a MVE that is uninstalled in member servers, the dynamic replacement of MVEs occurs in member servers with increment of the CPU load. To determine the CPU load by replacing MVEs, it is necessary to obtain the number of replacements, which is equivalent to the number of requests to all MVEs uninstalled in member servers.

Let p_U be the summation of popularity of n_U uninstalled MVEs. Because uninstalled MVEs are dynamically changed, we try to figure out p_U by bounding the range of p_U. Because n_U least popular MVEs are uninstalled initially, the lower bound of p_U can be obtained by

$$\sum_{j \in \mathcal{N}_{\mathcal{UL}}} p_j, \text{ where } \mathcal{N}_{\mathcal{UL}} = \{n - n_U + 1, n - n_U + 2, ..., n\}.$$

If a request requires MVE j', where $j' \in \mathcal{N}_{\mathcal{UL}}$, then the least popular MVE deployed on a member server, MVE $n - n_U$, is replaced by MVE j'. If another request requires MVE j'', where $j'' \in \mathcal{N}_{\mathcal{UL}}$ and $j'' \neq j'$, then MVE j' is replaced by MVE j''. Hence, the upper bound of p_U can be obtained by the summation of popularity of MVE j for $j \in \mathcal{N}_{\mathcal{UU}}$,

$$\sum_{j \in \mathcal{N}_{\mathcal{UU}}} p_j, \text{ where } \mathcal{N}_{\mathcal{UU}} = \{n - n_U, n - n_U + 1, ..., n - 1\}.$$

Thus, the boundary of p_U can be obtained by

$$\sum_{j \in \mathcal{N}_{\mathcal{UL}}} p_j \leq p_U \leq \sum_{j \in \mathcal{N}_{\mathcal{UU}}} p_j.$$

Because the difference between $\sum_{j \in \mathcal{N}_{\mathcal{UL}}} p_j$ and $\sum_{j \in \mathcal{N}_{\mathcal{UU}}} p_j$ is $p_{n-n_U} - p_n$, it can be negligible. Thus, p_U can be obtained by

$$p_U = \sum_{j \in \mathcal{N}_{\mathcal{U}}} p_j$$

, where $\mathcal{N}_{\mathcal{U}}$ denotes the set of uninstalled MVEs and $\mathcal{N}_{\mathcal{U}} = \{n - n_U + 1, n - n_U + 2, ..., n\}$. The average number of requests to $\mathcal{N}_{\mathcal{U}}$ MVEs during $T(k)$, r_u, can be obtained by

$$r_u = n_f n_{req} p_U.$$

Let the amount of time consumed for replacing an MVE be $\psi T(k)$, where ψ is the portion of time dedicated to replace an MVE, and $\hat{l}_{rep}$ be the CPU load of a server generated by the replacement of an MVE; then the average CPU load generated by replacing MVEs for one second in each member server can be obtained by

$$l_{rep} = \frac{\psi r_u \hat{l}_{rep} T(k)}{(m-k)T(k)} = \frac{\psi r_u \hat{l}_{rep}}{m-k} \tag{2}$$

3.2.2 Average CPU load by processing requests in member servers

In order to figure out the average CPU load by processing requests in member servers, it is necessary to obtain the average popularity of each member server. This can be done by dividing the entire popularity of MVEs installed in member servers by the number of member servers. The entire popularity of MVEs installed in member servers can be obtained by $1 - p_L - p_U$ and the average popularity of each member server, p_M, can be obtained by

$$p_M = \frac{1 - p_L - p_U}{m - k}. \tag{3}$$

Because each MVE in a member server is mutually exclusive, so the popularity of a member server can be divided into the popularity of MVEs deployed in the member server for calculating the probability of a certain MVE being busy. Let $p_{j_{iv}}$ be the popularity of v-th MVE in i-th member server, where $j_{iv} \in \mathcal{L}_{\mathcal{M}}$ for $i \in \{1, 2, ..., m - k\}$ and $v \in \{1, 2, ..., c\}$; the probability that MVE j_{iv} is busy during one second is then given by

$$p_{mve_{j_{iv}}} = \frac{n_f n_{req} p_{j_{iv}} t_{j_{iv}}}{T(k)},$$

where $t_{j_{iv}}$ is the processing time of the request of MVE j_{iv}. Let $t_{j_{iv}} = \eta_{j_{iv}} T(k)/n_{req}$, where $\eta_{j_{iv}}$ is the portion of time dedicated to process a request in MVE j_{iv} and $\hat{l}_{j_{iv}}$ is the CPU load generated by a request that comes into MVE

j_{iv}, then the CPU load generated by a MVE j_{iv} during one second, $l_{j_{iv}}$, is given by

$$l_{j_{iv}} = p_{mve_{j_{iv}}} \hat{l}_{j_{iv}} = n_f p_{j_{iv}} \eta_{j_{iv}} \hat{l}_{j_{iv}}$$

Since we assume a VM with a MVE fully operates when a request comes into a MVE, $\hat{l}_{j_{iv}}$ is equal to 1, consequently,

$$l_{j_{iv}} = n_f p_{j_{iv}} \eta_{j_{iv}}$$

The operation of each MVE in a member server is mutually exclusive, and therefore the average CPU load of the member server M_i can be obtained by

$$l_{M_i} = \sum_{v \in \{1,2,\ldots,C\}} n_f \eta_{j_{iv}} p_{j_{iv}}.$$

Because

$$\sum_{i \in \{1,\ldots,m-k\}} \sum_{v \in \{1,\ldots,c\}} p_{j_{iv}} = 1 - p_L - p_U$$

and Eq. 3, if $\eta_{j_{iv}} = \eta$, the average CPU load of a member server generated by operating deployed MVEs for one second is obtained by

$$l_M = n_f \eta p_M. \tag{4}$$

3.3 Power consumption of MEC servers

Finally, using Eqs. 1, 2, and 4, we can obtain the power consumption model of MEC servers, which is given by

$$P_{mec}(k) = kP(l_{LC}) + (m-k)P(l_M + l_{rep}), \tag{5}$$

where $P(u)$ [19] is the empirical power consumption equation of a physical server according to CPU load with a dynamic voltage frequency system (DVFS) [16] that is adopted at CPUs of the data center for energy savings;

$$P(u) = P_i + \frac{P_b - P_i}{2}\left(1 + u^3 - \exp\left(\frac{-u^3}{a}\right)\right),$$

where P_i and P_b are the power consumption of a physical server when the CPU is idle and busy, respectively, and a is the level of utilization of power consumption model, $a \in [0.2, 0.5]$. The power consumption model of MEC servers is then given by

$$P_{mec}(k) = kP\left(\frac{n_f \eta p_L}{k}\right) + (m-k) \times P\left(\frac{n_f \eta (1 - p_L - p_U) + \psi n_f n_{req} p_U \hat{l}_{rep}}{m-k}\right). \tag{6}$$

Algorithm 1 PECS algorithm

Phase 1: Define constraint set $\mathcal{K}$
set $\mathcal{K}$ of available k based on constraints in (7)
if $\mathcal{K} \subset \phi$ **then ABORT**
Phase 2: Figure out optimal number of clusters k
if $1 \le n < (1 + m/2)c$ AND $a_{th} \le a$ **then**
  $k_{conv} = arg_k[\partial P_{mec}(k)/\partial k = 0]$
  if $P_{mec}(\lfloor k_{conv} \rfloor) > P_{mec}(\lceil k_{conv} \rceil)$ **then** $k_{temp} \leftarrow \lceil k_{conv} \rceil$
  else $k_{temp} \leftarrow \lfloor k_{conv} \rfloor$
  end if
  if $k_{temp} \in \mathcal{K}$ **then** $k_{opt} \leftarrow k_{temp}$
  else if $k_{temp} < min(\mathcal{K})$ **then** $k_{opt} \leftarrow min(\mathcal{K})$
  else $k_{opt} \leftarrow max(\mathcal{K})$
  end if
else
  find k_{opt} within $\mathcal{K}$ by **exhaustive search algorithm**
end if
return k_{opt}

4 Proposed Power Efficient Clustering Scheme (PECS)

We formulate PECS in the MEC environment as follows:

$$\begin{aligned} \min_k \quad & P_{mec}(k) \\ \text{s.t.} \quad & T(k) \le T_{req}, 1 \le k \le \frac{m}{2} \\ & 0 \le l_{LC}, l_M + l_{rep} \le 1, \end{aligned} \tag{7}$$

where k is the number of clusters in the MEC environment, $P_{mec}(k)$ is the power consumption mode of MEC servers, $T(k)$ is the expectation of the duration of processing n_f flows within the MEC environment, m is the number of physical MEC servers, l_{LC} is the average CPU load of a LC server for one second, l_M is the average CPU load of a member server generated by operating deployed MVEs for one second, and l_{rep} is the average CPU load for replacing MVEs for one second.

The goal of the proposed utility function is to minimize the power consumption of the MEC environment while sustaining average delay of inflows processed in MEC servers, $T(k)$, at the acceptable level, T_{req}. To do this, it is necessary to determine the optimal k, which is the number of clusters in the MEC environment. We investigate and find the condition that $P_{mec}(k)$ is convex in order to efficiently obtain the optimal number of clusters to minimize the power consumption of MEC servers. Thus, we solve the problem in two cases. Under the condition that satisfies convexity of power consumption model, PECS determines the optimal number of clusters by convex optimization. On the other

hand, under the other conditions, the exhaustive search algorithm finds the optimal number of clusters.

When $1 \leq n < (1 + m/2)c$, there are no uninstalled MVEs in the MEC environment; that is, n_U and p_U is 0 for $k \in \mathcal{K}$, where $\mathcal{K}$ is defined based on constraints in Eq. 7. In this case, the problem can be solved by verifying the convexity of $P_{mec}(k)$ and Lemma 1.

Lemma 1 *Let* $1 \leq n < (1 + m/2)c$, $P_{mec}(k)$ *is convex if* $P(u)$ *is convex, where* $u \in [0, 1]$ *and* $k \in \mathcal{K}$.

Proof When $1 \leq n < (1+m/2)c$, $P_{mec}(k)$ can be given by

$$kP\left(\frac{n_f \eta p_L}{k}\right) + (m-k)P\left(\frac{n_f \eta(1-p_L)}{m-k}\right).$$

Thus

$$P''_{mec}(k) = P''\left(\frac{n_f \eta p_L}{k}\right)\frac{(n_f \eta p_L)^2}{k^3} + P''\left(\frac{n_f \eta(1-p_L)}{m-k}\right)\frac{(n_f \eta(1-p_L))^2}{(m-k)^3}.$$

Because k is positive and $m-k$ is also positive, k^3 and $(m-k)^3$ are also positive. $(n_f \eta p_L)^2$ and $(n_f \eta(1-p_L))^2$ are also positive. If $P(u)$ is convex, $P''\left(\frac{n_f \eta p_L}{k}\right)$ and $P''\left(\frac{n_f \eta(1-p_L)}{m-k}\right)$ are also positive. Thus, if $P(u)$ is convex, then $P''_{mec}(k)$ is always positive. That is, if $P(u)$ is convex, then $P''_{mec}(k)$ is convex. □

Because $P(u)$ is an empirically measured equation, it is not always convex. We find that $P(u)$ has convexity for $u \in [0, 1]$ when $a_{th} \leq a$ and $1 \leq n < (1 + m/2)c$. We find $a_{th} \simeq 0.289$ empirically investigated by MATLAB.

Under the other conditions, we thus propose the efficient exhaustive search algorithm to obtain the optimal k value in the MEC environment without concern of the convexity of $P(u)$. The efficient exhaustive search algorithm reduces the possible candidates of optimal number of clusters by defining $\mathcal{K}$ based on constraints in Eq. 7 and find the optimal number of clusters k, where $k \in \mathcal{K}$ and $\mathcal{K} \subset \mathbb{N}$, where $\mathbb{N}$ is the set of natural numbers. Therefore, it is possible to figure out the optimal number of clusters using PECS within a limited search space (e.g. 563,000 MEC installment in [5]). PECS is described in Algorithm 1.

In first phase, we define the constraint set $\mathcal{K}$ to reduce the possible candidates of optimal solution. In second phase, in order to efficiently solve the problem, we divide second phase into two cases. When $1 \leq n < (1 + m/2)c$ and $a_{th} \leq a$, $P_{mec}(k)$ is convex. Thus, in this case, k_{conv} can be obtained by differentiation of $P_{mec}(k)$, where k_{conv} is an intermediate optimal number of clusters and $k_{conv} \in \mathbb{R}$, where $\mathbb{R}$ is the set of real numbers. After matching k_{conv} to k_{temp}, where k_{temp} is an intermediate optimal number of clusters and $k_{conv} \in \mathbb{N}$, PECS checks the availability of k_{temp}. If k_{temp} is in $\mathcal{K}$, k_{temp} becomes k_{opt} and if not, one of the boundaries of set $\mathcal{K}$ becomes k_{opt}, where k_{opt} is the optimal number of clusters. Under the other conditions, PECS finds k_{opt} within $\mathcal{K}$ by exhaustive search algorithm.

5 Numerical results

To evaluate the performance of the proposed PECS, the power consumption according to the number of clusters is compared by utilizing a MATLAB simulation. The performance is examined over a random network formed by 100 servers that are positioned following a uniformly random distribution over a normalized circular area with n_f flows. We assume that the number of hops between MEC servers is proportional to the linear distance between them and, according to [20], the average linear distance between MEC servers is $64/45\pi$. We follow the empirically measured power consumption of the physical server [21] System x3530 M4 which uses two Intel Xeon E5-2470 processors. The parameter values are listed in Table 1.

Because $P(u)$ with $a = 0.3$ is convex, it is possible to figure out the convexity of the power consumption

Table 1 Parameter values

Symbols	Value	Specifications
n_{req}	30	the average number of requests in each flow
c	10	the number of VMs in a physical server
a	0.3	level of utilization of power consumpti-on model
P_b	274W	the power consumption of a physical server when the CPU is fully utilized
P_i	62.6W	the power consumption of a physical server when the CPU is idle
η	0.0006	the portion of time dedicated to process a request in the VE
ψ	0.01	the portion of time dedicated to replace an MVE
$\hat{l}_{rep}$	0.005	the CPU load of a server generated by the replacement of an MVE
H	0.8	Hurst parameter
α	0.8	parameter of Zipf's law
β	30ms	the deterministic sum of delays per one hop between MEC servers
h	$64/45\pi$	average number of hop count between MEC servers in the MEC environment
l_{cap}	10Gbps	the average capacity of a link between MEC servers
r	$1 \sim 3Gbps$	the data rate of a flow

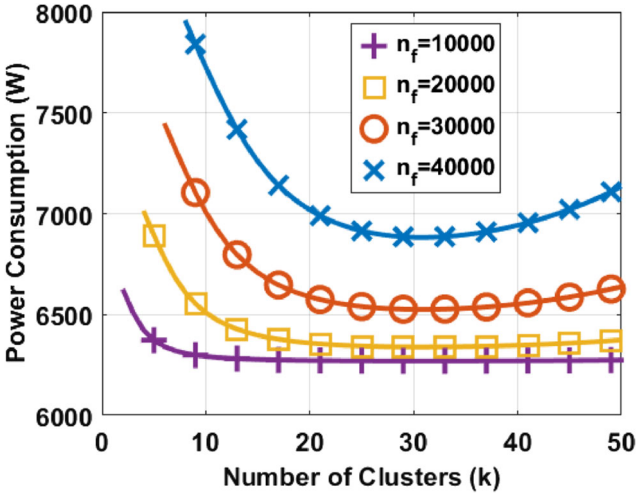

Fig. 3 Power consumption of MEC based on the number of clusters when $n = 400$

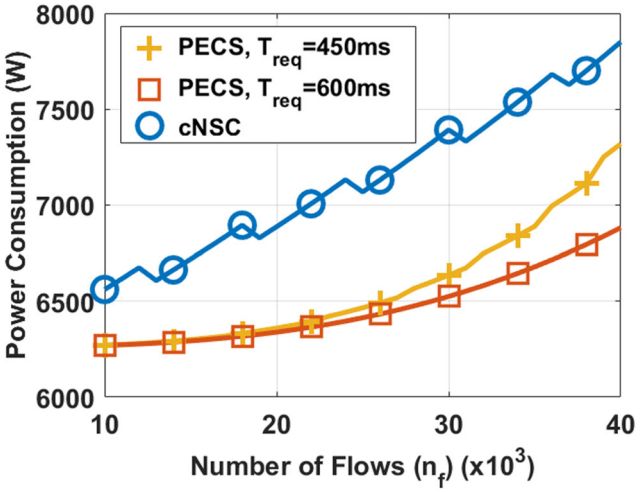

Fig. 5 Power consumption comparison when $n = 600$

model, $P_{mec}(k)$, when $n = 400$, that is, n is less than $(1 + m/2)c$ in Fig. 3. That is because the case that a few servers are busy to handle with concentrated requests whereas many remaining servers are in idle state consumes more electrical power than the case that all servers handle moderately distributed requests through PECS. If k is small, since requests to LC servers and to member servers are constant according to our system model, a few LC servers handle all requests to LC servers and many member servers serve requests equally among them. We can thus see in Fig. 3 that the power consumption is high when k is small, and the consumption decreases as k increases due to the load balancing among MEC servers. After a minimum point, however, the power consumption starts to increase because of the concentration of requests to a few member servers. In Fig. 4, when $n = 600$, that is, n is more than $(1 + m/2)c$, there are uninstalled MVEs in the MEC environment when $k \geq 42$. Accordingly, additional power consumption is generated by dynamic placements of MVEs. The beginning of the graph is removed because k should be included in $\mathcal{K}$. When the number of flows increases, $P_{mec}(k)$ also increases because more requests require more CPU load.

In Figs. 5 and 6, we compare the power consumption and average delay of the cNSC scheme [9] with two cases of PECS serving 5G services with T_{req} of 450ms and 600ms, respectively. When the number of flows increases, $P_{mec}(k)$ also increases. When k that minimizes the average delay get out of $\mathcal{K}$, we find an alternative k that is near the original k value for $k \in \mathcal{K}$. That is why the cNSC graph has a saw-tooth pattern. When T_{req} is 600ms, PECS shows the most power efficient performance. When n_f is 40000, PECS achieves a 12.32% reduction of power consumption

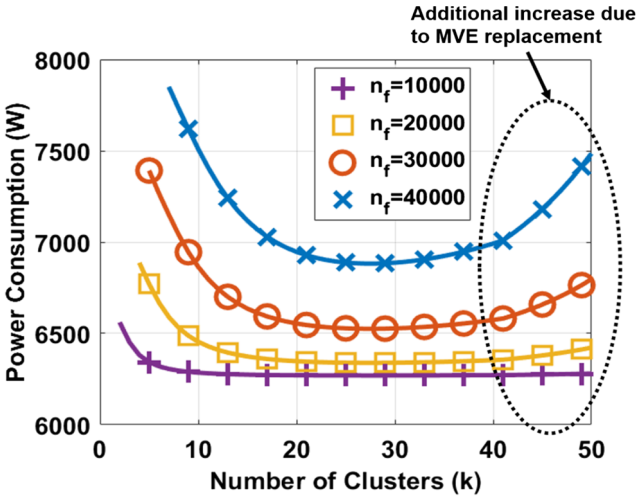

Fig. 4 Power consumption of MEC based on the number of clusters when $n = 600$

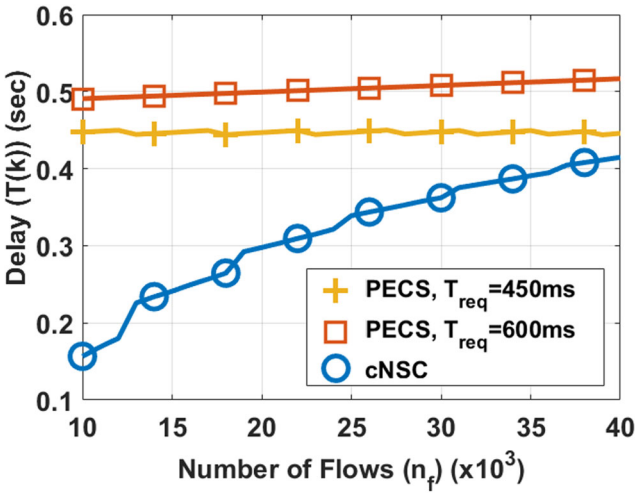

Fig. 6 Average delay comparison when $n = 600$

compared to that of the cNSC model without turning off any MEC servers. Fig. 6 shows that PECS sustains average delay of inflows processed in MEC servers, $T(k)$, at the acceptable level, T_{req}, regardless of the n_f value. When a few flows are in the MEC environment, the average delay of PECS is higher than that of the cNSC scheme because PECS seeks an optimal k to minimize power consumption without any concern for the average delay if it is less than T_{req}. When T_{req} is 450ms, our model sustains average delay of inflows processed in MEC servers at the acceptable level. However, it consumes more power than when T_{req} is 600ms as n_f increases because the available k included in $\mathcal{K}$ becomes small as n_f increases.

6 Conclusion

In this paper, we proposed a power-efficient clustering scheme (PECS) for the MEC environment where all MEC servers are turned on. To do this, we first formulated the power consumption model of MEC servers according to CPU loads by considering the power consumption behavior of the MEC server using CPUs with dynamic voltage frequency scaling. Using the power consumption model of MEC servers, we determined the optimal number of clusters to reduce the power consumption of the overall MEC servers. At the obtained optimal number of clusters in MEC, we found that the CPU load of local cluster servers and member servers is balanced to minimize the power consumption. Finally, we showed that the proposed scheme achieves a 12.32% reduction of power consumption compared with an existing scheme while sustaining average delay of inflows processed in MEC servers at the acceptable level.

Acknowledgements This work was partly supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (2018-0-00691, Development of Autonomous Collaborative Swarm Intelligence Technologies for Disposable IoT Devices) and in part by the Basic Science Research Program of the National Research Foundation of South Korea under Grant NRF-2018R1C1B6001849.

References

1. Mao Y, You C, Zhang J, Huang K, Letaief KB (2017) A survey on mobile edge computing: The communication perspective. IEEE Commun Surveys Tutor 19(4):2322–2358
2. Nguyen N, Duong TQ, Ngo HQ, Hadzi-Velkov Z, Shu L (2016) Secure 5g wireless communications: a joint relay selection and wireless power transfer approach. IEEE Access 4:3349–3359
3. Vo N, Duong TQ, Guizani M, Kortun A (2018) 5G optimized caching and downlink resource sharing for smart cities. IEEE Access 6:31457–31468
4. Vo N, Duong TQ, Tuan HD, Kortun A (2018) Optimal video streaming in dense 5g networks with d2d communications. IEEE Access 6:209–223
5. The Business Case for MEC in Retail: A TCO Analysis and its Implications in the 5G Era
6. Shehabi A, Smith S, Sartor D, Brown R, Herrlin M, Koomey J, Masanet E, Horner N, Azevedo I, Lintner W (2016) United States data center energy usage report
7. Masanet E, Shehabi A, Koomey J (2013) Characteristics of low-carbon data centres. Nat Clim Chang 3:627
8. Mehraghdam S, Keller M, Karl H (2014) Specifying and placing chains of virtual network functions. In: 2014 IEEE 3rd International conference on cloud networking, pp 7–13
9. Nam Y, Song S, Chung J-M (2016) Clustered NFV service chaining optimization in mobile edge clouds. IEEE Commun Lett 21(2):1–1
10. Sciancalepore V, Giust F, Samdanis K, Yousaf Z (2016) A double-tier mec-nfv architecture: design and optimisation. In: 2016 IEEE Conference on standards for communications and networking (CSCN), pp 1–6
11. ETSI. GS MEC 003 - V1.1.1 - Mobile Edge Computing (MEC); framework and reference architecture
12. ETSI. GR MEC 017 - V1.1.1 - Mobile Edge Computing (MEC); Deployment of Mobile Edge Computing in an NFV environment
13. Zhang K, Mao Y, Leng S, Zhao Q, Li L, Peng X, Pan L, Maharjan S, Zhang Y (2016) Energy-efficient offloading for mobile edge computing in 5g heterogeneous networks. IEEE Access 4:5896–5907
14. Mao Y, Zhang J, Letaief KB (2016) Dynamic computation offloading for Mobile-Edge computing with energy harvesting devices. IEEE J Sel Areas Commun 34(12):3590–3605
15. Nguyen LD (2018) Resource allocation for energy efficiency in 5g wireless networks. EAI Endorsed Trans Ind Netw Intell Syst 5(14):6
16. Pouwelse J, Langendoen K, Sips H (2001) Energy priority scheduling for variable voltage processors. In: ISLPED'01, pp 28–33. ACM
17. Wu D, Zeng Y, He J, Liang Y, Wen Y (2012) On p2p mechanisms for vm image distribution in cloud data centers: modeling, analysis and improvement. In: 4Th IEEE international conference on cloud computing technology and science proceedings, pp 50–57
18. Rizk A, Fidler M (2012) Non-asymptotic end-to-end performance bounds for networks with long range dependent fBm cross traffic. Comput Netw 56(1):127–141
19. Kliazovich D, Bouvry P, Granelli F, da Fonseca NLS (2015) Energy consumption optimization in cloud data centers. Cloud services, networking, and management, pp 191–215
20. Hyytia E, Lassila P, Virtamo J (2006) Spatial node distribution of the random waypoint mobility model with applications. IEEE Trans Mob Comput 5(6):680–694
21. Quesnel F, Mehta HK, Menaud JM (2013) Estimating the power consumption of an idle virtual machine. In: 2013 IEEE Greencom

Jaewon Ahn received the B.S., M.S degrees from the Korea Advanced Institute of Science and Technology, Daejeon, South Korea, in 2009, and 2011, respectively. He is now pursuing the Ph.D degree with the School of Electrical Engineering, KAIST. His research interests include wireless mesh networks, Internet of things, software defined network, network function virtualization (NFV) orchestration, multi-access edge computing, and energy efficient technologies. He has contributed several articles to the International Telecommunication Union Tele-communication (ITU-T) study group 13 as an editor.

Joohyung Lee He received the B.S., M.S., and Ph.D. degrees from the Korea Advanced Institute of Science and Technology, Daejeon, South Korea, in 2008, 2010, and 2014, respectively. From 2012 to 2013, he was a Visiting Researcher with the Information Engineering Group, Department of Electronic Engineering, City University of Hong Kong, Hong Kong. From 2014 to 2017, he was a Senior Engineer with Samsung Electronics. He is currently an Assistant Professor with the Department of Software, Gachon University, South Korea. He has contributed several articles to the International Telecommunication Union Telecommunication and 3rd Generation Partnership Project. His current research interests include resource allocation and optimization, with a focus on resource management and system design for future media (e.g., augmented reality and virtual reality), 5G networks, edge/cloud computing, machine learning, internet of things and smart grids (future power grids).

Dr. Lee was an active member of the GreenTouch Consortium. He received the Best Paper Award at the Integrated Communications, Navigation, and Surveillance Conference in 2011 and Award for Outstanding Contribution in Reviewing at Elsevier Computer Communications in 2017. He has been a technical reviewer for several conferences and journals.

Sangdon Park received the B.S., M.S., and Ph.D. degrees from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2011, 2013, and 2017, respectively. He is currently a PostDoctoral Researcher with the Information and Electronics Research Institute, KAIST. He has contributed several articles to the International Telecommunication Union Telecommunication. His research interests include resource allocation and optimization in smart grids, wireless networks, and cloud computing. He received the Best Student Paper Award at the 11th International Conference on Queueing Theory and Network Applications in 2016.

Hong-Shik Park received the B.S. degree from Seoul National University, Seoul, South Korea in 1977, the M.S. and Ph.D. degrees from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea in 1986, 1995, respectively, all in electrical engineering. He joined ETRI in 1977, where he was involved in developing the TDX-1, TDX-1A, TDX-1B, TDX-10, and ATM switching systems. In 1998, he moved to Information and Communications University, Daejeon, South Korea as a faculty. Since 2009, he has been a Professor of KAIST. He is currently a Director of the BcN engineering research center. His research interests include network architecture and protocols, traffic control, and performance analysis of telecommunication systems.