Cost-Efficient Request Scheduling and Resource Provisioning in Multiclouds for Internet of Things

Xin Chen, Yongchao Zhang, and Ying Chen, *Member, IEEE*

***Abstract*—To satisfy the increasingly complex demands of the Internet of Things (IoT) applications, multiclouds are a promising solution that can provide scalable, various, and abundant resources. However, in multiclouds, each cloud has its specific virtual machine (VM) type and pricing scheme. In addition, the request arrival, network bandwidth, and VM's price all vary with time and are hardly predicted. In such cases, the request scheduling and resource provisioning (RSRP) for cost efficiency becomes a highly challenging work. In this article, to capture the dynamics in the multiclouds environment, we formulate a stochastic optimization problem where the aim is to minimize the system cost and guarantee the IoT applications' queueing delay. By applying stochastic optimization theory, the original problem is transformed into a deterministic optimization problem in each slot, and then the deterministic problem is further decomposed into three independent subproblems. An online RSRP algorithm is devised to obtain these subproblems' optimal solutions. Mathematical analysis shows that RSRP can approach the optimal system cost while bounding the queueing delay, and make an arbitrary tradeoff between system cost and queueing delay as well. Moreover, trace-driven simulation results show the effectiveness of RRSP.**

***Index Terms*—Cost efficient, Internet of Things (IoT), multiclouds, request scheduling, resource provisioning.**

I. Introduction

With the rapid development of the Internet of Things (IoT) technology and the increasing popularity of IoT devices, more and more computing-hungry IoT applications become available [1]–[3]. However, due to the limited resources, it is very hard for IoT devices to process computing-hungry applications by themselves. Cloud computing is a popular solution to this issue, which can provide resources in a cost effective and elastic way [4], [5]. The capital and operational expenditure are not required by users to build and maintain a computing system, and they just need to rent resources from the cloud providers (such as Amazon EC2 and Microsoft Azure). In addition, the resources rented from clouds can be dynamically scaled up or down according to the requesters' demand. Infrastructure-as-a-Service (IaaS), where clouds furnish virtual machines (VMs) basically, is one major type of cloud services [6].

However, there are some problems when using only one single cloud. First, a single cloud may not have sufficient resources to offer services for all the IoT applications from geographically different regions, and the cloud sometimes needs to fix the system vulnerabilities, which will result in a serious performance degradation if there is only one cloud offering services [7]. Thus, in order to provide continuous and stable services, cloud providers usually deploy servers in different regions. Second, the pricing schemes are not only time-varying but also different in different clouds, therefore, deploying applications in one single cloud may not be the most economical way. It is because sometimes other clouds may have a lower price. Third, according to [8], with the increasing number in cloud providers, IoT applications are willing to be launched in different cloud providers to avoid the provider lock-in. Therefore, it is more cost effective and stable to deploy IoT applications in multiclouds.

In multiclouds, in order to provide IoT applications with a good quality of service in a cost-effective manner [9], a careful and well-designed request scheduling and resource provisioning (RSRP) strategy is very critical. First, with the dramatic increase in the amount of IoT devices, a massive amount of data is generated and needs to be uploaded to clouds. But the network bandwidth of each cloud is often limited [10], [11]. Thus, the RSRP should take into account the network bandwidth. For the cloud with low network bandwidth, scheduling too many requests will cause network congestion, and renting lots of resources will lead to a large waste of resources. Second, the existing clouds generally have two basic resource pricing options (i.e., reserved pricing and on-demand pricing). Reserved pricing is more economical to deal with the long-term demand, while on-demand pricing is more suitable for short-term demand. Resource provisioning should be decided properly based on the application requests. Third, the VM's price varies across time. Postponing the request processing until the VM's price becomes lower can help reduce the cost, but the IoT applications' performance may be degraded.

It is of many difficulties and challenges in multiclouds to devise such a strategy. On the one hand, IoT application's request arrival, network bandwidth, and VM's price in each cloud are all highly dynamic and hardly predicted. How to jointly optimize the RSRP adapting to the highly

Manuscript received July 10, 2019; revised September 7, 2019; accepted October 4, 2019. Date of publication October 21, 2019; date of current version March 12, 2020. This work was supported in part by the National Natural Science Foundation of China under Grant 61872044 and Grant 61902029, in part by the Key Research and Cultivation Projects at Beijing Information Science and Technology University under Grant 5211910958, and in part by the Supplementary and Supportive Project for Teachers at Beijing Information Science and Technology University under Grant 5111911128. *(Corresponding author: Ying Chen.)*

The authors are with the Computer School, Beijing Information Science and Technology University, Beijing 100101, China (e-mail: chenxin@bistu.edu.cn; zhangyongchao@mail.bistu.edu.cn; chenying@bistu.edu.cn).

Digital Object Identifier 10.1109/JIOT.2019.2948432

dynamic environment is a challenging work. On the other hand, every cloud has its network bandwidth, pricing scheme, and resource configuration. These differences among clouds further complicate the RSRP.

There have been some existing works in task scheduling and resource allocation for cloud computing. References [12]–[15] studied the task scheduling and resource allocation for multiclouds. But they all focused on the static environment or assumed that the arrival pattern followed the fixed distributions, which resulted in that their methods could not well apply to the real scene. Alsarhan *et al.* [16] investigated the VM provisioning problem for the multiservice environment and maximized the cloud gain while guaranteeing all the clients' quality of service. But they only focused on the single cloud without the multiple clouds whose resource provisioning strategy is more complex than that of a single cloud. Xiao *et al.* [17] studied the request redirection and resource provisioning for the video service and formulated a cost minimization optimization problem. But the authors only considered a single pricing approach and did not take the network bandwidth into account. In this article, we consider both the network bandwidth capacity and different pricing schemes and pay attention to the highly dynamic multiclouds environment.

This article studies the RSRP in multiclouds for IoT and tries to minimize the system cost while guaranteeing the queueing delay of IoT application. A stochastic optimization problem is formulated to capture the high dynamics in request arrival, network bandwidth, and VM's price. Then, we transform the original stochastic optimization problem into a deterministic optimization problem and decompose the transformed problem into three independent subproblems. An online RSRP algorithm is designed to get the optimal solutions of these subproblems. We theoretically analyze the performance of RSRP and also evaluate RSRP using the real traces.

The remainder of this article is as follows. In Section II, we present the system model and formulate an optimization problem. Section III proposes an online algorithm, RSRP, to solve this problem. Section IV presents RSRP's performance through mathematical analysis. Section V gives the simulation results. In Section VI, we conclude this article.

II. System Model

Consider N IoT applications index by $\mathcal{N} = \{1, 2, \ldots, N\}$, and M geo-distributed clouds indexed by $\mathcal{M} = \{1, 2, \ldots, M\}$. Each cloud has the VMs which can process all the IoT applications on behalf of the IoT devices [18]. Two different pricing options for VM instances are provided in each cloud, including a reserved pricing approach and an on-demand approach [19]. These clouds are supported by different cloud providers, and their processing capacities and VMs' prices vary from each other. Similar to many previous works [13], [17], [20], a cloud provider agent (CPA) is introduced to receive IoT application requests, rent VM instances from the clouds and schedule the arrived requests to the clouds. The system operates in a time-slotted manner, which is indexed by $t = 0, 1, \ldots, T$ [17], [21]. The slot length is equal to τ. At the beginning of each slot, CPA observes the system state (such as arrived IoT application requests, VM's prices, and network bandwidth), and then makes the request scheduling (i.e., determines the number of IoT application requests scheduled to each cloud) and resource provisioning (i.e., determines the number of each type VM instances rented from each cloud) decisions to optimize the cost (i.e., the total VM instances' rental). In Table I, the main notations in the system model are provided.

TABLE I
Notations and Definitions

Notion	Definition
$\mathcal{N}$	IoT applications set
$\mathcal{M}$	Geo-distributed clouds set
$A_i(t)$	The number of requests from i-th IoT application
$C_j(t)$	The bandwidth between CPA and cloud j
$a_{ij}(t)$	The number of i-th IoT application's requests scheduled to cloud j
u_i	The data size of unit i-th IoT application's request
$b_{ij}^r(t)$	The processing rate of the reserved VM instance
$b_{ij}^o(t)$	The processing rate of the on-demand VM instance
$d_{ij}^r(t)$	The number of reserved VM instances rented for i-th IoT application in cloud j
$d_{ij}^o(t)$	The number of on-demand VM instances rented for i-th IoT application in cloud j
$p_{ij}^r(t)$	The price of reserved VM instance for one billing cycle
$p_{ij}^o(t)$	The price of on-demand VM instance for one billing cycle
$P(t)$	The cost for renting the new VM instances
$Q_{ij}(t)$	The queue length of the i-th IoT application in cloud j

A. Application Request Model

Let $A_i(t)$ represent the number of requests from the ith IoT application within slot t. $A_i(t)$ varies over time slots and is independent among the N IoT applications. Notice that there may exist bursty request arrival for each IoT application. Let $a_{ij}(t) \in \{0, 1, 2, \ldots\}$ denote the number of ith IoT application's requests scheduled to cloud j at slot t. We have

$$\sum_{j=1}^{M} a_{ij}(t) = A_i(t) \quad \forall i \in \mathcal{N}. \tag{1}$$

It means that for each IoT application, the total number of requests allocated to all the clouds should be equal to the number of arrived requests.

In addition, the network bandwidth between CPA and cloud is usually limited [22]. Let $C_j(t)$ denote the bandwidth between CPA and cloud j at slot t. $C_j(t)$ varies over time slots due to the fluctuation in the system network environment. Since both the geographic location and network infrastructure of each cloud are different from others, the network bandwidth of each cloud is obviously different. At each slot, the requests' number scheduled to each cloud cannot exceed its network

bandwidth [23], which is expressed by

$$\sum_{i=1}^{N} u_i a_{ij}(t) \le C_j(t) \quad \forall j \in \mathcal{M} \tag{2}$$

where u_i represents the data size of unit ith IoT application's request. Notice that we can see that our model does not require any prior statistical information about the request arrival and network bandwidth.

B. VM Provisioning Model

For the VMs running ith IoT application in cloud j, let $b_{ij}^r(t)$ denote the processing rate (i.e., the number of requests which can be processed successfully) of each reserved VM instance during slot t, and $b_{ij}^o(t)$ denote the processing rate of each on-demand VM instance. At each slot, CPA makes the decisions to determine how many VM instances will be rented to process the received IoT application requests. Let $d_{ij}^r(t)$ denote the number of reserved VM instances rented for ith IoT application in cloud j at slot t, and $d_{ij}^o(t)$ denote the number of on-demand VM instances rented for ith IoT application in cloud j at slot t. Then, the total processing rate of rented VM instances for IoT application i in cloud j is

$$k_{ij}(t) = l_{ij}(t) + d_{ij}^r(t) b_{ij}^r(t) + d_{ij}^o(t) b_{ij}^o(t) \tag{3}$$

where $l_{ij}(t)$ denotes the total processing rate of the available VM instances which are rented before slot t. Let $d_{ij}^{r,\max}$ denote the maximum number of reserved VM instances CPA can rent per time slot. Similarly, let $d_{ij}^{o,\max}$ represent the maximum number of on-demand VM instances CPA can rent per time slot. Therefore, in each slot, the number of newly rented reserved and on-demand VM instances cannot exceed their limits, which are

$$0 \le d_{ij}^r(t) \le d_{ij}^{r,\max} \quad \forall t \in \{0, 1, \ldots, T\} \tag{4}$$

$$0 \le d_{ij}^o(t) \le d_{ij}^{o,\max} \quad \forall t \in \{0, 1, \ldots, T\}. \tag{5}$$

For the VMs running ith IoT application in cloud j, let $p_{ij}^r(t)$ represent the price of reserved VM instance for one billing cycle, and $p_{ij}^o(t)$ represent the price of on-demand VM instance for one billing cycle. Generally, the billing cycle of the reserved VM instance is much longer than the on-demand. For example, in Amazon EC2, the minimum billing cycle of the reserved VM instance is about one year, and the on-demand VM instance's billing cycle is 1 h. Without loss of generality, the billing cycle of the reserved VM instance is W times that of the on-demand VM instance [24]. Inspired by the previous works [17], [24], the slot length in the article is set to be equal to the billing cycle of on-demand VM instance. The VM's price in each cloud may change with time, and the VM's prices among the different clouds are also different. Let $P(t)$ denote the cost for renting the new VM instances in slot t, which can be obtained by

$$P(t) = \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{M}} \Big[p_{ij}^r(t) d_{ij}^r(t) + p_{ij}^o(t) d_{ij}^o(t) \Big]. \tag{6}$$

C. Request Queueing Model

For each type of IoT application, every cloud maintains a queue to store the received but not yet processed requests. Let $Q_{ij}(t)$ denote the queue length (i.e., the number of requests in the queue) of the ith IoT application in cloud j. Recall that the number of newly arrived ith IoT application's requests in cloud j is $a_{ij}(t)$, and the maximum number of the ith IoT application requests cloud j can process is $k_{ij}(t)$. Thus, for the ith IoT application in cloud j, the update of its queue length can be expressed by

$$Q_{ij}(t+1) = \max\big\{Q_{ij}(t) - k_{ij}(t), 0\big\} + a_{ij}(t). \tag{7}$$

The long-term time average queue length of ith IoT application in cloud j is

$$\bar{Q}_{ij} = \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbf{E}\big\{Q_{ij}(t)\big\}. \tag{8}$$

According to Little's law [25], queueing delay is proportional to the queue length. A larger queue length leads to a longer queueing delay. In other works, when the queue length of each IoT application can be upper bounded, the corresponding queueing delay can be guaranteed as well. In such a context, instead of considering the queueing delay directly, we try to bound the time average queue length of each application, which is

$$\bar{Q}_{ij} = \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbf{E}\big\{Q_{ij}(t)\big\} \le \varsigma, \quad \exists\ \varsigma \in \mathbb{R}^+. \tag{9}$$

D. Optimization Problem

From the perspective of IoT application, it is obvious that there exists a tradeoff between the cost and performance (i.e., queueing delay). Specifically, improving performance requires renting more VMs, which results in a larger cost. On the contrary, it will lead to severe performance degradation when excessively reducing the cost. In such a context, this article aims at reducing the cost as much as possible while guaranteeing the performance of each IoT application. Therefore, the goal of the formulated optimization problem is to minimize the time average system cost in long time, meanwhile bound the queue length of each IoT application. The long-term time average cost can be expressed by

$$\begin{aligned} \bar{P} &= \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbf{E}\{P(t)\} \\ &= \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbf{E}\left\{ \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{M}} \Big[p_{ij}^r(t) d_{ij}^r(t) + p_{ij}^o(t) d_{ij}^o(t) \Big] \right\}. \end{aligned} \tag{10}$$

Then, the form of the formulated optimization problem **P1** is as follows:

$$\mathbf{P1:} \min_{a_{ij}(t), d_{ij}^r(t), d_{ij}^o(t)} \bar{P} \tag{11}$$

subject to constraints (1), (2), (4), (5), and (9).

Obviously, **P1** is a stochastic optimization problem because many variables in this optimization problem [such as $A_i(t), C_j(t), p_{ij}^r(t), p_{ij}^o(t), b_{ij}^r(t)$, and $b_{ij}^o(t)$] are all time-varying and stochastic. It is of much challenges for a strategy to make the optimal RSRP decisions without any prior statistical information about these stochastic variables. To tackle this challenge, the stochastic optimization technique is applied in the following section.

III. Algorithm Design

For **P1**, we can find that deriving its offline solution is an extremely difficult work because this process requires the future information of the stochastic variables. How to optimize the RSRP decisions to adapt to the dynamic environment is of great challenge. In such a context, we take the advantage of the Lyapunov optimization techniques to solve this problem. In this section, based on a Lyapunov function, we first transform the stochastic optimization problem within a long period to a deterministic problem under each time slot. Then, we propose an online algorithm that makes the decisions based on only the current information.

A. Problem Transformation

Following the framework of the Lyapunov optimization approach, let $\Theta(t)$ denote the queue length matrix of all the N IoT applications in all the M clouds, and $L(\Theta(t))$ denote the Lyapunov function which is calculated by

$$L(\Theta(t)) = \frac{1}{2}\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{M}} Q_{ij}^2(t). \tag{12}$$

From (12), we can observe that even if only one queue length is large, the value of $L(\Theta(t))$ will also be large. In other words, only when all the queue lengths are small enough, the value of $L(\Theta(t))$ will be small. Therefore, the value of $L(\Theta(t))$ can reflect the queueing backlog state in the system [26]. In this case, we reduce the value of $L(\Theta(t))$ for keeping the system at a low queueing backlog. Then, let $\Delta(\Theta(t))$ denote *Lyapunov drift* function, which is

$$\Delta(\Theta(t)) = \mathbf{E}\{L(\Theta(t+1)) - L(\Theta(t))|\Theta(t)\}. \tag{13}$$

It is evident that if the value of $\Delta(\Theta(t))$ can be minimized in each slot, the value of *Lyapunov drift* function in slot $t+1$ will be smaller than the last slot t. Correspondingly, the queueing backlog of the system will tend to be lower, and the performance of each IoT application can be improved.

Recall that the goal of the optimization problem is to minimize the cost and bound the IoT applications' queueing delay. Therefore, according to the Lyapunov optimization theory, we define the *drift-plus-cost* function as

$$\Delta(\Theta(t)) + V\mathbf{E}\{P(t)|\Theta(t)\} \tag{14}$$

where V is a parameter used to adjust the tradeoff between the cost and queue length. Increasing V means that more weight is put on cost. Without loss of generality, the value of V is greater than or equal to zero.

In the following, we give the supremum bound of *drift-plus-cost* function.

Theorem 1: If existing $A_i^{\max} \geq A_i(t)$, $b_{ij}^{r,\max} \geq b_{ij}^r(t)$ and $b_{ij}^{o,\max} \geq b_{ij}^o(t)$ for any $t \in \{0, 1, \ldots\}$, for any value of $\Theta(t)$, the value of *drift-plus-cost* function under any feasible strategy will satisfy

$$\begin{aligned}\Delta(\Theta(t)) + V\mathbf{E}\{P(t)|\Theta(t)\} &\leq Y \\ &+ V\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{M}}\mathbf{E}\Big\{p_{ij}^r(t)d_{ij}^r(t) + p_{ij}^o(t)d_{ij}^o(t)|\Theta(t)\Big\} \\ &+ \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{M}} Q_{ij}(t)\mathbf{E}\big\{a_{ij}(t) - k_{ij}(t)|\Theta(t)\big\}.\end{aligned} \tag{15}$$

Here, $Y = (1/2)\sum_{i\in\mathcal{N}}(A_i^{\max})^2 + (1/2)\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{M}}[Mb_{ij}^{r,\max}d_{ij}^{r,\max} + b_{ij}^{o,\max}d_{ij}^{o,\max}]$ is a constant.

Proof: According to (7), we have

$$\begin{aligned}Q_{ij}^2(t+1) &= \big(\max\big\{Q_{ij}(t) - k_{ij}(t), 0\big\} + a_{ij}(t)\big)^2 \\ &= \big(\max\big\{Q_{ij}(t) - k_{ij}(t), 0\big\}\big)^2 + a_{ij}^2(t) \\ &\quad + 2a_{ij}(t)\max\big\{Q_{ij}(t) - k_{ij}(t), 0\big\}.\end{aligned} \tag{16}$$

As $(\max\{a-b,0\})^2 \leq (a-b)^2$ and if $a, b \geq 0$, $\max\{a-b,0\} \leq a$, we then have

$$\begin{aligned}Q_{ij}^2(t+1) &\leq \big(Q_{ij}(t) - k_{ij}(t)\big)^2 + a_{ij}^2(t) + 2a_{ij}(t)Q_{ij}(t) \\ &\leq Q_{ij}^2(t) + a_{ij}^2(t) + k_{ij}^2(t) + 2Q_{ij}(t)\big(a_{ij}(t) - k_{ij}(t)\big).\end{aligned} \tag{17}$$

Rearranging (17), it holds

$$\begin{aligned}\frac{Q_{ij}^2(t+1) - Q_{ij}^2(t)}{2} &\leq \frac{a_{ij}^2(t) + k_{ij}^2(t)}{2} \\ &\quad + Q_{ij}(t)\big(a_{ij}(t) - k_{ij}(t)\big).\end{aligned} \tag{18}$$

Summing (18) over sets $\mathcal{N}$ and $\mathcal{M}$, and taking conditional expectations, we obtain

$$\begin{aligned}&\mathbf{E}\{L(\Theta(t+1)) - L(\Theta(t))|\Theta(t)\} \\ &\quad\leq \frac{1}{2}\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{M}}\mathbf{E}\Big\{a_{ij}^2(t) + k_{ij}^2(t)|\Theta(t)\Big\} \\ &\qquad+\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{M}} Q_{ij}(t)\mathbf{E}\big\{a_{ij}(t) - k_{ij}(t)|\Theta(t)\big\}.\end{aligned} \tag{19}$$

Since $\sum_{j=1}^M a_{ij}(t) = A_i(t) \leq A_i^{\max}$, according to $(a+b)^2 \geq a^2 + b^2$, we can obtain

$$\sum_{j=1}^{M} a_{ij}^2(t) \leq \big(A_i^{\max}\big)^2. \tag{20}$$

Additionally, recall that $b_{ij}^r(t) \leq b_{ij}^{r,\max}$, $d_{ij}^r(t) \leq d_{ij}^{r,\max}$, $b_{ij}^o(t) \leq b_{ij}^{o,\max}$, and $d_{ij}^o(t) \leq d_{ij}^{o,\max}$, then we can obtain

$$k_{ij}^2(t) \leq Mb_{ij}^{r,\max}d_{ij}^{r,\max} + b_{ij}^{o,\max}d_{ij}^{o,\max}. \tag{21}$$

Combining with (20) and (21), it yields

$$\begin{aligned}\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{M}}\mathbf{E}\Big\{a_{ij}^2(t) + k_{ij}^2(t)|\Theta(t)\Big\} &\leq \sum_{i\in\mathcal{N}}\big(A_i^{\max}\big)^2 \\ &+\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{M}}\Big[Mb_{ij}^{r,\max}d_{ij}^{r,\max} + b_{ij}^{o,\max}d_{ij}^{o,\max}\Big].\end{aligned} \tag{22}$$

Then, let $Y = (1/2)\sum_{i\in\mathcal{N}}(A_i^{\max})^2 + (1/2)\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{M}}[Mb_{ij}^{r,\max}d_{ij}^{r,\max} + b_{ij}^{o,\max}d_{ij}^{o,\max}]$, and (19) can be rewritten as

$$\Delta(\Theta(t)) \le Y + \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{M}} Q_{ij}(t)\mathbf{E}\big\{(a_{ij}(t) - k_{ij}(t))|\Theta(t)\big\}. \quad (23)$$

Adding $V\mathbf{E}\{P(t)|\Theta(t)\}$ to (23), and substituting (6), we can obtain (15) in Theorem 1. ■

Following the Lyapunov optimization theory, instead of solving the original stochastic optimization problem **P1** directly, we minimize the upper bound of *drift-plus-cost* in each slot, which is a deterministic optimization problem. This deterministic optimization problem can be written as

$$\begin{aligned}\min\ & Y + V\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{M}}\Big[p_{ij}^r(t)d_{ij}^r(t) + p_{ij}^o(t)d_{ij}^o(t)\Big] \\ & + \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{M}} Q_{ij}(t)\big[a_{ij}(t) - k_{ij}(t)\big] \end{aligned} \quad (24)$$

subject to constraints (1), (2), (4), and (5).

Recall that $k_{ij}(t) = l_{ij}(t) + d_{ij}^r(t)b_{ij}^r(t) + d_{ij}^o(t)b_{ij}^o(t)$. Since Y, $Q_{ij}(t)$, and $l_{ij}(t)$ are all constants in each slot, problem (24) is equivalent to the following optimization problem **P2**:

$$\begin{aligned}\textbf{P2:}\ \min\ & \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{M}} Q_{ij}(t)a_{ij}(t) \\ & + \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{M}} d_{ij}^r(t)\Big(Vp_{ij}^r(t) - Q_{ij}(t)b_{ij}^r(t)\Big) \\ & + \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{M}} d_{ij}^o(t)\Big(Vp_{ij}^o(t) - Q_{ij}(t)b_{ij}^o(t)\Big) \end{aligned} \quad (25)$$

subject to constraints (1), (2), (4), and (5).

We can observe in (25) that it only needs the current system state information (such as to request arrival, VM's price, and processing rate) to get the optimal solutions of **P2**, while solving **P1** also requires the future system information. Therefore, the transformed problem **P2** is more trackable than the original problem **P1**. In the following, we try to design an optimal algorithm to get the optimal solutions to **P2**.

B. Request Scheduling and Resource Provisioning Algorithm

Here, an online RSRP algorithm is designed to solve **P2**. We can easily observe that there are three variables to be solved in **P2**, which are $a_{ij}(t)$, $d_{ij}^r(t)$, and$d_{ij}^o(t)$. It should be noticed that $a_{ij}(t)$, $d_{ij}^r(t)$, and$d_{ij}^o(t)$ are all decoupled in the objective and constraints of **P2**. So, **P2** is able to be decomposed into three independent subproblems. More specifically, these three subproblems are request scheduling, reserved VM provisioning, and on-demand VM provisioning, respectively. In the next, the optimal solutions to these subproblems are obtained separately.

1) Request Scheduling: Considering the first term of objective in **P2** and constraint (1) and (2), and letting $\tilde{Q}_{ij}(t) = ((Q_{ij}(t))/u_i)$, the form of the request scheduling optimization subproblem is as follows:

$$\begin{aligned}\min\ & \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{M}} \tilde{Q}_{ij}(t)u_i a_{ij}(t) \\ \text{s.t.}\ & \sum_{j=1}^{M} u_i a_{ij}(t) = u_i A_i(t) \quad \forall i \in \mathcal{N} \\ & \sum_{i=1}^{N} u_i a_{ij}(t) \le C_j(t) \quad \forall j \in \mathcal{M}. \end{aligned} \quad (26)$$

In problem (26), $u_i a_{ij}(t)$ can be treated together as a single variable. Obviously, problem (26) is a simple integer linear programming problem. An optimal request scheduling algorithm, as shown in Algorithm 1, is devised to obtain the optimal solutions.

Algorithm 1 Optimal Request Scheduling Algorithm

```
Get the current queue lengths of all the IoT applications
Sort the queues of all the IoT applications in all the clouds in the
  ascending order of Q̃_ij(t), and the set of sorted queues is indexed by
  Q̃ = {Q̃^1_ij(t), Q̃^2_ij(t), ··· , Q̃^NM_ij(t)} where Q̃^1_ij(t) ≤ Q̃^2_ij(t)
Set a_ij(t) = 0 for all i ∈ N and j ∈ M, and u = 1
while Σ_{j=1}^{M} a_ij(t) ≠ A_i(t) for all i ∈ N do
    Select the u-th queue in Q and obtain the corresponding i and j
    Set a_ij(t) = A_i(t) − Σ_{j=1}^{M} a_ij(t)
    if Σ_{i=1}^{M} u_i a_ij(t) > C_j(t) then
        a_ij(t) = ⌊(C_j(t) − Σ_{i=1}^{M} u_i a_ij(t)) / u_i⌋ + a_ij(t)
    end if
    u = u + 1
end while
```

Remark: It can be seen that the main idea of Algorithm 1 is to schedule requests as many as possible to the queue with a shorter length. This is because scheduling more requests to the queue with shorter length can help keep all the queues stable. Moreover, for each type of application, if its queue length in one cloud is smaller than other clouds, it means that this cloud's processing rate is larger than others. In this case, it is obviously reasonable for Algorithm 1 to schedule more requests to the cloud with a larger processing rate.

2) Reserved VM Provisioning: Considering the second term of objective in **P2** and constraint (4), the form of the reserved VM provisioning optimization subproblem is as follows:

$$\begin{aligned}\min\ & \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{M}} d_{ij}^r(t)\Big(Vp_{ij}^r(t) - Q_{ij}(t)b_{ij}^r(t)\Big) \\ \text{s.t.}\ & 0 \le d_{ij}^r(t) \le d_{ij}^{r,\max} \quad \forall t \in \{0, 1, \ldots, T\}. \end{aligned} \quad (27)$$

Problem (27) can be considered as a generalized min-weight problem. Its optimal solution is as follows:

$$d_{ij}^r(t) = \begin{cases} d_{ij}^{r,\max}, & Q_{ij}(t)b_{ij}^r(t) - Vp_{ij}^r(t) > 0 \\ 0, & \text{otherwise.} \end{cases} \quad (28)$$

Remark: Intuitively, if a VM has the lower price and larger processing rate, the profit of renting this VM is larger. Therefore, in (28), $G_{ij}^r(t) = Q_{ij}(t)b_{ij}^r(t) - Vp_{ij}^r(t)$ can be regarded as the *renting profit* of the reserved VM instance. We can find that when processing ability $b_{ij}^r(t)$ rises or price $p_{ij}^r(t)$ decreases, the value of $G_{ij}^r(t)$ increases. We can also see that the value of $G_{ij}^r(t)$ also depends on $Q_{ij}^r(t)$ and V. When

Algorithm 2 Online RSRP Algorithm

1: Observe the current queue length of each IoT application in every cloud, i.e., $Q_{ij}(t)$
2: Obtain $a_{ij}(t)$ for all $i \in \mathcal{N}$ and $j \in \mathcal{M}$ using Algorithm 1
3: **for all** $i \in \mathcal{N}$ and $j \in \mathcal{M}$ **do**
4: Compute $G^r_{ij}(t) = Q_{ij}(t)b^r_{ij}(t) - Vp^r_{ij}(t)$
5: **if** $G^r_{ij}(t) > 0$ **then**
6: Set $d^r_{ij}(t) = d^{r,max}_{ij}$
7: **else**
8: Set $d^r_{ij}(t) = 0$
9: **end if**
10: Compute $G^o_{ij}(t) = Q_{ij}(t)b^o_{ij}(t) - Vp^o_{ij}(t)$
11: **if** $G^o_{ij}(t) > 0$ **then**
12: Set $d^o_{ij}(t) = d^{o,max}_{ij}$
13: **else**
14: Set $d^o_{ij}(t) = 0$
15: **end if**
16: **end for**

$Q_{ij}(t)$ rises, the *renting profit* $G^r_{ij}(t)$ also increases. The reason is that a large $Q_{ij}(t)$ means that the processing rate of the VM instances which have been rented is smaller than the request arrival rate. To guarantee the IoT application's performance, more VM instances need to be rented. To this end, a larger *renting profit* will help increase the willingness to buy more VM instances. Moreover, increasing V will reduce *renting profit* $G^r_{ij}(t)$. It is because larger V represents that more weight is put on cost. In order to reduce the cost, the *renting profit* will decrease.

3) Reserved VM Provisioning: Considering the last term of objective in **P2** and constraint (5), the form of the on-demand VM provisioning optimization subproblem is as follows:

$$\begin{aligned} \min \quad & \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{M}} d^o_{ij}(t)\Big(Vp^o_{ij}(t) - Q_{ij}(t)b^o_{ij}(t)\Big) \\ \text{s.t.} \quad & 0 \le d^o_{ij}(t) \le d^{o,\max}_{ij} \quad \forall t \in \{0, 1, \ldots, T\}. \end{aligned} \tag{29}$$

Similar to problems (27) and (29), it can also be considered as a generalized min-weight problem. Its optimal solution is as follows:

$$d^o_{ij}(t) = \begin{cases} d^{o,\max}_{ij}, & Q_{ij}(t)b^o_{ij}(t) - Vp^o_{ij}(t) > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{30}$$

Remark: Like the aforementioned $G^r_{ij}(t)$, in (30), $G^o_{ij}(t) = Q_{ij}(t)b^o_{ij}(t) - Vp^o_{ij}(t)$ can be regarded as the *renting profit* of the on-demand VM instance. When $b^o_{ij}(t)$ or $Q_{ij}(t)$ rises, the value of $G^o_{ij}(t)$ will increase. When V or $p^o_{ij}(t)$ rises, the value of $G^o_{ij}(t)$ will reduce. For the above two optimal solutions, their main idea is to rent the VMs with as much positive *renting profit* as possible. Through maximizing the total renting profit, both the cost and queue length can be reduced.

According to the solution results of the above subproblems, RSRP is proposed to solve **P2** in an optimal way. RSRP is summarized in detail in Algorithm 2.

IV. Algorithm Analysis

In this section, we theoretically analyze the performance of RSRP. The analysis results show that RSRP is able to approximate the minimum cost arbitrarily, and the time average queue length can also be stabilized.

Let $\bar{Q}$ represent the time average queue length of all the IoT applications in all clouds, which is as follows:

$$\bar{Q} = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{i=1}^{N} \sum_{j=1}^{M} \mathbf{E}\big\{Q_{ij}(t)\big\}. \tag{31}$$

To help analyze the performance of RSRP, we first give Lemma 1.

Lemma 1: If **P1** is feasible, given $\varepsilon > 0$, there exists a randomized and optimal strategy π independent of the queue length which satisfies the following:

$$\begin{aligned} &\mathbf{E}\big\{P^\pi(t)\big\} = P^*(\varepsilon) \\ &\mathbf{E}\Big\{a^\pi_{ij}(t)\Big\} \le \mathbf{E}\Big\{k^\pi_{ij}(t)\Big\} - \varepsilon \end{aligned}$$

where $P^*(\varepsilon)$ is the minimum cost with ε.

Proof: Lemma 1 can be proven using the Caratheodorys theorem [27]; the detailed proof is omitted here for brevity. ■

Let P^{RSRP} and $\bar{Q}^{\mathrm{RSRP}}$ denote the time average cost and queue length obtained by RSRP, respectively. We present the upper bounds of P^{RSRP} and $\bar{Q}^{\mathrm{RSRP}}$ in Theorem 2.

Theorem 2: Under the same assumptions as in Lemma 1, for any value of V, both P^{RSRP} and $\bar{Q}^{\mathrm{RSRP}}$ can be bounded, which are as follows:

$$P^{\mathrm{RSRP}} \le P^* + \frac{Y}{V} \tag{32}$$

$$\bar{Q}^{\mathrm{RSRP}} \le \frac{Y + V(P^*(\varepsilon) - P^*)}{\varepsilon} \tag{33}$$

where P^* is the minimum cost of **P1**, and Y is the constant defined in Theorem 1.

Proof: Recall that RSRP can minimize the upper bound of *drift-plus-cost*. Therefore, the value of *drift-plus-cost* under RSRP will certainly be not greater than other feasible policies including the optimal policy of **P1**. Then, we have

$$\begin{aligned} &\Delta\Big(\Theta^{\mathrm{RSRP}}(t)\Big) + V\mathbf{E}\Big\{P^{\mathrm{RSRP}}(t)|\Theta^{\mathrm{RSRP}}(t)\Big\} \le Y \\ &\quad + V\mathbf{E}\Big\{P^\pi(t)|\Theta^{\mathrm{RSRP}}(t)\Big\} \\ &\quad + \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{M}} Q^{\mathrm{RSRP}}_{ij}(t)\mathbf{E}\Big\{\Big(a^\pi_{ij}(t) - k^\pi_{ij}(t)\Big)|\Theta^{\mathrm{RSRP}}(t)\Big\}. \end{aligned} \tag{34}$$

Applying Lemma 1, we can obtain

$$\begin{aligned} &\Delta\Big(\Theta^{\mathrm{RSRP}}(t)\Big) + V\mathbf{E}\Big\{P^{\mathrm{RSRP}}(t)|\Theta^{\mathrm{RSRP}}(t)\Big\} \\ &\quad \le Y + VP^*(\varepsilon) - \varepsilon\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{M}} Q^{\mathrm{RSRP}}_{ij}(t). \end{aligned} \tag{35}$$

Using (13), taking expectations on (35) and summing over slots, we can obtain

$$\begin{aligned} &\mathbf{E}\Big\{L\Big(\Theta^{\mathrm{RSRP}}(T)\Big)\Big\} - \mathbf{E}\Big\{L\Big(\Theta^{\mathrm{RSRP}}(0)\Big)\Big\} + V\sum_{t=0}^{T-1}\mathbf{E}\Big\{P^{\mathrm{RSRP}}(t)\Big\} \\ &\quad \le YT + VTP^*(\varepsilon) - \varepsilon\sum_{t=0}^{T-1}\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{M}}\mathbf{E}\Big\{Q^{\mathrm{RSRP}}_{ij}(t)\Big\}. \end{aligned} \tag{36}$$

In general, $L(\Theta(0)) = 0$ and $L(\Theta(T)) \geq 0$. Then, we have

$$V\sum_{t=0}^{T-1}\mathbf{E}\left\{P^{\mathrm{RSRP}}(t)\right\} \leq YT + VTP^*(\varepsilon) - \varepsilon\sum_{t=0}^{T-1}\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{M}}\mathbf{E}\left\{Q_{ij}^{\mathrm{RSRP}}(t)\right\}. \quad (37)$$

According to (37), since ε and $Q_{ij}^{\mathrm{RSRP}}(t)$ are both nonnegative, we can obtain

$$V\sum_{t=0}^{T-1}\mathbf{E}\left\{P^{\mathrm{RSRP}}(t)\right\} \leq YT + VTP^*(\varepsilon). \quad (38)$$

Dividing (38) by VT, and letting $\varepsilon \to 0$, $T \to \infty$, we have (32) in Theorem 2.

In addition, dividing (37) by εT, we can obtain

$$\frac{1}{T}\sum_{t=0}^{T-1}\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{M}}\mathbf{E}\left\{Q_{ij}^{\mathrm{RSRP}}(t)\right\} \leq \frac{Y + VP^*(\varepsilon)}{\varepsilon} - \frac{V}{\varepsilon}\frac{1}{T}\sum_{t=0}^{T-1}\mathbf{E}\left\{P^{\mathrm{RSRP}}(t)\right\}. \quad (39)$$

Letting $T \to \infty$, and since $P^{\mathrm{RSRP}} \geq P^*$, we can obtain (33) in Theorem 2. ■

Remark: From (32) in Theorem 2, we can find that the gap of cost between RSRP and the optimal policy is upper bounded by (Y/V). With the increase of V, this gap will reduce, and when V is sufficiently large, (Y/V) will be close to zero. In other words, RSRP can achieve the minimum cost with a sufficiently large V. In (33), we can observe that the average queue length of RSRP can be upper bounded. It means that RSRP can keep the queue stable all the time, and thus guarantee the queueing delay of all the IoT applications according to Little's law. In addition, combining with (32) and (33), RSRP can also achieve an arbitrary tradeoff between cost and queue length by tuning V.

Next, we analyze the time complexity of RSRP. To this end, we first analyze the time complexity of Algorithm 1. Specifically, for line 2 in Algorithm 1, it will take $O(MN\log_2(MN))$ operations to sort all the IoT applications in all the clouds in the ascending order of $\tilde{Q}_{ij}(t)$. For lines 4–10, each queue will be traversed once in the worst case, then it terminates within $O(MN)$ operations in the worst case. Thus, we can have that the time complexity of Algorithm 1 is $O(MN\log_2(MN))$. Next we analyze the time complexity of RSRP (i.e., Algorithm 2). For line 2 in Algorithm 2, it will take $O(MN\log_2(MN))$ operations as derived above. For the following loop (lines 3–16), RSRP will traverse each queue in every cloud once. Therefore, this loop terminates in $O(MN)$ operations. In conclusion, we can obtain that the whole time complexity of RSRP is $O(MN\log_2(MN))$.

V. Evaluation

In this section, a series of numerical results are presented to show the efficiency of RSRP. We first introduce the simulation settings and then show the effects of several parameters on RSRP. Finally, we compare RSRP with two other algorithms in terms of cost and queue length.

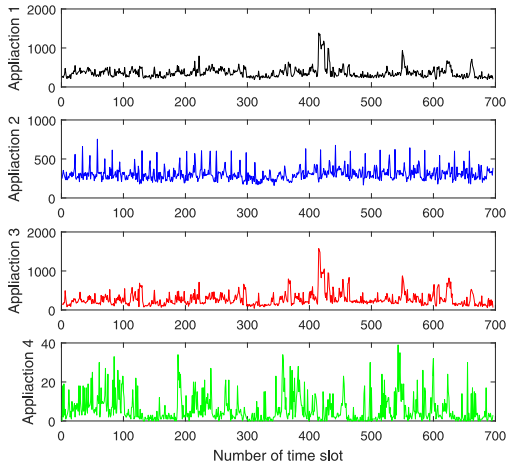

Fig. 1. Number of arrived requests from each type of IoT application.

TABLE II
Prices ($) of the VM Instances

Provider (VM type)	Pricing option	Up-front	Hourly
Amazon EC2 (t3.large)	1-year reserved	515	0.059
	on-demand	0	0.104
Microsoft Azure (B2MS)	1-year reserved	680	0.078
	on-demand	0	0.132
Aliyun (ecs.g5.large)	1-year reserved	336	0.038
	on-demand	0	0.129

A. Simulation Settings

In this simulation, there are $N = 4$ types of IoT applications to be scheduled. We adopt the realistic application requests from the Google cluster traces [28]. These real traces include the data collected from a set of machines over about a month-long period in May 2011. In this data set, every job event table contains the timestamp, missing info, event type, scheduling class, user name, etc. We consider the job request whose event type is 0 which represents that this request becomes eligible for scheduling. Moreover, we use the scheduling class (which is labeled by 0, 1, 2, or 3 in the data set) to distinguish the type of application request, and the timestamp to count the number of arrived application requests in each slot (i.e., hour). Fig. 1 plots the request arrival rate of each type of IoT application.

In addition, three geographically distributed clouds are considered to process these IoT application requests. These three clouds are supported by three different cloud providers, which are Amazon EC2, Microsoft Azure, and Aliyun. The price of each VM instance in each cloud is obtained directly from the official pricing websites [29]–[31]. Table II shows the prices (in dollars) of the VM instances in the three clouds. We can see that no matter which cloud, the price of the reserved VM instance is much lower than that of the on-demand VM instance. For each on-demand VM instance, its price in every slot fluctuates by $\pm 10\%$ based on the price in Table II. According to [23], we set the data size of each application request as $u_1 = 14$ (MB), $u_2 = 16$ (MB), $u_3 = 40$ (MB), and $u_4 = 64$ (MB). Besides, the bandwidth of each cloud is set as $C_1 \sim U[4 \times 10^4, 6 \times 10^4]$ MB, $C_2 \sim U[5 \times 10^4, 7 \times 10^4]$ MB, and $C_3 \sim U[7 \times 10^4, 8 \times 10^4]$. For each type of application, the processing rate of each VM type per time slot is set as Table III.

TABLE III
PROCESSING RATE (I.E., THE NUMBER OF REQUESTS) OF EACH VM TYPE PER TIME SLOT

Application type \ VM type	t3.large	B2MS	ecs.g5.large
1	45	50	45
2	40	45	40
3	35	40	40
4	4	5	4

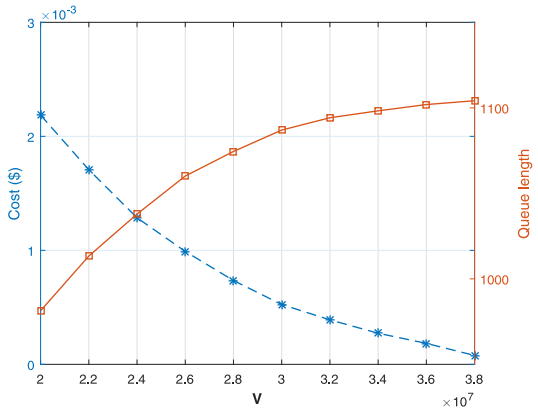

Fig. 2. Cost and queue length under different V.

B. Parameter Analysis

A set of experiments are conducted to show the effects of parameters including tradeoff parameter, VM's price, and processing rate on RSRP.

1) Effect of Tradeoff Parameter V: Fig. 2 plots the cost and queue length of RSRP with an increasing V. We can observe that when V increases, the cost decreases, which verifies the correctness of conclusion (32) in Theorem 2. It is because, with the increase of V, more weight is put on the cost. Therefore, RSRP will dynamically adjust the resource provisioning decision to further reduce the cost. We can also observe that when V rises, queue length also increases, which is consistent with (33) in Theorem 2. It is because with an increasing V, the weight put on cost will rise, and the weight put on queue length will reduce accordingly. In this case, the queue length of RSRP will rise. Nevertheless, we can see that the queue length tends to be stable as V rises. In addition, from Fig. 2, it can be seen that RSRP can make a tradeoff between cost and queue length by adjusting V. In reality, CPA can set the value of V by jointly considering the application's performance requirement and cost.

2) Effect of VM's Price: Fig. 3 plots the number of application requests scheduled to the three clouds with different price scale factors α. In the experiment, we set the VM's prices in Amazon EC2 as $\alpha \cdot p^r_{ij}(t)$ and $\alpha \cdot p^o_{ij}(t)$, where α varies from 1 to 2 with an increment of 0.25. We can see that with the increase of α, the number of application requests allocated to Amazon EC2 decreases, and the number of application requests scheduled to two other clouds (i.e., Microsoft Azure and Aliyun) increases. The reason is that when the VM's price in Amazon EC2 rises, the cost in Amazon EC2 for processing the same request is higher. Thus, in order to reduce the whole cost, RSRP will schedule more requests to the cloud with the lower price, and reduce the number of requests scheduled to the cloud which has a higher price. It shows that RSRP can effectively adapt to the changes in price to reduce the cost.

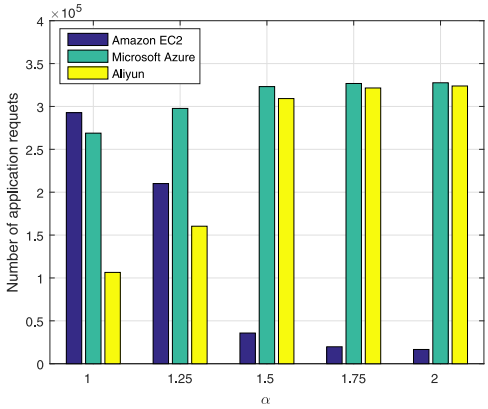

Fig. 3. Number of application requests scheduled to three clouds under different price scale factors α.

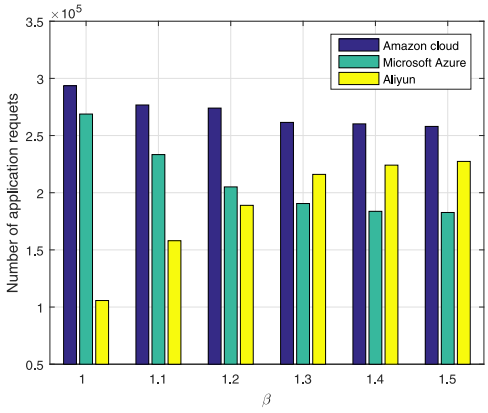

Fig. 4. Number of application requests scheduled to three clouds under different processing rate scale factors β.

3) Effect of Processing Rate: Fig. 4 plots the number of application requests scheduled to the three clouds with different processing rate scale factors β. In the experiment, we set the VM's processing rates in Aliyun as $\beta \cdot b^r_{ij}(t)$ and $\beta \cdot b^o_{ij}(t)$, where β varies from 1 to 1.5 with an increment of 0.1. It can be observed that as β increases, the number of application requests scheduled to Aliyun rises, and the number of application requests scheduled to two other clouds (i.e., Amazon EC2 and Microsoft Azure) reduces. It is because when the processing rate of Aliyun increases, it produces a smaller cost for Aliyun to process the same application request. Thus, more application requests will be scheduled to the cloud with the stronger processing ability, and the number of requests which are scheduled to cloud with the smaller processing rate will be decreased. It shows that in the face of the changeable processing rate, RSRP can dynamically adjust the task scheduling and resource provisioning decisions to reduce the cost.

C. Comparison Experiment

In this experiment, we compare the cost and queue length of RSRP with two other algorithms (called LCC and OVMP in this article), which are proposed in [7] and [24], respectively.

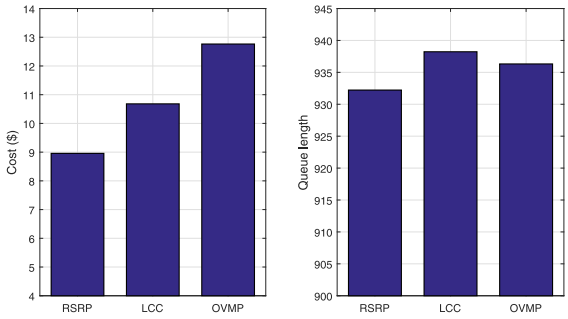

Fig. 5. Cost and queue length of three different algorithms.

As OVMP only considers the resource provisioning but without the request scheduling, we assume that OMVP schedules the application requests to the different clouds equally, which is helpful to balance the workload among the clouds.

Fig. 5 plots the cost and queue length of RSRP, LCC, and OVMP. From Fig. 5, we can see that both the cost and queue length of RSRP are lower than that of LCC and OVMP. The reason is that RSRP can refer to the changes in VM's price and processing rate to dynamically adjust the RSRP decisions as illustrated in Section V-B. However, when making the resource provisioning decision, LCC does not take into consideration the dynamics in the VM's price; and for OVMP, it does not take into account the dynamic VM's processing rate. Numerically, RSRP can reduce the cost by about 16% compared with LCC, and by about 30% compared with OVMP. It shows that RSRP can effectively reduce the cost and stabilize the queue length at the same time.

VI. Conclusion

In this article, we focused on the RSRP problem in the highly dynamic multiclouds environment. An online algorithm, RSRP, was proposed which can minimize the system cost while bounding the queueing delay. RSRP can obtain a close-to-optimal system cost, and make an arbitrary tradeoff between the system cost and queue backlog. The simulation results showed that RSRP can effectively reduce the system cost and maintain the queue backlog at a lower level.

References

[1] J. Chen, K. Hu, Q. Wang, Y. Sun, Z. Shi, and S. He, "Narrowband Internet of Things: Implementations and applications," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 2309–2314, Dec. 2017.

[2] J. Ni, K. Zhang, X. Lin, and X. S. Shen, "Securing fog computing for Internet of Things applications: Challenges and solutions," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 601–628, 1st Quart., 2018.

[3] Y. Chen, N. Zhang, Y. Zhang, and X. Chen, "Dynamic computation offloading in edge computing for Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4242–4251, Jun. 2019.

[4] L. Xing, G. Levitin, and Y. Xiang, "Defending N-version programming service components against co-resident attacks in IoT cloud systems," *IEEE Trans. Services Comput.*, early access, Mar. 13, 2019, doi: 10.1109/TSC.2019.2904958.

[5] X. Zhang, C. Wu, Z. Li, and F. C. M. Lau, "A truthful $(1-\epsilon)$-optimal mechanism for on-demand cloud resource provisioning," *IEEE Trans. Cloud Comput.*, early access, Apr. 3, 2018, doi: 10.1109/TCC.2018.2822718.

[6] S. Mireslami, L. Rakai, M. Wang, and B. H. Far, "Dynamic cloud resource allocation considering demand uncertainty," *IEEE Trans. Cloud Comput.*, early access, Feb. 4, 2019, doi: 10.1109/TCC.2019.2897304.

[7] F. Liu, B. Luo, and Y. Niu, "Cost-effective service provisioning for hybrid cloud applications," *Mobile Netw. Appl.*, vol. 22, no. 2, pp. 153–160, Apr. 2017.

[8] H. Wang, P. Shi, and Y. Zhang, "Jointcloud: A cross-cloud cooperation architecture for integrated Internet service customization," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Atlanta, GA, USA, Jun. 2017, pp. 1846–1855.

[9] Y. Chen, J. Huang, C. Lin, and J. Hu, "A partial selection methodology for efficient QoS-aware service composition," *IEEE Trans. Services Comput.*, vol. 8, no. 3, pp. 384–397, May/Jun. 2015.

[10] G. Premsankar, M. Di Francesco, and T. Taleb, "Edge computing for the Internet of Things: A case study," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1275–1284, Apr. 2018.

[11] Q. Ye, W. Zhuang, X. Li, and J. Rao, "End-to-end delay modeling for embedded VNF chains in 5G core networks," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 692–704, Feb. 2019.

[12] Z. Wu, Z. Lu, P. C. K. Hung, S.-C. Huang, Y. Tong, and Z. Wang, "QaMeC: A QoS-driven IoVs application optimizing deployment scheme in multimedia edge clouds," *Future Gener. Comput. Syst.*, vol. 92, pp. 17–28, Mar. 2019.

[13] J. Mei, K. Li, Z. Tong, Q. Li, and K. Li, "Profit maximization for cloud brokers in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 1, pp. 190–203, Jan. 2019.

[14] F. Zhang *et al.*, "A load-aware resource allocation and task scheduling for the emerging cloudlet system," *Future Gener. Comput. Syst.*, vol. 87, pp. 438–456, Oct. 2018.

[15] D. T. Nguyen, L. B. Le, and V. Bhargava, "Price-based resource allocation for edge computing: A market equilibrium approach," *IEEE Trans. Cloud Comput.*, early access, Jun. 6, 2018, doi: 10.1109/TCC.2018.2844379.

[16] A. Alsarhan, A. Itradat, A. Y. Al-Dubai, A. Y. Zomaya, and G. Min, "Adaptive resource allocation and provisioning in multi-service cloud environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 1, pp. 31–42, Jan. 2018.

[17] W. Xiao, W. Bao, X. Zhu, C. Wang, L. Chen, and L. T. Yang, "Dynamic request redirection and resource provisioning for cloud-based video services under heterogeneous environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 7, pp. 1954–1967, Jul. 2016.

[18] F. Liu, Z. Zhou, H. Jin, B. Li, B. Li, and H. Jiang, "On arbitrating the power-performance tradeoff in SaaS clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 10, pp. 2648–2658, Oct. 2014.

[19] L. Chen, X. Li, Y. Guo, and R. Ruiz, "Hybrid resource provisioning for cloud workflows with malleable and rigid tasks," *IEEE Trans. Cloud Comput.*, early access, Jan. 23, 2019, doi: 10.1109/TCC.2019.2894836.

[20] C.-T. Fan, Y.-S. Chang, and S.-M. Yuan, "VM instance selection for deadline constraint job on agent-based interconnected cloud," *Future Gener. Comput. Syst.*, vol. 87, pp. 470–487, Oct. 2018.

[21] Y. Chen, N. Zhang, Y. Zhang, X. Chen, W. Wu, and X. S. Shen, "TOFFEE: Task offloading and frequency scaling for energy efficiency of mobile devices in mobile edge computing," *IEEE Trans. Cloud Comput.*, early access, Jun. 20, 2019, doi: 10.1109/TCC.2019.2923692.

[22] Q. Ye, J. Li, K. Qu, W. Zhuang, X. S. Shen, and X. Li, "End-to-end quality of service in 5G networks: Examining the effectiveness of a network slicing framework," *IEEE Veh. Technol. Mag.*, vol. 13, no. 2, pp. 65–74, Jun. 2018.

[23] H. Yuan, J. Bi, and M. Zhou, "Multiqueue scheduling of heterogeneous tasks with bounded response time in hybrid green IaaS clouds," *IEEE Trans. Ind. Informat.*, vol. 15, no. 10, pp. 5404–5412, Oct. 2019.

[24] S. Shi, C. Wu, and Z. Li, "Cost-minimizing online VM purchasing for application service providers with arbitrary demands," in *Proc. IEEE 8th Int. Conf. Cloud Comput.*, New York, NY, USA, Jun./Jul. 2015, pp. 146–154.

[25] D. P. Bertsekas, R. G. Gallager, and P. Humblet, *Data Networks*, vol. 2. Prentice-Hall: Englewood Cliffs, NJ, USA, 1992,

[26] Y. Chen, N. Zhang, Y. Zhang, X. Chen, W. Wu, and X. S. Shen, "Energy efficient dynamic offloading in mobile edge computing for Internet of Things," *IEEE Trans. Cloud Comput.*, early access, Feb. 11, 2019, doi: 10.1109/TCC.2019.2898657.

[27] M. J. Neely, *Stochastic Network Optimization With Application to Communication and Queueing Systems*. San Rafael, CA, USA: Morgan & Claypool, 2010.

[28] C. Reiss, J. Wilkes, and J. Hellerstein. (May 2011). *Google Cluster Data*. [Online]. Available: https://github.com/google/cluster-data

[29] *Amazon EC2 Pricing*. Accessed: May 2019. [Online]. Available: http://aws.amazon.com/ec2/pricing/

[30] *Microsoft Azure Pricing*. Accessed: May 2019. [Online]. Available: https://azure.microsoft.com/zh-cn/pricing/

[31] *Aliyun Pricing*. Accessed: May 2019. [Online]. Available: https://www.aliyun.com/price/