Future Generation Computer Systems 117 (2021) 338–358

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Online cloud resource prediction via scalable window waveform sampling on classified workloads

Xiaogang Wang [a,e,*], Jian Cao [b], Dingyu Yang [c], Zhen Qin [d], Rajkumar Buyya [e]

[a] School of Electronic and Information, Shanghai Dianji University, Pudong New Area District, Shanghai, China
[b] Department of Computer Science and Engineering, Shanghai Jiao Tong University, Minghang District, Shanghai, China
[c] Alibaba Group, 969 Wenyi Road, Hangzhou, China
[d] Shanghai Servinet Technology Co., Ltd, 305 Liu Ying Road, Jingan District, Shanghai, China
[e] CLOUDS Laboratory, School of Computing and Information Systems, The University of Melbourne, Parkville, Melbourne, Australia

ARTICLE INFO

Article history:
Received 28 April 2020
Received in revised form 15 September 2020
Accepted 8 December 2020
Available online 13 December 2020

Keywords:
Cloud computing
Resource prediction
Trend degree
Workload waveform sampling
Optimal error gradient boosting regression

ABSTRACT

Accurate prediction on the utilization of cloud resources is increasingly important for public cloud users, as it relates to the reasonable reservation of resources for minimizing the usage costs. However, the existing relevant approaches fail to predict the usage amount of cloud resources on the basis of the requested workloads of users' applications, and the characteristics of changing workload data are rarely considered for the real-time prediction. To address these challenges, we propose an online cloud resource prediction model (OCRPM) to timely predict the proper resource usage amount. Firstly, all of the requested workloads are classified into three types of waveform trend patterns using the trend degree (TD). Next, a scalable window waveform sampling method (SWWS) on the classified patterns is devised to extend the suitable workload waveform interval window for supporting the subsequent high accurate prediction on the cloud resources. Finally, an optimal error gradient boosting regression (OEGBR) algorithm is given to train the data model and to predict the reasonable cloud resource usage amount in light of the requested workloads. The simulation results indicate that the proposed method can adjust the suitable workload waveform sampling window, and achieve higher prediction accuracy than the state-of-the-art relevant approaches and existing statistical learning models.

1. Introduction

Increasing enterprise applications and scientific computing tasks are deployed in public cloud systems such as Amazon AWS, Windows Azure, Aliyun and Google Cloud. Cloud service providers allow users to lease a certain amount of cloud resources or virtual machines (VMs) when needed [1] while satisfying specific service level agreements [2], and the users pay the cloud providers on a pay-as-you-go mode. Due to the diversity of types, cost effectiveness and operational reliability of cloud services provisioned by these cloud providers, users are more willing to reserve suitable cloud resources to meet their varied business application demands. However, it is quite difficult for users to accurately reserve how much cloud resources (e.g., CPU cores, Memory size, Disk size and Bandwidth) are needed for their requested workloads (i.e., the task processes that constitute the running applications of users in this paper). Because when the reserved cloud resources exceed the actual ones used, the resources will be wasted or in a state of inefficient utilization. Conversely, if the reserved cloud resources are insufficient, the execution of task workloads will be delayed or suspended. In addition, the workload fluctuation in cloud systems may also result in the over-provisioning or under-provisioning situation of resources [3,4]. The former situation will bring economic losses due to the higher usage cost of cloud resources (i.e., the more payment for users, and the greater energy waste of cloud data centers for cloud providers), the latter one may delay or even interrupt the execution of user tasks. As the reports show, the hourly mean CPU utilization in Google cloud cluster-usage trace [5] and Aliyun trace [6] is only of 25 to 35%. Therefore, for improving the resource utilization and reducing the usage cost while guaranteeing the cloud service performance desired by users, a timely and efficiently cloud resource prediction method needs to be devised to determine the reasonable resource quantity that should be reserved according to the historical and current resource consumption.

However, accurately predicting the usage amount of cloud resources for requested workloads is still confronted with the following challenges:

* Corresponding author at: School of Electronic and Information, Shanghai Dianji University, Pudong New Area District, Shanghai, China.
E-mail addresses: wangxg@sdju.edu.cn (X. Wang), cao-jian@cs.sjtu.edu.cn (J. Cao), dingyu.ydy@alibaba-inc.com (D. Yang), zhen.qin@servinet.cn (Z. Qin), rbuyya@unimelb.edu.au (R. Buyya).

https://doi.org/10.1016/j.future.2020.12.005
0167-739X/

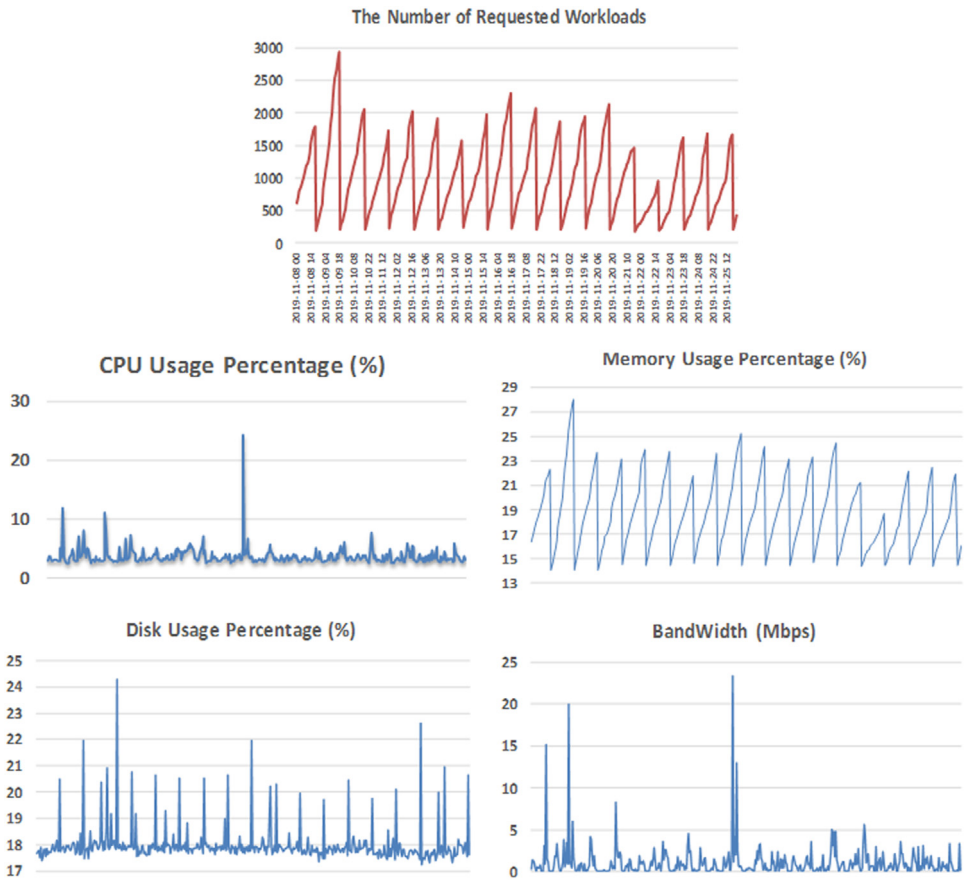

Fig. 1. The number of relatively-periodic requested workloads and corresponding usage amount of cloud resources.

- **Constantly changing data of requested workloads and cloud resource usage.** The number of requested workloads varies with time, and some are relatively periodic (e.g., the general user Web applications [7] running in the public cloud system), while others are random (e.g., the randomly-changing workloads from the cluster-usage traces in Google cloud data centers [5]). The usage amount of each type of cloud resources, such as the average usage percentage of CPU, Memory and Disk, and the average number of Bandwidth, fluctuates in a certain degree for the relatively-periodic workloads similar to Fig. 1, or considerably changes for the randomly-changing ones during a day [5,6] like Fig. 2. These cases reflect that the accurate prediction of the cloud resource usage amount with the continuously changing requested workloads becomes so hard.
- **Nonlinear relationship between requested workloads and corresponding cloud resource usage.** Regardless of whether requested workloads are relatively periodic or randomly changing, due to the nonlinear relationship between requested workloads and corresponding usage amount of cloud resources, it is not easy to establish the mapping relationship between them through employing historical and current data. However, the establishment of this relationship is the key to the cloud resource prediction.
- **Time-lagging cloud resource prediction.** For predicting the cloud resource usage amount, it needs to continually generate request workloads and to survey the changes on the related resource usage in the existing practice. However, during these processes, much time can be taken to wait for multi-round measurement, and the data of the workloads and resource usage cannot be immediately obtained. It is unacceptable for time-sensitive users to reserve the cloud resources with the time-lagging prediction.

In the face of these problems, the existing approaches do not analyze the essential characteristics of workload waveforms to find the intrinsic rules of those workload changes. The data relationship between requested workloads and related cloud resource usage is not also further established. In the aspect of the timely cloud resource prediction, there are no existing methods to carry out the online cloud resource prediction through quickly fetching the real-time cloud resource monitoring data.

Different from the current approaches, we consider the consecutive fluctuation of users' requested task workloads with the important waveform features, further perform the scalable window waveform sampling on the classified workloads, and finally generate the nonlinear predictable relationship between the requested workloads and their usage amount of cloud resources by training the data model of requested workloads with the corresponding usage amount of cloud resources (called as resource usage labels). Integrating these new solutions can make the fine-grained prediction of cloud resource usage on the basis of continuous changing waveform features of requested workloads, and achieve the online cloud resource prediction with the higher accuracy.

The latest approaches [8–10] only predict the future trends of users' workloads or demands in the volatile business requirements for making decisions to adjust the cloud resource allocation instead of establishing the relationship between the business workloads and the cloud resource usage. On the other hand, some prediction methods [3,11–13] predict the future cloud workload trends through the historical CPU utilization or usage information used by the different models or algorithms such as the deep

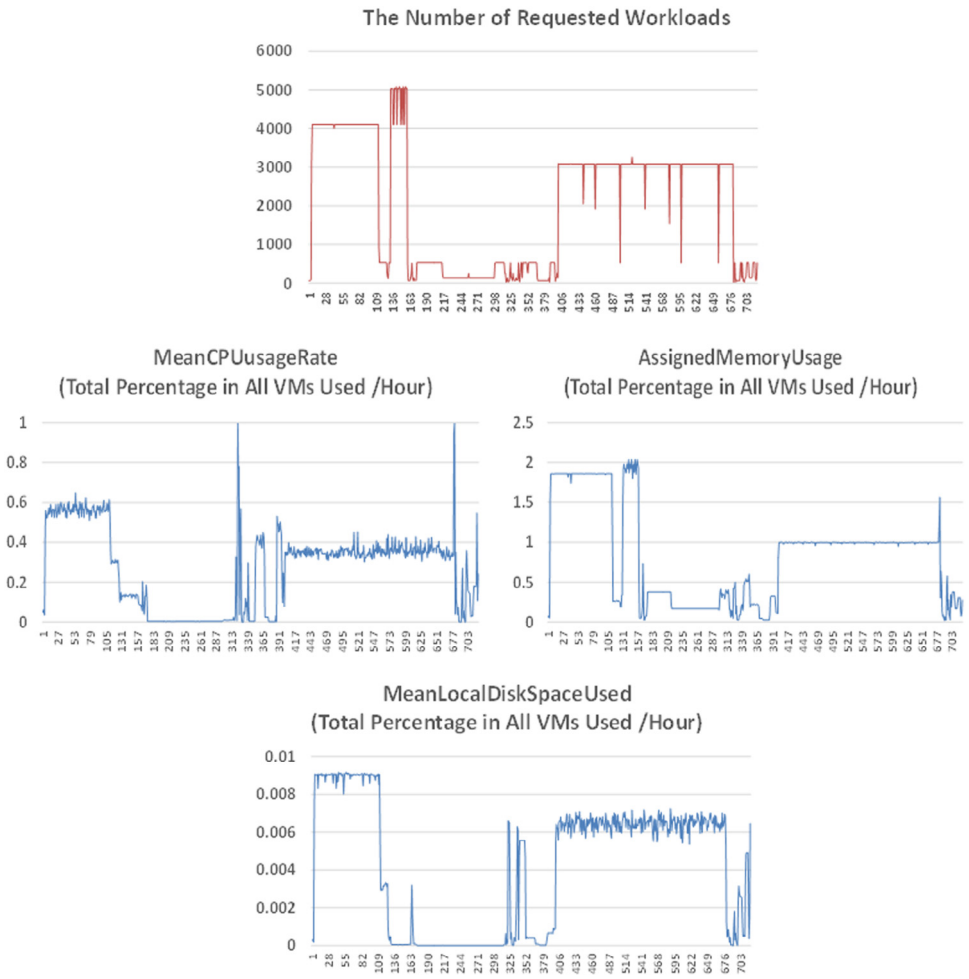

Fig. 2. The number of randomly-changing requested workloads and corresponding usage amount of cloud resources.

learning, time series, swarm and evolution, and ensemble-based regression respectively. However, these methods reversely infer the changes of workloads based on cloud resource usage, without paying attention to the changing characteristics of requested workloads from users' applications. Unlike the above methods, we should find the intrinsic relevance between constantly changing requested workloads and corresponding cloud resource usage. In the cloud resource allocation based on the workload prediction, some researches [14,15] propose the skewness-avoidance (balanced) cloud resource allocation in physical machines (PMs) according to the diversified requirement workloads predicted in advance, which relates the different types of cloud resource configurations with the performance of cloud provisioning. However, their work does not also form the relationship between requested workloads and related cloud resource usage.

In the just-in-time prediction, the existing methods via the post-historical data analysis [16,17], such as generating different users' workloads and profiling the performance of cloud systems running the applications, do not use a timely manner to predict the cloud resource consumption of users' applications. When the users' demand patterns change, new round of profiling jobs must restart. However, users cannot wait for a long time of the profiling process, on the contrary, they expect to know as soon as possible about the cloud resource usage amount for their applications so that the resource reservation can be adjusted quickly to save money.

To address the above challenges, we put forward an online cloud resource prediction model (OCRPM), which encompasses the requested workload trend classification (RWTC), the scalable window waveform sampling (SWWS) on classified workloads, and the optimal error gradient boosting regression (OEGBR) training and prediction. We firstly obtain the data of requested workloads and cloud resource usage (e.g., CPU, Memory, Disk and Bandwidth) through online querying the cloud monitoring trace database from VMs, and further classify the requested workloads into three types of trend waveform patterns (Peak, Steadiness and Trough) in hourly time intervals. Next, with these classified data, a scalable window waveform sampling method is proposed to support the model training on the data of workloads and cloud resource usage. Finally, the OEGBR is designed to train the data model (i.e., establishing the mapping relationship between the requested workloads and corresponding cloud resource usage) and to predict the cloud resource usage amount by the test data. All of the above-mentioned modules and processes are implemented in the cloud service broker layer similar to our previous work [18,19].

The main contributions of this paper are summarized as follows:

- An OCRPM is proposed to timely and efficiently predict the reasonable resource usage amount that can be reserved in the future according to the historical and current resource consumption.
- A way of online querying and sampling cloud trace data from VMs is adopted. To differentiate the waveform features of requested workloads sampled, a measurement called the trend degree (TD) is presented, combining the skewness

and kurtosis of requested workloads in a certain time scale. Through using the feature TD, all the requested workloads are classified into three types of waveform trend patterns in hourly time intervals such as Peak, Steadiness and Trough.
- A SWWS method on the three types of waveform trend patterns is put forward to extend the suitable workload waveform interval window for supporting the model training with high accuracy on the sampled data of workloads and cloud resource usage, and its sampling algorithm is also designed.
- An OEGBR algorithm is devised to train the data model and to predict the reasonable cloud resource usage amount.
- Extensive simulation experiments, adopting the datasets of the real-world business applications of an IT company and the opening Google cloud cluster-usage trace, are conducted to evaluate the effectiveness of the proposed method.

The rest of the paper is organized as follows. Section 2 reviews the related work. In Section 3, we demonstrate the overall process of the proposed OCRPM. Section 4 describes the workload trend classification, the SWWS method and its sampling algorithm in detail. In Section 5, we give the OEGBR algorithm on the model training and prediction. Section 6 evaluates the proposed method and algorithms using the different cloud trace data. Finally, we draw the conclusions in Section 7.

2. Related work

In recent years, the cloud resource prediction in different cloud systems has received great attention, which mainly involves two aspects, the trend prediction on requested task workloads in cloud systems and the usage prediction on cloud resources running users' task workloads. We analyze the main characteristics of the existing methods and discuss their deficiency in the cloud resource usage prediction.

2.1. Requested workload trend prediction approaches

Many proactive cloud computing models [20–22] predicted the volume of requests called the arrival rate for users' applications hosted on cloud data-centers to auto-scale cloud resources. Using the request arrival rate as prediction information only partially captures the features of the workloads and changing systems. A more realistic proactive prediction of workload patterns was proposed [23]. It adds the volume of requests, analyzes the web application access logs to discover URI (Uniform Resource Identifier) space partitions based on the response time and the document size features, and uses URI's distribution across these partitions to compute the probabilistic workload pattern (PWP) for predicting the workload pattern of the next interval. Some classical methods, which use the time-series models such as ARMA [24], ARIMA [25], EWMA [26] and HMM [27] to predict data center workloads with the seasonal trends, are hard to capture the patterns in the non-obvious seasonal workloads.

Kumar et al. [8] developed a workload prediction model based on neural network and self adaptive differential evolution that can predict the workloads with higher result accuracy. Wang et al. [10] researched the trace logs of real datacenter workload arrival trend, and presented a new LSTMtsw based on recurrent neural networks LSTM (Long Short-Term Memory) to predict the future resource request trend of users. Panneerselvam et al. [28] proposed a novel prediction model named InOt-RePCoN which aims at a tri-fold forecast for predicting the expected number of job submissions in the incoming workloads. A novel classification and prediction framework on the changing demands was proposed in [9], which adopts Piecewise Linear Representation (PLR) to segment the changing time series of cloud resource demands, and the authors devised the cloud resource demand prediction as a weighted three-class (stable, peak, or trough) classification problem adopting Weighted Support Vector Machines (WSVM) that adds the extra weights for the abrupt changing number of requests.

Although the above approaches use different models respectively to achieve the better predictive goals for requested workloads, these researches do not simultaneously consider both the periodic and random fluctuation of the workloads. Also, these models prefer to make prediction only for the trend of the requested workloads or demands from users so as to prepare for the next allocation or reservation of cloud resources. However, we should pay close attention to both the users' requested workloads and their corresponding cloud resource usage instead of focusing on unilateral fluctuation in requested workloads, because the requested workloads can greatly impact on the consumption of cloud resources.

2.2. Cloud resource usage prediction methods

A lot of classic methods adopt the regression and artificial neural network (ANN) models to predict the cloud resource usage. Hu et al. [11] used an auto-regression (AR) technology to predict the future cloud resource workloads in which the time-series based historical CPU utilization is employed. An ensemble-based method [13] to predict the CPU usage of scientific applications was presented, which considers the average accuracy of eight regression-based prediction models. These regression models generally deal with the seasonal trends with a certain limitation on cloud resource usage. Chen et al. [3] designed a top-sparse auto-encoder (TSA) and an efficient deep Learning based algorithm to predict the cloud CPU, Memory and Disk I/O usage with high-dimensional and highly-variable cloud resource workloads. An evolutionary neural networks model [12] was adopted to predict the host CPU utilization. Tang et al. [29] used the linear regression (LR) and wavelet neural network (WNN) to predict the short-term cloud resource workloads. But these ANN methods either suffer from high prediction errors under the long-term case or have the difficulty in selecting training parameters.

Qiu et al. [30,31] put forward a probabilistic demand allocation (PDA) system to solve the demand allocation problem for the cloud service brokerage (CSB). This system not only predicts tenants' CPU and Memory demands according to their historical records, but also estimates the probability distribution about the prediction errors. A workload aware resource allocation mechanism for containerized online services was presented in [32] which consists of workload predictor, resource reservation and online controller. The workload predictor can precisely predict the allocated CPU cores in the periodic CPU utilization by a LSTM network, and the prediction results serve for cloud resource reservation and online controller. However, these methods focus on making the prediction on the cloud resource usage according to the historical usage, and do not correlate the resource usage with the requested task workloads.

Some recent work [33,34] proposed the cloud resource reservation and provisioning system for the predictable resource usage of the big data analysis, which is a rack-level coalition formation mechanism and an accurate LSTM-based price prediction method, respectively. Recent methods [16,17] used the profiling cloud resource and performance to predict the usage amount. Marques et al. [35] introduced a novel model called Escada to predict the network bandwidth for guiding the provisioning of VMs through a VM workload profile. Whereas Hauser et al. [36] presented an approach to monitor resource statistics on the physical level only, and provided the profiles of CPU cores and its changing utilization

to cloud middleware and customers for decision making. However, in terms of reserving cloud resources for users, there is a time-lagging problem that can result in some economic losses in the time-limited applications.

The above methods mainly concern with the usage prediction of cloud resources about the CPU utilization based on the cloud historic usage trace, so that the cloud systems can improve the provisioning amount of resources. However, we should simultaneously take the fluctuation of users' requested workloads into consideration, acquire the nonlinear relationship between the requested workloads and the usage amount of cloud resources by a way of model training on the real-time sampling data, and eventually realize the online cloud resource prediction.

In addition, there are several new methods proposed in recent years to predict the cloud resource (CPU) utilization. Liu et al. [37] presented an adaptive categorical workload prediction method that categorizes the workloads and selects the Linear Regression or the Support Vector Machines to predict CPU utilization of VMs. However, this method only uses the slow and fast time-scale workload data feature instead of considering the continuous changing features of task workloads. Baig et al. [38] proposed a novel approach to adaptively determine the best machine learning method to predict future CPU utilization. Even though this method extracts some time-series features on the CPU utilization and adaptively selects a suitable machine learning method to realize the prediction, however, the features of the task workloads are not extracted, and the relationship between the workloads and the cloud resource usage amount is also not mentioned.

3. Online cloud resource prediction model

On account of the periodically or randomly changing trends of requested workloads and cloud resource usage, it is difficult for users to make effective resource reservation decisions quickly, and the similar case is true for cloud providers to provision the cloud resources. The over-reservation of cloud resources can lead to unnecessary reservation cost and energy waste of cloud resources while the under-reservation may delay or impede the execution of users' applications. In response to these problems, we propose an online cloud resource prediction model (OCRPM). In this section, the system framework implementing OCRPM is briefly introduced. On this basis, we describe the overall OCRPM's model and process which are the focus of this paper.

3.1. System framework implementing OCRPM

The proposed OCRPM is implemented in a cloud system framework shown in Fig. 3. The framework has three layers: the PM Resource Network layer, the VM Resource Provisioning layer and the Cloud Service Broker layer.

The PM Resource Network layer builds the physical cloud infrastructure composed of many local or cross-regional networked physical machines (PMs) that contain different types of abundant cloud resources such as CPUs, RAMs (Memory), Disks, and Bandwidth (BW) of Network. The VM Resource Provisioning layer consists of many different sizes of virtual machine (VM) bundles reserved from a large pool of virtual machines. A bundle of VMs may include one or more types of VMs. Each VM contains different number of virtualized cloud resources from the PM Resource Network layer. The cloud resources in the above two layers are provisioned by the cloud service providers of IaaSs.

The Web business applications of users are deployed into one or more VM bundles reserved in advance, and then converted to the task processes (called requested workloads) running in the VM bundles. The application systems can receive the requests of task execution from cloud users in any time. In addition, Monitor Agents, which are deployed in the proxy monitor nodes on the cloud VMs reserved, collect the trace data of workloads and cloud resource usage from different VM bundles running the users' workloads, and send these data to the VM Resource Data Collection Server.

The Cloud Service Broker (CSB) layer comprises VM Resource Data Collection Server, VM Resource Usage Trace Database, the Online Cloud Resource Prediction module and the Cloud Resource Reservation module. VM Resource Data Collection Server, which is installed in the CSB layer, receives the trace data of workloads and cloud resource usage from the Monitor Agents, converges these data and stores them into VM Resource Usage Trace Database. This trace database obtains the real-time trace data, and writes them into different data tables. The Online Cloud Resource Prediction module is the key component in this paper, which mainly includes three processes: RWTC, SWWS and OEGBR. This prediction module fetches the trace data of workloads and corresponding cloud resource usage that need to be calculated, then carries out the process of prediction, and finally outputs the cloud resource prediction results. More details about the OCRPM will be expounded in the following sections. The Cloud Resource Reservation module can accept the new resource reservation from cloud users, and may also help users quickly modify the reservation amount according to the real-time prediction results.

3.2. Overall model and process of OCRPM

The whole model of the proposed OCRPM is shown in Fig. 4, and its overall process mainly involves four processing steps in sequence, each of which contains different method modelings or algorithm designs that are briefly described as data acquisition and feature extraction, requested workload trend classification, scalable window waveform sampling, and data model training and prediction.

Data acquisition and feature extraction. The OCRPM firstly online fetches the raw trace data with hourly time series from the VM Resource Usage Trace Database. The trace data includes the number of requested workloads, the usage percentage (utilization) of CPU, Memory and Disk, and the usage value (/Mbps) of Bandwidth. Next, the model extracts the four-dimensional statistical feature data of the requested workloads on the scale of hour as the input data, where the trend degree (TD) of these workloads in an initial sampling time window are calculated as an important digital feature in the four-dimensional data. Finally, the model separately labels the usage value of CPU, Memory, Disk and Bandwidth in the hourly intervals as the output data corresponding to the hourly requested workloads. In addition, the reason for using the hourly intervals is that the cloud resources are generally reserved and used on pay-per-hour mode, and the trace data of requested workloads and cloud resource usage are aggregated in hourly time-series intervals in current well-known public cloud systems.

Requested workload trend classification. According to the waveform feature data of hourly requested workloads, the OCRPM uses the proposed *TD* method to classify all of the requested workloads into three types of waveform trends in a certain sampling time window such as Peak, Steadiness and Trough, which denote sharply increasing to the highest points, gently gradient or flat points, and sharply decreasing to the lowest points, respectively. As shown in the classification chart of the top right side of Fig. 4, the left, middle and right parts are the waveform trends of Peak, Steadiness and Trough in sequence.

Scalable window waveform sampling. The OCRPM analyzes the waveform patterns of the requested workloads in light of the three types of waveform trends in the continuous hourly time-series data. And then, it uses the scalable window waveform

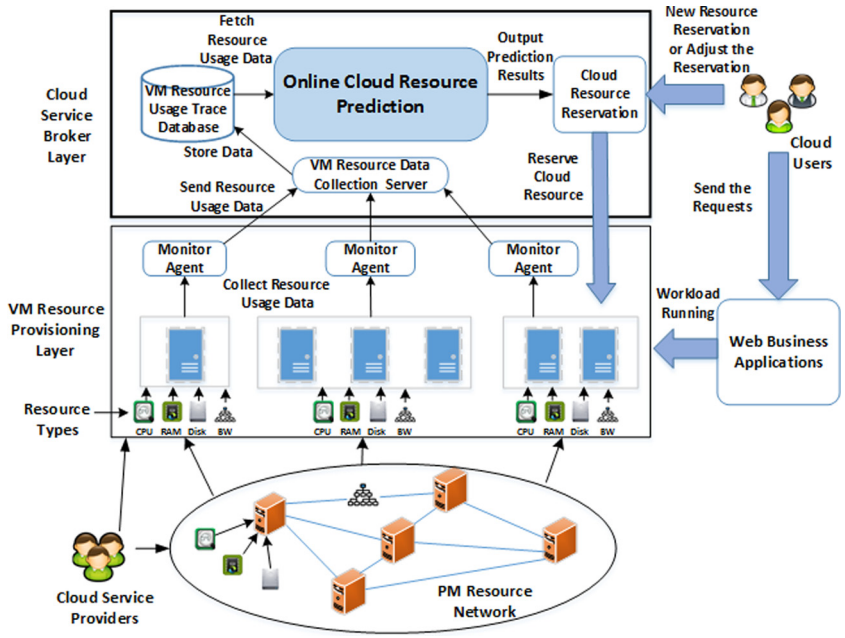

Fig. 3. The system framework implementing OCRPM.

sampling method (SWWS) to dynamically extend the suitable workload waveform interval window for performing the model training with high accuracy on these sampled data. The middle chart of the right side of Fig. 4 shows that the continuously changing requested workloads in a month have the time-varying waveforms which are sampled by means of extending or reducing the statistical interval windows during training the data model.

Data model training and prediction. After the workload waveform sampling, the OCRPM uses the OEGBR algorithm to train the data model about the requested workloads with corresponding cloud resource usage label data, and to predict the usage amount of the cloud resources by the test data. Afterwards, it evaluates the prediction error of the trained data model, namely, if the error is increased in comparison with the predicted results using the last waveform window, the OEGBR algorithm reduces the waveform sampling window for guaranteeing the lower prediction error. Otherwise, the algorithm outputs the predicted results about the usage amount of CPU, Memory, Disk and Bandwidth related with the number of requested workloads on the hourly time scale.

4. Workload trend classification and the SWWS method

This section presents the detailed formulation about the proposed feature extraction, waveform trend classification and scalable window waveform sampling on requested workloads labeled respective cloud resource usage amount. For ease of reading, the notations used in this paper are summarized in Table 1.

Table 1
Notations used in this paper.

Notation	Description
X	Input space
x_i	The sample data of requested workloads in the ith hour defined as a four-dimensional feature vector
Y	Output space
y_i	The usage amount of cloud resources in the ith hour
φ_i^{rw}	The number of requested workloads in the ith hour, which is the value of first feature $x_i^{(1)}$
$\Delta\tau$	A workload waveform sampling window
$TD(\varphi^{rw}\|_i^{i+\Delta\tau})$	The trend degree combining the skewness and kurtosis of requested workloads during a sampling window $\Delta\tau$ from an hourly time point i to $i+\Delta\tau$, which is the value of second feature $x_i^{(2)}$
D_i	The variance of requested workloads' number φ_i^{rw}, which is the value of third feature $x_i^{(3)}$
N_i^{cr}	The number of CPU cores of VMs reserved by users in the ith hour, which is the value of fourth feature $x_i^{(4)}$
ξ^+	Peak waveform trend of requested workloads
ξ°	Steadiness waveform trend of requested workloads
ξ^-	Trough waveform trend of requested workloads
T	Training dataset
$f(\hat{X})$	The prediction model
$L(y, f(\hat{x}))$	The squared error loss function
R_j	The subregion divided by the segmentation point s in the gradient boosting regression model, and $j = 1, 2$
N_j	The number of sample data on the subregion R_j
$\hat{c}_j$	The optimal mean value of the output y_i corresponding to all input samples x_i on the subregion R_j

4.1. Feature extraction and trend classification

The proposed OCRPM firstly fetches the number of requested workloads and respective usage amount of cloud resources (i.e., CPU, Memory, Disk and Bandwidth) in hourly time-series intervals through using the SQL statements to online query the VM Resource Usage Trace Database during a given period of time. After that, the model carries out the feature extraction from the number of workloads acquired. The processes of the data feature extraction and formulation are described as follows.

Since only most related and important features that improve the performance of each proposed model should be extracted according to [39,40], for decreasing the complexity of model training and reducing the overfitting of prediction results, we extract and reconstruct several important features of requested workload data in hourly time-series intervals such as the number, the skewness and kurtosis, and the sample variance, which reflect the size, the state of waveform and the degree of deviation on requested workloads, respectively.

The data of requested workloads in the ith hour is defined as a four-dimensional feature vector $x_i = (x_i^{(1)}, x_i^{(2)}, x_i^{(3)}, x_i^{(4)})$, where $x_i \in X \subseteq \mathbf{R}^N$, X is the input space, and $\mathbf{R}^N$ denotes N dimensional Euclidean space. The usage amount of cloud resources in the ith hour is represented as $y_i = \{y_i^{cpu}, y_i^{mem}, y_i^{disk}, y_i^{bw}\}$, where $y_i \in$

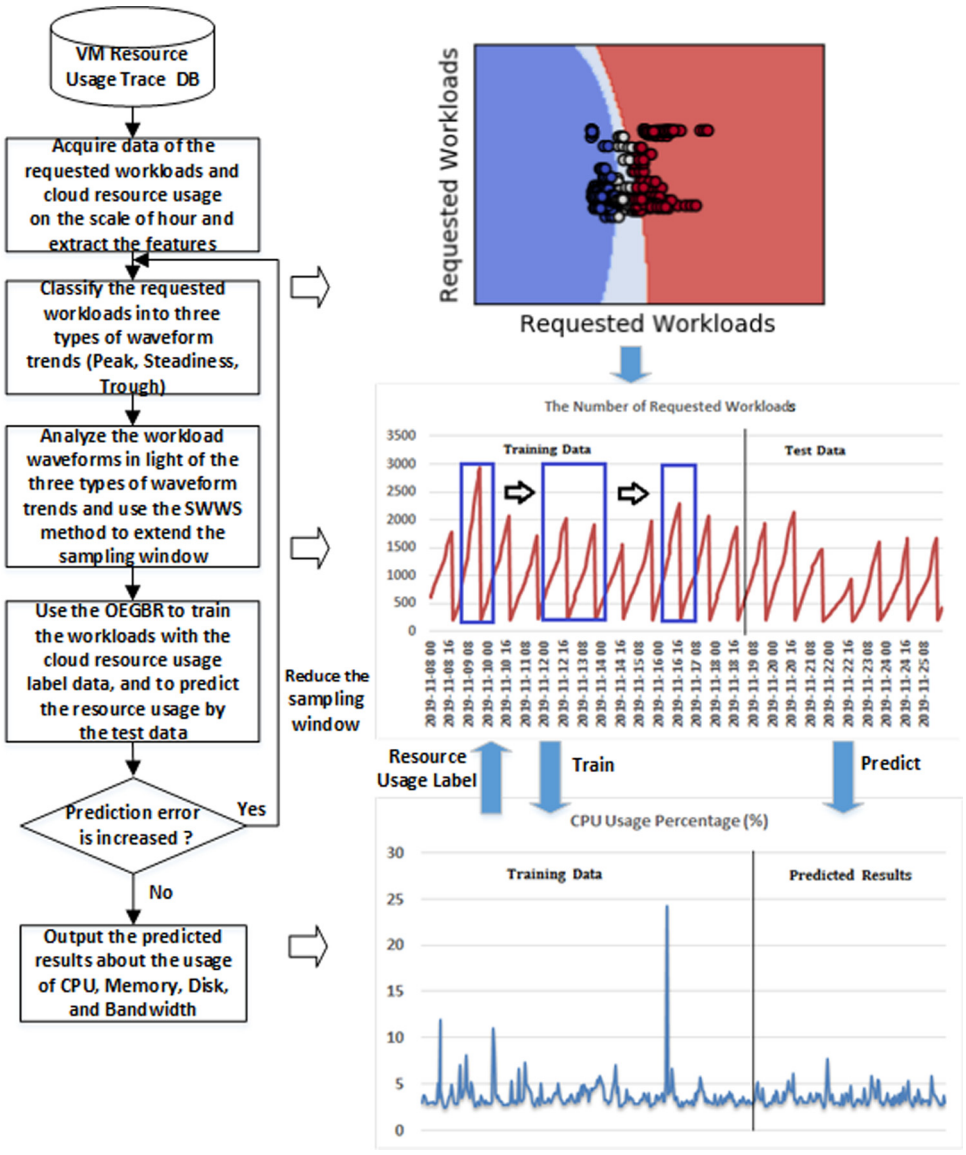

Fig. 4. The online cloud resource prediction model (OCRPM).

$Y \subseteq \mathbf{R}$, the elements in y_i contain the usage amount of CPU, Memory, Disk and Bandwidth in the ith hour, Y is the output space, and i is a sequential hourly time point. A training dataset is denoted as $T = \{(x_1, y_1), (x_2, y_2), \ldots, (x_i, y_i), \ldots, (x_n, y_n)\}$, where n is total hourly time points (i.e., maximum number of the input samples at the mode of one sample point per hour, and the same hereinafter) of the dataset. In the feature extraction of four-dimensional vector x_i, $x_i^{(1)}$ is assigned as the number φ_i^{rw} of requested workloads in the ith hour, and $x_i^{(2)}$ is expressed by the trend degree (TD) of requested workloads in a scalable sampling time window, which is a geometric synthesis formula using the skewness and kurtosis (i.e., the two digital features on the degree of distribution of sample data in statistics), concretely explained as follows.

The skewness, a measurement of the direction and degree of skewness in the distribution of sample data, is the feature value representing the degree of asymmetry of the probability distribution density curve relative to the mean, and its formula is $Skewness(X) = \mu_3/\sigma^3$, where μ_3 is the third order central moment, and σ is the standard deviation. When the probability distribution density curve of the sample data is right-skewed distribution (i.e, the tail on the right of the sample data curve is longer than the one on the left), $Skewness(X) > 0$, and the greater its value, the higher the right-skewed degree. Conversely, when it is left-skewed distribution (i.e, the tail on the left of the sample data curve is longer than the one on the right), $Skewness(X) < 0$, and the smaller its value, the higher the left-skewed degree. When the statistical data is symmetrically normal distribution, $Skewness(X) = 0$. The skewness curve is shown in Fig. 5.

The kurtosis is a feature measurement that describes the steepness of the distribution pattern of all values in sample data, and its formula denotes $Kurtosis(X) = \mu_4/\sigma^4 - 3$, where μ_4 is the fourth order central moment, and σ is also the standard deviation. This kurtosis has to be compared to a normal distribution. $Kurtosis(X) > 0$ indicates that the distribution pattern of the sample data is steeper than the normal distribution, and it is a sharp peak (leptokurtic). On the contrary, $Kurtosis(X) < 0$ means that the distribution pattern is relatively flat in comparison with the normal distribution, i.e., low peak (platykurtic). $Kurtosis(X) = 0$ shows that the distribution pattern of flat peak is as steep as the normal distribution. The kurtosis curve is shown in Fig. 6.

Accordingly, we propose a measurement of the trend degree (TD), i.e., a geometric synthesis formula combining the skewness and kurtosis of requested workloads during a waveform sampling

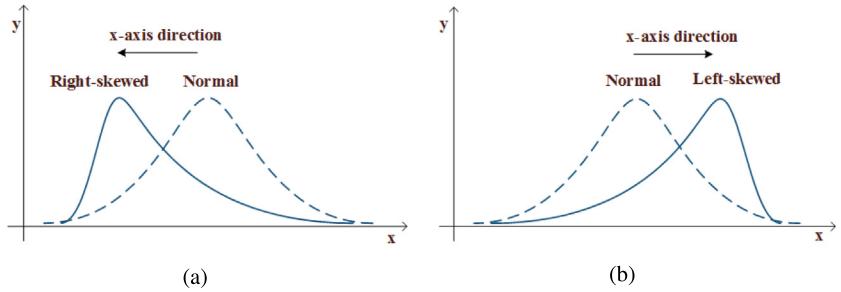

Fig. 5. The skewness curve has two types which are the right-skewed distribution (a) and the left-skewed one (b), and the virtual curves among them are all the symmetrically normal distribution. The shape in (a) is skewed to the left along the *x*-axis direction, and the one in (b) is skewed to the right along the *x*-axis direction.

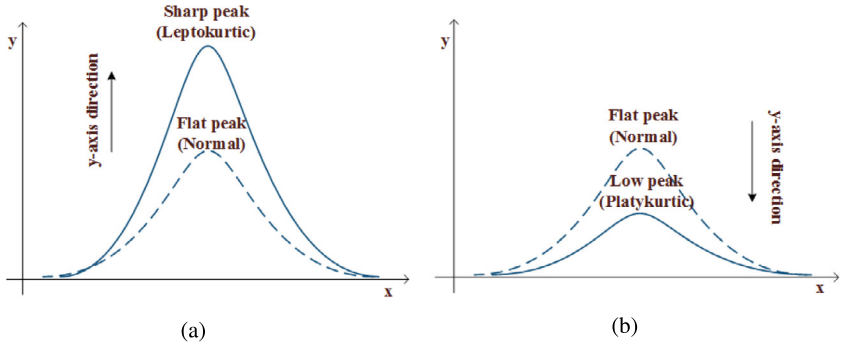

Fig. 6. The kurtosis curve has two types which are the sharp peak (leptokurtic) distribution (a) and the low peak (platykurtic) one (b), and the virtual curves among them are all the flat peak (normal) distribution. The shape in (a) goes up along the *y*-axis direction, and the one in (b) goes down along the *y*-axis direction.

window $\Delta\tau$ on the hourly time scale. The trend degree is defined as

$$TD(\varphi^{rw}|_i^{i+\Delta\tau}) = \sqrt{(Skewness(\varphi^{rw}|_i^{i+\Delta\tau}))^2 + (Kurtosis(\varphi^{rw}|_i^{i+\Delta\tau}))^2} \quad (1)$$

where $\varphi^{rw}|_i^{i+\Delta\tau}$ denotes all of the number of requested workloads from an hourly time point i to $i+\Delta\tau$.

To facilitate the understanding of the following scalable window waveform sampling method in the next section, we use a group of trend degree curves in Fig. 7 with different oblique angles to illustrate the meaning of the *TD* in Eq. (1). These trend degree curves involve the right-skewed and sharp peak (Fig. 7(a)), the left-skewed and sharp peak (Fig. 7(b)), the right-skewed and low peak (Fig. 7(c)), and the left-skewed and low peak (Fig. 7(d)) distribution patterns. They start from the respective normal origin (0, 0) (i.e., a kind of origin relative to the normal distribution) and move to their deflection points at the four different oblique angles in $\Delta\tau$ hour windows. According to the relative distance from the respective normal origin, we adopt the Euclidean distance to construct the respective trend degree $TD(\varphi^{rw}|_i^{i+\Delta\tau})$ of the four kinds of trend degree curves.

$x_i^{(3)}$ is expressed by the variance $D_i = (\varphi_i^{rw} - \overline{\varphi^{rw}})^2$ of requested workloads' number, where $\overline{\varphi^{rw}}$ denotes the sample mean of requested workloads. $x_i^{(4)}$ is filled by the number N_i^{cr} of CPU cores of VMs reserved by users in the ith hour. The reason for extracting the number of CPU cores is that it is an important system performance metric such as some recent related work [3,30,31].

To sum up, the four-dimensional feature vector x_i of requested workloads in each hour is specifically represented as $(\varphi_i^{rw}, TD(\varphi^{rw}|_i^{i+\Delta\tau}), D_i, N_i^{cr})$.

After the feature extraction, this system adopts the proposed *TD* method with the judgment of a workload waveform kurtosis to classify all of the requested workloads that have been featured. In a workload waveform sampling time window $\Delta\tau$, the proposed *TD* method classifies the requested workloads into three types of waveform trends, i.e., Peak (whose waveform trend is ξ^+), Steadiness (whose waveform trend is ξ°), and Trough (whose waveform trend is ξ^-), which mean sharply increasing to the highest points, gently gradient or flat points, and sharply decreasing to the lowest points, respectively. The values of three kinds of workload waveforms from the hourly time point i to $i+\Delta\tau$ are separately calculated by the conditional assignment equation Eq. (2).

$$\begin{cases} \xi_i^+ \sim \xi_{i+\Delta\tau}^+ \leftarrow TD(\varphi^{rw}|_i^{i+\Delta\tau}), & Kurtosis(\varphi^{rw}|_i^{i+\Delta\tau}) > 0 \\ \xi_i^\circ \sim \xi_{i+\Delta\tau}^\circ \leftarrow TD(\varphi^{rw}|_i^{i+\Delta\tau}), & Kurtosis(\varphi^{rw}|_i^{i+\Delta\tau}) = 0 \\ \xi_i^- \sim \xi_{i+\Delta\tau}^- \leftarrow -TD(\varphi^{rw}|_i^{i+\Delta\tau}), & Kurtosis(\varphi^{rw}|_i^{i+\Delta\tau}) < 0 \end{cases} \quad (2)$$

4.2. Scalable window waveform sampling

To accurately predict the cloud resource usage amount under the changing waveform trend of request workloads in the continuous hourly time series, we propose the scalable window waveform sampling method (SWWS) to dynamically extend the suitable workload waveform interval window. The waveform sampling results are utilized to the model training with high accuracy, and the detailed formulation of SWWS method is demonstrated as follows.

The workload waveform sampling window is devised as $\Delta\tau = 2^k$, whose value can be adjustable, and k is a positive integer greater than 1, where the binary exponential window extension is a commonly smooth timing scaling method similar to the sliding window technology. We stipulate that k cannot be 1, otherwise $\Delta\tau = 2$ makes the waveform trend degree *TD* uncomputable. Whether it is for relatively-periodic or randomly-changing workloads, one or two days of data can generally reflect the varied workload waveform trends. Thus, for reducing the possible prediction error, we set $k \in [2, 6)$, such that a sampling

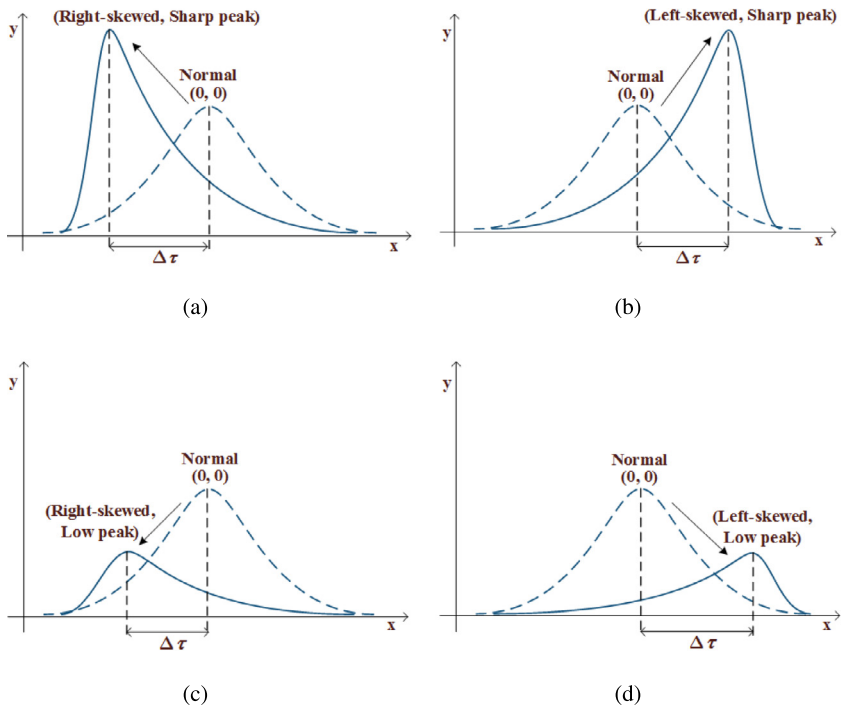

Fig. 7. The trend degree curves, which include the right-skewed and sharp peak (a), the left-skewed and sharp peak (b), the right-skewed and low peak (c), and the left-skewed and low peak (d) distribution patterns, start from the respective normal origin (i.e., a kind of origin relative to the normal distribution) and move to their deflection points at the four different oblique angles in $\Delta\tau$ hour windows.

time interval $\Delta\tau$ is limited to a maximum of two days (i.e., 4 to 48 h).

The SWWS adheres to a heuristic principle, "homogeneous extension but heterogeneous maintenance", which means if two adjacent workload waveform trend classes ξ_i and $\xi_{i+\Delta\tau+1}$ belong to the same trend class, then the workload waveform sampling window must be extended by 2, otherwise, the sampling window remains unchanged. This method better considers the diversity of sampled workload waveform trends, and performs the cloud resource prediction according to the changing waveform trends. Its final objective is to minimize the prediction error. Thereby, we give the following conditional assignment formula:

$$\begin{cases} \Delta\tau \leftarrow 2\Delta\tau, & \xi_i \text{ and } \xi_{i+\Delta\tau+1} \in \xi^{+} \\ \Delta\tau \leftarrow 2\Delta\tau, & \xi_i \text{ and } \xi_{i+\Delta\tau+1} \in \xi^{\circ} \\ \Delta\tau \leftarrow 2\Delta\tau, & \xi_i \text{ and } \xi_{i+\Delta\tau+1} \in \xi^{-} \\ \Delta\tau \text{ remains unchanged}, & \text{otherwise} \end{cases} \tag{3}$$

To explain the SWWS method in Eq. (3) intuitively, we depict the processes of extending or not extending the sampling time window by $2\Delta\tau$ in Fig. 8 that contains four cases of requested workload waveform curves corresponding to the four conditions in Eq. (3).

Fig. 8(a) shows that when the requested workload waveform has the peak trend during two continuous time window $\Delta\tau$ that means ξ_i *and* $\xi_{i+\Delta\tau+1} \in \xi^{+}$, the waveform sampling time window $\Delta\tau$ is extended by $2\Delta\tau$. Fig. 8(b) indicates that when the workload waveform shows the steadiness trend during two continuous time window $\Delta\tau$ that denotes ξ_i *and* $\xi_{i+\Delta\tau+1} \in \xi^{\circ}$, the sampling time window $\Delta\tau$ is extended by $2\Delta\tau$. Fig. 8(c) expresses that when the workload waveform presents the trough trend during two continuous time window $\Delta\tau$, i.e., ξ_i *and* $\xi_{i+\Delta\tau+1} \in \xi^{-}$, the sampling time window $\Delta\tau$ is extended by $2\Delta\tau$. The final case in Fig. 8(d) shows that when the workload waveform has the regular changing characteristic in which the peak, steadiness and trough waveform trends alternately appear in turn during two or more continuous time window $\Delta\tau$, the sampling time window $\Delta\tau$ is not extended and remains unchanged.

In light of the proposed workload trend classification, the SWWS method and Eq. (1), (2), (3), we design Algorithm 1 to efficiently classify and sample the requested workload trends so as to prepare for the data model training and cloud resource usage prediction in the next section. In Algorithm 1, Lines 1 and 2 describe the initialization and preparation of the feature extraction on the raw dataset, and Line 3 conducts the first classification by using initial parameters. Next, the key to this algorithm is to implement the scalable window waveform sampling on classified workloads, as shown in Lines 4 to 17. In Lines 18 and 19, the algorithm recalculates the workload trend degree and reclassifies the sampled data based on the new waveform sampling window. Finally, the algorithm generates new four-dimensional feature vector of all the requested workloads in Line 20, and outputs the sampled dataset in Line 21. The time complexity of Algorithm 1 is $O(2n + \frac{2n}{\Delta\tau_h} + 2^k n)$, where n is the number of the workload sample data, $\frac{\Delta\tau_h}{\Delta\tau}$ denotes the times of classified judgment on the workload waveform, and k is the extending exponent of workload waveform sampling window. Thus, the overall time complexity of this algorithm is $O(2^k n)$.

5. OEGBR prediction mechanism and algorithm

After completing the workload data sampling, this section puts forward the optimal error gradient boosting regression (OEGBR) mechanism and algorithm to carry out the data model training and prediction. Its goal is to construct the mapping relationship between requested workloads and corresponding cloud resource usage amount, and to predict the reasonable usage amount of cloud resources in the future.

5.1. OEGBR prediction mechanism

The OEGBR prediction mechanism has two processes: learning sample data (i.e., training model) and predicting future result. Given a training dataset $T = \{(x_1, y_1), (x_2, y_2), \ldots, (x_i, y_i),$

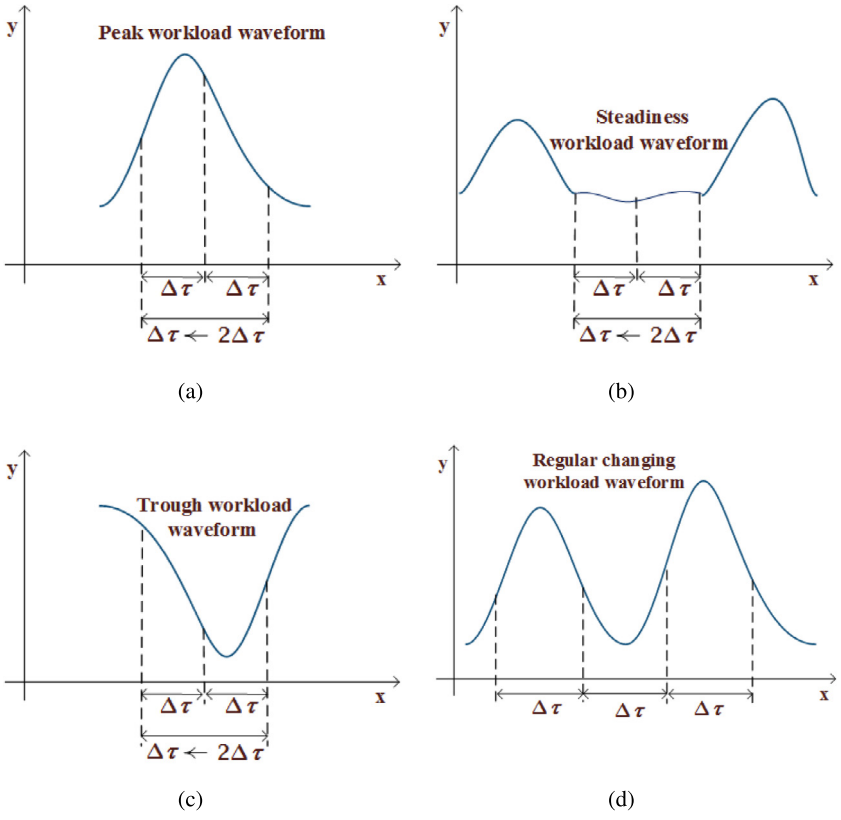

Fig. 8. The four cases of extending or not extending the sampling time window by $2\Delta\tau$ corresponding to the four conditions in Eq. (3), i.e., extending the sampling time window by $2\Delta\tau$ in (a), (b) or (c) when ξ_i *and* $\xi_{i+\Delta\tau+1} \in \xi^+$, ξ_i *and* $\xi_{i+\Delta\tau+1} \in \xi^\circ$, or ξ_i *and* $\xi_{i+\Delta\tau+1} \in \xi^-$ is satisfied, respectively. Otherwise, not extending the sampling time window $\Delta\tau$ and the window size remains unchanged in (d).

$\ldots, (x_n, y_n)\}$, where n is total hourly time points of the dataset as described in Section 4.1, $x_i \in X$ is requested workloads (i.e., the sample data that is input), and y_i is cloud resource usage amount (i.e., the output data corresponding to the input data x_i) that can be used as the label data of x_i. Based on the training dataset T, our learning system repeatedly trains the model $Y = f(X)$ for multiple rounds, and finally obtains the prediction model which is a mapping function $Y = f(\hat{X})$. For a new requested workload x_{n+1} on an hourly time point, the prediction system uses $Y = f(\hat{X})$ to determine its cloud resource usage amount y_{n+1} that is output. In the process of training model, the learning system adopts the cross validation as the model selection method in which a part of sample data is randomly selected for the training set and the rest for the test set. The training sample data is generally more than the test sample data.

The OEGBR is based on the existing gradient boosting regression (GBR), which in advance needs to generate a least squares regression tree $f(x)$. In the input space of the training dataset, the GBR recursively divides each region into two subregions, and determines the output value on each subregion to construct a binary decision tree.

Specifically, the GBR firstly selects the optimal segmentation variable j and segmentation point s, and solves the following equation:

$$\min_{j,s}(\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2) \tag{4}$$

where it traverses the variables j, scans the segmentation point s regarding the fixed variables j whose range of value is $1 \le j \le n$, n is maximum number of the input samples, and selects the pair (j, s) that minimizes Eq. (4).

Secondly, it divides the region with the selected pair (j, s) and determines the corresponding output value as follows:

$$R_1(j, s) = \{x | x^j \le s\}, \quad R_2(j, s) = \{x | x^j > s\} \tag{5}$$

$$\hat{c}_j = \frac{1}{N_j} \sum_{x_i \in R_j(j,s)} y_i, \quad x \in R_j, \ j = 1, 2 \tag{6}$$

where N_j is the number of samples on the subregion R_j, and $\hat{c}_j$ (i.e., the optimal value of c_j) denotes the mean value of the output y_i corresponding to all input samples x_i on the subregion R_j.

Thirdly, it continues calling the above two steps on both subregions until the stop condition is met.

Finally, the input space is divided into J subregions $R_1, R_2, \ldots, R_J$ to generate the decision tree:

$$f(x) = \sum_{j=1}^{J} \hat{c}_j I(x \in R_j) \tag{7}$$

The key difference between the proposed OEGBR and the existing GBR has two aspects: (a) The segmentation variable on the subregions into which the input space is divided, has been adjusted according to the workload waveform sampling window $\Delta\tau$ from the proposed SWWS method. Since we employ the SWWS method on the classified workload trends for realizing more accurate cloud resource usage predictions, the above segmentation variable j needs to be adjusted by using the sampling window $\Delta\tau$. (b) When the prediction error rises prior to outputting the final prediction value, for decreasing the next prediction error, the OEGBR can reduce the sampling window by $\frac{\Delta\tau}{2}$, and return to the SWWS algorithm for next round of workload waveform sampling and cloud resource usage prediction. To this end, we propose the preliminary theorem for the OEGBR algorithm as follows:

Algorithm 1: Workload trend classification and SWWS

Input: the raw dataset with requested workloads and cloud resource usage amount in a given hourly period n through online querying, $T = \{(x_1^{(1)}, y_1), \cdots, (x_i^{(1)}, y_i), \ldots, (x_n^{(1)}, y_n)\}$, and a starting value of the workload waveform sampling window τ_{st}

Output: the dataset sampled by using the SWWS $\overrightarrow{T} = \{(x_i^{(l)}, y_i) | 1 \le i \le n, 1 \le l \le 4\}$

1 Extract the features of the dataset T according to Section 4.1;
2 Initialize a waveform sampling window $\Delta\tau \leftarrow \tau_{st}$, and calculate the trend degree $TD(\varphi^{rw}|_i^{i+\Delta\tau})$ of each sample data by using Eq. (1);
3 Initially classify the workload sample data by using Eq. (2), and form three-class workload waveform trends, *Peak*, *Steadiness* and *Trough*, whose value are respectively assigned as ξ^+, ξ° and ξ^-;
4 **while** $4 \le \Delta\tau < 64$ **do**
5 $last\,\Delta\tau \leftarrow \Delta\tau$;
6 **foreach** $i = 1, 2, \ldots, n$ **do**
7 **if** (ξ_i *and* $\xi_{i+\Delta\tau+1} \in \xi^+$) *or* ($\xi_i$ *and* $\xi_{i+\Delta\tau+1} \in \xi^\circ$) *or* ($\xi_i$ *and* $\xi_{i+\Delta\tau+1} \in \xi^-$) **then**
8 $\Delta\tau \leftarrow 2\Delta\tau$;
9 **else**
10 **beak;**
11 **end**
12 **end**
13 $this\Delta\tau \leftarrow \Delta\tau$;
14 **if** $this\Delta\tau = last\,\Delta\tau$ **then**
15 **beak**;
16 **end**
17 **end**
18 Recalculate the trend degree $TD(\varphi^{rw}|_i^{i+\Delta\tau})$ of each sample data by using Eq. (1);
19 Reclassify the workload sample data by using the same steps as Line 3 ;
20 Generate the new four-dimensional feature vector $x_i = (\varphi_i^{rw}, TD(\varphi^{rw}|_i^{i+\Delta\tau}), D_i, N_i^{cr})$ of requested workloads in each hour;
21 Output the sampled dataset $\overrightarrow{T} = \{(x_i^{(l)}, y_i) | 1 \le i \le n, 1 \le l \le 4\}$;

Theorem 1. *The SWWS method that is on the basis of Eq. (1) to Eq. (3) can reduce the prediction error of cloud resource usage amount according to the requested workload sampled, thus improve the accuracy of prediction.*

Proof. In light of Eq. (1) to Eq. (3), the SWWS method uses the workload waveform sampling window $\Delta\tau = 2^k$, which extends the optimal segmentation variable j and changes the segmentation point s in Eqs. (4) and (5), further decreases the total value of $\hat{c}_j$ in Eqs. (6) and (7). This is because that our given maximum segmentation $\lfloor \frac{n}{2^k} \rfloor$ is lower than the maximum value n of the variable j in Eq. (5), such that $\sum_{j=1}^{\lfloor \frac{n}{2^k} \rfloor} \hat{c}_j I(x \in R_j) < \sum_{j=1}^{n} \hat{c}_j I(x \in R_j)$ according to the GBR, where $k \in [2, 6)$ in Section 4.2, and $I(x)$ is the indicator function. Thus, we give the following proof on the error reduction of the prediction on cloud resource usage amount.

As for the GBR, the gradient boosting algorithm finally achieves the whole boosting decision tree: $f(\hat{x}) = f_M(x) = \sum_{m=1}^{M} \sum_{j=1}^{J} c_{m,j} I(x \in R_{m,j})$, where M is the number of decision trees. The $f(\hat{x})$ is used to fit the training data, and the prediction error can be evaluated by the squared error loss function $L(y, f(\hat{x})) = \sum_{i=1}^{n} (y_i - f(\hat{x}))^2$. Further, we solve the partial derivative of the function L with respect to the segmentation variable j as follows:

$$\begin{aligned} \frac{\partial L(y, f(\hat{x}))}{\partial j} &= \frac{\partial L(y, f(\hat{x}))}{\partial f(\hat{x})} \cdot \frac{\partial f(\hat{x})}{\partial j} \\ &= \frac{\partial \sum_{i=1}^{n} (y_i - \sum_{m=1}^{M} \sum_{j=1}^{J} c_{m,j} I(x \in R_{m,j}))^2}{\partial \sum_{m=1}^{M} \sum_{j=1}^{J} c_{m,j} I(x \in R_{m,j})} \cdot \frac{\partial f(\hat{x})}{\partial j} \\ &= 2 \frac{\partial f(\hat{x})}{\partial j} \sum_{i=1}^{n} (\sum_{m=1}^{M} \sum_{j=1}^{J} c_{m,j} I(x \in R_{m,j}) - y_i) \end{aligned} \tag{8}$$

When M remains the same, and we let the value of J decrease from n to $\lfloor \frac{n}{2^k} \rfloor$, that of $\sum_{m=1}^{M} \sum_{j=1}^{J} c_{m,j} I(x \in R_{m,j})$ will become gradually small, such that the value of $\frac{\partial f(\hat{x})}{\partial j} = \frac{\partial \sum_{m=1}^{M} \sum_{j=1}^{J} c_{m,j} I(x \in R_{m,j})}{\partial j}$ is greater than 0. Moreover, because each of prediction goal y_i does not change, this makes the value of $\sum_{i=1}^{n} (\sum_{m=1}^{M} \sum_{j=1}^{J} c_{m,j} I(x \in R_{m,j}) - y_i)$ go down to 0 until it is less than 0. The above reasoning process comes to this conclusion that the value of $\frac{\partial L(y, f(\hat{x}))}{\partial j}$ will be finally less than 0 when the value of J is decreased from n to $\lfloor \frac{n}{2^k} \rfloor$. This result indicates that Eq. (8) turns into a negative gradient. Therefore, the prediction error $L(y, f(\hat{x}))$ of cloud resource usage amount can be reduced with a decrease of the maximum value n of the segmentation variable j, in other words, the accuracy of prediction can be improved.

5.2. OEGBR algorithm description

According to the proposed Theorem 1 and the existing GBR algorithm, we give the OEGBR algorithm to conduct the data model training and prediction as described in Algorithm 2. This algorithm firstly initializes a root tree with a constant value that minimizes the squared error loss function (see Line 1). From Lines 3 to 5, the algorithm calculates the value of the negative gradient (i.e., residual) of the squared error loss function in the current model, and then fits the approximate residuals in Line 6 when giving all the leaf terminal nodes' subregions on the range of the proposed 1 to $\lfloor \frac{n}{2^k} \rfloor$. Next, from Lines 7 to 9, the algorithm searches for the estimated mean value $c_{m,j}$ on the area of the leaf terminal nodes, which minimizes the squared error loss function, and updates the regression tree in Line 10. From Lines 12 to 14, we devise the judgment on whether the loss has increased, if so, Algorithm 2 reduces the value of the workload waveform sampling window $\Delta\tau$, and returns Algorithm 1 to generate a new dataset sampled by using the SWWS. Otherwise, Algorithm 2 outputs the final decision regression tree $f(\hat{x})$ in Line 15. The overall time complexity of this algorithm is $O(nM)$.

6. Performance evaluation

We conducted the simulation experiments on two kinds of datasets to evaluate the prediction accuracy of the proposed OEGBR algorithm with the SWWS method. All simulation experiments were implemented through combining Java with Python programming, and the configuration of computer was Intel Core i5-3337U CPU 1.80 GHz, 4.0 GB RAM and the Windows 7 (64 bits) operating system.

6.1. Experimental setup

6.1.1. Experimental datasets

The simulation experiments use the relatively-periodic and randomly-changing datasets, which respectively correspond to

Algorithm 2: OEGBR prediction algorithm

Input: the dataset sampled by using the SWWS $\overrightarrow{T} = \{(x_i^{(l)}, y_i) | 1 \le i \le n, 1 \le l \le 4\}$

Output: the decision regression tree $f(\hat{x})$

Initialize $f_0(x) \leftarrow \arg\min_c \sum_{i=1}^{n} L(y_i, c)$;
foreach $m = 1, 2, \ldots, M$ **do**
  foreach $i = 1, 2, \ldots, n$ **do**
    $r_{m,i} = -[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}]_{f(x)=f_{m-1}(x)}$;
  end
  Fit a regression tree to the $r_{m,i}$ when giving terminal subregions $R_{m,j}, j = 1, 2, \cdots, \lfloor \frac{n}{2^k} \rfloor$;
  foreach $j = 1, 2, \ldots, \lfloor \frac{n}{2^k} \rfloor$ **do**
    $c_{m,j} = \arg\min_c \sum_{x_i \in R_{m,j}} L(y_i, f_{m-1}(x_i) + c)$;
  end
  Update $f_m(x) \leftarrow f_{m-1}(x) + \sum_{j=1}^{\lfloor \frac{n}{2^k} \rfloor} c_{m,j} I(x \in R_{m,j})$;
end
if $L(y, f_M(x))$ *increases* **then**
  $\tau_{st} \leftarrow \frac{\Delta\tau}{2}$, and go to Algorithm 1;
end
Output $f(\hat{x}) = f_M(x) = \sum_{m=1}^{M} \sum_{j=1}^{\lfloor \frac{n}{2^k} \rfloor} c_{m,j} I(x \in R_{m,j})$;

real-world web application [7] and the Google cluster-usage traces [5]. The real-world web application has been deployed in the Microsoft Azure China East2 Data Center (Shanghai), and rented a *F2* instance of the *F* series which contains 2 cores of CPU, 4 GB of Memory (RAM), 32 GB of Disk, and 320Mbps of maximum transmission bandwidth (BW). In acquiring the real-world dataset, we utilize the free Zabbix [41] cloud resource monitor system which consists of an agent monitor, a Web front-end system and a Zabbix database server. The agent monitor has been installed in the *F2* instance of Azure China East2 (Shanghai) running the real-world web application. It collects the trace data of workloads and cloud resource usage from the VM instance, and sends the data to the Zabbix database server in real time. In order to facilitate the dataset querying and analysis, we have installed the Web front-end system and the Zabbix database server in another rented *F2* instance of Microsoft Azure (Asia Pacific) Australia East Data Center (Sydney). Thus, the real-world dataset is easy to be quickly and timely fetched for our simulation experiments. In the experiments, we online queried the 432 h (18 days) of time-series dataset from 2019-11-08 to 2019-11-25 at random, which includes the hourly number of requested workloads, the hourly mean usage percentage of CPU, Memory and Disk, and the hourly mean Bandwidth usage.

On the other hand, we obtained a Google cluster-usage time-series dataset, which involves the mean CPU usage rate, assigned memory usage, mean local disk space used, job ID and task index. We selected a total 720 h (30 days) of cluster dataset (the resource usage trace of 8640 VMs) that is summarized hourly-interval data in time order. The number of requested workloads in the Google cluster dataset is not directly given, however, we find that there is an obvious pattern similarity between the workload number and the memory usage from the real-world dataset (see Figs. 1 and 2), therefore, we adopted the Binary Exponential Interpolation method to simulate an approximate requested workload curve according to the job ID, task index and assigned memory usage, and then used this simulated workload curve to train and predict the usage amount of cloud resources (CPU, Memory and Disk) in the selected Google cluster dataset.

The list of original relevant features of datasets and the extracted key features for the final prediction from the real-world web application and the Google cluster-usage traces are shown in Table 2. In the experiments, we use the workload trend classification and the SWWS method in Algorithm 1 to generate the sample dataset, and then input the dataset into Algorithm 2 for conducting the cloud resource usage prediction. To unify the measurement scale, we adopt the maximum and minimum normalization method to limit the value of each four-dimensional requested workload sample data to a range of 0 to 1, and label each sample data with the four types (CPU, Memory, Disk and Bandwidth) of cloud resource usage value. The entire generated sample dataset is shown in Table 3.

6.1.2. The brief description of the compared approaches

We performed the different experiments on the data model training and prediction by using the proposed OEGBR, and also compared with 6 types of existing statistical learning regression approaches such as LinearRegression, BayesianRidge, ElasticNet, ARDRegression, KernelRidge and SVR in which the first four types are different linear regression models [42]. All of compared models are briefly introduced as follows.

In the linear regression approaches, the expected result $\hat{y}$ of each model is a linear combination of input variables represented as:

$$\hat{y}(w, x) = w_0 + w_1 x_1 + \cdots + w_p x_p \tag{9}$$

where $X = (x_1, \ldots, x_p)$ is the input feature vector, and $w = (w_1, \ldots, w_p)$ denotes the coefficient vector, and w_0 stands for the intercept.

LinearRegression is an ordinary linear regression method. It fits a linear model like Eq. (9) through utilizing the Ordinary Least Squares to minimize the residual sum of squares between the predicted and the actual results, i.e., $\min_w (\|Xw - y\|_2)^2$, where $\|\cdot\|_2$ is L_2 norm.

BayesianRidge estimates a probabilistic model of the regression problem in which there is a Ridge regression, i.e., the ridge coefficients minimize a penalized residual sum of squares such as $\min_w (\|Xw - y\|_2)^2 + \alpha(\|w\|_2)^2$, where the complexity parameter $\alpha \ge 0$ is a parameter that controls the amount of shrinkage, i.e., the larger α, the greater the amount of shrinkage, thus the coefficients will be more robust to collinearity. The expected result is a fully probabilistic model $p(y \mid X, w, \alpha) = \mathcal{N}(y \mid Xw, \alpha)$, where the output y is assumed to be Gaussian distributed of Xw. The prior of the coefficient w is given by a spherical Gaussian $p(w \mid \lambda) = \mathcal{N}(w \mid 0, \lambda^{-1}\mathbf{I}_p)$, where the priors α and λ are determined by the gamma distribution. In the end, the generated model is called Bayesian Ridge regression.

ElasticNet denotes a linear regression model called Elastic Net that uses L_1 and L_2 norm-regularization of the coefficients to train. It is well suited for situations where there are interdependent multiple features, and is allowed to inherit some Ridge's stability under rotation. The objective function of minimization is expressed as $\min_w \frac{1}{2n_{samples}}(\|Xw - y\|_2)^2 + \alpha\rho\|w\|_1 + \frac{\alpha(1-\rho)}{2}(\|w\|_2)^2$.

ARDRegression is called Automatic Relevance Determination regression similar to Bayesian Ridge regression except it can lead to sparser coefficients w. ARDRegression gives a different prior over w through neglecting the assumption of the Gaussian being spherical. The distribution over w is assumed to be an axis-parallel and elliptical Gaussian distribution. Thus, each coefficient w_i is obtained from a zero-centered Gaussian distribution with a precision λ_i such as $p(w \mid \lambda) = \mathcal{N}(w \mid 0, A^{-1})$ where $diag(A) = \lambda = \{\lambda_1, \ldots, \lambda_p\}$.

Table 2
The original and extracted key features of datasets.

Feature item	Real-world web application dataset	Google cluster-usage trace dataset
The original relevant features	(Per millisecond) The number of requested workloads, Maximum requested workloads, Processor load, CPU usage percentage, Memory usage percentage, Disk usage percentage, Bandwidth	(Per 5 min) The number of requested workloads, Machine number, Mean CPU usage rate, Maximum CPU usage, Assigned memory usage, Maximum memory usage, Mean local disk space used
The extracted key features for the final prediction (Per-hour sampling interval)	The number of requested workloads, The trend degree of requested workloads, The variance of requested workloads' number, The number of CPU cores of VMs, CPU usage percentage, Memory usage percentage, Disk usage percentage, Bandwidth	The number of requested workloads, The trend degree of requested workloads, The variance of requested workloads' number, The number of CPU cores of VMs, Mean CPU usage rate, Assigned memory usage, Mean local disk space used

Table 3
The entire generated sample dataset for the final prediction. (cloud resource requirements: 1∼2 cores of CPU, 4 GB of Memory, 32 GB of Disk and 320 Mbps of maximum Bandwidth)

The different starting value τ_{st} of workload waveform sampling window	Real-world web application dataset (432 h of hourly-interval dataset)	Google cluster-usage trace dataset (720 h of hourly-interval dataset)
4, 8, 16, and 32	232 training data + 200 test data 282 training data + 150 test data 332 training data + 100 test data 382 training data + 50 test data	320 training data + 400 test data 420 training data + 300 test data 520 training data + 200 test data 620 training data + 100 test data
No using the sampling window	232 training data + 200 test data 282 training data + 150 test data 332 training data + 100 test data 382 training data + 50 test data	

KernelRidge [43] combines Ridge regression and classification with the kernel trick, which uses the linear Ordinary Least Squares with L_2 norm regularization, so it is called Kernel Ridge regression (KRR). This regression model is supplemented with the kernel trick $\sum_{j=1}^{n} \alpha_j \kappa(\vec{x}_j, \vec{x}_i)$ and ridge (regularization term) $\lambda \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \kappa(\vec{x}_j, \vec{x}_i)$, accordingly, the KernelRidge's loss function of minimization can be expressed as $\min \sum_{i=1}^{n} (y_i - \sum_{j=1}^{n} \alpha_j \kappa(\vec{x}_j, \vec{x}_i))^2 + \lambda \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \kappa(\vec{x}_j, \vec{x}_i)$.

SVR [44] refers to Support Vector regression, an application method from Support Vector Machines (SVMs). SVR uses ε-insensitive loss to solve the minimization problem with L_2 norm regularization such as $\min_{w,b,\zeta,\zeta^*} \frac{1}{2} w^T w + C \sum_{i=1}^{n} (\zeta_i + \zeta_i^*)$ is subject to $y_i - w^T \phi(x_i) - b \leq \varepsilon + \zeta_i$, $w^T \phi(x_i) + b - y_i \leq \varepsilon + \zeta_i^*$, and $\zeta_i, \zeta_i^* \geq 0, i = 1, \ldots, n$, where the model penalizes samples whose prediction is at least ε away from their true target. Depending on whether their predictions are above or below the ε tube, the above samples penalize the objective through using ζ_i or ζ_i^*.

Each of the mentioned regression methods performs the four-step model training and prediction process as follows:

• *Preparing the training dataset*

(a) divide the dataset with extracted features into two parts, one is the training dataset and the other is the test dateset, (b) read in the sampled training dataset that contains the features of requested workloads x and the related usage amount y of cloud resources in each hour, and (c) separate the independent variables x and the dependent variables y.

• *Training the regression model*

(a) build the regression model object, (b) input the regression model into the cross validation model for training the model, (c) output the result of the cross validation, and (d) output the predicted value y obtained in the regression training.

• *Evaluating the trained model*

(a) establish the regression evaluation indicator objects, (b) calculate each regression indicator, and (c) output the model evaluation results.

• *Performing the regression prediction*

(a) read in the test dataset to be predicted, (b) use the trained model to perform the prediction, and (c) output the prediction results.

In the process of the model training, the feature selection is very important, and a good feature selection can improve the performance of the model. The main functions of the feature selection are to reduce the number and dimension of features, make the model generalization ability stronger, and decrease the overfitting. The compared regression methods adopt the feature scoring mechanism to select the features. Specifically, the coefficient w of each regression model is used to select the features. The more important the features are, the higher the coefficients will be in the model, and the more irrelevant the features are to the output variables, the closer the coefficients will be to 0. As described in the mentioned regression approaches, the feature extraction uses the regularization which is a method for adding additional constraints or penalty to an existing model (loss function) so as to prevent overfitting and improve generalization. Accordingly, minimizing a loss function becomes $E(X, Y) + \alpha \|w\|$ from $E(X, Y)$, where α means a tunable parameter that controls the intensity of regularization, w denotes the vector of model coefficients (parameters), and $\|\cdot\|$ is L_1 or L_2 norm.

In terms of the cloud resource prediction, the latest researches [3,31] use the same Google cluster-usage dataset as this paper, and their evaluation metrics on the prediction error adopt the mean square error (MSE) which is equal to the square of root mean square error ($RMSE^2$) while our statistical regression method (OEGBR) uses the $RMSE$. The method in [3] gives a top-sparse auto-encoder (TSA) and an efficient deep Learning based algorithm to predict the cloud resource usage, and the approach in [31] devises a probabilistic demand allocation (PDA) system to solve the demand allocation problem in which the tenants' cloud resource demands are predicted according to their historical usage records. However, our approach integrates the

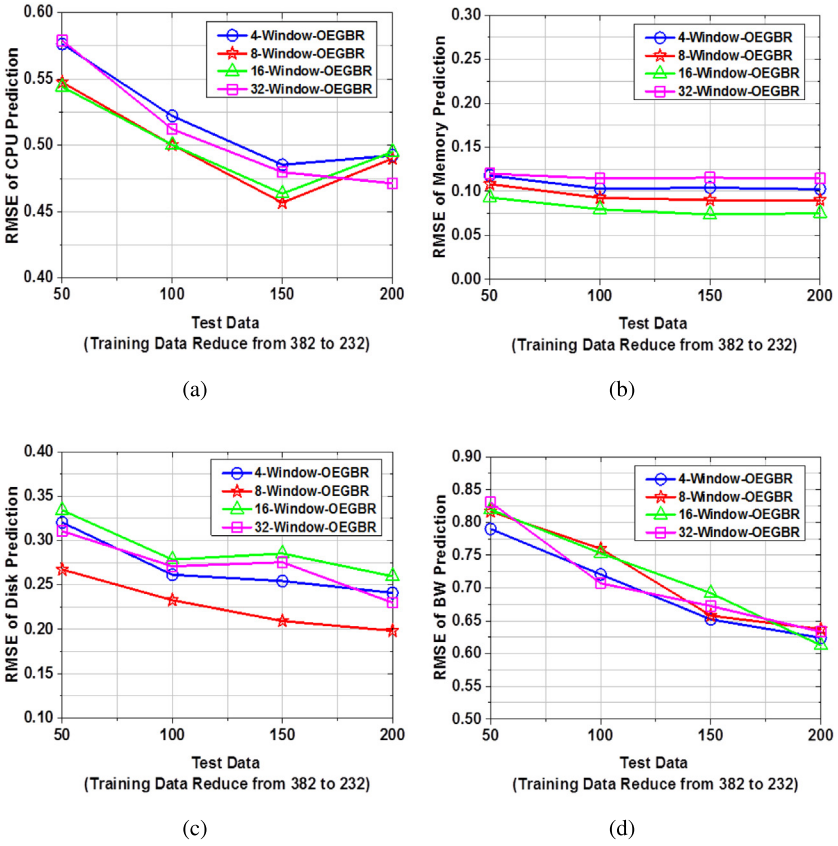

Fig. 9. The prediction accuracy of cloud resource usage amount (CPU, Memory, Disk and Bandwidth) regarding different requested workloads with four starting values ($\tau_{st} = 4 \sim 32$) of workload waveform sampling windows on different number of training and test data for the relatively-periodic real-world dataset.

statistical learning model with the task workload waveform sampling different from the two latest researches, and as for the *MSE*, our average prediction error (less than 0.0036, i.e., the square of *RMSE* 0.06 shown in Fig. 13 of this paper) is lower than those of the two methods (0.0038 shown in the page 931 of [3], and 0.02 shown in the page 10 starting from the page 1 of [31]) under the same prediction time and number of cloud resource demands. Thus, our model has the higher accuracy for predicting cloud resource usage. Based on the above analysis, we focus on comparing our method with the 6 types of existing statistical learning regression models described in this section.

6.1.3. Performance evaluation metrics

In the experiments, we use the root mean square error (*RMSE*) and the mean absolute error (*MAE*) to evaluate the prediction accuracy and the effectiveness of our method by comparing with the six existing statistical learning regression approaches under different parameter settings, and the two metrics are given as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(predicted\_data_i - true\_data_i)^2} \tag{10}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|predicted\_data_i - true\_data_i| \tag{11}$$

where n is the total number of sample data, $predicted\_data_i$ and $true\_data_i$ represent the predicted value and the true (actual) value of ith sample data, respectively.

6.2. Experimental results

6.2.1. For the relatively-periodic real-world dataset

In this section, for the relatively-periodic real-world dataset, we evaluate the prediction accuracy of cloud resource (CPU, Memory, Disk, and Bandwidth) usage amount corresponding to different requested workloads on hourly time point through adopting the proposed SWWS method and OEGBR algorithm. We firstly conducted the data model training and prediction by using four input starting values τ_{st} (i.e., X-Window-OEGBR) of workload waveform sampling windows on the training and test data. Fig. 9 depicts that the *RMSE* of all types of the predicted cloud resources gradually reduces with an increase of the test sample data, and the predicted results with the starting value τ_{st} of 8 on the SWWS windows (8-Window-OEGBR) have the smaller prediction errors, especially for the prediction on the usage amount of CPU and Disk, the errors are minimum. This is because that when the reduction of total training sample data is not great, more test sample data, higher prediction accuracy. Moreover, for the data of changing requested workloads and cloud resource usage in the dataset, our algorithms can auto-calculate the suitable sampling window for the whole sample data according to the starting value τ_{st}, and the value τ_{st} of 8 in Fig. 9 is the optimal value.

Further, we input the workload waveform sample window value τ_{st} of 8 into Algorithm 1 to predict the usage value of the four different cloud resources. From Fig. 10, we can see that all of fitting results between the predicted value and the true value of cloud resource usage amount (CPU, Memory and Disk) are very well except the Bandwidth. Because the running patterns of requested workloads in the cloud systems are closely related with the usage amount of CPU, Memory and Disk reserved, the predicted value can approximate the true value. But the usage

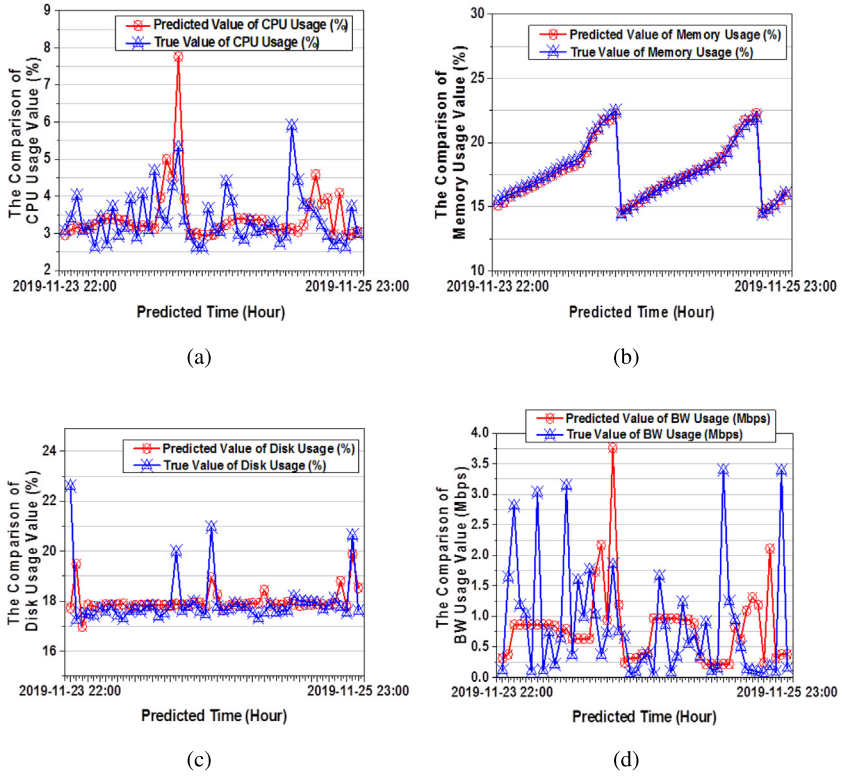

Fig. 10. The fitting comparison of cloud resource usage amount (CPU, Memory, Disk and Bandwidth) between the predicted value and the true (actual) value with the starting window value τ_{st} of 8 on the training data of 382 and the test data of 50 from 2019-11-23 22:00 to 2019-11-25 23:00 (continuous 50 h).

amount of Bandwidth may be affected by the data being transmitted in the data center, so its fitting result has a degree of deviation.

Next, we conducted the data model training and prediction adopting the 7 types of statistical learning regression models, i.e., LinearRegression, BayesianRidge, ElasticNet, ARDRegression, KernelRidge, SVR and the proposed OEGBR, whose training and test data are the same as the previous settings. Fig. 11 illustrates that the *RMSE* and *MAE* of all the cloud resource prediction using the proposed OEGBR are minimum, i.e., owning highest prediction accuracy. Furthermore, with an increase of the test sample data, the prediction errors fall slightly. On one hand, these results show that the gradient boosting regression model has the best stability in response to different datasets, on the other hand, due to the SWWS on the classified requested workload waveform trends (Peak, Steadiness and Trough), the OEGBR algorithm can utilize the adjusted waveform window to train data model, and make the accurate prediction.

Finally, we compared the prediction accuracy of different cloud resource usage amount with other statistical learning regression models in the case of the two ways to use the waveform sampling window (X) and not to use the one (NoWin-X), where X denotes one of statistical learning regression models compared. In the former method, τ_{st} of 4 is not the optimal starting window, but using the value can reflect the prediction effect under the worse parameter. As shown in Fig. 12, the cloud resource prediction errors in most of regression models change little before and after using the SWWS method on the requested workloads, however, the prediction errors of the OEGBR algorithm with the SWWS are all lower than those of other X and NoWin-X regression models without the SWWS. These results show that our method can sample and train the workload data based on the varied characteristics of the workload waveforms, and achieve higher prediction accuracy.

6.2.2. For the randomly-changing Google cluster dataset

To further verify the effectiveness of the presented method, we selected a randomly-changing Google cluster dataset (8640 VMs) including the usage amount of CPU (mean CPU usage rate per hour), Memory (assigned memory usage per hour) and Disk (mean local disk space used per hour) that were preprocessed according to the way in Section 6.1.1, and then input into our algorithms. It is noted that there is no bandwidth data provided in Google cluster datasets. Since the prediction accuracy and the effectiveness are proved experimentally in Section 6.2.1 by comparing with other regression models, we only give the comparison of the prediction accuracy and the fitting degree under different training and test data with four starting values ($\tau_{st} = 4 \sim 32$) and τ_{st} of 16 on workload waveform sampling windows, respectively. Fig. 13 indicates that with an increase of the test sample data on requested workloads, the prediction errors on the usage amount of cloud resources generally reduce as a whole. Moreover, the predicted data using the starting waveform window value τ_{st} of 16 (16-Window-OEGBR) shows the relatively small prediction errors, where the prediction error on the usage amount of CPU is minimum. But the predicted effect using the starting window value τ_{st} of 4 is the worst. The reasons for these results are similar to Fig. 9, however, the only difference is that because the workload waveforms randomly vary in the selected Google cluster dataset, our algorithms need a bigger initial sampling window to cover the waveform segments that change too frequently, such that the predicted results have the higher accuracy.

Next, we conducted the comparison of fitting degree between the predicted value and the true value of cloud resource amount (CPU, Memory and Disk) with starting window value τ_{st} of 16 during continuous 100 h. As shown in Fig. 14, the fitting results between the predicted value and the true value of CPU and Memory usage amount are very well, and that of Disk usage amount is well on the whole but there is some deviation locally.

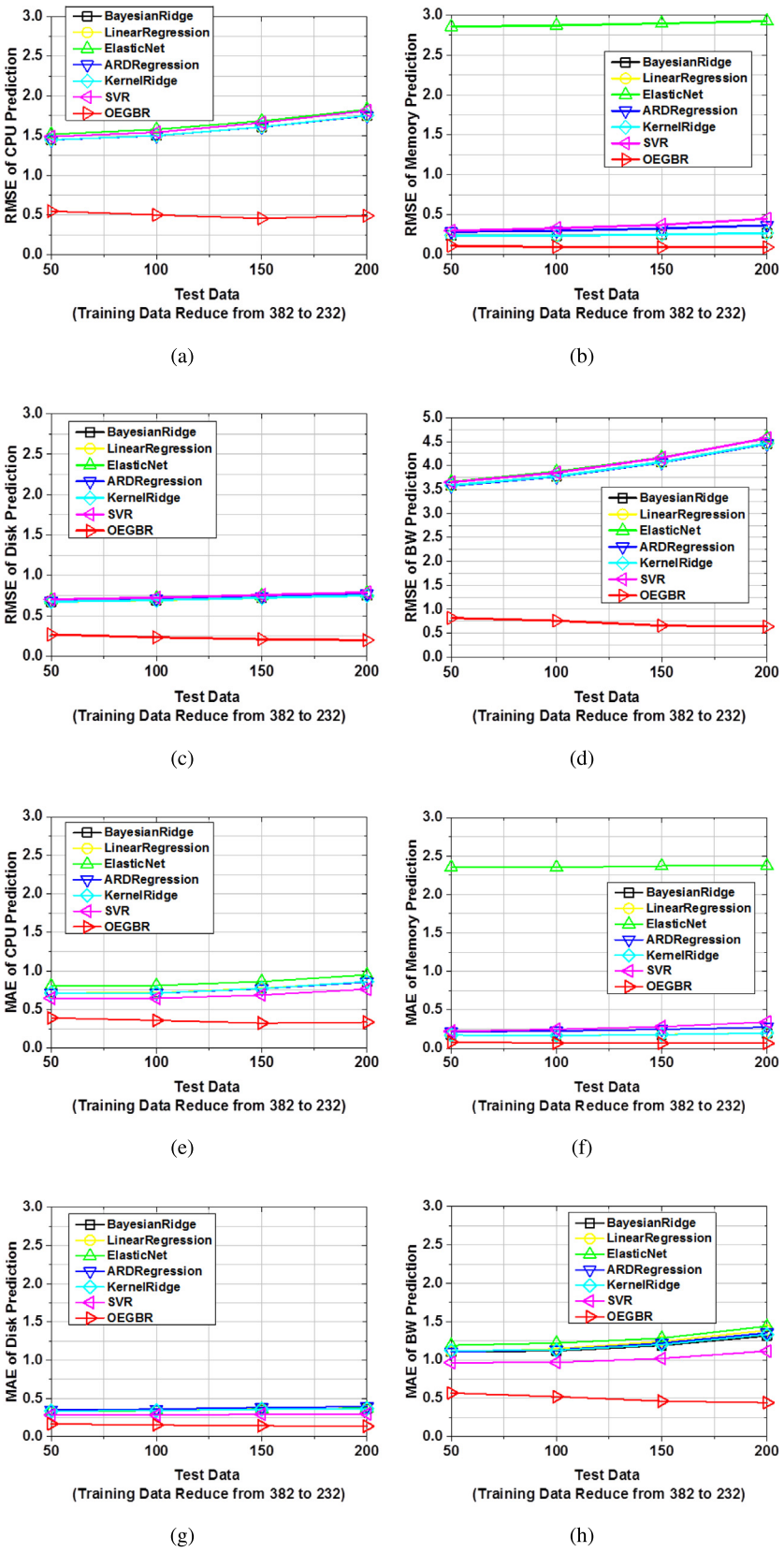

Fig. 11. The different methods' prediction accuracy comparison of cloud resource usage amount (CPU, Memory, Disk and Bandwidth) regarding different requested workloads with the starting window value τ_{st} of 8 on different number of training and test data.

This is because that using the starting waveform window value τ_{st} of 16 can obtain the higher prediction accuracy in line with Fig. 13.

6.2.3. The impact of predicted results on the resource reservation

The predicted results in Figs. 10 and 14 reflect the estimated level on the usage amount (value) of different kinds of cloud

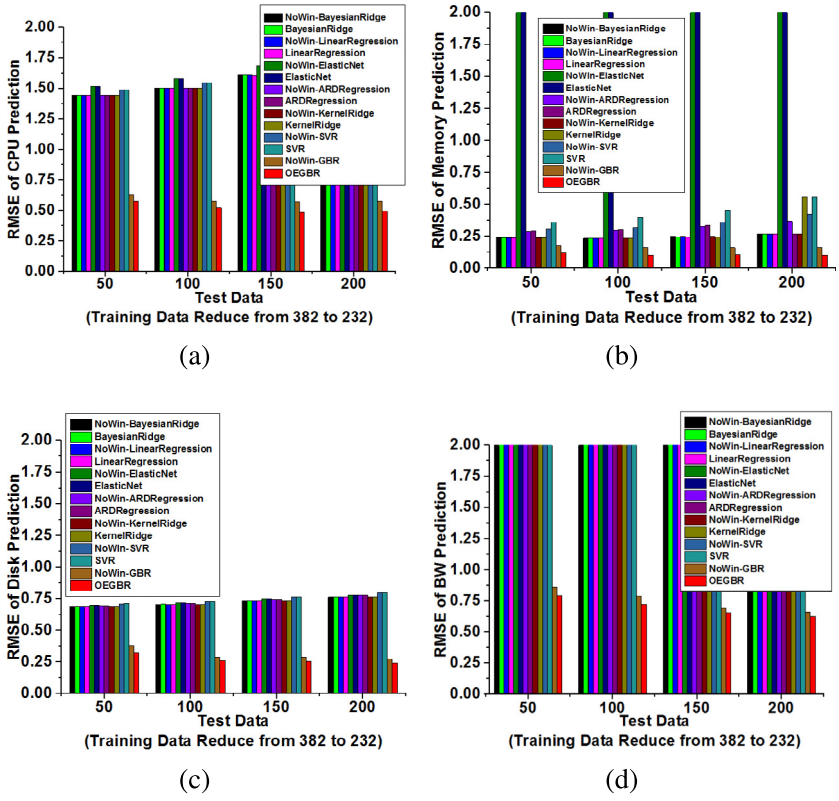

Fig. 12. The different methods' prediction accuracy of cloud resource usage amount (CPU, Memory, Disk and Bandwidth) on different number of training and test data when inputting the workload waveform sampling window ($\tau_{st} = 4$ that is not optimal starting window) and not using the one (NoWin-X).

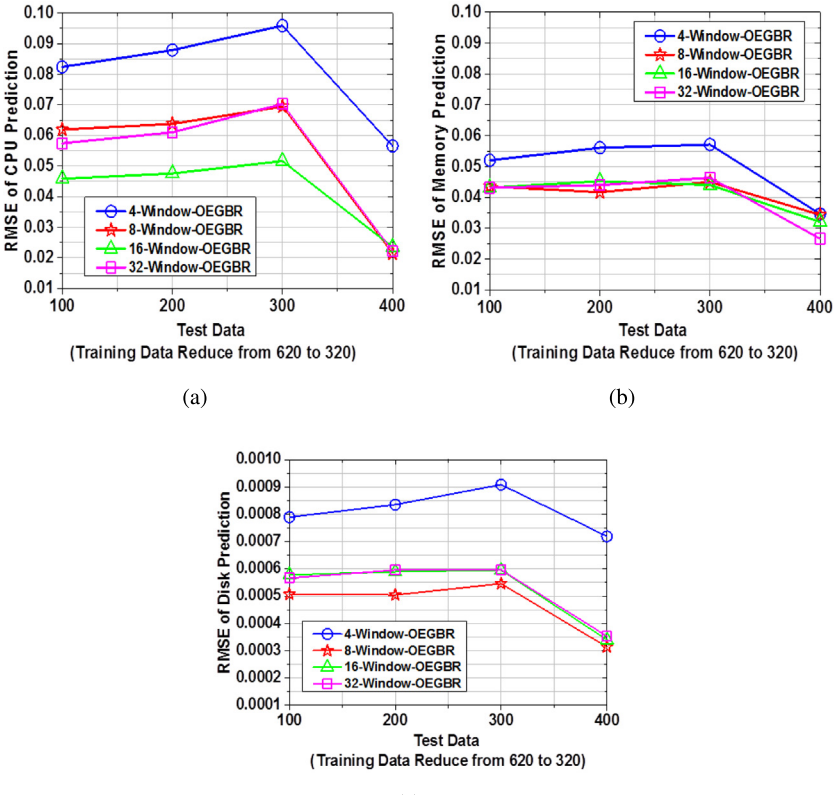

Fig. 13. The prediction accuracy of cloud resource usage amount (CPU, Memory and Disk) regarding different requested workloads with four starting values ($\tau_{st} = 4 \sim 32$) of workload waveform sampling windows on different number of training and test data for the randomly-changing Google cluster dataset.

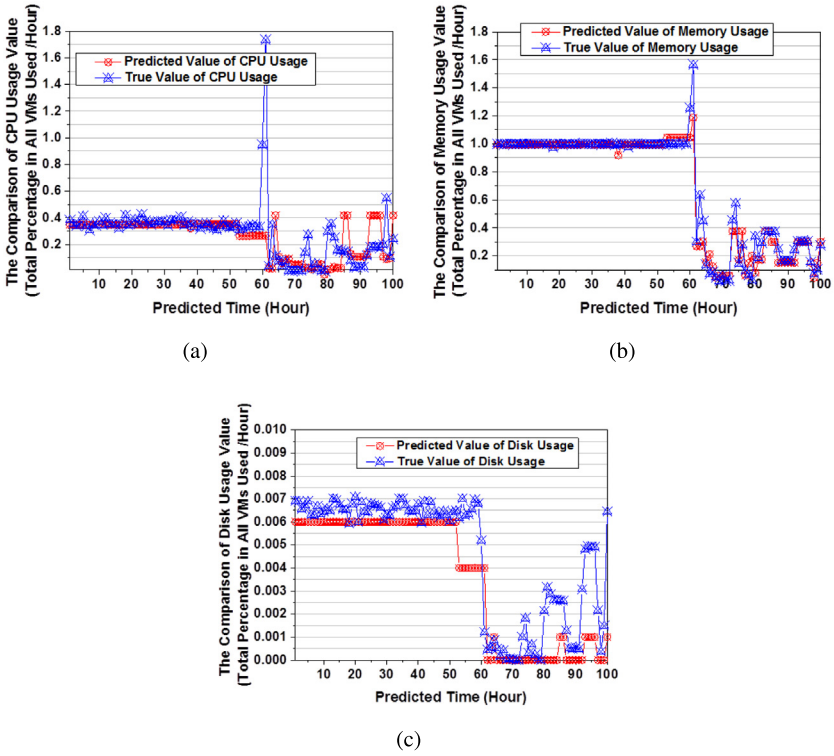

Fig. 14. The fitting comparison of cloud resource usage amount (CPU, Memory and Disk) between the predicted value and the true (actual) value with the starting window value τ_{st} of 16 on the training data of 620 and the test data of 100 (continuous 100 h).

resources, and also impact on the decision to adjust the resource reservation for users and the resource provisioning for cloud providers. We leverage the difference (error) between the predicted value and the true (actual) value of cloud resource usage to measure the error level that involves the overestimation, the unbiased estimation and the underestimation as expressed by the following conditional judgment formula:

$$\begin{cases} Overestimation, & (pd_i - td_i) > 0 \\ Unbiased\ estimation, & (pd_i - td_i) = 0 \\ Underestimation, & (pd_i - td_i) < 0 \end{cases} \tag{12}$$

where pd_i and td_i denote the predicted value and the true (actual) value of ith cloud resource usage as the same as Eqs. (10) and (11), respectively. The overestimation means that the predicted usage amount of cloud resources is more than the actual one, the unbiased estimation indicates that the predicted usage amount just matches with the actual one, and the underestimation denotes that the predicted usage amount is lower than the actual one.

In Fig. 15, for the relatively-periodic real-world test dataset, we analyzed the usage value errors of cloud resources (CPU, Memory, Disk and BW) according to the predicted results in Fig. 10. The input data prior to the prediction is 50 different number of requested workloads on the continuous 50 hourly predicted time points, i.e., the final 50 h of requested workload data as shown in the top part of Fig. 1. Most of the usage amount errors (deviations) of cloud resources are within ±1%. The usage amount errors of Memory and Disk are close to 0, i.e., near to the state of the unbiased estimation. For the cases that the usage amount errors of CPU and BW are beyond ±1%, there are certain number of the overestimation and the underestimation. In the case of the overestimation, for reducing the usage cost of running applications and improving resource utilization, we can appropriately decrease the reserved amount of cloud resources for users according to the percentage of resource usage amount errors. Conversely, in the case of the underestimation,

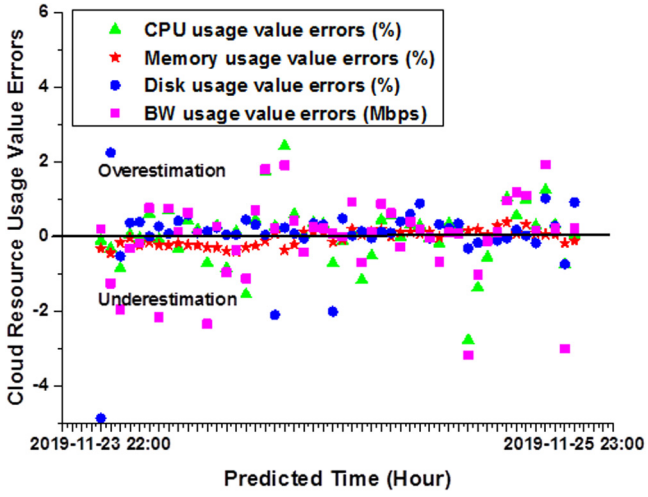

Fig. 15. The error estimation analysis of predicted cloud resource usage amount on the relatively-periodic real-world test dataset that contains 50 different number of requested workloads on the continuous 50 hourly predicted time points.

the suitable amount of cloud resources should be increased to guarantee the running performance of applications on the basis of the percentage of resource usage amount errors, i.e., adding the reserved amount of cloud resources for users or enhancing the provisioning amount for providers.

In Fig. 16 for the randomly-changing Google cluster test dataset, we conducted the similar analysis on the usage value errors of cloud resources (CPU, Memory and Disk) in light of the predicted results in Fig. 14. Before performing the prediction, we input the data that is 100 different number of requested workloads on the continuous 100 hourly predicted time points,

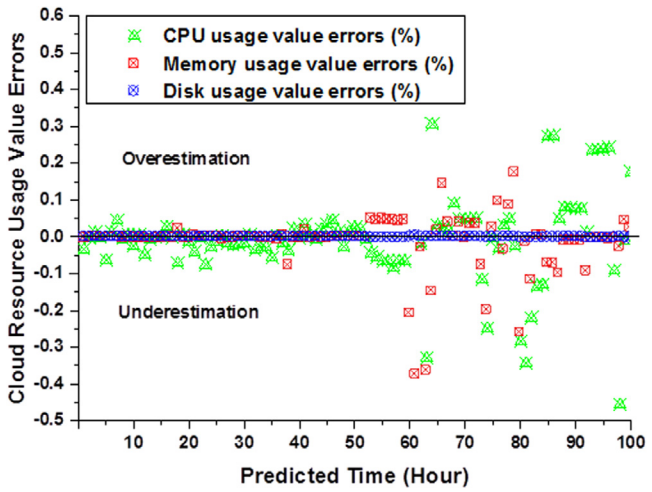

Fig. 16. The error estimation analysis of predicted cloud resource usage amount on the randomly-changing Google cluster test dataset that contains 100 different number of requested workloads on the continuous 100 hourly predicted time points.

i.e., the final 100 h of requested workload data as shown in the top part of Fig. 2. The usage amount errors of Disk is close to 0, i.e., near to the state of the unbiased estimation. Those of CPU and Memory are within ±0.05% from 0 to 60 h, after that, most of the usage amount errors are within ±0.35%. For the cases that the usage amount errors of CPU and Memory exceed ±0.05%, some quantity of the overestimation and the underestimation also exist. In terms of these cases, we can take the measures similar to Fig. 15 to adjust the reserved amount of cloud resources.

7. Conclusions and future work

There are some challenges in current cloud resource usage predictions such as the continuous changes of requested workloads and respective cloud resource usage, the difficulties in predicting the usage amount of cloud resources according to requested workloads, and the time-lagging prediction. Toward this end, we propose an online cloud resource prediction model (OCRPM) to timely and efficiently predict the reasonable resource usage amount according to the historical and current resource consumption queried in real time. This model includes the requested workload trend classification (RWTC) that generates three types of workload trend waveform patterns (Peak, Steadiness and Trough) in hourly time intervals, the scalable window waveform sampling (SWWS) on classified workloads, and the optimal error gradient boosting regression (OEGBR) training and prediction. Through the theoretical demonstration and the extensive experiments on different types of datasets, the proposed method can reduce the fitting error better and improve the accuracy of prediction results compared with the latest relevant approaches and existing statistical learning regression models. Future work will add some requirement parameters of users in the cloud resource prediction, and further realize the adaptive reservation and provisioning of cloud resources according to the resource usage prediction results.

CRediT authorship contribution statement

Xiaogang Wang: Conceptualization, Methodology, Software, Formal analysis, Writing - original draft. **Jian Cao:** Method guidance, Writing - review & editing. **Dingyu Yang:** Software, Validation. **Zhen Qin:** Data curation, Running environment, Visualization. **Rajkumar Buyya:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is partially supported by National Natural Science Foundation of China under grant 61702320, "Teacher Professional Development" Project of Shanghai Municipal Education Commission of China in 2019, and ARC Discovery Project of Australia. The work was carried out within the Melbourne CLOUDS Laboratory at the University of Melbourne.

References

[1] O. Adam, Y.C. Lee, A.Y. Zomaya, Stochastic resource provisioning for containerized multi-tier web services in clouds, IEEE Trans. Parallel Distrib. Syst. 28 (7) (2017) 2060–2073, http://dx.doi.org/10.1109/TPDS.2016.2639009.

[2] R. Buyya, C.S. Yeo, S. Venugopal, Market-oriented cloud computing: Vision, hype, and reality for delivering IT services as computing utilities, in: Proc. 20th Int. Conf. High Performance Computing and Communications, 2008, pp. 5–13, http://dx.doi.org/10.1109/HPCC.2008.172.

[3] Z. Chen, J. Hu, G. Min, A.Y. Zomaya, T. El-Ghazawi, Towards accurate prediction for high-dimensional and highly-variable cloud workloads with deep learning, IEEE Trans. Parallel Distrib. Syst. 31 (4) (2020) 923–934, http://dx.doi.org/10.1109/TPDS.2019.2953745.

[4] Z. Wang, M.M. Hayat, N. Ghani, K.B. Shaban, Optimizing cloud-service performance: Efficient resource provisioning via optimal workload allocation, IEEE Trans. Parallel Distrib. Syst. 28 (6) (2017) 1689–1702, http://dx.doi.org/10.1109/TPDS.2016.2628370.

[5] C. Reiss, J. Wilkes, J.L. Hellerstein, Google Cluster-Usage Traces: Format + Schema, Technical Report, Google Inc., Mountain View, CA, USA, 2011, Revised 2014-11-17 for version 2.1. Posted at https://github.com/google/cluster-data.

[6] J. Guo, Z. Chang, S. Wang, H. Ding, Y. Feng, L. Mao, Y. Bao, Who limits the resource efficiency of my datacenter: An analysis of alibaba datacenter traces, in: Proc. the Int. Symp. Quality of Service, 2019, pp. 1–10, http://dx.doi.org/10.1145/3326285.3329074.

[7] Servinet, https://www.servinet.cn/en/, 2019.

[8] J. Kumar, A.K. Singh, Workload prediction in cloud using artificial neural network and adaptive differential evolution, Future Gener. Comput. Syst. 81 (2018) 41–52, http://dx.doi.org/10.1016/j.future.2017.10.047.

[9] B. Xia, T. Li, Q. Zhou, Q. Li, H. Zhang, An effective classification-based framework for predicting cloud capacity demand in cloud services, IEEE Trans. Serv. Comput. (2018) http://dx.doi.org/10.1109/TSC.2018.2804916.

[10] H. Wang, J. Panneresselvam, L. Liu, Y. Lu, X. Zhai, H. Ali, Cloud workload analytics for real-time prediction of user request patterns, in: Proc. 20th Int. Conf. High Performance Computing and Communications, 2018, pp. 1677–1684, http://dx.doi.org/10.1109/HPCC/SmartCity/DSS.2018.00272.

[11] Y. Hu, B. Deng, F. Peng, D. Wang, Workload prediction for cloud computing elasticity mechanism, in: Proc. IEEE Int. Conf. Cloud Computing and Big Data Analysis, 2016, pp. 244–249, http://dx.doi.org/10.1109/ICCCBDA.2016.7529565.

[12] K. Mason, M. Duggan, E. Barrett, J. Duggan, E. Howley, Predicting host CPU utilization in the cloud using evolutionary neural networks, Future Gener. Comput. Syst. 86 (2018) 162–173, http://dx.doi.org/10.1016/j.future.2018.03.040.

[13] G. Kaur, A. Bala, I. Chana, An intelligent regressive ensemble approach for predicting resource usage in cloud computing, J. Parallel Distrib. Comput. 123 (2019) 1–12, http://dx.doi.org/10.1016/j.jpdc.2018.08.008.

[14] L. Wei, B. He, C.H. Foh, Towards multi-resource physical machine provisioning for IaaS clouds, in: Proc. IEEE Int. Conf. Communications, 2014, pp. 3469–3472, http://dx.doi.org/10.1109/ICC.2014.6883858.

[15] L. Wei, C.H. Foh, B. He, J. Cai, Towards efficient resource allocation for heterogeneous workloads in iaas clouds, IEEE Trans. Cloud Comput. 6 (1) (2018) 264–275, http://dx.doi.org/10.1109/TCC.2015.2481400.

[16] M. Baughman, R. Chard, L.T. Ward, J. Pitt, K. Chard, I.T. Foster, Profiling and predicting application performance on the cloud, in: Proc. IEEE/ACM 11th Int. Conf. Utility and Cloud Computing, 2018, pp. 21–30.

[17] J. Scheuner, P. Leitner, Estimating cloud application performance based on micro-benchmark profiling, in: Proc. IEEE 11th Int. Conf. Cloud Comput, 2018, pp. 90–97, http://dx.doi.org/10.1109/CLOUD.2018.00019.

[18] X. Wang, J. Cao, Y. Xiang, OSPN: Optimal service provisioning with negotiation for bag-of-tasks applications, IEEE Trans. Serv. Comput. (2017) http://dx.doi.org/10.1109/TSC.2017.2787707.
[19] X. Wang, J. Cao, Y. Xiang, Dynamic cloud service selection using an adaptive learning mechanism in multi-cloud computing, J. Syst. Softw. 100 (2015) 195–210, http://dx.doi.org/10.1016/j.jss.2014.10.047.
[20] P. Bodík, R. Griffith, C.A. Sutton, A. Fox, M.I. Jordan, D.A. Patterson, Statistical machine learning makes automatic control practical for internet datacenters, in: Proc. IEEE Workshop on Hot Topics in Cloud Computing, 2009, pp. 1–5.
[21] J. Dejun, G. Pierre, C.-H. Chi, Resource provisioning of web applications in heterogeneous clouds, in: Proc. the 2nd USENIX Conference on Web Application Development, 2011, pp. 49–60.
[22] J. Bi, H. Yuan, W. Tan, M. Zhou, Y. Fan, J. Zhang, J. Li, Application-aware dynamic fine-grained resource provisioning in a virtualized cloud data center, IEEE Trans. Autom. Sci. Eng. 14 (2) (2017) 1172–1184, http://dx.doi.org/10.1109/TASE.2015.2503325.
[23] W. Iqbal, A. Erradi, A. Mahmood, Dynamic workload patterns prediction for proactive auto-scaling of web applications, J. Netw. Comput. Appl. 124 (2018) 94–107, http://dx.doi.org/10.1016/j.jnca.2018.09.023.
[24] N. Roy, A. Dubey, A. Gokhale, Efficient autoscaling in the cloud using predictive models for workload forecasting, in: Proc. IEEE 4th Int. Conf. Cloud Comput, 2011, pp. 500–507, http://dx.doi.org/10.1109/CLOUD.2011.42.
[25] W. Fang, Z. Lu, J. Wu, Z. Cao, RPPS: A novel resource prediction and provisioning scheme in cloud data center, in: 2012 IEEE Ninth International Conference on Services Computing, 2012, pp. 609–616, http://dx.doi.org/10.1109/SCC.2012.47.
[26] D. Ardagna, S. Casolari, M. Colajanni, B. Panicucci, Dual time-scale distributed capacity allocation and load redirect algorithms for cloud systems, J. Parallel Distrib. Comput. 72 (6) (2012) 796–808, http://dx.doi.org/10.1016/j.jpdc.2012.02.014.
[27] A. Khan, X. Yan, S. Tao, N. Anerousis, Workload characterization and prediction in the cloud: A multiple time series approach, in: Proc. IEEE Network Operations and Management Symposium, 2012, pp. 1287–1294, http://dx.doi.org/10.1109/NOMS.2012.6212065.
[28] J. Panneerselvam, L. Liu, N. Antonopoulos, InOt-RePCoN: Forecasting user behavioural trend in large-scale cloud environments, Future Gener. Comput. Syst. 80 (2018) 322–341, http://dx.doi.org/10.1016/j.future.2017.05.022.
[29] X. Tang, X. Liao, J. Zheng, X. Yang, Energy efficient job scheduling with workload prediction on cloud data center, Cluster Comput. 21 (3) (2018) 1581–1593, http://dx.doi.org/10.1007/s10586-018-2154-7.
[30] C. Qiu, H. Shen, L. Chen, Probabilistic demand allocation for cloud service brokerage, in: Proc. 35th Annual IEEE Int. Conf. Computer Communications, 2016, pp. 1–9, http://dx.doi.org/10.1109/INFOCOM.2016.7524611.
[31] C. Qiu, H. Shen, Dynamic demand prediction and allocation in cloud service brokerage, IEEE Trans. Cloud Comput. (2019) http://dx.doi.org/10.1109/TCC.2019.2913419.
[32] Y. Yang, L. Zhao, Z. Li, L. Nie, P. Chen, K. Li, ElaX: Provisioning resource elastically for containerized online cloud services, in: Proc. 21th Int. Conf. High Performance Computing and Communications, 2019, pp. 1987–1994, http://dx.doi.org/10.1109/HPCC/SmartCity/DSS.2019.00274.
[33] D. Marinescu, A. Paya, J.P. Morrison, A cloud reservation system for big data applications, IEEE Trans. Parallel Distrib. Syst. 28 (3) (2017) 606–618, http://dx.doi.org/10.1109/TPDS.2016.2594783.
[34] F. Xu, H. Zheng, H. Jiang, W. Shao, H. Liu, Z. Zhou, Cost-effective cloud server provisioning for predictable performance of big data analytics, IEEE Trans. Parallel Distrib. Syst. 30 (5) (2019) 1036–1051, http://dx.doi.org/10.1109/TPDS.2018.2873397.
[35] J. Marques, R.R. Obelheiro, Escada: Predicting virtual machine network bandwidth demands for elastic provisioning in IaaS clouds, in: Proc. IEEE Int. Conf. Cloud and Autonomic Computing, 2017, pp. 10–21, http://dx.doi.org/10.1109/ICCAC.2017.9.
[36] C. Hauser, S. Wesner, Reviewing cloud monitoring: Towards cloud resource profiling, in: Proc. IEEE 11th Int. Conf. Cloud Comput, 2018, pp. 678–685, http://dx.doi.org/10.1109/CLOUD.2018.00093.
[37] C. Liu, C. Liu, Y. Shang, S. Chen, B. Cheng, J. Chen, An adaptive prediction approach based on workload pattern discrimination in the cloud, J. Netw. Comput. Appl. 80 (2017) 35–44, http://dx.doi.org/10.1016/j.jnca.2016.12.017.
[38] S.-u.-R. Baig, W. Iqbal, J.L. Berral, A. Erradi, D. Carrera, Adaptive prediction models for data center resources utilization estimation, IEEE Trans. Netw. Serv. Manag. 16 (4) (2019) 1681–1693, http://dx.doi.org/10.1109/TNSM.2019.2932840.
[39] G. Chandrashekar, F. Sahin, A survey on feature selection methods, Comput. Electric. Eng. 40 (1) (2014) 16–28, http://dx.doi.org/10.1016/j.compeleceng.2013.11.024.
[40] B. Gregorutti, B. Michel, P. Saint-Pierre, Correlation and variable importance in random forests, Statist. Comput. 27 (3) (2017) 659–678, http://dx.doi.org/10.1007/s11222-016-9646-1.
[41] Zabbix, https://www.zabbix.com/cn/download, 2019.
[42] LinearModels, https://scikit-learn.org/stable/modules/linear_model.html, 2020.
[43] KernelRidgeRegression, https://scikit-learn.org/stable/modules/kernel_ridge.html, 2020.
[44] SupportVectorRegression, https://scikit-learn.org/stable/modules/svm.html, 2020.

Xiaogang Wang received the Ph.D. degree in computer science and technology from Shanghai Jiao Tong University, China, in 2018. He is currently an Associate Professor in the School of Electronics and Information at Shanghai Dianji University, Shanghai, China. He was also the Visiting Research Scholar of the CLOUDS Laboratory in the School of Computing and Information Systems at the University of Melbourne, Australia, from Sep. 2019 to Sep. 2020. He has published over 20 papers in some journals and conferences such as IEEE Transactions on Services Computing, Journal of Systems and Software, Applied Intelligence, WI-IAT, APSCC, CSCWD, and ICSAI. His main research interests include cloud computing, service computing, multi-agent systems, and data analysis and processing. He is a member of the IEEE and a member of the China Computer Federation.

Jian Cao received the Ph.D. degree from Nanjing University of Science and Technology in 2000. He is currently a Professor in the Department of Computer Science and Engineering at Shanghai Jiao Tong University. His main research interests include service computing, cloud computing, cooperative information systems and software engineering. He has published more than 150 papers in prestigious journals, such as IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Mobile Computing, Journal of Systems and Software, Future Generation Computer Systems, and Journal of Super Computing. He is a senior member of the IEEE and a distinguished member of the China Computer Federation.

Dingyu Yang received the B.E. and M.E. degrees from Kunming University of Science and Technology, and the Ph.D. degree from Shanghai Jiao Tong University, all in computer science. He is currently a Senior Engineer at Alibaba Group, Hangzhou, China. His research interests include resource prediction, anomaly detection in cloud computing and distributed stream processing. He has published over 10 papers in some journals and conference such as Journal of Systems and Software, VLDB, and VLDBJ.

Zhen Qin worked as a Field Support Manager at Sandvik (China) Group, Beijing, China, from 2004 to 2013, served as the Technical Director at Shanghai Norgin System Co., Ltd, Shanghai, China, from 2013 to 2015, and is currently the Technical Director at Shanghai Servinet Technology Co., Ltd, Shanghai, China, since 2015. His main work includes IT physical infrastructure, network infrastructure, enterprise network security, public cloud, and enterprise digital transformation support.

Rajkumar Buyya is a Redmond Barry Distinguished Professor and Director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne, Australia. He is also serving as the founding CEO of Manjrasoft, a spin-off company of the University, commercializing its innovations in Cloud Computing. He served as a Future Fellow of the Australian Research Council during 2012–2016. He received the Ph.D. degree in Computer Science and Software Engineering from Monash University, Melbourne, Australia, in 2002. He has authored over 625 publications and seven text books. He is one of the highly cited authors in computer science and software engineering worldwide (h-index=132, g-index=294, 92,500+ citations). He is recognized as a "Web of Science Highly Cited Researcher" for four consecutive years since 2016, and Scopus Researcher of the Year 2017 with Excellence in Innovative Research Award by Elsevier

for his outstanding contributions to Cloud computing and distributed systems. He has led the establishment and development of key community activities, including serving as foundation Chair of the IEEE Technical Committee on Scalable Computing and five IEEE/ACM conferences. These contributions and international research leadership of him are recognized through the award of "2009 IEEE Medal for Excellence in Scalable Computing" from the IEEE Computer Society TCSC. He served as the founding Editor-in-Chief of the IEEE Transactions on Cloud Computing. He is currently serving as Co-Editor-in-Chief of Journal of Software: Practice and Experience. He is a fellow of the IEEE.