Multiple Adaptive-Resource-Allocation Real-Time Supervisor (MARS) for Elastic IIoT Hybrid Cloud Services

Younghwan Shin, Wonsik Yang, Sangdo Kim, and Jong-Moon Chung, *Senior Member, IEEE*

***Abstract*—In this paper, a Multiple Adaptive-resource-allocation Real-time Supervisor (MARS) scheme for hybrid cloud-assisted Industrial Internet of Things (IIoT) is proposed to support reliable cloud services even under rapidly changing service demands that occur in massive IIoT networks. Virtual Machines (VMs) in both private cloud and public clouds can be elastically and accurately allocated through the proposed MARS scheme, which uses Karush-Kuhn-Tucker (KKT) optimization applied to the VM Continuous-Time Markov Chain (CTMC) scheme. Because the MARS scheme can immediately determine the optimal number of VMs based on the hybrid cloud situation, a significant improvement in the elasticity performance can be obtained. Compared to using the CTMC scheme, the results show that the MARS scheme can improve the response time up to 19.3 $\sim$ 73% (based on the activation rate) and the elasticity by 26.7%, and reduce the cost by 1.2%.**

***Index Terms*—Private cloud, public cloud, elastic service, optimization.**

I. Introduction

INTERNET connectivity is becoming more complex and complicated as various massive Machine Type Communications (mMTCs) and computationally-intensive Internet of Things (IoT) applications are now prevalent [1], [2]. Rapid growth of the scale of IoT networks is also no exception to the industrial domain [3]. For the year 2020, the global market value of Industrial IoT (IIoT) is $82.4 Billion U.S. dollars and a 21.3% Compounded Annual Growth Rate (CAGR) is forecasted for the period of 2020 to 2028 [4]. In addition, estimations have been made that the number of massive IoT connections have doubled to reach an approximate 200 million connections during 2020. Even considering the 2019 Coronavirus disease (COVID-19) pandemic influence, it is predicted that broadband IoT services will consume about 44% of all cellular IoT connections by the end of 2026 [5].

One of the more notable things about IIoT is that the number of IoT devices and network can become even larger and more complicated due to the industry infrastructure, and therefore, the IIoT network may operate slower and may be more vulnerable to security threats [6]. IIoT has been emerging as an prominent technology that has the potential to change many industrial services, which include renewable energy sources (e.g., solar), smart power grids, smart cities, Intelligent Building System (IBS), Intelligent Transportation Systems (ITS), security systems, eXtended Reality (XR) facility remote management, distributed blockchain databases, smart office equipment, as well as smart resource control for water, gas, electricity, street lamps, and personal mobility systems [7], [8]. These systems experience significant changes in utilization patterns based on the time and location. Therefore, the data exchanged across massive IIoT networks supporting these systems will show large changes in significant scales. Because massive IIoT networks can generate large amounts of data in irregular and bursty patterns, cloud computing support is needed to provide data storage and computational processing. Cloud support for massive IIoT networks is essential because common IIoT devices have very small memory and limited processing capabilities [6], [9]. For this reason, in recent years, numerous studies have attempted to overcome various issues of massive IIoT networks by proposing cloud support [10]. Using a private cloud close to the IIoT devices can provide many benefits, such as, the latency can be reduced along with having the advantages of privacy and more protection. However, for a massive IIoT network covering a wide region, a hybrid cloud consisting of a private cloud and public cloud would be needed because the public cloud would be able to provide much more processing capability and storage space, as well as enable easier access and more diverse connectivity. Therefore, more consideration needs to be given to hybrid clouds when trying to support massive IIoT networks.

One of the most important features for a hybrid cloud that is supporting a massive IIoT network is the cloud's on-demand resource allocation capability, which include computational resources, such as, Virtual Machines (VMs) or containers. On-demand resource allocation has always been one of the key research topics in cloud computing [11]. When it comes to hybrid cloud-assisted IIoT, on-demand resource allocation becomes a more crucial and difficult issue to handle. The dynamic and massive scale of IIoT networks makes it difficult to predict how much computational resources will be needed [12]. Moreover, it is difficult to accurately and elastically allocate resources in large scale hybrid clouds in real-time. In order to

Manuscript received March 27, 2021; revised November 8, 2021; accepted January 14, 2022. Date of publication January 25, 2022; date of current version May 23, 2022. This work was supported in part by the Ministry of Science and ICT under Grant D0318-21-1008 and in part by the National Fire Agency of the Republic of Korea under Grant 20017102. Recommended for acceptance by Prof. Falko Dressler. *(Corresponding author: Jong-Moon Chung.)*

The authors are with the School of Electrical and Electronic Engineering, Yonsei University, Seoul 03722, South Korea (e-mail: shinyh1115@yonsei.ac.kr; yangws95@yonsei.ac.kr; sangdokim96@yonsei.ac.kr; jmc@yonsei.ac.kr).

Digital Object Identifier 10.1109/TNSE.2022.3145876

overcome this limitation, in this paper, an accurate and elastic resource allocation technology that is called the Multiple Adaptive-resource-allocation Real-time Supervisor (MARS) scheme is proposed. The MARS scheme can instantaneously calculate the optimal number of VMs to allocate in the private and public clouds based on real-time demand changes in the massive IIoT network. This is a significant improvement compared to the currently most advanced Continuous-Time Markov Chain (CTMC) scheme used in cloud computing, which determines the number of required VMs based on gradually increasing the number of VMs until the performance conditions are satisfied. Since the MARS scheme can instantaneously inform the optimal number of VMs required in the private and public clouds, resource allocation can be conducted immediately, saving significant performance losses compared to when the optimal number of VMs is unknown and searched progressively.

For cloud computing systems, accuracy and elasticity are two major Service-Level Objectives (SLOs) that make on-demand resource allocation feasible. Accuracy is defined as the degree to which computational resources can be precisely allocated as much as users require. Elasticity is defined as how fast a cloud system can adapt to changes based on users' demand [13]–[16]. In [13], elasticity is defined as the ability of a system to add and remove resources (e.g., CPU/GPU cores, memory, VMs and container instances) to adapt to the load variation in real time. In addition, in [14], elasticity is defined as the capability to allow users to dynamically acquire and release the proper amount of computing resources according to their needs. The CTMC scheme is acknowledged to be one of the best feasible solutions to achieve satisfactory accuracy in cloud computing. The CTMC scheme provides a stable mechanism to assign the required amount of resources (e.g., VMs) to the cloud. Including [17]–[21], many CTMC schemes have been proposed so far. Each proposed CTMC scheme achieves accurate resource allocation. However, the CMTC base schemes proposed in these papers do not consider elasticity, and thus have limitations in terms of their elasticity performance. This is mainly because the CTMC scheme is based on increasing or decreasing only one computational resource (e.g., VM) at a time. Thus, CTMC based cloud services are suitable in accurately supporting gradually changing cloud service demands. However, for massive IIoT networks that can have significant changes in services demands, the CTMC scheme is slow in making resource allocation adaptations. This is even a more significant issues when it comes to hybrid cloud services, where resource allocation changes in the private and public cloud need to be adjusted quickly, simultaneously, and cooperatively.

Therefore, in this paper, a highly elastic and accurate resource allocation scheme is proposed to satisfy dynamically changing real-time hybrid cloud service requirements. The proposed MARS scheme significantly improves the elasticity performance of the CTMC scheme, and extends the CTMC scheme to real-time hybrid cloud control. The MARS scheme uses Karush-Kuhn-Tucker (KKT) conditions to derive the minimal upper bound of the optimal number of VMs that are needed. The KKT solution makes it possible to deploy the near-optimal amount of VMs instantaneously, which significantly helps to improve the elasticity. It is worth noting that no research has provided a closed-form resource allocation solution of the CTMC approach so far, which is a unique feature of this paper.

The MARS scheme builds on the CTMC scheme proposed in [22]. Along with the CTMC scheme, the authors of [22] provide a way to analyze elasticity in any cloud platform by presenting a quantifiable and calculable definition, which is applied in this paper. A comparable technique is the Moving Average (MA) scheme of [23], which uses the elasticity metric of [22] in its performance analysis. The results show that the proactive buffer management scheme of [23] can improve the elasticity performance when compared to using the CTMC scheme of [26]. However, these schemes are not suitable to deal with the dynamically changing bursty conditions, which will be demonstrated in the simulation results later in this paper. On the contrary, the elasticity of the MARS scheme is sufficient to handle bursty changing conditions.

In addition, most existing papers regarding elasticity, including the above-mentioned papers (that deal with CTMC based techniques), fail to consider hybrid cloud support. In order to achieve accurate, elastic, and cost-effective resource allocation, a hybrid cloud architecture is more suitable for many applications [24]. Hybrid cloud-assisted IIoT can offer variant decision options to users. For massive IIoT networks, hybrid cloud systems are an attractive option for the sake of security issues and access control [25]. In this regard, the proposed MARS scheme focuses on providing instantaneous and optimized VM resource allocation for hybrid clouds.

In the view of taxonomy of elastic cloud computing, the KKT optimization procedures of MARS operates in proactive mode and the CTMC procedures of MARS operates in reactive mode [13], [14], which is why MARS can be considered as a mixed mode hybrid cloud control scheme.

In summary, the proposed MARS scheme was designed to accomplish the objective of supporting a massive IIoT network with elasticity and accuracy to quickly handle bursty service demand changes. The contributions of this paper are as follows.

1) Existing papers do not provide a closed form solution to optimized resource allocation in hybrid clouds. Considering that hybrid clouds are very common and essential for massive IIoT networks, the proposed MARS scheme can be very useful.
2) The MARS scheme uses optimal KKT solutions to provide the optimal number of VMs needed in the private and public clouds instantaneously, which helps to improve the elasticity and accuracy performance in hybrid cloud resource allocation thereby minimizing cost and time delay.

II. System Environment

A. General Concept of the Scheme

Hybrid clouds are made up of public clouds and a private cloud, where data traffic is distributed throughout the private and public clouds. It is assumed that the average distribution ratio is known, where the data traffic arrival rate from the IIoT devices through the private cloud to the public clouds are

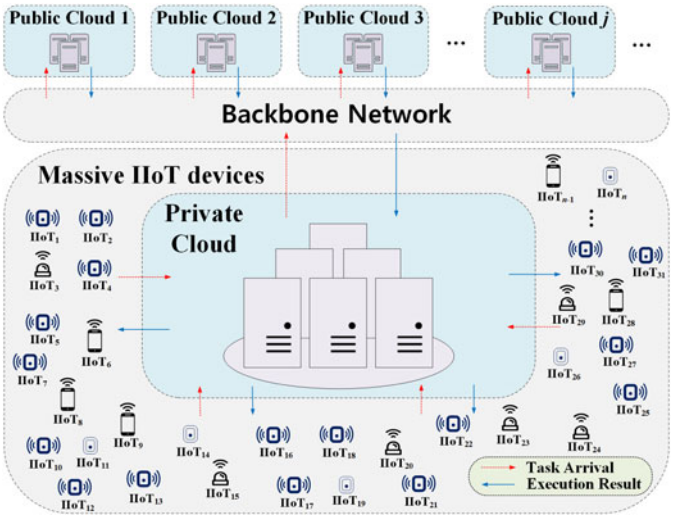

Fig. 1. Hybrid cloud-assisted IIoT architecture.

known, based on the IIoT network architecture presented in Fig. 1. In Fig. 1, the hybrid cloud supports massive tasks from *n* IoT devices. The private cloud takes charge of dealing with tasks in advance, and distributes them to the public clouds. According to the amount of tasks, the MARS scheme allocates the appropriate amount of VMs to the private cloud and the public clouds.

Although an external broker or internal broker can be used in a hybrid cloud [27], this paper does not adopt an external broker for the following reasons. Generally, a broker handles tasks through the private cloud rather than the public clouds, since use of the public cloud inevitably generates network delay and additional networking costs if there is no significant performance difference between a public and private cloud [28].

In order to enhance the stability of the hybrid cloud services, risk management needs to be considered [29], where the data may need to be pre-processed by the private cloud before it is sent to the public cloud. There are various reasons for the pre-processing. For instance, confidential data may need to be encrypted on the private cloud for security issues [30], a task may demand access to a database in the private cloud where principal or confidential data is stored [31], and data formats may be different between the public and private clouds where reformatting is needed. Due to these reasons, pre-processing data at the private cloud is very common in cloud computing [32], [33]. This type of risk management is called *access control* [34]. Many papers exploited access control for identity management [35]–[37]. This way of handling dataflows is called *automigration* where workloads are automatically (or manually) transferred between clouds using well-defined protocols [38].

For these reasons, in this hybrid cloud model, the private cloud receives data in advance and acts as a broker to allocate tasks to the clouds. Even if this model is modified to use an external broker, there will be no notable difference in the operational algorithmic procedures or in the mathematical optimization point of view, which is why the proposed MARS scheme can be applied the same way.

TABLE I
SUMMARY OF PARAMETERS

Parameter	Definition
n	Number of IIoT devices
M	Number of public clouds
λ	Mean arrival rate
ϕ_0	Purchasing cost per VM
ϕ_j	Sum of rental cost and static power consumption cost
$\psi_G \mu_G^{d_G}$	Power consumption for processing groundwork
$\psi_0 \mu_0^{d_0}$	Power consumption for data processing in private cloud
$\psi_j \mu_j^{d_j}$	Power consumption for data processing in public cloud j
Γ_0	Cost coefficient of private cloud
Γ_j	Cost coefficient of public cloud j
$\mathbb{C}_{pri}$	Total cost of private cloud
$\mathbb{C}_{pub}$	Total cost of public cloud j
$\mathbb{C}_{tot}$	Total cost of all clouds
α_j	DTC from private cloud to public cloud j
β_j	DTC from public cloud j to private cloud
μ_0	Service rate of private cloud
μ_j	Service rate of public cloud j
μ_G	Service rate of groundwork
ω_j	Data distribution rate for public cloud j
T_{equip}	Equipment delay
T_{prop}	Propagation delay
T_{trans}	Transmission delay
T_{0j}	Transmission delay from private cloud to public cloud j
T_{j0}	Transmission delay from public cloud j to private cloud
τ_{oj}	Network delay from private cloud to public cloud j
τ_{j0}	Network delay from public cloud j to private cloud
τ_N	Total network delay that embodies all the networking
τ_G	Time delay for processing groundwork
τ_j	Time delay for data processing in public cloud j
τ_0	Time delay for data processing in private cloud
N_0	Number of VMs in private cloud
N_j	Number of VMs in public cloud j
L_I	Average data size of an input task from an IIoT device
L_O	Average output data size after being processed
R	Bandwidth
D_0	Time required to process the tasks using private cloud
D_j	Time required to process the tasks using public cloud j
D^*	Delay threshold
C	Erlang's C formula
U_G	Upper bound of the Erlang C formula for groundwork
U_0	Upper bound of the Erlang C formula for private cloud
U_j	Upper bound of the Erlang C formula for public cloud j
$\urcorner_G$	Delay upper bound for τ_G
$\urcorner_0$	Delay upper bound for τ_0
$\urcorner_j$	Delay upper bound for τ_j
S_{normal}	Total time of the system in normal
S_{over}	Total time of the system that is over-provisioned
S_{under}	Total time of the system that is under-provisioned
S_{total}	Total time the cloud computing system is running
E	Elasticity
$\boldsymbol{\pi}$	KKT multiplier vector

B. Modeling Data Traffic and VMs

In the proposed hybrid cloud system model (Table I), there are numerous IIoT devices that request their tasks to be dealt with. The arrival of task X_k from the kth IIoT device can be described by an independent renewal process [39]. Since a superimposed large number of renewal processes can be approximated with a Poisson process by the Palm-Khintchine theorem [40], the total task arrival rate can be expressed using a Poisson distribution with the mean arrival rate λ as follows

$$\sum_{i=1}^{n} X_i \sim Poisson(\lambda) \tag{1}$$

where n denotes the total number of IIoT devices. The service rates of VMs in the jth cloud are assumed to be all the same and follow the exponential distribution with the mean μ_j. Thus, each cloud can be deemed a M/M/c queue. In reference to the cloud service rates μ_j, index $j = 0$ represents the private cloud, and index j (which is a non-zero natural number) refers to the jth public cloud.

Generally, data sizes vary after being processed [41]. Assuming that the average data size of an input task from an IIoT device is L_I, the average output data size L_O can be represented by the Data Traffic Coefficient (DTC) times the size of the input data size, resulting in the relation of (2).

$$L_O = L_I \times DTC \tag{2}$$

Since the DTC of the public and private clouds can be different, the DTC from the private cloud towards the jth public cloud is denoted as α_j and the DTC in the reverse direction is denoted as β_j.

The input data stream is appropriately distributed by the broker to the clouds with the distribution ratio set of $W = \left\{\omega_j \in \mathbb{R} | 0 \le \omega_j \le 1, \sum_{\forall j \in J} \omega_j = 1\right\}$, where J is the set of all public clouds.

C. Cost Model

There are numerous cost types for setting up and managing cloud computing services, such as, electricity, hardware, software, labor fees, business premises, cloud services, and deployment [42]. Each type embodies many cost factors.

The set-up cost for a hybrid cloud is twofold, the cost for the private cloud and the cost for the public clouds. This paper adopts the cost model in [22] for the jth public cloud which is represented as

$$\mathbb{C}_{pub} = N_j\left(\phi_j + \psi_j \mu_j^{d_j}\right) \tag{3}$$

where ϕ_j is the sum of the rental cost and the cost for static power consumption, $\psi_j \mu_j^{d_j}$ is the dynamic power consumption, and N_j is the number of VMs in the jth public cloud. This paper focuses on the very specific subject of how many VMs are needed. The above cost model is adopted among many cost factors and models in this regard. This model embodies the renting cost and energy consumption cost of the public clouds.

As for the private cloud, many aspects of cost are slightly different from the public cloud. For the sake of simplicity, all kinds of cost associated with a private VM are assumed to be linearly dependent on the number of private VMs $\mathbb{C}_{pri}$ as shown in the following

$$\mathbb{C}_{pri} = N_0\left(\phi_0 + \psi_0 \mu_0^{d_0} + \psi_G \mu_G^{d_G}\right) \tag{4}$$

where N_0 is the number of VMs in the private cloud and ϕ_0 is the purchasing cost per VM, which is assumed to have a fixed cost value involved in the purchase of a VM. To be more specific, the purchasing cost for a VM includes the costs of the network device(s), VM software license(s), middleware license(s), and application software license(s). The cost of facility space, non-electronic equipment, and cabling are also included. This approach is well described in [42]. Among many kinds of variable costs, only the energy consumption is directly associated with the VM performance, where $\psi_0 \mu_0^{d_0} + \psi_G \mu_G^{d_G}$ represents the dynamic power consumption amount. The total cost $\mathbb{C}_{tot}$ can be written as

$$\mathbb{C}_{tot} = N_0\left(\phi_0 + \psi_0 \mu_0^{d_0} + \psi_G \mu_G^{d_G}\right) + \sum_{\forall j \in J, j \neq 0} N_j\left(\phi_j + \psi_j \mu_j^{d_j}\right) \tag{5}$$

where $\mathbb{C}_{tot}$ can be neatly sorted out by replacing $(\phi_j + \psi_j \mu_j^{d_j})$ and $(\phi_0 + \psi_0 \mu_0^{d_0} + \psi_G \mu_G^{d_G})$ with Γ_j and Γ_0. As a result,

$$\mathbb{C}_{tot} = N_0 \Gamma_0 + \sum_{\forall j \in J, j \neq 0} N_j \Gamma_j \tag{6}$$

where μ_G is described in the following section.

III. Latency Models

This section provides two latency models. When allocating the optimal number of VMs to each cloud given a time deadline, these two latency models are used to determine whether the time deadline will be breached or not.

Cloud computing cannot avoid generating network delay because some data has to be exchanged between the VMs. This is why network delay has to be considered when designing a cloud computing model, especially one that supports real-time applications. The network delay model in this paper predominantly focuses on data size, because the data size is proportional to the data processing time, number of packets to exchange, network transfer time, and queueing time at the receiver. Based on the representation of network delay in terms of data size, this paper offers an in-depth analysis on how bursty traffic affects network delay, and consequently how many VMs are needed to compensate for the adverse effects of network delay. Network delay consists of four major factors, which are, processing delay, queuing delay, transmission delay, and propagation delay. All of the four kinds of delay arise every time a packet goes through all the routers and receives services from the network management entities [43].

However, it is almost impossible to consider all routers across the Wide Area Network (WAN). Here arises the need for some assumptions to be made. In [44], a combination of packet switching delay, processing delay, and queueing delay are collectively defined as equipment delay. Equipment delay is caused when network processes run on intermediary network devices, such as, switches, routers, and firewalls. Equipment delay is mainly influenced by the network load, congestion, and performance of intermediary network devices. Processing delay and packet switching delay is generated by all intermediary network devices, which can be regarded as a constant value owing to the consistent high-performance level of nowadays network devices [43]. If the backbone network environment is unstable, stable real-time service support based on cloud computing will be impossible to achieve. Therefore, it is assumed that the performance of all intermediary network devices is satisfactory and stable, and the

queues of the intermediary networking systems will not overflow due to the stable utilization conditions. Under this environment, it is assumed that the queuing delays have little variation due to changing traffic conditions, and therefore, the queueing delay can be approximated as a constant value. Based on these considerations, the equipment delay can be denoted by a single constant value represented as $equipment\ delay\ T_{equip}$. The equipment delay of this paper is a collective term referring to all packet switching delays, processing delay, and queueing delays generated by the intermediate network devices.

Propagation delay can be derived from dividing the distance between the sender and the VM by the signal propagation speed. This means that the propagation delay T_{prop} is independent of the data size. On the other hand, the data size has a crucial effect on the transmission delay T_{trans}, which can be derived from $T_{trans} = \frac{L}{R}$, where L is the data size (in bits) and R is the average data rate (in bits per second). Taking everything into consideration, the total network delay is defined as $T_{equip} + T_{prop} + T_{trans}$. Indeed, many studies have demonstrated that transmission delay is a decisive factor for network delay in these environments [45], [46].

A. Latency Model for Public Clouds

In some hybrid architectures, the broker is separated from the private cloud. This type of external broker model can make the workflow very inefficient, especially when a requested task needs to be pre-processed in the private cloud before being sent to the public cloud. As a result, the external broker model commonly is not capable of achieving maximum productivity. If the private cloud does not know which public cloud is busy, a random selection or round-robin selection scheme may be used in choosing the public cloud to use. In some cases, the pre-processed data may be sent back to the broker if a public cloud cannot accept to process it, which generates additional network delay. Needless to say, this will result in an additional networking cost [28].

The cloud model applied in this paper assumes that tasks are required to be pre-processed, and an internal broker is used for this purpose. The two processes, pre-processing and scheduling, are collectively called *groundwork* altogether in this paper.

The service time of the groundwork is assumed to have an exponential distribution with parameter μ_G. In addition, all VMs in the private cloud are assumed to be able to conduct the required groundwork as well as other remaining required tasks to complete the assigned job. Furthermore, the service time for the groundwork is assumed to take relatively less time compared to the service time needed to complete the remaining tasks. This type of an application is quite common. For instance, the Avian flu workflow consists of three procedures, PrepareGPF, AutoGrid, and AutoDock. The first procedure takes only about 120 seconds while the remaining tasks take about 2,040 seconds. As for the Motif workflow, the first procedure Pre-interproscan takes 30 seconds, whereas the remaining tasks takes 9,060 seconds [41]. When it comes to Bag of Tasks (BoT) applications, it is usual that compute-intensive BoTs are outsourced [47].

As mentioned in the previous section, the service time for the remaining task also follows an exponential distribution with mean μ_j. Since the arriving input data traffic follows a Poisson distribution, the groundwork of the private cloud can be modeled as a $M/M/N_0$ queuing model. Hence, the delay for the groundwork τ_G can be expressed as [48]

$$\tau_G = \frac{C\left(N_0, \frac{\lambda}{\mu_G}\right)}{N_0\mu_G - \lambda} + \frac{1}{\mu_G} \tag{7}$$

where $C(N_0, \frac{\lambda}{\mu})$ is the Erlang's C formula defined as follows.

$$C\left(N_0, \frac{\lambda}{\mu_G}\right) = \frac{1}{1 + (1-\rho_G)\left(\frac{N_0!}{(N_0\rho_G)^{N_0}}\right)\sum_{k=0}^{N_0-1}\frac{(N_0\rho_G)^k}{k!}} \tag{8}$$

The VM utilization $\rho_G = \frac{\lambda}{N_0\mu_G}$ should be less than one for (8) to hold true, otherwise the queue will increase infinitely. For this reason, $\rho_G < 1$ is assumed in this paper. This assumption leads to the following inequality.

$$\frac{\lambda}{\mu_G} < N_0 \tag{9}$$

After the groundwork is completed, the private cloud sends pre-processed data to the public cloud(s) if necessary. The private cloud may handle the task without offloading. However, this section focuses on the scenario under which the public cloud(s) are used. The case when only the private cloud is needed to be used will be introduced in the next section. The transmission delay T_{0j} caused by transmitting data from the private cloud to the jth public cloud is as follows.

$$T_{0j} = \frac{L\alpha_j}{R_0} \tag{10}$$

The subscript '$0j$' of T_{0j} refers to the transmission from the private cloud to the jth public cloud. Note that the DTC α_j is multiplied to L as a result of the groundwork. Following the pre-processed data transmission, the backbone network is accessed for a public cloud networking task assignment. As discussed in the previous section, T_{equip} and T_{prop} can be regarded as constant values. The total network delay for transmitting data to the jth public cloud is $T_{equip} + T_{prop} + T_{0j}$, which can be rewritten as (11).

$$\tau_{0j} = T_{equip} + T_{prop} + \frac{L\alpha_j}{R_0} \tag{11}$$

After the networking process, the public cloud processes the remaining tasks. The following lemma deals with the data arrival distribution to the public clouds.

Lemma 1: The arrival of IIoT data into the jth public cloud follows a Poisson distribution with mean $\omega_j\lambda$.

Proof: Based on Burke's Theorem [49], even after the initial IIoT data has been processed through the private cloud, the departure process of the pre-processed data still follows a Poisson distribution with the same mean λ. The pre-processed data is distributed after the departure of the pre-processed

data. At this time, by the thinning property of the Poisson process, alias the Prekopa's theorem [50], data distributed to each cloud forms independent Poisson processes with the respective rate of $\omega_j\lambda$ for the jth public cloud. ■

Based on lemma 1, the arrival rate of the pre-processed data into the jth public cloud follows a Poisson process with $\omega_j\lambda$, each public cloud can be formulated as a $M/M/N_j$ queueing model. The execution time taken at the jth cloud VM can be expressed as follows (which is similar to (7)),

$$\tau_j = \frac{C\left(N_j, \frac{\omega_j\lambda}{\mu_j}\right)}{N_j\mu_j - \omega_j\lambda} + \frac{1}{\mu_j} \tag{12}$$

where $C(N_j, \frac{\omega_j\lambda}{\mu_j})$ refers to the Erlang's C formula which can be written as

$$C\left(N_j, \frac{\omega_j\lambda}{\mu_j}\right) = \frac{1}{1 + \left(1-\rho_j\right)\left(\frac{N_j!}{\left(N_j\rho_j\right)^{N_j}}\right)\sum_{k=0}^{N_j-1}\frac{\left(N_j\rho_j\right)^k}{k!}} \tag{13}$$

where $\rho_j = \frac{\omega_j\lambda}{N_j\mu_j}$ and $\rho_j < 1$ is assumed. Hence, as for N_j, the inequality $\frac{\omega_j\lambda}{\mu_j} < N_j$ can be established.

After completing the remaining task at the jth public cloud, the jth public cloud sends the output data back to the private cloud. The transmission delay for this process T_{j0} is $\frac{L\alpha_j\beta_j}{R_j}$, which is similar to (10). The DTC β_j is multiplied as a result of the processing. The network delay for transferring the output data back to the private cloud can be written as

$$\tau_{j0} = \frac{L\alpha_j\beta_j}{R_j} + T_{equip} + T_{prop}. \tag{14}$$

The total time required to process the remaining tasks using the jth public cloud D_j is

$$D_j = \tau_G + \tau_j + \tau_N \tag{15}$$

where $\tau_N = \tau_{0j} + \tau_{j0}$, which embodies all delays regarding the networking.

B. Latency Model for the Private Cloud

The groundwork is required even when using a private cloud. Thus, τ_G is also required for the groundwork of the public clouds processing the remaining tasks. When the task is handled by the private cloud, the networking process is not required, therefore, $\tau_N = 0$ in this case.

The time delay experienced by the private cloud in processing the input IIoT data is expressed in (16).

$$\tau_0 = \frac{C\left(N_0, \frac{\omega_0\lambda}{\mu_0}\right)}{N_0\mu_0 - \omega_0\lambda} + \frac{1}{\mu_0} \tag{16}$$

In (16), $C(N_0, \frac{\omega_0\lambda}{\mu_0})$ is also refered to as the Erlang's C formula. Thus, the time taken to process the remaining task at the private cloud D_0 is

$$D_0 = \tau_G + \tau_0. \tag{17}$$

IV. Problem Formulation for Cost Optimization

A. Difficulty in Applying KKT

In this section, the minimum cost hybrid cloud computing scheme is derived based on a mandatory requirement that the task has to be completed within the delay threshold D^*. The main objective of this problem is to find the minimum number of VMs $N_0^*, N_1^*, \ldots, N_j^*$ that will satisfy the time delay threshold. In problem (P1), the vector $\boldsymbol{N} = (N_0, N_1, \ldots, N_M)$. The shape of this problem resembles a typical KKT problem. However, it is difficult to solve problem (P1) using general KKT procedures because the Erlang C formulas in (7), (12), and (16) are not differentiable. Therefore, another method is needed. This problem will be solved by using the upper bound of the Erlang C formula.

$$(\mathbf{P1}) \quad \min_{\boldsymbol{N}} \ \mathbb{C}_{tot} \quad \text{Subject to} \quad \begin{aligned} D_0 &\le D^* \\ D_1 &\le D^* \\ &\vdots \\ D_M &\le D^* \end{aligned}$$

B. Upper Bound of the Erlang C Formula

In this section, the upper bounds U_G, U_0, and U_j of the Erlang C formula are introduced in order to make the constraints in problem (P1) differentiable. According to [51], each Erlang C formula has the following upper bound, $C(N_0, \frac{\lambda}{\mu_G}) \le U_G$, $C(N_0, \frac{\omega_0\lambda}{\mu_0}) \le U_0$, and $C(N_j, \frac{\omega_j\lambda}{\mu_j}) \le U_j$, where

$$U_G = 1 + \frac{N_0(1-\rho_G)^2}{2\rho_G} - \frac{1-\rho_G}{2\rho_G}\sqrt{4N_0\rho_G + N_0^2(1-\rho_G)^2} \tag{18}$$

$$U_0 = 1 + \frac{N_0(1-\rho_0)^2}{2\rho_0} - \frac{1-\rho_0}{2\rho_0}\sqrt{4N_0\rho_0 + N_0^2(1-\rho_0)^2} \tag{19}$$

$$U_j = 1 + \frac{N_j\left(1-\rho_j\right)^2}{2\rho_j} - \frac{1-\rho_j}{2\rho_j}\sqrt{4N_j\rho_j + N_j^2\left(1-\rho_j\right)^2} \tag{20}$$

By differentiating U_G and U_0 by N_0, and differentiating U_j by N_j, the derivatives of U_G, U_0, and U_j can be obtained as follows.

$$\frac{\partial U_G}{\partial N_0} = \frac{3-2\rho_G}{2\rho_G} - \sqrt{4N_0\rho_G + N_0^2(1-\rho_G)^2} - \frac{N_0(1-\rho_G)^2}{2\rho_G\sqrt{4N_0\rho_G + N_0^2(1-\rho_G)^2}} \tag{21}$$

$$\frac{\partial U_0}{\partial N_0} = \frac{3-2\rho_0}{2\rho_0} - \sqrt{4N_0\rho_0 + N_0^2(1-\rho_0)^2} - \frac{N_0(1-\rho_0)^2}{2\rho_0\sqrt{4N_0\rho_0 + N_0^2(1-\rho_0)^2}} \tag{22}$$

$$\frac{\partial U_j}{\partial N_j} = \frac{3-2\rho_j}{2\rho_j} - \sqrt{4N_j\rho_j + N_j^2\left(1-\rho_j\right)^2} - \frac{N_j\left(1-\rho_j\right)^2}{2\rho_j\sqrt{4N_j\rho_j + N_j^2\left(1-\rho_j\right)^2}} \tag{23}$$

Now The delay upper bounds $\urcorner_G$, $\urcorner_0$, and $\urcorner_j$ are defined by replacing the Erlang C formulas respectively with the upper bounds U_G, U_0, and U_j. Then, $\urcorner_G$, $\urcorner_0$, and $\urcorner_j$ are respectively defined in (24)–(26).

$$\urcorner_G = \frac{U_G}{N_0\mu_G - \lambda} + \frac{1}{\mu_G} \tag{24}$$

$$\urcorner_0 = \frac{U_0}{N_0\mu_0 - \omega_0\lambda} + \frac{1}{\mu_0} \tag{25}$$

$$\urcorner_j = \frac{U_j}{N_j\mu_j - \omega_j\lambda} + \frac{1}{\mu_j} \tag{26}$$

Obviously, the three inequalities, $\tau_G \leq \urcorner_G$, $\tau_0 \leq \urcorner_0$, and $\tau_j \leq \urcorner_j$ hold.

C. New Optimization Statement With Upper Bounds

In this section, problem (P1) is re-formulated as the relaxation problem (P2) with the delay upper bounds which were introduced in the previous section. Then, the KKT solution for problem (P2) is subsequently presented. In addition, vector $\boldsymbol{g}$ is defined in the following form.

$$\boldsymbol{g} = [g_0, g_1, \ldots, g_M]^T \tag{27}$$

where

$$g_0 = \urcorner_G + \urcorner_0 - D^* \tag{28}$$

$$g_{\forall j \in J, j \neq 0} = \urcorner_G + \urcorner_j + \tau_N - D^* \tag{29}$$

Equations (28) and (29) are made up by respectively replacing τ_G, τ_0, and τ_j with $\urcorner_G$, $\urcorner_0$, and $\urcorner_j$ in D_0 and D_j, and then subtracting D^*. Thus, the inequalities $D_j - D^* \leq g_j$ for all j are established. These inequalities represents the time constraints.

Although KKT could not be applied to (P1) (due to the inability to differentiate the formulas), because $\boldsymbol{g}$ is differentiable, by replacing the constraints in (P1) with $\boldsymbol{g}$, the relaxation problem (P2) is formulated as follows.

Based on (P2), the following three lemmas can be established.

$$(\mathbf{P2}) \quad \begin{aligned} \min_{\boldsymbol{N}} \quad & \mathbb{C}_{tot} \\ \text{Subject to} \quad & g_0 \leq 0 \\ & g_1 \leq 0 \\ & \vdots \\ & g_M \leq 0 \end{aligned}$$

Lemma 2: The solutions of (P2) satisfy the constraints of (P1).

Proof: The solution of (P2) satisfies the constraints $g_0 \leq 0$, $g_1 \leq 0, \cdots, g_M \leq 0$. Due to the fact that the inequality $D_j - D^* \leq g_j$ holds, the solution of (P2) satisfies $D_j - D^* \leq g_j \leq 0$ for $\forall j$. ■

Just as every relaxation problem gives a nearby solution, rather than an optimum solution, the solution of (P2) is not the optimum solution of (P1). In the next section, the exact optimum solution will be searched in the adjacent region of the solution of (P2) using the proposed MARS scheme. Convexity is proved in lemma 3 so that the KKT conditions can be applied.

Lemma 3: The inequality constraints in (P2) are all convex.

Proof: Note that $\urcorner_G$, $\urcorner_0$, and $\urcorner_j$ in constraint $\boldsymbol{g}$ are the only functions associated with convexity. The other parts can be considered as constants. Since a convex set is closed under addition, the proofs of convexity of $\urcorner_G$, $\urcorner_0$, and $\urcorner_j$ are sufficient to prove convexity of $\boldsymbol{g}$. Given $\urcorner_G$, $\urcorner_0$, and $\urcorner_j$ have the same form, proving one of $\urcorner_G$, $\urcorner_0$, and $\urcorner_j$ is sufficient to deem the other functions are convex. By rewriting $\urcorner_j$ in (26) considering $\rho_j = \frac{\omega_j\lambda}{N_j\mu_j}$, $\urcorner_j$ becomes

$$\urcorner_j = \frac{1 + \frac{\left(N_j - \frac{\omega_j\lambda}{\mu_j}\right)^2}{2\frac{\omega_j\lambda}{\mu_j}} - \frac{N_j - \frac{\omega_j\lambda}{\mu_j}}{2\frac{\omega_j\lambda}{\mu_j}}\sqrt{4\frac{\omega_j\lambda}{\mu_j} + \left(N_j - \frac{\omega_j\lambda}{\mu_j}\right)^2}}{\mu_j\left(N_j - \frac{\omega_j\lambda}{\mu_j}\right)}. \tag{30}$$

By substituting $\frac{\omega_j\lambda}{\mu_j}$ with δ, the following can be obtained.

$$\urcorner_j = \frac{1 + \frac{\left(N_j - \delta\right)^2}{2\delta} - \frac{N_j - \delta}{2\delta}\sqrt{4\delta + \left(N_j - \delta\right)^2}}{\mu_j\left(N_j - \delta\right)} \tag{31}$$

By decomposing the above fraction,

$$\urcorner_j = \left(\frac{1}{\mu_j}\right)\left\{\frac{1}{N_j - \delta} + \frac{N_j - \delta}{2\delta} - \frac{\sqrt{4\delta + \left(N_j - \delta\right)^2}}{2\delta}\right\} \tag{32}$$

$\frac{1}{N_j - \delta}$ is convex since $N_j \geq \delta$, $\frac{N_j - \delta}{2\delta}$ is convex since it is a linear function. The term $\frac{\sqrt{4\delta + (N_j - \delta)^2}}{2\delta}$ is the square root of the quadratic function, so that it is concave. Hence, $-\frac{\sqrt{4\delta + (N_j - \delta)^2}}{2\delta}$ is convex. Because $\frac{1}{N_j - \delta}$, $\frac{N_j - \delta}{2\delta}$, and $-\frac{\sqrt{4\delta + (N_j - \delta)^2}}{2\delta}$ are convex, correspondingly $\urcorner_j$ is convex. As a result, $\urcorner_G$ and $\urcorner_0$ are also convex, and therefore, it can be concluded it that the constraints of $\boldsymbol{g}$ are convex. ■

Lemma 4: If optimum values are found by using KKT, $N^*_{\forall j \in J}$ are the global optimal solutions of (P2).

Proof: If the objective function and inequality constraints are convex functions, the solution obtained from KKT is sufficient to be the global optimum [52]. Since $\mathbb{C}_{tot}$ is a linear function, $\mathbb{C}_{tot}$ is a convex function. The inequality constraints are also convex as proved in Lemma 3. Hence, the solution obtained from KKT is a global optimum solution. ■

The next step is to find the solution of (P2) using KKT. The KKT conditions for (P2) are as follows.

$$\boldsymbol{\pi}^* \geq 0 \tag{33}$$

$$\nabla\mathbb{C}_{tot}(\boldsymbol{N}^*) + \nabla\boldsymbol{g}(\boldsymbol{N}^*) \cdot \boldsymbol{\pi}^* = 0 \tag{34}$$

$$\boldsymbol{\pi}^{*T} \cdot \boldsymbol{g}(\boldsymbol{N}^*) = 0 \tag{35}$$

The superscript '*' indicates the solution and $\boldsymbol{\pi}$ denotes the KKT multiplier vector defined as $\boldsymbol{\pi} = [\pi_0, \pi_1, \ldots, \pi_M]^T$. In

addition, the feasibility of the solution of (P2) is as follow.

$$\boldsymbol{g}(\boldsymbol{N}^*) \leq 0 \tag{36}$$

These are a general form of the KTT conditions, thus it does not deviate much from the general way to find the KKT solution. The solution is as follows. By differentiating $\mathbb{C}_{tot}(\boldsymbol{N})$ and $\boldsymbol{g}(N)$, the gradients $\nabla\mathbb{C}_{tot}(N)$ and $\nabla\boldsymbol{g}(\boldsymbol{N})$ are

$$\nabla\mathbb{C}_{tot}(\boldsymbol{N}) = [\Gamma_0, \Gamma_1, \ldots, \Gamma_M] \tag{37}$$

and $\nabla\boldsymbol{g}(\boldsymbol{N})$ is represented in (38), shown at the bottom of this page. To satisfy the condition of (34), $\nabla\mathbb{C}_{tot}(\boldsymbol{N}^*) + \nabla\boldsymbol{g}(\boldsymbol{N}^*) \cdot \boldsymbol{\pi}^{*T} = 0$, the below equality should hold.

$$\Gamma_j = -\pi_0\left(\frac{\frac{\partial U_G}{\partial N_0}\left(N_0\mu_G - \lambda\right) - U_G\mu_G}{\left(N_0\mu_G - \lambda\right)^2}\right) - \pi_j\left(\frac{\frac{\partial U_j}{\partial N_j}\left(N_j\mu_j - \omega_j\lambda\right) - U_j\mu_j}{\left(N_j\mu_j - \omega_j\lambda\right)^2}\right) \text{ for } \forall j \tag{39}$$

To meet the condition of (35), $\boldsymbol{\pi}^{*T} \cdot \boldsymbol{g}(\boldsymbol{N}^*) = 0$ is required. This equality can be re-written as (40).

$$\pi_0(\urcorner_G + \urcorner_0 - D^*) + \sum_{j=1}^{M}\pi_j\left(\urcorner_G + \urcorner_j + \tau_N - D^*\right) = 0 \tag{40}$$

If $\pi_0 = 0$, then $\Gamma_0 = 0$ in (39), which is a contradiction since $\Gamma_0 = 0$ is the cost coefficient. Thus, $\pi_0 \neq 0$. If $\pi_j = 0$ for $j \neq 0$, a contradiction $\Gamma_1 = \Gamma_2 = \cdots = \Gamma_M$ occurs. Thus, $\pi_j \neq 0$. From (40), $\urcorner_G + \urcorner_0 - D^* = 0$ and $\urcorner_G + \urcorner_j + \tau_N - D^* = 0$ for $j \neq 0$ should hold since $\pi_j \neq 0$ for $\forall j$. In addition, because the service time for the groundwork commonly takes less time compared to the service time to process the remaining main tasks, the following inequality can be considered to hold true.

$$\urcorner_G \ll \urcorner_j \text{ for } \forall j \tag{41}$$

In other words, $\urcorner_G$ is less than an arbitrary small number ϵ, where $\epsilon \ll D^*$, thus the following equalities roughly holds true.

Algorithm 1: MARS.

1) **INPUT** data traffic λ
2) **IF** state (m_j, k_j) is not in normal conditions
3) **CHECK** size of m_j and k_j
4) **IF** in the *likely-over-provisioned* state
5) **THEN ALLOCATE** VMs corresponding to the number of $N_j^* = \frac{\zeta_j}{3\eta_j} - \frac{6\nu_j - \nu_j^2\eta_j^2}{3\zeta_j} + \frac{1}{3}\nu_j(\eta_j + 3)$ to the jth cloud, for $\forall j$ **AFTER** $\frac{\alpha_j\lambda}{\mu_j N_j}$
6) **ELSE IF** in the *likely-under-provisioned* state
7) **THEN ALLOCATE** VMs corresponding to the number of $N_j^* = \frac{\zeta_j}{3\eta_j} - \frac{6\nu_j - \nu_j^2\eta_j^2}{3\zeta_j} + \frac{1}{3}\nu_j(\eta_j + 3)$ to the jth cloud, for $\forall j$
8) **IF** not in the normal state
9) **START** CTMC scheme until the state returns to normal state
10) **END**

$$\urcorner_0 \approx D^* - \epsilon \tag{42}$$

$$\urcorner_j \approx D^* - \tau_N - \epsilon \text{ for } j \neq 0 \tag{43}$$

Equation (42) and (43) are for N_0 and N_j, respectively. As a result, the solutions become

$$N_j^* = \frac{\zeta_j}{3} - \frac{6\nu_j - \nu_j^2\eta_j^2}{3\zeta_j} + \frac{1}{3}\nu_j\left(\eta_j + 3\right) \text{ for } \forall j \tag{44}$$

where

$$\nu_j = \frac{\omega_j\lambda}{\mu_j} \text{ for } \forall j \tag{45}$$

$$\eta_j = \begin{cases} \mu_0 D^* - \mu_0\epsilon - 1 & \text{for } j = 0 \\ \mu_j D^* - \mu_j\tau_N - \mu_j\epsilon - 1 & \text{for } j \neq 0 \end{cases} \tag{46}$$

$$\zeta_j = \frac{3\sqrt{2\nu_j^3\eta_j^6 - 18\nu_j^2\eta_j^4 + 3\sqrt{3}\sqrt{27\nu_j^2\eta_j^4 - 4\nu_j^3\eta_j^6} + 27\nu_j\eta_j^2}}{3\sqrt{2}\eta_j} \text{ for } \forall j. \tag{47}$$

The KKT solutions of (44) are used in the MARS scheme, which is introduced in Algorithm 1. Details of Algorithm 1 are provided in the next section.

$$\nabla\boldsymbol{g}(\boldsymbol{N}) = \begin{pmatrix} \frac{\frac{\partial U_G}{\partial N_0}(N_0\mu_G - \lambda) - U_G\mu_G}{(N_0\mu_G - \lambda)^2} + \frac{\frac{\partial U_0}{\partial N_0}(N_0\mu_0 - \omega_0\lambda) - U_0\mu_0}{(N_0\mu_0 - \omega_0\lambda)^2} & 0 & \cdots & 0 \\ \frac{\frac{\partial U_G}{\partial N_0}(N_0\mu_G - \lambda) - U_G\mu_G}{(N_0\mu_G - \lambda)^2} & \frac{\frac{\partial U_1}{\partial N_1}(N_1\mu_1 - \omega_1\lambda) - U_1\mu_1}{(N_1\mu_1 - \omega_1\lambda)^2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\frac{\partial U_G}{\partial N_0}(N_0\mu_G - \lambda) - U_G\mu_G}{(N_0\mu_G - \lambda)^2} & 0 & \cdots & \frac{\frac{\partial U_M}{\partial N_M}(N_M\mu_M - \omega_M\lambda) - U_M\mu_M}{(N_M\mu_M - \omega_M\lambda)^2} \end{pmatrix} \tag{38}$$

V. MARS Scheme

This section describes the method of how to apply the KKT solution obtained in the previous section to the MARS algorithm. This paper adopts the CTMC scheme in [22], and the auto-scaling scheme as well. The following section briefly introduces the original auto-scaling scheme proposed in [22]. This paper adds two new states, *likely-over-provisioned* and *likely-under-provisioned*, to the scheme.

A. Auto-Scaling Scheme

- m_j: Number of active VMs in the jth cloud
- k_j: Number of tasks in the jth cloud
- (m_j, k_j): State of the jth cloud
- $[a(m_j), b(m_j)]$: Pair of integers that determine the status of the state of the jth cloud, where $b(m_j) > a(m_j) \geq m-1, b(m_j) \geq a((m+1)_j)$, and $a(1_j) < a(2_j) < a(3_j) < \cdots, b(1_j) < b(2_j) < b(3_j) < \cdots$.

$$State = \begin{cases} Over-provisioned, & \text{if } 0 \leq k_j \leq a(m_j) \\ Normal, & \text{if } a(m_j) < k_j \leq b(m_j) \\ Under-provisioned, & \text{if } b(m_j) < k_j \end{cases}$$

The operation mode of this scheme can be classified as a *reactive mode* for the reason that it acts in response to workloads in the system [13]. Reactive mode has a significant defect; it is vulnerable to bursty situations, because it will adapt its conditions after a problematic condition has occurred. To better cope with problematic conditions, proactive mode adaptation is required [13], [14]. In [22], this shortcoming was pointed out, and a multiple start-up and shut-down scheme was suggested as future work, which is the objective of this paper. In [23], a proactive mode scheme that is called ControCity has been proposed, and it has been shown that the CTMC scheme in [26], which is very similar to [22], can be surpassed by using *proactive mode*. The result has been analyzed with the elasticity metric in [22]. ControCity predicts workloads with recent trends. In other words, time-series analysis has been applied.

However, time-series analysis is not suitable for periodic workloads [14]. For instance, IIoT devices commonly access clouds much more during the daytime than during the night hours. To overcome this problem, the new concepts of the two states *likely-over-provisioned* and *likely-under-provisioned* were added. With the indicators $o(m_j)$ and $u(m_j)$, the *likely-over-provisioned* and *likely-under-provisioned* states can be defined as

$$State = \begin{cases} Likely-over-provisioned, & \text{if } o(m_j) \leq m_j \\ Over-provisioned, & \text{if } k_j \leq a(m_j) \\ Normal, & \text{if } a(m_j) < k_j \leq b(m_j) \\ Under-provisioned, & \text{if } b(m_j) < k_j \\ Likely-under-provisioned, & \text{if } m_j < u(m_j) \end{cases} \tag{48}$$

where $o(m_j) = L_o N_j^*$ and $u(m_j) = L_v N_j^*$. In (48), L_o and L_v are adjustable and arbitrary positive real values, $o(m_j)$ and $u(m_j)$ decide whether the currently assigned VMs to the jth cloud will be *likely-over-provisioned* or *likely-under-provisioned* in the near future. Note that $o(m_j)$ and $u(m_j)$ are irrelevant to the number of tasks in progress, but they have relevance to the active number of VMs. Recall that the optimum number of VMs N_j^* is based on the Poisson process λ which was established in (1) for the first time. Therefore, it can be stated that N_j^* technically depends on the number of active IIoT devices. For this reason, $o(m_j)$ and $u(m_j)$ are proactive and suitable for periodic workloads under which the active number of IIoT devices are predictable.

This paper adopts the elasticity metric $\mathbf{E}$ defined in [22]

$$\mathbf{E} = \frac{S_{normal}}{S_{total}} = 1 - \frac{S_{over} + S_{under}}{S_{total}} \tag{49}$$

where S_{normal}, S_{over}, and S_{under} are respectively the total time of the system in normal, over-provisioned, and under-provisioned conditions when the cloud computing system operates for a time period of S_{total}.

B. CTMC Scheme

In the CTMC scheme [22], a new VM can be activated or deactivated at any time. The time to activate a new VM follows an exponential distribution with mean π_{ac}, and the time to deactivate an active VM follows an exponential distribution with mean π_{de}. However, this CTMC scheme can activate only one VM at a time, which results in degradation of elasticity since S_{over} or S_{under} in (49) can rapidly increase. This argument will be confirmed in the simulation results in the following section.

C. KKT and CTMC Applied to MARS

In the CTMC scheme, start-up or shut-down of VMs occurs once at a time. However, when the status of a state is likely-over-provisioned or likely-under-provisioned, the CTMC scheme takes too much time in allocating the appropriate amount of VMs to return back to the normal state. In this section the KKT solution of (44) is applied to solve this problem. Multiple VMs can be activated or de-activated based on the solution of the KKT solution. The proposed MARS scheme consists of two steps. First, when in the likely-over-provisioned or likely-under-provisioned states, MARS allocates VMs corresponding to the solution of the KKT N_j^* to each cloud. Second, adjust the number of VMs by applying the CTMC scheme. This second procedure is not necessary when in normal state. The Algorithm 1 elaborates the procedures of MARS. One noteworthy thing is that when in a likely-over-provisioned state, the KKT solution should be applied after $\frac{\alpha_j \lambda}{\mu_j N_j}$ of time. By reducing the number of VMs after waiting a short time, $\frac{\alpha_j \lambda}{\mu_j N_j}$, instead of immediately, the system can handle bursty workloads currently queued.

In short, MARS consists of two stages, the KKT stage and the CTMC stage. In proactive mode, KKT offers an approximate guide to the number of VMs needed. In reactive mode, CTMC finds the exact number of VMs needed to recover to normal state. This MARS scheme can dramatically increase the elasticity as

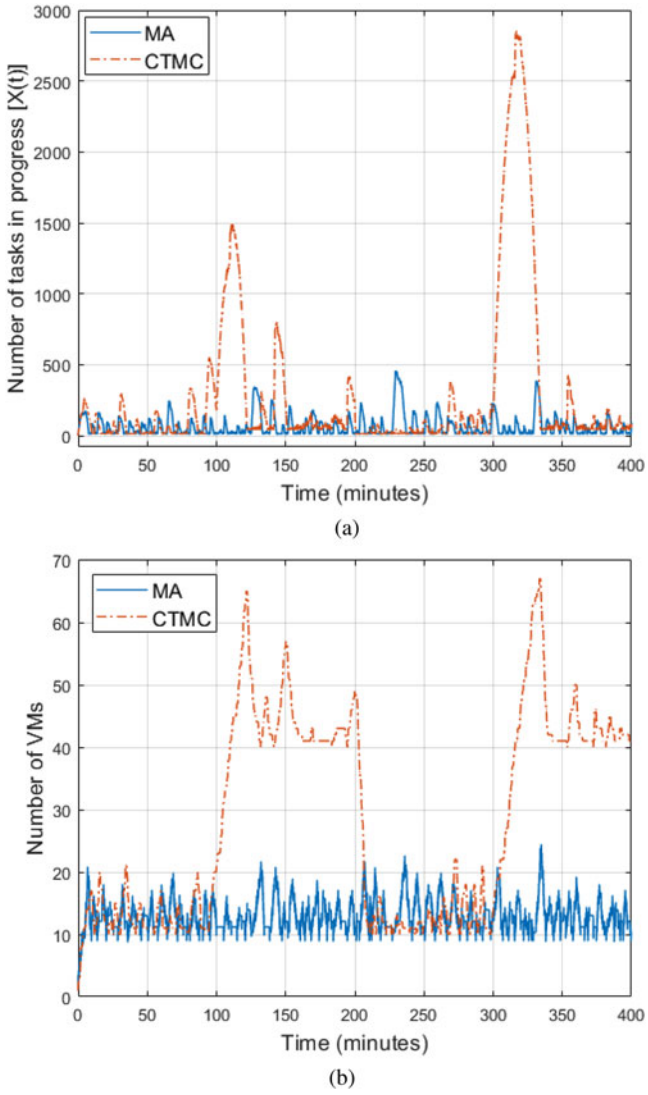

Fig. 2. Comparison between MA and CTMC based on (a) tasks in progress in cloud (CTMC) vs. prediction from MA and (b) allocated VMs by CTMC vs. prediction from MA.

the following simulation results demonstrate. One notable thing is that the MARS scheme does not increase the time complexity at all. This is definitely different from computationally intensive approaches such as artificial intelligence or ML.

Lemma 5: The MARS scheme does not increase the time complexity of the original CTMC scheme.

Proof: As mentioned earlier, the MARS scheme is a form of reactive mode (the CTMC scheme) plus proactive mode which use the KKT solutions for likely-over-provisioned states and likely-under-provisioned states. As can be seen in steps 4 through 7 of Algorithm I, the proposed MARS proactive mode does not increase the running time as the input grows. To be more specific, the inputs (N_j, λ, ζ_j, η_j, ν_j, α_j, μ_j) are used only to compute the closed form solution of N_j^* using (44), which leads to the conclusion that the size of the

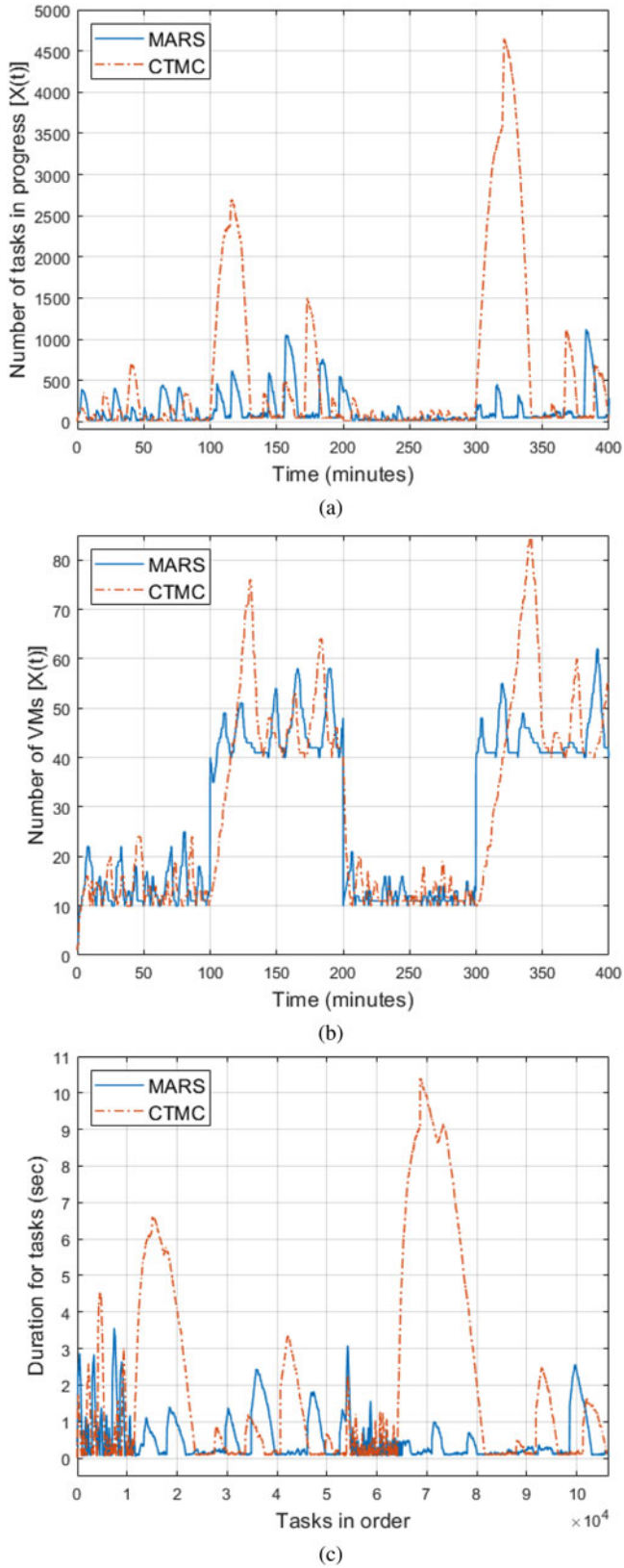

Fig. 3. Comparison between the MARS and CTMC schemes, based on (a) number of tasks in progress, (b) number of allocated VMs, and (c) the time taken to process the tasks.

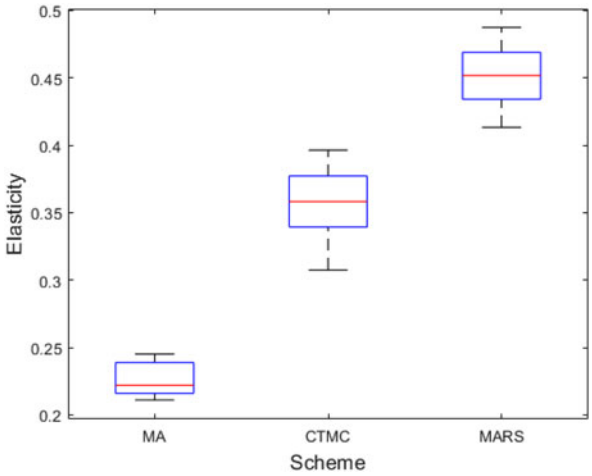

Fig. 4. Elasticities for MARS, CTMC, and MA.

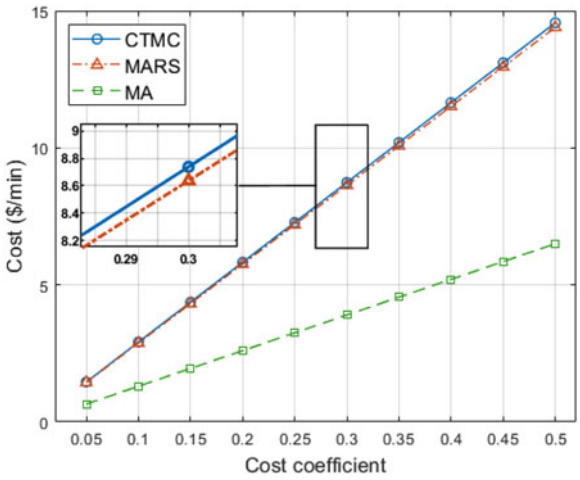

Fig. 5. Average costs for running MARS, CTMC, and MA.

input does not influence the time complexity. Thus, the time complexity in proactive mode is $O(1)$, and therefore, does not increase the time complexity of the original CTMC scheme. ■

VI. Simulations and Results

The ultimate goal of the simulation is to investigate how well the MARS scheme works under bursty situations. To be specific, the goal is to measure the elasticity, which is a prediction capability about VMs needed in the near future according to the current workloads and the remaining tasks in the queue. Average cost and response time for each scheme were also estimated. Characteristics of the workloads were time-series and contains seasonal variation. Based on the performance of the CTMC scheme in [22], the performance of the MA scheme proposed in [23] and the MARS scheme were compared. The MA scheme was chosen to check the performance of the ML scheme of [23], which claims that the MA based ML scheme can provide a performance exceeding the CTMC scheme.

In summary, the simulation results of Fig. 2 show that the MA scheme of [23] is less elastic than the CTMC scheme as it

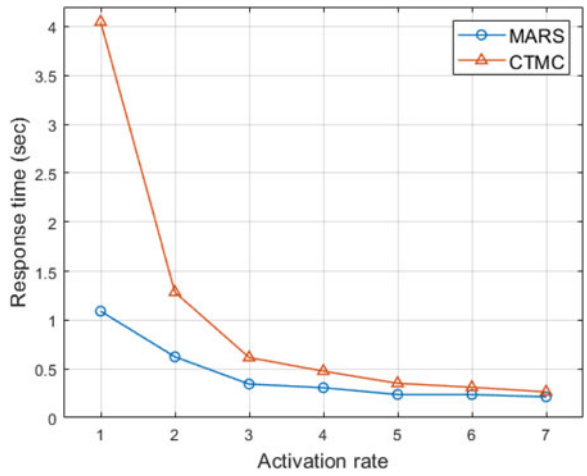

Fig. 6. Average response time for MARS and CTMC.

performs worse when the data pattern changes are bursty. Fig. 3 demonstrates how effective the MARS scheme can make quick adjustments to bursty changes. Fig. 4 further illustrates how elastic the MARS scheme can perform. Figs. 5 and 6 show the performance of MARS in terms of cost and response time. The simulation experiments have been conducted using Python 3.7.0 along with MATLAB 2018b. Details of the simulation experiments are provided in the following sections.

A. Comparative Analysis of MA, CTMC, and MARS

Fig. 2 shows a comparison between MA and CTMC. Fig. 3 represents a comparison between CTMC and MARS. The simulation environments for Figs. 2 and 3 were the same and had the following details. Based on a M/M/c queueing model, the CTMC scheme in [22] adjusts the number of VMs in response to current workloads in its queues. In order to focus on the prediction effect, a single public cloud is allocated in the first simulation. There are 100 IIoT devices in the cloud-assisted massive IIoT network in the beginning and each IIoT device sends their task, which results in $\lambda = 100$ that is applied to (1). The deadline D^* for processing each task was set for 18 seconds, and the service rate of the VMs are set to $\mu = 6$. The activation rate was set to $\pi_{ac} = 2$ and the deactivation rate was set to $\pi_{de} = 5$ in the simulation experiments, which is the same experimental environment as used in [22] (and very similar to [17]) for performance comparison purposes. After 100 minutes, a bursty situation occurs where the number of IIoT devices increases to 400, and 100 minutes later, the number of IIoT devices decreases to 100 again. Then, another 100 minutes later, a burst situation occurs again, where 400 IIoT devices access the cloud once again. Overall, the simulation experiment was executed for 400 minutes. There are two MA schemes that were used. The first was a scheme for predicting the remaining tasks, and the second was a scheme for predicting the required VMs. The order of the first MA scheme was 10 (denoted as 'MA(10)'), and a 5th order MA scheme (denoted as 'MA(5)') was used secondly. The two MA schemes had been trained only for 100 IIoT devices, so that it was not trained based on a bursty situation.

The time taken to process the tasks was not measurable because the MA scheme went into an overflow condition. As shown in Fig. 2, the MA scheme is compared to the CTMC scheme in terms of (a) task processing capability and (b) elasticity performance. The MA based scheme and CTMC scheme both were not able to accurately adjust to the changing bursty conditions, showing a lack of elasticity and accuracy in dynamic resource allocation. This is because, in bursty situations, the remaining tasks and the number of required VMs are very difficult to predict. Existing papers on cloud computing commonly consider the cloud usage cost, traffic arrival rate, activation rate, and deactivation rate as parameters. However, network delay (which includes the equipment delay, propagation delay, and private to public cloud time delay) and the processing job division between the private cloud and the *j*th public cloud have not been considered. Therefore, the existing schemes cannot accurately estimate the number of VMs to use in the private and public clouds. In the proposed MARS scheme, all of the parameters are included in the KKT optimization process, such that the optimal number of VMs to use in the private and public clouds are exactly obtained (i.e., N_j^*) using (44), which significantly improves the elasticity and accuracy performance. As a result, the proposed MARS scheme is very effective, as shown in Fig. 3.

Fig. 3 compares the performance of the CTMC scheme to the MARS scheme, where the results show that the MARS scheme outperforms the CTMC scheme in terms of both elasticity and accuracy. The time instances of 100, 200, and 300 minutes are when the number of IIoT devices rapidly change in the simulation experiments. As shown in Fig. 3(a), MARS adaptively keeps a relatively small number of tasks in the queue compared to the CTMC scheme. When bursty situations occur at the 100 minutes and 300 minutes instances, the CTMC scheme was not able to control the number of tasks in the queue, as a result, an excessive overflow occurred. The main difference of the MARS and CTMC performance results are based on the fact that the proposed MARS scheme can accurately provide the optimal number of VMs instantaneously using the simple KKT solution derived in (44). Fig. 3(b) shows the effectiveness of the MARS scheme, as it can accurately compute and allocate the required number of VMs when a bursty changing situation suddenly begins or ends. On the other hand, the CTMC scheme takes a relatively long time to adapt to the bursty changed situation. When the bursty situation occurs at 100 minutes and 300 minutes, the CTMC scheme took about 50 minutes to properly allocate the proper number of VMs. Using the optimal number of VMs computed, the MARS scheme is able to accurately adapt to changes very quickly as shown in Fig. 3(c). Due to the slower adaptability, the experiment results show that the CTMC scheme will take about 10 seconds to process a task when a bursty situation occurs, which is not a desirable phenomenon.

B. Elasticity

Fig. 4 compares the elasticity performance of MA, CTMC, and MARS using a boxplot. The results show that the elasticity performance of MARS is far superior compared to the CTMC and MA schemes. Fig. 4 shows the elasticity performance based on an average of 10 simulation experiments. The average elasticity of MARS (represented by the red line) is 0.4516, which shows an improvement of 26.7% when compared to the average of CTMC, which is at 0.3564. Because the MA scheme fails to predict bursty situations, the average elasticity is at the 0.2295 level, which is lower than the CTMC scheme and significantly lower than the MARS scheme.

C. Cost and Response Time

The average cost and average response time of the MA, CTMC, and MARS schemes are analyzed in this section. The experimental environment is the same as the previous simulation experiments. The average response time of the MA scheme could not be obtained. This is because, in bursty situations, most of the tasks remain unprocessed because there are not enough VMs assigned, which results in a large accumulation of backlogged tasks in the queue. The cost coefficient Γ_j was set to have an interval of 0.05 from 0.05 to 0.5 in the analysis. This cost coefficient value selection was based on the following records of actual events. As of April 2020, in the eastern United States, one of the GPU instances of Amazon EC2, p3.2xlarge cost was priced as approximately 0.05 dollars/minute. The other GPU instances, p3.8xlarge and p3.16xlarge cost approximately 0.2 dollars/minute and 0.4 dollars/minute, respectively [53]. Fig. 5 shows that the MARS scheme requires a smaller cost, as much as 1.2% less than the CTMC scheme. The estimated cost of running the MA scheme was much lower, however this was due to the failure to add the appropriate number of VMs to satisfy the changed bursty situation. Therefore, this can be considered meaningless as the MA scheme's performance should be considered a failure. Thus, it can be seen that the proposed MARS scheme performs much better at a lesser average price than the CTMC scheme. In the response time experiment, the activation rate π_{ac} was increased by 1 from 1 to 7 with the deactiviation rate π_{de} fixed at 5. Five experiments were performed for each activation rate and the response time was averaged. In terms of response time, MARS performs much better than the CTMC scheme as shown in Fig. 6. The average response times of MARS and CTMC are 1.0910 and 4.0468, respectively, when $\pi_{ac} = 1$. When the activation rate is 1, the time for allocating VMs is relatively long when using the CTMC scheme in comparison to the MARS's deployment solution. The average response times of MARS and CTMC are 0.2134 and 0.2644, respectively, when $\pi_{ac} = 7$. The gap is reduced when $\pi_{ac} = 7$, but MARS is still more effective. The MARS scheme requires only 27.0% of the response time of the CTMC scheme when $\pi_{ac} = 1$, which is the largest gain among the simulation experiments conducted. In addition, the MARS scheme requires 80.7% of the response time of the CTMC scheme when $\pi_{ac} = 7$, which is the smallest gain among the simulation experiments conducted. In summary, for the range of interest, the results show that the proposed MARS scheme has a significantly smaller response time than the CTMC scheme, which results in a superior elasticity performance.

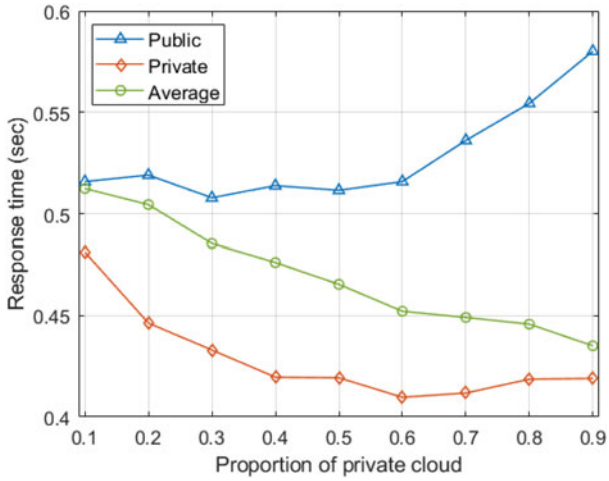

Fig. 7. Average response time of MARS according to the proportion of the private cloud.

D. Influence of the Private Cloud to Public Cloud Ratio on Response Time

The influence of the private cloud to public cloud ratio on the average response time is analyzed in this section. The analysis is based on an environment of one private cloud and one public cloud. Every setting is same as the previous simulation experiments in section IV.A. In this experiment, the proportion of tasks to be processed by the private cloud was tested for the range of 0.1 to 0.9, where the public cloud is responsible for the remaining portion of tasks, which ranges from 0.9 to 0.1 to make the sum of the two clouds equal 1. The total network delay τ_N was set to 100 ms. In Fig. 7, regardless of the proportion of tasks, the average response time of the MARS scheme only changes from 0.5123 to 0.4351, which is a vary small amount of variation considering the significant changes (from 10% to 90%) in the division of task assignments to the private and public clouds tested. This demonstrates that the MARS scheme is is capable of elastically allocating tasks to both the private cloud and the public cloud according to the overall amount of tasks. The average response time of tasks decreases as the proportion of tasks processed by the private cloud increases, which is because the network delay of the private cloud is very small. If the MARS scheme had failed to elastically allocate the proper number of VMS to both clouds, a large increase (or big fluctuations) would have resulted in the average response time. In summary, the results show that the MARS scheme can provide an improved response time performance with more stability in hybrid cloud operations.

VII. Conclusion and Future Work

In this paper, the MARS scheme is proposed, which was developed from the CTMC scheme in [22]. The CTMC scheme was able to allocate VMs accurately but failed to assign them quickly, resulting in a lack of elasticity. Analysis results show that MA time-series forecasting based ML schemes perform poorly when coping with bursty situations as well. The proposed MARS scheme overcomes the shortcomings of existing schemes, as the optimal number of VMs are precisely derived using a KKT solution value that is easy to compute. Furthermore, when using the MARS scheme, the network condition and private and public could(s) resources are fully considered, which significantly broadens the usability and applicability of the proposed scheme. The simulation results show that the MARS scheme (when compared to CTMC scheme) can improve the elasticity by approximately 26.7% and improve the response time up to 19.3 $\sim$ 73% (based on the scenarios range of interest investigated) and reduce the cost by 1.2%.

In future research, edge computing systems (e.g., fog computing, Multiple-access Edge Computing (MEC), cloudlets, etc.) in the private and public clouds need to be included into the resource allocation and performance optimization process by considering networking and processing time factors as well as other computational offloading parameters. Based on the processing capability, memory size, and cost of the edge cloud in the private or public cloud, the MARS scheme can be applied to help assign VMs in real time. Future research needs to focus on how to achieve energy savings by dividing IIoT tasks into subtasks and offloading those subtasks to nearby edge clouds to assist the central cloud (that is normally at a further distance) while achieving optimal resource allocation with elasticity and accuracy.

References

[1] S. K. Sharma and X. Wang, "Toward massive machine type communications in ultra-dense cellular IoT networks: Current issues and machine learning-assisted solutions," *IEEE Commun. Surv. Tuts.*, vol. 22, no. 1, pp. 426–471, Jan.–Mar. 2020.

[2] E. Dutkiewicz, X. Costa-Perez, I. Kovacs, and M. Mueck, "Massive machine-type communications," *IEEE Netw.*, vol. 31, no. 6, pp. 6–7, Nov./Dec. 2017.

[3] S. Mumtaz, A. Alsohaily, Z. Pang, A. Rayes, K. F. Tsang, and J. Rodriguez, "Massive Internet of Things for industrial applications: Addressing wireless IIoT connectivity challenges and ecosystem fragmentation," *IEEE Ind. Electron. Mag.*, vol. 11, no. 1, pp. 28–33, Mar. 2017.

[4] Industrial Internet of Things (IoT) market to grow at a. CAGR of 21.3% during (2020). To 2028. Accessed: Dec. 2020. [Online]. Available: https://www.globenewswire.com/news-release/2020/08/29/2085754/0/en/Industrial-Internet-of-Things-IoT-Market-To-Grow-At-A-CAGR-of-21-3-During-2020-To-2028-Quince-Market-Insights.html

[5] IoT connections outlook. Accessed: Dec. 2020. [Online]. Available: https://www.ericsson.com/en/mobility-report/dataforecasts/IoT-connections-outlook?gclid=CjwKCAiAoOz-BRBdEiwAyuvA6_jrnTCk-N4YZZrP7Anl2nPhLP_aBYh_vJIlvdDFFFoU-CVSxAYBoCyVQQAvD_BwE&gclsrc=aw.ds

[6] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial Internet of Things: Challenges, opportunities, and directions," *IEEE Trans. Ind. Informat.*, vol. 14, no. 11, pp. 4724–4734, Nov. 2018.

[7] Understanding massive IoT and why it matters for your business Accessed: Dec. 2020. [Online]. Available: https://www.rogers.com/business/blog/en/understanding-massive-IoT-and-why-it-matters-for-your-business

[8] J. Yun *et al.*, "MMOG user participation based decentralized consensus scheme and proof of participation analysis on the bryllite blockchain system," *KSII Trans. Internet Info. Syst.*, vol. 13, no. 8, pp. 4093–4107, Aug. 2019.

[9] M. Aazam, S. Zeadally, and K. Harras, "Deploying fog computing in industrial Internet of Things and industry 4.0," *IEEE Trans. Ind. Inform.*, vol. 14, no. 19, pp. 4674–4682, Oct. 2018.

[10] W. Z. Khan, M. H. Rehman, H. M. Zangoti, M. K. Afzal, N. Armi, and K. Salah, "Industrial Internet of Things: Recent advances, enabling technologies and open challenges," *Comput. Elect. Eng.*, vol. 81, Jan. 2020, Art. no. 106522.
[11] P. Mell and T. Grance, "The NIST definition of cloud computing," Special Publication 800-145, National Inst. of Standards Technol., U.S. Dept. Commerce, Sep. 2011.
[12] H. Truong and S. Dustdar, "Principles for engineering IoT cloud systems," *IEEE Cloud Comput.*, vol. 2, no. 2, pp. 68–76, Mar./Apr. 2015.
[13] Y. Al-Dhuraibi, F. Paraiso, N. Djarallah, and P. Merle, "Elasticity in cloud computing: State of the art and research challenges," *IEEE Trans. Serv. Comput.*, vol. 11, no. 2, pp. 430–447, Mar. 2018.
[14] C. Qu, R. Calheiros, and R. Buyya, "Auto-scaling web applications in clouds: A taxonomy and survey," *ACM Comput. Surv.*, vol. 51, no. 4, pp. 73:1-73:33, Jul. 2018.
[15] N. Herbst, S. Kounev, and R. Reussner, "Elasticity in cloud computing: What it is, and what it is not," in *Proc. 10th Int. Conf. Autonomic Comput.*, San Jose, CA, USA, 2013, pp. 23–27.
[16] E. Coutinho *et al.*, "Elasticity in cloud computing: A survey," *Ann. Telecommun.-Ann. des Telecommun.*, vol. 70, no. 7, pp. 289–309, 2015.
[17] B. Wan, J. Dang, Z. Li, H. Gong, F. Zhang, and S. Oh, "Modeling analysis and cost-performance ratio optimization of virtual machine scheduling in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no 7, pp. 1518–1532, Jul. 2020.
[18] H. Routaib, E. Badidi, M. Elmachkour, E. Sabir, and M. Elkoutbi, "Modeling and evaluating a cloudlet-based architecture for mobile cloud computing," in *Proc. 9th Int. Conf. Intell. Syst. Theories Appl.*, Rabat, Morocco, 2014, pp. 1–7.
[19] S. Kikuchi and Y. Matsumoto, "Performance modeling of concurrent live migration operations in cloud computing systems using PRISM probabilistic model checker," in *Proc. 4th IEEE Int. Conf. Cloud Comput.*, Washington, DC, USA, 2011, pp. 49–56.
[20] H. Khazaei, J. Mišić, V. Mišić, and S. Rashwand, "Analysis of a pool management scheme for cloud computing centers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 5, pp. 849–861, May 2013.
[21] H. Khazaei, J. Mišić, and V. Mišić, "A fine-grained performance model of cloud computing centers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 11, pp. 2138–2147, Nov. 2013.
[22] K. Li, "Quantitative modeling and analytical calculation of elasticity in cloud computing," *IEEE Trans. Cloud Comput.*, vol. 8, no. 4, pp. 1135–1148, Oct.-Dec. 2020.
[23] M. Ghobaei-Arani, A. Souri, T. Baker, and A. Hussien, "ControCity: An autonomous approach for controlling elasticity using buffer management in cloud computing environment," *IEEE Access*, vol. 7, pp. 106912–106924, 2019.
[24] J. Weinman, "Hybrid cloud economics," *IEEE Cloud Comput.*, vol. 3, no. 1, pp. 18–22, Jan./Feb. 2016.
[25] J. Singh, T. Pasquier, J. Bacon, H. Ko and D. Eyers, "Twenty security considerations for cloud-supported Internet of Things," *IEEE Internet Things J.*, vol. 3, no. 3, pp. 269–284, Jun. 2016.
[26] K. Salah, K. Elbadawi, and R. Boutaba, "An analytical model for estimating cloud resources of elastic services," *J. Netw. Syst. Manage.*, vol. 24, no. 2, pp. 285–308, 2016.
[27] J. Weinman, "The economics of the hybrid multicloud fog," *IEEE Cloud Comput.*, vol. 4, no. 1, pp. 16–21, Jan./Feb. 2017.
[28] J. Weinman, "The economics of networking and the cloud," *IEEE Cloud Comput.*, vol. 3, no. 3, pp. 12–15, May/Jun. 2016.
[29] K. Popović and Ž. Hocenski, "Cloud computing security issues and challenges," in *Proc. 33rd Int. Conv. Manufactured Imports Promotion Org.*, Opatija, 2010, pp. 344–349.
[30] J. Li, R. Ma, and H. Guan, "TEES: An efficient search scheme over encrypted data on mobile cloud," *IEEE Trans. Cloud Comput.*, vol. 5, no. 1, pp. 126–139, Jan.-Mar. 2017.
[31] Y. Zhang, H. Huang, Y. Xiang, L. Zhang, and X. He, "Harnessing the hybrid cloud for secure big image data service," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1380–1388, Oct. 2017.
[32] M. Ali, S. Khan, and A. Vasilakos, "Security in cloud computing: Opportunities and challenges," *Inf. Sci.*, vol. 305, pp. 357–383, Jun. 2015.
[33] V. Chang, Y. Kuo, and M. Ramachandran, "Cloud computing adoption framework: A security framework for business clouds," *Future Gener. Comput. Syst.*, vol. 57, pp. 24–41, Apr. 2016.
[34] W. Jansen, "Cloud hooks: Security and privacy issues in cloud computing," in *Proc. 44th Hawaii Int. Conf. Syst. Sci.*, Kauai, HI, USA, 2011, pp. 1–10.
[35] S. Sood, "A combined approach to ensure data security in cloud computing," *J. Netw. Comput. Appl.*, vol. 35, no. 6, pp. 1831–1838, Nov. 2012.
[36] Y. Tang, P. Lee, J. Lui, and R. Perlman, "Secure overlay cloud storage with access control and assured deletion," *IEEE Trans. Dependable Secure Comput.*, vol. 9, no. 6, pp. 903–916, Nov./Dec. 2012.
[37] Q. Liu, G. Wang, and J. Wu, "Time-based proxy re-encryption scheme for secure data sharing in a cloud environment," *Inf. Sci.*, vol. 258, pp. 355–370, Feb. 2014.
[38] D. Linthicum, "Emerging hybrid cloud patterns," *IEEE Cloud Comput.*, vol. 3, no. 1, pp. 88–91, Jan./Feb. 2016.
[39] F. Metzger, T. Hoßfeld, A. Bauer, S. Kounev, and P. E. Heegaard, "Modeling of aggregated IoT traffic and its application to an IoT cloud," *Proc. IEEE*, vol. 107, no. 4, pp. 679–694, Apr. 2019.
[40] D. P. Heyman and M. J. Sobel, *Stochastic Models in Operations Research: Stochastic Processes and Operating Characteristics*. Mineola, NY, USA: Dover, 2003.
[41] L. Ramakrishnan and B. Plale, "A multi-dimensional classification model for scientific workflow characteristics," in *Proc. 1st Int. Workshop Workflow Approaches New Datacentric Sci.*, New York, NY, USA, 2010, pp. 1–12.
[42] J. Altmann and M. Kashef, "Cost model based service placement in federated hybrid clouds," *Future Gener. Comput. Syst.*, vol. 41, no. 1, pp. 79–90. Aug. 2014.
[43] A. Hernandez and E. Magana, "One-way delay measurement and characterization," in *Proc. Int. Conf. Netw. Serv.*, Athens, 2007, pp. 114–114.
[44] L. Vito, S. Rapuano, and L. Tomaciello, "One-way delay measure-ment: State of the art," *IEEE Trans. Instrum. Meas.*, vol. 57, no. 12, pp. 2742–2750, Dec. 2008.
[45] Q. Nguyen, P. Ghosh, and B. Krishnamachari, "End-to-end network performance monitoring for dispersed computing," in *Proc. Int. Conf. Comput. Netw. Commun.*, 2018, pp. 707–711.
[46] A. Toosi, R. Sinnott, and R. Buyya, "Resource provisioning for data-intensive applications with deadline constraints on hybrid clouds using aneka," *Future Gener. Comput. Syst.*, vol. 79, no. 2, pp. 765–775, Feb. 2018.
[47] S. Abdi, L. Pourkarimi, M. Ahmadi, and F. Zargari, "Cost minimization for deadline-constrained bag-of-tasks applications in federated hybrid clouds," *Future Gener. Comput. Syst.*, vol. 71, pp. 113–128, Jun. 2017.
[48] A. Papoulis and S. U. Pillai, *Probability, Random Variables and Stochastic Processes*. New York, NY, USA: McGraw-Hill, Dec. 2001.
[49] P. J. Burke, "The output of a queuing system," *Oper. Res.*, vol. 4, no. 6, pp. 699–704, Dec. 1956.
[50] D. Stoyan, W. Kendall, and J. Mecke, *Stochastic Geometry and Its Applications*. 3rd ed. New York, NY, USA: Wiley, Oct. 2013.
[51] A. Harel, "Sharp bounds and simple approximations for the Erlang delay and loss formulas," *Manage. Sci.*, vol. 34, no. 8, pp. 959–972, Aug. 1988.
[52] E. Chong and S. Zak, *An Introduction to Optimization*. 3rd ed. Hoboken, NJ, USA: Wiley, 2008.
[53] Amazon.com, "Pricing for Amazon EMR and Amazon EC2 (On-Demand)," Accessed: Apr. 14, 2020. [Online]. Available: https://aws.amazon.com/emr/pricing/

Younghwan Shin received the B.S. and the combined M.S. and Ph.D. degrees from the School of Electrical and Electronic Engineering, Yonsei University, Seoul, South Korea, in 2016 and 2022, respectively. He is currently a Research Member with Communications and Networking Laboratory, Yonsei University. His research interests include IoT, cloud computing, augmented reality, and deep learning for 5G and medical systems.

Wonsik Yang received the B.S. degree from the School of Electrical and Electronic Engineering, Yonsei University, Seoul, South Korea, where he is currently working toward the combined M.S. and Ph.D. degree in electrical and electronic engineering. He is currently a Research Member with Communications and Networking Laboratory, Yonsei University. His research interests include medical and 5G signal processing, deep learning, and cloud computing.

Sangdo Kim received the B.S. degree from the School of Electrical and Electronic Engineering, Yonsei University, Seoul, South Korea, where he is currently working toward the combined M.S. and Ph.D. degree in electrical and electronic engineering. He is currently a Research Member with Communications and Networking Laboratory, Yonsei University. His research interests include IoT, cloud computing, AR, and deep learning for 5G and medical systems.

Jong-Moon Chung (Senior Member, IEEE) received the B.S. and M.S. degrees in electronic engineering from Yonsei University, Seoul, South Korea, and the Ph.D. degree in electrical engineering from the Pennsylvania State University, State College, PA, USA. Since 2005, he has been a Professor with the School of Electrical and Electronic Engineering, Yonsei University, where he is currently the Associate Dean with the College of Engineering and a Professor with the Department of Emergency Medicine, College of Medicine, Yonsei University. From 1997 to 1999, he was an Assistant Professor and Instructor with the Department of Electrical Engineering, Pennsylvania State University. From 2000 to 2005, he was a tenured Associate Professor with the School of Electrical and Computer Engineering, Oklahoma State University (OSU), Stillwater, OK, USA. He is currently the Vice President of the IEEE CONSUMER TECHNOLOGY SOCIETY and IEEE PRODUCT SAFETY ENGINEERING SOCIETY, Editor of the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, Senior Editor for the IEEE TRANSACTIONS ON CONSUMER ELECTRONICS, Section Editor of the Wiley *ETRI Journal*, and Co-Editor-in-Chief of the *KSII Transactions on Internet and Information Systems*. In 2019 and 2021, he received the Award of Excellence from the Ministry of the Interior and Safety. In 2022, he received the Haedong Award of Excellence from the National Academy of Engineering of Korea. From Yonsei University, in 2019, 2018, and 2008, he received Outstanding Accomplishment Faculty Awards, and in 2021, 2019, 2014, 2009, and 2007, he received Outstanding Teaching Awards. In 2012, he received the Republic of Korea government's Defense Acquisition Program Administration Award. As a tenured Associate Professor with OSU, in 2005 he was the recipient of the Regents Distinguished Research Award, Halliburton Outstanding Young Faculty Award, and Technology Innovator Award and Distinguished Faculty Award from OSU, in 2004 and 2003, respectively. In 2000, he was the recipient of the First Place Outstanding Paper Award at the IEEE EIT 2000 conference held in Chicago, USA. He is also a pledge book Award Winning Member of the Eta Kappa Nu Epsilon Chapter. In addition, Dr. Chung served as the General Co-Chair of IEEE ICCE 2022 and General Chair of several conferences including IEEE ICCE-Asia 2020 and IEEE MWSCAS 2011.