Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 6421607, 15 pages
https://doi.org/10.1155/2018/6421607

Research Article

Dynamic Resource Allocation for Load Balancing in Fog Environment

Xiaolong Xu,[1,2,3,4] **Shucun Fu**,[1,2] **Qing Cai**,[1,2] **Wei Tian**,[1,2] **Wenjie Liu**,[1,2] **Wanchun Dou**,[3] **Xingming Sun**,[1,2] **and Alex X. Liu**[1,4]

[1] *School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing, China*
[2] *Jiangsu Engineering Centre of Network Monitoring, Nanjing University of Information Science and Technology, Nanjing, China*
[3] *State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China*
[4] *Department of Computer Science and Engineering, Michigan State University, East Lansing, MI, USA*

Correspondence should be addressed to Wanchun Dou; douwc@nju.edu.cn

Received 6 December 2017; Accepted 19 March 2018; Published 26 April 2018

Academic Editor: Deepak Puthal

Fog computing is emerging as a powerful and popular computing paradigm to perform IoT (Internet of Things) applications, which is an extension to the cloud computing paradigm to make it possible to execute the IoT applications in the network of edge. The IoT applications could choose fog or cloud computing nodes for responding to the resource requirements, and load balancing is one of the key factors to achieve resource efficiency and avoid bottlenecks, overload, and low load. However, it is still a challenge to realize the load balance for the computing nodes in the fog environment during the execution of IoT applications. In view of this challenge, a dynamic resource allocation method, named DRAM, for load balancing in fog environment is proposed in this paper. Technically, a system framework for fog computing and the load-balance analysis for various types of computing nodes are presented first. Then, a corresponding resource allocation method in the fog environment is designed through static resource allocation and dynamic service migration to achieve the load balance for the fog computing systems. Experimental evaluation and comparison analysis are conducted to validate the efficiency and effectiveness of DRAM.

1. Introduction

In recent years, the Internet of Things (IoT) has attracted attention from both industry and academia, which is beneficial to humans' daily lives. The data extracted from the smart sensors are often transmitted to the cloud data centers and the applications are generally executed by the processors in the data centers [1]. The cloud computing paradigm is efficient in provisioning computation and storage resources for the IoT applications, but the ever-increasing amount of resource requirements of the IoT applications leads to the explosively increased energy consumption and the performance degradation of the computing nodes due to data transmission and computing node migration; thus, how to perform IoT applications becomes an urgent issue [2–6]. Fog computing extends the computing process to the edge of the network rather than performing the IoT applications in the cloud platforms.

In the fog environment, the routers are the potential physical servers which could provision resources for the fog services at the edge of the network [7, 8]. The routers could enhance the performance of computation and storage, which could be fully utilized as the computing nodes. In the big data era, there are different performance requirements for the IoT applications, especially for the real-time applications; thus, such applications choose the edge computing nodes as a priority to host [9, 10]. In the fog environment, the users could access and utilize the computation, storage, and network resources, like the way the customers use the cloud resources, and the virtualized technology is also applicative to provision the on-demand resources dynamically [11]. The IoT applications could be performed by the fog computing nodes and the physical resources deployed in the remote cloud data center. The resource allocation for the IoT applications should take into account both the centralized and the geodistributed computing nodes, and the resource schedulers and managers

should choose the appropriate computing nodes to host the fog services combined in the IoT applications through the design of resource allocation strategies.

Resource allocation and resource scheduling are the key technologies to manage the data centers, which contribute a great deal to lowering the carbon emission, improving the resource utilization, and obtaining load balancing for the data centers [12–14]. In the cloud environment, the main goal of resource allocation is to optimize the number of active physical machines (PMs) and make the workloads of the running PMs distributed in a balanced manner, to avoid bottlenecks and overloaded or low-loaded resource usage [14–17]. In the fog environment, resource allocation becomes more complicated since the applications could be responded to by the computing nodes both in fog and in clouds. The computing nodes in the fog are distributed dispersedly in the network of edge, while the computing nodes in the cloud are distributed in a centralized data center. The resource requirements of the IoT applications for the computing nodes are various as the applications have different demands of computing power, storage capacity, and bandwidth. Therefore, it is necessary to undergo resource allocation for the dynamic resource requirements of the IoT applications, to achieve the goal of load balancing.

With the above observations, it is still a challenge to realize the load balance for the computing nodes in the fog environment during the execution of IoT applications. In view of this challenge, a dynamic resource allocation method, named DRAM, for load balancing in fog environment is proposed in this paper. Specifically, our main contributions are threefold. Firstly, we present a system framework for IoT applications in fog environment and conduct the load-balance analysis for various types of computing nodes. Then, a corresponding resource allocation method in the fog environment is designed through static resource allocation and dynamic service migration to achieve the load balance for the fog computing systems. Finally, adequate experimental analysis is conducted to verify the performance of our proposed method.

The rest of this paper is organized as follows. In Section 2, formalized concepts and definitions are presented for load-balance analysis in the fog environment. Section 3 elaborates the proposed resource allocation method DRAM. Section 4 illustrates the comparison analysis and performance evaluation. Section 5 summarizes the related work, and Section 6 concludes the paper and presents the prospect for the future work.

2. Preliminary Knowledge

In this section, a fog computing framework for IoT applications is designed and the load-balance analysis is conducted as well.

To facilitate dynamic resource allocation for load balancing in fog environment, formal concepts and load-balance analysis are presented in this section. Key notations and descriptions used in this section are listed in the section named "Key Terms and Descriptions Involved in Resource Scheduling in Fog Environment."

2.1. System Framework for Fog Computing. Fog computing is a new computing paradigm, which sufficiently leverages the decentralized resources through the fog and cloud environments, to provision the computation and storage services for the IoT applications. Fog computing extends the data process and data storage between the smart sensors and the cloud data centers. Some of the tasks from the IoT applications could be processed in the fog rather than performing all the tasks in the cloud environment. The virtualized technology could be employed to improve the resource usage in the fog environment.

Figure 1 shows a hierarchical framework for computing tasks from IoT applications in the fog environment. There are four layers in this framework, that is, the IoT application layer, the service layer, the fog layer, and the cloud layer. The IoT application contains a large amount of service requirements that need to be responded to by selecting appropriate computing nodes according to the time urgency and the resource amount from fog and cloud. The fog layer consists of the edge computing nodes and the intermediate computing nodes. The services with the highest time urgency and less computation density can be calculated by the edge computing nodes in the fog layer, and the less urgent tasks could choose intermediate computing nodes for execution. The cloud layer is appropriate to hosting the loose-fitting tasks with high-density computation and huge-volume storage which often demand a large amount of physical resources. The intermediate computing nodes could be the routers for data transmission, the edge computing nodes could be the mobile devices or sensors, and the computing nodes in the cloud are the PMs.

Fog computing is useful for IoT applications which combine many fog services. The fog services cover all the procedures of the data extraction, data transmission, data storage, and service execution from the IoT applications.

Definition 1 (fog service). The services requested by the IoT applications are available to be performed in the fog and remote cloud data centers, denoted as $S = \{s_1, s_2, \ldots, s_N\}$, where N is the number of fog services generated by the IoT applications.

The PMs in the remote cloud data centers, the intermediate computing nodes, and the edge computing nodes in the fog environment are all available to be leveraged to provision physical resources for the fog services. Suppose there are M computing nodes in the fog and cloud environment, denoted as $P = \{p_1, p_2, \ldots, p_M\}$. The virtualized technology has been widely applied in the cloud environment, which is also adaptive in the fog environment to measure the resource capacity of all the computing nodes and the resource requirement of the fog services.

Definition 2 (resource capacity of computing nodes). For all the computing nodes in the fog and cloud, their resource capacities are quantified as the number of resource units, and each resource unit contains various physical resources, including CPU, memory, and bandwidth.

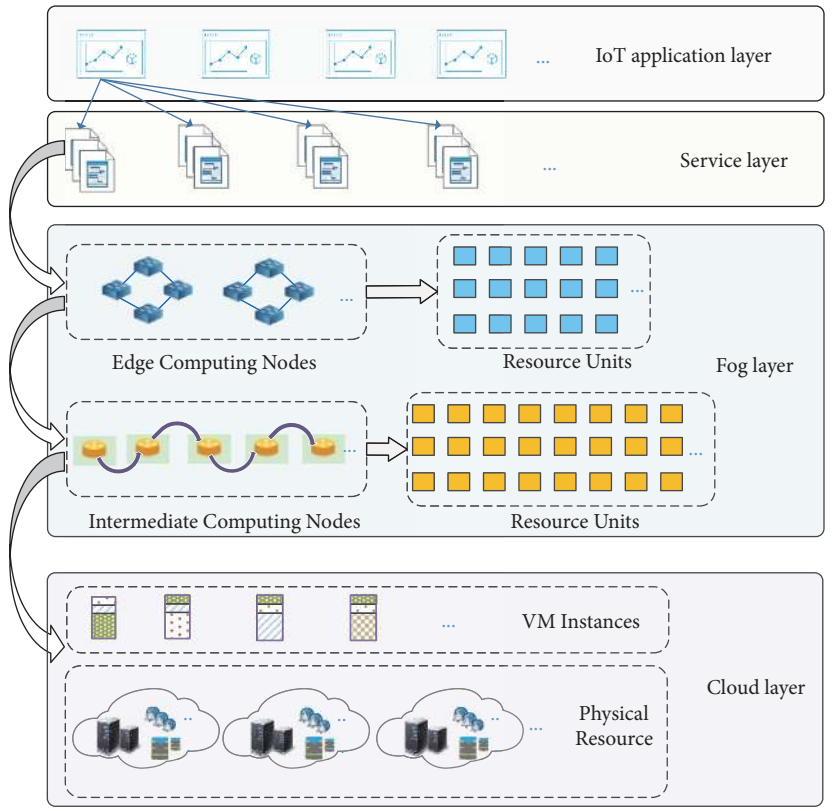

Figure 1: Fog computing framework for IoT applications.

Definition 3 (resource requirement of s_n). The corresponding resource requirement of the nth fog service s_n could be quantified as the time requirements, the resource type, and the resource amount to perform s_n, denoted as $r_n = \{stim_n, dutim_n, type_n, cou_n\}$, where $stim_n$, $dutim_n$, $type_n$, and cou_n represent the request start time, the duration time, the resource type, and the requested amount of s_n, respectively.

Note that the resource requirements in this paper are measured by the amount of resource units. For example, in the cloud data centers, the resource units are the VM instances, and the customers often rent several VM instances to host one application.

2.2. Load Balancing Model in Fog Environment. Fog computing paradigm releases the load distribution in the cloud environment. Due to the diversity of execution duration and specifications for the computing nodes in the fog, the resources could not be fully utilized. In the fog environment, we try to achieve load balancing for the computing nodes to avoid low utilization or overload of the computing nodes.

Let $I_n^m(t)$ be the binary variable to judge whether s_n $(1 \le n \le N)$ is assigned to the computing node p_m $(1 \le m \le M)$ at time instant t, which is calculated by

$$I_n^m(t) = \begin{cases} 1, & \text{if } s_n \text{ is assigned to } p_m, \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

With the judgement of fog service distribution, the resource utilization for the mth computing node p_m at time t is calculated by

$$ru_m(t) = \frac{1}{c_m} \sum_{n=1}^{N} I_n^m(t) \cdot r_n, \quad (2)$$

where c_m is the capacity of the computing node p_m.

According to the service distribution, the number of services deployed on p_m at time instant t is calculated by

$$ns_m(t) = \sum_{n=1}^{N} I_n^m(t). \quad (3)$$

The load-balance variance should be specified for each type of computing node. Suppose there are W types of computing nodes in cloud and fog environment for performing the fog services.

The load-balance variance is closely relevant to the resource utilization. To calculate the resource utilization, the employed amount of computing nodes with type w at the time instant t is calculated by

$$a_w(t) = \sum_{m=1}^{M} f_m(t) \cdot \beta_w^m, \quad (4)$$

where β_w^m is a flag to judge whether p_m is a type w of computing nodes, which is described in (5), and $f_m(t)$ is used to judge whether p_m is empty at t, presented in (6).

$$\beta_w^m = \begin{cases} 1, & \text{if } p_m \text{ is a } w\text{th type computing node,} \\ 0, & \text{otherwise.} \end{cases} \quad (5)$$

$$f_m(t) = \begin{cases} 1, & \text{if } ns_m > 0, \\ 0, & \text{otherwise.} \end{cases} \quad (6)$$

The resource utilization for the computing nodes with wth type at time t is calculated by

$$RU_w(t) = \frac{1}{a_w(t)} \sum_{m=1}^{M} \sum_{n=1}^{N} I_n^m(t) \cdot ru_m(t) \cdot f_m(t). \quad (7)$$

Definition 4 (variance of load balance of p_m at time instant t). The load-balance value is measured by the variance of the resource utilization. The variance value for p_m is calculated by

$$lb_m(t) = \left(ru_m(t) - \sum_{w=1}^{W} RU_w(t) \cdot \beta_w^m \right)^2. \quad (8)$$

Then, the average variance value for all the type w computing nodes is calculated by

$$LB_w(t) = \frac{1}{a_w(t)} \sum_{m=1}^{M} \sum_{n=1}^{N} I_n^m(t) \cdot lb_m(t) \cdot f_m(t). \quad (9)$$

For the execution period $[T_0, T]$ in the fog and cloud environment, the load-balance variance could be calculated by

$$LB_w = \frac{1}{T - T_0} \int_{T_0}^{T} LB_w(t)\, dt. \quad (10)$$

With these observations, the problem of minimizing the variance of load balance can be formulated as follows:

$$\min \quad LB_w, \quad \forall w = 1, \ldots, W \quad (11)$$

$$\text{s.t.} \quad a_w(t) \le \sum_{m=1}^{M} \beta_w^m, \quad (12)$$

$$\sum_{n=1}^{N} r_n \le \sum_{m=1}^{M} c_m, \quad (13)$$

$$\sum_{n=1}^{N} I_n^m r_n \le c_m, \quad (14)$$

where $\sum_{n=1}^{N} r_n$ in formula (13) represents the resource requirements for all services, $\sum_{m=1}^{M} c_m$ in formula (14) represents the capacity of all computing nodes, and $\sum_{n=1}^{N} I_n^m r_n$ in formula (14) represents the resource requirements for all services allocated to the type m computing node.

From (11) to (14), we can find that the final objective solved in this paper is an optimization problem with multiple constraints [18–20].

3. A Dynamic Resource Allocation Method for Load Balancing in Fog Environment

In this section, we propose a dynamic resource allocation method, named DRAM, for load balancing in fog environment. Our method aims to achieve high load balancing for all the types of computing nodes in the fog and the cloud platforms.

3.1. Method Overview. Our method consists of four main steps, that is, fog service partition, spare space detection for computing nodes, static resource allocation for fog service subset, and the load-balance driven global resource allocation, as shown in the section named "Specification of our Proposed Resource Allocation Method for Load Balancing in Fog Environment." In this method, Step 1 is the preprocess procedure, Step 2 is employed to detect the resource usage for Steps 3 and 4, and Step 3 is designed for static resource allocation for the fog services in the same subset and it provides the primary resource provision strategies for Step 4. Finally, Step 4 is a global resource allocation method to realize dynamic load balance.

Specification of Our Proposed Resource Allocation Method for Load Balancing in Fog Environment

Step 1 (fog service partition). There are different types of computing nodes for the performance of fog services. To efficiently provision resources, the fog services are classified as several sets based on the resource requirements of node type. Furthermore, these sets are divided into multiple subsets according to the request start time.

Step 2 (spare space detection for computing nodes). To judge whether a computing node is portable to host the fog service, it is necessary to detect the spare space of all the computing nodes. We analyze the employed resource units through the analysis of occupation records, and then the spare space of the computing nodes could be obtained.

Step 3 (static resource allocation for fog service subset). For the fog services in the same service subset, the proper computing nodes are identified to host these services. When allocating resource units for a fog service, the computing node with the least and enough spare space is selected. Besides, some workloads from the computing nodes with higher resource usage are migrated to the computing nodes with low resource usage.

Step 4 (load-balance driven global resource allocation). For all the fog service subsets, we could find the initialized resource allocation strategies in Step 4, and then the dynamic resource allocation adjustment is conducted at the competition moments of the fog services to achieve the global load balance during the execution period of the fog services.

3.2. Fog Service Partition. The fog services from different IoT applications have different requirements of computing resources; that is, the fog services need to choose different

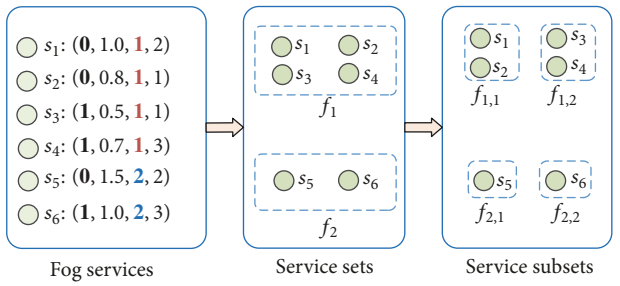

Figure 2: An example of subset acquisition with 6 fog services (i.e., s_1~s_6).

types of computing nodes for resource response. Suppose there are W types of processing nodes, including the PMs in cloud platforms, the intermediate nodes, and the edge nodes near the sensors.

In this paper, we try to achieve the goal of load balancing for each type of computing node; we assume that the fog services need to be performed with the same type of computing nodes. As a result, the fog services could be partitioned as W different service sets, denoted as $\mathbf{F} = \{\mathbf{f_1}, \mathbf{f_2}, \ldots, \mathbf{f_W}\}$.

The resource requirements of the fog services in the same set have different resource response time. To efficiently realize resource allocation for the services, the fog services in the same set should be partitioned to several subsets according to the start time for occupying the resource units of the computing nodes. Then, we can allocate resource units for the fog services in the same set to achieve high load balancing.

The subset $\mathbf{f_w}$ $(w = 1, 2, \ldots, W)$ in $\mathbf{F}$ is divided into multiple subsets according to the requested start time of the fog services. Let $\mathbf{f_{w,i}}$ be the ith $(1 \le i \le |\mathbf{f_w}|)$ subset contained in $\mathbf{f_w}$. After the partition process, the fog services in $f_{w,i}$ have the same request start time.

For example, there are 6 fog services, that is, s_1~s_6, and the resource requirements of these 6 services are s_1: (0, 1, **1**, 2), s_2: (0, 0.8, **1**, 1), s_3: (1, 0.5, **1**, 1), s_4: (1, 0.7, **1**, 3), s_5: (0, 1.5, **2**, 2), and s_6: (1, 1, **2**, 3), as shown in Figure 2. These 6 fog services are put in two different sets f_1 and f_2 according to the requested type of computing nodes, $f_1 = \{s_1, s_2, s_3, s_4\}$ and $f_2 = \{s_5, s_6\}$. Then, f_1 and f_2 are partitioned to the subsets according to the requested start time. Partition $f1$ is divided into 2 subsets, $f_{1,1} = \{s_1, s_2\}$ and $f_{1,2} = \{s_3, s_4\}$; meanwhile, f_2 is also separated as 2 subsets, that is, $f_{2,1} = \{s_5\}$ and $f_{2,2} = \{s_6\}$.

Algorithm 1 specifies the process of fog services subset acquisition. In Algorithm 1, the input is the resource requirements of IoT applications, that is, R, and the output is the partitioned fog service set $\mathbf{F}$. We traverse all the fog services (Line (1)), and the services are put in different sets according to the requested resource type (Lines (1) to (6)), and then the fog services are put in W different service sets. Then, the set $\mathbf{f_w}$ is divided to multiple subsets, according to the required start time (Lines (8) to (17)).

3.3. Spare Space Detection for Computing Nodes. The fog services need to be put on the computing nodes; thus, those computing nodes with spare space should be identified. For all the computing nodes, the allocation records are employed

Input: The resource requirements of IoT applications R
Output: The partitioned fog service set $\mathbf{F}$
(1) for $i = 1$ to N **do**
(2) **for** $j = 1$ to W **do**
//There are W types of computing nodes in fog and cloud
(3) **if** $type_i == j$ **then**
(4) Add r_i to f_i
(5) **end if**
(6) **end for**
(7) end for
(8) for $i = 1$ to W **do**
(9) $nn = 0, q = 0, f = stim_0$
(10) **while** $nn < |fs_i|$ **do**
(11) **if** $stim_q \le f$ **then**
(12) Add the mth fog service to $fs_{i,nn}$
(13) **else** $nn = nn + 1, q = q + 1, f = stim_q$
(14) Add the qth fog service to $fs_{i,nn}$
(15) **end if**
(16) **end while**
(17) end for
(18) Return F

Algorithm 1: Fog service subset acquisition.

to monitor the resource usage of all the computing nodes in the fog and cloud.

Definition 5 (occupation record $rs_{m,i}$). The ith $(1 \le i \le |rs_m|)$ occupation record in rs_m contains the response fog service, the occupation start time, the duration time, and the resource units, which is a 4-tuple, denoted as $rs_{m,i} = (fs_{m,i}, st_{m,i}, dt_{m,i}, us_{m,i})$, where $fs_{m,i}$, $st_{m,i}$, $dt_{m,i}$, and $us_{m,i}$ are the fog service, the start time, the duration time, and the resource unit sets of $rs_{m,i}$, respectively.

A computing node has a set of occupation records, since it could host several fog services. The record set for the computing node p_m $(1 \le m \le M)$ is recorded as rs_m. Then, for all the computing nodes in the fog and cloud, there are M occupation record sets, denoted as $RS = \{rs_1, rs_2, \ldots, rs_M\}$. In rs_m, there are many occupation records, which reflect the resource usage of the computing node p_m.

The occupation records are dynamically updated according to the real-time resource provisioning for the fog services. Once the fog services are moved to the other computing nodes during their lifetime, the occupation time parameter should be updated accordingly for the relevant records. The occupation records are generated from the users when they apply for resources in the fog and cloud data center for implementing the services generated from the IoT applications.

Benefiting from the resource monitoring, the occupied resource units of all the computing nodes and the spare units could be detected for resource allocation at any time.

Definition 6 (spare space of p_m). The spare space of the computing node p_m is defined as the idle amount of resource units, which is measured by the difference value of the capacity and the occupied amount of resource units on p_m.

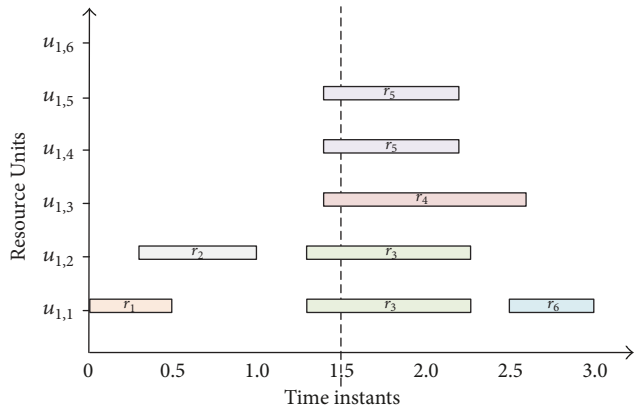

Figure 3: An example of spare space detection with 6 occupation records (i.e., $r_1 \sim r_6$).

Input: The occupation record for the computing node p_m
Output: The spare space for p_m
(1) $cou = 0$
(2) for $i = 1$ to $|rs_m|$ **do**
(3) $ct_{m,i} = st_{m,i} + dt_{m,i}$
//$ct_{m,i}$ is the finish time for the occupation of resource units
(4) **if** $t \geq st_{m,i}$ && $t < ct_{m,i}$ **then**
//t is the request time for statistics
(5) $cou = cou + |us_{m,i}|$
(6) **end if**
(7) end for
(8) $cou = c_m - cou$
(9) Return cou

Algorithm 2: Spare space detection for computing nodes.

The spare space of the computing node p_m could be detected from the analysis of occupation records. In these records, if occupation start time is less than the statistic time instant for checking the PM status and the occupation finish time is over the statistic time, the relevant resource units combined in the occupation records could be obtained. With these acquired resource units and the resource capacity of p_m, the spare space could be finally detected.

For example, there are 6 occupation records for the computing node p_1, that is, $r_1 : (1, \mathbf{0}, \mathbf{0.5}, \{u_{1,1}\})$, $r_2 : (2, \mathbf{0.3}, \mathbf{0.7}, \{u_{1,2}\})$, $r_3 : (3, \mathbf{1.3}, \mathbf{1}, \{u_{1,1}, u_{1,2}\})$, $r_4 : (4, \mathbf{1.4}, \mathbf{1.2}, \{u_{1,3}\})$, $r_5 : (5, \mathbf{1.4}, \mathbf{0.8}, \{u_{1,4}, u_{1,5}\})$, and $r_6 : (6, \mathbf{2.5}, \mathbf{0.5}, \{u_{1,1}\})$, as shown in Figure 3. If the statistic instant for spare space detection is 1.5, the identified occupation records within the statistic time are r_3, r_4, and r_5. Based on the analysis of the occupation records, the employed resource units at this moment are $u_{1,1}, u_{1,2}, u_{1,3}, u_{1,4}$, and $u_{1,5}$. Suppose the capacity of p_1 is 6; then, the spare space of p_1 at time instant 1.5 is 1.

Algorithm 2 specifies the key idea of spare space detection for computing nodes. In Algorithm 2, the input and the output are the occupation records of the computing node p_m and the spare space for p_m. According to Definition 7, we need to calculate the occupation amount of resource units on p_m first (Lines (1) to (8)), and then the spare space could be detected (Line (8)).

3.4. Static Resource Allocation for Fog Service Subset. Based on the fog service partition in Section 3.1 and spare space detection for computing nodes in Section 3.2, the fog services that need to be processed and the available computing resources for fog service execution are identified, which are all beneficial for resource allocation.

In the fog environment, the fog services need to be responded to by the computing nodes, and the time requirements also should be presented when allocating the resources to the fog services. In this section, we define the resource allocation records to reserve the allocation history about resource provisioning for the fog services.

Definition 7 (resource allocation record for s_n). The resource allocation record for s_n consists of the node type, the number of resource units, the start time, and the desired duration time, which is denoted as $a_n = (nt_n, num_n, st_n, dt_n)$, where nt_n, num_n, st_n, and dt_n are the node type, the amount of resource units, the service start time, and the resource occupation time for s_n, respectively.

The fog services in the same fog service subset have the same required node type and start time. When allocating resource units for the fog service subset, each fog service should find a portable computing node to host it. Thus, we need to find the available nodes first. The computing nodes with the requested type, which have spare space, which could be detected by Algorithm 2, are chosen as the candidate resources to be provided for the fog services in the subset.

To achieve the load balancing of the computing nodes, we try to achieve high resource usage of the employed computing nodes. The problem of static resource allocation is like bin packing problem, which is NP-hard. Here, we leverage the idea of Best Fit Decreasing to realize the process of computing node matching for the fog services. Before resource allocation, the fog services are sorted in the decreasing order of requested resource amount. The service with more required resource units will be processed first. And it chooses the computing node with the least and enough spare space for hosting the service.

For example, there are 3 computing nodes P_1, P_2, and P_3, and the spare spaces of these 3 computing nodes are 4, 6, and 9, respectively, as shown in Figure 4. There are two fog services in the same subset, that is, S_1 and S_2, and the requested resource amounts of these two services are 2 and 6. When conducting resource allocation for S_1 and S_2, S_2 has more resource requirements, and thus S_2 is processed first. After resource allocation, S_2 chose P_3 for hosting and S_1 chose P_1 for hosting.

The above allocation may lead to the unbalanced distribution of the workloads of some computing nodes. In this section, a threshold ρ is employed to judge whether the computing node is in low resource utilization. If a computing node is in low resource utilization and there are no other computing nodes that could host the workloads in this computing node, we choose to move some workloads to this computing node to improve the load balance.

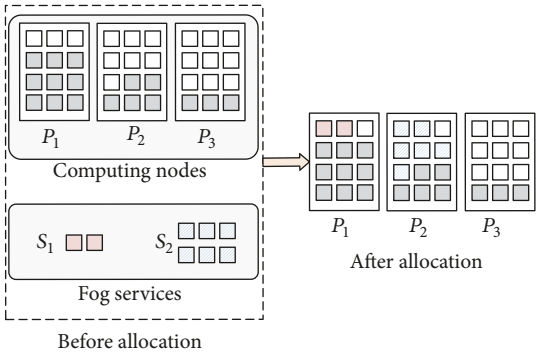

Figure 4: An example of static resource allocation for fog services S_1 and S_2, with computing nodes P_1, P_2, and P_3.

After resource allocation, there are several allocation records generated, to record the allocation history for the fog services in the service subset. Meanwhile, there are some computing nodes, providing resource units for executing the fog services, which also generate some occupation records.

Algorithm 3 illustrates the key idea of static resource allocation for fog service subset $\mathbf{f_{w,i}}$. The input of Algorithm 3 is the fog service subset $\mathbf{f_{w,i}}$, and the output of this algorithm is the resource allocation records for the fog services in $\mathbf{f_{w,i}}$. The required computing nodes of fog services in the same subset have the same node type, which should be obtained first (Line (1)). The services with fewer requested resource units will be processed first, so $\mathbf{f_{w,i}}$ should be sorted in the increasing order of the amount of values of required resources. Then, each service could be responded to with computing nodes sequentially (Line (3)). When selecting a computing node to provision resources for a fog service, the available computing nodes with enough spare space to host the service, calculated by Algorithm 2, should be achieved first (Lines (4) to (11)). The computing nodes also should be sorted in the increasing order of spare space (Line (12)). Then, the computing node with the least spare space will be selected to accommodate the service (Line (13)). Some workloads from the computing nodes with higher resource usage will be migrated to the computing nodes with low resource utilization to improve load balance (Lines (15) to (24)). Finally, the relevant occupation records and the resource allocation record for the fog service should be updated (Lines (25) and (26)).

3.5. Load-Balance Driven Global Resource Allocation. From the analysis in Sections 3.2 and 3.4, the initialized resource allocation is conducted, and the different types of computing nodes could achieve high resource utilization and load balancing at the allocation moment. However, the static resource allocation only could achieve the temporary load balancing at the service arrival moments. During the service running, the resource usage of the computing nodes is dynamically changed due to the various lifetimes of the fog services. In this section, a global resource allocation strategy is designed to make load balancing come true during the execution of the fog services.

Input: The fog service subset $\mathbf{f_{w,i}}$
Output: The relevant resource allocation records
(1) Get the node type nt in the service subset $\mathbf{f_{w,i}}$
(2) Sort $\mathbf{f_{w,i}}$ in decreasing order of required resource amount
(3) for each fog service in $\mathbf{f_{w,i}}$ **do**
(4) **for** $i = 1$ to M **do**
(5) **if** p_m has the same type with nt **then**
(6) Get the spare space by **Algorithm 2**
(7) **if** p_m has enough space to host the service **then**
(8) Add the computing node to CL
(9) **end if**
(10) **end if**
(11) **end for**
(12) Sort CL in increasing order of spare space
(13) Put the service in the first computing node in CL
(14) end for
(15) flag = 1, $i = 1$
(16) Identify the occupied computing nodes from CL to CL'
(17) Sort CL' in the decreasing order of spare space
(18) while flag == 1 **do**
(19) **if** the resource usage of cl'_i is less than ρ **then**
(20) Select the tasks to migrate to the computing node
(21) $i = i + 1$
(22) **else** flag = 0
(23) **end if**
(24) end while
(25) Update the relevant occupation records
(26) Generate an allocation records

Algorithm 3: Static resource allocation for fog service subset.

The fog service subsets demand the same type of computing nodes for hosting sequentially according to the requested start time of the resources. Let $bt_{w,i}$ be the requested start time for resource occupation of the ith subset $f_{w,i}$ in $\mathbf{f_w}$. For dynamic adjustment of resource allocation at the finish time of the fog services in $bt_{w,i}$ ($1 \leq i < |\mathbf{f_w}|$), the scheduling time should be with the time period $(bt_{w,i}, bt_{w,i+1})$. When $i = |\mathbf{f_w}|$, the scheduling time should be the competition time for the rest of the fog service loads during the period for the execution of the fog services in $\mathbf{f_{w,i}}$.

At the scheduling time, the running workloads occupy some computing nodes, and these computing nodes may be with low resource usage due to the service competition. The computing nodes with the same type in fog or cloud could be sorted in the decreasing order of the resource usage. The workloads in the computing nodes with the lower resource usage could be migrated to the computing nodes with higher resource usage, to achieve higher resource utilization. Besides, the migration of workloads could also help to realize the goal of load balancing as the computing nodes with more spare space could be moved vacant and shut down further.

For the workloads from different fog services, it is necessary to find the destination computing nodes to host them. The selection of destination computing nodes decides on the resource requirements of the workloads and the spare space of the computing nodes. If all the workloads from the

Input: The fog service set S
Output: The resource allocation records
The occupation records on computing nodes
(1) Obtain fog service subset **F** by **Algorithm 1**
(2) for $i = 1$ to W **do**
(3) **for** $j = 1$ to $|f_w|$ **do**
(4) **Algorithm 3** Static resource allocation for $\mathbf{f_{w,j}}$
(5) Calculate CT
// CT is the competition time list
// CT = $\{ct_1, ct_2, \ldots, ct_K\}$
(6) **for** $k = 1$ to K **do**
(7) Update the current run list in $\mathbf{f_{w,j}}$
(8) **for** $l = 1$ to M **do**
(9) Get spare space by **Algorithm 2** at ct_k
(10) **if** p_l has spare space and is not empty **then**
(11) Add p_l to SL
(12) **end if**
(13) **end for**
(14) Sort SL in increasing order of spare space
(15) flag = 1, $q = 1$
(16) **while** flag == 1 **do**
(17) Get the occupied resources sets on sl_q
(18) **for** each occupied resource set **do**
(19) Confirm the destination PM
(20) **end for**
(21) **if** the resource sets can be moved **then**
(22) $q = q + 1$
(23) Update the relevant allocation records
(24) Update the occupation records
(25) **else** flag = 0
(26) **end if**
(27) **end while**
(28) **end for**
(29) **end for**
(30) end for

Algorithm 4: Load-balance driven global resource allocation.

same computing node could find the destination computing nodes, these workloads could be migrated to the destination computing nodes. Finally, the resource allocation records and the occupation records are generated or updated according to the real occupation computing nodes and the usage time of the corresponding resource units.

Algorithm 4 illustrates the key process of load-balance driven global resource allocation. The key idea of Algorithm 4 is to conduct static resource allocation for the fog services at the start execution time and dynamically adjust the service placement according to the resource usage of all the employed computing nodes. The input of this algorithm is the fog service set S, and the final output of this algorithm is the resource allocation records and the occupation records. In this algorithm, the fog service subset is achieved first by Algorithm 1 (Line (1)), and then we traverse each subset for resource allocation (Line (2)) and conduct static resource allocation for the subsets by Algorithm 3 (Line (3)). Then, for each subset, the competition time list CL is extracted for load-balance driven dynamic resource allocation (Line (5)). Then, at each competition instant, we adjust the resource allocation records for the running services (Line (6)). The fog services on the computing node with less spare space would be moved to the computing node with higher resource usage, which has enough spare space to host the services (Lines (8) to (20)). When all the fog services on a computing node could find the destination node, the relevant allocation records and the occupation records for the resource units will be generated and updated (Lines (15) to (27)).

Table 1: Parameter settings.

Parameter	Domain
Number of fog services	{500, 1000, 1500, 2000}
Number of computing nodes	3000
Number of node types	3
Resource capacity	{7, 12, 18}
Resource requirements of fog services	[1, 15]
Duration for each service	[0.1, 4.8]

4. Experimental Analysis

In this section, the cloud simulator Cloudsim is applied to evaluate our proposed method DRAM. The intermediate computing nodes and the edge computing nodes are simulated as two computing data centers. The resource allocation method for fog environment is NP-hard, like the bin packing problem; thus, the typical and efficient resource allocation methods FF, BF, FFD, and BFD are employed for comparison analysis.

4.1. Experimental Context. To discuss the effectiveness of DRAM, 4 datasets with different scale of fog services are utilized, which are shared at https://drive.google.com/drive/folders/0B0T819XffFKrZTV4MFdzSjg0dDA?usp=sharing. The parameters for experimental evaluation are presented in Table 1.

The fog services employ three types of computing nodes, that is, the edge computing node, the intermediate node, and the PMs in cloud for resource response. The number of services for each type of computing node contained in the 4 different datasets is shown in Figure 5. For example, when the number of fog services is 1000, there are 324 fog services that need edge computing nodes, 353 fog services that need intermediate computing nodes, and 323 fog services that need PMs in the remote cloud for resource response.

4.2. Performance Evaluation. Our proposed method tends to minimize the load-balance variance, which is relevant to the resource utilization of each computing node and the average resource utilization. Therefore, we conduct performance evaluation for this fog computing system on the employed number of computing nodes, resource utilization, and load-balance variance.

(1) Performance Evaluation on the Employed Number of Computing Nodes. The number of the computing nodes could reflect the efficiency of resource usage. Figure 6 shows the comparison of the number of employed computing nodes

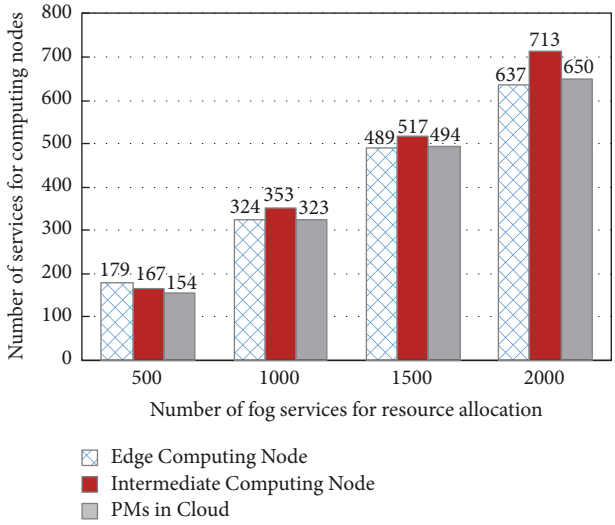

Figure 5: Number of fog services for the 3 types of computing nodes with different datasets.

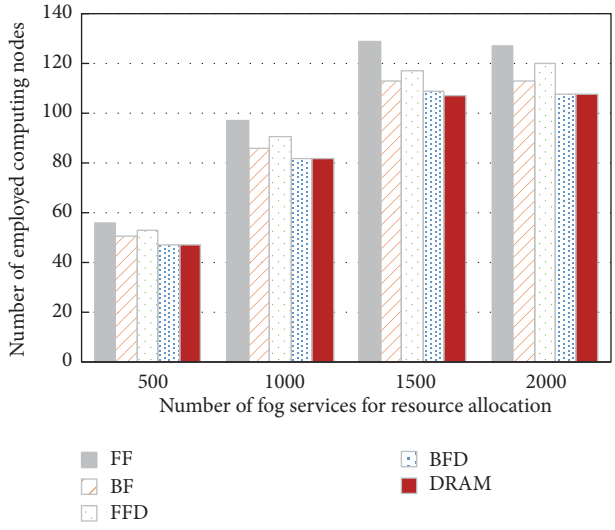

Figure 6: Comparison of the number of employed computing nodes by FF, BF, FFD, BFD, and DRAM with different datasets.

by FF, BF, FFD, BFD, and DRAM by using the 4 different scales of datasets. From Figure 6, we can find that our proposed method DRAM as well as BFD could employ fewer computing nodes, compared with FF, BF, and FFD, when the number of fog services is 500, 1000, and 2000. When the number of fog services is 1500, our proposed DRAM could even get higher efficiency on the employed number of computing nodes than BFD. In the process of static resource allocation, DRAM leverages the basic idea of BFD; thus, in most cases, DRAM and BFD have similar performance on the amount of employed computing nodes. But when some of the computing nodes are spared through the process of DRAM, thus in some cases, DRAM is superior to BFD.

As there are 3 types of computing nodes in our experimental evaluations, we should evaluate DRAM for the different types of computing nodes, compared to the other four methods. The 4 subfigures in Figure 7 show the comparison of the number of the employed computing nodes with different types by FF, BF, FFD, BFD, and DRAM using different datasets. It is intuitive from Figure 7 that our proposed method DRAM is fit for all kinds of computing nodes, which employs fewer computing nodes than FF, BF, and FFD, and gets similar performance to BFD in most cases.

(2) Performance Evaluation on Resource Utilization. The resource utilization is a key factor to decide the load-balance variance; thus we evaluate this value to discuss the resource usage achieved by FF, BF, FFD, BFD, and DRAM with different datasets. The resource utilization is referenced to the resource usage of the resource units on the computing nodes. Figure 8 shows the comparison of average resource utilization by FF, BF, FFD, BFD, and DRAM with different datasets. It is intuitive from Figure 8 that DRAM could obtain better resource utilization than FF, BF, FFD, and BFD, since DRAM is a dynamic and adaptive method which could adjust the load distribution during the fog service execution.

Similar to the evaluation on the employed amount of computing nodes, the performance evaluation on resource utilization is conducted from the perspective of the different types of the computing nodes. Figure 9 shows the comparison of resource utilization for different types of computing nodes by FF, BF, FFD, BFD, and DRAM with different datasets. From Figure 9, we can find that DRAM could achieve higher resource utilization than FF, BF, FFD, and BFD. For example, in Figure 9(c), when the number of fog services is 1500, DRAM obtains the resource utilization over 80%, whereas FF, BF, FFD, and BFD obtain near or below 70% resource utilization for each type of computing node.

(3) Performance Evaluation on Load-Balance Variance. The evaluation of the load-balance variance is also conducted by FF, BF, FFD, BFD, and DRAM using 4 different scales of datasets. Figure 10 shows the comparison of average load-balance variance, where we can find that our proposed method DRAM is superior to the other methods, that is, FF, BF, FFD, and BFD. For example, when the number of fog services is 500, the load-balance variance obtained by DRAM is near 2.5×10^{-2}, whereas FF, BF, FFD, and BFD obtain the load-balance value over 3×10^{-2}.

The evaluation on the load-balance variance also should take into consideration the computing node type. Figure 11 shows the comparison of load-balance variance values for different types of computing nodes by FF, BF, FFD, BFD, and DRAM with different datasets. From Figure 11, we can find that when changing the scale of datasets, our method can keep the priority on the load-balance variance for each type of computing node.

5. Related Work

The IoT technology has been widely applied in many fields, including weather forecasting and traffic monitoring. The

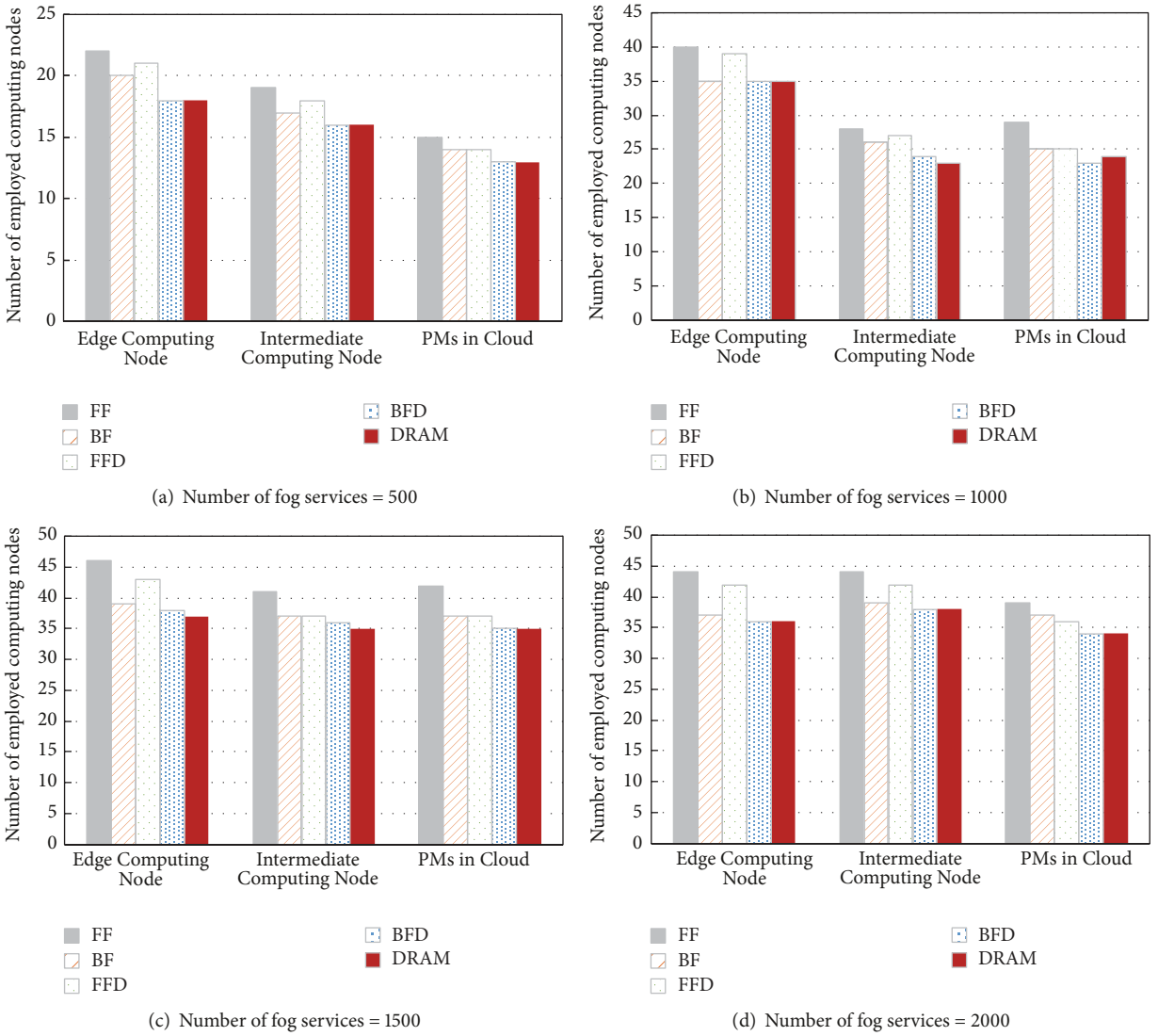

FIGURE 7: Comparison of the number of the employed computing nodes with different types by FF, BF, FFD, BFD, and DRAM using different datasets.

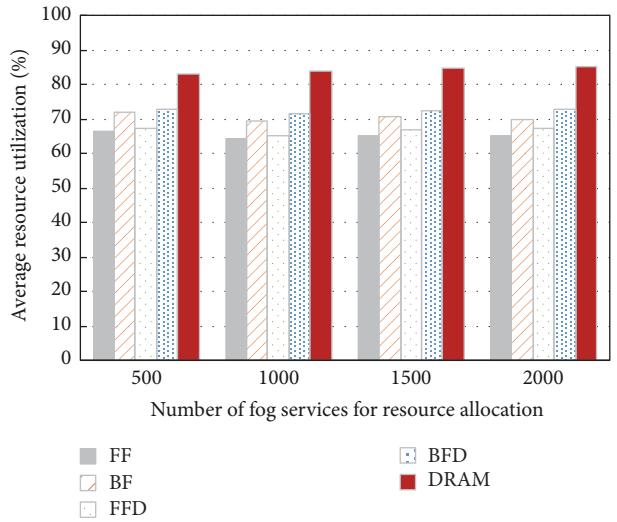

FIGURE 8: Comparison of average resource utilization by FF, BF, FFD, BFD, and DRAM with different datasets.

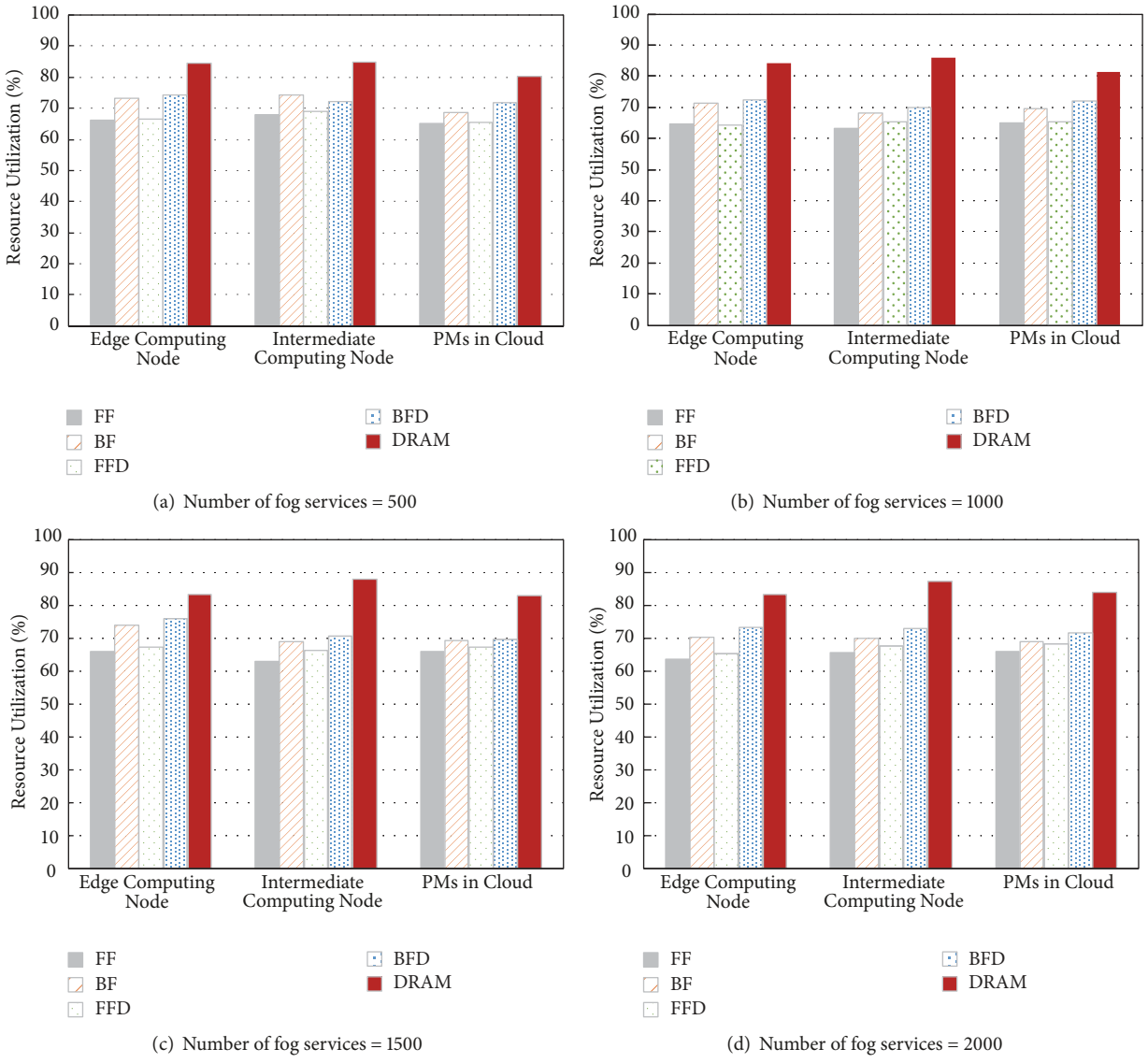

FIGURE 9: Comparison of resource utilization for different types of computing nodes by FF, BF, FFD, BFD, and DRAM with different datasets.

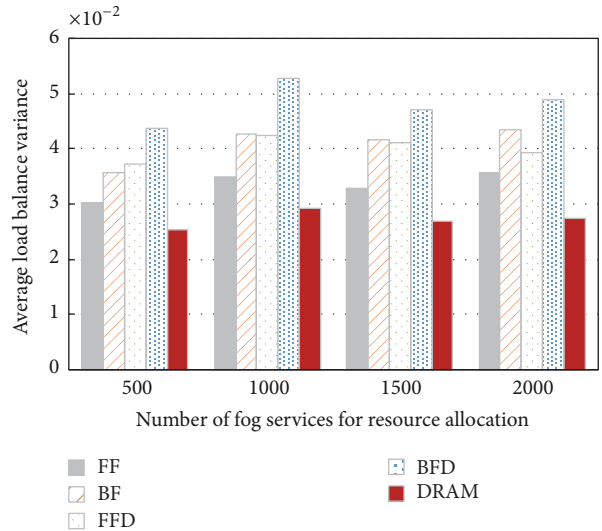

FIGURE 10: Comparison of average load-balance variance by FF, BF, FFD, BFD, and DRAM with different datasets.

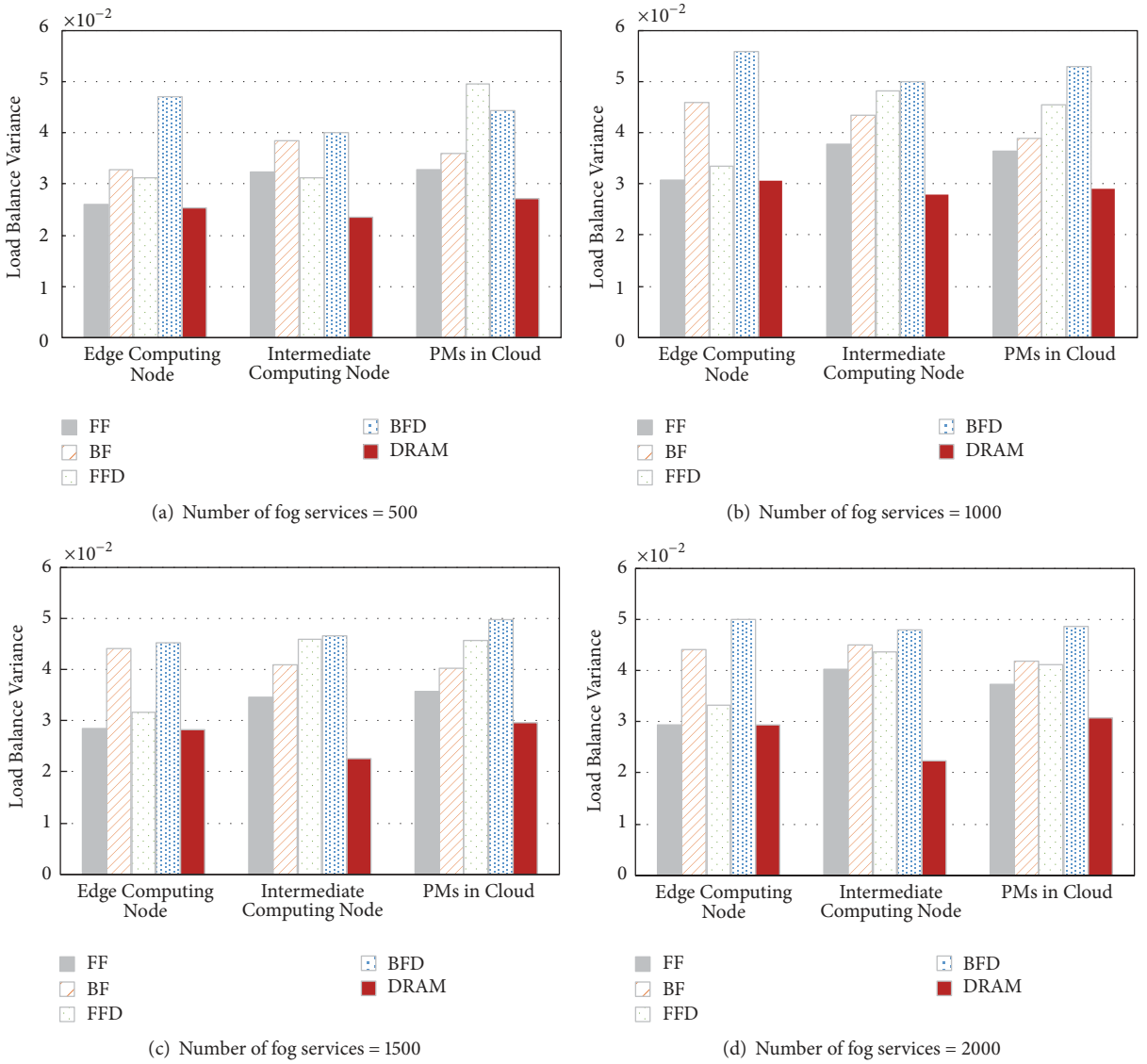

FIGURE 11: Comparison of load-balance variance values for different types of computing nodes by FF, BF, FFD, BFD, and DRAM with different datasets.

data storage and processing usually benefit from the cloud computing which provides scalable and elastic resources for executing the IoT applications [21–26]. In the ever-expanding data volume, cloud computing is difficult to provide efficient, low-latency computing services, and fog computing is proposed to complement the above shortage of cloud computing [1, 2].

Compared to the remote cloud computing center, fog computing is closer to the Internet of Things devices and sensors. Fog computing can quickly solve lightweight tasks with fast response. In the era of big data, with cloud computing expansion, fog computing is widely used in the medical, transportation, and communication fields, to name a few [21, 27–30].

Hu et al. [21] designed a fog calculation framework in detail and compared it with the traditional cloud computing, and a practical application case in the fog computing environment was put forward. Similarly, Kitanov and Janevski [27] compared the cloud computing on computing performance with fog computing in the 5G network, reflecting the superiority of fog computing. Akrivopoulos et al. [28] presented a technology to respond to the combination of IoT applications and the fog computing and introduced the use of fog computing in an automated medical monitoring platform to improve the medical work for the patients. Arfat et al. [29] not only unprecedentedly proposed the integration of mobile applications, big data analysis, and fog computing, but also introduced Google Maps as an example, showing the system of information feedback diversification. Taneja and Davy [30] studied the computational efficiency in the fog environment and constructed a resource-aware placement.

Generally, resource allocation refers to the allocation of specific, limited resources and effective management to achieve the optimal use of resources. The original intention of cloud computing is to allocate network resources on demand, so that it is the same as the use of water and electricity billing

[31, 32]. In the cloud computing, resource allocation method can effectively help to achieve the goal of high resource usage and energy saving for centralized resource management for different types of applications [4, 33–35]. Fog computing, as an extension of cloud computing paradigm, also needs to conduct resource allocation to achieve high-efficiency resource usage.

Mashayekhy et al. [36] proposed an auction-based online mechanism; it can access in real time the actual needs of users and the allocation of appropriate resources to the user price. Kwak et al. [37] developed a DREAM algorithm for complex tasks in mobile devices, saving 35% of total energy and managing network resources. To address the challenges of high latency and resource shortage in clouds, Alsaffar et al. [38] proposed the resource management framework, collaborating the cloud computing and the fog computing, and then they optimized the resource allocation in fog computing. Xiang et al. [39] designed a RAN (F-RAN) architecture based on atomization calculation, which effectively achieved the high resource usage and could coordinate the global resource scheduling.

Load balancing is an effective factor to determine the resource allocation strategy. For multiple computing tasks, load balancing could promote the resource managers to assign these tasks to multiple computing nodes for execution. The realization of load balancing not only can save the cost of hardware facilities but also can improve resource efficiency.

Banerjee and Hecker [40] proposed a distributed resource allocation protocol algorithm to realize load balancing in a large-scale distributed network; as a result, compared to the FIFO, the response time and resource utilization could be greatly improved. Govindaraju and Duran-Limon [41] designed a method based on the lifecycle-related Service Level Agreement (SLA) parameters of the virtual machines in cloud environment to address resource utilization and cost issues. Evolution algorithms are proved to be powerful to solve the multiobjective problem, which could be leveraged in the resource scheduling in the fog computing [42]. Jeyakrishnan and Sengottuvelan [43] developed a new algorithm, while saving operating costs, while maximizing the use of resources, in the balanced scheduling compared to SA, PSO, and ADS being more outstanding.

For the load balancing maximization problem solved in this paper, the traditional operations research is proved to be efficient in optimization problem with constraints [44, 45]. The game theory is also efficient for the resource allocation with resource competition, and the Nash Equilibria are often needed to be verified first [46, 47].

To the best of our knowledge, there are few studies focusing on the resource allocation of fog services in the fog environment which aims to realize the load balancing for the computing nodes in both fog and cloud.

6. Conclusion and Future Work

In recent years, IoT has been one of the most popular technologies for daily lives. With rapid development of IoT, fog computing is emerging as one of the most powerful paradigms for processing the IoT applications. In the fog environment, the IoT applications are performed by the edge computing nodes and the intermediate computing nodes in the fog, as well as the physical machines in the cloud platforms. To achieve the dynamic load balancing for each type of computing node in the fog and cloud, a dynamic resource allocation method, named DRAM, for load balancing has been developed in this paper. Firstly, a system framework in fog computing was presented and load balancing for the computing nodes is analyzed accordingly. Then, the DRAM method has been implemented based on the static resource allocation and dynamic resource scheduling for fog services. As a result, the experimental evaluations and comparison analysis were carried out to verify the validity of our proposed method.

For future work, we try to analyze the negative impact of the service migration, including the traffic for different types of computing nodes, the cost for service migration, the performance degradation for the service migration, and the data transmission cost. Furthermore, we will design a corresponding method to balance the negative effects and the positive impacts for service migration.

Key Terms and Descriptions Involved in Resource Scheduling in Fog Environment

M:	The number of computing nodes
P:	The set of computing nodes, $P = \{p_1, p_2, \ldots, p_M\}$
N:	The number of services
S:	The set of services, $S = \{s_1, s_2, \ldots, s_N\}$
p_m:	The mth ($1 \le m \le M$) computing node in P
s_n:	The nth ($1 \le n \le N$) service in S
c_m:	The resource capacity of the mth computing node p_m
r_n:	The set of resource requirements of the nth service s_n
W:	The number of types of processing nodes
β_w^m:	A flag to judge whether p_m is a type w of computing nodes
$ru_m(t)$:	The resource utilization for p_m at time t
$RU_w(t)$:	The resource utilization for the wth type computing nodes at time t
$lb_m(t)$:	The load-balance variance for p_m at time t
$LB_w(t)$:	The load-balance variance for the wth type computing nodes at time t
LB_w:	The average load-balance variance for the wth type computing nodes.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research is supported by the National Natural Science Foundation of China under Grants nos. 61702277, 61672276, 61772283, 61402167, and 61672290, the Key Research and Development Project of Jiangsu Province under Grants nos.

BE2015154 and BE2016120, and the Natural Science Foundation of Jiangsu Province (Grant no. BK20171458). Besides, this work is also supported by the Startup Foundation for Introducing Talent of NUIST, the Open Project from State Key Laboratory for Novel Software Technology, Nanjing University, under Grant no. KFKT2017B04, the Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD) fund, Jiangsu Collaborative Innovation Center on Atmospheric Environment and Equipment Technology (CICAEET), and the project "Six Talent Peaks Project in Jiangsu Province" under Grant no. XYDXXJS-040. Special thanks are due to Dou Ruihan, Nanjing Jinling High School, Nanjing, China, for his intelligent contribution to our algorithm discussion and experiment development.

References

[1] Y. Kong, M. Zhang, and D. Ye, "A belief propagation-based method for task allocation in open and dynamic cloud environments," *Knowledge-Based Systems*, vol. 115, pp. 123–132, 2017.

[2] S. Sarkar, S. Chatterjee, and S. Misra, "Assessment of the Suitability of Fog Computing in the Context of Internet of Things," *IEEE Transactions on Cloud Computing*, vol. 6, no. 1, pp. 46–59, 2015.

[3] K. Peng, R. Lin, B. Huang, H. Zou, and F. Yang, "Link importance evaluation of data center network based on maximum flow," *Journal of Internet Technology*, vol. 18, no. 1, pp. 23–31, 2017.

[4] E. Luo, M. Z. Bhuiyan, G. Wang, M. A. Rahman, J. Wu, and M. Atiquzzaman, "PrivacyProtector: Privacy-Protected Patient Data Collection in IoT-Based Healthcare Systems," *IEEE Communications Magazine*, vol. 56, no. 2, pp. 163–168, 2018.

[5] P. Li, S. Zhao, and R. Zhang, "A cluster analysis selection strategy for supersaturated designs," *Computational Statistics & Data Analysis*, vol. 54, no. 6, pp. 1605–1612, 2010.

[6] G.-L. Tian, M. Wang, and L. Song, "Variable selection in the high-dimensional continuous generalized linear model with current status data," *Journal of Applied Statistics*, vol. 41, no. 3, pp. 467–483, 2014.

[7] S. Wang, T. Lei, L. Zhang, C.-H. Hsu, and F. Yang, "Offloading mobile data traffic for QoS-aware service provision in vehicular cyber-physical systems," *Future Generation Computer Systems*, vol. 61, pp. 118–127, 2016.

[8] S. Yi, C. Li, and Q. Li, "A survey of fog computing: concepts, applications and issues," in *Proceedings of the Workshop on Mobile Big Data (Mobidata '15)*, pp. 37–42, ACM, Hangzhou, China, June 2015.

[9] X. L. Xu, X. Zhao, F. Ruan et al., "Data placement for privacy-aware applications over big data in hybrid clouds," *Security and Communication Networks*, vol. 2017, Article ID 2376484, 15 pages, 2017.

[10] X. Xu, W. Dou, X. Zhang, C. Hu, and J. Chen, "A traffic hotline discovery method over cloud of things using big taxi GPS data," *Software: Practice and Experience*, vol. 47, no. 3, pp. 361–377, 2017.

[11] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, "Fog computing: A platform for internet of things and analytics," in *Big Data and Internet of Things: A Roadmap for Smart Environments*, vol. 546 of *Studies in Computational Intelligence*, pp. 169–186, 2014.

[12] X. Xu, X. Zhang, M. Khan, W. Dou, S. Xue, and S. Yu, "A balanced virtual machine scheduling method for energy-performance trade-offs in cyber-physical cloud systems," *Future Generation Computer Systems*, 2017.

[13] L. Yu, L. Chen, Z. Cai, H. Shen, Y. Liang, and Y. Pan, "Stochastic Load Balancing for Virtual Resource Management in Datacenters," *IEEE Transactions on Cloud Computing*, pp. 1–14, 2016.

[14] Y. Sahu, R. K. Pateriya, and R. K. Gupta, "Cloud server optimization with load balancing and green computing techniques using dynamic compare and balance algorithm," in *Proceedings of the 5th International Conference on Computational Intelligence and Communication Networks, CICN 2013*, pp. 527–531, India, September 2013.

[15] G. Soni and M. Kalra, "A novel approach for load balancing in cloud data center," in *Proceedings of the 2014 4th IEEE International Advance Computing Conference, IACC 2014*, pp. 807–812, India, February 2014.

[16] X. Xu, W. Dou, X. Zhang, and J. Chen, "EnReal: An Energy-Aware Resource Allocation Method for Scientific Workflow Executions in Cloud Environment," *IEEE Transactions on Cloud Computing*, vol. 4, no. 2, pp. 166–179, 2016.

[17] S. Li and Y. Zhang, "On-line scheduling on parallel machines to minimize the makespan," *Journal of Systems Science & Complexity*, vol. 29, no. 2, pp. 472–477, 2016.

[18] G. Wang, X. X. Huang, and J. Zhang, "Levitin-Polyak well-posedness in generalized equilibrium problems with functional constraints," *Pacific Journal of Optimization. An International Journal*, vol. 6, no. 2, pp. 441–453, 2010.

[19] B. Qu and J. Zhao, "Methods for solving generalized Nash equilibrium," *Journal of Applied Mathematics*, vol. 2013, Article ID 762165, 2013.

[20] S. Lian and Y. Duan, "Smoothing of the lower-order exact penalty function for inequality constrained optimization," *Journal of Inequalities and Applications*, Paper No. 185, 12 pages, 2016.

[21] P. Hu, S. Dhelim, H. Ning, and T. Qiu, "Survey on fog computing: architecture, key technologies, applications and open issues," *Journal of Network and Computer Applications*, vol. 98, pp. 27–42, 2017.

[22] A. V. Dastjerdi and R. Buyya, "Fog Computing: Helping the Internet of Things Realize Its Potential," *The Computer Journal*, vol. 49, no. 8, Article ID 7543455, pp. 112–116, 2016.

[23] J. Shen, T. Zhou, D. He, Y. Zhang, X. Sun, and Y. Xiang, "Block design-based key agreement for group data sharing in cloud computing," *IEEE Transactions on Dependable and Secure Computing*, vol. PP, no. 99, 2017.

[24] Z. Xia, X. Wang, L. Zhang, Z. Qin, X. Sun, and K. Ren, "A privacy-preserving and copy-deterrence content-based image retrieval scheme in cloud computing," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 11, pp. 2594–2608, 2016.

[25] J. Shen, D. Liu, J. Shen, Q. Liu, and X. Sun, "A secure cloud-assisted urban data sharing framework for ubiquitous-cities," *Pervasive and Mobile Computing*, vol. 41, pp. 219–230, 2017.

[26] Z. Fu, X. Wu, C. Guan, X. Sun, and K. Ren, "Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 12, pp. 2706–2716, 2016.

[27] S. Kitanov and T. Janevski, "Energy efficiency of Fog Computing and Networking services in 5G networks," in *Proceedings of*

the 17th IEEE International Conference on Smart Technologies, EUROCON 2017, pp. 491–494, Macedonia, July 2017.

[28] O. Akrivopoulos, I. Chatzigiannakis, C. Tselios, and A. Antoniou, "On the Deployment of Healthcare Applications over Fog Computing Infrastructure," in *Proceedings of the 41st IEEE Annual Computer Software and Applications Conference Workshops, COMPSAC 2017*, pp. 288–293, Italy, July 2017.

[29] Y. Arfat, M. Aqib, R. Mehmood et al., "Enabling Smarter Societies through Mobile Big Data Fogs and Clouds," in *Proceedings of the International Workshop on Smart Cities Systems Engineering (SCE 2017)*, vol. 109, pp. 1128–1133, Procedia Computer Science, Portugal, May 2017.

[30] M. Taneja and A. Davy, "Resource Aware Placement of Data Analytics Platform in Fog Computing," in *Proceedings of the 2nd International Conference on Cloud Forward: From Distributed to Complete Computing, CF 2016*, pp. 153–156, Procedia Computer Science, Spain, October 2016.

[31] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: a survey," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 84–106, 2013.

[32] S. S. Manvi and G. Krishna Shyam, "Resource management for Infrastructure as a Service (IaaS) in cloud computing: a survey," *Journal of Network and Computer Applications*, vol. 41, no. 1, pp. 424–440, 2014.

[33] Y. Ren, J. Shen, D. Liu, J. Wang, and J.-U. Kim, "Evidential quality preserving of electronic record in cloud storage," *Journal of Internet Technology*, vol. 17, no. 6, pp. 1125–1132, 2016.

[34] Y. Chen, C. Hao, W. Wu, and E. Wu, "Robust dense reconstruction by range merging based on confidence estimation," *Science China Information Sciences*, vol. 59, no. 9, Article ID 092103, pp. 1–11, 2016.

[35] T. Ma, Y. Zhang, J. Cao, J. Shen, M. Tang, and Y. Tian, "Abdullah Al-Dhelaan, Mznah Al-Rodhaan, KDVEM: a k-degree anonymity with Vertex and Edge Modification algorithm," *Computing*, vol. 70, no. 6, pp. 1336–1344, 2015.

[36] L. Mashayekhy, M. M. Nejad, D. Grosu, and A. V. Vasilakos, "An online mechanism for resource allocation and pricing in clouds," *Institute of Electrical and Electronics Engineers. Transactions on Computers*, vol. 65, no. 4, pp. 1172–1184, 2016.

[37] J. Kwak, Y. Kim, J. Lee, and S. Chong, "DREAM: Dynamic Resource and Task Allocation for Energy Minimization in Mobile Cloud Systems," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 12, pp. 2510–2523, 2015.

[38] A. A. Alsaffar, H. P. Pham, C.-S. Hong, E.-N. Huh, and M. Aazam, "An Architecture of IoT Service Delegation and Resource Allocation Based on Collaboration between Fog and Cloud Computing," *Mobile Information Systems*, vol. 2016, Article ID 6123234, pp. 1–15, 2016.

[39] H. Xiang, M. Peng, Y. Cheng, and H.-H. Chen, "Joint mode selection and resource allocation for downlink fog radio access networks supported D2D," in *Proceedings of the 11th EAI International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness, QSHINE 2015*, pp. 177–182, Taiwan, August 2015.

[40] S. Banerjee and J. P. Hecker, "A Multi-agent System Approach to Load-Balancing and Resource Allocation for Distributed Computing," in *First Complex Systems Digital Campus World E-Conference*, pp. 393–408, 2017.

[41] Y. Govindaraju and H. Duran-Limon, "A QoS and energy aware load balancing and resource allocation framework for iaas cloud providers," in *Proceedings of the 9th IEEE/ACM International Conference on Utility and Cloud Computing, UCC 2016*, pp. 410–415, China, December 2016.

[42] Y. Yuan, H. Xu, B. Wang, and X. Yao, "A new dominance relation-based evolutionary algorithm for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 1, pp. 16–37, 2016.

[43] V. Jeyakrishnan and P. Sengottuvelan, "A Hybrid Strategy for Resource Allocation and Load Balancing in Virtualized Data Centers Using BSO Algorithms," *Wireless Personal Communications*, vol. 94, no. 4, pp. 2363–2375, 2017.

[44] H. Wu, Y. Ren, and F. Hu, "Continuous dependence property of BSDE with constraints," *Applied Mathematics Letters*, vol. 45, pp. 41–46, 2015.

[45] Y. Wang, X. Sun, and F. Meng, "On the conditional and partial trade credit policy with capital constraints: a Stackelberg model," *Applied Mathematical Modelling: Simulation and Computation for Engineering and Environmental Systems*, vol. 40, no. 1, pp. 1–18, 2016.

[46] J. Zhang, B. Qu, and N. Xiu, "Some projection-like methods for the generalized Nash equilibria," *Computational optimization and applications*, vol. 45, no. 1, pp. 89–109, 2010.

[47] C. Wang, C. Ma, and J. Zhou, "A new class of exact penalty functions and penalty algorithms," *Journal of Global Optimization*, vol. 58, no. 1, pp. 51–73, 2014.