Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 6901084, 18 pages
https://doi.org/10.1155/2020/6901084

Research Article

Optimizing the Procurement of IaaS Reservation Contracts via Workload Predicting and Integer Programming

Huamin Zhu, Jun Luo, and Hongyao Deng

College of Big Data and Intelligent Engineering, Yangtze Normal University, Chongqing, China

Correspondence should be addressed to Huamin Zhu; zhuhuamin2001@163.com

Received 4 July 2020; Revised 16 August 2020; Accepted 17 August 2020; Published 6 November 2020

Academic Editor: Wei-Chiang Hong

Cloud-based web applications are proliferating fast. Owing to the elastic capacity and diverse pricing schemes, cloud Infrastructure-as-a-Service (IaaS) offers great opportunity for web application providers to optimize resource cost. However, such optimization activities are confronting the challenges posed by the uncertainty of future demand and the increasing reservation contracts. This work investigates the problem of how to minimize IaaS rental cost associated with hosting web applications, while meeting the demand in the future business cycle. First, an integer liner program model is developed to optimize reservation-contract procurement, in which reserved and on-demand resources are planned for multiple provisioning stages as well as a long-term plan, e.g., twelve stages in an annual plan. Then, a Long Short-Term Memory (LSTM) based algorithm is designed to predict the workload in the future business cycle. In addition, the approaches for determining virtual instance capacity and the baseline workload of planning time slot are also presented. Finally, the experimental prediction results show the LSTM-based algorithm gains an advantage over several popular models, such as the Holter–Winters, the Seasonal Autoregressive Integrated Moving Average (SARIMA), and the Support Vector Regression (SVR). The simulations of resource planning show that the provisioning scheme based on our reservation-optimization model obtains significant cost savings than other typical provisioning schemes, while satisfying the demands.

1. Introduction

Cloud computing is a large-scale distributed computing paradigm in which a pool of computing resources is available via the Internet. As the most widely applied service model in cloud computing, the IaaS liberates organizations from the expensive infrastructure investment with the virtually infinite resources and the elasticity. In this model, infrastructure resources such as computing, storages, and networks can be rented to customers in the form of virtual machine instances. Each instance belongs to a specific instance type specifying the hardware configuration (CPU cores and speed, memory, and I/O channels). The consumers can quickly deploy the packaged OS and application images to the leased IaaS instances and start them. Meanwhile, web application workload generally exhibits inherent seasonality, stochastic volatility, and aggregated volatility [1]. Therefore, web applications are well suitable for the deployment on the instances rented from IaaS providers, as it makes easy to quickly scale resources so as to deal with varying workload. For example, the 12306 e-ticket site in China is now very stable [2], but before being deployed to the cloud, it gets stuck or even crashed almost whenever the peak of visits appears.

IaaS providers usually offer customers two types of resource provisioning plans, namely, on-demand and reservation plans with different charging schemes. The on-demand plans charge customers on a pay-as-you-go basis and enable them to start or terminate instances at any moment according to needs without paying any penalty. However, comparing the unit price, the on-demand resources are often more expensive than the reserved ones. With the reservation plans, virtual instances are reserved in the form of long-term contracts. Through the use of reservation plans, customers can get significant price discount compared with on-demand plans and pay once for the contract duration (e.g., 1 month, 3 months, 6 months, 9 months, or 1 to 5 years at Aliyun [3], 6 to 36 months at Rackspace [4], and 1 year or 3 years at Amazon [5]). Taking an Aliyun's ecs.g5.large instance in the Region Qingdao of North China, for example, compared with the on-demand plan, the discount rates of monthly fees for 1 month, 3 months, 6 months, and 1 year reservation contracts are 60.9%, 64.8%, 66.7%, and 69.0%, respectively.

In fact, for the web applications with time-varying workload, using only reserved resources or on-demand resources is generally not the best choice. Imagine a web application with changing resources demand, as shown in the curve of Figure 1. If only reserved resources are planed, e.g., $\mathrm{N_H}$ instances are reserved, then lots of instances will not be efficiently utilized, resulting in significant waste of resources.. On the contrary, if only on-demand resources are used, high unit price of resources will lead to a large total cost. Apparently, the best decision is to reserve N_R (namely, a number between N_H and N_L) instances and then supplement several on-demand instances when needed. As such, an optimal total cost can be obtained, while meeting the workload demand.

Nowadays, more and more web applications are migrated to the cloud. Meanwhile, more and more IaaS reservation contracts are also offered by cloud providers. For web applications providers, it has become very necessary to optimize the provisioning of IaaS resources for saving cost. However, most of the existing approaches have employed the deterministic resources provisioning schemes [6–10]. In these studies, the uncertain nature of the user's demands is neglected by assuming the demand as a deterministic value. To address the demand uncertainties, in [11–14], some dynamic resource provisioning schemes are proposed. These schemes are more flexible and provision resources dynamically to meet fluctuating workload. However, these studies do not exploit the cost benefits of reservation contracts, resulting in failure to achieve economical solutions. Given the disadvantages of two categories of schemes above, several studies have employed the hybrid schemes to provision resources [15–18]. Although the decision making is more complex, the hybrid schemes take advantage of reserved and on-demand resources simultaneously so as to save cost, while better meeting varying demand. The hybrid provisioning scheme is generally carried out in two phases. Prior to the start of the workload cycle, the resource-reservation contract procurement is planned in advance based on an estimated or predicted workload. During the workload cycle, the previous obtained reservation plans are carried out successively and then the reserved resources are utilized, while additional on-demand resources may be provisioned whenever necessary.

For cloud-based web applications, we prefer the hybrid scheme and believe that an excellent provisioning scheme should use as many reserved resources as possible to satisfy long-term stable demands in the future and only use a small amount of on-demand resources to deal with sudden demands so as to minimize the total resources cost. However, as more and more IaaS reservation contracts are offered, for a long web application workload cycle, how to combine multiple reservation contracts as well as determine the numbers and start times of them so as to optimize the total cost? is the first major challenge for the IaaS resource decision-makers of web applications.

Besides, planning resource for the future business cycle of a web application requires an estimation/prediction of the future workloads. Some studies have employed the simulated workloads [6, 9, 11]; meanwhile, some studies directly take the workloads in the historical cycle as an estimation of the workloads in the future cycle [14, 15, 17]. But, the two approaches generally could not obtain a good accuracy. There are also some studies to develop the stochastic programming models for future workloads based on the historical workloads' summary [18–20] (e.g., the mean and standard deviation). However, such models are only applicable to stochastic workload series and cannot handle the workload series with the trend and seasonality. The most widely employed schemes are workload predictions. One group of prediction approaches for web workloads is statistical models such as Autoregression (AR), Autoregressive Moving Average (ARMA), Autoregressive Integrated Moving Average (ARIMA) [21], Exponential Smoothing (ES) [22], and Linear Regression (LR) [13] models. These statistical models are effective for the short-term prediction of stationary series, while their prediction accuracy for nonstationary series is very poor. This is because erratic fluctuations, which are typical for web workload series, are practically impossible to predict. This problem can be resolved by using machine-learning techniques such as the Support Vector Regression (SVR) [23–25] and the Deep Belief Networks (DBN) [26]. The advantages of these approaches are that they can learn from historical data (search connections among features) and build a prediction model for future workloads [27]. However, due to the lack of long-term memory ability, these models are still difficult to learn long-term inherent patterns of workload series. In view of the facts that web workload is often affected by many factors and the workload cycle is usually long, how to deal with the uncertainty of workload prediction in the future business cycle for web applications? is the second major challenge for the IaaS resource decision-makers of web applications.

The Recurrent Neural Network (RNN) is a novel neural network architecture specially designed for the sequence data and has been proven successful in time series prediction tasks [28]. However, a traditional RNN performs poorly at handling long-term dependencies, mainly due to the exploding and vanishing gradient problem [29, 30]. As a redesigned architecture of RNN, the LSTM network addresses the shortcomings by replacing the RNN cell with an LSTM cell in the hidden layer and thus has the ability of learning long-term dependencies [31]. Owing to the superior long-term memory ability, the LSTM exhibits excellent potential for predicting the long time series. There have been several good attempts on applying the LSTM to carry out the mid- and long-term prediction for time series, such as traffic flow [32], bank business [33], and earthquakes [34]. Under such background, we choose to employ the LSTM for

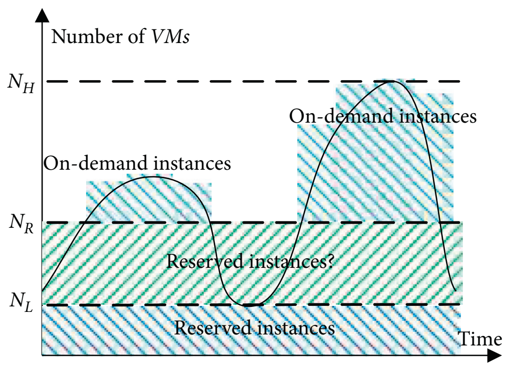

Figure 1: A workload example.

predicting the future cycle workload of web applications and evaluate its prediction performance by comparing with other popular approaches.

In addition, planning resource for the future business cycle of a web application depends not only on the future workload but also on the processing capacity of each IaaS instance. However, this value is not fixed and is closely related to the threshold of service response time. At the same time, for a long planning cycle, because it is hard to perform fairly fine granularity of prediction, the duration of planning time slot is not usually short, such as a day. Also, the predicted workload is generally the total or average number of requests in a time slot. If the predicted average values are directly used as the baseline workload for planning slot, it is clear that the underprovisioning will happen frequently. Therefore, how to determine the processing capacity of single instance and the baseline workload of planning time slot so that planned resources can better meet the demand? is the third major challenge for the IaaS resource decision-makers of web applications.

This work focuses on planning resource reservations prior to the beginning of the workload cycle, aimed at achieving the optimal plan of IaaS reservation contract procurement through the use of workload prediction. We studied the above several challenge problems in depth, and the major contributions are threefold.

(1) Based on the divisions of the provisioning cycle and the description of reservation contracts, an integer linear program model is developed to optimize reservation contract procurement.

(2) Given the inherent pattern of web cycle workload series, a Long Short-Term Memory (LSTM) network-based algorithm is designed to predict the cycle workload of web applications.

(3) The approach for determining instance capacity is presented by using an M/M/n queuing system. Time-slot baseline workload is also determined based on the average of historical peak workloads.

The remainder of this paper is organized as follows. Section 2 discusses the related work. Section 3 presents the problem domain and assumptions. Section 4 develops the reservation contract procurement-optimization model. In Section 5, the LSTM-based prediction algorithm is designed. In Section 6, the approaches for determining the instance capacity and time-slot baseline workload are introduced. Experimental settings and results are presented in Section 7, followed by conclusions and future work in Section 8.

2. Related Work

In the past years, the problem of cloud resource planning has attracted many researchers' attention to develop resource provisioning algorithms and techniques [6, 9, 35–37]. A deeply survey can separate the studies into three categories: deterministic resource provisioning, dynamic resource provisioning, and hybrid resource provisioning. In the following sections, the existing studies are discussed in these categories, and finally the prediction approaches for web application workload are also discussed.

2.1. Deterministic Resource Provisioning. Most of the studies model this problem as a single phase optimization algorithm that only considers resources with reserved contracts from IaaS providers. These studies neglect the uncertainty of users' demands and regard the demands as fixed values and then employ deterministic provisioning schemes to deal with future workload [8, 9]. Mireslami et al. [6] planned the number of service instances according to the instance's minimum service rate. Imai et al. [7] used an expensive overprovisioning scheme for the worst-case demand. Jiao et al. [38] designed a cost optimization model for online social network deployment in geo-distributed clouds. The work regarded the demand of each cloud as a deterministic value. Similarly, in [10], a multiobjective algorithm was developed to minimize total deployment cost and maximize service of quality (QoS) performance. Chen et al. [9] constructed a resource cost optimization model for periodical workflow applications based on fixed workload.

Deterministic resource provisioning is better suited for constant workload scenarios (e.g., batch processing tasks) rather than web applications with varying workload.

2.2. Dynamic Resource Provisioning. In order to deal with the uncertainty of users' demands, some studies employ elastic mechanisms to provision cloud resources. Zhao et al. [11] constructed a resource cost optimization model for computational and data intensive applications, which is performed periodically at hourly intervals. Antonescu et al. [13] dynamically adjusted resources to meet predicted short-term workload so as to minimize the cost, while avoiding the service level agreement (SLA) violations. Sniezynski et al. [14] used linear regression, neural networks, etc., to learn resource usage patterns from the historical records so as to predict and update resource capacity periodically.

Although these dynamic provisioning schemes better meet the varying demands, they result in considerable cost because of using only expensive on-demand resources.

2.3. Hybrid Resource Provisioning. The hybrid resource provisioning uses deterministic reserved resources to deal with long-term stable workload and uses dynamic on-demand resources to deal with short-term sudden workload. Stijven et al. [39] proposed a scheme to plan reserved resources based on short-term workload prediction but only one kind of contract could be used. Candeia et al. [15] designed the algorithms to select IaaS reservation markets and determine the numbers of instances as well as their lifespans, without considering multiple kinds of contracts simultaneously. Similarly, Chen et al. [17] also presented a hybrid short-term provisioning scheme that could only include one contract type. Mireslami et al. [18] proposed two-stage provisioning scheme for web applications. In the first stage, they decide which contract to purchase based on the minimum workload, and in the second stage, additional on-demand resources was provisioned dynamically. Their scheme is similar to this work but only one type of contracts is considered.

For all above studies, only one reservation-contract type can be included, and the reserved resources are constant. In this work, the problem of reservation-resource planning for the entire workload cycle is investigated. A workload cycle is divided into multiple provisioning stages uniformly so that multiple reservation contract types with different durations can be combined to provision resources so as to obtain a minimum total cost.

2.4. Prediction of Web Application Workload. Calheiros et al. [21] presented the realization of a workload prediction module for cloud-based applications based on the ARIMA. However, the ARIMA cannot deal with the seasonal variations of workload series. Dhib et al. [40] employed the SARIMA to fit the workload of the Massively Multiplayers Online Gaming and allocated resources according to predicted workload. Although the experimental results show that the quality of experience is improved, the SARIMA still cannot fit the nonlinear variations of the workload well. Ma et al. [26] designed a workload-prediction algorithm for web applications based on the Deep Belief Networks but only verified its short-term prediction effect. Zhao et al. [23] employed the SVR to predict the workload of web application, and the prediction accuracy reached 89% but only verified the short-term prediction for future three steps. Singh et al. [41] proposed an adaptive prediction model for web application workload using Linear Regression, ARIMA, and SVR models. Similarly, they only verified short-term prediction effect. Given the sufficient long-term memory ability of the LSTM, some scholars attempted to employ it for predicting the long time series. Tian et al. [32], Liu et al. [33], and Wang et al. [34] designed the mid- and long-term prediction models for the traffic flow, the reserve requirements of bank outlets, and the earthquakes based on the LSTM, respectively. As a result, they all obtained good prediction accuracy. However, so far the LSTM was still seldom applied in the mid- and long-term prediction for web-application workloads. Kumar et al. [28] employed the LSTM to carry out the long-term prediction for HTTP requests to web servers in cloud datacenter and claimed to have obtained ideal results, but they did not present the details of the design. Tran et al. [42] designed a LSTM-based algorithm for predicting cloud resource consumption with multivariate time series, but only verified the short-term prediction effect. By contrast, we specially designed a LSTM-based long-term prediction algorithm for the future cycle workload of web application and presented the design details. From experimental results, our LSTM-based prediction algorithm outperforms existing common models and achieves a good accuracy.

3. Problem Domain and Assumptions

First, this work targets at interactive web applications deployed in IaaS cloud. There are various applications deployed in IaaS cloud, such as interactive applications [2, 43], scientific computing [44, 45], and batch processing tasks [46, 47]. Among them, interactive applications usually have a certain business cycle (e.g., one year), the workloads of which generally show similar patterns in the long run, while being stochastic in the short run. Due to the complexity of enterprise-level application architecture, it is difficult to conduct general research on the resource planning of whole application. But, the large application is generally orchestrated by multiple web services or subapplications. Especially in the rise of microservices architecture today, more and more web subapplications run independently as services. This work focuses on planning reserved resources for such web services or subapplications. In addition, such a subapplication is generally composed of several components such as web server, database server, and hard disk. Among them, nonservice components can be statically configured, and service-oriented components need to be scalable. According to the experience knowledge of web development and operation, as long as the numbers of instances of several service-oriented components satisfy a certain ratio with each other, the system can be in a stable state. This ratio can be obtained through application-specific benchmarking.

Next, this work does not involve the IaaS discovery and the selection of cloud providers. These problems belong to another research domain. We assume that the matching instance type of each service component has been found, and the provider has also been selected.

Additionally, only the horizontal scaling scheme is considered in this work. Horizontal scaling adjusts service capacity through dynamically changing the number of instances, while vertical scaling does this through dynamically changing the instance's configuration. However, most providers have not yet opened the services to support dynamic vertical scaling.

Finally, in view of the fact that most providers have sufficient resource capacity nowadays, it is assumed that the provisioning of on-demand instances is not restricted by the quantity. Also, we assume that all reservation contracts are paid completely in advance so as to obtain a larger discount and simplify the problem although several providers also support partial payment.

4. Problem Description and Model Construction

4.1. Provisioning Phases. As illustrated in Figure 2, over the provisioning time horizon, there are three provisioning phases: reservation, utilization, and on-demand phases. The corresponding actions of these phases are performed in different points of time (or events). In the short reservation phase, the decision maker develops a resource-reservation plan and conducts it. In the following utilization phase, the reserved instances are used to deal with incoming workload. During the ongoing utilization phase, once the workload exceeds the processing capacity of reserved instances, an on-demand phase starts, during which additional on-demand instances are provisioned. The reservation and utilization phases always appear in pairs in a sequential order. A utilization phase may contain several on-demand phases. Over the provisioning horizon, there may be multiple pairs of reservation and utilization phases, and the reservation durations may be contained or overlapped by each other.

4.2. Divisions of Resource Provisioning Cycle. As illustrated in Figure 3, we regard a web-application's business cycle as its resource provisioning cycle, namely, resource planning cycle, which consists of several equal-duration provisioning stages.

4.2.1. Resource Planning Cycle. Let T denote a resource planning cycle, which is a relatively long workload-processing cycle defined by the web-application provider. The cycle has a definite beginning and a definite end. During the cycle, although the workload seems to fluctuate randomly in the short term, there is usually a certain pattern implied in workloads from long-term observations. This makes it possible and meaningful to plan resources for a business cycle. Since such a cycle is generally long (e.g., one year), multiple reservation contracts with equal duration or unequal duration can be included in the plan so as to obtain a lower total cost.

4.2.2. Provisioning Stage. As shown in Figure 3, a resource planning cycle T can be divided into several provisioning stages uniformly. Let T_i be the i-th provisioning stage. The duration of a provisioning stage is generally equal to the greatest common divisor of the durations of all reservation contracts so as to ensure that each contract can cover an integer number of stages. For example, an annual planning cycle can be divided into 12 monthly provisioning stages T_1, $T_2, \ldots, T_{12}$. Each provisioning stage can contain one reservation phase ΔT and the whole or part of utilization phases (namely, a utilization phase may cover one or more provisioning stages), as well as one or more on-demand provisioning phases. In particular, as seen in Figure 3, the optimal procurement plan of reservation contracts for the entire cycle T is decided in the phase ΔT_1 of the first stage T_1, and the subplan of procurements corresponding to T_1 is also carried out in ΔT_1. In each subsequent ΔT, its corresponding contract procurements are carried out according to the optimal plan developed in ΔT_1.

4.2.3. Provisioning Time Slot. Due to the workload is usually fluctuating during a provisioning stage, it is not appropriate to provide fixed resources during a provisioning stage. Therefore, as presented in Figure 3, we divide each provisioning stage into several provisioning time slots (e.g., T_{11}, T_{12}, and T_{24}) uniformly for planning resources. Due to the duration of any reserved contract is not shorter than the one of any provisioning stages, the available reserved resources are exactly same for all slots in the same stage. Given it is difficult to obtain the fairly fine granularity of predicted workloads, the duration of each slot is usually set as one day.

4.3. Reservation Contracts. An IaaS provider usually offers multiple reservation-contract types with different durations for consumers. Let K be the set of reservation-contract types, and any contract type $k \in K$ can be expressed as $\langle v, l, p^r \rangle$, where v, l, and p^r denote the offered instance type, contract duration, and unit price, respectively. To describe the conditionality of procurement and utilization of reservation contracts, an annual plan case with 12 months (K_1), 6 months (K_2), 3 months (K_3), and 1 month (K_4) reservation contracts is illustrated in Figure 4. The boxes over the time horizon represent the time coverage of these contracts.

We take the planning cycle $T = \{T_1, T_2, \ldots, T_{12}\}$. Let $|k|$ denote the duration (in unit of provisioning stages) of any k-type contract. Due to only the contracts with a duration of not longer than T are considered, $|k| \le |T|$. Let $Ŧ_k$ denotes the set of stages at which IaaS providers can start provisioning resources with a k-type contract, and then $Ŧ_k$ can be expressed as formula (1). According to Section 4.2.2, $Ŧ_k$ is also the set of stages at which a k-type contract can be purchased. This is because only when a k-type contract is purchased at the stages from $Ŧ_k$ can this contract be properly terminated during T.

$$\mathbf{T}k = \{1, \ldots, |T| - |k| + 1\}. \tag{1}$$

Let F_{ki} denote the set of stages at which some resources reserved by a k-type contract can be utilized in the stage T_i. It means that only when the k-type contracts are purchased at stages belonging to F_{ki}, the resources reserved by these contracts can be utilized during the i-th stage. F_{ki} can be expressed as formula (2). In Figure 4, any set F_{ki} can be obtained. For example, $F_{K_1T_3} = \{T_1\}$, $F_{K_2T_3} = \{T_1, T_2, , T_3\}$, $F_{K_3T_{11}} = \{T_9, T_{10}\}$, and $F_{K_4T_3} = \{T_3\}$. Let n^r_{vki} be the number the k-type contracts with instance type v available in the stage T_i. Let r_{vkj} be the number of the k-type contracts with instance type v purchased at the stage T_j. Based on F_{ki} and r_{vkj}, $n^r_{vki}$is calculated by using the following formula:

$$F_{ki} = \{\max(1, i - |k| + 1) \& \min(i, |T| - |k| + 1)\}, \tag{2}$$

$$n^r_{vki} = \sum_{j \in F_{ik}} r_{vkj}. \tag{3}$$

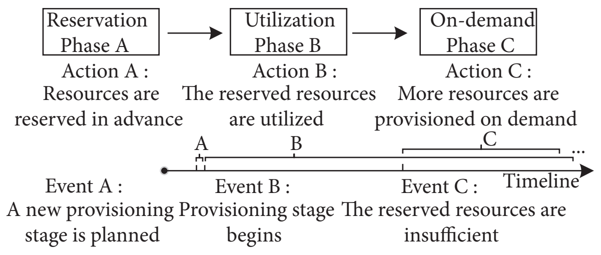
Reservation Phase A
Utilization Phase B
On-demand Phase C
Action A : Resources are reserved in advance
Action B : The reserved resources are utilized
Action C : More resources are provisioned on demand
A
B
C
Timeline
Event A : A new provisioning stage is planned
Event B : Provisioning stage begins
Event C : The reserved resources are insufficient

Figure 2: Transition of provisioning phases.

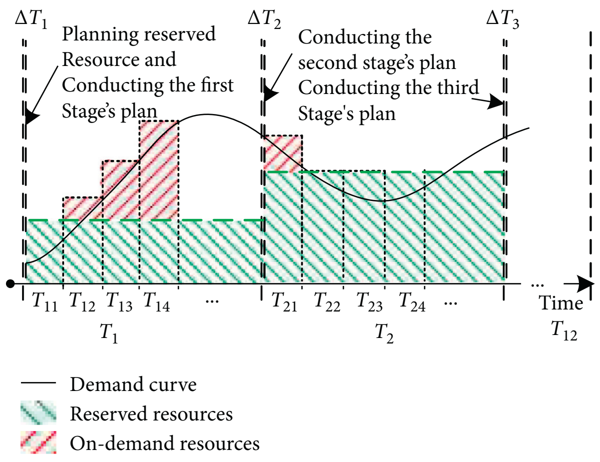
ΔT_1
Planning reserved Resource and Conducting the first Stage's plan
ΔT_2
Conducting the second stage's plan Conducting the third Stage's plan
ΔT_3
T_{11} T_{12} T_{13} T_{14} ...
T_1
T_{21} T_{22} T_{23} T_{24} ...
T_2
Time
T_{12}
Demand curve
Reserved resources
On-demand resources

Figure 3: Divisions of a resource planning cycle.

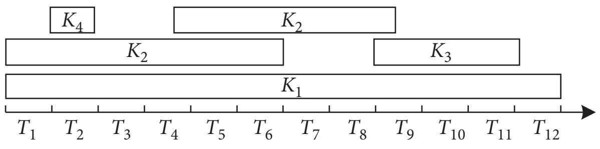
K_4
K_2
K_2
K_3
K_1
T_1 T_2 T_3 T_4 T_5 T_6 T_7 T_8 T_9 T_{10} T_{11} T_{12}

Figure 4: An annual plan case with 5 reservation contracts.

4.4. Model Construction. We choose to perform resources planning based on each time slot rather than each provisioning stage. Owing to the fine granularity of a time slot, the planned resources based on time slots are more adaptable to fluctuating demand, and the amount of overprovisioning and underprovisioning can be reduced greatly. Thus, for a web application service, the resource planning goal is to minimize the total cost of reserved and on-demand resources, while meeting any time-slot's demand in the entire business cycle.

For a specific web application, according to the assumptions in Section 3, several instance types have been selected for its service components. The processing capacity of single web-server instance as well as the optimal ratios between the other server instances and the web-server instances have been determined (the method for determining the former is presented in Section 6, while the latter can be obtained by benchmarking). Besides, it is also assumed that the workload of each time slot has been predicted. Based on these assumptions, the numbers of various service instances required in each time slot are determined. Finally, we have defined some necessary parameters as presented in Table 1 so as to construct the optimization model.

In particular, in the context of horizontal scaling, a service component is deployed on a cluster of instances with the same type; therefore, the instance types correspond to the component types one by one. In addition, let S_i represent the number of time slots in the i-th provisioning stage for $S_i \in S$, while let s represent any one in S_i. Based on defined parameters and variables above, the model is constructed as follows.

First, the cost of all reservation contracts charged to v-type instances in T_i, namely, c^r_{vi}, is expressed as formula (4). Note that r_{vki} is equal to 0 when $i \notin Ŧ_k$.

$$c^r_{vi} = \sum_{k\in K} p^r_{vk} r_{vki}. \tag{4}$$

Next, because the number of v-type instances reserved by k-type contracts available in T_i, namely, n^r_{vki}, is obtained by formula (3), the number of v-type instances available in T_i, namely, n^r_{vi}, can be expressed as follows:

$$n^r_{vi} = \sum_{k\in K} \sum_{t\in F_{ki}} r_{vkt}. \tag{5}$$

In addition, the cost of all v-type instances provisioned on demand in T_i, namely, c^o_{vi}, is expressed as follows:

$$c^o_{vi} = \sum_{j\in S_i} p^o_v \cdot n^o_{vij} \cdot |s|, \tag{6}$$

where the number n^o_{vij} is equal to the maximum of 0 and $n^d_{vij} - n^r_{vi}$, n^d_{vij} denotes the number of v-type instances required in the j-th time slot of the stage T_i, and $|s|$ denotes the number of hours in time slot s.

As a result, the total cost of v-type instances provisioned in T_i, namely, c_{vi}, is equal to the sum of reservation cost c^r_{vi} and on-demand cost c^o_{vi} in T_i. Therefore, the total cost of v-type instances provisioned in the entire planning cycle, namely, c_v, can be expressed as follows:

Table 1: Notation box.

Symbol	Definition
Input parameter	
K	Set of reservation contract types
I	Set of service instance types
T	Set of provisioning stages
P^r	Set of unit prices of reservation contracts while $p^r_{vk} \in P^r$ denotes the price of the k-type reservation-contract charged to a v-type instance
P^o	Set of unit prices of on-demand instances while $p^o_v \in P^o$ denotes the unit price of on-demand v-type instance
S	Set of numbers of time slots in each stage
D	Set of numbers of various instances required in each time slot while $n^d_{vij} \in D$denotes the number of v-type instances required in the j-th time slot of the stage T_i
Decision variable	
r_{vki}	Number of k-type reservation-contracts with v-type instance purchased in T_i
Other parameters	
$Ŧ_k$	Set of stages at which k-type reservation-contracts can be purchased for planning cycle T (obtained by formula (1))
F_{ki}	Set of stages at which some instances reserved by k-type contracts can be utilized during T_i (obtained by formula (2))
n^r_{vki}	Number of v-type instances reserved by k-type contracts available in T_i (obtained by formula (3))
n^r_{vi}	Number of v-type reserved instances available in T_i
n^o_{vij}	Number of v-type instances provisioned on demand for the j-th time slot in T_i
c_v	Total cost of v-type instances provisioned in T
c_{vi}	Cost paid to v-type instances provisioned in T_i
c^r_{vi}	Cost of all reserved contracts charged to v-type instances in T_i
c^o_{vi}	Cost of all v-type instances provisioned on demand in T_i
c^o_{vij}	Cost of all v-type instances provisioned on demand in the j-th time slot in T_i

$$c_v = \sum_{i\in T}\left(\sum_{k\in K} p^r_{vk} r_{vki} + \sum_{j\in S_i} p^o_v \cdot n^o_{vij} \cdot |s|\right). \tag{7}$$

Finally, for the entire planning cycle T, the optimization model of reservation contract procurement for various required instances is constructed as follows:

$$\begin{aligned}
&\min \sum_{v\in I}\sum_{i\in T}\left(\sum_{k\in K} p^r_{vk} r_{vki} + \sum_{j\in S_i} p^o_v \cdot n^o_{vij} \cdot |s|\right)\\
&\text{s.t.} \quad \begin{aligned} & r_{vki} \le \max\left(n^d_{vij}\right), v \in I, k \in K, i \in T, j \in S_i, n^d_{vij} \in D\\ & r_{vki} = 0, v \in I, k \in K, i \in T, Ŧ_k \end{aligned}\\
&n^o_{vij} = \max\left(0, n^d_{vij} - \sum_{k\in K}\sum_{t\in F_{ki}} r_{vkt}\right) \begin{aligned} & v \in I, i \in T, j \in S_i, k \in K, n^d_{vij} \in D\\ & r_{vki} \in \mathrm{N}, p^r_{vk} \in P^r, p^o_v \in P^o, s \in S, S_i \subset S, \end{aligned}
\end{aligned} \tag{8}$$

where only r_{vki} is the decision variables, and the objective function is the linear function of r_{vki}; therefore, this is a Pure Integer Linear Programming (PILP) problem, which can be solved by using the classical Branch and Bound method.

5. Workload Prediction

Considering that the LSTM is designed to combine the short-term and long-term temporal information and exhibits superior long time-series prediction performance, we attempt to design a LSTM-based algorithm for predicting the future cycle workload of a web application.

5.1. Prediction Algorithm Based on the LSTM

5.1.1. Typical LSTM Architecture and Principles. The key to the LSTM is the cell state. Figure 5 illustrates the typical architecture of the LSTM memory cell and the cell's state transition at time $t-1$, t, and $t+1$, and in practice the transition flow usually contains more moments. It can be seen that the cell state runs straight down the entire chain with only some linear interactions, which makes it easy for information to be propagated over time. For the memory cell at time t, there are three inputs: the current input $\mathbf{x}_t$, the previous output $\mathbf{h}_{t-1}$, and the previous state $\mathbf{c}_{t-1}$, and two outputs: the current output $\mathbf{h}_t$ and the current state $\mathbf{c}_t$. The

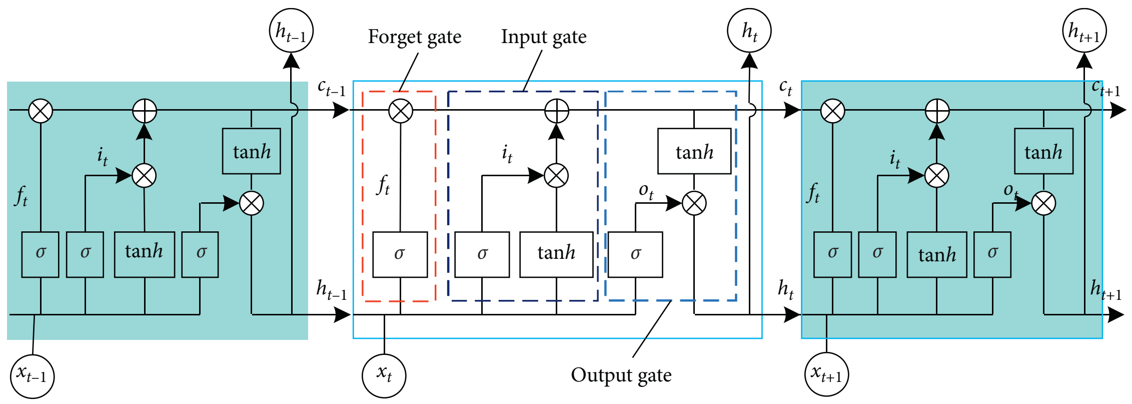

Figure 5: The LSTM memory cell and its state transition.

LSTM uses three gates to control the cell state transition. The forget gate determines how much information of the previous state $\mathbf{c}_{t-1}$ is retained to the current state $\mathbf{c}_t$, while the input gate determines how much information of the current input $\mathbf{x}_t$ is saved to the state $\mathbf{c}_t$. The output gate determines how much information of the current state $\mathbf{c}_t$ is output to $\mathbf{h}_t$, which controls the influence of long-term memory on the current output. The forward calculation of the LSTM is expressed as follows:

$$\mathbf{f}_t = \sigma\left(\mathbf{W}_f \mathbf{h}_{t-1} + \mathbf{U}_f \mathbf{x}_t + \mathbf{b}_f\right), \tag{9}$$

$$\tilde{c}_t = \tan h\left(\mathbf{W}_c \mathbf{h}_{t-1} + \mathbf{U}_c \cdot \mathbf{x}_t + \mathbf{b}_c\right), \tag{10}$$

$$\mathbf{i}_t = \sigma\left(\mathbf{W}_i \cdot \mathbf{h}_{t-1} + \mathbf{U}_i \cdot \mathbf{x}_t + \mathbf{b}_i\right), \tag{11}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{c}_t, \tag{12}$$

$$\mathbf{o}_t = \sigma\left(\mathbf{W}_o \cdot \mathbf{h}_{t-1} + \mathbf{U}_o \cdot \mathbf{x}_t + \mathbf{b}_o\right), \tag{13}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tan h\left(\mathbf{c}_t\right), \tag{14}$$

where **f**, **i**, and **o** denote the forget gate, the input gate, and the output gate, respectively, **W** and **U** matrices are the network parameters, **b** denotes the bias, σ is a sigmoid function, and $\odot$ denotes the product operation.

The LSTM is trained with the Back Propagation Through Time (BPTT) algorithm, which is similar to the Back Propagation (BP) algorithm in principle. The main process is as follows: (1) obtain the outputs by the forward calculation (formulas (9)–(14)); (2) calculate the loss function of each LSTM cell from two backward propagation directions of time and network; and (3) select a gradient optimization algorithm to minimize the loss function and hence optimize system parameters. There are several commonly used gradient optimization algorithms such as the SGD, AdaGrad, RMSProp, and Adam optimizers. Among them, the Adam is a stochastic gradient descent algorithm that combines the advantages of the AdaGrad and RMSProp and can adaptively adjust the learning rate of parameters. By comparison, the Adam performs better in practice.

5.1.2. Prediction Framework Based on the LSTM. As web workload is usually influenced by many factors, such as date, time, and business events, we express web workload as a multivariate time series for training and predicting. The designed workload prediction framework based on the LSTM is illustrated in Figure 6, which contains four functional parts, namely, the data, the LSTM network, the training, and the prediction parts. The data part performs preprocessing on raw historical workload data, such as missing data processing, abnormal data processing, feature extraction, workload series generation, supervised data generation, normalization, and division of training and test sets. The designed LSTM network contains an input layer, a hidden layer, and an output layer. The number of nodes in the input layer and the number of LSTM cells in the hidden layer are both equal to the number of time steps of a workload sequence sample. In Figure 6, **c** and **h** are the state and output of each cell, respectively. The output layer contains an output node p^y, which saves the output of an input sample. In the training part, the process is as follows: (1) the samples are continuously fed into the network, and then the errors are calculated based on formula (15) (where *num* is the number of samples); (2) the network parameters are updated by the Adam optimizer based on the errors; (3) the two steps above are performed iteratively a specified number of times, and finally the network parameters are saved. In the prediction part, the trained network is used to iteratively predict future workload.

$$\text{loss} = \sum_{i=1}^{\text{num}} \frac{\left(p_i^y - y_i\right)^2}{\text{num}}. \tag{15}$$

5.1.3. Supervised Data Generation. The workload at each moment is not only related to its previous values but also related to the date, time, holiday, and other information. Therefore, we express a workload series with length n as $F' = \{X_1', X_2', \ldots, X_n'\}$, where $X_i' = \{x_{i1}', x_{i2}', \ldots, x_{ik}', r_i'\}$ represents the observations at time i, r_i' denotes the workload value, and x_{i1}' to x_{ik}' denote the measurements of the k variables related to r_i'. To avoid the influence of inconsistent

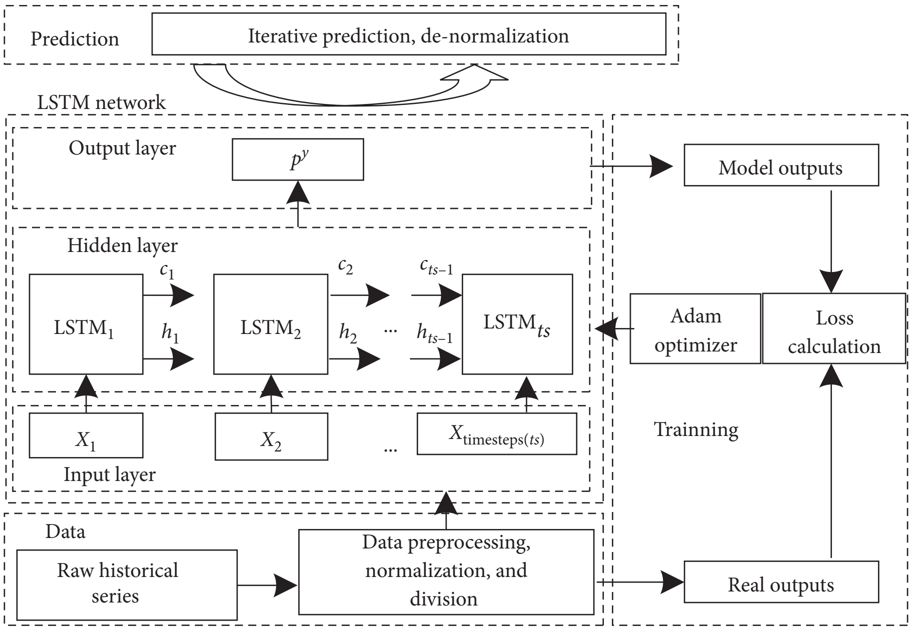

Figure 6: The LSTM-based prediction framework for workload series.

dimensions on the learning, all feature values in the series F' are normalized to unified dimension [0, 1]. As such, a new normalized workload series is obtained as $\mathbf{F} = \{\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n\}$, $\mathbf{X}_i = \{x_{i1}, x_{i2}, \ldots, x_{ik}, r_i\}$, where r_i and x_{i1} to x_{ik} denote normalized r_i' and x_{i1}' to x_{ik}', respectively. Let ts be the number of time steps of a workload sequence that is used as an input of the LSTM network, while let $\mathbf{S}$ be the set of inputs. Based on ts and $\mathbf{F}$, $\mathbf{S}$ is expressed as follows:

$$\begin{aligned} \mathbf{S} &= \{\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_{n-ts}\}, \\ \mathbf{S}_i &= \{\mathbf{X}_i, \mathbf{X}_{i+1}, \ldots, \mathbf{X}_{i+ts-1}\}, \end{aligned} \tag{16}$$

where $i \leq n - ts$ (S_{n-ts+1} has been removed from $\mathbf{S}$ so as to make each input correspond to an output) and $\mathbf{S}_i$ denotes the i-th input sequence. Let $\mathbf{Y}$ be the corresponding output set, which is expressed as follows:

$$\mathbf{Y} = \{y_1, y_2, \ldots, y_{n-ts}\}. \tag{17}$$

Here, y_i is equal to r_{i+ts}. As a result, n-ts samples are obtained based on $\mathbf{S}$ and $\mathbf{Y}$, and then the training and test sets are also be easily obtained after a simple division.

5.1.4. Network Training and Data Predicting. To obtain the best result, we use grid search to optimize three key hyperparameters of the LSTM network: ts (the number of time steps of a input sequence), *units* (the number of neurons in the hidden layer), and η (the Adam optimizer's initial learning rate). Other hyperparameters are set according to general experience. The designed training and predicting process is presented in Algorithm 1. There are several inputs, where ts_l, ts_u, $step_{ts}$, $units_l$, $units_u$, and $step_{units}$ denote the lower bounds, upper bounds, and growth step sizes of ts as well as *units*, respectively. η_Array, m, *seed*, and *epochs* denote the value range of learning rate η, sample division ratio, random-number seed, and iteration times, respectively. The outputs include possible combinations of hyperparameters in the grid and their corresponding test errors, as well as the optimal predicted result and its error.

The algorithm traverses the hyperparameters space, looping the training, and predicting process as shown in lines 7 to 22. In particular, based on the $\mathbf{S}$, $\mathbf{Y}$, and m, line 4 obtains the input set $\mathbf{S}_{tr}$ and the output set $\mathbf{Y}_{tr}$ for the training, as well as the corresponding sets $\mathbf{S}_{te}$ and $\mathbf{Y}_{te}$ for the testing. Line 7 and line 8, respectively, create and initialize the pLSTM model, lines 9 to 12 perform training, and lines 13 to 17 iteratively predict test data. Line 17 uses the current predicted result to update the workload value of the last observations in the next input sequence. $\mathbf{P}_{tr}^y$ and $\mathbf{P}_{te}^y$ are the output sets of the training and the testing, respectively.

6. Determination of Instance Capacity and Time-Slot Baseline Workload

6.1. Determination of Service Instance Capacity. For interactive web applications, service response time is the most important QoS index, and the system designer usually specifies an upper bound for it so as to ensure a good user experience. In fact, there is an inherent relationship among service response time, request arrival rate, and system service capacity. Let C be the service capacity of a virtual instance, which refers to the maximum request arrival rate supported by this instance while the response time index is met. Due to the web request arrival process is a Poisson process and the service time complies with negative exponential distribution, therefore, a service instance with n vCPUs can be modeled as an $M/M/n$ queuing system. Let μ and λ be the average service rate and request arrival rate, respectively, then service intensity ρ is equal to $(\lambda/(n\mu))$. Let ρ_1 be (λ/μ) and p_k be the probability of the state that there are k requests in the system. According to K's algebraic equation, when $k < n$,

$p_k = ((n^n \rho^k p_0)/k!)$ and when $k \geq n$, $p_k = ((n^n \rho^k p_0)/n!)$ [48]. Obviously, $(\sum_{k=0}^{\infty} p_k = 1)$. After deducing the formulas, p_0 is expressed as follows (where $(\rho < 1)$):

$$p_0 = \left(\sum_{k=0}^{n-1} \frac{\rho_1^k}{k!} + \frac{\rho_1^n}{n!} \frac{1}{1-\rho} \right)^{-1}. \tag{18}$$

Additionally, let L_s, L_q, and L_{busy} be the average number of requests, the number of queued requests, and the number of busy vCPUs in the system, respectively. Apparently, $L_s = L_q + L_{busy}$, $L_{q=}\sum_{k=n}^{+\infty} (k-n) p_k$, and $L_{busy\,=\,}\rho_1$. After some derivations, L_s is expressed as follows:

$$L_s = \frac{\rho \rho_1^n p_0}{n!(1-\rho)^2} + \rho_1. \tag{19}$$

According to Little's formula, the service response time, namely, the average staying time $\overline{t}_s$ of a request in the system is calculated as follows:

$$\overline{t}_s = \frac{L_s}{\lambda} = \frac{\rho_1^n p_0}{\mu n \cdot n!(1-\rho)^2} + \frac{1}{\mu}. \tag{20}$$

If $\overline{t}_{\max}$ is the upper bound of acceptable response time, that is, $\overline{t}_s \leq \overline{t}_{\max}$, then the allowable maximum request arrival rate is determined based on $\overline{t}_{\max}$ by the formulas (18) and (20), which is exactly the service capacity C of the instance.

6.2. Determination of Time-Slot Baseline Workload. We consider that the baseline workload for planning slot should be set this way as far as possible to meet all workload demands after excluding few abnormal values. As the workload distribution of the adjacent planning cycle is similar, the workload statistics of the last cycle can be used to transform current predicted workload so as to obtain the reasonable baseline workload. It is assumed that the last historical cycle contains m time-slots (e.g., a year contains 365 days), any one of which contains ΔT(e.g., 10 minutes), and the number of requests during each ΔT has been counted. First, the request numbers of all ΔTs are sorted in the descending order, and then a two-dimensional array d is obtained, where d_{ij} represents the j-th largest workload in time slot i. Next, we specify a workloads-ratio threshold fr (e.g., $fr = 0.1$) and then calculate the average of time-ratios of m slots, namely, tr, as follows:

$$tr = \frac{\left(\sum_{i=1}^{m} (l_i/n)\right)}{m}, \tag{21}$$

where l_i is calculated based on the constraints: $((\sum_{j=1}^{l_i-1} d_{ij})/(\sum_{j=1}^{n} d_{ij})) \leq \mathrm{fr} \leq ((\sum_{j=1}^{l_i} d_{ij})/(\sum_{j=1}^{n} d_{ij}))$.

In particular, tr denotes the average cumulative-time-ratio of several sequenced peak workloads with a cumulative-workloads-ratio fr for time slots in a stage. For example, $fr = 0.2$ and $tr = 0.1$ means that, on average, 20 percent of peak workloads in a slot only takes up 10 percent of time. Then, the average of a certain percentage of sequenced peak workloads can be used as the time-slot's baseline workload. Let D_s denotes the number of requests in slot s, while $|s|$ denotes the number of seconds in s, then the baseline workload λ_s of time slot s is calculated as follows:

$$(\lambda_s = (\mathrm{fr} \cdot D_s)/(\mathrm{tr} \cdot |s|)). \tag{22}$$

7. Experimental Evaluation

7.1. Experimental Environment, Datasets, and Evaluation Criteria

7.1.1. Experimental Environment. The experiments were run on an OS Win10 machine with 16 GB of memory and 3.0 GHz Intel Core i7 processor. By using Python 3.7 under the PyCharm 2019.1.3, the LSTM-based prediction algorithm was developed through the use of the Tensorflow 1.13.1 framework, and the experimental SARIMA and Holter–Winters models were developed based on Statsmodels 0.10.1 package, while the experimental SVR model was developed based on the machine-learning toolkit Sklearn 0.21. To solve the optimization problem, LINGO 15 [49] solver was used.

7.1.2. Datasets. The LAcity.org website traffic dataset from Kaggle [50] was chosen to evaluate the workload prediction approaches. This is a dataset hosted by the city of Los Angeles, which contains detailed daily traffic data from January 1, 2014, to July 12, 2019, for lacity.org, the main website for the city of Los Angeles. We obtained the numbers of daily requests after preprocessing and then intercepted the data from January 1, 2014, to December 31, 2018, for evaluations. The distribution of workloads is shown in Figure 7. To obtain fine-grained web-traffic data to simulate the determination of time-slot baseline workload, the YOOCHOOSE dataset was also downloaded from Kaggle [51], in which all clicks of users over a retailer's website had been recorded. After preprocessing, we obtained the numbers of requests per 10 minutes over the website from June 1, 2014 to August 31, 2014. The distribution of workloads is shown in Figure 8.

7.1.3. Evaluation Criteria. We mainly used the Mean Absolute Error (MAE), the Mean Absolute Percentage Error (MAPE), and the Root Mean Square Error (RMSE) as the evaluation criteria to gauge the prediction accuracy, which were calculated as the formulas (27)–(29), respectively, where the parameter n denotes the number of observations, y_i is the actual workload, and $\widehat{y}_i$ represents the predicted workload.

$$\mathrm{MAE} = \frac{1}{n} \sum_{i=1}^{n} \left| \widehat{y}_i - y_i \right|, \tag{23}$$

$$\mathrm{MAPE} = \frac{100\%}{n} \sum_{i=1}^{n} \left| \frac{\widehat{y}_i - y_i}{y_i} \right|, \tag{24}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (\widehat{y}_i - y_i)^2}. \tag{25}$$

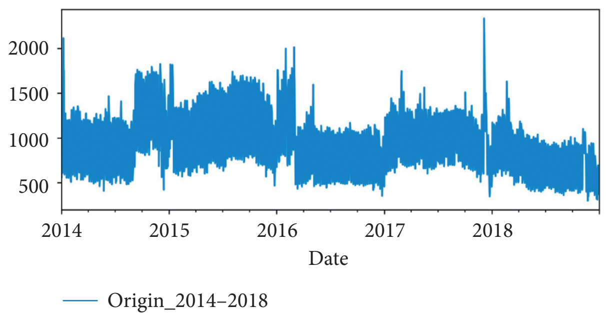

FIGURE 7: Distribution of daily request rate for lacity.org.

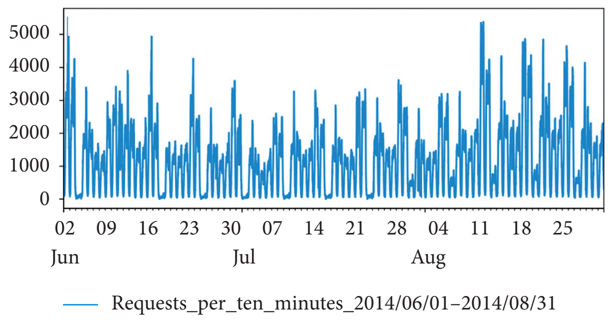

FIGURE 8: Distribution of requests per 10 minutes for a retailer's website.

7.2. Evaluation for Workload Estimation/Prediction. In this section, we first introduce several typical estimation or prediction approaches for future cycle workloads and then present the result of the LSTM-based prediction algorithm.

7.2.1. Historical-Workload-Based Estimations. Some studies directly used the historical cycle's workloads as an estimation of the current cycle's workloads [14, 15, 17]. Similarly, we used the workloads from 2014 to 2017 as the estimations of the workloads in the following years, namely, 2015 to 2018, respectively. The results are shown in Figure 9, where only dark red overlapping areas are accurately estimated areas. Their Mean Absolute Percentage Errors (MAPEs) are 27.0%, 69.1%, 29.4%, and 51.7%, respectively. Apparently, the accuracies are poor.

7.2.2. Holter–Winters Seasonal Models. Several typical exponential smoothing models are often used in workload predicting, which include single exponential smoothing, double exponential smoothing, and multiple-parameter exponential smoothing (namely, Holter–Winters seasonal models). Among them, cubic exponential smoothing-based Holter–Winters seasonal models can deal with seasonality and trends, which are classified as additive model and multiplicative model. In the additive model, several components such as level value, seasonal trend, and linear trend are considered to be independent of each other, and so they are directly added. In the multiplicative model, these components are considered to be influenced by each other, and so they are directly multiplied. Given the workload fluctuations is relatively gentle, we selected the Holter–Winters additive model for predicting. The prediction formula is shown as follows:

$$\hat{x}_{t+k} = a_t + b_t k + S_{t-s+k}, \quad k \in \mathrm{N}^+, \tag{26}$$

where a_t and b_t are the intercepts and S_{t-s+k} and s denote seasonal component and period length, respectively. The first three parameters are calculated as follows:

$$\begin{aligned} a_t &= \alpha\left(x_t - S_{t-s}\right) + (1-\alpha)\left(a_{t-1} + b_{t-1}\right),\\ b_t &= \beta\left(a_t - a_{t-1}\right) + (1-\beta)b_{t-1},\\ S_t &= \gamma\left(x_t - a_t\right) + (1-\gamma)S_{t-s}. \end{aligned} \tag{27}$$

Here, α, β, and γ are three damping factors, and $\alpha, \beta, \gamma \in (0, 1)$. For the purpose of comparison, we used both additive and multiplicative models for predicting, and the results are shown in Figure 10. It can be seen that the trends predicted by the multiplicative model decay dramatically from the beginning so that the prediction cannot continue after a while. The additive model can basically predict the trends and periods, but the MAPE reached 33.8%, and obviously the overall accuracy is still low.

7.2.3. SARIMA Model. AR, MA, ARMA, ARIMA, and SARIMA are several typical time series models. The first three models are only suitable for stationary series, while the ARIMA can make some nonstationary series become stationary through differencing. Given the SARIMA can further deal with the seasonal trends compared with the ARIMA, we employed the SARIMA to perform the prediction and the comparison. The SARIMA model is generally expressed as follows:

$$\phi(B)\Phi\left({}^{BS}\right)^{\nabla d}\nabla_{S\,xt}^{D} = c + \theta(B)\Theta(BS)\varepsilon_t. \tag{28}$$

Here,

$$\begin{cases} \phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \cdots \phi_p B^p,\\ \theta(B) = 1 - \theta_1 B - \theta_2 B^2 - \cdots \theta_q B^q,\\ \Phi\left(B^S\right) = 1 - \Phi_1 B^S - \Phi_2 B^{2S} - \cdots - \Phi_P B^{PS},\\ \Theta\left(B^S\right) = 1 - \Theta_1 B^S - \Theta_2 B^{2S} - \cdots - \Theta_Q B^{QS},\\ \nabla^d = (1-B)^d,\\ \nabla_S^D = (1-BS)^D, \end{cases} \tag{29}$$

where S, D, d, ε_t, and c denote the length of seasonal period, the times of seasonal difference and ordinary difference, the white Gaussian noise, and the constant term, respectively. $\phi(B)$is an autoregressive polynomial, $(\phi_1, \phi_2, \ldots, \phi_p)$ are the autoregressive coefficients, $\Phi(B^S)$ is a seasonal autoregressive polynomial, and $(\Phi_1, \Phi_2, \ldots, \Phi_P)$ are the seasonal autoregressive coefficients. Meanwhile, $\theta(B)$ is a moving average polynomial, $(\theta_1, \theta_2, \ldots, \theta_q)$ are the moving average coefficients, $\Theta(B^S)$ is a seasonal moving average polynomial, and $(\Theta_1, \Theta_2, \ldots, \Theta_Q)$ are the seasonal moving average coefficients. Here, p, P, q, and Q are the orders of $\phi(B)$, $\Phi(B^S)$, $\theta(B)$, and $\Theta(B^S)$, respectively. In addition, B is the ordinary

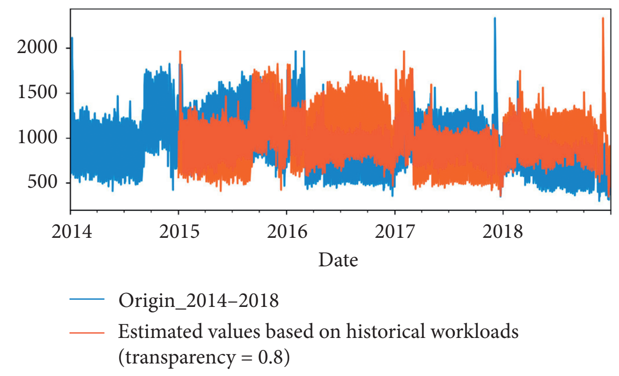

FIGURE 9: Estimated results based on the historical cycle's workloads.

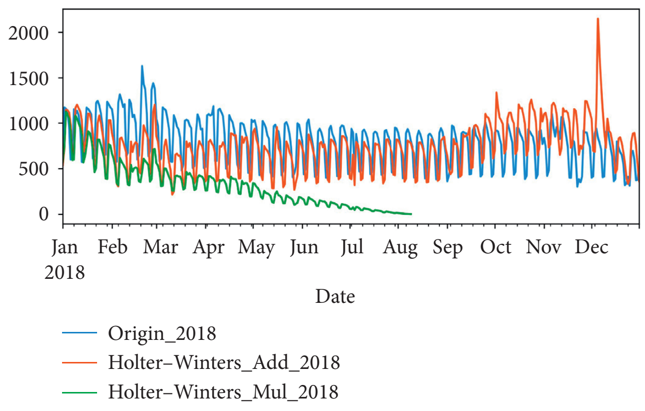

FIGURE 10: Comparison of predicted results with origin data.

lag operator, B^S is the seasonal lag operator, and ∇^d is the ordinary difference operator, while ∇_S^D is the seasonal difference operator. $\nabla^d\nabla_S^D x_t$ represents a stationary time series. The model expressed as equation (28) can be abbreviated as SARIMA (p, d, q) $(P, D, Q)_S$, which is constructed based on p, d, q, P, D, Q, and S. The process of determining these parameters is as follows.

First, the S-steps (S is equal to period length) periodic difference is performed D times to eliminate seasonal trends, and then the ordinary difference is performed d times based on the results of stationarity checking so that the series become stationary. In this process, S is determined by observing the time-series diagram, while D is equal to the times of periodic difference and d is equal to the times of ordinary difference. In general, D and d do not exceed 3. Second, the order p can be determined based on the tailing or truncation of partial autocorrelation coefficients in the partial autocorrelogram. Meanwhile, the order q can be determined based on the tailing or truncation of autocorrelation coefficients in the autocorrelogram. Similarly, the orders P and Q can also be determined based on the tailing or truncation of the autocorrelation and partial autocorrelation coefficients over the time-lag points with several times of period length. Finally, the SARIMA model is created based on the determined parameters above and then is fitted based on the samples. The results of iterative prediction are shown in Figure 11. The MAPE is 22.3%, the MAE is 190.5, and the RMSE is 253.3, respectively. It can be seen that the overall prediction for seasonal and linear trends is relatively accurate, but the detailed prediction is poor.

7.2.4. SVR Model. The Support Vector Machine (SVM) is an innovative statistical learning model proposed by Cortes and Vapnik based on the principle of structural risk minimization [52]. It has excellent generalization capabilities and can deal with small sample, nonlinear, high-dimensional learning problems. The SVR is the application of the SVM in the data regression and prediction. The applications of SVR in workload forecasting have been also widely studied [23, 53–57]. In the process of the SVR nonlinear regression and prediction, the original data is mapped to the high-dimensional space through the use of a nonlinear mapping, where a linear function can be found to fit the input and output values of samples, and then the prediction is done based on this function. Given the workload of the LAcity.org exhibits obvious nonlinear characteristics, we choose to use the nonlinear $\varepsilon - \mathrm{SVR}$ model for the prediction and the comparison.

Suppose there is a sample set: $\{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \ldots, (\mathbf{x}_n, \mathbf{y}_n)\}$, $\mathbf{x}_i \in \mathbf{R}^d$, and $y_i \in \mathbf{R}$, where d denotes the feature dimension and n denotes the number of samples. After the set is mapped to the high-dimensional space, its linear fitted function can be expressed as follows:

$$f(\mathbf{x}) = \boldsymbol{\omega} \cdot \varphi(\mathbf{x}) + b, \tag{30}$$

where $\varphi(\mathbf{x})$ is the nonlinear mapping function from the original data to the high-dimensional space, ω is the coefficient vector, and b is the offset. According to the principle of the $\varepsilon - \mathrm{SVR}$ model, the goal of learning is to make $f(\mathbf{x})$ and y as close as possible but tolerate a deviation with the maximum value ε between $f(\mathbf{x})$ and y; that is, the loss is calculated only when the deviation is greater than ε. Also, considering a few samples is still unable to be fitted under the accuracy ε, the slack variables (ξ_i) and (ξ_i^*) are introduced. Thus, based on the principle of structural risk minimization, the function estimation problem is transformed into the following optimization problem:

$$\min \frac{1}{2}\|\boldsymbol{\omega}\|^2 + C\frac{1}{n}\sum_{i=1}^{n}(\xi_i + \xi_i^*)$$
$$\text{s.t.} \begin{cases} y_i - \boldsymbol{\omega} \cdot \varphi(\mathbf{x}_i) - b \le \varepsilon + \xi_i \\ \boldsymbol{\omega} \cdot \varphi(\mathbf{x}_i) + b - y_i \le \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \ge 0, \quad i = 1, 2, \ldots, n, \end{cases} \tag{31}$$

where C is the penalty coefficient, which determines how well the regression function fits the data. In order to facilitate solving the problem, the Lagrange multipliers α and α^* are introduced, and the above problem is transformed into the dual problem:

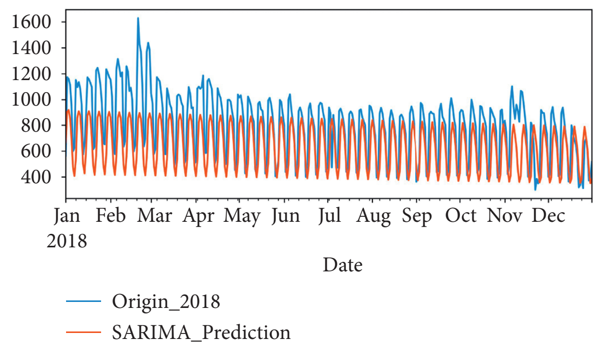

Figure 11: Distribution of predicted values from the SARIMA model.

$$\min \frac{1}{2}\sum_{i,j=1}^{n}\left(\alpha_i-\alpha_i^*\right)\left(\alpha_j-\alpha_j^*\right)K\langle \mathbf{x}_i,\mathbf{x}_j\rangle+\varepsilon\sum_{i=1}^{n}\left(\alpha_i+\alpha_i^*\right)+\sum_{i=1}^{n}y_i\left(\alpha_i-\alpha_i^*\right)\text{s.t.}\quad \left\{\sum_{i}^{n}\left(\alpha_i-\alpha_i^*\right)=0 0\le\alpha_i,a_i^*\le C,\quad i=1,2,\ldots,n,\right. \tag{32}$$

where through the use of kernel function $\mathbf{K}\langle\mathbf{x}_i,\mathbf{x}_j\rangle=\langle\varphi(\mathbf{x}_i),\varphi(\mathbf{x}_j)\rangle$, the calculation of vector inner product in the high-dimensional space is converted to the corresponding calculation in the original low-dimensional space, avoiding the problem of dimension explosion. By solving this problem, α_i and a_i^* are obtained, then multiple samples satisfying $0<\alpha_i$, $a_i^*<C$ can be chosen to solve for b and the average value of b is used (namely, $\overline{b}$), so the regression function is obtained as follows:

$$f(\mathbf{x})=\sum_{i=1}^{n}\left(\alpha_i-\alpha_i^*\right)\mathbf{K}\langle\mathbf{x}_i,\mathbf{x}\rangle+\overline{b}. \tag{33}$$

After data preprocessing, we obtained the normalized daily-workload series of the LAcity.org from 2014 to 2018: $R=\{r_1, r_2,\ldots, r_n\}$, the corresponding month-feature series: $M=\{m_1, m_2,\ldots, m_n\}$, and the corresponding workday-feature series: $D=\{d_1, d_2,\ldots, d_n\}$, where n is the length of these series, r_i denotes the value of the i-th workload (namely, the number of daily requests), m_i denotes the month-number feature corresponding to the i-th workload, and d_i denotes the corresponding workday-number feature. The reason for choosing the feature m and d is that the analysis found that the daily workload is closely related to its date attributes. Then, we designed the input set as $\mathbf{X}=\{\mathbf{x}_1, \mathbf{x}_2,\ldots, \mathbf{x}_{n-lag}\}$ and the output set as $\mathbf{Y}=\{y_1, y_2, \ldots, y_{n-lag}\}$ for the SVR model, where $\mathbf{x}_i=\{r_i, r_{i+1},\ldots, r_{i+lag-1}, m_{i+lag}, d_{i+lag}\}$, $y_i=r_{i+lag}$, and the adjustable parameter *lag* denotes the length of time lag. The value of *lag* implies that the current workload is most relevant to the recent *lag* historical workloads. Finally, we developed the SVR-based prediction algorithm through the use of the toolkit Sklearn. Considering the good adaptability of the radial basis function, we chose it as the kernel function (namely, $K\langle x_i,x\rangle=\exp(-\gamma\|x_i-x\|^2)$). Also, we applied grid search and cross-validation to determine the values of *lag*, C (penalty coefficient) and γ (width coefficient of the radial basis function) and sorted the predicted results according to the MAPE. The top five optimal combinations of hyperparameters, corresponding errors, and time cost are illustrated in Table 2. Meanwhile, the distributions of predicted results and original workloads are shown in Figure 12.

From the results, the prediction accuracy reaches 86% and the computational overhead is low, which is mainly due to the use of kernel function. As can be seen from Figure 12, the predicted results fit well with the original series in terms of the level values, the seasonality, and the trends, and the prediction of the details is also good. The disadvantage is that the predicted values of valley workloads are generally higher than the actual values.

7.2.5. Prediction Algorithm Based on the LSTM. We implemented the training and prediction process of the LSTM network according to Algorithm 1. First, several general parameters were set empirically, where the random-number seed was set as 1 and the number of iterations was set as 200. Then, the value range of three key hyperparameters was set. We let the number of time steps of a sequence sample, namely, *ts*, belong to $\{2,3,\ldots,60\}$, let the neuron number of the hidden layer, namely, *units*, belong to $\{2,3,\ldots,60\}$, and let the learning rate, namely, η, belong to $\{0.001, 0.003, 0.005, 0.007, 0.01, 0.02, 0.03, 0.04, 0.05, 0.07, 0.1\}$. The step sizes of *ts* and *units* were all set to be 1, and the loss function was set as the Mean Square Error (MSE) according to formula (15). Finally, we ran this program to traverse all combinations of hyperparameters. According to the MAPE values, the top five optimal combinations, corresponding errors, and time cost are illustrated in Table 3. Meanwhile, the distributions of predicted workloads and original workloads are shown together in Figure 13.

From the results, the LSTM model makes a better prediction for the annual workloads than previous several

```
Input: $(ts_1, ts_u, step_{ts})$, $(units_1, units_u, step_{units})$, $\eta\_Array$, $m$, $seed$, $epochs$, and $min\_error = +\infty$
Output: $pra\_results$, $best\_pred$, and $min\_error$
(1) $\mathbf{F} = \text{normalize}\ (F')$;
(2) for each $ts$ in $ts_1$: $ts_u$ by $step_{ts}$
(3)   get $\mathbf{S}$, $\mathbf{Y}$ from $\mathbf{F}$ by $ts$;
(4)   get $\mathbf{S}_{tr}$, $\mathbf{Y}_{tr}$, $\mathbf{S}_{te}$, $\mathbf{Y}_{te}$ from $\mathbf{S}$, $\mathbf{Y}$ by $m$;
(5)   for each $\eta$ in $\eta\_Array$
(6)     for each $q$ in $units_1$: $units_u$ by $step_{units}$
(7)       create pLSTM by $ts$, $q$;
(8)       initialize pLSTM by $seed$;
(9)       for each step in 1: $epochs$
(10)        $\mathbf{P}^y_{tr} = \text{pLSTM}_{\text{forward}}\ (\mathbf{S}_{tr})$;
(11)        get $loss$ from $\mathbf{P}^y_{tr}$, $\mathbf{Y}_{tr}$;
(12)        update pLSTM by Adam with $loss$ and $\eta$;
(13)      for each $i$ in 0: length $(\mathbf{S}_{te}) - 1$
(14)        $p^y_i = \text{pLSTM}\ (\mathbf{S}_{te}\ [i])$;
(15)        append $p^y_i$ to $\mathbf{P}^y_{te}$;
(16)        if $i <$ length $(\mathbf{S}_{te}) - 1$
(17)          $\mathbf{S}_{te}\ [i+1][ts-1][k-1] = p^y_i$;
(18)      $\mathbf{P}^y_{te} = \text{denormalize}\ (\mathbf{P}^y_{te})$;
(19)      get $error$ by $\mathbf{P}^y_{te}$, $\mathbf{Y}_{te}$;
(20)      append $[ts, \eta, q, error]$ to $pra\_results$;
(21)      if $error < min\_error$
(22)        $best\_pred = \mathbf{P}^y_{te}$; $min\_error = error$;
(23) return $pra\_results$, $best\_pred$, $min\_error$;
```

Algorithm 1: Network training and data predicting.

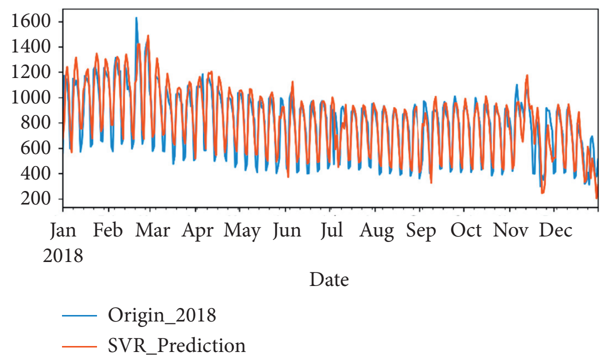

Figure 12: Distribution of predicted values from the SVR model.

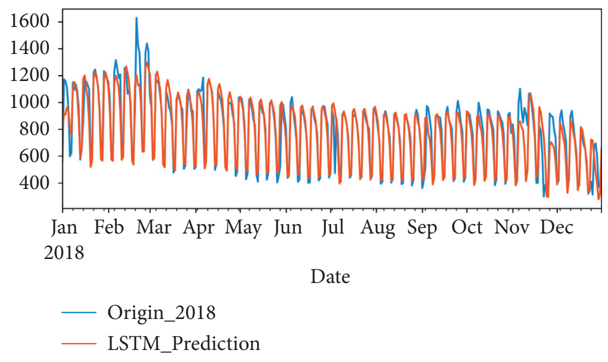

Figure 13: Distribution of predicted values from the LSTM model.

Table 2: The best five hyperparameter combinations of the SVR model.

Rank	*lag*	*C*	*γ*	*Mape*	*Rmse*	*mae*	*time_cost*
1	9	16	0.25	13.2%	117	90	4.61
2	45	32	0.004	13.3%	114	85	6.19
3	21	8	0.125	13.7%	120	95	5.17
4	25	2	0.125	14.4%	119	93	5.73
5	12	8	0.25	14.7%	123	97	4.38

approaches. For the several optimal hyperparameter combinations, the average accuracy is more than 90%, and the computational overhead is also low. As can be seen from Figure 13, the predicted results fit well with the original series in terms of the level values, the seasonality, the trends, and the details. Compared with the SVR model, the LSTM model is superior in terms of overall accuracy and detailed prediction. The LSTM model exhibits excellent prediction performance for long time series, which is mainly attributed to its strong ability of learning the long-term and short-term temporal information simultaneously.

7.3. Evaluation for the Optimization Model

7.3.1. Simulations of Resource Provisioning Scenarios. Ideally, the evaluation for the reservation-contract-procurement-optimization model should be based on a real web application and its workloads. However, there are only some public web-traffic datasets are available. Given the simulation of the running of web applications does not affect the evaluation, we used the predicted workloads of LAcity.org in 2018 from the LSTM model to simulate the running of a real web application. We assumed that the application contained two elastic components, namely, the web server and the database server. Meanwhile, we also assumed that Aliyun's ecs.g5.large and mysql.n4.medium instances in North China [3] had been selected for the two servers, respectively. For the ecs.g5.large and mysql.n4.medium instances, the on-demand unit prices (namely, hourly rate), the reservation-contract unit prices, and their corresponding discount rates of monthly cost compared with the on-demand plan are listed in Table 4 in turn, where the unit of cost is RMB Yuan. Apparently, these reservation contracts offer considerable cost discounts, and the longer the contract duration, the higher the discount rate.

7.3.2. Determination of Instance Capacity. In real scenarios, the instance's average service rate can be obtained through benchmarking, combined with the specified response time index; the instance processing capacity can be calculated according to formulas (18) and (20). However, the capacity is difficult to be determined without the related application suite or benchmarking abilities. As this number only has an influence on the absolute cost figure and does not affect the evaluation of optimization model. Therefore, we assumed that the capacity of single web-server instance in the simulated application was 50 requests per second. Moreover, it was also assumed that the system performance was optimal when the instance numbers of web server and database server meet the ratio of 1 : 1.

7.3.3. Determination of Time-Slot Baseline Workload. We used the YOOCHOOSE dataset to simulate the determination of baseline workload for time slot. After preprocessing, the numbers of requests per 10 minutes from June 1, 2014, to August 31, 2014, were obtained. According to Section 6.2, we treated a day as a time slot and then calculated the baseline workloads from August 1 to August 31 based on the statistics of workloads from June 1 to July 31. First, we let *fr* belong to [0.001, 0.4], namely, the range of cumulative-peak-workloads ratio and traversed this range in a step size equaling to 0.002 to calculate the corresponding *tr*, namely, the average cumulative-time ratio from June 1 to July 31. Second, the daily baseline workloads in August were calculated, respectively, based on each pair of *fr* and *tr* above, and then the average probability that time-slot workloads were met (let *avg_fullfilled_rate* represents this probability) was obtained for each pair of *fr* and *tr*. Finally, as presented in Table 5, the best five results are listed after ranking *avg_fullfilled_rate*. In fact, in this way, the baseline workload is equal to fr/tr times of the average original workload. As can be seen from Table 5, the best five *avg_fullfilled_rates* are above 96%. To obtain a higher *avg_fullfilled_rate*, the fr/tr coefficient can be finely increased manually, also resulting in more resource costs. In short, determining the time-slot's baseline workload in this way greatly alleviates the adverse impact of coarse-grained predicted workloads on resource planning.

7.3.4. Model Evaluation. Based on the above simulated scenarios and related data, combined with the predicted daily workloads of LAcity.org in 2018, the reservation-contract-procurement-optimization model for the simulated application was constructed and solved in LINGO15. To evaluate the effect, several resource provisioning schemes were compared, and the results were presented in Table 6, where the Reservation Contract Procurement Optimization Based on Predicted Workload (RCPOBPW) scheme means to first determine the reservation contract procurement plan based on the approaches presented in this paper and then carry out the plan as well as supplement necessary on-demand resources during the future business cycle. The Reservation Contract Procurement Based on Average Predicted Workloads (RCPBAPW) scheme means to first purchase the fixed number of reservation contracts once based on average predicted workload and then supplement necessary on-demand resources during the future cycle. The Using only Reserved Resources Provisioned by One Kind of Contracts (URRPOC) scheme means to purchase the fixed number of reservation contracts once based on the maximum predicted workload and do not use any on-demand resources. Additionally, the Using only Reserved Resources (URR) scheme and the Using only On-demand Resources (UOR) scheme mean to use only reserved resources and use only on-demand resources during the future cycle, respectively. The Reservation Contract Procurement Optimization Based on Real Workloads (RCPOBRW) scheme is theoretically optimal, which differs from RCPOBPW only in that it determines the reservation plan based on real workloads. In Table 6, WN_{C1}~WN_{C4} are, respectively, the numbers of web server instances with 1 month, 3 months, 6 months, and 1 year reservation contracts, while DN_{C1}~DN_{C4} are, respectively, the corresponding numbers of database server instances. C, C_0/C, and R_S denote total resource cost, the ratio of on-demand resource costs to total costs, and the cost ratio of each scheme to the RCPOBPW scheme, respectively, while R_{SLA} denotes the SLA satisfaction rate of each scheme.

As can be seen from the *WN* and *DN* columns, except for the UOR, all schemes use reserved resources, and the reservation contracts with the longest duration are purchased the most. Except that the RCPBAPW and URRPOC scheme only purchase the longest-duration contracts based on fixed workloads, other schemes using reserved resources (e.g., RCPOBPW, URR, and RCPOBRW) have purchased various contracts. From the total cost, the UOR scheme using only on-demand resources is the highest, the URRPOC scheme using only reserved resources provided by one kind of contracts is the second, and followed by the RCPBAPW and URR. Obviously, our RCPOBPW is the least costly practical scheme, and its cost is only 0.4% more than the theoretical optimal scheme. From the SLA satisfaction rate, all schemes can fully meet the demands except for the URRPOC and URR schemes, which do not use on-demand resources. Overall, our RCPOBPW scheme is the best among the five practical schemes. Finally, several conclusions can be drawn as follows: (1) it is not appropriate to use completely on-demand resources, which will result in huge expenditures; (2) it is also not appropriate to use completely reserved

Table 3: The best five hyperparameter combinations of the LSTM model.

Rank	*ts*	*lr*	*units*	*Mape* (%)	*Rmse*	*mae*	*time_cost* (s)
1	42	0.05	36	9.12	104	65	5.87
2	42	0.05	33	9.31	106	66	5.21
3	45	0.007	45	9.34	106	68	7.23
4	45	0.005	33	9.37	107	68	6.95
5	36	0.05	30	9.40	107	67	4.65

Table 4: The unit prices and discount rates of on-demand plans and reservation contracts for the cs.g5.large and mysql.n4.medium instances.

Duration	1 hour	1 month	3 months	6 months	12 months
Price1	0.91	256.25	692.25	1308.00	2437.80
Discount rate1	0	60.9%	64.8%	66.7%	69.0%
Price2	1.03	346	975.63	1821.72	3387.84
Discount rate2	0	53.3%	56.1%	59.1%	61.9%

Table 5: The best five pairs of *fr* and *tr* as well as corresponding results.

fr	*tr*	*fr/tr*	*avg_fullfilled_rate* (%)
0.111	0.061	1.828	96.5
0.125	0.068	1.827	96.5
0.141	0.077	1.825	96.5
0.137	0.075	1.821	96.4
0.113	0.062	1.820	96.4

Table 6: Comparison of several typical resource provisioning schemes.

Scheme	WN_{C1}	WN_{C2}	WN_{C3}	WN_{C4}	DN_{C1}	DN_{C2}	DN_{C3}	DN_{C4}	C	C_O/C (%)	R_S (%)	R_{SLA} (%)
RCPOBPW	3	2	2	18	4	3	2	17	123611	6.85	100	100
RCPBAPW	0	0	0	16	0	0	0	16	134183	33.4	108.6	100
URRPOC	0	0	0	27	0	0	0	27	157292	0	127.2	98.33
URR	14	5	1	19	14	5	1	19	130588	0	106	96.67
UOR	0	0	0	0	0	0	0	0	277079	100	224.2	100
RCPOBRW	7	1	2	18	8	8	8	18	123174	6.2	99.6	100

resources, as it is likely that some unexpected workloads cannot be handled; and (3) it is advisable to use a combination of on-demand and reserve resources and take as many contracts as possible into account to maximize the share of reserved resources so as to achieve the greatest cost discounts while meeting the demand.

8. Conclusions and Future Work

In this paper, we investigated the resource-reservation-planning problems for cloud-based web applications. First, we developed an integer linear program model for optimizing the reservation-contracts procurement. Then, we designed the LSTM-based algorithm for predicting the business cycle's workloads of web applications. Thereafter, the approaches for determining the instance capacity and the baseline workload of time slot were also presented. Finally, experimental evaluations were carried out based on several real datasets. From the comparison of predicted results, our LSTM-based algorithm achieves better effect than the Holter–Winters, SARIMA, and SVR models, with an accuracy of about 90%. This result is attributed to the LSTM network's good memory and learning ability for long time series and also related to its learning of workload-related information such as date and time. Meanwhile, from the comparative results of several typical practical provisioning schemes, the scheme based on the optimization model presented in this paper achieves the least resource cost while entirely satisfying future demands.

However, for a cloud-based web application, although the optimal resource-reservation plan can be obtained based on the proposed solution in this paper, the problem of how to dynamically provision on-demand resources during the business cycle remains to be solved, which is worth in-depth study.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the Youth Project of Science and Technology Research Program of the Chongqing Education Commission of China (no. KJQN201901414) and the Startup Foundation for Introducing Talent of Yangtze Normal University, China (no. 0107/010721481).

References

[1] P. A. Dinda and D. R. O'hallaron, "Host load prediction using linear models," *Cluster Computing*, vol. 3, no. 4, pp. 265–280, 2000.

[2] 12306, https://www.12306.cn/index/, 2019.

[3] Aliyun, https://www.aliyun.com/, 2019.

[4] Rackspace, https://www.rackspace.com/, 2019.

[5] Aws, https://aws.amazon.com/cn/ec2/pricing/reserved-instances/, 2019.

[6] S. Mireslami, L. Rakai, M. Wang et al., "Minimizing deployment cost of cloud-based web application with guaranteed QoS," in *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, San Diego, CA, USA, December 2014.

[7] S. Imai, S. Patterson, and C. A. Varela, "Cost-Efficient elastic stream processing using application-agnostic performance prediction," in *Proceedings of the IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pp. 604–607, Cartagena, CO, USA, May 2016.

[8] M. Anastasopoulos, A. Tzanakaki, and D. Simeonidou, "Stochastic energy efficient cloud service provisioning deploying renewable energy sources," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3927–3940, 2016.

[9] L. Chen, X. Li, and R. Ruiz, "Resource renting for periodical cloud workflow applications," *IEEE Transactions on Services Computing*, vol. 13, no. 1, pp. 130–143, 2020.

[10] S. Mireslami, L. Rakai, B. H. Far, and M. Wang, "Simultaneous cost and QoS optimization for cloud resource allocation," *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 676–689, 2017.

[11] H. Zhao, M. Pan, X. Liu, X. Li, and Y. Fang, "Exploring fine-grained resource rental planning in cloud computing," *IEEE Transactions on Cloud Computing*, vol. 3, no. 3, pp. 304–317, 2015.

[12] Y. Ran, B. Yang, W. Cai et al., "Cost-Efficient provisioning strategy for multiple services in distributed clouds," in *Proceedings of the International Conference on Cloud Computing Research & Innovations (ICCCRI)*, Singapore, May 2016.

[13] A.-F. Antonescu and T. Braun, "Simulation of SLA-based VM-scaling algorithms for cloud-distributed applications," *Future Generation Computer Systems*, vol. 54, pp. 260–273, 2016.

[14] B. Sniezynski, P. Nawrocki, M. Wilk, M. Jarzab, and K. Zielinski, "VM reservation plan adaptation using machine learning in cloud computing," *Journal of Grid Computing*, vol. 17, no. 4, pp. 797–812, 2019.

[15] D. Candeia, R. A. Santos, and R. Lopes, "Business-driven long-term capacity planning for SaaS applications," *IEEE Transactions on Cloud Computing*, vol. 3, no. 3, pp. 290–303, 2015.

[16] A. Wolke, M. Bichler, and T. Setzer, "Planning vs. Dynamic control: resource allocation in corporate clouds," *IEEE Transactions on Cloud Computing*, vol. 4, no. 3, pp. 322–335, 2016.

[17] J. Chen, H. Zhou, and W. Wang, "A provision algorithm for cloud resources with cost optimization," *Journal of Xi'an Jiaotong University*, vol. 51, no. 10, pp. 135–141, 2017.

[18] S. Mireslami, L. Rakai, M. Wang, and B. H. Far, "Dynamic cloud resource allocation considering demand uncertainty," *IEEE Transactions on Cloud Computing*, vol. 51, p. 1, 2019.

[19] Y. Lei and H. Shen, "Towards bandwidth guarantee for virtual clusters under demand uncertainty in multi-tenant clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 2, pp. 450–465, 2018.

[20] Z. Cao, J. Lin, C. Wan et al., "Optimal cloud computing resource allocation for demand side management in smart grid," *IEEE Transactions on Smart Grid*, vol. 8, no. 4, pp. 1943–1955, 2017.

[21] R. N. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, "Workload prediction using ARIMA model and its impact on cloud applications' QoS," *IEEE Transactions on Cloud Computing*, vol. 3, no. 4, pp. 449–458, 2015.

[22] R. V. D. Bossche, K. Vanmechelen, and J. Broeckhove, "IaaS reserved contract procurement optimisation with load prediction," *Future Generation Computer Systems*, vol. 53, pp. 13–24, 2015.

[23] L. Zhao, "Load forecasting model of cloud computing resources based on support vector machine," *Journal of Nanjing University of Science and Technology*, vol. 42, no. 6, pp. 687–692, 2018.

[24] Z. Zhang, S. Ding, and Y. Sun, "A support vector regression model hybridized with chaotic krill herd algorithm and empirical mode decomposition for regression task," *Neurocomputing*, vol. 410, no. 14, pp. 184–201, 2020.

[25] W.-C. Hong, Y. Dong, C.-Y. Lai, L.-Y. Chen, and S.-Y. Wei, "SVR with hybrid chaotic immune algorithm for seasonal load demand forecasting," *Energies*, vol. 4, no. 6, pp. 960–977, 2011.

[26] A. Ma, C. Zhang, B. Zhang et al., "Load prediction approach for cloud application based on Deep Belief networks," *Journal of Northeastern University(Natural Science)*, vol. 38, no. 2, pp. 209–213, 2017.

[27] K. Cetinski and M. B. Juric, "AME-WPC: advanced model for efficient workload prediction in the cloud," *Journal of Network and Computer Applications*, vol. 55, pp. 191–201, 2015.

[28] J. Kumar, R. Goomer, and A. K. Singh, "Long short term memory recurrent neural network (lstm-rnn) based workload forecasting model for cloud datacenters," *Procedia Computer Science*, vol. 125, pp. 676–682, 2018.

[29] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang, "Long short-term memory neural network for traffic speed prediction using remote microwave sensor data," *Transportation Research Part C: Emerging Technologies*, vol. 54, pp. 187–197, 2015.

[30] X. Wang, J. Wu, C. Liu et al., "Fault time series prediction based on LSTM recurrent neural network," *Journal of Beijing University of Aeronautics and Astronautics*, vol. 44, no. 4, pp. 772–784, 2018.

[31] A. Graves, "Long short-term memory," *Studies in Computational Intelligence*, vol. 385, 2012.

[32] Y. Tian, K. Zhang, J. Li, X. Lin, and B. Yang, "LSTM-based traffic flow prediction with missing data," *Neurocomputing*, vol. 318, no. 27, pp. 297–305, 2018.

[33] Y. Liu, S. Dong, M. Lu et al., "LSTM based reserve prediction for bank outlets," *Tsinghua Science and Technology*, vol. 24, no. 1, pp. 77–85, 2018.

[34] Q. Wang, Y. Guo, L. Yu et al., "Earthquake prediction based on spatio-temporal data mining: an LSTM network approach," *IEEE Transactions on Emerging Topics in Computing*, vol. 8, no. 1, pp. 148–158, 2017.

[35] N. Sfika, A. Korfiati, C. Alexakos et al., "Dynamic cloud resources allocation on multidomain/multiphysics problems," in *Proceedings of the International Conference on Future Internet of Things and Cloud*, pp. 31–37, Rome, Italy, August 2015.

[36] S. Khatua, P. K. Sur, R. K. Das, and N. Mukherjee, "Heuristic-based resource reservation strategies for public cloud," *IEEE Transactions on Cloud Computing*, vol. 4, no. 4, pp. 392–401, 2016.

[37] R. I. Meneguette, A. Boukerche, A. H. M. Pimenta et al., "A resource allocation scheme based on Semi-Markov Decision Process for dynamic vehicular clouds," in *Proceedings of the IEEE International Conference on Communications (ICC)*, pp. 1–6, Paris, France, May 2017.

[38] L. Jiao, J. Li, T. Xu, W. Du, and X. Fu, "Optimizing cost for online social networks on geo-distributed clouds," *IEEE/ACM Transactions on Networking*, vol. 24, no. 1, pp. 99–112, 2016.

[39] S. Stijven, R. Van den Bossche, E. Vladislavleva et al., "Optimizing a cloud contract portfolio using genetic programming-based load models," *Genetic Programming Theory and Practice XI*, vol. 22, pp. 47–63, 2014.

[40] E. Dhib, N. Zangar, N. Tabbane et al., "Impact of seasonal ARIMA workload prediction model on QoE for massively multiplayers online gaming," in *Proceedings of the 5th International Conference on Multimedia Computing and Systems (ICMCS)*, pp. 737–741, Marrakech, Morocco, September 2016.

[41] P. Singh, P. Gupta, and K. Jyoti, "Tasm: technocrat arima and svr model for workload prediction of web applications in cloud," *Cluster Computing*, vol. 22, no. 2, pp. 619–633, 2019.

[42] N. Tran, T. Nguyen, B. M. Nguyen, and G. Nguyen, "A multivariate fuzzy time series resource forecast model for clouds using LSTM and data correlation analysis," *Procedia Computer Science*, vol. 126, pp. 636–645, 2018.

[43] Alibaba, https://www.taobao.com, 2020.

[44] Huawei, https://lab.huaweicloud.com, 2020.

[45] Chinese Academy of Sciences Innovative Center in Quantum Information and Quantum Physics, http://quantumcomputer.ac.cn, 2020.

[46] Alibaba, https://www.aliyun.com/product/rsimganalys, 2020.

[47] Tableau, https://www.tableau.com/about/mission, 2020.

[48] C. Lu, *Queueing Theory*, Beijing University of Posts and Telecommunications Press, Beijing, China, Second edition, 2009.

[49] Lindo, https://www.lindo.com, 2020.

[50] Lacity.org, https://www.kaggle.com/cityofLA/lacity.org-websitetraffic, 2019.

[51] Yoochoose, https://www.kaggle.com/jacobgreen4477/weblog, 2020.

[52] C. Cortes and V. Vapnik, "Support vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[53] G.-F. Fan, S. Qing, H. Wang, W.-C. Hong, and H.-J. Li, "Support vector regression model based on empirical mode decomposition and auto regression for electric load forecasting," *Energies*, vol. 6, no. 4, pp. 1887–1901, 2013.

[54] Y. H. Chen, W. C. Hong, W. Shen et al., "Electric load forecasting based on a least squares support vector machine with fuzzy time series and global harmony search algorithm," *Energies*, vol. 9, no. 2, 2016.

[55] Y. Dong, Z. Zhang, and W. C. Hong, "A hybrid seasonal mechanism with a chaotic cuckoo search algorithm with a support vector regression model for electric load forecasting," *Energies*, vol. 11, no. 4, 2018.

[56] Z. Zhang and W.-C. Hong, "Electric load forecasting by complete ensemble empirical mode decomposition adaptive noise and support vector regression with quantum-based dragonfly algorithm," *Nonlinear Dynamics*, vol. 98, no. 2, pp. 1107–1136, 2019.

[57] Z. Zhang, W.-C. Hong, and J. Li, "Electric load forecasting by hybrid self-recurrent support vector regression model with variational mode decomposition and improved cuckoo search algorithm," *IEEE Access*, vol. 8, pp. 14642–14658, 2020.