Developing User Perceived Value Based Pricing Models for Cloud Markets

Peijin Cong, Liying Li, Junlong Zhou, *Member, IEEE*, Kun Cao, Tongquan Wei, *Member, IEEE*, Mingsong Chen, *Member, IEEE*, and Shiyan Hu, *Senior Member, IEEE*

Abstract—With the rapid deployment of cloud computing infrastructures, understanding the economics of cloud computing has become a pressing issue for cloud service providers. However, existing pricing models rarely consider the dynamic interactions between user requests and the cloud service provider. Thus, the law of supply and demand in marketing is not fully explored in these pricing models. In this paper, we propose a dynamic pricing model based on the concept of user perceived value that accurately captures the real supply and demand relationship in the cloud service market. Subsequently, a profit maximization scheme is designed based on the dynamic pricing model that optimizes profit of the cloud service provider without violating service-level agreement. Finally, a dynamic closed loop control scheme is developed to adjust the cloud service price and multiserver configurations according to the dynamics of the cloud computing environment such as fluctuating electricity and rental fees. Extensive simulations using the data extracted from real-world applications validate the effectiveness of the proposed user perceived value-based pricing model and the dynamic profit maximization scheme. Our algorithm can achieve up to 31.32 percent profit improvement compared to a state-of-the-art approach.

Index Terms—Cloud computing, dynamic pricing model, user perceived value, profit maximization, augmented Lagrange function

1 Introduction

Cloud computing has become a popular commercial computing model that distributes user requests to a set of servers and delivers services over communication networks. As a business model, it turns resources of computing, storage, and communication into ordinary commodities and utilities in a pay-as-you-go manner [1], [2], [3], [4]. It is natural for cloud service providers to pursue the goal of profit maximization. Thus, the cloud service pricing strategy is of particular importance to cloud service providers.

The pricing model of a cloud service provider consists of two parts, namely, the revenue and the cost [5]. The revenue is the income that the cloud service provider gets through the sales of cloud services. The cost is the expenditure that includes not only the rental and electricity fees to operate multiserver systems, but also the reward and penalty paid by the cloud service provider to users based on service-level agreement. Profit maximization can be achieved by increasing revenue or reducing cost. On one hand, cloud service providers attempt to increase revenue by setting a high price for cloud services and attracting a great amount of service purchases. However, service price and purchase activity interplay, which cannot be optimized simultaneously [6]. On the other hand, cloud service providers try to reduce operational cost, such as electricity bill and rental fees, which are related to multiserver configurations. Thus, aspects including electricity price and multiserver configurations need to be considered in cloud pricing modeling.

Numerous investigations have been made into pricing mechanisms for profit maximization in cloud computing. Fixed pricing strategies such as pay-per-use, subscription based pricing, and tiered pricing are the most common pricing methods used by major cloud service providers [7], [8], [9]. For example, Li [7] proposes a flat rate pricing strategy that sets a fixed price for all service requests. Kesidis et al. [8] point out that usage-based pricing strategy can use cloud resources more efficiently when compared with flat rate pricing strategy. However, these fixed pricing methods cannot meet the dynamic needs of users and cannot capture the dynamics of supply and demand in market.

Handling the disadvantages of fixed pricing strategies necessitates the dynamic pricing strategies that adjust price of cloud services according to market situations and user requirements for service quality. Macias et al. [10] propose a genetic model based dynamic pricing strategy that obtains optimal pricing in an iterative way. This strategy offers competitive prices in the negotiation of services in cloud

- P. Cong, L. Li, K. Cao, and T. Wei are with the Department of Computer Science and Technology, East China Normal University, Shanghai 200062, China. E-mail: {51164500019, 52174506005}@stu.ecnu.edu.cn, 874023104@qq.com, tqwei@cs.ecnu.edu.cn.
- J. Zhou is with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China. E-mail: jlzhou@njust.edu.cn.
- M. Chen is with the Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai 200062, China. E-mail: mschen@sei.ecnu.edu.cn.
- S. Hu is with the Department of Electrical and Computer Engineering, Michigan Technological University, Houghton, MI 49931. E-mail: shiyan@mtu.edu.

Manuscript received 4 Dec. 2017; revised 7 Apr. 2018; accepted 16 May 2018. Date of publication 4 June 2018; date of current version 9 Nov. 2018. (Corresponding author: Tongquan Wei.) Recommended for acceptance by L. Wang. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TPDS.2018.2843343

computing markets. Amazon [11], [12] utilizes a spot pricing strategy that dynamically adjusts prices for a virtual service instance to accommodate changes in supply and demand. Based on a study of the spot price history of Amazon, Xu et al. [13] propose a dynamic pricing strategy to better understand the current market demand. Zhao et al. [14] design an efficient online algorithm for dynamic pricing of virtual machine resources across datacenters in a geo-distributed cloud to pursue long-term profit maximization. Although these works investigate dynamic pricing strategies from different perspectives, service-level agreement is not considered in their pricing mechanisms which is however important.

A service-level agreement is defined as an official commitment between a service provider and a client [15]. It uses a price compensation mechanism that provides certain compensations to users when their service requests are processed with low quality of service. Cao et al. [5] present a pricing model that takes service-level agreement and consumer satisfaction into considerations to maximize the profit of cloud service providers. Ghamkhari et al. [16] propose a two-tier ladder charging method to ensure user satisfaction. Specifically, a cloud service provider will charge users if their requests are processed before deadlines. Otherwise, the cloud service provider will not charge users for this execution. Lee and Irwin et al. [17], [18] claim that the price of the cloud service will decrease as the waiting time of service requests grows until the cloud service is free. These works study service-level agreement to ensure user satisfaction in the pricing process for profit maximization. However, they ignore the crucial concept of user perceived value in traditional market environment, which reflects the users' willingness to purchase cloud services. User perceived value is an important concept in pricing process since it ultimately will impact the profit of cloud service providers.

In this paper, we propose a user perceived value-based dynamic pricing mechanism that conforms to the law of supply and demand in economics. The contributions of this paper are summarized as follows.

- A dynamic pricing model that considers the interaction between users and the cloud service provider is proposed. The model is built upon the concept of user perceived value, user reward, and cloud service provider penalty in the domain of economics, which accurately captures the dynamics of supply and demand in cloud pricing strategies. In particular, user perceived value is nicely modeled using kernel density estimation method in our context.
- A profit maximization scheme is developed based on the dynamic pricing model to optimize the profit of the cloud service provider by configuring multiserver systems under service-level agreement constraint. Our scheme also includes a runtime control loop to adjust service price and muitlserver configurations to the dynamics of cloud computing environment such as fluctuating electricity and rental fees.
- Extensive simulations using the data extracted from real-world applications validate the effectiveness of our proposed user perceived value-based pricing model and the dynamic profit maximization scheme. Our algorithm achieves 31.32 percent profit improvement compared to a state-of-the-art approach.

The remainder of the paper is organized as follows. Section 2 presents the system architecture and models, Section 3 presents the problem definition and overview of the proposed scheme. Section 4 describes the proposed user perceived value-based pricing mechanism. The effectiveness of the proposed scheme is validated in Section 5 and concluding remarks are given in Section 6.

2 System Architecture and Models

We consider a common three-tier cloud service provision structure that consists of users, cloud service providers, and infrastructure vendors [5], [9], [19], [20]. Among the three entities that form a market in cloud computing, the infrastructure vendor charges the cloud service provider for renting infrastructures to deploy service capacity, and the cloud service provider charges users for processing their service requests. In this paper, users and the cloud service provider are of our particular interest. We introduce our user model and cloud service provider model in the following sections.

2.1 Cloud User Model

To maximize the profit of a cloud service provider, the cloud service provider needs to know the aggregate demands of users. When a cloud service provider sets up the price of a service, different users have different responses to this price. Based on the concepts of user perceived value, we give the first derivation of such a model in the cloud computing context. In the following, we introduce the concepts of user perceived value and then present our derivations.

2.1.1 *User Perceived Value*

In conventional markets, the arrival rate of customers to a store is often a response to their regular buying patterns rather than a reaction to individual prices [6]. Thus, it is reasonable to assume that the change of the list price has no effect on the total number of customers who are visiting the store. Typically, not all of the customers are willing to buy a specific commodity. That is, the total number of customers who buy commodities are no larger than the total number of people that visit the store.

Customer perceived value is the fundamental basis for all marketing activities [21]. It reflects the worth that a product or service has in the mind of a consumer and this concept has been widely used in modeling other markets [22]. In general, customers are unaware of the true cost of production for the products they buy, which means that they simply have an internal feeling for how much certain products are worth to them. In the conventional market environment, only the customer whose perceived value is higher than the real price of the product is willing to pay for the product.

In this paper, we use X_i to denote the perceived value that user i has for the service. X_i is a continuous random variable and $0 \leq X_i < \infty$ holds. As with other pricing models [23], $X_1, X_2, \ldots, X_n$ are assumed to be independent and identical random variables. The probability density function of the perceived value X, denoted by $f(x)$, is known or can be estimated a priori. Perceived value is a process of valuing and is much harder to determine. Roig et al. [24] observe that the customer value is perceived by customers, and cannot be determined objectively by the seller. Factors such as scarcity, marketing efforts, novelty, and brand associations all play into customer perceived value [25]. Usually, consumers will offer a range of price options. Thus, in the experimental section, a normal distribution is used to describe the initial distribution of user perceived value. Subsequently, the distribution of user perceived value is fitted using kernel density estimation based on historical price data. Kernel density estimation is a non-parametric way to estimate the probability density function of a random variable based on a finite data sample [26].

In the following sections, we adopt the terminology of customer perceived value used in traditional market. The cloud computing environment is taken as a store and the cloud service is deemed as a special commodity provided in the store. The terminology of customer perceived value and user perceived value are used interchangeably.

2.1.2 User Demand Distribution

Unlike traditional methods that use the expected demand to model user behavior [23], [27], the probability distribution of total demands is used in this work to model user service requests. We consider a slotted time model that deals with the pricing decision and constraints for sales periods T of equal length. Let τ denote the length of each time slot over the sales period T, and N be the number of time slots τ over the sales period T. That is, $T = N \cdot \tau$. A cloud service provider sets list price for the service at the beginning of regular sales periods. The list price during each sales period is assumed to be constant, but varies from period to period.

Suppose that the cloud service provider will charge ω per user for a specific cloud service during a sales period T. Let n denote the total number of users that have interest in the service at the price of ω during the sales period T, and λ_u denote the number of users arriving per unit time, respectively. n is assumed to be independent of all other parameters of the system, and follows a discrete Poisson distribution as [27]

$$P(n|\lambda_u) = \frac{(\lambda_u T)^n e^{-\lambda_u T}}{n!}, \ n = 0, 1, 2, \ldots, \infty. \tag{1}$$

The user arrival rate λ_u may not be constant in many situations. Taking into account the heterogeneity of arrival rate, a Gamma distribution characterized by parameters (α, β) is utilized to represent the arrival rate λ_u. α is the shape parameter that determines the shape of the distribution curve while β is the scale parameter that decides the size of the distribution curve. The probability density function of arrival rate λ_u is given by

$$g(\lambda_u) = \frac{1}{\Gamma(\alpha)\beta^\alpha} \lambda_u^{(\alpha-1)} e^{-\lambda_u/\beta}, \ 0 \le \lambda_u \le \infty, \tag{2}$$

where the expectation and variance of λ_u is given by $E[\lambda_u] = \alpha\beta$ and $Var[\lambda_u] = \alpha\beta^2$, respectively, and $\Gamma(\alpha)$ is a complete gamma function.

Among the n users, any one whose perceived value of the service is no less than the list price ω is considered as a potential buyer of the service. Let m denote the number of potential buyers. It is a non-negative discrete random variable taking the value of $0, 1, 2, \ldots, \infty$ and $m \le n$ holds. Based on user perceived value, we use $f(x)$ to denote the probability density function of the perceived value X, and $F(\omega)$ to represent the cumulative distribution function of x evaluated at ω. $F(\omega)$ describes the probability that users do not want to pay ω to buy a cloud service. It is a non-decreasing function of ω, and $0 \leqslant F(\omega) \leqslant 1$ and $\lim_{\omega\to\infty} F(\omega) = 1$ hold [28]. Let $P_\omega(m|n)$ be the probability that m out of n users are inclined to buy in the sales period when the service price is set equal to ω. It follows a binomial distribution of probability, which is given by

$$P_\omega(m|n) = \binom{n}{m} [1 - F(\omega)]^m [F(\omega)]^{(n-m)}. \tag{3}$$

Combining (1), (2), and (3), we can derive the probability of having m potential buyers during the sales period T when the service price is set equal to ω. The probability is denoted by $P_\omega(m)$ and given by

$$\begin{aligned} P_\omega(m) &= \int_{\lambda_u=0}^{\infty} \sum_{n=0}^{\infty} P_\omega(m|n) P(n|\lambda_u) g(\lambda_u) d\lambda_u \\ &= \binom{m+\alpha-1}{m} [\frac{\beta T[1-F(\omega)]}{1+\beta T[1-F(\omega)]}]^m [\frac{1}{1+\beta T[1-F(\omega)]}]^\alpha. \end{aligned} \tag{4}$$

Clearly, it is a negative binomial distribution. Thus, the expected number of actual buyers of the service at price ω during period T, denoted by $E_\omega(m)$, can be computed as

$$E_\omega(m) = \alpha\beta T(1 - F(\omega)), \tag{5}$$

where α and β are parameters of the Gamma distribution of user arrival rate λ_u, and $F(\omega)$ is the cumulative distribution function of x evaluated at ω. The revenue of the cloud service provider in a sales period T is thus given by

$$Revenue = \omega \cdot E_\omega(m) = \omega\alpha\beta T(1 - F(\omega)). \tag{6}$$

2.2 Cloud Service Provider Model

The cloud service provider rents a multiserver system that is constructed and maintained by an infrastructure vendor to serve user service requests. The architecture details of the multiserver system are quite flexible [5]. They can be blade centers where each server is a server blade [29], clusters of traditional servers where each server is an ordinary processor [30], and multicore server processors where each server is a single core [31]. For the ease of presentation, these blades/processors/cores are simply called servers. Users submit their service requests to the cloud service provider, and the cloud service provider serves these service requests (i.e., run these tasks) on the multiserver system.

2.2.1 Multiserver Model

We consider a multiserver system that consists of M homogeneous servers operating at a common speed of s. The multiserver system can be modeled as an M/M/M queuing system where arrivals of user service requests governed by a Poisson process form a single queue and M servers can process these service requests in parallel. Let ϱ be the service rate of user service requests that arrive at the rate of λ_u. It is clear that ϱ user service requests can be processed by servers if the number of user service requests in the system is not greater than M. The service time of a user service request on a server is an exponential random variable denoted by $x_1 = r/s$ with mean $\overline{x_1} = \overline{r}/s$, where r is the number of instructions to be executed for the service request. A first-come-first-served (FCFS) queue of infinite capacity is maintained by the multiserver system for waiting tasks when all the servers are busy. Let ρ be server utilization, which is defined as the average percentage of time that a server is busy. It can be expressed as

$$\rho = \frac{\lambda_u}{M\varrho} = \frac{\lambda_u}{M\frac{s}{\overline{r}}} = \frac{\lambda_u \overline{r}}{Ms}. \quad (7)$$

Let P_k be the probability of k service requests being waiting or processing in the M/M/M queuing system. Based on queuing theory [5], [32], [33], P_k is given by

$$P_k = \begin{cases} P_0 \frac{(M\rho)^k}{k!}, & k \le M \\ P_0 \frac{M^M \rho^k}{M!}, & k \ge M \end{cases}, \quad (8)$$

where P_0 is the probability that there are no tasks in the queue, and is formulated into [32]

$$P_0 = \left(\sum_{k=0}^{M-1} \frac{(M\rho)^k}{k!} + \frac{(M\rho)^M}{M!} \cdot \frac{1}{1-\rho} \right)^{-1}. \quad (9)$$

The probability that there are exact M service requests in the system is thus given by $P_M = P_0 \frac{(M\rho)^M}{M!}$. Through Taylor series expansions of $\sum_{k=0}^{M-1} (M\rho)^k / k! \approx e^{M\rho}$ and $M! \approx \sqrt{2\pi M}(\frac{M}{e})^M$, it can be rewritten as

$$P_M = \frac{1-\rho}{\sqrt{2\pi M}(1-\rho)(\frac{e^{\rho-1}}{\rho})^M + 1}. \quad (10)$$

This form of P_M is necessary for deriving multiserver configurations in Section 4.

When all the servers in the system are busy, a newly submitted service request must wait and will be inserted into the FCFS queue. Let P_q denote the probability of queuing a newly arrived task when no servers are idle at the time of arrival. P_q can be formulated as

$$P_q = \sum_{k=M}^{\infty} P_k = \frac{P_M}{1-\rho}. \quad (11)$$

Let $\overline{N}$ be the average number of requests being waiting or executing in the multiserver system. $\overline{N}$ is calculated as

$$\overline{N} = \sum_{k=0}^{\infty} kP_k = M\rho + \frac{\rho}{1-\rho} P_q. \quad (12)$$

The average service response time $\overline{R}$ is defined as the average time elapsed between the time when a service request is submitted and the time when the service request is finished. In this paper, it is adopted to evaluate the service quality. It is in fact the sum of task execution time and waiting time, and can be derived by applying Little's Law [34] as

$$\overline{R} = \frac{\overline{N}}{\lambda_u} = \overline{x_1}\left(1 + \frac{P_q}{M(1-\rho)}\right) = \overline{x_1}\left(1 + \frac{P_M}{M(1-\rho)^2}\right). \quad (13)$$

The average service response time $\overline{R}$ is utilized in this paper as a metric for service-level agreement. If the response time of a service exceeds the predefined deadline, service-level agreement is deemed to be violated.

2.2.2 *Bill and Rent*

A cloud service provider needs to rent infrastructure and pay electricity to maintain the operation of the computing infrastructure. Let δ be the fee the cloud service provider pays to rent a server per second during a sales period T, the rent the cloud service provider needs to pay for a system of M servers during the sales period T is

$$Rent = M\delta \cdot T. \quad (14)$$

As a portion of the cloud service cost, electricity fee has become a significant expense for today's data centers. It can be derived by multiplying energy consumed by a server with electricity price. The energy consumed by a server can be modeled at different levels of abstraction. At the abstraction level of digital CMOS circuit, the power consumption, which is denoted by P_{tot}, can be modeled as

$$P_{tot} = P_{sta} + P_{dyn}, \quad (15)$$

where P_{sta} is the static power dissipation while P_{dyn} is the dynamic power dissipation. P_{sta} is independent of switching activity and maintains the basic circuit state, thus can be deemed as a constant [5]. P_{dyn} is related to processor switching activity and dominates the total power consumption, which can be formulated as a function of supply voltage v and processing speed s. In addition, the supply voltage is usually linearly proportional to the processing speed, i.e., $v \propto s$. The dynamic power consumption P_{dyn} is then expressed as ξs^γ, where ξ is a processor dependent coefficient and γ is a constant that equals to $2\phi + 1$ ($\phi > 0$). Based on the static and dynamic power consumption described above, Eq. (16) is used to denote the total power consumption of a multiserver system, which is,

$$P_{tot} = M((P_{dyn} - P_{sta})\rho + P_{sta}), \quad (16)$$

where M is the server number and ρ is the server utilization.

Let E^T denote the energy consumed by all M servers in the system during the sales period T. It is given by

$$E^T = M((P_{dyn} - P_{sta})\rho + P_{sta}) \cdot T. \quad (17)$$

Let $C^T(E^T)$ be the price of the energy consumed by all servers in the period T, then $C^T(E^T)$ can be formulated as

$$C^T(E^T) = \begin{cases} k_1^T, & 0 \le E^T \le l_{th}^T \\ k_2^T, & E^T > l_{th}^T, \end{cases} \quad (18)$$

where $k_1^T, k_2^T > 0$ are differentiated price and l_{th}^T is the energy consumption threshold in the sales period T. The electricity bill of the multiserver system in the sales period T is hence formulated as

$$\begin{aligned} Bill &= E^T \cdot C^T(E^T) \\ &= M((P_{dyn} - P_{sta})\rho + P_{sta}) \cdot T \cdot C^T(E^T). \end{aligned} \quad (19)$$

2.3 Reward and Penalty Mechanism

Oftentimes, users have different sensitivities to postponing their requests. For users whose service requests can be deferred to a certain extent, the cloud service provider will reward them based on the degree of deferment. However, once the deferment of service requests exceeds a threshold, the cloud service provider will compensate users based on the degree of the deferment. In the following, we will discuss the reward and penalty mechanism from perspectives of users and the cloud service provider, respectively.

2.3.1 User Reward Model

We divide users into I types, each type of users has a sensitivity to the service deferment of their service requests. We define a sensitivity factor, denoted by ψ_i, to quantify the sensitivity of users of type i to the deferment of their service requests, which is denoted by D_i. For the users of type i, the factor ψ_i is negatively proportional to the sensitivity of the users to the deferment of their service requests. That is, a larger sensitivity factor indicates a more delay-sensitive user service request. For users running interactive applications with no delay tolerance, set $\psi_i = \infty$.

The cloud service provider will return more rewards to those users who are less sensitive to service deferment. Let L_i be the monetary loss of the users of type i due to their degree of sensitivity to service deferment. The users with a larger degree of sensitivity (ψ_i) to service deferment (D_i) will get less rewards from the cloud service provider, resulting in a larger amount of monetary loss (L_i) of the users of type i. The monetary loss function of the users of type i is given by $L_i = \psi_i D_i$.

We define a reward function, denoted by $\hbar_i$, to represent the reward the cloud service provider returns to users of type i. The reward $\hbar_i$ is a function of the service deferment D_i, and is given by $\hbar_i = \theta \cdot \log(1 + D_i)$, where $\theta > 0$ and $0 \leq D_i \leq D_{max}$ hold. θ is called the reward factor and D_{max} is the maximum value of service deferment. The log function is adopted to prevent users from setting excessive service deferment, which is unfair to cloud service providers.

Users need to make decisions on their own service deferment D_i to get the maximum reward from the cloud service provider based on the monetary loss and reward function. Thus, the optimization problem is to maximize $(\hbar_i - L_i)$ subject to $(0 \leq D_i \leq D_{max})$, where D_i is regarded as a continuous variable to simplify the optimization problem, and the solution to the problem is given by

$$D_i = \max\left(\min\left(\frac{\theta}{\psi_i} - 1, D_{max}\right), 0\right). \tag{20}$$

Substitute D_i back into reward function $\hbar_i$, then one has

$$\hbar_i = \theta \log\left(1 + \max\left(\min\left(\frac{\theta}{\psi_i} - 1, D_{max}\right), 0\right)\right). \tag{21}$$

Let $Reward$ denote the total monetary reward returned to users from the cloud service provider over the sales period T, then one has

$$Reward = \sum_{N'=1}^{N} \sum_{i=1}^{I} (\hbar_i - L_i)\lambda_u^i[N']\tau, \tag{22}$$

where $\lambda_u^i[N']$ denotes the arrival rate of the users of type i in the $N'th$ time slot. In practice, the cloud service provider can learn the sensitivity factor ψ_i from experiments or historical data. The adopted user reward model is similar to the one presented in [35].

2.3.2 Cloud Service Provider Penalty Model

The high degree of user satisfaction is determined by the fast response of a cloud service provider to service requests. Once the service response time exceeds the threshold value specified in service-level agreement, users will be compensated by the cloud service provider for low quality of service. Given server benchmarking speed (s_0), the average response time of service requests ($\overline{R}$), and the number of instructions for each service request (r), the penalty function of users of type i, denoted by u_i, can be formulated as [5]

$$u_i(r, \overline{R}) = \begin{cases} 0, & 0 \leqslant \overline{R} \leqslant \frac{r}{s_0} + D_i \\ d(\overline{R} - \frac{r}{s_0} - D_i), & \frac{r}{s_0} + D_i < \overline{R} \leqslant (1 + \frac{\omega}{d})\frac{r}{s_0} + D_i \\ \omega, & \overline{R} > (1 + \frac{\omega}{d})\frac{r}{s_0} + D_i, \end{cases} \tag{23}$$

where d is the degree of punishment, D_i denotes the service deferment of users of type i, and ω is the cloud service price charged by the cloud service provider to users. In the future we will adopt super linear function to describe the relation between compensation and average response time.

The details of Eq. (23) are described below. For users of type i, if the average response time satisfies $0 \leqslant \overline{R} \leqslant \frac{r}{s_0} + D_i$, the cloud service provider will regard this execution of the service request as a successful process with high quality of service and users will not be compensated by the cloud service provider. Otherwise, if the average response time satisfies $\frac{r}{s_0} + D_i < \overline{R} \leqslant (1 + \frac{\omega}{d})\frac{r}{s_0} + D_i$, the cloud service provider will regard this execution of the service request as a process with low quality of service. In this case, the compensation provided by the cloud service provider to users will increase linearly as the average response time $\overline{R}$ increases. Finally, if the average response time satisfies $\overline{R} > (1 + \frac{\omega}{d})\frac{r}{s_0} + D_i$, the cloud service provider will regard this execution of the service request as a failed process and will not charge users for this execution.

We use $Penalty$ to denote the total compensation provided by the cloud service provider to users, then we have

$$Penalty = \sum_{N'=1}^{N} \sum_{i=1}^{I} u_i(r, \overline{R})\lambda_u^i[N']\tau, \tag{24}$$

where $\lambda_u^i[N']\tau$ is the average number of user requests of type i in the time slot τ, and $u_i(r, \overline{R})$ is the incurred penalty for the service requests due to low quality of service.

2.3.3 Gross Profit

The gross profit a cloud service provider earns is the total revenue subtracted by the cost of generating that revenue. In other words, gross profit is sales minus cost of the cloud service sold. Assuming the price of cloud service is constant in a sales period T, the revenue earned is given by $\omega \cdot E_\omega(m)$, where ω denotes the service price per user and $E_\omega(m)$ indicates the expected number of actual buyers at price ω during the sales period T.

Besides the reward for flexible users and penalty for low quality of service mentioned above, the cost of cloud service provider sold also consists of the cost paid to rent cloud computing infrastructures, and the electricity expense incurred by the cloud service provider to maintain the operation of the computing infrastructures. We define the profit of the cloud service provider in a sales period T as the revenue minus the various expenses including the reward cost, penalty cost, electricity cost, and rental cost, that is,

$$Profit = Revenue - Reward - Penalty - Bill - Rent, \quad (25)$$

where $Revenue$, $Reward$, $Penalty$, $Bill$, and $Rent$ are given in Eqs. (6), (22), (24), (19), and (14), respectively.

3 Problem Definition and Overview of the Proposed Approach

In this section, we formally define the profit maximization problem, followed by a brief overview of our proposed approach to the profit maximization problem.

3.1 Problem Definition

The price of a cloud service interplays with users who purchase the service, which in turn affects the revenue of the cloud service provider. This paper aims to maximize the profit of the cloud service provider by deriving the optimal number of servers, operating speed of servers, and price of cloud services provided without violating service-level agreement. Assume that the cloud service provider optimizes its decisions at the beginning of each sales period T. Let b_1 denote the upper bound on the power consumption of the M servers, and b_2 be the upper bound on the expected response time of user requests. Our optimization problem can be formulated as

$$\text{Maximize}: \quad Profit \quad (26)$$

$$\text{Subject to}: \quad \theta \geq 0 \quad (27)$$

$$P_{tot} \leq b_1 \quad (28)$$

$$\overline{R} \leq b_2 \quad (29)$$

$$0 \leq \phi_i[N'] \leq \lambda_u^i[N']\tau, \quad \forall i \in I, N' \in N \quad (30)$$

$$\sum_{N'=1}^{N+\lfloor D_i \rfloor} (\lambda_u^i[N']\tau - \phi_i[N']) \geq \sum_{N'=1}^{N} \lambda_u^i[N']\tau, \quad \forall i \in I, N' \in N, \quad (31)$$

where $Profit$, P_{tot}, and $\overline{R}$ are given in Eqs. (25), (16), and (13), respectively.

In the above formulation, the reward factor θ is nonnegative (Eq. (27)), the total energy consumption P_{tot} of the multiserver within the sales period T can not exceed b_1 (Eq. (28)), and the average service response time $\overline{R}$ can not exceed b_2 (Eq. (29)). Eq. (30) ensures that the amount of delayed service requests is nonnegative and not larger than the total number of service requests in each time slot τ, where $\phi_i[N']$ and $\lambda_u^i[N']\tau$ are the number of delayed and total service requests in the N'th time slot, respectively. Eq. (31) ensures that the processing of the arrived user requests of type i in a sales period T can not be delayed longer than the allowed service deferment D_i of user service requests. We will then use the augmented Lagrange multiplier method to solve the optimization problem, which will be described in detail in Section 4.

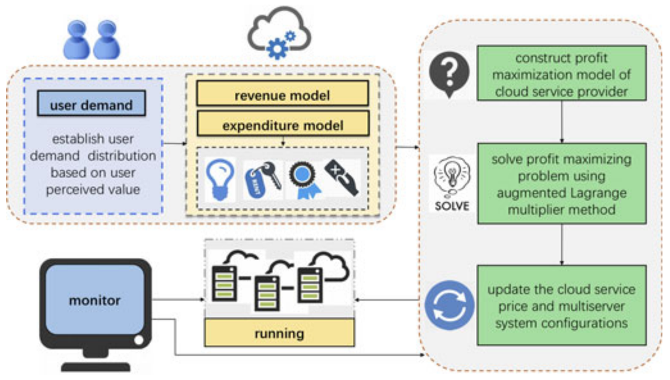

user demand
establish user demand distribution based on user perceived value
revenue model
expenditure model
monitor
running
construct profit maximization model of cloud service provider
SOLVE
solve profit maximizing problem using augmented Lagrange multiplier method
update the cloud service price and multiserver system configurations

Fig. 1. Overview of the proposed approach.

3.2 Overview of the Proposed Approach

The optimization problem given in Eq. (26) tries to maximize the $Profit$ of the cloud service provider under the constraints mentioned above. Fig. 1 outlines the overview of our proposed approach to solve the optimization problem. We first establish user demand distribution based on the concept of user perceived value. Subsequently, based on user demand distribution, the revenue model and the expenditure model are developed to construct the profit maximization problem. This optimization problem is then solved using the augmented Lagrange multiplier method. Finally, since the parameters of electricity bill and rental fees change over time, these parameters are monitored and a dynamic closed loop control scheme is proposed to adapt the service price and mulitserver configurations to the changes in these parameters. The details of the proposed scheme are provided in Section 4.

4 User Perceived Value-Aware Profit Optimization Scheme

In this section, we first leverage the augmented Lagrange multiplier method to compute the optimal solution, including the service price, the number of servers, and the speed of servers. Subsequently, a dynamic closed loop control scheme is proposed to adapt the service price and multiserver configurations to the dynamic cloud computing environment.

4.1 Build Augmented Lagrange Function

Unconstrained optimization can be solved in many ways, such as steepest descent method [36], Newton's method [37], multiplier method [38], etc. However, it is difficult to optimize constrained optimization directly. A common method to solve constrained optimization is to transform the constrained optimization problem into an unconstrained optimization problem. Numerous techniques on constrained optimization have been investigated in the literature [39], [40], [41]. Of these techniques, the augmented Lagrange multiplier method is a powerful tool for solving this class of problems, which is adopted in this work to solve the profit optimization problem. Refer to Eq. (26), we first build augmented Lagrange function to convert the constrained optimization problem into an unconstrained optimization problem, and then use the multiplier method to solve the unconstrained optimization problem.

The *Bill* given in Eq. (19) is a function of power consumption of the multiserver system, the length of the sales period T, and real-time price of electricity. Since real-time price is flat within each sales period T and T itself is constant, the *Bill* for T is fixed and can be expressed as $Bill = b_3P_{tot}$, where P_{tot} given in Eq. (16) is the total power consumed by the multiserver system and b_3 is a constant coefficient. The profit optimization problem given in Eq. (26) can then be re-written as

$$\begin{cases} O(\omega, M, s) & = \omega E_{\omega}[m] - b_3P_{tot} - \delta MT \\ & \quad - \sum_{N'=1}^{N}\sum_{i=1}^{I}(\hbar_i - L_i)\lambda_u^i[N']\tau \\ & \quad - \sum_{N'=1}^{N}\sum_{i=1}^{I}u_i(r,\overline{R})\lambda_u^i[N']\tau \\ g_1(M,s) & = b_2 - \overline{x_1}\Big(1+\frac{P_M}{M(1-\rho)^2}\Big) \geq 0 \\ g_2(M,s) & = b_1 - M((\xi s^{\gamma} - P_{sta})\rho + P_{sta}) \geq 0, \end{cases} \tag{32}$$

where $O(\omega, M, s)$ denotes the objective function of *Profit* given in Eq. (25), and $g_1(M,s)$ and $g_2(M,s)$ are constraint equations of M and s, respectively.

Next, we convert the problem given in Eq. (32) with inequality constraints into an augmented Lagrange function. Let $\mathbf{y}$ be the vector that converts the problem with inequality constraints to a problem with equality constraints, and $\mathbf{v}$ be the Lagrange multiplier vector, the augmented Lagrange function is thus given by

$$\begin{aligned}\phi(\omega, M, s, \mathbf{y}, \mathbf{v}, \sigma) = O(\omega, M, s) - \sum_{j=1}^{2} v_j(g_j(M,s) - y_j^2) \\ + \frac{\sigma}{2}\sum_{j=1}^{2}(g_j(M,s) - y_j^2)^2,\end{aligned} \tag{33}$$

where the constant parameter σ denotes the penalty factor and $\sigma > 0$ holds. The augmented Lagrange function given in Eq. (33) can be converted into the form of

$$\begin{aligned}\phi(\omega, M, s, \mathbf{y}, \mathbf{v}, \sigma) &= O(\omega, M, s) \\ &+ \sum_{j=1}^{2}\left[\frac{\sigma}{2}\left[y_j^2 - \frac{1}{\sigma}(\sigma g_j(M,s) - v_j)\right]^2 - \frac{v_j^2}{2\sigma}\right],\end{aligned} \tag{34}$$

by using the method of completing the square, a technique to derive the quadratic formula [42], and the function given in (34) can be easily maximized when

$$y_j^2 = \frac{1}{\sigma}\max(0, \sigma g_j(M,s) - v_j), j = 1, 2. \tag{35}$$

Plugging y_j^2 given in Eq. (35) back into the original Eq. (33), one obtains the augmented Lagrange function

$$\begin{aligned}\phi(\omega, M, s, \mathbf{v}, \sigma) &= O(\omega, M, s) \\ &+ \frac{1}{2\sigma}\sum_{j=1}^{2}[[\max(0, v_j - \sigma g_j(M,s))]^2 - v_j^2].\end{aligned} \tag{36}$$

Through this quadratic relaxation of the original problem given in Eq. (32), we can derive analytical form of solutions to the profit maximization problem. We aim to maximize the profit of the cloud service provider and obtain the optimum solutions including service price ω, number of servers M, and speed of servers s. Specifically, we seek to solve the augmented Lagrange function given in Eq. (36) by first computing partial derivatives of Eq. (32) with respect to ω, M, and s. Here, the details are omitted. We only show key steps for solving the partial derivatives of Eq. (32) with respect to ω, M, and s.

Calculate the Partial Derivative of Eq. (32) with Regard to ω. The partial derivative of $E_{\omega}[m]$ with regard to ω is $\frac{\partial E_{\omega}[m]}{\partial \omega} = -\alpha\beta t f(\omega)$, thus, the partial derivative of $O(\omega, M, s)$ with regard to ω is

$$\begin{aligned}\frac{\partial O(\omega, M, s)}{\partial \omega} &= \frac{\partial E_{\omega}[m]}{\partial \omega}\cdot\omega + E_{\omega}[m] \\ &= \alpha\beta t(1 - \omega f(\omega) - F(\omega)),\end{aligned}$$

where $f(\omega)$ and $F(\omega)$ are the probability density and the cumulative distribution function at ω, respectively.

Calculate the Partial Derivative of Eq. (32) with Regard to M. The partial derivative of $g_1(M,s)$ with regard to M can be expressed as

$$\begin{aligned}\frac{\partial g_1(M,s)}{\partial M} = \frac{\partial}{\partial M}\left[-\frac{\overline{x_1}}{M}\left(\frac{1}{[\sqrt{2\pi M}(1-\rho)(e^{\rho}/e\rho)^M + 1](1-\rho)}\right)\right] \\ + \frac{\overline{x_1}}{M^2}\frac{P_M}{(1-\rho)^2}.\end{aligned} \tag{37}$$

Let $D_1 = \sqrt{2\pi M}(1-\rho)(e^{\rho}/e\rho)^M + 1 = \sqrt{2\pi M}(1-\rho)L + 1$, $D_2 = 1-\rho$, and $L = (e^{\rho}/e\rho)^M$, then Eq. (37) becomes

$$\frac{\partial g_1(M,s)}{\partial M} = \frac{\partial}{\partial M}\left[-\frac{\overline{x_1}}{M}\left(\frac{1}{D_1D_2}\right)\right] + \frac{\overline{x_1}}{M^2}\frac{P_M}{D_2^2}. \tag{38}$$

The partial derivative of L, D_1, and D_2 with regard to M are calculated as follows:

$$\frac{\partial L}{\partial M} = L(\rho - \ln\rho - 1) + LM\left(1 - \frac{1}{\rho}\right)\frac{\partial \rho}{\partial M},$$

$$\begin{aligned}\frac{\partial D_1}{\partial M} &= \sqrt{2\pi}\left(\frac{1}{2\sqrt{M}}(1-\rho)L + \sqrt{M}\left(-\frac{\partial\rho}{\partial M}\right)L + \sqrt{M}(1-\rho)\frac{\partial L}{\partial M}\right) \\ &= \sqrt{2\pi}\left(\frac{1}{2\sqrt{M}}(1+\rho)L - \sqrt{M}(1-\rho)\ln\rho L\right),\end{aligned}$$

$$\frac{\partial D_2}{\partial M} = -\frac{\partial \rho}{\partial M} = \frac{\rho}{M}.$$

Substitute $\frac{\partial L}{\partial M}$, $\frac{\partial D_1}{\partial M}$, and $\frac{\partial D_2}{\partial M}$ back into the Eq. (38), one has

$$\frac{\partial g_1(M,s)}{\partial M} = \frac{\overline{x_1}}{MD_1D_2}\left[\frac{\frac{\partial D_1}{\partial M}D_2 + \frac{\partial D_2}{\partial M}D_1}{D_1D_2} + \frac{1}{M}\right].$$

The partial derivative of $g_2(M,s)$ with regard to M can be easily calculated as

$$\frac{\partial g_2(M,s)}{\partial M} = (\xi s\gamma - P_{sta})\cdot\rho + P_{sta}.$$

Caculate the Partial Derivative of Eq. (32) with Regard to s. The partial derivative of L, D_1, and D_2 with regard to s are calculated as follows:

$$\frac{\partial L}{\partial s} = LM\left(1 - \frac{1}{\rho}\right)\frac{\partial \rho}{\partial s} = \frac{LM}{s}(1-\rho),$$

$$\frac{\partial D_1}{\partial s} = \sqrt{2\pi M}\left[\left(-\frac{\partial \rho}{\partial s}L + (1-\rho)\frac{\partial L}{\partial s}\right)\right] = \sqrt{2\pi M}[\rho + M(1-\rho)^2]\frac{L}{s},$$

$$\frac{\partial D_2}{\partial s} = -\frac{\partial \rho}{\partial s} = \frac{\rho}{s}.$$

The partial derivatives of $g_1(M,s)$ and $g_2(M,s)$ with regard to s are hence computed as

$$\frac{\partial g_1(M,s)}{\partial s} = -\frac{\overline{x_1}}{M}\cdot\frac{\partial}{\partial s}\left(\frac{P_M}{(1-\rho)^2}\right) = -\frac{\overline{x_1}}{M}\left[\frac{\frac{\partial D_1}{\partial s}D_2 + \frac{\partial D_2}{\partial s}D_1}{(D_1D_2)^2}\right],$$

$$\frac{\partial g_2(M,s)}{\partial s} = M\rho\xi\gamma s^{\gamma-1}.$$

Once we obtain the above partial derivatives of Eq. (32) with regard to ω, M and s, we can compute and obtain the optimal solutions by letting these partial derivatives of Eq. (32) with regard to ω, M, and s equal 0.

4.2 Solve Augmented Lagrange Function

We present in this section an augmented Lagrange multiplier method based algorithm that solves the profit optimization problem given in (33) and derives its optimum solutions, including the service price and multiserver configurations. The proposed algorithm first computes an optimum Lagrange multiplier, which guarantees that the solution of original objective function and the solution of Lagrange function are consistent in the case where the optimal multiplier is obtained. Subsequently, the optimal service price and multiserver configurations are determined.

Let $M^{(k)}$, $s^{(k)}$, and $v^{(k)}$ indicate the kth iteration of M, s, and v in the algorithm. Let ε, η, and Ψ be three positive numbers, l be the number of iterations, and L be the maximum number of iterations. Algorithm 1 describes the proposed augmented Lagrange algorithm. Inputs to the algorithm are electricity price C^τ during time slot τ, the rent δ, and user requests arrival rate λ_u. The algorithm iteratively derives the optimal cloud service price ω and multiserver configurations which includes the optimal number of servers M, the server speed s, and the $Profit$ of the cloud service provider.

The algorithm works as follows. It first formulates the optimization problem into the form in Eq. (26), then sets parameters of ε, η, Ψ, and L, and initializes variables of $M^{(0)}$, $s^{(0)}$, v^1, and l (lines 1-3). In each round of iteration, the algorithm calls the augmented Lagrange function solver, denoted by $\mathbf{ALF}-\mathbf{Solver}(\phi(\omega, M^{(l-1)}, s^{(l-1)}, v^{(l)}, \sigma))$, to obtain a local optimum of the ω, M, and s (line 5). The $\mathbf{ALF}-\mathbf{Solver}(\phi(\omega, M^{(l-1)}, s^{(l-1)}, v^{(l)}, \sigma))$ derives the local optimum by computing partial derivatives of $\phi(\omega, M, s, \mathbf{v}, \sigma)$ with regard to ω, M, and s, and solving a system of equations of ω, M, and s (lines 18-21).

The algorithm finishes if the Lagrange multiplier vector $\mathbf{v}$ converges and approximates the optimum by an error of ε. Let $Q_j(M^{(l)}, s^{(l)}) = g_j(M^{(l)}, s^{(l)}) - y_j^2$ for $j = 1,2$ be the penalty item of the augmented Lagrange function given in Eq. (33), then the Lagrange multiplier vector $\mathbf{v}$ converges if $\|Q(M^{(l)}, s^{(l)})\| < \varepsilon$ holds (lines 6-10). If it does not converge or converges too slowly, that is, $\|Q(M^{(l)}, s^{(l)})\|/\|Q(M^{(l-1)}, s^{(l-1)})\| \geq \Psi$ holds for a positive number Ψ, the penalty factor σ is updated to $\eta\sigma$ for $\eta > 1$ to speed up the convergence process (lines 11-13). Accordingly, the Lagrange multiplier for the next iteration is updated to $v_j^{l+1} = \max(0, v_j^{(l)} - \sigma g_j(M^{(l)}, s^{(l)}))(j = 1,2)$ (lines 14-15), and the procedure moves to the next iteration. Once the algorithm converges, the optimum of ω, M, and s are obtained, and the optimal $Profit$ of the cloud service provider can be calculated by using Eq. (25) (line 7). Line 17 returns the optimal service price, multiserver configurations, and $Profit$ of the cloud service provider.

Algorithm 1. Iteratively Solve the Augmented Lagrange Function

```
Input: Electricity price $C^\tau$ during sales period $\tau$, rent $\delta$, user
    requests arrival rate $\lambda_u$;
Output: The optimal service price $\omega$, number of servers $M$,
    server speed $s$, and $Profit$;
1  Formulate the optimization problem into the form in
   Equation (26);
2  Set parameters $\alpha$, $\beta$, $\gamma$, $\varepsilon$, $\eta$, $\Psi$, and $L$;
3  Initialize $M^{(0)}$, $s^{(0)}$, $v^{(1)}$, and $l = 1$;
4  while $l < L$ do
5      $[\omega^{(l)}, M^{(l)}, s^{(l)}] = \mathbf{ALF}-\mathbf{Solver}(\phi(\omega, M^{(l-1)}, s^{(l-1)}, v^{(l)}, \sigma))$;
       // exit when $\{v^{(l)}\}$ converges;
6      if $\|Q(M^{(l)}, s^{(l)})\| < \varepsilon$ then
7          Calculate the $Profit$ using the Equation (25);
           // record the optimal solution;
8          $Result = [Profit, \omega^{(l)}, M^{(l)}, s^{(l)}]$;
9          break;
10     end
       // otherwise, increase penalty factor $\sigma$;
11     else if $\|Q(M^{(l)}, s^{(l)})\|/\|Q(M^{(l-1)}, s^{(l-1)})\| \geq \Psi$ then
12         $\sigma = \eta\sigma$;
13     end
       // update the multiplier vector $\mathbf{v}$;
14     $v_j^{l+1} = \max(0, v_j^l - \sigma g_j(M^{(l)}, s^{(l)}))(j = 1,2)$;
15     $l = l + 1$;
16 end
17 return $Result$;
   // solve the Lagrange function in (36);
18 $\mathbf{ALF}-\mathbf{Solver}(\phi(\omega, M^{(l-1)}, s^{(l-1)}, v^{(l)}, \sigma))$
19 Compute partial derivatives of $\phi$ w.r.t. $\omega, M$, and $s$ as $\partial\phi(\omega,$
   $M^{(l-1)}, s^{(l-1)}, v^{(l)}, \sigma)/\partial(\omega, M, s)$;
20 Calculate $\omega$, $M$, and $s$ based on a system of equations of $\frac{\partial\phi}{\partial\omega}$,
   $\frac{\partial\phi}{\partial M}$, and $\frac{\partial\phi}{\partial s}$;
21 return $[\omega, M, s]$;
```

4.3 Design a Dynamic Closed Loop Control Scheme

The solution to the profit maximization problem described above focuses on the interaction between users and the cloud service provider. However, the impact of dynamic cloud computing environment such as fluctuating electricity bill and rental fees on profit maximization mechanism is not investigated. On one hand, the variation of electricity bill or rental fees has a direct impact on the expenditure of the cloud service provider. On the other hand, the variation of electricity bill or rental fees has an indirect influence on user perceived value which affects the user demand of the cloud service, and ultimately impacts the revenue of the cloud

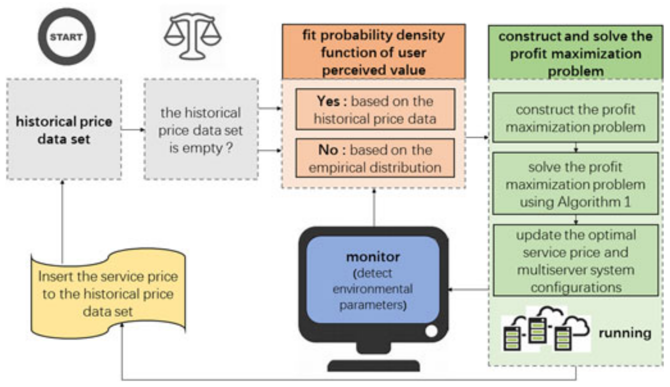

START
historical price data set
the historical price data set is empty ?
fit probability density function of user perceived value
Yes : based on the historical price data
No : based on the empirical distribution
construct and solve the profit maximization problem
construct the profit maximization problem
solve the profit maximization problem using Algorithm 1
update the optimal service price and multiserver system configurations
running
monitor (detect environmental parameters)
Insert the service price to the historical price data set

Fig. 2. Overview of closed loop control scheme.

service provider. Thus, it is necessary to design a scheme to adjust service price and multiserver configurations according to the dynamics of the cloud computing environment.

In this section, a closed loop control scheme is designed to dynamically update the optimal service price and multiserver configurations. As illustrated in Fig. 2, the runtime control scheme monitors the dynamic cloud computing environment. Once the electricity bill or rental fees changes, the proposed control scheme first fits a new probability distribution function of user perceived value using kernel density estimation based on the historical price data set Ω. Subsequently, it reconstructs and resolves the profit maximization problem based on the new probability distribution and the variation of electricity bill or rental fees.

The kernel density estimation technique adopted in the proposed control scheme is a stochastic non-parametric way to estimate the probability density function of a random variable [26]. It is a fundamental data smoothing technique where inferences about the population are made based on a finite data sample. Given a univariate independent and identically distributed sample drawn from some distribution with an unknown density function, the technique can be used to estimate the shape of the density function.

The details of the proposed runtime control scheme are described in Algorithm 2. Inputs to the algorithm are the historical price data set Ω and the output of system monitor. The closed loop control scheme works as follows. It monitors whether parameters of the cloud computing environment change at all times (line 2). If no change, the system will run with the current multiserver configurations (lines 3-5). Otherwise, it updates and solves the profit maximization problem (lines 6-15). Lines 7-8 fit the probability density function (pdf) of user perceived value using MATLAB function **ksdensity**$(\cdot)$ based on historical price data set Ω. Line 9 updates the profit maximization problem according to the change of the cloud computing environment (i.e., electricity bill or rental fees). Line 10 solves the profit maximization problem using algorithm 1. The algorithm updates the optimal cloud service price and multiserver configurations in line 11, and calculates the profit of the cloud service provider using Eq. (25) in line 12. Finally, it inserts the service price ω into the historical price data set Ω in line 13. Line 14 returns the optimal cloud service price, multiserver configurations, and $Profit$ of the cloud service provider.

The MATLAB function **ksdensity**$(\cdot)$ is used to fit the probability density function of user perceived value based on historical price data by using kernel smoothing density estimation. Line 18 first calculates the number of samples in the historical price data set Ω. The kernel bandwidth h is a free parameter that exhibits a strong influence on the resulting estimate [26]. Here, Gaussian basis functions are used to approximate univariate data. Thus, the optimal choice for kernel bandwidth h is calculated as line 19, which minimizes the mean integrated squared error used in density estimation [43]. **std**(Ω) computes the standard deviation of the samples in Ω. Line 20 uses operator $@(x)$ to define the function handle g, which represents a normal probability density function. **exp**(x) and **sqrt**(x) represent exponential function and square root function, respectively. Based on normal probability density function g and bandwidth h, line 21 computes the kernel density, that is probability density function by defining the function handle $ksden$. $mean(x)$ is used to compute the average of the array. Line 22 returns the final fitted probability density function.

Algorithm 2. Dynamic Closed Loop Control Scheme

```
    Input: The historical price data set $\Omega$, the output of system
        monitor;
    Output: The optimal service price $\omega$, number of servers $M$,
        server speed $s$, and $Profit$;
 1  while true do
 2    Monitor if parameters of cloud computing environment
      change;
 3    if no change then
 4      continue;
 5    end
 6    else
 7      $\Omega$ = historical price data set;
        // fit pdf using $ksdensity(\Omega)$;
 8      $f_X(\omega) \leftarrow$ ksdensity($\Omega$);
 9      Update profit maximization problem given in (26);
10      Solve profit maximization problem using Algorithm 1;
11      Update the optimal service price $\omega$, number of servers
        $M$, and server speed $s$;
12      Calculate the $Profit$ using Equation (25);
13      Insert price $\omega$ into historical price data set $\Omega$;
14      return [$\omega, M, s, Profit$];
15    end
16  end
    // fit pdf of user perceived value using MATLAB
    function $ksdensity(\Omega)$;
17  ksdensity($\Omega$);
    // get the sample number of $\Omega$;
18  $n$ = length($\Omega$);
    // set the optimal bandwidth $h$;
19  $h$ = std($\Omega$) $* (4/3/n)\hat{}(1/5)$;
    // obtain the normal pdf;
20  $g = @(x)(\mathbf{exp}(-.5 * x.^2)/\mathbf{sqrt}(2 * pi))$;
    // compute kernel density with $g$ and $h$;
21  $ksden = @(x)\mathbf{mean}(g((x - \Omega)/h)/h)$;
22  return $ksden$;
```

5 Simulation-Based Evaluation

Extensive simulation experiments have been conducted to validate the effectiveness of the proposed scheme. We first describe simulation settings in detail, and then verify the effectiveness of the proposed user perceived value-based

TABLE 1
Experimental Parameters Table

Parameter	Definition	Value
T	sales period	30 d
τ	time slot	1 h
N	number of time slot	720
D_{max}	maximum value of service deferment	24
ψ_1	sensitivity factor of users of type 1	∞
ψ_2	sensitivity factor of users of type 2	0.1
ψ_3	sensitivity factor of users of type 3	0.11

dynamic pricing model, followed by the validation of the optimal pricing and multiserver configurations and a comparison study with benchmarking schemes in terms of the profit of the cloud service provider.

5.1 Simulation Settings

The simulation experiments are conducted on a machine equipped with 2.56 GHz Intel i7 quad-core processor and 8 GB DDR4 memory, and running a Windows version of Matlab_x64. For the sake of a fair comparison, three types of users used in [35] are also adopted in our simulation experiments. Users of type 1 are delay-sensitive while users of type 2 and 3 are delay-insensitive to the deferment of the service requests. Data of type 1 are extracted from Youtube U.S. traffic from January 1, 2014 to January 31, 2014 [44]. Data of type 2 and 3 are extracted from GMaps and GMail U.S. traffic from January 1, 2014 to January 31, 2014 [44], respectively. The one day ahead real-time pricing data released by Ameren Illinois Power Corporation at January 2014 are taken as the price input in the experiment [45]. We also assume that user perceived value X obeys the following normal distribution, $X \sim N(0, 0.22)$[6], [22]. In addition, the value of other parameters used in our simulation experiment are shown in Table 1.

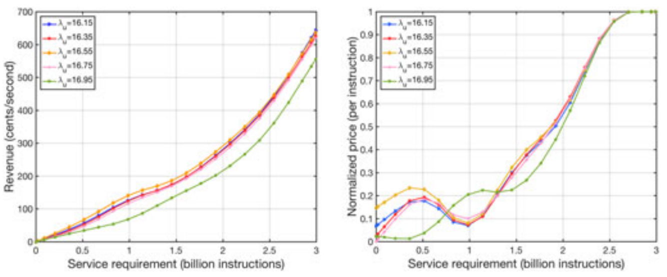
5.2 Verify User Perceived Value-Based Dynamic Pricing Model

This section verifies the proposed user perceived value-based dynamic pricing model from the perspective of supply and demand law.

5.2.1 Revenue versus Service Requirement

We first analyze the relationship between the service requirement in terms of the number of instructions, which is denoted by r, and the revenue of the cloud service provider. In addition to the parameters given in Table 1, we set the average service requirement denoted by $\overline{r}$ to 1 billion instructions. The number of servers M is initialized to 7, the base speed s_0 and speed s of servers are both initialized to 1 billion instructions per second, and the static power consumption P_{sta} is set to 2W. The parameters of dynamic power consumption are assumed to be $\gamma = 2.0$ and $\xi = 9.4192$, and parameters of Gamma distribution are assumed to be $\alpha = 2.0$ and $\beta = 1.5$ [5].

Fig. 3a shows the relationship between service requirements and the revenue of the cloud service provider when service request arrival rate λ_u is 16.15, 16.35, 16.55, 16.75, and 16.95 billions instructions per second, respectively. It can be seen from Fig. 3a that the revenue increases as



(a) Revenue vs. service requirement. (b) Normalized price vs. service requirement.

Fig. 3. Relationship between service requirement and revenue/normalized price.

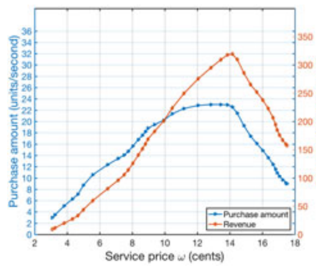
service requirements increase. This indicates that the usage of cloud services and the revenue obtained are positively correlated under the user perceived value-based pricing model. In addition, as shown in the figure, the revenue decreases as λ_u increases. This is because with the increase of λ_u, servers can not process service requests in time, leading to a higher response time and lower quality of service. Low quality of service will result in a smaller number of users to purchase cloud services, thus, the revenue of the cloud service provider decreases accordingly.

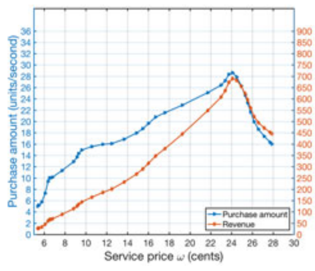
Fig. 3b shows the relationship between service requirements and the normalized service price when service request arrival rate λ_u is 16.15, 16.35, 16.55, 16.75, and 16.95 billions instructions per second, respectively. From the figure, we can see that when service requirement $r < 1.4$ billions, the normalized price fluctuates with the increase of the service requirement. When service requirement $1.4 < r < 2.6$ billions, the normalized price increases with the increase of the service requirement. When service requirement $r > 2.6$ billions, the normalized price eventually converges to a stable value with the increase of the service requirement.

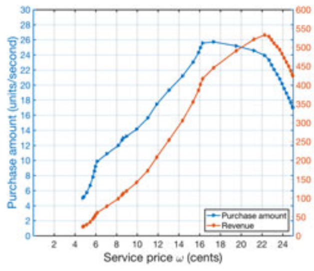
5.2.2 Purchase Amount and Revenue versus Service Price

Figs. 4a, 4b, 4c, and 4d demonstrate how the relationship among the cloud service purchase amount, revenue, and the price of cloud service changes when service request arrival rate λ_u is 16.15, 16.55, 16.75, and 16.95 billions instructions per second, respectively. As we can see from these figures, before the service price reaches user perceived value of the service, the purchase amount of the cloud service increases with the increases of the price. Once the price exceeds user perceived value of the service, the purchase amount declines sharply. This observation is consistent with real market situation, that is, users are willing to accept a price and purchase when the price is lower than their perceived value. However, the user's purchase intention will decline sharply when the price is beyond user perceived value.

It also can be seen from Figs. 4a, 4b, 4c, and 4d that the point where purchase amount is maximum is not necessarily the point where the revenue is maximum. That is, the revenue for the scenario of the low price and high purchase amount is not necessarily higher than the revenue for the scenario of the high price and low purchase amount. When service request arrival rate $\lambda_u = 16.55$, the cloud service provider can get the maximum revenue.


(a) $\lambda_u = 16.15$.


(b) $\lambda_u = 16.55$.


(c) $\lambda_u = 16.75$.

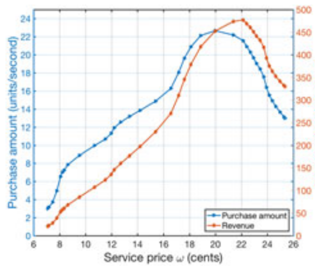
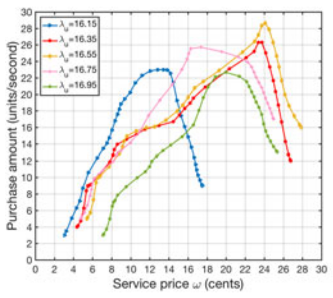

(d) $\lambda_u = 16.95$.

Fig. 4. Purchase amount and revenue versus service price.

5.2.3 Purchase Amount and Revenue versus Service request arrival rate

Figs. 5a and 5b demonstrate that how the cloud service purchase amount and revenue change when service request arrival rate λ_u is 16.15, 16.35, 16.55, 16.75, and 16.95 billions instructions per second, respectively. From Fig. 5a, we can see that the optimal prices for the maximum service purchase are different under diverse service request arrival rate λ_u. For the case where $\lambda_u = 16.55$ billions instructions per second, the service purchase amount and the service price ω reach the maximum value at the same time when compared to cases of different service request arrival rates. Meanwhile, the maximum purchase amount at $\lambda_u = 16.35$ is approximately the same as the maximum purchase amount at $\lambda_u = 16.75$. This situation holds for the case where $\lambda_u = 16.15$ and $\lambda_u = 16.95$. This is because with the increase of λ_u, limited number of servers can not process arrived service requests in time, leading to a higher response time, lower quality of service, and thus a lower maximum purchase amount of cloud services.

The revenue in Fig. 5b is obtained by multiplying the purchase amount and service price in Fig. 5a. Fig. 5b shows that the optimal prices for the maximum revenue are different under various service request arrival rate λ_u. From this figure, we observe that with the increase of λ_u, the maximum revenue at different λ_u increases first and then decreases. The cloud service provider obtains the maximum revenue when $\lambda_u = 16.55$ billions instructions per second. Similarly, this is because with the increase of λ_u, limited number of servers can not process arrived service requests in time, leading to a lower maximum purchase amount of services, and thus a lower revenue. Based on above experimental results, our user perceived value-based pricing model conforms to the supply and demand law in market.


(a) Purchase amount vs. service request arrival rate.

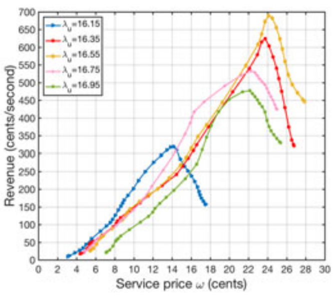

(b) Revenue vs. service request arrival rate.

Fig. 5. Purchase amount and revenue versus service request arrival rate.

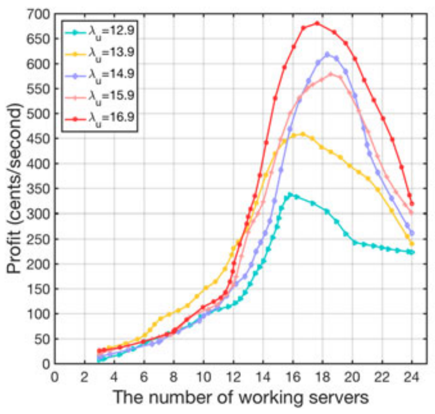
5.3 Validate Multiserver Configurations for Profit Maximization

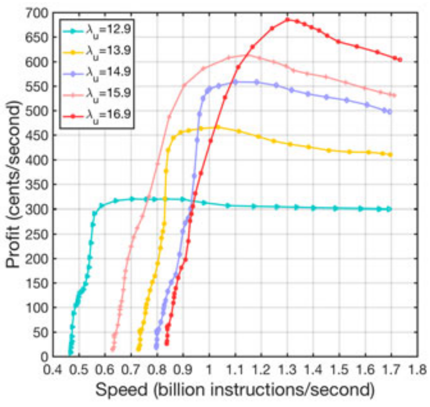
We set the response time constraint for user requests, denoted by b_1, to 0.33 seconds and the power consumption of the server system, denoted by b_2, to 10^6W. The rental cost denoted by δ is set to 1.5 cents per second [6].

Fig. 6a shows the relationship between profit and the number of working servers. It can be seen from the figure that when service request arrival rate λ_u = 12.9, 13.9, 14.9, 15.9, and 16.9 billions instructions per second, the optimal number of servers denoted by M is 16, 17, 19, 18, and 17, respectively. It is clear that when M is small, the utilization of working servers is approaching 1, leading to a long response time for user requests and low quality of service accordingly, and in turn a low profit under the user perceived value-based dynamic pricing model. As M increases, the number of user requests in the waiting queue decreases quickly, the user requests do not have to wait too long, and thus the profit increases under the user perceived value-based dynamic pricing model. However, as M continues increasing, the profit does not increase. This is because the increase in the number of servers leads to an increase in the maintenance cost of working servers including electricity and rental cost.

Fig. 6b shows the relationship between profit and the server speed s. We notice from the figure that in order to maximize the profit, the optimal speed s is set to 0.7642, 0.9435, 1.1044, 1.1293, and 1.2838 billions instructions per second when the service request arrival rate λ_u = 12.9, 13.9, 14.9, 15.9, and 16.9 billions instructions per second, respectively. It is clear that when the server speed s is low, the utilization of servers is approaching 1, leading to a long response time for user requests and low quality of service accordingly, and in turn a low profit under the user perceived value-based pricing model. When the server speed s is high, service requests are more likely to be executed on time, leading to an increase in the profit under the user perceived value-based pricing model. However, with the continued increase in s, the profit does not increase as expected. This is because the increase in s leads to an increase in the cost of operating a multiserver system.

Fig. 6c gives the optimal M and s of servers that maximize the profit when λ_u = 16.9 billions instructions per second. It can be seen that the maximal profit is obtained when s and M is set to 1.4351 billions instructions per second and 17, respectively. That is, 687.9 cents of profit is obtained when 17 servers are open and each server runs at 1.4351 billions instructions per second.


(a) Profit vs. number of severs (M).


(b) Profit vs. server speed (s).

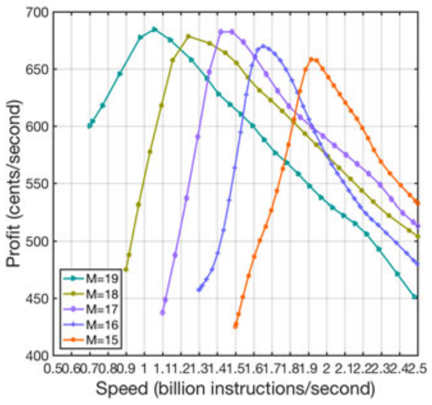
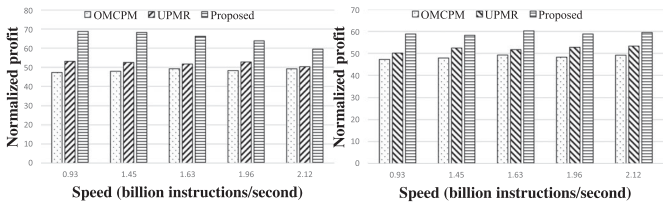

(c) Optimal server configurations.

Fig. 6. Validate server configurations for profit maximization.

5.4 Compare the Maximal Profit with Benchmarking Pricing Strategies

The proposed user perceived value-based profit maximization scheme is compared with two benchmarking methods OMCPM [5] and UPMR [35]. OMCPM [5] is an efficient pricing model that takes such factors into considerations as service-level agreement and customer satisfaction. It derives an optimal server configuration and service price for profit maximization. UPMR [35] is a usage based pricing model used by today's major cloud service providers. The UPMR model rewards users proportionally based on the time length that users set as deadlines for completing their service requests. Compared with OMCPM and UPMR, our pricing method is based on user perceived value that reflects users willingness to purchase cloud services.

We compare the maximal profit generated by proposed pricing model with that generated by the two benchmarking pricing models under the same experimental settings. Two comparison experiments are conducted. In the first experiment, user service request arrival rate λ_u is set to 16.9 billions instructions per second and the number of working servers M is set to 17. In the second experiment, λ_u is set to 12.55 billions instructions per second and M is set to 18. It is clear from Fig. 7 that our proposed dynamic pricing model is superior to the two benchmarking models. For instance, the proposed pricing model can obtain up to 21.55 cents per second more (31.32 percent) as compared to OMCPM method, and 15.66 cents per second more (22.76 percent) as compared to UPMR when λ_u = 16.9 billions instructions per second, M = 17, and s = 0.93 billion instructions per second. Thus, the pricing strategy based on user perceived value can better reflect the market demand and the cloud service provider can obtain higher profit.


(a) λ_u=16.9, M=17.

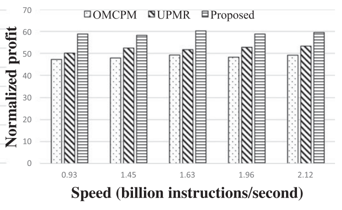

(b) λ_u=12.55, M=18.

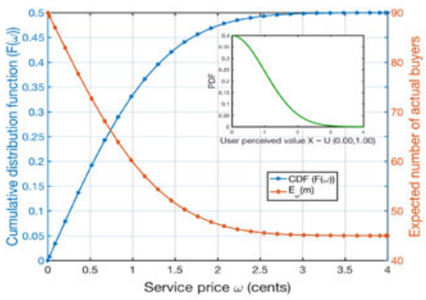
Fig. 7. Compare the maximal profit with two benchmarking pricing models.

We further verify how the expected number of actual buyers ($E_\omega(m)$) and the corresponding revenue change when user perceived value obeys normal distributions with different parameters. Figs. 8a, 8b, 8c, 8d, 8e, and 8f show the expected number of actual buyers ($E_\omega(m)$) under different expectations μ and variances σ^2 of user perceived value in our proposed dynamic pricing model. From Figs. 8a, 8b, and 8c, we can see that under different expectations μ, as μ increases, the cloud service provider needs to increase the service price ω to obtain the same amount of purchases. From Figs. 8d, 8e, and 8f, we can see that under different variances σ^2, when service price ω is less than μ, as σ^2 increases, the cloud service provider needs to decrease the service price ω to obtain the same amount of purchases. However, when service price ω is greater than μ, as σ^2 increases, the cloud service provider needs to increase the service price ω to obtain the same amount of purchases. This is because the larger the σ^2, the more dispersed the perceived value's distribution. Thus, in the case of the same service price ω, the purchase amount changes accordingly.

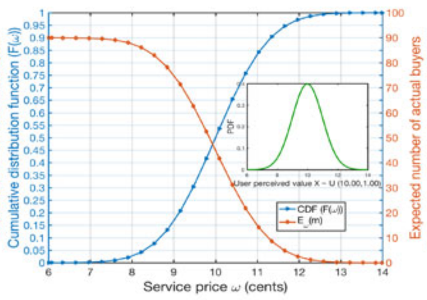
Figs. 9a, 9b, 9c, 9d, 9e, and 9f show the revenue under different expectations μ and variances σ^2 of user perceived value in our proposed dynamic pricing model. From Figs. 9a, 9b, and 9c, we can find that under the same purchase amount, the cloud service provider needs to increase expectation μ of normal distribution, that is, users' perceived value of services, to achieve higher revenue. From Figs. 9d, 9e, and 9f, we can see that under the same purchase amount, the cloud service provider needs to decrease variance σ^2 of normal distribution to achieve higher revenue. In general, to obtain the higher revenue, the cloud service provider needs to carry out market strategies to improve perceived value of service in users' mind. This is because under the same purchase amount, that is, under the same number of requests that the cloud service provider needs to process, the corresponding expenses are the same. Thus, it is reasonable to grow the profit of the cloud service provider by increasing the revenue.

6 Conclusion

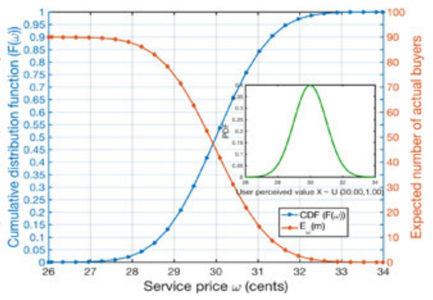
In this paper, we first propose a user perceived value-based dynamic profit maximization mechanism that takes into account the interaction between users and the cloud service



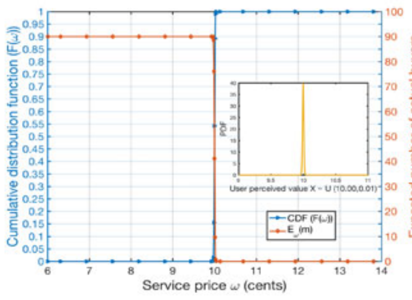
(a) $X \sim N(0,1)$.



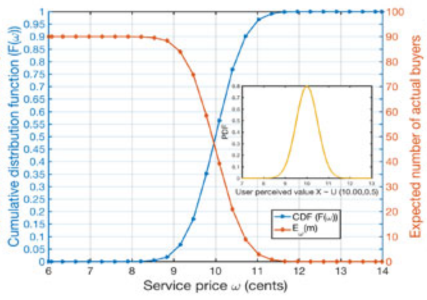
(b) $X \sim N(10,1)$.



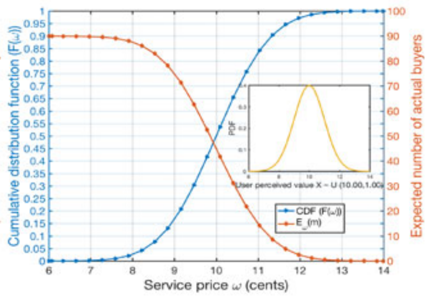
(c) $X \sim N(30,1)$.



(d) $X \sim N(10,0.01)$.

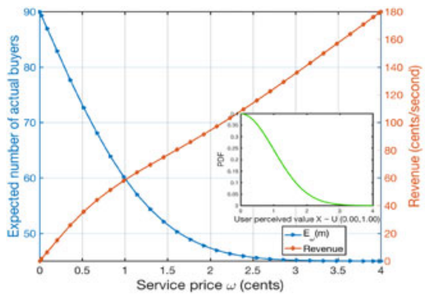


(e) $X \sim N(10,0.5)$.

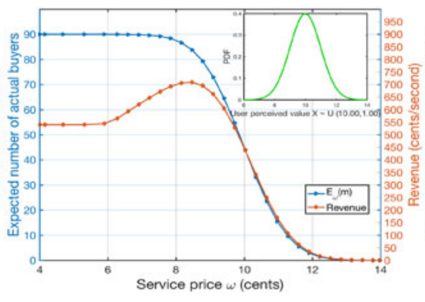


(f) $X \sim N(10,1.0)$.

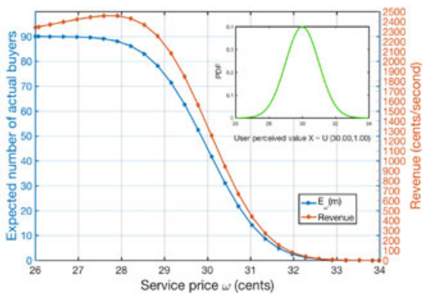
Fig. 8. Verify the change in the expected number of actual buyers when user perceived value obeys normal distributions with different parameters (i.e., μ and σ^2).



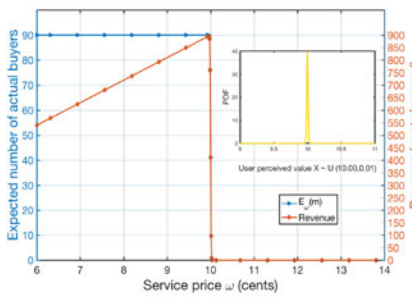
(a) $X \sim N(0,1)$.



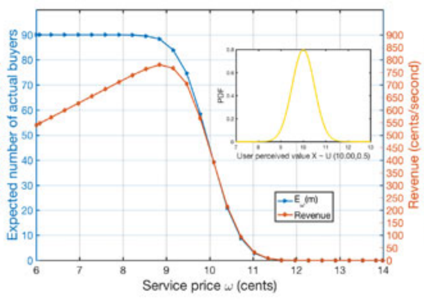
(b) $X \sim N(10,1)$.



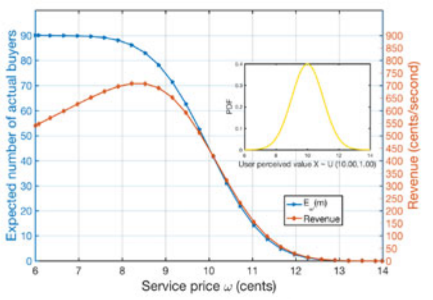
(c) $X \sim N(30,1)$.



(d) $X \sim N(10,0.01)$.



(e) $X \sim N(10,0.5)$.



(f) $X \sim N(10,1.0)$.

Fig. 9. Verify the change in the revenue when user perceived value obeys normal distributions with different parameters (i.e., μ and σ^2).

provider. Subsequently, the augmented Lagrange multiplier method is leveraged to solve the optimization problem to derive the optimal solution, including the service price, number of servers, and speed of servers. Finally, a dynamic closed loop control scheme is designed to update the service price and multiserver configurations using kernel density estimation method. Extensive simulation results demonstrate that our proposed profit maximization scheme follows the supply and demand law in market, and are able to obtain 31.32 and 22.76 percent more profit compared to the state of the art benchmarking methods OMCPM [5] and UPMR [35], respectively.

ACKNOWLEDGMENTS

This work was partially supported by Shanghai Municipal Natural Science Foundation (Grant No. 16ZR1409000) and Natural Science Foundation of China (Grant No. 61672230).

REFERENCES

[1] K. Hwang, J. Dongarra, and G. Fox, *Distributed and Cloud Computing*. Burlington, MA, USA: Morgan Kaufmann, 2012.
[2] L. Wang, Y. Ma, J. Yan, V. Chang, and A. Zomaya, "pipsCloud: High performance cloud computing for remote sensing big data management and processing," *Future Generation Comput. Syst.*, vol. 78, pp. 353–368, 2018.

[3] P. Mell and T. Grance, "The NIST definition of cloud computing, Nat. Inst. Standards Technol., Gaithersburg, MD, Tech. Rep. SP 800–145, 2011.

[4] P. Cong, L. Li, G. Shao, J. Zhou, M. Chen, K. Huang, and T. Wei, "User perceived value-aware cloud pricing for profit maximization of multiserver systems," in *Proc. IEEE Int. Conf. Parallel Distrib. Syst.*, 2017, pp. 537–544.

[5] J. Cao, K. Hwang, K. Li, and A. Zomaya, "Optimal multiserver configuration for profit maximization in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1087–1096, Jun. 2013.

[6] Y. Chun, "Optimal pricing and ordering policies for perishable commodities," *Eur. J. Oper. Res.*, vol. 144, pp. 68–82, 2003.

[7] C. Li, "Cloud computing system management under flat rate pricing," *J. Netw. Syst. Manage.*, vol. 19, no. 3, pp. 305–318, 2011.

[8] G. Kesidis, A. Das, and G. Veciana, "On flat-rate and usage-based pricing for tiered commodity internet services," in *Proc. Annu. Conf. Inf. Sci. Syst.*, 2008, pp. 304–308.

[9] Y. Lee, C. Wang, A. Zomaya, and B. Zhou, "Profit-driven scheduling for cloud services with data access awareness," *J. Parallel Distrib. Comput.*, vol. 72, no. 4, pp. 591–602, 2012.

[10] M. Macias and J. Guitart, "A genetic model for pricing in cloud computing markets," in *Proc. ACM Symp. Appl. Comput.*, 2011, pp. 113–118.

[11] Amazon EC2, 2018. [Online]. Available: http://aws.amazon.com

[12] Amazon EC2 spot instances, 2018. [Online]. Available: https://aws.amazon.com/cn/ec2/spot/pricing

[13] H. Xu and B. Li, "Dynamic cloud pricing for revenue maximization," *IEEE Trans. Cloud Comput.*, vol. 1, no. 2. pp. 158–171, Jul.–Dec. 2013.

[14] J. Zhao, H. Li, C. Wu, Z. Li, Z. Zhang, and F. Lau, "Dynamic pricing and profit maximization for the cloud with geo-distributed data centers," in *Proc. IEEE Conf. Comput. Commun.*, 2014, pp. 118–126.

[15] Service-level agreement, 2018. [Online]. Available: https://en.wikipedia.org/wiki/Service-level_agreement

[16] M. Ghamkhari and H. Mohsenian-Rad, "Energy and performance management of green data centers: A profit maximization approach," *IEEE Trans. Smart Grid*, vol. 4, no. 2, pp. 1017–1025, Jun. 2013.

[17] Y. Lee, C. Wang, A. Zomaya, and B. Zhou, "Profit-driven service request scheduling in clouds," in *Proc. Int. Conf. Cluster Cloud Grid Comput.*, 2010, pp. 15–24.

[18] D. Irwin, L. Grit, and J. Chase, "Balancing risk and reward in a market-based task service," in *Proc. Int. Conf. High Perform. Distrib. Comput.*, 2004, pp. 160–169.

[19] A. Khoshkbarforoushha, R. Ranjan, R. Gaire, E. Abbasnejad, L. Wang, and A. Zomaya, "Distribution based workload modelling of continuous queries in clouds," *IEEE Trans. Emerging Topics Comput.*, vol. 5, no. 1, pp. 120–133, Jan.–Mar. 2017.

[20] L. Wang, Y. Ma, A. Zomaya, R. Ranjan, and D. Chen, "A parallel file system with application-aware data layout policies for massive remote sensing image processing in digital earth," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 6, pp. 1497–1508, Jun. 2015.

[21] M. Leppaniemi, H. Karjaluoto, and H. Saarijarvi, "Customer perceived value, satisfaction, and loyalty: The role of willingness to share information, *The Int. Rev. Retail Distrib. Consum. Res.*, vol. 27, no. 2, pp. 164–188, 2017.

[22] Z. Yang and R. Peterson, "Customer perceived value, satisfaction and loyalty: The role of switching costs," *Psychology Marketing*, vol. 21, no. 10, pp. 799–822, 2004.

[23] S. Karlin and C. Carr, "Prices and optimal inventory policy," *Stud. Appl. Probability Manage. Sci.*, pp. 159–172, 1962.

[24] J. Roig, J. Garcia, M. Tena, and J. Monzonis, "Customer perceived value in banking services," *Int. J. Bank Marketing*," vol. 24, no. 5, pp. 266–283, 2006.

[25] Definition of perceived value, 2018. [Online]. Available: http://smallbusiness.chron.com/definition-perceived-value-23017.html

[26] Kernel density estimation, 2018. [Online]. Available: https://en.wikipedia.org/wiki/Kernel_density_estimation

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

[28] P. Pfeiffer, *Probability for Applications*. Berlin, Germany: Springer, 2012.

[29] K. Li, "Optimal load distribution for multiple heterogeneous blade servers in a cloud computing environment," *J. Grid Comput.*, vol. 11, no. 1, pp. 27–46, 2013.

[30] B. Chun and D. Culler, "User-centric performance analysis of market-based cluster batch schedulers," in *Proc. 2nd Int. Symp. Cluster Comput. Grid*, 2002, pp. 30–30.

[31] K. Li, "Optimal configuration of a multicore server processor for managing the power and performance tradeoff," *J. Supercomput.*, vol. 61, no. 1, pp. 189–214, 2012.

[32] L. Kleinrock, *Queueing Systems, Volume 1: Theory*. Hoboken, NJ, USA: Wiley, 1975.

[33] J. Zhou, J. Chen, K. Cao, T. Wei, and M. Chen, "Game theoretic energy allocation for renewable powered in-situ server systems," in *Proc. IEEE Int. Conf. Parallel Distrib. Syst.*, 2016, pp. 721–728.

[34] J. Little and S. Graves, Little's law, *Int. Series Operations Res. Manag. Sci.*, pp. 81–100, 2008.

[35] Y. Zhan, M. Ghamkhari, D. Xu, S. Ren, and H. Mohsenian-Rad, "Extending demand response to tenants in cloud data centers via non-intrusive workload flexibility pricing," *IEEE Trans. Smart Grid*, vol. 9, no. 4, pp. 3235–3246, Jul. 2018.

[36] Steepest descent method-wikipedia, 2018. [Online]. Available: https://en.wikipedia.org/wiki/Method_of_steepest_descent

[37] Newton's method-wikipedia, 2018. [Online]. Available: https://en.wikipedia.org/wiki/Newton%27s_method

[38] D. Bertsekas, "Multiplier methods: A survey," *Autom.*, vol. 12, pp. 133–145, 1976.

[39] W. Long, X. Liang, S. Cai, J. Jiao, and W. Zhang, "A modified augmented Lagrangian with improved grey wolf optimization to constrained optimization problems," *Neural Comput. Appl.*, vol. 28, pp. 421–438, 2017.

[40] Y. Zheng and Z. Meng, "A new augmented Lagrangian objective penalty function for constrained optimization problems," *Open J. Optimization*, vol. 6, pp. 39–46, 2017.

[41] N. Boland, J. Christiansen, B. Dandurandet, A. Eberhard, and F. Oliveira, "A parallelizable augmented Lagrangian method applied to large-scale non-convex-constrained optimization problems, pp. 1–34, in press, 2018.

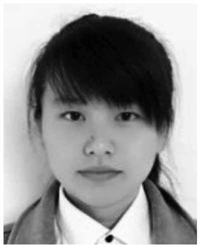
[42] Completing the square-wikipedia, 2018. [Online]. Available: https://en.wikipedia.org/wiki/Completing_the_square

[43] Mean integrated squared error, 2018. [Online]. Available: https://en.wikipedia.org/wiki/Mean_integrated_squared_error

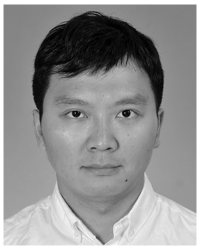
[44] Browse real-time traffic to google products and services, 2018. [Online]. Available: http://www.google.com/transparencyreport/traffic/explorer

[45] Real time prices-ameren, 2018. [Online]. Available: https://www.ameren.com/RetailEnergy/RealTimePrices

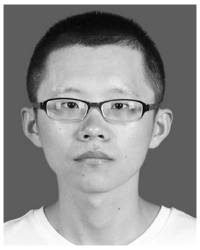
Peijin Cong received the BS degree from the Department of Computer Science and Technology, East China Normal University, Shanghai, China, in 2016. She is currently working toward the master's degree in the Department of Computer Science and Technology, East China Normal University, Shanghai, China. Her current research interests include power management in mobile devices and cloud computing.



Liying Li received the BS degree from the Department of Computer Science and Technology, East China Normal University, Shanghai, China, in 2017. She is currently working toward the master's degree in the Department of Computer Science and Technology, East China Normal University, Shanghai, China. Her current research interests include cyber physical systems and IoT resource management.



Junlong Zhou received the PhD degree in computer science from East China Normal University, in 2017. He is currently an assistant professor with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing. His research interests include real-time embedded systems, cyber physical systems, and cloud computing. He has been an associate editor for the *Journal of Circuits, Systems, and Computers* since 2017. He is a member of the IEEE.



Kun Cao is currently working toward the PhD degree in the Department of Computer Science and Technology, East China Normal University, Shanghai, China. His current research interests include high performance computing, multiprocessor systems-on-chip, and cyber physical systems.

Tongquan Wei received the PhD degree in electrical engineering from Michigan Technological University, in 2009. He is currently an associate professor with the Department of Computer Science and Technology, East China Normal University. His research interests are in the areas of Internet of Things, cloud computing, edge computing, intelligent and CPS system design. He serves as a regional editor for the *Journal of Circuits, Systems, and Computers* since 2012. He is a member of the IEEE.

Mingsong Chen received the PhD degree in computer engineering from the University of Florida, Gainesville, in 2010. He is currently a full professor with the Department of Embedded Software and Systems, East China Normal University. His research interests include design automation of cyber-physical systems, formal verification techniques and mobile cloud computing. He is a member of the IEEE.

Shiyan Hu received the PhD degree in computer engineering from Texas A&M University, in 2008. He is an associate professor with Michigan Tech, and he was a visiting associate professor with Stanford University from 2015 to 2016. His research interests include cyber-physical systems (CPS), CPS security, data analytics, and computer-aided design of VLSI circuits, where he has published more than 100 refereed papers. His publications have been awarded several Best Paper Award and Best Paper Finalists, which include the 2018 IEEE Systems Journal Best Paper Award. He is the Chair for the *IEEE Technical Committee on Cyber-Physical Systems*. He is an associate editor for the *IEEE Transactions on Computer-Aided Design*, the *IEEE Transactions on Industrial Informatics*, the *IEEE Transactions on Circuits and Systems*, the *ACM Transactions on Design Automation for Electronic Systems*, and the *ACM Transactions on Cyber-Physical Systems*. He is a fellow of the IET.