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 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. Ghakhar 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 multi-server systems under service-level agreement constraint. Our scheme also includes a runtime control loop to adjust service price and multi-server 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% 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 subsections.

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 $X_i$ 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 $\tau$ denote the length of each time slot over the sales period $T$, and $N$ be the number of time slots $\tau$ 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 a sales period $T$. Let $\lambda$ denote the number of users arriving per unit time, respectively. $n$ is assumed to be constant, but varies from period to period.

Suppose that the cloud service provider will charge the service at the price of $\omega$ during the sales period $T$, and follows a discrete Poisson distribution

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

(1)

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

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

(2)

where the expectation and variance of $\lambda_u$ 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 $\omega$ 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 $m \leq n$ 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 $\omega$. $F(\omega)$ describes the probability that users do not want to pay $\omega$ to buy a cloud service. It is a non-decreasing function of $\omega$, and $0 \leq F(\omega) \leq 1$ and $\lim_{\omega \to \infty} F(\omega) = 1$ hold [28]. Let $P_u(m|n)$ 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 $\omega$. It follows a binomial distribution of probability, which is given by

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

(3)

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

$$P_\omega(m) = \int_{\lambda_u=0}^{\infty} \sum_{n=0}^{\infty} P_u(m|n) P(n|\lambda_u) g(\lambda_u) d\lambda_u$$

$$= \left\{ \binom{m+\alpha-1}{m} \left[ \frac{\beta T [1 - F(\omega)]}{1 + \beta T [1 - F(\omega)]} \right]^{m+\alpha} \frac{1}{\Gamma(\alpha+\beta)} \beta^\alpha m! \right\} \cdot \frac{1}{\Gamma(\alpha+\beta)} \beta^\alpha.$$

(4)

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

$$E_\omega(m) = \alpha \beta T (1 - F(\omega)),$$

(5)

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

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

(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 queueing 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 $\rho$ be the service rate of user service requests that arrive at the rate of $\lambda_u$. It is clear that $\rho$ 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 $\frac{1}{\bar{r}} = \frac{1}{r}$, 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 $\rho$ be server utilization, which is defined as the average percentage of time that a server is busy. It can be expressed as

$$\rho = \frac{\lambda_u}{M \bar{r}} = \frac{\lambda_u}{M \frac{1}{\bar{r}}} = \frac{\lambda_u \bar{r}}{M}.$$  

(7)
Let $P_k$ be the probability of $k$ service requests being waiting or processing in the M/M/M queueing system. Based on queuing theory [5, 32, 33], $P_k$ is given by

$$P_k = \begin{cases} P_0 \frac{(M \rho)^k}{k!}, & k \leq M \\ P_0 \frac{M^k}{M!}, & k \geq M \end{cases},$$

(8)

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

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

(9)

The probability that there are exactly $M$ 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 $M! \approx \sqrt{2\pi M (\frac{e}{M})^M}$, it can be rewritten as

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

(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

$$P_q = \sum_{k=q}^{\infty} P_k = \frac{P_M}{1 - \rho},$$

(11)

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

$$\lambda = \sum_{k=0}^{\infty} kP_k = M \rho + \frac{\rho}{1 - \rho} P_q.$$  

(12)

The average service response time $R$ 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 the sum of task execution time and waiting time, and can be derived by applying Little’s Law [34] as

$$R = \frac{\lambda}{\lambda_u} = \frac{P_q}{M (1 - \rho)} = \frac{P_M}{M (1 - \rho)^2}.$$  

(13)

The average service response time $R$ 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 $\delta$ 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

$$Rent = M \delta \cdot T.$$  

(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

$$P_{tot} = P_{sta} + P_{dyn},$$

(15)

where $P_{sta}$ is the static power consumption while $P_{dyn}$ is the dynamic power consumption. $P_{sta}$ is independent of switching activity and dominates 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 $\nu$ and processing speed $s$. In addition, the supply voltage is usually linearly proportional to the processing speed, i.e., $\nu \propto s$. The dynamic power consumption $P_{dyn}$ is then expressed as $\nu \xi \rho$, where $\xi$ is a processor dependent coefficient and $\rho$ is a constant that equals to $2\phi+1$ ($\phi > 0$). Based on the static and dynamic power consumption described above, Equation (16) is used to derive the total power consumption of a multiserver system, which is

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

(16)

where $M$ is the server number and $\rho$ 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

$$E_T = M((P_{dyn} - P_{sta}) \rho + P_{sta}) \cdot T.$$  

(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

$$C_T(E_T) = \begin{cases} k_1T, & \text{if } 0 \leq E_T \leq l_1^T \\ k_2T, & \text{if } E_T > l_1^T \end{cases}.$$  

(18)

where $k_1^T, k_2 > 0$ are differentiated price and $l_1^T$ 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

$$Bill = E_T \cdot C_T(E_T) = M((P_{dyn} - P_{sta}) \rho + P_{sta}) \cdot T \cdot C_T(E_T).$$  

(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 $\psi_i$, 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 $\psi_i$ 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 (\( \psi_i \)) 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 \( h_i \), to represent the reward the cloud service provider returns to users of type \( i \). The reward \( h_i \) is a function of the service deferment \( D_i \), and is given by \( h_i = \theta \log(1 + D_i) \), where \( \theta > 0 \) and \( 0 \leq D_i \leq D_{\max} \) hold. \( \theta \) 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 \( (h_i - L_i) \) 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

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

Substitute \( D_i \) back into reward function \( h_i \), then one has

\[
h_i = \theta \log(1 + \max(\min(\frac{\theta}{\psi_i} - 1, D_{\max}), 0)). \tag{21}
\]

Let \( \text{Reward} \) denote the total monetary reward returned to users from the cloud service provider over the sales period \( T \), then one has

\[
\text{Reward} = \sum_{N'=1}^{N} \sum_{i=1}^{I} (h_i - L_i) \lambda^i_{N'}[N'] \tau, \tag{22}
\]

where \( \lambda^i_{N'}[N'] \) 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 \( \psi_i \) 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 \( (\bar{R}) \), 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]

\[
u_i(r, \bar{R}) = \begin{cases} 0, & 0 \leq \bar{R} \leq \frac{s_0}{\psi_i} + D_i, \\ \omega, & \bar{R} > (1 + \frac{\omega}{\theta}) \frac{s_0}{\psi_i} + D_i, \\ \infty, & \bar{R} \leq (1 + \frac{\omega}{\theta}) \frac{s_0}{\psi_i} + D_i, \end{cases}
\]

where \( \omega \) is the degree of punishment, \( D_i \) denotes the service deferment of users of type \( i \), and \( \omega \) 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 Equation (23) are described below. For users of type \( i \), if the average response time satisfies \( 0 \leq \bar{R} \leq \frac{s_0}{\psi_i} + D_i \), 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 \( \frac{s_0}{\psi_i} + D_i < \bar{R} \leq (1 + \frac{\omega}{\theta}) \frac{s_0}{\psi_i} + D_i \), 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 \( \bar{R} \) increases. Finally, if the average response time satisfies \( \bar{R} > (1 + \frac{\omega}{\theta}) \frac{s_0}{\psi_i} + D_i \), 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 \( \text{Penalty} \) to denote the total compensation provided by the cloud service provider to users, then we have

\[
\text{Penalty} = \sum_{N'=1}^{N} \sum_{i=1}^{I} u_i(r, \bar{R}) \lambda^i_{N'}[N'] \tau, \tag{24}
\]

where \( \lambda^i_{N'}[N'] \tau \) is the average number of user requests of type \( i \) in the time slot \( \tau \), and \( u_i(r, \bar{R}) \) 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 \( \omega \cdot E_w(m) \), where \( \omega \) denotes the service price per user and \( E_w(m) \) indicates the expected number of actual buyers at price \( \omega \) 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,

\[
\text{Profit} = \text{Revenue} - \text{Reward} - \text{Penalty} - \text{Bill} - \text{Rent}, \tag{25}
\]

where \( \text{Revenue}, \text{Reward}, \text{Penalty}, \text{Bill}, \) and \( \text{Rent} \) are given in Equations (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

Maximize: $\text{Profit}$
Subject to: $\theta \geq 0$
$P_{\text{tot}} \leq b_1$
$R \leq b_2$
$0 \leq \phi_i[N'] - \lambda_i[N'] \tau, \quad \forall i \in I, N' \in N$
$\sum_{N'=1}^{N} (\lambda_i[N'] \tau - \phi_i[N']) \geq \sum_{N'=1}^{N} \lambda_i[N'] \tau, \quad \forall i \in I, N' \in N$

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

In the above formulation, the reward factor $\theta$ is nonnegative (Equation 27), the total energy consumption $P_{\text{tot}}$ of the multiserver system within the sales period $T$ can not exceed $b_1$ (Equation 28), and the average service response time $R$ can not exceed $b_2$ (Equation 29). Equation (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 $\tau$, where $\phi_i[N']$ and $\lambda_i[N']$ are the amount of delayed and total service requests in the $N'$th time slot, respectively. Equation (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.

3.2 Overview of the Proposed Approach

The optimization problem given in Equation (26) tries to maximize the $\text{Profit}$ of the cloud service provider under the constraints mentioned above. Figure 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 multiserver 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 Equation (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 $\text{Bill}$ given in Equation (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 $\text{Bill}$ for $T$ is fixed and can be expressed as $\text{Bill} = b_3 P_{\text{tot}}$, where $P_{\text{tot}}$ given in Equation (16) is the total power consumed by the multiserver system and $b_3$ is a
constant coefficient. The profit optimization problem given in Equation (26) can then be re-written as
\[
\begin{align*}
O(\omega, M, s) &= \omega E_c[m] - b_1 P_{tot} - \delta MT \\
&= \sum_{N'=1}^N \sum_{l'=1}^l (h_i - L_i) \lambda_i e^{-h_i} \tau \\
g_1(M, s) &= b_2 - \pi \left(1 + \frac{P_{sta}}{\pi (1 + \rho)}\right) \geq 0 \\
g_2(M, s) &= b_1 - M (\xi - P_{sta}) \rho + P_{sta} \geq 0
\end{align*}
\]
(32)
where \(O(\omega, M, s)\) denotes the objective function of Profit given in Equation (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 Equation (32) with inequality constraints into an augmented Lagrange function. Let \(\lambda\) be the vector that converts the problem with inequality constraints to a problem with equality constraints, and \(v\) be the Lagrange multiplier vector, the augmented Lagrange function is thus given by
\[
\phi(\omega, M, s, y, v, \sigma) = O(\omega, M, s) - \sum_{j=1}^2 v_j (g_j(M, s) - y_j^2)
+ \sigma \sum_{j=1}^2 (g_j(M, s) - y_j^2)^2
\]
(33)
where the constant parameter \(\sigma\) denotes the penalty factor and \(\sigma > 0\) holds. The augmented Lagrange function given in Equation (33) can be converted into the form of
\[
\phi(\omega, M, s, y, v, \sigma) = O(\omega, M, s)
+ \sum_{j=1}^2 \left[ \frac{1}{\sigma} v_j^2 - \frac{1}{\alpha} (\sigma g_j(M, s) - v_j)^2 + \frac{v_j^2}{2\sigma} \right]
\]
(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
\[
y_j^2 = \frac{1}{\sigma} \max(0, \sigma g_j(M, s) - v_j), \quad j = 1, 2.
\]
(35)
Plugging \(y_j^2\) given in Equation (35) back into the original Equation (33), one obtains the augmented Lagrange function
\[
\phi(\omega, M, s, v, \sigma) = O(\omega, M, s)
+ \frac{1}{2\sigma} \sum_{j=1}^2 \left[ \max(0, v_j - \sigma g_j(M, s))^2 - v_j^2 \right].
\]
(36)
Through this quadratic relaxation of the original problem given in Equation (32), we can derive analytical 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 \(\omega\), number of servers \(M\), and speed of servers \(s\). Specifically, we seek to solve the augmented Lagrange function given in Equation (36) by first computing partial derivatives of Equation (32) with respect to \(\omega, M, and s\). Here, the details are omitted. We only show key steps for solving the partial derivatives of Equation (32) with respect to \(\omega, M, and s\).

**Calculate the partial derivative of Equation (32) with regard to \(\omega\)**: The partial derivative of \(E_c[m]\) with regard to \(\omega\) is \(\frac{\partial E_c[m]}{\partial \omega} = -\alpha \beta f(\omega)\), thus, the partial derivative of
\[
O(\omega, M, s)\text{ with regard to } \omega \text{ is } \frac{\partial O(\omega, M, s)}{\partial \omega} = \frac{\partial E_c[m]}{\partial \omega}.\]

**Calculate the partial derivative of Equation (32) with regard to \(M\)**: The partial derivative of \(g_1(M, s)\) with regard to \(M\) can be expressed as
\[
\frac{\partial g_1(M, s)}{\partial M} = \frac{\partial}{\partial M} \left[ \frac{1}{\pi M} \frac{1}{\sqrt{2\pi M(1 - \rho)(\epsilon/\rho)^M + 1}(1 - \rho)} \right]
+ \frac{\pi}{M^2 (1 - \rho)^2}.
\]
(37)
Let \(D_1 = \sqrt{2\pi M(1 - \rho)(\epsilon/\rho)^M + 1} = \sqrt{2\pi M(1 - \rho)L + 1}\), \(D_2 = 1 - \rho\), and \(L = (\epsilon/\rho)^M\), then Equation (37) becomes
\[
\frac{\partial g_1(M, s)}{\partial M} = \frac{\partial}{\partial M} \left[ \frac{1}{\pi D_1 D_2} \right] + \frac{\pi}{M^2 D_1^2 D_2^2}.
\]
(38)
The partial derivative of \(L, D_1,\) and \(D_2\) with regard to \(M\) are calculated as follows:
\[
\frac{\partial L}{\partial M} = L(\rho - \ln \rho - 1) + LM(1 - \frac{1}{\rho}) \frac{\partial \rho}{\partial M},
\]
\[
\frac{\partial D_1}{\partial M} = \sqrt{2\pi M(1 - \rho)(\epsilon/\rho)^M} + \sqrt{M(1 - \rho)} \frac{\partial L}{\partial M}
= \sqrt{2\pi M(1 + \rho)L - \sqrt{M(1 - \rho)} \ln \rho L},
\]
\[
\frac{\partial D_2}{\partial M} = - \frac{\partial \rho}{\partial M} = \frac{\rho}{M}.
\]
Substitute \(\frac{\partial L}{\partial M}, \frac{\partial D_1}{\partial M}\), and \(\frac{\partial D_2}{\partial M}\) back into the Equation (38), one has
\[
\frac{\partial g_1(M, s)}{\partial M} = \frac{\pi}{M D_1 D_2} \frac{\partial D_1}{\partial M} D_1 D_2 + \frac{\pi}{M^2 D_2} + \frac{1}{M}.
\]
(39)
The partial derivative of \(g_2(M, s)\) with regard to \(M\) can be easily calculated as
\[
\frac{\partial g_2(M, s)}{\partial M} = (\xi - P_{sta}) \cdot \rho + P_{sta}.
\]
**Calculate the partial derivative of Equation (32) with regard to \(s\)**: The partial derivative of \(L, D_1,\) and \(D_2\) with regard to \(s\) are calculated as follows:
\[
\frac{\partial L}{\partial s} = LM(1 - \frac{1}{\rho}) \frac{\partial \rho}{\partial s} = \frac{LM}{s} (1 - \rho),
\]
\[
\frac{\partial D_1}{\partial s} = \sqrt{2\pi M[(\frac{\partial D_1}{\partial \rho} L + (1 - \rho) \frac{\partial L}{\partial s})]} = \sqrt{2\pi M(\rho + (1 - \rho)^2) \frac{L}{s}},
\]
\[
\frac{\partial D_2}{\partial s} = - \frac{\partial \rho}{\partial s} = \frac{\rho}{s}.
\]
The partial derivatives of \(g_1(M, s)\) and \(g_2(M, s)\) with regard to \(s\) are hence computed as
\[
\frac{\partial g_1(M, s)}{\partial s} = \frac{\pi}{M} \frac{\partial P_M}{\partial s} (1 - \rho) = \frac{\pi}{M} \frac{\partial D_1}{\partial s} D_1 D_2 + \frac{\partial D_2}{\partial s} D_1.
\]
\[ \frac{\partial g_2(M, s)}{\partial s} = M \rho \gamma s^{\gamma - 1}. \]

Once we obtain the above partial derivatives of Equation (32) with regard to \( \omega, M \) and \( s \), we can compute and obtain the optimal solutions by letting these partial derivatives of Equation (32) with regard to \( \omega, M \), 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 maximization 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 \( M^{(0)}, s^{(0)} \), and \( v^{(0)} \) indicate the \( k^{th} \) iteration of \( M, s \), and \( v \) in the algorithm. Let \( \varepsilon, \eta, \) and \( \Psi \) 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 \( C^T \) during time slot \( \tau \), the rent \( \delta \), and user requests arrival rate \( \lambda_u \). The algorithm iteratively derives the optimal cloud service price \( \omega \) and multiserver configurations which includes the optimum 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 Equation (26), then sets parameters of \( \varepsilon, \eta, \Psi, \) 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 ALF-Solver\((\omega, M^{(l-1)}, s^{(l-1)}, v^{(l-1)}, \sigma)\), to obtain a local optimum of the \( \omega, M \), and \( s \) (line 5). The ALF-Solver\((\omega, M^{(l-1)}, s^{(l-1)}, v^{(l-1)}, \sigma)\) derives the local optimum by computing partial derivatives of \( \phi(\omega, M, s, v, \sigma) \) with regard to \( \omega, M, \) and \( s \) and solving a system of equations of \( \omega, M, \) and \( s \) (lines 18-21).

The algorithm finishes if the Lagrange multiplier vector \( v \) converges and approximates the optimum by an error of \( \varepsilon \). Let \( Q_j(M^{(0)}, s^{(0)}) = g_j(M^{(0)}, s^{(0)}) - y_j^2 \) for \( j = 1, 2 \) be the penalty item of the augmented Lagrange function given in Equation (33), then the Lagrange multiplier vector \( v \) converges if \( \|Q(M^{(0)}, s^{(0)})\| < \varepsilon \) holds (lines 6-10). If it does not converge or converges too slowly, that is, \( \|Q(M^{(0)}, s^{(0)})\|/\|Q(M^{(l-1)}, s^{(l-1)})\| \geq \Psi \) holds for a positive number \( \Psi \), the penalty factor \( \sigma \) 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 \( v_j^{l+1} = \max(0, v_j^{l-1} - \sigma g_j(M^{(0)}, s^{(0)})) \) (lines 14-15), and the procedure moves to the next iteration. Once the algorithm converges, the optimum of \( \omega, M, \) and \( s \) are obtained, and the optimal Profit of the cloud service provider can be calculated by using Equation (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 \( C^T \) 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^{(0)}, M^{(0)}, s^{(0)} = \text{ALF-Solver}(\omega, M^{(l-1)}, s^{(l-1)}, v^{(l-1)}, \sigma));  
6. if \( \|Q(M^{(0)}, s^{(0)})\| < \varepsilon \) then  
7. Calculate the Profit using the Equation (25);  
8. Result = Profit, \( \omega^{(0)}, M^{(0)}, s^{(0)} \);  
9. break;  
10. end  
11. else if \( \|Q(M^{(l)}, s^{(l)})\|/\|Q(M^{(l-1)}, s^{(l-1)})\| \geq \Psi \) then  
12. \( \sigma = \eta \sigma \);  
13. end  
14. \( v^{l+1} = \max(0, v_j^{l-1} - \sigma g_j(M^{(0)}, s^{(0)})) \) (\( j = 1, 2 \));  
15. \( l = l + 1 \);  
16. end  
17. return Result;  
18. ALF-Solver\((\omega, M^{(l-1)}, s^{(l-1)}, v^{(l-1)}, \sigma)\);  
19. Compute partial derivatives of \( \phi \) w.r.t. \( \omega, M, \) and \( s \) as \( \partial \phi(\omega, M^{(l-1)}, s^{(l-1)}, v^{(l-1)}, \sigma) / \partial (\omega, M, s) \);  
20. Calculate \( \omega, M, \) and \( s \) based on a system of equations of \( \partial_{\omega}, \partial_{M}, \) and \( \partial_{s} \);  
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 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 subsection, a closed loop control scheme is designed to dynamically update the optimal service price and multiserver configurations. As illustrated in Figure 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 \( \Omega \). 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 \( \Omega \) 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 \texttt{ksdensity}() based on historical price data set \( \Omega \). 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 Equation (25) in line 12. Finally, it inserts the service price into the historical price data set \( \Omega \) in line 13. Line 14 returns the optimal cloud service price, multiserver configurations, and Profit of the cloud service provider.

The MATLAB function \texttt{ksdensity}() 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 \( \Omega \). 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], \( \text{std}(\Omega) \) computes the standard deviation of the samples in \( \Omega \). Line 20 uses operator \( \hat{g}(x) \) to define the function handle \( g \), which represents a normal probability density function. \( \exp(x) \) and \( \text{sqrt}(x) \) 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 \texttt{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. **else**
6.   **end**
7.   \( \Omega = \) historical price data set;
8.   \( f(x) \leftarrow \texttt{ksdensity}(\Omega) \); // fit pdf using \texttt{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**

**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 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 2.56GHz Intel i7 quad-core processor and 8GB 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
5.2 Verify User Perceived Value-Based Dynamic Pricing Model

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

5.2.1 Revenue Vs. 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 \( T \) 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 \( P_{us} \) 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].

Figure 3(a) shows the relationship between service requirements and the revenue of the cloud service provider when service request arrival rate \( \lambda_u \) 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 request 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 Vs. Service Price

Figure 4(a)-4(d) demonstrate how the relationship among the cloud service purchase amount, revenue, and the price of cloud service changes when service request arrival rate \( \lambda_u \) 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 Figure 4(a)-4(d) 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.

5.2.3 Purchase Amount and Revenue Vs. Service request arrival rate

Figure 5(a) and 5(b) demonstrate how the cloud service purchase amount and revenue change when service request arrival rate \( \lambda_u \) is 16.15, 16.35, 16.55, 16.75, and 16.95 billions instructions per second, respectively. From Figure 5(a), we can see that the optimal prices for the maximum service purchase are different under diverse service request arrival rate \( \lambda_u \). For the case where \( \lambda_u = 16.55 \) billions instructions per second, the service purchase amount and

---

Table 1: Experimental parameters table.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T )</td>
<td>sales period</td>
<td>30 d</td>
</tr>
<tr>
<td>( \tau )</td>
<td>time slot</td>
<td>1 h</td>
</tr>
<tr>
<td>( N )</td>
<td>number of time slot</td>
<td>720</td>
</tr>
<tr>
<td>( D_{max} )</td>
<td>maximum value of service deferment</td>
<td>24</td>
</tr>
<tr>
<td>( \psi_1 )</td>
<td>sensitivity factor of users of type 1</td>
<td>( \infty )</td>
</tr>
<tr>
<td>( \psi_2 )</td>
<td>sensitivity factor of users of type 2</td>
<td>0.1</td>
</tr>
<tr>
<td>( \psi_3 )</td>
<td>sensitivity factor of users of type 3</td>
<td>0.11</td>
</tr>
</tbody>
</table>

---

Figure 3: Relationship between service requirement and revenue/normalized price.

(a) Revenue vs. service requirement (b) Normalized price vs. service requirement.
At the 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 \( \lambda_u \), 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.

From this figure, we observe that with the increase of \( \lambda_u \), 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. 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.

Figure 6(b) 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 \( \lambda_u = 12.9, 13.9, 14.9, 15.9, \) and 16.9 billions instructions per second. 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.

Figure 6(c) gives the optimal \( M \) and \( s \) of servers that maximize the profit when \( \lambda_u = 16.9 \) 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.
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 \( \lambda_u \) is set to 16.9 billions instructions per second and the number of working servers \( M \) is set to 17. In the second experiment, \( \lambda_u \) is set to 12.55 billions instructions per second and \( M \) is set to 18. It is clear from Figure 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%) as compared to OMCPM method, and 15.66 cents per second more (22.76%) as compared to UPMR when \( \lambda_u = 16.9 \) 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.

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. Figure 8(a)-8(f) show the expected number of actual buyers (\( E_\omega(m) \)) under different expectations \( \mu \) and variances \( \sigma^2 \) of user perceived value in our proposed dynamic pricing model. From Figure 8(a)-8(c), we can see that under different expectations \( \mu \), as \( \mu \) increases, the cloud service provider needs to increase the service price \( \omega \) to obtain the same amount of purchases. From Figure 8(d)-8(f), we can see that under different variances \( \sigma^2 \), when service price \( \omega \) is less than \( \mu \), as \( \sigma^2 \) increases, the cloud service provider needs to decrease the service price \( \omega \) to obtain the same amount of purchases. However, when service price \( \omega \) is greater than \( \mu \), as \( \sigma^2 \) increases, the cloud service provider needs to increase the service price \( \omega \) to obtain the same amount of purchases. This is because the larger the \( \sigma^2 \), the more dispersed the perceived value’s distribution. Thus, in the case of the same service price \( \omega \), the purchase amount changes accordingly.

Figure 9(a)-9(f) show the revenue under different expectations \( \mu \) and variances \( \sigma^2 \) of user perceived value in our proposed dynamic pricing model. From Figure 9(a)-9(c), we can find that under the same purchase amount, the cloud service provider needs to increase expectation \( \mu \) of normal distribution, that is, users’ perceived value of services, to achieve higher revenue. From Figure 9(d)-9(f), we can see that under the same purchase amount, the cloud service provider needs to decrease variance \( \sigma^2 \) 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 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% more profit compared to the state of the art benchmarking methods OMCPSM [5] and UPMR [35], respectively.

Figure 8: Verify the change in the expected number of actual buyers when user perceived value obeys normal distributions with different parameters (i.e., $\mu$ and $\sigma^2$).

REFERENCES

[21] M. Leppaniemi, H. Karitalo, and H. Saarijarvi, Customer perceived value, satisfaction, and loyalty: The role of willingness to
Figure 9: Verify the change in the revenue when user perceived value obeys normal distributions with different parameters (i.e., $\mu$ and $\sigma^2$).


Liying Li received her B.S. degree from the Department of Computer Science and Technology, East China Normal University, Shanghai, China, in 2017. She is currently pursuing the master degree with 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 his Ph.D. 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 pursuing his Ph.D. degree with 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 his Ph.D. degree in Electrical Engineering from Michigan Technological University in 2009. He is currently an Associate Professor in the Department of Computer Science and Technology at East China Normal University. His current research interests include Internet of Things, edge computing, cloud computing, and design automation of intelligent and CPS systems. He serves as a Regional Editor for Journal of Circuits, Systems, and Computers since 2012. He is a member of the IEEE.

Mingsong Chen received his Ph.D. 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 of 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 his Ph.D. in Computer Engineering from Texas A&M University in 2008. He is an Associate Professor at Michigan Tech, and he was a Visiting Associate Professor at 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. Prof. Hu is the Chair for IEEE Technical Committee on Cyber-Physical Systems. He is an Associate Editor for IEEE Transactions on Computer-Aided Design, IEEE Transactions on Industrial Informatics, IEEE Transactions on Circuits and Systems, ACM Transactions on Design Automation for Electronic Systems, and ACM Transactions on Cyber-Physical Systems. He is a Fellow of IET.