A Fairness-Aware Pricing Methodology for Revenue Enhancement in Service Cloud Infrastructure

Yuanfang Chi, *Student Member, IEEE*, Xiuhua Li, *Student Member, IEEE*, Xiaofei Wang, *Member, IEEE*, Victor C. M. Leung, *Fellow, IEEE*, and Abdallah Shami, *Senior Member, IEEE*

***Abstract*—Infrastructure-as-a-service (IaaS) cloud providers sell their resources as different types of virtual machines (VMs) called instances. Any resource is limited in capacity, so is the resource in cloud. Pricing models can be used as a tool for cloud providers not only to cover their costs and realize a profit but also to encourage cloud tenants to use cloud resources efficiently and achieve high utilization. In this paper, we propose a new pricing methodology that encourages cloud tenants, whose requested VMs can be allocated easily and fairly, to use more cloud service by offering them lower prices, while discouraging cloud tenants, whose requested VMs are difficult to allocate, from using cloud service by charging them higher prices, while enhancing the revenue that a cloud provider receives. We perform a case study with a multiresource allocation fairness algorithm, i.e., the dominant resource sharing algorithm. Then, we further conduct case studies with two ways of price calculation: the total unit price redistribution and total revenue redistribution. Evaluation results show that the proposed pricing model with total unit price redistribution and total revenue redistribution can increase the overall revenue of cloud providers by up to 11.60% and 11.18%, respectively.**

***Index Terms*—Cloud computing, fairness, pricing model, pricing sensitivity, revenue enhancement.**

I. Introduction

CLOUD computing is transforming information technology around the world. The computational and storage resources provided by infrastructure-as-a-service (IaaS) cloud, through different types of *instances*, are easy to access and maintain. Thus, large investments have been made to move business services into cloud and implementing/managing data centers to support cloud services. This raises a number of concerns with respect to the cost efficiency of the cloud, from the perspectives of both the cloud providers and the cloud consumers or tenants. Upon the request of an *instance* by a tenant, if the cloud has enough resources to host the instance, a virtual machine (VM) is allocated onto a server, so that the cloud tenant could run her applications or other computational tasks on the *instance*, or the VM to be specific. Many research works [1]–[3] have been devoted to leverage server virtualization and allocation techniques to optimize data center resource allocation via VM placement optimization. However, optimization from any aspect alone is limiting. The amount of resources that a cloud tenant needs varies from time to time. Traditional resource allocation and provisioning techniques still require data centers to be prepared for the intense resource demand during peak period [4]. Incorrect estimations of user demand levels may lead to costly overprovisioning of resources. Moreover, regardless of how the cloud is considered to be an unlimited resource pool, any resource has fixed capacity. It is obvious that having an optimal resource allocation algorithm to squeeze more capacity to serve more tenants is the key to increase the cloud provider's revenue [5]. It is important to incentivize cloud tenants to request for cloud resources reasonably, by devising a pricing methodology that charges each cloud tenant fairly, so that no one could use up a large portion of the resource and leave few to others. Therefore, user behaviors and usage patterns should also be considered as inputs to the VM placement problem. Many research works [6], [7] have shown that the use of pricing to induce desirable user behavior is a successful approach.

Furthermore, most cloud providers do not offer their tenants a service-level agreement (SLA) with the exact measures specifying the service provided. For example, Amazon Elastic Compute Cloud (EC2) only describes its central processing unit (CPU) resource in terms of equivalent Xeon processors and its input/output (I/O) performance as "high", "moderate", and "low", which are hardly measurable by cloud tenants [8]. Google Compute Engine advertises that its load balancing technique would let its user achieve maximum performance[1] without specifying what "maximum performance" means. For services with best effort, no cloud service provider would promise that the service would meet some definite standards. The SLA of Amazon EC2 guarantees a service availability of 99.95% without mentioning performance.[2] Although Xu and Li [8] have pointed out that a number of measurement studies have reported computational performance degradations of cloud services, most cloud tenants understand that they are using a best-effort service with performance variations, and hence, they tolerate minor performance degradations [9]. In fact, it is difficult for cloud tenants to determine whether the performance degradation is due to the lack of resources. For

Manuscript received November 16, 2014; revised March 30, 2015; accepted June 2, 2015. Date of publication July 13, 2015; date of current version June 26, 2017. This work was supported by the Canadian Natural Sciences and Engineering Research Council under Grant STPGP 447230.

Y. Chi, X. Li, X. Wang, and V. C. M. Leung are with the Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, BC V6T 1Z4, Canada (e-mail: yuanchi@ece.ubc.ca; lixiuhua@ece.ubc.ca; xfwang@ece.ubc.ca; xiaofeiwang@ieee.org; vleung@ece.ubc.ca).

A. Shami is with the Department of Electrical and Computer Engineering, Western University, London, ON N6A 5B9, Canada (e-mail: ashami@eng.uwo.ca).

Digital Object Identifier 10.1109/JSYST.2015.2448719

[1]https://cloud.google.com/products/compute-engine/
[2]http://aws.amazon.com/ec2/sla/

a moment or two, people using applications that are run as a cloud service may experience a slow response time. However, it is almost impossible for an end user to determine if it is the cloud provider or the network provider that should be blamed. Thus, many researchers have proposed revenue enhancement strategies [8], [10], such as resource overbooking, capacity right sizing, and resource throttling, to increase server utilization levels and save maintenance and operation costs. These all proved to be truly revenue-increasing techniques. However, would it be fair to cloud tenants that the cloud provider profits in such a way? First of all, cloud tenants' inability to detect resource shortages or performance degradation is critical for making these techniques feasible and profitable. Second, all the costs of implementing and operating such revenue enhancement techniques are eventually paid by each cloud tenant. It would be unfair to those cloud tenants who have kept their resources highly utilized. Third, a cloud service with a flat rate but utilizes resource throttling and overbooking to try to realize a higher profit is untruthful to its users.

To address these problems, we propose a pricing methodology with dynamic pricing [11], which has been shown to be beneficial in many industries [12]. Under this methodology, an IaaS cloud provider may charge different tenants different prices. Inspired by the principle of the law of supply and demand, we also aim to use pricing to incentivize cloud tenants to use data center resources in a way that is fair, while ensuring high utilization levels. Given that the types of VMs and the number of VMs requested by all active cloud tenants are known at any point in time, our pricing methodology leverages a task scheduling algorithm to evaluate how many VMs each tenant should receive, so that the total resources allocated to each cloud tenant is fair and resource utilization is higher compared to other fair-sharing resource allocation algorithms [13]. A pricing weight for each tenant is derived according to the number of VMs allocated to the tenant. Finally, new prices are determined based on the prices that the cloud provider was charging originally and the weight derived. The total number of requests of different VMs changes as the result of price changes; hence, system optimal resource utilization and optimal usage behaviors are eventually achieved. Furthermore, we study our pricing methodology with the dominant resource fairness (DRF) algorithm [14] and show that the total revenue that a cloud provider receives is enhanced. Moreover, we study our pricing methodology with total unit price redistribution, where we sum up the unit prices charged by a cloud provider originally and redistribute the charges among users according to the pricing weight, and total revenue redistribution, where we sum up the revenue received by a cloud provider originally and redistribute the charges among users according to the pricing weight. The proposed pricing methodology follows a pay-as-you-go pricing model. It suits the fundamental characteristics of the cloud as an on-demand and usage-based service. The proposed pricing methodology is transparent and truthful because it takes the total allocable resources into consideration, so that any manipulation from the backend such as resource throttling or capacity right sizing is reflected in the new prices. Unlike the spot instance [15], our proposed pricing methodology does not produce service termination. In addition, instead of setting the price to the highest possible rate, the prices generated by our pricing methodology are determined by looking at the utilization of system resources.

To the best of our knowledge, this is the very first work to introduce a fairness-aware pricing model that increases IaaS cloud service revenue. The remainder of this paper is organized as follows: We study the existing pricing models in Section II. Then, we present our new pricing methodology in Section III and its formulation in Section IV. In Section V, we use the heterogeneous DRF (DRFH) allocation algorithm, as an example, to study the efficiency of our proposed pricing methodology and show that it is revenue enhancing and achieves personal and social fairness. The results of a numerical experiment and a trace-driven simulation are shown in Section VI. Section VII concludes this paper.

II. Related Work

For the IaaS cloud, computing infrastructures, such as CPUs, storage, and network, are virtualized and provisioned to cloud tenants as *instances*. Each cloud tenant simply requests *instances* to run her applications or other services. Some examples of IaaS are Amazon EC2 and Google Compute Engine [16]. The flat-rate pricing model is used by most of the cloud providers, such that cloud tenants are charged according to the time of usage regardless of network congestions or system workload. Many researchers have proposed pricing models that provide dynamic rates to encourage cloud tenants to lease *instances* in a desired manner.

A. Pricing Methodologies

1) Flat-Rate Pricing: **Reserved Instances:** Amazon EC2 provides the option for customers to subscribe to its cloud service for one- or three-year periods with a nonrefundable one-time payment at a lower rate. Cloud tenants can use their reserved resources at anytime during the reserved period, while Amazon ensures that the reserved resources are always available [17]. However, resources may be wasted if the cloud tenant has reserved an instance but for most of the time left it idle. **Freemium and Usage Based:** To encourage potential cloud tenants to try their cloud services, both Amazon EC2 and Google Compute Engine provide free but limited amount of resources for a limited time period. Once the actual usage exceeds the limits, usage-based pricing is generally applied. Usage-based pricing is the most common pricing model for cloud services because it is elastic and charges a tenant based on the actual usage. A typical Google Compute Engine standard instance may contain one virtual core and 3.75-GB memory and charges in minute-level increments for the time that the cloud tenant runs her instance.[1] Amazon EC2 bills to the nearest server hour or gigabyte month [17]. Usage-based pricing allows users to use the cloud service anytime without a long-term commitment, but the access to cloud service is not always guaranteed. **Financial Option:** Sharma *et al.* [18] proposed a pricing model that employs the financial option theory and Moore's law. They treated the cloud computing commodities as assets and mapped cloud parameters to the

Black–Scholes–Merton model. Moore's law is used to describe the value of the resources in cloud. The price determined by their model is the optimal price that the cloud provider should charge the clients to recover its initial cost.

2) Dynamic-Rate Pricing: Although flat-rate pricing is widely adopted by mobile, Internet, and television service providers, Gallego and van Ryzin [19] have shown that flat-rate pricing is optimal when the capacity or quantity of the expected sales is infinite. On the other hand, the dynamic pricing method has shown its benefit, when the capacity is fixed and unsold volume is worthless, such as in the airline, hotel, and electric utility industries [12]. **Spot Instances:** Amazon provides its unsold cloud capacity as spot instances for customers to bid on. Amazon sets its spot prices through a market-driven auction and publishes its spot price for the next time period online, so that customers can run those instances as long as their bids exceed the current spot price. Spot instance pricing allows Amazon to sell more of its unused resources at the highest possible rate, while preserving control over the spot price [15]. Unfortunately, for cloud tenants, spot instances introduce uncertainty to their access to the cloud service because they may be terminated at any time. **Smart Metering:** The smart metering model proposed by Narayan *et al.* [20] provides dynamic pricing of the cloud service based on the load condition. The resource usage is metered and recorded. The customer's historical utilization statistics are used to predict the load condition for the next time interval of operation and thus determine the price. The price is then published on the cloud provider's website, so that the customer could decide whether to continue her usage.

B. Fairness-Aware Pricing

There has been some research work considering resource fairness in cloud servers. Fairness–efficiency tradeoffs were studied in [21] with multiple resource types. The authors defined percentage efficiency as the percentage difference between the total number of jobs processed in a given allocation and the maximum number of jobs that can be processed with the same capacity constraint, and the leftover capacity as the amount of unused resources. Since one of the goals of our pricing methodology is to increase the revenue of a cloud provider while achieving a high level of resource utilization, finding the most efficient resource fair-sharing algorithm is crucial for our pricing methodology. Another work [22] considered the price competitions among different providers in the open market. They tackled the cloud competition problem and proposed a noncooperative game to investigate the price competition among cloud providers and its impact on profit of all cloud providers, tenant satisfaction, and final instance prices. However, the competition among multiple cloud providers is out of the scope of this paper.

III. Fairness-Aware Pricing Methodology

Here, we introduce our fairness-aware pricing methodology for IaaS cloud computing service providers.

A. Motivation

Our pricing methodology determines prices, for each active user in the system, according to how easily the system could provision VMs requested by the user. With the allocation goal of both fair sharing among all users and high system utilization, fewer resources allocated to a user by a task scheduling algorithm indicates that the system has more difficulty to place the user-requested VM. Therefore, we are motivated to raise the prices for these users who overload the system. As a result of price changes, requests for a specific type of VM by existing or potential users are encouraged or discouraged according to the *law of supply and demand*. Our pricing methodology is based on a pay-as-you-go pricing model with dynamic unit prices, so that the fundamental characteristics of the cloud as an on-demand and usage-based service are preserved. Furthermore, new prices are determined according to the total allocable resources; system optimal resource utilization is achieved by using pricing to incentivize users to use the cloud service in the desired way. Hence, manipulations from the system backend by the cloud provider, such as capacity right sizing, resource throttling, and overbooking, are not necessary. Compared to the spot instance [15] provided by Amazon, our proposed pricing methodology does not produce service termination. In addition, instead of setting the prices to the highest possible rate, prices generated by our pricing methodology are determined by considering the utilization of system resources.

B. Methodology

As mentioned earlier, the main motivation of this work is leveraging a pricing policy to manipulate user behavior, for the purpose of increasing system resource utilization. Fig. 1 shows a sequence diagram that demonstrates the working mechanism of the proposed fairness-aware pricing methodology. Users request VMs with known current prices of VM types. Then, upon the arrival of a new user's VM request, task scheduling algorithms are executed to dynamically allocate cloud resources, subject to the cloud server's existing tasks and available resources. The algorithm not only determines how easily the VM could be hosted but also decides on which server the VM is to be placed. While the user's request is being fulfilled, the real-time cloud information, including incoming VM settings and current cloud workload, is updated to the decision-making module that implements our proposed pricing model. The new price is generated accordingly after the allocation decision. With the new prices, existing users react by increasing or decreasing their usage percentage of the cloud service. Once the user reactions are submitted to our system, the algorithm starts again from the beginning as a new loop. In addition, potential users may decide to start to use the cloud service. This procedure continues as the users' demands fluctuate. It is apparent that a decision-making module that predicts users' reaction to future prices and subsequently adjusts prices to achieve predefined optimization goals is the key of our methodology. Details are described in following sections.

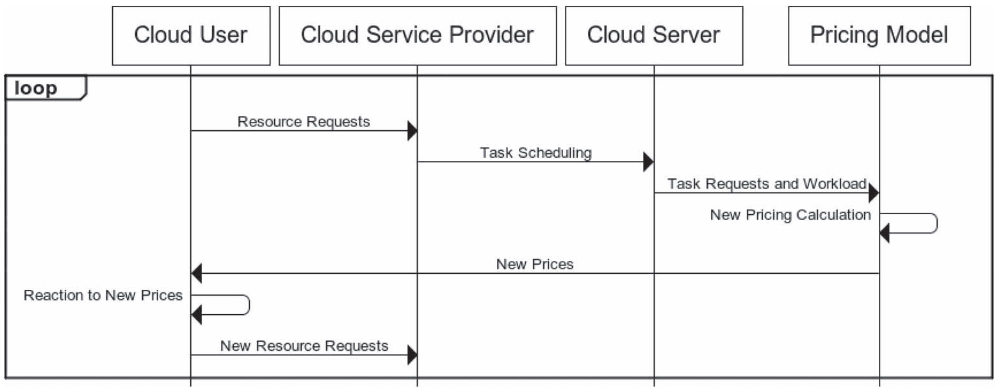


Fig. 1. Sequence diagram for fairness-aware pricing.

TABLE I
DEFINITION OF INPUT VARIABLES

$S = \{1, \ldots, k\}$	Set of servers the cloud system contains.
$R = \{1, \ldots, m\}$	Resource types each server contains.
$c_l = \{c_{l1}, \ldots, c_{lm}\}$	Normalized resource capacity vector for each server l, where c_{lr} denotes the total amount of resource r available in server l.
$U = \{1, \ldots, n\}$	Set of cloud tenants active in the cloud system.
$D_i = \{D_{i1}, \ldots, D_{im}\}$	Resource vector of the VM requested by user i, where D_{ir} denotes the fraction of resource r required by user i's VM.
$T_s = \{T_{s_1}, \ldots, T_{s_n}\}$	The number of VMs submitted to the system by all tenants, where T_{s_i} denotes the number of VMs submitted by user i.

IV. PROBLEM DISCUSSION

With the pricing methodology proposed in the last section, a number of open issues remain to be addressed before the methodology can be put in practice. For instance, on which server or servers should the cloud system place the requested VMs? How would pricing affect users' demands? How to determine the price, in order to enhance the revenue? Here, we discuss these issues.

We define the input variables, as shown in Table I. Suppose that the set of servers S, resource types R, and resource capacity vector c_l are known for each server l. In addition, suppose that the set of cloud tenants U, the resource vector of their desired types of VM D_i, and the number of VMs that users initially requested T_s are known.

A. Price Sensitivity

Our pricing methodology leverages the concept of price sensitivity to model the user reaction to the new price generated by our pricing model. In a competitive market, price sensitivity defines the highest price a customer would pay for the desired product and the lowest price a customer would pay without second thought of the product quality [23], [24]. As the most powerful tool to marketers, price sensitivity is well studied when setting the price of a new product to maximize the demand of the product and the business outcome [25]. For a majority of the buyers, the price is not only a key factor that will influence their purchase decisions but also an indicator for them to perceive product or service quality. The price threshold that captures consumer insensitivity to small price changes was examined in [25]. Harmon *et al.* [23] studied the price sensitivity measurement (PSM) model and incorporated it into the value-based software engineering (VBSE) process. PSM check is used twice during the VBSE process to first refine customer value assessment of the potential product and then help in finalizing the development of a marketing plan prior to commercialization. Moreover, the law of supply and demand suggests that the availability and desirability of a product has a great effect on the product price [26]. If the supply is sufficient but the demand is low, the price will be low. In contrast, if the supply is inadequate but the demand is high, the price will be high. Then, given that all other factors are equal, the demand of the good or service decreases as the price increases.[3] The demand curve of a commodity is downward sloped, as shown in Fig. 2. However, to the best of our knowledge, the specific demand curves of cloud services have not been well studied.

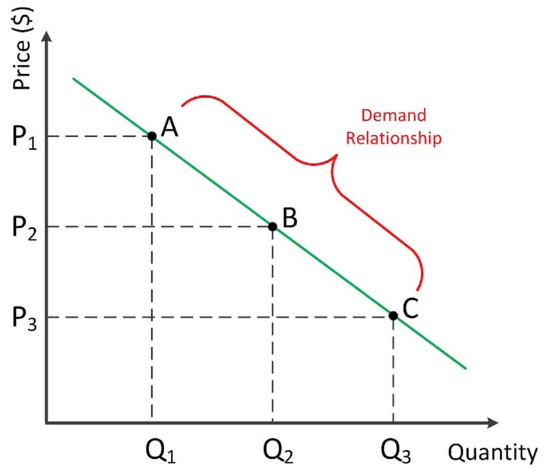


Fig. 2. Commodity demand curve.[3]

In this paper, we formulate the user sensitivity to cloud usage price as a *market reaction function* f_R, which represents the relationship between the price p and the cloud users' corresponding demand level $D(p)$. That is

$$D(p) = f_R(p). \tag{1}$$

In this paper, a higher demand level indicates an increasing need of cloud VMs.

[3]http://www.investopedia.com/terms/l/lawofdemand.asp

B. Cloud Task Scheduling

One of the most important steps of our proposed pricing methodology is to determine how easily a VM can be handled by the cloud. The computing resources provided by the cloud consist of heterogeneous servers, where each server may contain a different amount of resources. Hence, the allocation of various forthcoming VM requests onto the distributed servers in the cloud is of great importance for the overall system efficiency. The VM placement algorithm in [27] leverages the ability of VM migration to dynamically migrate VMs from servers with low utilization, so that unutilized servers could be switched off to save energy. A dynamic VM placement algorithm is proposed in [28], which migrates VMs according to different user-defined policies. In [29], an algorithm is proposed based on a binary search tree to find the suitable server for optimal VM placement. A VM placement algorithm is proposed in [30] with the goal of minimizing the active servers. Considering the multiresource nature of cloud platform, Ghodsi *et al.* [14] introduced the idea of DRF to this area. DRF is a multiresource allocation problem with the goal to equalize the dominant share, which is the maximum ratio of any resource allocated to a user in the system, among all cloud tenants. It is modeled as a max–min optimization problem and is shown to possess a number of fairness properties such as sharing incentive, strategy proofness, envy freeness, and Pareto efficiency. A higher system utilization is also promised compared to a slot-based fair scheduling and a max–min fair-sharing algorithm that focuses on a single-resource type [14]. However, one limitation of DRF is that it simplifies resources in a cloud computing system as resources in a supercomputer. It does not consider the capacity limitation of each server and the fact that, after several VMs are placed in a server, the remaining resources of this server might not be sufficient to host other VMs. DRFH is proposed in [31], which generalizes DRF by applying it to real cloud computing systems with heterogeneous servers.

C. Fairness

Some of the desirable fairness properties are sharing incentive that guarantees that no user is better off in a system where resources are equally partitioned among all users; strategy proofness, where no user can be better off by providing untruthful resource demands; Pareto efficiency, where user requests are allocated without preempting existing allocations; envy freeness, where no user prefers the allocation of another user [14].

Two important fairness properties that users care the most is personal and social fairness. Personal fairness means that the price meets each user's personal expectation [32]. Most users would expect that a VM consisting of smaller resources is cheaper than a VM that consists of larger resources. Social fairness means that the provider does not profit unreasonably and the price only increases based on the increase of costs. Since server capacity is limited, one VM with a larger amount of resources allocated in a server could use up a large portion of the server's resources, and the remaining resources might not be sufficient for allocation to other users' VMs. Therefore, a VM with larger amount of resources is more costly to allocate. Our proposed methodology charges more for those VMs that require a large amount of resources, which indicates social fairness.

D. Revenue Enhancement

One of the goals of our proposed pricing methodology is to enhance the revenue for cloud service providers. Given the number of VMs that each user has initially requested T_{s_i}, we first determine T_{n_i}, i.e., the number of VMs that each user is allocated by the cloud system. Then, let P_{e_i} be the unit price that the cloud system charges each user. The total revenue R_e that a cloud provider receives is calculated as

$$R_e = \sum_{i \in U} T_{s_i} \times P_{e_i}, \qquad \forall\, i \in U. \tag{2}$$

User elasticity is determined through the demand function

$$T_{s_i} = f_R\left(P_{e_i}\right), \qquad \forall\, i \in U. \tag{3}$$

We input the same user demand vectors and system resource vectors into our pricing model to obtain the new prices P_{d_i}. Then, we use the demand function again to derive the number of VMs that users would request after they receive the new prices, which is denoted by T'_{s_i}. That is

$$T'_{s_i} = f_R\left(P_{d_i}\right), \qquad \forall\, i \in U. \tag{4}$$

All users' VM requests are scheduled using first-in–first-out (FIFO) scheduling again. Let T'_{n_i} be the number of VMs that each user is allocated. The total revenue R_d that a cloud provider receives with the new prices is calculated as

$$R_d = \sum_{i \in U} T'_{n_i} \times P_{d_i}, \qquad \forall\, i \in U. \tag{5}$$

V. Case Study on DRFH Fairness

In [31], authors have proved that the DRFH allocation algorithm achieves significant improvements in resource utilization, as compared to the traditional slot scheduler (e.g., Hadoop fair scheduler). Thus, here, we study our proposed pricing methodology with DRFH allocation algorithm.

A. Market Reaction Function

Since the supply and demand functions of the cloud service as the price changes have not been well studied, we utilize an isoelastic demand function to model how the level of user demand changes as price changes [8], i.e.,

$$f_R(p) = \left(\frac{1}{p}\right)^{\left(\frac{1}{\alpha}\right)}, \qquad \alpha \in (0,1) \tag{6}$$

where α is the elasticity coefficient.

From (5) and (6), we deduce that

$$R_d = \sum_{i \in U} \left(\frac{1}{P_{d_i}}\right)^{(1/\alpha - 1)}, \qquad \alpha \in (0,1). \tag{7}$$

When the price is smaller than 1, the revenue generated by our pricing methodology decreases as α increases. In contrast, when the price is larger than 1, the revenue generated by our pricing methodology increases as α increases.

B. DRFH Scheduling

DRFH takes the set of heterogeneous servers S, server resource types (e.g., CPU, memory, and storage) R, and normalized resource capacity vector for each server l, c_l as one input and takes the set of cloud tenants U and resource vector of requested VM D_i as another input. Let r_i^* be the largest resource required by user i's VM, then the normalized demand of user i's VM is calculated as

$$d_{ir} = \frac{D_{ir}}{D_{ir_i^*}}, \qquad \forall\, i \in U,\ r \in R. \tag{8}$$

For example, consider a system with two servers, i.e., Server 1 with 2 CPUs and 12-GB RAM and Server 2 with 12 CPUs and 2-GB RAM. The system has 14 CPUs and 14-GB RAM in total. Thus, Server 1 and Server 2 have normalized resource capacity vectors $\langle 1/7, 6/7\rangle$ and $\langle 6/7, 1/7\rangle$, respectively. Suppose that there are two users. User1 requests VMs with 0.1 CPUs and 0.5-GB RAM. User2 requests VMs with 1 CPU and 0.2-GB RAM. User1 and user2 have normalized demand vectors $\langle 1/5, 1\rangle$ and $\langle 1, 1/5\rangle$, respectively.

Let $A_{il} = \{A_{il1}, \ldots, A_{ilm}\}$ be the resource allocation vector for user i on server l, $A_i = \{A_{i1}, \ldots, A_{ik}\}$ be the allocation matrix of user i, and $A = \{A_1, \ldots, A_n\}$ be the overall allocation for all users. An allocation is feasible only if no server has used more resources than its total resources, i.e.,

$$\sum A_{ilr} \le c_{lr}, \qquad \forall\, l \in S,\ r \in R. \tag{9}$$

Let $N_{il}(A_{il})$ be the maximum number of VMs that can be scheduled for user i in server l and D_{ir} be the fraction of the total resource r required by user i, then

$$N_{il}(A_{il}) = \min_{r \in R} \left\{ \frac{A_{ilr}}{D_{ir}} \right\}, \qquad \forall\, l \in S,\ r \in R. \tag{10}$$

Let $N_i(A_i)$ be the total number of VMs that can be scheduled for user i under allocation A_i, then

$$N_i(A_i) = \sum_{l \in S} N_{il}(A_{il}). \tag{11}$$

The dominant resource allocated to user i in server l is

$$G_{il}(A_{il}) = N_{il}(A_{il}) D_{ir_i^*} = \min_{r \in R} \left\{ \frac{A_{ilr}}{d_{ir}} \right\}. \tag{12}$$

Therefore, the global dominant share allocated to user i is

$$G_i(A_i) = \sum_{l \in S} G_{il}(A_{il}) = \sum_{l \in S} \min_{r \in R} \left\{ \frac{A_{ilr}}{d_{ir}} \right\}. \tag{13}$$

The goal of DRFH is to maximize the minimum global dominant share among all users, subject to the capacity constraints of each server, i.e.,

$$\max_{A} \quad \min_{i \in U} G_i(A_i)$$
$$\text{s.t.} \quad \sum A_{ilr} \le c_{lr}, \qquad \forall\, l \in S,\ r \in R. \tag{14}$$

DRFH allocates resource to the user with minimum global dominant share among all active users. When a user has all its VMs placed in a server, it is removed from the user set U, and DRFH repeats the allocation process with the updated user set.

C. DRFH-Based Revenue Enhancement

According to the law of supply and demand, given that other factors such as advertisement, brand preferences, product differentiation, and segment membership are kept the same, the quantity demanded and the price of a commodity are inversely related [23]. For simplicity, let

$$w_i = \frac{\frac{1}{N_i(A_i)}}{\sum_{i \in U} \frac{1}{N_i(A_i)}}, \qquad \forall\, i \in U \tag{15}$$

where $w = \{w_1, \ldots, w_n\}$ denote the weights assigned to n tenants.

1) Total Unit Price Redistribution: We first study the case in which we keep the total unit price for all users the same and redistribute the sum among users according to the weights. To be specific, the new price for each user i is calculated according to the weight of user i and the price that the cloud provider charges each tenant. For example, a typical Google Compute Engine standard instance may contain one virtual core and 3.75-GB memory and charges in minute-level increments for the time that the cloud tenant runs her instance at \$0.07/h.[1] Let $P_e = \{P_{e_1}, \ldots, P_{e_n}\}$ denote the prices that the cloud provider charges n tenants and $P_d = \{P_{d_1}, \ldots, P_{d_n}\}$ denote the new prices generated by our pricing model for the n tenants, then

$$P_{d_i} = w_i \times \sum_{i \in U} P_{e_i}, \qquad \forall\, i \in U. \tag{16}$$

The algorithm for calculating unit prices for user i is shown in Algorithm 1.

Algorithm 1 Pricing Calculator for User i

```
1: Given $S$, $R$, $c_l$, $U$, $d_i$, $T_s$, $P_e$ as input.
2: Set $P_{e_{\text{sum}}} \leftarrow 0$
3: Get $N_i(A_i) \leftarrow$ DRFH $(S, R, c_l, U, d_i, T_s)$
4: for $n = 1$ to $N$ do
5:     $w_i \leftarrow ((1/N_i(A_i))/(\sum_{i \in U}(1/N_i(A_i))))$
6:     $P_{e_{\text{sum}}} \leftarrow P_{e_{\text{sum}}} + P_{e_i}$
7: end for
8: $P_{d_i} \leftarrow w_i \times P_{e_{\text{sum}}}$
```

2) Total Revenue Redistribution: Next, we study the case in which we keep the total revenue of all users the same and redistribute the sum among users according to the weights, given the number of VMs that can be scheduled for each user by DRFH. To be specific, the revenue generated by user i, i.e., R_{e_i}, is calculated as

$$R_{e_i} = N_i(A_i) \times P_{e_i}, \qquad \forall\, i \in U. \tag{17}$$

New unit price $P_d = \{P_{d_1}, \ldots, P_{d_n}\}$ is calculated as

$$P_{d_i} = \frac{w_i \times \sum_{i \in U} R_{e_i}}{N_i(A_i)}, \qquad \forall\, i \in U. \tag{18}$$

The algorithm for calculating unit prices for user i is shown in Algorithm 2.

Algorithm 2 Pricing Calculator for User i

```
1: Given $S$, $R$, $c_l$, $U$, $d_i$, $T_s$, $P_e$ as input.
2: Set $P_{e_{\text{sum}}} \leftarrow 0$
3: Get $N_i(A_i) \leftarrow$ DRFH ($S$, $R$, $c_l$, $U$, $d_i$, $T_s$)
4: **for** $n = 1$ to $N$ **do**
5:     $w_i \leftarrow ((1/(N_i(A_i)))/\sum_{i \in U}(1/(N_i(A_i))))$
6:     $R_{e_{\text{sum}}} \leftarrow R_{e_{\text{sum}}} + P_{e_i} \times N_i(A_i)$
7: **end for**
8: $P_{d_i} \leftarrow ((w_i \times P_{e_{\text{sum}}})/N_i(A_i))$
```

D. DRFH-Based Fairness

In [14], authors showed that, unlike the fair division mechanisms of competitive equilibrium from equal incomes [33], which does not satisfy strategy proofness and population monotonicity, and asset fairness, which does not satisfy sharing incentive and bottleneck fairness, DRF satisfies sharing incentive, strategy proofness, envy freeness, Pareto efficiency, single-resource fairness, bottleneck fairness, and population monotonicity at the mean time. In [31], authors proved by deduction that DRFH, as an extension of DRF, satisfies strategy proofness, envy freeness, Pareto efficiency, single-resource fairness, bottleneck fairness, and population monotonicity as the DRF does. Furthermore, authors in [31] showed through the trace-driven evaluation that, although DRFH does not guarantee 100% sharing incentive for all users, it provides 98% of users the same task completion ratio as the traditional slot schedulers do. Since the calculation of our new prices is inversely proportional to the number of tasks that the DRFH algorithm can schedule for a user, our proposed pricing methodology satisfies the same fairness properties as DRFH. That is, our proposed pricing methodology provides sharing incentive for most of the users; it is strategy proof, envy free, and Pareto efficient and satisfies single-resource fairness, bottleneck fairness, and population monotonicity.

For the same example described previously, suppose that a cloud provider charges for one VM with 0.1 CPU and 0.5-GB RAM at \$0.6/h and one VM with 1 CPU and 0.2-GB RAM at \$1.2/h. In addition, assume that each user requests an infinite number of VMs. DRFH allocates 12 VMs to user1 and 6 VMs to user2. The new prices are calculated as \$0.4/h and \$1.6/h for user1 and user2, respectively.

TABLE II
SYSTEM RESOURCE VECTOR

Server 1	$< 27CPU, 50GB >$
Server 2	$< 13CPU, 50GB >$
Total Resource	$< 40CPU, 100GB >$

Personal Fairness: Since most users would expect that a VM consisting of smaller resources is cheaper than a VM that consists of larger resources, the new prices of \$0.4/h, for a VM with 0.1 CPU and 0.5-GB RAM, and \$1.6/h, for a VM with 1 CPU and 0.2-GB RAM, indicate personal fairness of our pricing model.

Social Fairness: Our new prices are inversely proportional to the number of VMs that a user receives. With the goal of the DRFH allocation algorithm to equalize the share of resources among all active users, fewer VMs allocated indicates that the type of VM requested consists of larger amount of resources. A VM with larger amount of resources is more costly to allocate; hence, it should be charged more. Comparing to the original prices, the decrease in user1's price and the increase in user2's price indicate social fairness of our pricing model.

VI. EVALUATIONS

In order to evaluate the proposed fairness-aware pricing methodology, we conduct experiments to compare the total revenue that our pricing model produces with the total revenue that a pricing model with flat-rate unit price would produce. To be specific, in numerical evaluations, we use actual Google Compute Engine VM types to model our cloud system. We compare the revenue of our pricing model with the revenue that a cloud provider would receive with Google Compute Engine unit prices. In the trace-driven evaluation, we use the sum of CPU and memory of each requesting VM as the unit price of this VM. Then, we compare the revenue of our pricing model with the revenue that a cloud provider would receive with the calculated unit prices. New prices are generated once a request of VM has arrived at the system. Then, a user responds to the new prices that are modeled according to the demand function described in Section V-A. Finally, the revenue of our pricing model is calculated as the product of the new unit prices and users' final decision of the amount of VMs that they request.

A. Numerical Evaluations

Here, we demonstrate the efficiency of our algorithm with a numerical simulation with two types of computing resources, including CPU units and memory devices. A cloud service provider with two servers is hosting applications of four users simultaneously. We formulate the servers' resource as a vector $\langle$CPU, Memory$\rangle$ and all users' request as a resource vector $\langle$CPU, Memory$\rangle$ and corresponding unit prices from Google Compute Engine. These numeric data are listed in Tables II and III, respectively. We use FIFO scheduling to simulate the process of VM allocation and, at the end of the scheduling process, get the actual number of VMs that each user could receive.

1) Total Unit Price Redistribution: Fig. 3 illustrates the comparison of revenues achieved with unit prices of Google

TABLE III
USER DEMAND VECTOR AND PRICES OF GOOGLE COMPUTE ENGINE

User A	$< 1CPU, 3.75GB >$	\$ 0.07/hour
User B	$< 4CPU, 3.6GB >$	\$ 0.176/hour
User C	$< 2CPU, 7.5GB >$	\$ 0.14/hour
User D	$< 4CPU, 15GB >$	\$ 0.28/hour

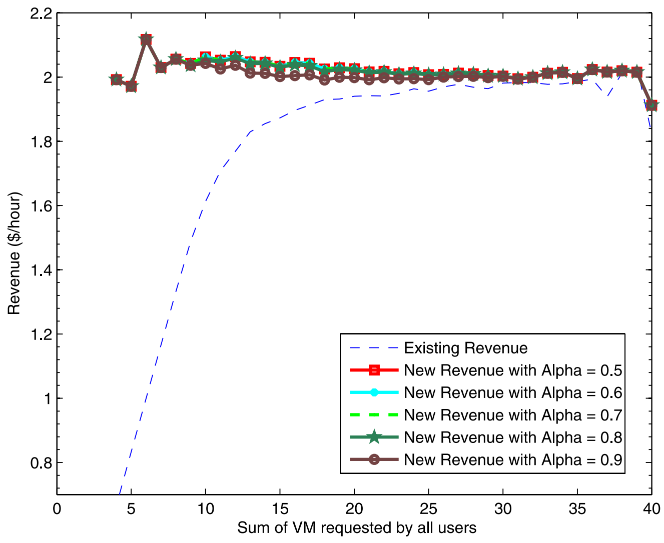


Fig. 3. Comparison of revenues generated by existing and proposed pricing models.

TABLE IV
NEW UNIT PRICES COMPARED TO GOOGLE COMPUTE ENGINE UNIT PRICES

VM Type	Number of VMs Requested	Google Prices	New Prices
$< 1CPU, 3.75GB >$	5	\$ 0.07/hour	\$ 0.097/hour
$< 4CPU, 3.6GB >$	5	\$ 0.176/hour	\$ 0.162/hour
$< 2CPU, 7.5GB >$	5	\$ 0.14/hour	\$ 0.162/hour
$< 4CPU, 15GB >$	5	\$ 0.28/hour	\$ 0.244/hour

Compute Engine and proposed policies with α set to 0.5, 0,6, 0.7, 0.8, and 0.9. Revenues converge as the system resource is fully occupied, at which the sum of VMs requested by all users is approximately 15. Since the unit prices are smaller than 1, the revenue with a smaller α value is greater than the revenue with a larger α value, as depicted in Section V-A. Overall, new revenue generated by our proposed pricing methodology is higher than the existing revenue calculated with unit prices of Google Compute Engine.

In Table IV, we compare our new unit prices for each user with Google Compute Engine unit prices of the types of VMs used in the example, with each user requesting five VMs of their desired type. The VM with $\langle 1\text{CPU}, 3.75\text{ GB}\rangle$ requires the smallest portion of resources, so that it is easy to be placed on a server. Therefore, the unit price of VM $\langle 1\text{CPU}, 3.75\text{ GB}\rangle$ is the lowest. On the other hand, the VM with $\langle 4\text{CPU}, 15\text{ GB}\rangle$ requires the largest portions of resources, so that it is hard to be placed on a server. Hence, the new price of this type of VM is the highest.

In Fig. 4(a), we compare the change of user requests of our new unit prices of the types of VMs used in the example, with α set to 0.9. The VM with $\langle 1\text{CPU}, 3.75\text{ GB}\rangle$ is charged at the lowest price; therefore, the number of VMs that User A requests is increased the most. On the other hand, the VM with $\langle 4\text{CPU}, 15\text{ GB}\rangle$ is charged at the highest price; therefore, as a response to the price increases, User D requests less and less VMs. In Fig. 4(b), we compare the number of VMs allocated on a server for each user with our new unit prices of the types of VMs used in the example, with α set to 0.9.

In Fig. 5, we compare the revenue generated by each user with our new unit prices of the types of VMs used in the example, with α set to 0.9. Compared to the revenue achieved with Google Compute Engine unit prices, the results show that revenues generated with our pricing methodology by users B and D are decreased, whereas the revenues generated by users A and C are increased. Nevertheless, with our pricing methodology, the greatest revenue is generated by user D as it requires the type of VM that is most difficult to handle; the next greatest revenue is generated by user C and then user B and user A, which indicates personal and social fairness.

2) Total Revenue Redistribution: Fig. 6 illustrates the comparison of revenues achieved with unit prices of Google Compute Engine and proposed policies, with α set to 0.5, 0.6, 0.7, 0.8 and 0.9. Revenues converge as the system resource is fully occupied. Again, the revenue with $\alpha = 0.5$ is greater than the revenue with $\alpha = 0.9$. Overall, new revenue generated by our pricing methodology is higher than the revenue calculated with the existing unit prices of Google Compute Engine.

In Table V, with user requests indicated, we compare the unit prices generated for each user's requested VM type with the unit prices provided by Google Compute Engine. The result shows that the VM with $\langle 1\text{CPU}, 3.75\text{ GB}\rangle$ is the cheapest VM and the VM with $\langle 4\text{CPU}, 15\text{ GB}\rangle$ is the most expensive VM.

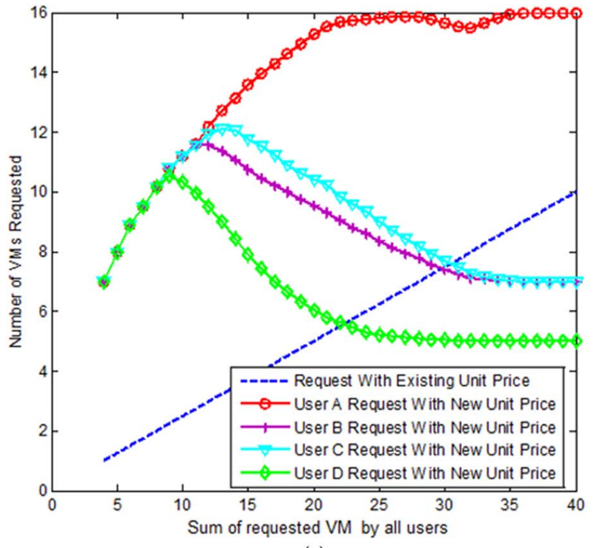
With α set to 0.9, in Fig. 7(a), we compare the requests submitted to the cloud system by each user with the existing and new unit prices, and in Fig. 7(b), we show the differences of the number of requested VMs allocated for each user with the existing and new unit prices.

In Fig. 8, we compare the revenue generated by each user with α set to 0.9. The results show that, compared to the revenue achieved with Google Compute Engine unit prices, the revenues generated by users A and B are decreased with our new unit prices, whereas revenues generated by users C and D are increased with our new unit prices. Furthermore, the greatest revenue is generated by user D as it requires the type of VM that is most difficult to handle; the next greatest revenue is generated by user C and then user B and user A, which indicates personal and social fairness.

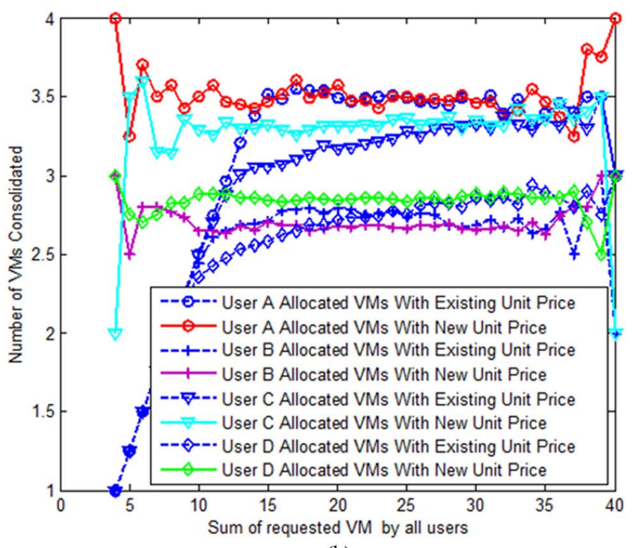
B. Trace-Driven Simulations

Here, we evaluate our proposed pricing model with empirical data from Google cluster-usage traces.[4] The traces contain user resource requests and system resource information for about a month-long period in May 2011. We extract the system resource vector (S and R), the set of active cloud tenants (U), the

[4]https://code.google.com/p/googleclusterdata/



(a)



(b)

Fig. 4. (a) Comparison of requests generated by each user. (b) Comparison of allocated VMs for each user.

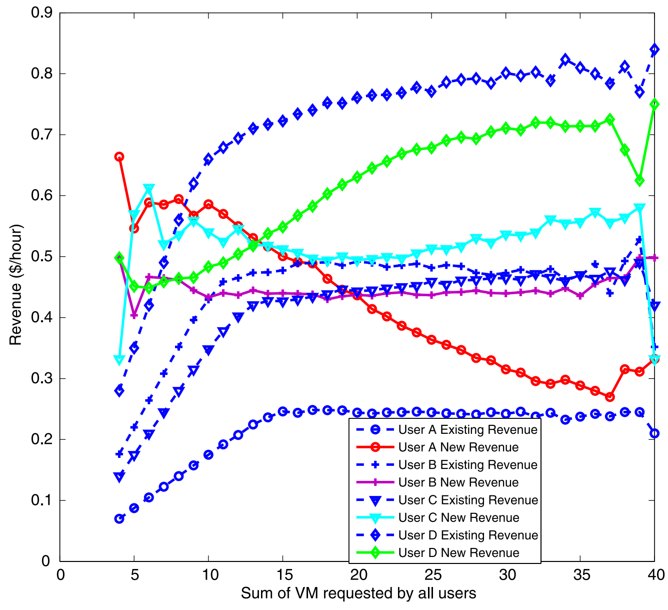


Fig. 5. Comparison of revenues generated by each user.

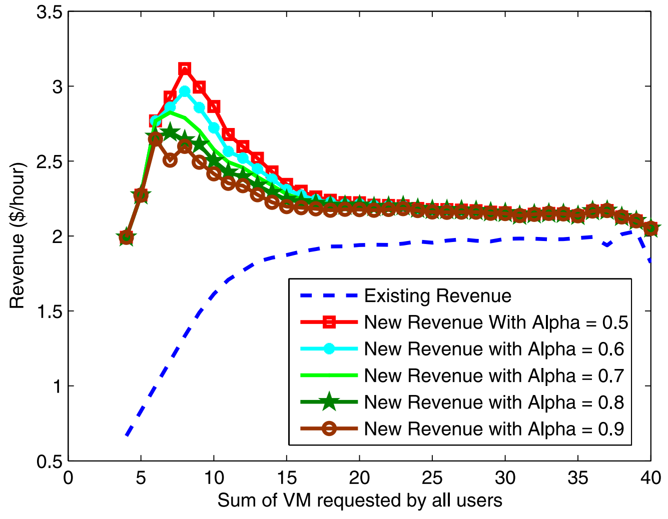


Fig. 6. Comparison of revenues generated by existing and proposed pricing models.

resource vector of the VM requested by the ith user (d_i), and the number of VM requests submitted (T_{si}) from the traces. Since server resources provided in Google cluster-usage traces are normalized to the largest server resource, the actual Google Compute Engine price of the requested type of VM is hard to determine. Thus, we assume that, originally, the cloud provider charges \$1/CPU/h and \$1/GB RAM/h, where both CPU and RAM are normalized numbers extracted from Google cluster-usage traces. For example, a server with normalized 0.5 CPU and 1-GB RAM is originally charged at \$1.5/h.

As described in Section V-A, elasticity coefficient α is the decisive parameter for market reaction. However, the values of α are different from user to user, subject to distinct application domain and various motivations. To simulate a reasonable scenario, the elasticity coefficient α is derived by dynamic calculations with price and demand settings in our experiments.

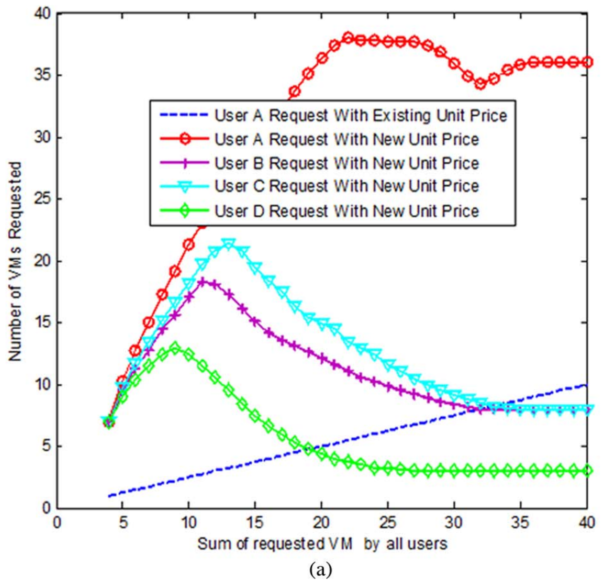
TABLE V
NEW UNIT PRICES COMPARED TO GOOGLE COMPUTE ENGINE UNIT PRICES

VM Type	Number of VMs	Google Prices	New Prices
$< 1CPU, 3.75GB >$	5	\$ 0.07/hour	\$ 0.054/hour
$< 4CPU, 3.6GB >$	5	\$ 0.176/hour	\$ 0.151/hour
$< 2CPU, 7.5GB >$	5	\$ 0.14/hour	\$ 0.151/hour
$< 4CPU, 15GB >$	5	\$ 0.28/hour	\$ 0.34/hour

Specifically, the value of α is calculated by the following equation for our trace-driven simulations:

$$\alpha = \frac{1}{\log_{1/p} f_R(p)}. \tag{19}$$

1) Total Unit Price Redistribution: The trace-driven simulation results for total unit price redistribution are depicted in Fig. 9. The revenue of the cloud provider is calculated according to the user demands extracted from the traces and



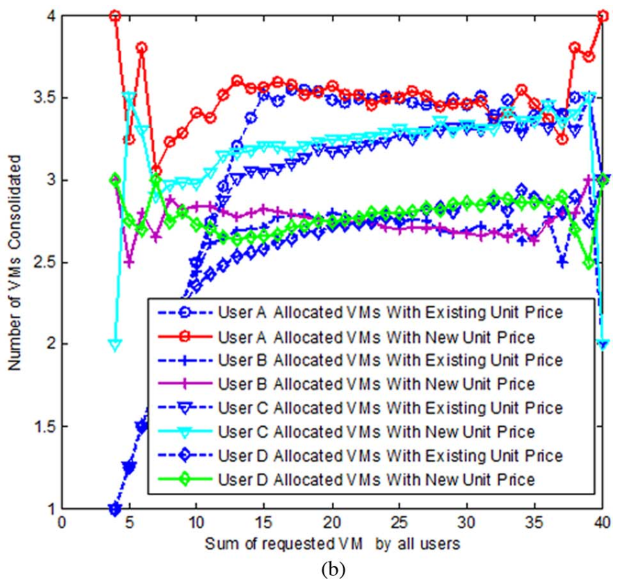


Fig. 7. (a) Comparison of requests generated by each user. (b) Comparison of allocated VMs for each user.

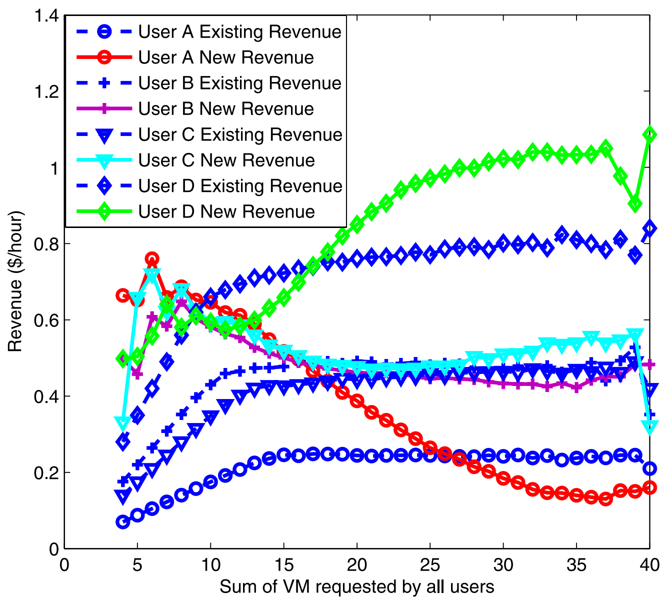


Fig. 8. Comparison of revenues generated by each user.

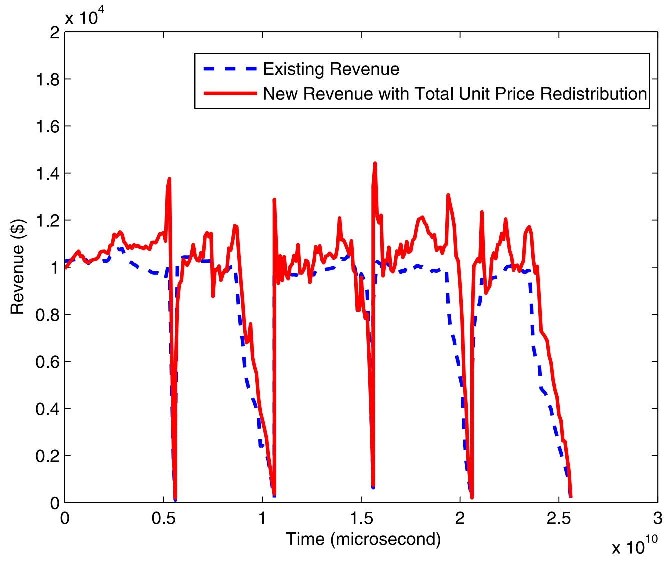


Fig. 9. Comparison between existing and proposed pricing models with data extracted from Google trace.

the assumed original unit prices from Google Compute Engine and unit prices generated by our proposed pricing methodology. The user demands vary throughout the monitoring period, so does the cloud provider's revenue, which ranges from around \$0 to around \$15 000. From the results, we observe that our proposal outperforms the existing pricing policy in most of the time slots. Cumulatively, the cloud service providers' overall revenue can be increased from \$2 173 700 to \$2 426 000, which is by up to 11.60%.

2) Total Revenue Redistribution: In contrast, we also conducted evaluations for the pricing of total revenue redistribution. From the results shown in Fig. 10, we observe that the cloud service provider derives more revenue with the novel pricing solution, comparing to the existing flat pricing approach. The overall revenue increased significantly from \$2 172 800 to \$2 415 600, which is around 11.18%.

VII. CONCLUSION

In this paper, we have proposed a pricing methodology that induces the optimal resource utilization and enhances the revenue of cloud providers. New prices are determined according to the number of tasks that a user could get scheduled by the cloud task scheduling algorithm. A user whose task is difficult for the system to process is discouraged from being submitted to the system as frequently as other users' tasks that can be handled easily. A case study of the methodology with the DRF allocation algorithm has been performed. Furthermore, we have studied our methodology with total unit price redistribution and total revenue redistribution. Numerical simulations have been conducted to compare the total revenue that a cloud provider would receive and the requests of resource submitted to the cloud system by each user with the new unit prices generated by our pricing methodology, in both of the cases, to the total

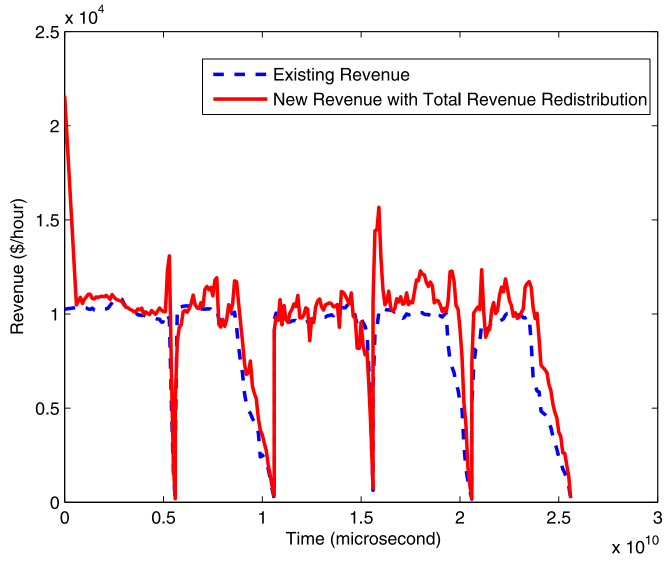


Fig. 10. Comparison between existing and proposed pricing models with data extracted from Google trace.

revenue and resource requests that a cloud provider would receive with its original prices. An empirical study with trace-driven simulation results has shown that the proposed pricing policy with total unit price redistribution and total revenue redistribution can increase the cloud service providers' overall revenue by up to 11.60% and 11.18%, respectively.

References

[1] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing SLA violations," in *Proc. Int. Symp. 10th IFIP/IEEE IM*, May 2007, pp. 119–128.

[2] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.

[3] M. Abu Sharkh, M. Jammal, A. Shami, and A. Ouda, "Resource allocation in a network-based cloud computing environment: Design challenges," *IEEE Commun. Mag.*, vol. 51, no. 11, pp. 46–52, Nov. 2013.

[4] H. Zhang *et al.*, "Resource allocation for cognitive small cell networks: A cooperative bargaining game theoretic approach," *IEEE Trans. Wireless Commun.*, vol. 14, no. 6, pp. 3481–3493, Jun. 2015.

[5] H. Zhang, C. Jiang, J. Cheng, and V. Leung, "Cooperative interference mitigation and handover management for heterogeneous cloud small cell networks," *Arxiv Preprint Arxiv:1504.08076*, 2015.

[6] S. Ha, S. Sen, C. Joe-Wong, Y. Im, and M. Chiang, "Tube: Time-dependent pricing for mobile data," *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 247–258, Aug. 2012.

[7] H. Shen and Z. Li, "New bandwidth sharing and pricing policies to achieve a win-win situation for cloud provider and tenants," in *Proc. IEEE INFOCOM*, Apr. 2014, pp. 835–843.

[8] H. Xu and B. Li, "A study of pricing for cloud resources," *SIGMETRICS Perform. Eval. Rev.*, vol. 40, no. 4, pp. 3–12, Apr. 2013.

[9] H. Zhang *et al.*, "Resource allocation in spectrum-sharing OFDMA femtocells with heterogeneous services," *IEEE Trans. Commun.*, vol. 62, no. 7, pp. 2366–2377, Jul. 2014.

[10] G. Birkenheuer, A. Brinkmann, and H. Karl, "The gain of overbooking," in *Job Scheduling Strategies for Parallel Processing*. Berlin, Germany: Springer-Verlag, 2009, pp. 80–100.

[11] W. Ma, H. Zhang, W. Zheng, and X. Wen, "Differentiated-pricing based power allocation in dense femtocell networks," in *Proc. 15th Int. Symp. WPMC*, Sep. 2012, pp. 599–603.

[12] W. Elmaghraby and P. Keskinocak, "Dynamic pricing in the presence of inventory considerations: Research overview, current practices, and future directions," *Manage. Sci.*, vol. 49, no. 10, pp. 1287–1309, Oct. 2003.

[13] H. Zhang, C. Jiang, X. Mao, and H. Chen, "Interference-limited resource optimization in cognitive femtocells with fairness and imperfect spectrum sensing," *IEEE Trans. Veh. Technol.*, vol. 65, no. 3, pp. 1761–1771, Mar. 2016.

[14] A. Ghodsi *et al.*, "Dominant resource fairness: Fair allocation of multiple resource types," in *Proc. NSDI*, 2011, vol. 11, pp. 24–24.

[15] O. A. Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and D. Tsafrir, "Deconstructing Amazon EC2 spot instance pricing," *ACM Trans. Econ. Comput.*, vol. 1, no. 3, pp. 16:1–16:20, Sep. 2013.

[16] M. Al-Roomi, S. Al-Ebrahim, S. Buqrais, and I. Ahmad, "Cloud computing pricing models: A survey," *Int. J. Grid Distrib. Comput.*, vol. 6, no. 5, pp. 93–106, Oct. 2013.

[17] A. Gohad, N. C. Narendra, and P. Ramachandran, "Cloud pricing models: A survey and position paper," in *Proc. IEEE CCEM*, Oct. 2013, pp. 1–8.

[18] B. Sharma, R. K. Thulasiram, P. Thulasiraman, S. K. Garg, and R. Buyya, "Pricing cloud compute commodities: A novel financial economic model," in *Proc. 12th IEEE/ACM Int. Symp. Ccgrid Comput.*, Washington, DC, USA, 2012, pp. 451–457.

[19] G. Gallego and G. van Ryzin, "Optimal dynamic pricing of inventories with stochastic demand over finite horizons," *Manage. Sci.*, vol. 40, no. 8, pp. 999–1020, Aug. 1994.

[20] A. Narayan, S. Rao, G. Ranjan, and K. Dheenadayalan, "Smart metering of cloud services," in *Proc. IEEE Int. SysCon*, Mar. 2012, pp. 1–7.

[21] C. Joe-Wong, S. Sen, T. Lan, and M. Chiang, "Multi-resource allocation: Fairness-efficiency tradeoffs in a unifying framework," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 1206–1214.

[22] X. Jin, Y. K. Kwok, and Y. Yan, "A study of competitive cloud resource pricing under a smart grid environment," in *Proc. IEEE Int. Conf. CLOUDCOM Technol. Sci.*, Washington, DC, USA, 2013, vol. 1, pp. 655–662.

[23] R. Harmon, D. Raffo, and S. Faulk, "Incorporating price sensitivity measurement into the software engineering process," in *Proc. PICMET—Technol. Manage. Reshaping World*, Jul. 2003, pp. 316–323.

[24] R. E. Goldsmith and S. J. Newell, "Innovativeness and price sensitivity: Managerial, theoretical and methodological issues," *J. Product Brand Manage.*, vol. 6, no. 3, pp. 163–174, 1997.

[25] S. Han, S. Gupta, and D. R. Lehmann, "Consumer price sensitivity and price thresholds," *J. Retailing*, vol. 77, no. 4, pp. 435–456, Winter 2001.

[26] D. Gale, "The law of supply and demand," *Math. Scand.*, vol. 3, pp. 155–169, 1955.

[27] V. Mann, A. Kumar, P. Dutta, and S. Kalyanaraman, "VMFlow: Leveraging VM mobility to reduce network power costs in data centers," in *NETWORKING 2011*, vol. 6640, *Lecture Notes in Computer Science*, J. Domingo-Pascual, P. Manzoni, S. Palazzo, A. Pont, and C. Scoglio, Eds. Berlin-Verlag: Springer-Verlag, 2011, pp. 198–211.

[28] C. Hyser, B. Mckee, R. Gardner, and B. J. Watson, "Autonomic virtual machine placement in the data center," Hewlett Packard Lab., Palo Alto, CA, USA, Tech. Rep. HPL-2007-189, pp. 2007–189, 2007.

[29] S. K. Mandal, On-demand VM placement on cloud infrastructure, Ph.D. dissertation, Dept Comput. Sci. Eng., Nat. Inst. Technol. Rourkela, Rourkela, India, 2013.

[30] H. N. Van, F. D. Tran, and J. M. Menaud, "SLA-aware virtual resource management for cloud infrastructures," in *Proc. 9th IEEE Int. Conf. CIT*, Oct. 2009, vol. 1, pp. 357–362.

[31] W. Wang, B. Li, and B. Liang, "Dominant resource fairness in cloud computing systems with heterogeneous servers," *CoRR*, vol. abs/1308.0083, 2013.

[32] H. Wang *et al.*, "Distributed systems meet economics: Pricing in the cloud," in *Proc. 2nd USENIX Conf. Hot Topics Cloud Comput.*, 2010, pp. 6–6.

[33] H. R. Varian, "Equity, envy, and efficiency," *J. Econ. Theory*, vol. 9, no. 1, pp. 63–91, Sep. 1974.

Yuanfang Chi (S'15) received the B.A.Sc. degree in 2012 from The University of British Columbia (UBC), Vancouver, Canada, where she is currently working toward the M.A.Sc. degree with the Department of Electrical and Computer Engineering.

As a Research Assistant in the UBC Wireless Networks and Mobile Systems (WiNMoS) Laboratory, she is conducting research work in cloud-pricing-related topics. Before her postgraduate studies, she was an R&D Software Engineer with Tsinghua University, Beijing, China, for one year. Her current research interests include cloud resource management, demand management, and pricing strategies.

Xiuhua Li (S'13) received the B.S. and M.S. degrees from the Honors School and the School of Electronics and Information Engineering, Harbin Institute of Technology, Harbin, China, in 2011 and 2013, respectively. He is currently working toward the Ph.D. degree with the Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, Canada.

His current research interests include resource allocation, optimization, distributed antenna systems, cooperative backhaul caching, and traffic offloading in mobile content-centric networks.

Xiaofei Wang (S'07–M'13) received the B.S. degree in computer science and technology from Huazhong University of Science and Technology, Wuhan, China, in 2005 and the M.S. and Ph.D. degrees in computer science and engineering from Seoul National University, Seoul, Korea, in 2008 and 2013, respectively.

He is currently a Postdoctoral Research Fellow with the Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, Canada. His current research interests include social-aware multimedia service in cloud computing, cooperative backhaul caching, and traffic offloading in mobile content-centric networks.

Dr. Wang was a recipient of the Korean Government Scholarship for Excellent Foreign Students in IT Field by the National IT Industry Promotion Agency (NIPA) from 2008 to 2011, and he also received the Global Outstanding Chinese Ph.D. Student Award in 2012.

Victor C. M. Leung (S'75–M'89–SM'97–F'03) received the B.A.Sc. (Honour) in Electrical Engineering in 1977 and Ph.D. in Electrical Engineering in 1982, both from the University of British Columbia.

He is a Professor in electrical and computer engineering and a holder of the TELUS Mobility Research Chair at The University of British Columbia (UBC), Vancouver, Canada. He has coauthored more than 800 technical papers in archival journals and refereed conference proceedings, several of which had won best paper awards. His research is in the areas of wireless networks and mobile systems.

Dr. Leung is a Fellow of the Royal Society of Canada, a Fellow of the Canadian Academy of Engineering, and a Fellow of the Engineering Institute of Canada. He is serving or has served on the editorial boards of the *Journal of Communications and Networks* (JCN), the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS (JSAC), the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, the IEEE WIRELESS COMMUNICATIONS LETTERS, and several other journals. He has provided leadership to the technical program committees and organizing committees of numerous international conferences. He was the recipient of the 1977 Association of Professional Engineers of British Columbia (APEBC) Gold Medal, the Canadian Natural Sciences and Engineering Research Council (NSERC) Postgraduate Scholarships from 1977 to 1981, a 2012 UBC Killam Research Prize, and an IEEE Vancouver Section Centennial Award.

Abdallah Shami (M'03–SM'09) received the B.E. degree in electrical and computer engineering from the Lebanese University, Beirut, Lebanon, in 1997 and the Ph.D. degree in electrical engineering from the Graduate School and University Center, The City University of New York, New York, NY, USA, in September 2002.

Since July 2004, he has been with Western University, London, Canada, where he is currently a Professor in the Department of Electrical and Computer Engineering. His current research interests are in the areas of network-based cloud computing and wireless/data networking.