IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 8, 2019, accepted June 4, 2019, date of publication June 14, 2019, date of current version June 28, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2922992

Energy-Efficient and Delay-Guaranteed Workload Allocation in IoT-Edge-Cloud Computing Systems

MIAN GUO[1], (Member, IEEE), LEI LI[2], AND QUANSHENG GUAN[2], (Member, IEEE)
[1]School of Electronic and Information Engineering, Guangdong University of Petrochemical Technology, Maoming 525000, China
[2]School of Electronic and Information Engineering, South China University of Technology, Guangzhou 510640, China

Corresponding author: Mian Guo (mian.guo123@gmail.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61431005, Grant 61671208, and Grant 61671211, in part by the Natural Science Foundation of Guangdong Province of China under Grant 2015A030310287 and Grant 2016A030308006, and in part by the Science and Technology Planning Project of Guangdong Province of China under Grant 2017A010101027.

ABSTRACT Edge computing has recently emerged as an extension to cloud computing for quality of service (QoS) provisioning particularly delay guarantee for delay-sensitive applications. By offloading the computationally intensive workloads to edge servers, the quality of computation experience, e.g., network transmission delay and transmission energy consumption, could be improved greatly. However, the computation resource of an edge server is so scarce that it cannot respond quickly to the bursting computation requirements. Accordingly, queuing delay is un-negligible in a computationally intensive environment, e.g., a computing environment consists of the Internet of Things (IoT) applications. In addition, the computation energy consumption in edge servers may be higher than that in clouds when the workload is heavy. To provide QoS for end users while achieving green computing for computing systems, the cooperation between edge servers and the cloud is significantly important. In this paper, the energy-efficient and delay-guaranteed workload allocation problem in an IoT-edge-cloud computing system are investigated. We formulate a delay-based workload allocation problem which suggests the optimal workload allocations among local edge server, neighbor edge servers, and cloud toward the minimal energy consumption as well as the delay guarantee. The problem is then tackled using a delay-base workload allocation (DBWA) algorithm based on Lyapunov drift-plus-penalty theory. The theoretical analysis and simulation results have been conducted to demonstrate the efficiency of the proposal for energy efficiency and delay guarantee in an IoT-edge-cloud system.

INDEX TERMS Edge computing, cloud computing, workload allocation, energy efficiency, delay guarantee.

I. INTRODUCTION

With the surging applications of Internet of Things (IoT), a tremendous amount of data is generated from massively distributed end users, e.g., IoT devices, requiring timely processing to extract its maximum value. Many IoT applications, such as augmented reality and self-driving, are delay sensitive and computation intensive [1]–[3]. Although the success of cloud computing for supporting high performance computing has been witnessed in recent years, its inefficiency in quality of service (QoS) provisioning for delay sensitive applications as well as high energy consumption has been the bottleneck for the development of delay-sensitive IoT applications [4], [5]. First, the cloud computing resource is often physically located in remote data centers, which often leads to a large network delay for transmitting distributed data to the remote cloud. Particularly, the ever-increasing network load leads to intolerable network delay with the ever-increasing mobile applications. Second, the behavior of transmitting enormous volumes of data from end users to remote clouds also consumes extremely high network bandwidth resource and transmission energy.

Edge computing has recently emerged as an extension to cloud computing for QoS provisioning particularly delay guarantee for delay-sensitive applications [1]. In edge computing, network edge devices, such as base stations, access points and edge routers, are endowed with cloud-like computing and storage capabilities [6]–[8]. Thus, the computation requests can be offloaded to a nearby edge server for processing, such that both network transmission delay and

The associate editor coordinating the review of this manuscript and approving it for publication was Ligang He.

transmission energy consumption can be reduced. However, comparing to the cloud, the computation resource of an edge server is far scarce. It cannot response quickly to all bursty computation requirements. Accordingly, some requests may experience long queuing delays in edge computing, violating their delay requirements. In addition, the computation energy consumption in an edge server may exceed that in a cloud in a heavily loaded environment. Therefore, to provide QoS for end users as well as achieving green computing for the computing system, the cooperation between edge servers and the cloud is significantly important [5], [9].

Workload allocation among edge servers and the remote cloud is a key edge-cloud cooperation technique that affects the QoS provisioning particularly delay-guarantee for delay sensitive applications as well as the energy consumption of the edge-cloud systems [1], [10]. However, the dynamic traffic characteristics as well as heterogeneous computing capabilities of edge and cloud servers challenge the workload allocation. First, the computation requests are generated stochastically and the computation amount is also varying over request and over time. Offline algorithms are unsuitable for solving such kind of workload allocation problem in an edge-cloud system. Second, the computation resource in an edge server is often far less than that in a cloud. Queuing delay in an edge server may dominate the transmission delay from an edge to a remote cloud. Therefore, the tradeoff between queuing delay and transmission delay due to the resource heterogeneous of edge servers and clouds complex the workload allocation scheme.

This paper investigates the energy-efficient and delay-guaranteed workload allocation problem in an IoT-edge-cloud system. In such a system, there are a number of IoT regions, which generate computation requests stochastically. Each of IoT region endowed with a local edge server and several neighbor servers. The goal of the paper is to find out optimal policies to allocate workloads among local edge server, neighbor servers and the remote cloud, aiming at minimize the energy consumption of the system and provide per-job granular delay guarantee for the requests. To this end, we formulate a delay-based workload allocation problem with the goal of minimizing energy consumption. Then, we tackle the problem by proposing a delay-based workload allocation (DBWA) algorithm based on Lyapunov drift-plus-penalty theory. The efficiency of the proposal in energy efficiency and delay guarantee has been demonstrated through theoretical analysis and simulation results. The main contributions of this paper are summarized as follows.

- A delay-based workload allocation problem is formulated, which suggests the optimal workload allocations among local edge server, neighbor edge servers and the remote cloud toward the minimal energy consumption of the IoT-edge-cloud system as well as delay guarantee for arrival jobs.
- The DBWA algorithm is developed to find out optimal solutions. Specifically, the drift-plus-penalty properties of energy consumption minimization constrained to system stable are analyzed. Then, DBWA is proposed to achieve the goal of minimizing energy consumption of the system as well as delay guarantee for arrival jobs. The theoretical analysis shown that, the energy consumption is within $O(1/V)$ of optimality under DBWA.
- The simulations have been conducted to illustrate that, DBWA can significantly improve the performance of edge computing and cloud computing in terms of reducing both of energy consumption and end-to-end (e2e) delay. DBWA can also bound the e2e delays of arrival jobs to their requirements.

The remainder of this paper is organized as follows. Section II introduces the related work. Section III describes the system, traffic, delay and energy consumption models and then formulates the problem. Section IV describes the details of the proposal. Simulation studies are conducted to demonstrate the efficiency of the proposal in Section V. Section VI concludes this paper.

II. RELATED WORK

Edge computing has attracted significant attentions in recent years [11]–[13]. It supports cloud-like computing in the network edge by deploying computing and network resources along the path between data source and cloud datacenters [1]. Fog computing [7], [14] and mobile edge computing [15], [16] are two typical edge computing paradigms. Fog computing focuses more on the infrastructure side and is generally deployed at the edge of core network, while mobile edge computing focuses more toward the mobile users' side and is generally deployed within the wireless access network. In this paper, edge computing is interchangeable with fog computing and mobile edge computing, but focuses more on the things' side.

In recent years, a number of offloading algorithms for edge computing have been proposed. Different task offloading criteria categorize into different offloading algorithms.

Chen *et al.* focused on the performance in terms of average number of beneficial cloud computing users and the average system-wide computation overhead [17]. They designed a distributed computation offloading algorithm to improve the wireless access efficiency for computation offloading in a mobile-edge cloud computing environment.

A number of computation offloading algorithms for reducing the service delay, including network delay and computation delay, have been proposed in recent years. Liu *et al.* formulated a power-constrained delay minimization problem for mobile-edge computing systems with Markov decision process and proposed an efficient one-dimensional search algorithm to solved it [18]. Yang *et al.* proposed a Multi-Dimensional Search and Adjust (MDSA) method to join computation partitioning and resource allocation to reduce the average delay for latency sensitive applications in mobile edge clouds [19]. Youselfpour *et al.* proposed a delay-minimizing policy for fog-capable devices to reduce the service delay for IoT applications [20]. Liu *et al.* studied the tradeoff between latency and reliability in task offloading to

mobile edge computing [21]. Zhang *et al.* studied the computation resource allocation problem in a three-tier IoT fog network [22], focusing on the performance in terms of utility, which is the revenue received from the workload data minus both the cost of service delay and payment to the data service operator. Li *et al.* studied the resource allocation and task offloading problem for heterogeneous real-time tasks in a fog queuing system [23], aiming to yield a tradeoff between high throughput and high task completion ratio. Different from the above proposals that concerned on average delay, our studies focus on satisfying individuals' delay requirements.

In addition to the aforementioned studies, which focus on the delay performance, a number of energy efficient algorithms have also been explored for edge computing. Wang *et al.* has proposed a resource allocation scheme to minimize the multi-antenna access point (AP)'s total energy consumption subject to the users' individual computation latency constraints [24]. Deng *et al.* investigated the optimal workload allocation problem in a fog-cloud computing system toward the minimal power consumption with the constrained service delay [25]. Zhang *et al.* has proposed an energy-efficient computation offloading scheme to minimize the energy consumption under the latency constraints for mobile edge computing in 5G heterogeneous networks [26]. A deep reinforcement learning based offloading scheme is proposed for maximizing the user's utility obtained by task execution while minimizing the energy consumption, task processing delay, task loss probability and required payment in adhoc mobile clouds [27]. Lyu *et al.* designed a selective offloading scheme to minimize the energy consumption of IoT devices, where the signaling overhead can be further reduced by enabling the devices to be self-nominated or self-denied for offloading [28].

This paper differs from the existing works in two aspects. First, the energy-efficient and delay-guaranteed workload allocation is studied in an IoT-edge-cloud system with dynamic workloads, where both computation and transmission energy consumptions are considered. Thus, we need to find out a sequence of workload allocation solutions among local edge server, neighbor edge servers and the cloud to minimize the energy consumption of the system. Second, we consider the users' individual e2e delays. Thus, a fine granular with low complexity workload allocation scheme is designed to provide per-job granular delay guarantee. To the best of our knowledge, this is the first work that tackles the optimal workload allocation for the minimal energy consumption as well as per-job granular delay guarantee in such an IoT-edge-cloud system.

III. MODEL DESCRIPTION

This section briefly introduces the system, traffic, delay and energy consumption models as well as problem formulation.

A. SYSTEM MODEL

As illustrated in Fig. 1, this paper considers an IoT-edge-cloud computing system with IoT regions, edge nodes and a cloud. Each of IoT regions endows with an edge node and a limited number of IoT devices. The edge node is the integration of an edge server and edge communication infrastructures. The edge server and the cloud have distinct computing capabilities. The IoT devices from IoT regions generate computation jobs stochastically. All the computation jobs from an IoT region are delivered to the edge node deployed in this region. The edge node makes workload allocation decisions for arrival jobs on computing locally, offloading to a neighbor edge or offloading to the cloud for computing. Notice that, to avoid ping-pong effect, we assume that when a job offloads from an edge node to a neighbor edge, it cannot be offloaded again.

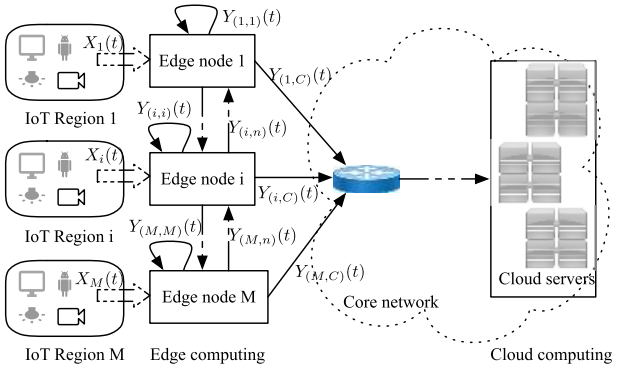

FIGURE 1. Overall architecture of an IoT-edge-cloud system.

We consider that there are M number of IoT regions and M number of edge nodes, each of which configures with an independent computing capability P_i^F [1] for $i \in \mathcal{M}$, where $\mathcal{M} = \{0, 1, \cdots, M-1\}$ is the IoT region space as well as edge node space. We assume that the computation resource of the cloud is unlimited (in comparison with the edge node), but the computation resource allocated to a job is limited (e.g., the job is computed in a virtual machine (VM) running in the cloud, where the VM is usually configured with a limited computation resource). For simplification, we assume that the computation resource allocated to a VM for a job is P^C. We assume that $P_i^F < P^C, \forall i \in \mathcal{M}$.

B. TRAFFIC MODEL

A dynamic workload model is considered: (1) the computation jobs are generated from each of IoT regions stochastically and independently; (2) in each region, the number of computation jobs per time slot follows an independent and identical distribution (i.i.d), and the sizes of jobs belonging to the same type also follow an i.i.d. Notice that, the size of a job is measured in bits in this paper. Then, according to references [29], [30], the CPU cycles required to execute a job with size S can be derived by

$$Cw = \chi S, \tag{1}$$

[1]The term computing capability refers to the maximum rate at which the server can process a computation task, e.g., $P_i^F \sim f_i^F$, where f_i^F is the CPU-cycle frequency. We assume that $f_i^F \leq f_i^{max}$.

where χ is the number of CPU cycles required to process one bit of a job.

As shown in Fig. 1, let $X_i(t)$ be the number of computation jobs generated from the i^{th} IoT region that arrive at edge node i in slot t[2] and $Xw_i(t) = \sum_{k=0}^{X_i(t)-1} S_i^k(t)$ be the corresponding workload, where $S_i^k(t)$ is the size of the k^{th} job. Let $\lambda_i = E[X_i(t)]$ be the long term job generation rate and $S_i = E[Xw_i(t)/X_i(t)]$ be the expected size of jobs generated in the i^{th} region.

Let $Y_{(i,i)}(t)$, $Y_{(i,n)}(t)$ and $Y_{(i,C)}(t)$ be the number of jobs belonging to the job set of $\{0, 1, \cdots, X_i(t) - 1\}$ that are determined to compute in the local edge node i, neighbor edge node $n \in \mathcal{M}$ and the cloud, respectively. Let $Yw_{(i,i)}(t) = \sum_{k=0}^{Y_{(i,i)}(t)-1} S_i^k(t)$, $Yw_{(i,n)}(t) = \sum_{k=0}^{Y_{(i,n)}(t)-1} S_i^k(t)$ and $Yw_{(i,C)}(t) = \sum_{k=0}^{Y_{(i,C)}(t)-1} S_i^k(t)$ be the corresponding workload.

Then, for $i \in \mathcal{M}$, we have

$$\begin{cases} X_i(t) = Y_{(i,i)}(t) + Y_{(i,n)}(t) + Y_{(i,C)}(t), \\ Xw_i(t) = Yw_{(i,i)}(t) + Yw_{(i,n)}(t) + Yw_{(i,C)}(t). \end{cases} \tag{2}$$

C. DELAY MODEL

A job may experience two types of delays, including transmission delay and computation delay.

1) TRANSMISSION DELAY

We consider two types of network transmission paths, including the path from an edge node to its neighbor edge node and the path from an edge node to the cloud. Let $bw_{(i,j)}$ denote the bandwidth of the transmission path from the node i (e.g., an edge node) to the node j (e.g., a neighbor edge node, or, the cloud). Let S^k be the size of the job k that transmits over the path. Then, the transmission delay of the job in the path is derived by

$$D^k_{comm(i,j)} = \alpha_{(i,j)} + \frac{S^k}{bw_{(i,j)}}, \tag{3}$$

where $\alpha_{(i,j)}$ is the factor of communication delay, e.g., the network delay introduced by network congestion.

2) COMPUTATION DELAY

Due to computation resource constraints in edge nodes, we assume a queuing subsystem for each of the edge servers. Let $Q_i(t)$ be the number of jobs queuing in the subsystem i at the beginning of slot t. Then, the evolution of the queue length Q_i follows

$$Q_i(t+1) = \max[Q_i(t) + Y_i(t) - r_i(t), 0], \tag{4}$$

where $Y_i(t)$ and $r_i(t)$ are the number of jobs arrive and service respectively at the subsystem i during slot t. $Y_i(t)$ is derived by

$$Y_i(t) = \sum_{j \in \mathcal{M}} Y_{(j,i)}(t). \tag{5}$$

Let $Qw_i(t)$ be the corresponding workload considering the number of jobs as well as job sizes queuing in the subsystem i at the beginning of slot t. Then, we have

$$Qw_i(t+1) = \max[Qw_i(t) + Yw_i(t) - \frac{P_i^F}{\chi}, 0], \tag{6}$$

where $\frac{P_i^F}{\chi}$ is the bits that the server can process in a slot; $Yw_i(t)$ is the aggregated workload arrives in slot t, which is derived by

$$Yw_i(t) = \sum_{k=0}^{Y_i(t)-1} S_i^k, \tag{7}$$

where S_i^k is the size of the k^{th} job that arrives in slot t.

Let $\mathcal{K}$ be the job space that consists of $Y_i(t)$ number of jobs that arrive in slot t for $t = 0, 1, \cdots, \infty$ in subsystem i (e.g., an edge node, or, the cloud). Then, the computation delay of the $k^{th}(k \in \mathcal{K})$ job with size S_i^k that arrives at the queuing subsystem $i \in \mathcal{M}$ in slot t could be derived by

$$D^{(i,k)}_{comp}(t) = D^{(i,k)}_{que}(t) + \frac{\chi S_i^k}{P_i^F}, \tag{8}$$

where $D^{(i,k)}_{que}$ is the queuing delay, which could be approximated by

$$D^{(i,k)}_{que}(t) = \frac{\chi(Qw_i(t) + \sum_{j=0}^{k-1} S_i^j)}{P_i^F}, \tag{9}$$

where $\sum_{j=0}^{k-1} S_i^j$ is the workload that arrives in slot t and ahead of the k^{th} job.

As for a job offloading to the cloud, we assume that the job can be computed immediately after its arrival at the cloud. Thus, the computation delay of the k^{th} job with size S^k that arrives at the cloud in slot t is derived by

$$D^{(C,k)}_{comp}(t) = \frac{\chi S^k}{P^C}. \tag{10}$$

3) END-TO-END DELAY

Accordingly, for the k^{th} job with size S_i^k that arrives at the computing node i (e.g., the local edge node, the neighbor edge node, or, the cloud) in slot t and determines to compute at this node, assuming the job was generated from IoT region j, the e2e delay of the job cloud be derived by

$$\begin{aligned} D^{(i,k)}(t) = {} & I_F^k(t) D^{(i,k)}_{comp}(t) + I_{NF}^k(t)[D^k_{comm(j,i)} + D^{(i,k)}_{comp}(t)] \\ & + I_C^k(t)[D^k_{comm(j,C)} + D^{(C,k)}_{comp}(t)], \end{aligned} \tag{11}$$

where $D^{(i,k)}_{comp}(t)$ and $D^{(C,k)}_{comp}(t)$ are the computation delays in the edge node i and the cloud C, respectively; $D^k_{comm(j,i)}$ and $D^k_{comm(j,C)}$ are the transmission delays in paths of local edge node-neighbor edge node and local edge node-the cloud that the job may experience, respectively; $I_F^k(t)$, $I_{NF}^k(t)$ and $I_C^k(t)$ are mutually exclusive binary computation decision indicators for the k^{th} job in slot t. When the job is determined to compute in the local edge node, $I_F^k(t) = 1$; when it is determined to compute in the neighbor edge node, $I_{NF}^k(t) = 1$;

[2]In this paper, the slot t refers to the time interval $[t, t+1)$.

when it is determined to compute in the cloud, $I_C^k(t) = 1$; otherwise, the indicators are set to zeros. Thus, the computation decision indicators should satisfy the following constraint:

$$I_F^k(t) + I_{NF}^k(t) + I_C^k(t) = 1, \quad \forall k \in \mathcal{K},\ t = 0,\ \cdots,\ \infty. \tag{12}$$

D. ENERGY CONSUMPTION MODEL

Similar to reference [25], we consider that the edge node and the cloud have different models of computation energy consumption.

1) COMPUTATION ENERGY CONSUMPTION IN AN EDGE NODE

As to an edge node, we model the computation energy consumption as a function of the workload, which is a monotonic increasing and strictly convex function.

Let $Pw_i^F(t)$ be the computation energy consumption in edge node $i \in \mathcal{M}$ in slot t, which is derived by

$$Pw_i^F(t) = a_f(Yw_i(t))^2 + b_f(Yw_i(t)) + c_f, \tag{13}$$

where $a_f > 0$ and $b_f, c_f \geq 0$ are factor parameters; $Yw_i(t)$ is the aggregated workload in slot t.

2) COMPUTATION ENERGY CONSUMPTION IN CLOUD

Since every job that arrives at the cloud could be handled immediately by allocating a VM with computation resource P_C, the computation energy consumption $Pw_{(i,k)}^C(t)$ for the k^{th} job with size S_i^k generated from IoT region i that arrives to the cloud in slot t can be derived by

$$Pw_{(i,k)}^C(t) = \frac{\chi S_i^k}{P_C}(A_c f_C^\theta(t) + B_c), \tag{14}$$

where $\frac{\chi S_i^k}{P_C}$ is the job execution time; $A_c > 0$, θ varies from 2.5 to 3 [25], [29] and $B_c \geq 0$ are factor parameters; $f_C(t)$ is the CPU-cycle frequency allocated to the job in the cloud, which is constrained by f_C^{max}.

Accordingly, the aggregated computation energy consumption in the cloud for the jobs arrival in slot t is derived by

$$Pw^C(t) = \sum_{i \in \mathcal{M}} \sum_{k=0}^{Y_{(i,C)}(t)-1} Pw_{(i,k)}^C(t). \tag{15}$$

3) TRANSMISSION ENERGY CONSUMPTION

We use the following model to represent the transmission energy consumption in the path from i to j in slot t.

$$Pw_{(i,j)}^{comm}(t) = a_{(i,j)}\lambda w_{(i,j)}(t), \tag{16}$$

where $a_{(i,j)} > 0$ is the transmission power in this path; $\lambda w_{(i,j)}(t)$ is the transmission workload during slot t.

Therefore, the aggregated energy consumption in terms of power consumption Pw of the IoT-edge-cloud system is the sum of per-slot energy consumptions in all computing nodes and in all communication paths in the long term, which is derived by

$$\begin{aligned} Pw &= E[Pw(t)] \\ &= E[\sum_{i \in \mathcal{M}} Pw_i^F(t) + Pw^C(t) + \sum_{i \in \mathcal{M}} \sum_{\substack{j \in \mathcal{M} \cup \mathcal{C}, \\ j \neq i}} Pw_{(i,j)}^{comm}(t)] \\ &= \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} [\sum_{i \in \mathcal{M}} Pw_i^F(\tau) + Pw^C(\tau) \\ &\quad + \sum_{i \in \mathcal{M}} \sum_{\substack{j \in \mathcal{M} \cup \mathcal{C}, \\ j \neq i}} Pw_{(i,j)}^{comm}(\tau)], \end{aligned} \tag{17}$$

where $\mathcal{C}$ is the cloud space that includes one cloud.

E. PROBLEM FORMULATION

Let $\mathcal{K}_i^X = \{0, 1, \cdots, X_i(t) - 1\}$ be the job space of IoT region i in slot t; let $\gamma_i^k(t) = (I_F^{(i,k)}(t), I_{NF}^{(i,k)}(t), I_C^{(i,k)}(t))$ be the decision vector for the k^{th} ($k \in \mathcal{K}_i^X$) job that generated from IoT region i in slot t, where $I_F^{(i,k)}(t), I_{NF}^{(i,k)}(t), I_C^{(i,k)}(t)$ are binary computation decision indicators that are constrained by Eq. (12). Specifically, $I_F^{(i,k)}(t) = 1$ if the job is determined to process in the local edge node; $I_{NF}^{(i,k)}(t) = 1$ if it is offloaded to the neighbor edge node for processing; $I_C^{(i,k)}(t) = 1$ if it is determined to process in the cloud; otherwise, the indicators are set to zeros.

Let $\gamma_i(t) = (\gamma_i^0(t), \cdots, \gamma_i^k(t), \cdots, \gamma_i^{X_i(t)-1}(t))$ be the decision vector for all jobs generated from IoT region i in slot t. Then, the decision vector for all jobs generated from all IoT regions in slot t can be represented by $\gamma(t) = (\gamma_0(t), \cdots, \gamma_i(t), \cdots, \gamma_{M-1}(t))$.

Accordingly, for jobs that are generated from IoT region i, we have

$$\begin{cases} Y_{(i,i)}(t) = \sum_{k \in \mathcal{K}_i^X} I_F^{(i,k)}(t), \\ Y_{(i,n)}(t) = \sum_{k \in \mathcal{K}_i^X} I_{NF}^{(i,k)}(t), \\ Y_{(i,C)}(t) = \sum_{k \in \mathcal{K}_i^X} I_C^{(i,k)}(t), \end{cases} \tag{18}$$

and

$$\begin{cases} Yw_{(i,i)}(t) = \sum_{k \in \mathcal{K}_i^X} I_F^{(i,k)}(t) S_i^k, \\ Yw_{(i,n)}(t) = \sum_{k \in \mathcal{K}_i^X} I_{NF}^{(i,k)}(t) S_i^k, \\ Yw_{(i,C)}(t) = \sum_{k \in \mathcal{K}_i^X} I_C^{(i,k)}(t) S_i^k, \end{cases} \tag{19}$$

where S_i^k is the size of the k^{th} job for $k \in \mathcal{K}_i^X$.

Therefore, the workload allocation problem in an IoT-edge-cloud computing environment for minimizing the energy consumption of the IoT-edge-cloud system while provisioning delay guarantee for end users (e.g., IoT devices)

is formulated as

$$\begin{aligned}
&\text{Minimize: } Pw = E[Pw(\gamma(t))] && (20)\\
&\text{Subject to : } (2),\\
&\qquad (18),\\
&\qquad (19),\\
&\qquad D^{(i,k)}(t) \le d^{(i,k)}, \quad k \in \mathcal{K}_i^X,\ i \in \mathcal{M} && (21)\\
&\qquad \overline{Q}_i < \infty, \quad i \in \mathcal{M} && (22)\\
&\qquad \overline{Qw}_i < \infty, \quad i \in \mathcal{M} && (23)
\end{aligned}$$

where Eq. (20) follows Eq.(17); Eq. (2) is the traffic constraint; Eqs. (18)-(19) follow the definition of $\gamma(t)$; $D^{(i,k)}(t)$ follows Eq. (11); $d^{(i,k)}$ is the maximum e2e delay that the k^{th} job can tolerate; Eqs. (22)-(23) are the stability constraint of the i^{th} ($i \in \mathcal{M}$) edge server, where $\overline{Q}_i$ and $\overline{Qw}_i$ are defined, respectively, as

$$\overline{Q}_i = E[Q_i(t)] = \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} Q_i(t), \tag{24}$$

and

$$\overline{Qw}_i = E[Qw_i(t)] = \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} Qw_i(t). \tag{25}$$

The decision variables are $\gamma(t)$. According to Eqs. (18)-(19), if $\gamma(t)$ is determined, then $Y_{(i,i)}(t)$, $Y_{(i,n)}(t)$, $Y_{(i,C)}(t)$ as well as $Yw_{(i,i)}(t)$, $Yw_{(i,n)}(t)$ and $Yw_{(i,C)}(t)$ are determined and Pw is also yielded. Therefore, the above problem is equivalent to determining a sequential optimal $\gamma^*(t)$ for $t = 0, \cdots, \infty$ to achieve the objective.

IV. DELAY-BASED WORKLOAD ALLOCATION

Although the behavior of computation offloading to edge nodes can reduce network transmission delay, it has the potential for extremely long queuing delay for jobs that offload to a heavily loaded edge node. Generally, a long queuing delay is introduced by the instantaneous workload requirements exceed the system capability, reflecting in a massive amount of queuing workload and a long queue length. In order to avoid extremely long queuing delay for jobs offloading to edge nodes, we first analyze the Lyapunov drift-plus-penalty [31], [32] of the queuing workloads of the IoT-edge-cloud system under any scheduling policy. Then, we propose a delay-based optimal workload allocation policy to solve the problems described in Eqs. (20)-(23) based on the analytical results.

A. LYAPUNOV DRIFT-PLUS-PENALTY

Let $Qw(t) = (Qw_0(t), \cdots, Qw_i(t), \cdots, Qw_{M-1}(t))$ be the vector of the queuing workload of M edge nodes in slot t, where $Qw_i(t)$ for $i \in \mathcal{M}$ is derived by Eq. (6). Let $L(t)$ be the Lyapunov function of the queuing workload, which is defined as

$$L(t) = \frac{1}{2} \sum_{i\in\mathcal{M}} Qw_i(t)^2. \tag{26}$$

Then, the one-step Lyapunov drift of Qw is defined as

$$\Delta L(t) = E[L(t+1) - L(t)|Qw(t)]. \tag{27}$$

We have the following lemma.

Lemma 1: Every slot t, for any value of Qw(t), and under any workload allocation policy, the Lyapunov drift satisfies

$$\Delta L(t) \le B - \sum_{i\in\mathcal{M}} Qw_i(t)\frac{P_i^F}{\chi} + \sum_{i\in\mathcal{M}} Qw_i(t)E[Yw_i(t)|Qw(t)], \tag{28}$$

where B is a finite constant.

Proof: According to Eq. (6), we have

$$\begin{aligned}
&Qw_i(t+1)^2\\
&\le \left(Qw_i(t) + Yw_i(t) - \frac{P_i^F}{\chi}\right)^2\\
&= Qw_i(t)^2 + \left(Yw_i(t) - \frac{P_i^F}{\chi}\right)^2 + 2Qw_i(t)\left(Yw_i(t) - \frac{P_i^F}{\chi}\right).
\end{aligned} \tag{29}$$

Substituting Eqs. (29) and (26) into Eq. (27), we have

$$\begin{aligned}
\Delta L(t) \le \frac{1}{2}\sum_{i\in\mathcal{M}} E\left[\left(Yw_i(t) - \frac{P_i^F}{\chi}\right)^2 |Qw(t)\right]\\
+ \sum_{i\in\mathcal{M}} Qw_i(t)E\left[Yw_i(t) - \frac{P_i^F}{\chi}|Qw(t)\right]
\end{aligned} \tag{30}$$

Since $0 \le Yw_i(t) \le \sum_{i\in\mathcal{M}} Xw_i(t)$, and $E[\sum_{i\in\mathcal{M}} Xw_i(t)] = \sum_{i\in\mathcal{M}} \lambda_i S_i = \lambda S$ according to Section III-B, where λ and S are the expectation of job generation rate and job size of the IoT-edge-cloud system, respectively. We have

$$\left(Yw_i(t) - \frac{P_i^F}{\chi}\right)^2 \le \max\left[(\frac{P_i^F}{\chi})^2, (\lambda S - \frac{P_i^F}{\chi})^2\right]. \tag{31}$$

Let $B = \sum_{i\in\mathcal{M}} \max\left[\frac{1}{2}(\frac{P_i^F}{\chi})^2, \frac{1}{2}(\lambda S - \frac{P_i^F}{\chi})^2\right]$ and substituting into Eq. (30), we have

$$\Delta L(t) \le B + \sum_{i\in\mathcal{M}} Qw_i(t)E\left[Yw_i(t) - \frac{P_i^F}{\chi}|Qw(t)\right].$$

Note that $\frac{P_i^F}{\chi}$ is independent of $Qw(t)$. Therefore, $E[\frac{P_i^F}{\chi}|Qw(t)] = \frac{P_i^F}{\chi}$. Accordingly,

$$\begin{aligned}
\Delta L(t) \le B - \sum_{i\in\mathcal{M}} Qw_i(t)\frac{P_i^F}{\chi}\\
+ \sum_{i\in\mathcal{M}} Qw_i(t)E[Yw_i(t)|Qw(t)].
\end{aligned} \tag{32}$$

Then the statement follows. □

Since our goal is to find out a sequential optimal workload allocation decisions $\gamma^*(t)$ for $t = 0, \cdots, \infty$ to minimize

the energy consumption Pw of the IoT-edge-cloud system constrained to service delay for end users, we add $Pw(\gamma(t))$ as a penalty to the Lyapunov drift of the queuing workload. Specifically, we define the drift-plus-penalty of Qw as $\Delta L(t) + VE[Pw(\gamma(t))|Qw(t)]$. According to *Lemma* 1, we have the following lemma.

Lemma 2: Every slot t, for any value of Qw(t), and under any workload allocation policy, we have

$$\begin{aligned}&\Delta L(t) + VE[Pw(\gamma(t))|Qw(t)]\\ &\quad\le B + VE[Pw(\gamma(t))|Qw(t)] - \sum_{i\in\mathcal{M}} Qw_i(t)\frac{P_i^F}{\chi}\\ &\quad\quad + \sum_{i\in\mathcal{M}} Qw_i(t)E[Yw_i(t)|Qw(t)],\end{aligned} \tag{33}$$

where B is the same constant from Lemma 1 that does not depend on V; V is a nonnegative control parameter that is chosen as desired and will affect the energy consumption-delay tradeoff.

Lemma 2 follows by substracting $VE[Pw(\gamma(t))|Qw(t)]$ into *Lemma* 1 from both sides.

B. DBWA

According to the Lyapunov drift theory, if an algorithm can be designed to control the Lyaponov drift-plus-penalty $\Delta L(t) + VE[Pw(\gamma(t))|Qw(t)]$ as described in inequality (33) towards negative, then the queue $Qw(t)$ would be stable while the optimal $E[Pw]$ would be approximated. Therefore, based on the results of *Lemmas* 1-2, we propose a delay-based workload allocation (DBWA) algorithm to find out a sequential optimal workload allocation decisions $\gamma^*(t)$ for $t = 0, \cdots, \infty$ to minimize a bound on the right-hand side of inequality. (33) every slot, such that minimize a bound on $\Delta L(t) + VE[Pw(\gamma(t))|Qw(t)]$. The detail of DBWA is shown in *Algorithm* 1.

In DBWA, since for the jobs generated from IoT region i, the offloading decisions are local edge i, neighbor edge n, or the cloud, we have $Yw_{(i,j')}(t) = 0$ for $j' \in \mathcal{M} - \{i, n\}$. Accordingly, we use $\sum_{j\in\{i,n\}} Qw_j(t)Yw_j(t)$ in Eq. (34) instead of $\sum_{j\in\mathcal{M}} Qw_j(t)Yw_j(t)$, as shown in *Algorithm* 1. This paper assumes that the edge nodes can obtain the workload states (e.g., Q and Qw) of other nodes at most once in a slot. Thus, an edge node cannot obtain the updated job arrival events of other nodes within a slot, e.g., the edge node i cannot obtain $Yw_{(j,n)}(t)$ for $j \in \mathcal{M} - \{i\}$ in slot t. Accordingly, we use Eq. (36) in *Algorithm* 1 to approximate the workload of the neighbor node.

Generally, a job in an IoT-edge-cloud system experiences two processes, including workload allocation and scheduling. Workload allocation process determines where to computation offload the job, while scheduling services the job based on the computation offloading decision. The detail of our proposal for handling jobs in the IoT-edge-cloud system based on DBWA is described in *Algorithm* 2.

Algorithm 1 Delay-Based Workload Allocation Algorithm (DBWA)

Input: $X_i(t)$, $Qw(t)$.

1) **Initialization:** $Yw_{(i,i)}(t) = 0$, $Yw_{(i,n)}(t) = 0$ for $n \in$ {i's neighbor list}.
2) **Decision process:** Choose $\gamma_i^*(t)$ as the solution to the following:
   Minimize:
$$VPw(\gamma_i(t)) + \sum_{j=\{i,n\}} Qw_j(t)Yw_j(\gamma_i(t)) \tag{34}$$
   Subject to:
$$(2),$$
$$(18),$$
$$(19),$$
$$Yw_i(t) = \sum_{\substack{j\in\mathcal{M},\\ j\neq i}} Yw_{(j,i)}(t) + Yw_{(i,i)}(t), \tag{35}$$
$$Yw_n(t) = Yw_{(i,n)}(t), \tag{36}$$
$$D^{(i,k)}(t) \le d^{(i,k)}. \tag{37}$$
   where Eq. (2) is the traffic constraint; Eqs. (18)-(19), (35)-(36) follow the definition of $\gamma(t)$.
   Default: if no solution satisfies Eq. (37), set $I_F^{(i,k)}(t) = I_{NF}^{(i,k)}(t) = 0$ and $I_C^{(i,k)} = 1$.
3) **Processing the decision:** observe $\gamma_i^*(t)$, for $k \in \mathcal{K}_i^X$,**do**
   a) If $I_F^{(i,k)}(t) = 1$: buffer the job into the local queuing system;
   b) Else if $I_{NF}^{(i,k)}(t) = 1$: transmit the job to the neighbor edge node n;
   c) Else: transmit the job to the cloud.

Output: $\gamma_i^*(t)$.

C. PERFORMANCE ANALYSIS

This subsection investigates the efficiency of the proposed DBWA scheme by analyzing the stability of the queuing system of the edge nodes and energy consumption performance of the IoT-edge-cloud system under DBWA.

Theorem 1: For $d_i^{max} < \infty$ and $0 < \overline{S_i} < \infty$, where $d_i^{max} = \max\{d^{(i,k)} : k \in \mathcal{K}_i^X\}$ for $i \in \mathcal{M}$ over all slots, and $\overline{S_i}$ is the expected size of jobs computed in edge server i, the edge computing subsystems of the IoT-edge-cloud system are stable under the DBWA scheme.

Proof: Let D_i^{max} be the maximum e2e delay given by edge server i for $i \in \mathcal{M}$ under DBWA. Let d_i be the delay requirement of the corresponding job. Then, according to *Algorithm* 1, we have

$$D_i^{max} \le d_i.$$

Since the job was computed in an edge node, according to Eq.(11), $I_F = 1$, or, $I_{NF} = 1$ for the job.

Algorithm 2 DBWA-Based Workload Allocation and Scheduling Processes

Initialization: $Q(0) = 0$, $Qw(0) = 0$.
Every slot $t \geq 0$, **do**

1) **Workload allocation process:**
   a) The edge node i ($i \in \mathcal{M}$) receives $X_i(t)$ number of jobs that were generated from IoT region i, **do**
      i) Observe $Qw(t)$, and initiate *Algorithm* 1 for edge node i;
      ii) Calculate $Y_{(i,i)}(t)$ and $Yw_{(i,i)}(t)$ with Eqs. (18)-(19), and update $Q_i(t)$ and $Qw_i(t)$ with Eq. (38).

$$\begin{cases} Q_i(t+1) = Q_i(t) + Y_{(i,i)}(t), \\ Qw_i(t+1) = Qw_i(t) + Yw_{(i,i)}(t). \end{cases} \tag{38}$$

   b) The edge node i ($i \in \mathcal{M}$) receives $Y_{(j,i)}(t)$ ($j \in \mathcal{M}$ and $j \neq i$) number of jobs with workload $Yw_{(j,i)}(t)$ that were offloaded from edge node j, **do**
      i) Buffer the jobs into the local queuing system;
      ii) Update $Q_i(t)$ and $Qw_i(t)$ with Eq. (39).

$$\begin{cases} Q_i(t+1) = Q_i(t) + Y_{(j,i)}(t), \\ Qw_i(t+1) = Qw_i(t) + Yw_{(j,i)}(t). \end{cases} \tag{39}$$

2) **Scheduling process:**
   a) Scheduling in edge node $i (i \in \mathcal{M})$:
      i) Process the waiting jobs with service rate P_i^F in first-in-first-out (FIFO) discipline;
      ii) Update $Q_i(t)$ and $Qw_i(t)$ with Eq. (40).

$$\begin{cases} Q_i(t+1) = \max[Q_i(t) - r_i(t), 0] \\ Qw_i(t+1) = \max[Qw_i(t) - \dfrac{P_i^F}{\chi}, 0]. \end{cases} \tag{40}$$

      where $r_i(t)$ represents the number of jobs that are processed in edge node i in slot t.
   b) Scheduling in the cloud:
      i) The cloud receives $\sum_{i\in\mathcal{M}} \sum_{k=0}^{Y_{(i,C)}(t)-1}$ number of jobs at the beginning of slot t;
      ii) Initiate $\sum_{i\in\mathcal{M}} \sum_{k=0}^{Y_{(i,C)}(t)-1}$ number of VMs to process these jobs with service rate P^C, respectively.

Case 1: When the job was computed in local edge node, $I_F = 1$. Then, substituting D_i^{max} with Eqs. (8)-(9), we have

$$\left((Qw_i(t) + Yw_i(t))\frac{\chi}{P_i^F}\right)^{max} \leq d_i,$$

Since $Yw_i(t) \geq 0$ and $\frac{\chi}{P_i^F} > 0$,

$$Qw_i^{max}(t) \leq d_i \frac{P_i^F}{\chi}.$$

Accordingly,

$$\overline{Qw_i} \leq E[Qw_i^{max}(t)] \leq E[d_i \frac{P_i^F}{\chi}] \leq d_i^{max} \frac{P_i^F}{\chi}.$$

Since $d_i^{max} < \infty$, we have $\overline{Qw_i} < \infty$.

Therefore, it is easy to obtain that

$$\overline{Q}_i = \frac{\overline{Qw_i}}{\overline{S_i}} \leq \frac{d_i^{max} P_i^F}{\chi \overline{S_i}} < \infty, \tag{41}$$

where $\overline{Q}_i$ follows Eq. (24).

Case 2: When the job was computed in a neighbor edge node, $I_{NF} = 1$. Then, we have

$$(D_{comm(j,i)} + D_{comp}^i(t))^{max} \leq d_i,$$

Since $D_{comm(j,i)} > 0$, we have

$$(D_{comp}^i(t))^{max} < d_i.$$

Substituting $(D_{comp}^i(t))^{max}$ with Eqs. (8)-(9), we have

$$\left((Qw_i(t) + Yw_i(t))\frac{\chi}{P_i^F}\right)^{max} < d_i.$$

Therefore, similar to the results in **Case 1**, $\overline{Qw_i} < \infty$ and $\overline{Q}_i < \infty$ hold.

According to the definition of the stability of an edge server in Eqs. (22)-(25), the edge computing subsystems of the IoT-edge-cloud system are stable. Thus, the statement follows. □

Lemma 3: Under a stable edge computing subsystem i for $i \in \mathcal{M}$, $E[Yw_i(t)] \leq \frac{P_i^F}{\chi}$ holds.

Proof: According to Eq. (6), we have

$$Qw_i(t+1) \geq Qw_i(t) + Yw_i(t) - \frac{P_i^F}{\chi}.$$

Therefore,

$$Yw_i(t) \leq Qw_i(t+1) - Qw_i(t) + \frac{P_i^F}{\chi}.$$

Summing the above over the first t slots and using the fact that $Qw_i(0) = 0$, we have

$$\sum_{\tau=0}^{t-1} Yw_i(\tau) \leq Qw_i(t) + t\frac{P_i^F}{\chi}.$$

Dividing by t and taking expectations, then

$$\frac{1}{t}\sum_{\tau=0}^{t-1} Yw_i(\tau) \leq \frac{E[Qw_i(t)]}{t} + \frac{P_i^F}{\chi}.$$

Since $\overline{Qw_i} = \lim_{t\to\infty} E[Qw_i(t)] < \infty$ according to the definition of the stability of an edge server in Eq. (25), $\lim_{t\to\infty} \frac{E[Qw_i(t)]}{t} = 0$ holds.

Accordingly,

$$E[Yw_i(t)] = \lim_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1} Yw_i(\tau) \leq \frac{P_i^F}{\chi}.$$

The statement then follows. □

Theorem 2: Suppose all queues are initially empty. Then, under DBWA, the achieved energy consumption Pw^* of the IoT-edge-cloud system satisfies

$$Pw^* \leq Pw^{**} + \frac{B}{V}, \tag{42}$$

where Pw^* follows Eq. (20), B is the constant from Lemma 1, V is a nonnegative control parameter from Lemma 2, and Pw^{**} is the optimal energy consumption, defined as the infimum energy consumption for the problem described in Eqs. (20)-(23).

Proof: Let $\gamma^*(t) = (\gamma_0^*(t), \cdots, \gamma_i^*(t), \cdots, \gamma_{M-1}^*(t))$ be the optimal decisions given by the DBWA scheme described in *Algorithm 2*. Let $\gamma^{**}(t) = (\gamma_0^{**}(t), \cdots, \gamma_i^{**}(t), \cdots, \gamma_{M-1}^{**}(t))$ represent the optimal decisions that achieve Pw^{**}.

Since $\gamma_i^*(t)$ satisfies Eq. (34) for every slot t, according to *Lemma 2*, we have

$$\begin{aligned}
&\Delta L(t) + VPw(\gamma_i^*(t)) \\
&\leq B + VPw(\gamma_i^*(t)) + \sum_{i\in\mathcal{M}} Qw_i(t)\left(Yw_i(\gamma_i^*(t)) - \frac{P_i^F}{\chi}\right) \\
&\leq B + VPw(\gamma_i^{**}(t)) + \sum_{i\in\mathcal{M}} Qw_i(t)\left(Yw_i(\gamma_i^{**}(t)) - \frac{P_i^F}{\chi}\right)
\end{aligned} \tag{43}$$

where the last inequality follows because $\gamma_i^*(t)$ under the DBWA algorithm minimizes the preceding expression over all other feasible policies, including $\gamma_i^{**}(t)$.

Taking expectations of the above inequality, we have

$$\begin{aligned}
&E[\Delta L(t) + VPw(\gamma_i^*(t))] \\
&\quad \leq B + VE[Pw(\gamma_i^{**}(t))] \\
&\qquad + \sum_{i\in\mathcal{M}} E[Qw_i(t)]E\left[Yw_i(\gamma_i^{**}(t)) - \frac{P_i^F}{\chi}\right]
\end{aligned} \tag{44}$$

Since $\gamma_i^{**}(t)$ is the optimal decision under a stable edge computing subsystem, according to *Lemma 3*, we have $E\left[Yw_i(\gamma_i^{**}(t)) - \frac{P_i^F}{\chi}\right] \leq 0$.

Accordingly, Eq. (44) becomes

$$\begin{aligned}
E[\Delta L(t) + VPw(\gamma_i^*(t))] &\leq B + VE[Pw(\gamma_i^{**}(t))] \\
&= B + VPw^{**},
\end{aligned}$$

where $Pw^{**} = E[Pw(\gamma_i^{**}(t))]$.

Summing the above over the first t slots and using the fact that $L(0) = 0$, we have

$$\sum_{\tau=0}^{t-1} VE[Pw(\gamma_i^*(\tau))] \leq (B + VPw^{**})t + E[L(t)].$$

Since $E[L(t)] \geq 0$, the right-hand side of the above inequality becomes $(B + VPw^{**})t$.

Dividing by Vt yields the following result, which holds for all $t > 0$.

$$Pw^* = \frac{1}{t}\sum_{\tau=0}^{t-1} E[Pw(\gamma_i^*(\tau))] \leq Pw^{**} + \frac{B}{V},$$

which ends the proof. □

The results in *Theorem 1* indicate that, the DBWA scheme can avoid the potential of extremely long queuing delay for jobs offloading to edge servers by controlling the queue lengths as well as queuing workloads of edge servers. The results in *Theorem 2* illustrate that, the time average expected per-slot energy consumption can be made arbitrarily close to the optimal value Pw^{**} by choosing V suitably large. Equivalently, the energy consumption under DBWA is within $O(1/V)$ of optimality.

V. PERFORMANCE EVALUATION

This section investigates the energy consumption as well as delay performance of the proposed DBWA algorithm. For simplicity but without loss of generality, we consider the scenario with three IoT regions, three edge nodes and a cloud in an IoT-edge-cloud system. It can be extended to more IoT regions and more edge nodes, with the similar results.

TABLE 1. The basic parameter settings.

Parameters	Region 1	Region 2	Region 3
P_i^F (GHz)	2.0	2.0	2.0
Energy factors (a_f, b_f, c_f)	$(0.18 \times 10^{-12}, 0, 0)$		
Neighbor	Region 2	Regions 1,3	Region 2
Job generation rate λ_i $(\sharp(jobs)/ms)$	10	15	10
Mean job size S_i $(Mbits/job)$	0.6	1.2	0.6
P^C (GHz)	3.2		
χ $(cycles/byte)$	2100		
Energy factors (A_c, θ, B_c)	$(1 \times 10^{-28}, 0, 3)$		

Some basic parameters used in the simulation are listed in Table 1. As shown Table 1, we use the poisson distribution with rate vector $\lambda = (10, 15, 10)$ *jobs/ms* to model the job generation rates of the simulated IoT regions 1-3, respectively. The corresponding job sizes follow the exponential distribution with an expected size vector $S = (0.6, 1.2, 0.6)$ *Mbits/job*. We set the computing capability of each of the edge nodes to 2.0 *GHz* CPU frequency, and the computing capability of the cloud is set to 3.2 *GHz* CPU frequency per-VM. Besides, the mean factors of communication delay in the paths of edge node-neighbor edge node and edge node-the cloud are set to 1.5 *ms* and 15 *ms*, respectively. The bandwidth of the local edge-neighbor edge link is set to 54 *Mbps*, while the bandwidth of the local edge-core network and the core-core paths are both set to 1 *Gbps*. The following results are obtained with a discrete event-based simulator that combines Matlab and C++.

A. ENERGY CONSUMPTION-DELAY TRADEOFF VS. V

We first evaluate the energy consumption-delay tradeoff of the proposed DBWA algorithm with control parameter V (defined in *Lemma 2*)by investigating the enery consumption of the IoT-edge-cloud system and the average e2e delay of all jobs over all time under various V values.

The value of V is varied from 0 to 3000. As shown in Fig. 2, when $V = 0$, the algorithm degenerates into a

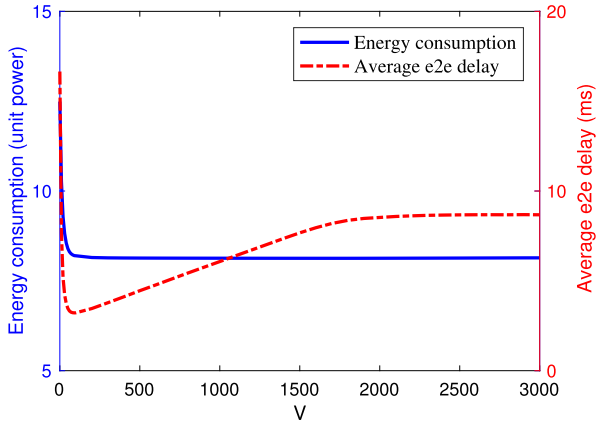

FIGURE 2. Energy consumption-delay tradeoff vs. V.

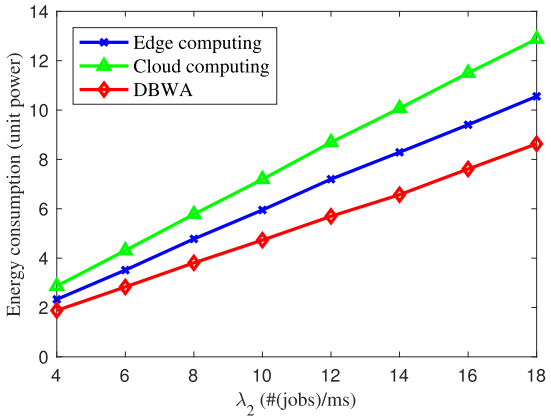

FIGURE 3. Energy consumption vs. job generation rate ($\lambda_1 = \lambda_3 = 0.5\lambda_2$).

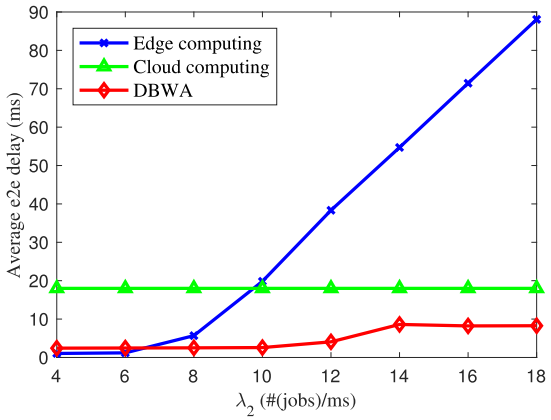

FIGURE 4. Average e2e delay vs. job generation rate ($\lambda_1 = \lambda_3 = 0.5\lambda_2$).

workload-based Lyapunov workload allocation approach. Therefore, it is unsurprising that both energy consumption and average e2e delay under $V = 0$ are the highest in comparison with those under $V > 0$. When $V > 0$, the average e2e delay first decreases with the increasing value of V when V is small (e.g, $V < 100$), and then increases when V is in some range (e.g., $100 < V < 1800$). Different from the results of the average e2e delay under various V, the energy consumption always decreases but slowly with the increasing value of V. However, when V is large enough, the energy consumption-delay tradeoff can reach a balance, e.g., $V \geq 1800$ as shown in Fig. 2.

B. VARIOUS JOB GENERATION RATES

To evaluate the efficiency of the proposed DBWA algorithm for energy efficiency and delay guarantee, we compare the performance of DBWA with the pure edge computing and the pure cloud computing algorithms under various job generation rates. In the pure edge computing algorithm, all jobs are computing in the local edge nodes. In the pure cloud computing algorithm, all jobs are offloaded to the cloud for computing.

We set $\lambda_1 = \lambda_3 = 0.5\lambda_2$, that is, the job generation rates in IoT regions 1 and 3 are half of that in IoT region 2, respectively. Then, we vary the job generation rate of IoT region 2 from 4 *jobs/ms* to 18 *jobs/ms*, to evaluate how they affect the energy consumption and delay performance of the investigated algorithms. We also set the maximum tolerable e2e delays for the jobs in the simulated regions 1-3 to $d = (20, 20, 20)$ *ms*, respectively.

As shown in Fig. 3, the energy consumption given by the pure cloud computing algorithm is the highest under various job generation rates. This is because, although the computation energy consumption in the cloud may be less than that in the edge for the same computation amount, the transmission energy consumption in the paths from edge nodes to the cloud cannot be ignored. The DBWA algorithm can make optimal offloading decisions among the local edge node, neighbor edge nodes and the cloud based on Lyapunov drift-plus-penalty for minimizing the total energy consumption. Therefore, it is unsurprising to see that the energy consumption given by the proposed DBWA algorithm is always the lowest under various job generation rates in comparison with the pure edge computing and cloud computing algorithms.

Due to the computation resource constraint, the queuing delay under edge computing increases exponentially with the increasing job generation rate. Thus, it is unsurprising to see that the average e2e delay given by edge computing increases exponentially with the increasing job generation rate, as shown in Fig. 4. Since the cloud resource is unlimited in comparison with edge nodes, every arrival computation job can be served immediately in the cloud. The e2e delay for cloud computing is mainly affected by the transmission delay. Therefore, the average e2e delay given by the cloud computing algorithm approximates a stable value (e.g., 20 *ms*) when the job transmission rate does not exceed the bandwidth (e.g., $\lambda < bw_{(edgenode,cloud)}$), as shown in Fig. 4.

Since the DBWA algorithm can adaptively switch among local edge node, neighbor edge nodes and the cloud for workload allocation, it provides the lowest average e2e delay under various job generation rates in comparison with the edge and cloud computing algorithms. When the job generation rate is low, e.g., $\lambda_2 \leq 8$ *jobs/ms*, edge computing dominates cloud computing for average e2e delay provisioning, thus

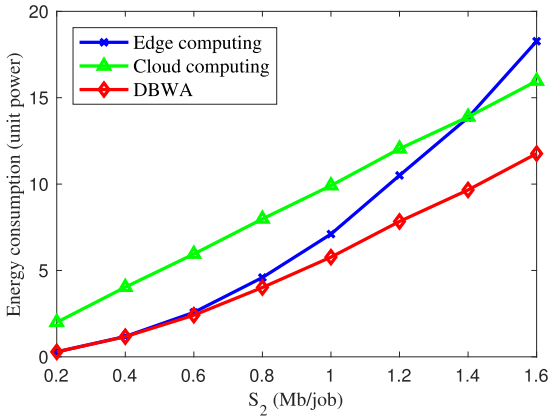

FIGURE 5. Energy consumption vs. job size ($S_1 = S_3 = 0.5S_2$).

most workload are allocated to edge nodes under the DBWA algorithm. Accordingly, it provides the average e2e delay similar to that provided by the edge computing algorithm, as shown in Fig. 4. When the job generation rate becomes large, e.g., $\lambda_2 > 10$ *jobs/ms*, the queuing delay under edge computing increases exponentially, such that cloud computing dominates edge computing. Therefore, under the DBWA algorithm, the workload allocated to the cloud increases adaptively to the job generation rate. Thus, the average e2e delay under DBWA can always be lower than that under the cloud computing algorithm, as shown in Fig. 4.

C. VARIOUS JOB SIZES

We further evaluate the efficiency of the proposal for energy efficiency and delay guarantee by investigating the performance in terms of energy consumption and average e2e delay in comparison with the pure edge computing and cloud computing algorithms under various job sizes.

We adopt the same traffic parameter settings as listed in Table 1. We set $S_1 = S_3 = 0.5S_2$, that is, the expected job sizes in IoT regions 1 and 3 are half of that in IoT region 2, respectively. Then we vary the expected job size of IoT region 2 from 0.2 *Mb/job* to 1.6 *Mb/job*, to evaluate how they affect the energy consumption and delay performance of the investigated algorithms.

As shown in Fig. 5, the energy consumption given by all the investigated algorithms increase with the increasing job sizes. The energy consumption given by edge computing increases the fastest in comparison with the other two algorithms. Particularly, when $S_2 \geq 1.4$ *Mb/job*, the energy consumption given by edge computing exceeds that given by cloud computing. This is because, the energy consumption model of an edge node is a convex function of the job size, while the energy consumption model of the cloud is a linear function of the job size, as discussed in Section III-D. Accordingly, when the job size is large enough, e.g., $S_2 \geq 1.4$ *Mb/job*, the energy consumption given by edge computing would exceed that given by cloud computing.

Since the DBWA algorithm can dynamically switch among local edge node, neighbor edge nodes and the cloud adaptive to job sizes for minimizing the energy consumption, it consumes the lowest energy under various job sizes in comparison with the edge and cloud computing algorithms.

The queuing delay under edge computing increases explicitly with the increasing job sizes, thus the average e2e delay given by the edge computing algorithm increases the fastest in comparison with the other two schemes. Therefore, it is unsurprising to see that, the average e2e delay under edge computing exceeds that under cloud computing when the job sizes exceed some value, e.g., $S_2 > 0.8$ *Mb/job*, as shown in Fig. 6. The DBWA algorithm always provides the lowest average e2e delay under various job sizes in comparison with the other two algorithms, as show in Fig. 6. This is because, DBWA can dynamically switch among local edge node, neighbor edge nodes and the cloud adaptive to the varying of job sizes for delay guarantee.

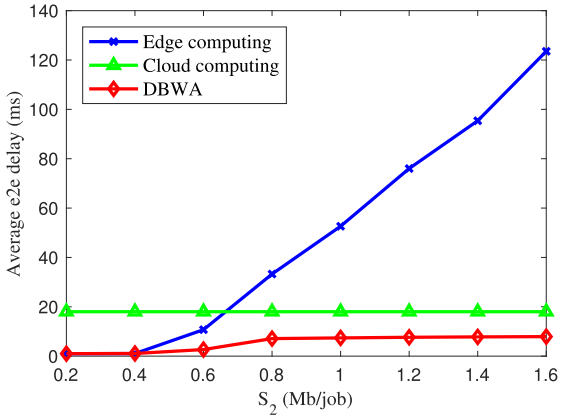

FIGURE 6. Average e2e delay vs. job size ($S_1 = S_3 = 0.5S_2$).

D. GOODPUT

Finally, we evaluate the efficiency of the proposal for per-job granular delay guarantee by investigating the delay performance in terms of goodput in comparison with the edge computing and cloud computing algorithms under scenarios of various job generation rates and job sizes simulated in Sections 5.2 and 5.3.

The goodput reflects the delay-guaranteed efficiency of a workload allocation algorithm in per-job granularity, which is defined as follows.

$$G = \frac{\sharp \text{ of jobs with job's e2e delay} \leq \text{job's delay bound}}{\text{total } \sharp \text{ of jobs arrival}} \times 100\% = \frac{\sum_{t=0}^{\infty} \sum_{i \in \mathcal{M}} \sum_{k=0}^{X_i(t)-1} I(D^{(i,k)}(t) \leq d^{(i,k)})}{\sum_{t=0}^{\infty} \sum_{i \in \mathcal{M}} X_i(t)} \times 100\%$$

where $I(k) = 1$ if k is true; $I(k) = 0$ otherwise.

The simulation results are summarized in Table 2. As shown in Table 2, the goodput given by edge computing decreases explicitly with the increasing job generation rate. Particularly, when λ_2 increases from 8 *jobs/ms* to 10 *jobs/ms*, the goodput decreases dramatically, e.g., from 93.88% to 63.58%. However, since that the cloud computing

TABLE 2. Goodput.

Job generation rate (λ_2 (♯ *(jobs)/ms*))	Goodput(%)			Job size (S_2 *(Mb/job)*)	Goodput(%)		
	Edge computing	Cloud Computing	DBWA		Edge computing	Cloud Computing	DBWA
4	100	100	100	0.2	100	100	100
6	100	100	100	0.4	100	100	100
8	93.88	100	100	0.6	76.49	100	100
10	63.58	100	100	0.8	62.85	100	100
12	56.95	100	100	1.0	60.60	100	100
14	54.59	100	100	1.2	59.81	100	100
16	53.55	100	100	1.4	59.20	100	100
18	52.99	100	100	1.6	53.51	100	100

resource is unlimited compared to an edge node, and that the total transmission rate (λS) does not exceed the bandwidth of the paths from edge nodes to the cloud, the goodput given by the cloud could reach 100%. Since DBWA dynamically adapts to the varying of job generation rate, it also bounds the goodput to 100% under various job generation rates, as shown in Table 2. Similar results are obtained under the scenario of varying job sizes. As shown in Table 2, the goodput given by edge computing decreases explicitly with the increasing job size, while the goodput under both of cloud computing and DBWA can be bounded to 100%. The simulation results in Table 2 illustrate the delay guarantee efficiency of the proposed DBWA algorithm.

VI. CONCLUSIONS

This paper has studied the energy-efficient and delay-guaranteed workload allocation problem in an IoT-edge-cloud computing system. We develop a systematic framework, including system, traffic, delay and energy consumption models, to investigate the issue of energy consumption minimization constrained to per-job granular delay guarantee in an IoT-edge-cloud computing system. We have formulated the workload allocation problem and developed a delay-based workload allocation scheme, e.g., DBWA scheme, to solve the problem. Specifically, the Lyapunov drift-plus-penalty properties of the queuing systems of edge servers are analyzed. Then, the DBWA scheme is proposed to minimize the drift-plus-penalty, for achieving the goal of minimizing the energy consumption of the system while provisioning per-job granular delay guarantee. The theoretical analysis and simulation results have illustrated the efficiency of the proposal in provisioning the energy efficiency and delay guarantee.

REFERENCES

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

[2] P. Ren, X. Qiao, J. Chen, and S. Dustdar, "Mobile edge computing—A booster for the practical provisioning approach of Web-based augmented reality," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Oct. 2018, pp. 349–350.

[3] A. Al-Shuwaili and O. Simeone, "Energy-efficient resource allocation for mobile edge computing-based augmented reality applications," *IEEE Wireless Commun. Lett.*, vol. 6, no. 3, pp. 398–401, Jun. 2017.

[4] A. V. Dastjerdi and R. Buyya, "Fog computing: Helping the Internet of Things realize its potential," *Computer*, vol. 49, no. 8, pp. 112–116, Aug. 2016.

[5] Z. Ning, X. Kong, F. Xia, W. Hou, and X. Wang, "Green and sustainable cloud of things: Enabling collaborative edge computing," *IEEE Commun. Mag.*, vol. 57, no. 1, pp. 72–78, Jan. 2019.

[6] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, Oct./Dec. 2009.

[7] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. ACM 1st Ed. MCC Workshop Mobile Cloud Comput. (MCC)*, New York, NY, USA, 2012, pp. 13–16.

[8] M. Patel, B. Naughton, C. Chan, N. Sprecher, S. Abeta, and A. Neal, "Mobile-edge computing introductory technical white paper," Mobile-Edge Comput. Ind. Initiative, Sophia-Antipolis, France, White Paper 1, 2014, pp. 1–36.

[9] L. Tong, Y. Li, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *Proc. IEEE INFOCOM 35th Annu. Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9.

[10] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, "A survey on the edge computing for the Internet of Things," *IEEE Access*, vol. 6, pp. 6900–6919, 2018.

[11] W. Shi and S. Dustdar, "The promise of edge computing," *Computer*, vol. 49, no. 5, pp. 78–81, 2016.

[12] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[13] T. H. Luan, L. Gao, Z. Li, Y. Xiang, G. Wei, and L. Sun, "Fog computing: Focusing on mobile users at the edge," *CoRR*, vol. abs/1502.01815, pp. 1–11, Feb. 2015.

[14] R. K. Naha, S. Garg, D. Georgakopoulos, P. P. Jayaraman, L. Gao, Y. Xiang, and R. Ranjan, "Fog computing: Survey of trends, architectures, requirements, and research directions," *IEEE Access*, vol. 6, pp. 47980–48009, 2018.

[15] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.

[16] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018.

[17] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[18] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2016, pp. 1451–1455.

[19] L. Yang, B. Liu, J. Cao, Y. Sahni, and Z. Wang, "Joint computation partitioning and resource allocation for latency sensitive applications in mobile edge clouds," in *Proc. IEEE 10th Int. Conf. Cloud Comput. (CLOUD)*, Jun. 2017, pp. 246–253.

[20] A. Yousefpour, G. Ishigaki, and J. P. Jue, "Fog computing: Towards minimizing delay in the Internet of Things," in *Proc. IEEE Int. Conf. Edge Comput. (EDGE)*, Jun. 2017, pp. 17–24.

[21] J. Liu and Q. Zhang, "Offloading schemes in mobile edge computing for ultra-reliable low latency communications," *IEEE Access*, vol. 6, pp. 12825–12837, 2018.

[22] H. Zhang, Y. Xiao, S. Bu, D. Niyato, F. R. Yu, and Z. Han, "Computing resource allocation in three-tier IoT fog networks: A joint optimization approach combining Stackelberg game and matching," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1204–1215, Oct. 2017.

[23] L. Li, Q. Guan, L. Jin, and M. Guo, "Resource allocation and task offloading for heterogeneous real-time tasks with uncertain duration time in a fog queueing system," *IEEE Access*, vol. 7, pp. 9912–9925, 2019.

[24] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 17, no. 3, pp. 1784–1797, Mar. 2018.
[25] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 1171–1181, Dec. 2016.
[26] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan, S. Maharjan, and Y. Zhang, "Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks," *IEEE Access*, vol. 4, pp. 5896–5907, 2016.
[27] D. Van Le and C.-K. Tham, "A deep reinforcement learning based offloading scheme in ad-hoc mobile clouds," in *Proc. IEEE INFOCOM Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2018, pp. 760–765.
[28] X. Lyu, H. Tian, L. Jiang, A. Vinel, S. Maharjan, S. Gjessing, and Y. Zhang, "Selective offloading in mobile edge computing for the green Internet of Things," *IEEE Netw.*, vol. 32, no. 1, pp. 54–60, Jan./Feb. 2018.
[29] Y. Mao, J. Zhang, Z. Chen, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.
[30] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proc. 2nd USENIX Conf. Hot Topics Cloud Comput. (HotCloud)*, Berkeley, CA, USA: USENIX Association, 2010, pp. 1–7.
[31] M. J. Neely, *Stochastic Network Optimization With Application to Communication and Queueing Systems*. San Rafael, CA, USA: Morgan & Claypool, 2010.
[32] S. P. Meyn and R. L. Tweedie, "Stability of Markovian processes III: Foster–Lyapunov criteria for continuous-time processes," *Adv. Appl. Probab.*, vol. 25, pp. 518–548, Sep. 1993.

MIAN GUO (S'11–M'13) received the Ph.D. degree in communication and information systems from the South China University of Technology, China, in 2012. She was a Visiting Professor with the University of Ottawa, Ottawa, ON, Canada, in 2016, and a Visiting Professor with Beihang University, China, in 2017. She is currently with the Guangdong University of Petrochemical Technology, China. Her research interests include resource allocation, QoS provisioning in computer and communication networks, big data applications, fog/edge computing, and reinforcement learning.

LEI LI received the B.S. and M.S. degrees from the School of Electronic and Information Engineering, South China University of Technology (SCUT), in 2006 and 2010, respectively, where he is currently pursuing the Ph.D. degree with the School of Electronic and Information Engineering. He is currently a Senior Experimentalist. His research interests include cloud computing, optimal control algorithm, performance evaluation, machine learning, and green computing systems. He has received the Guangdong Science and Technology Award, in 2018. He was a co-recipient of the Best Paper Award from the China Cloud Computing Conference (CCCC) 2016.

QUANSHENG GUAN (S'09–M'11) received the B.Eng. degree in electronic engineering from the Nanjing University of Post and Telecommunications, China, in 2006, and the Ph.D. degree from the South China University of Technology (SCUT), in 2011.

From 2009 to 2010, he was a Visiting Ph.D. Student with the University of British Columbia, Canada. From 2012 to 2013, he was a Postdoctoral Researcher with The Chinese University of Hong Kong. He was a Visiting Scholar with the Singapore University of Technology and Design, in 2013, and a Visiting Professor in Polytech Nantes, France. He is currently a Professor with the School of Electronic and Information Engineering, SCUT. His main research interests include the areas of wireless networks, underwater acoustic networks, and network games and economics. He was a co-recipient of the Best Paper Award from the IEEE ICCC 2014 and the IEEE ICNC 2016. He is a Guest Editor of Mobile Information System. He is a frequent TPC member of conferences and a frequent Reviewer of journals and conferences.