Hindawi
Journal of Electrical and Computer Engineering
Volume 2019, Article ID 2782349, 14 pages
https://doi.org/10.1155/2019/2782349

Research Article

A Hybrid Method for Short-Term Host Utilization Prediction in Cloud Computing

Jing Chen [1,2] and Yinglong Wang [2]

[1] *College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China*
[2] *Shandong Provincial Key Laboratory of Computer Networks, Shandong Computer Science Center (National Supercomputer Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences), Jinan 250014, China*

Correspondence should be addressed to Yinglong Wang; wangylscsc@126.com

Received 24 June 2018; Accepted 17 February 2019; Published 14 March 2019

Academic Editor: Sos S. Agaian

Dynamic resource scheduling is a critical activity to guarantee quality of service (QoS) in cloud computing. One challenging problem is how to predict future host utilization in real time. By predicting future host utilization, a cloud data center can place virtual machines to suitable hosts or migrate virtual machines in advance from overloaded or underloaded hosts to guarantee QoS or save energy. However, it is very difficult to accurately predict host utilization in a timely manner because host utilization varies very quickly and exhibits strong instability with many bursts. Although machine learning methods can accurately predict host utilization, it usually takes too much time to ensure rapid resource allocation and scheduling. In this paper, we propose a hybrid method, EEMD-RT-ARIMA, for short-term host utilization prediction based on ensemble empirical mode decomposition (EEMD), runs test (RT), and autoregressive integrated moving average (ARIMA). First, the EEMD method is used to decompose the nonstationary host utilization sequence into relatively stable intrinsic mode function (IMF) components and a residual component to improve prediction accuracy. Then, efficient IMF components are selected and then reconstructed into three new components to reduce the prediction time and error accumulation due to too many IMF components. Finally, the overall prediction results are obtained by superposing the prediction results of three new components, each of which is predicted by the ARIMA method. An experiment is conducted on real host utilization traces from a cloud platform. We compare our method with the ARIMA model and the EEMD-ARIMA method in terms of error, effectiveness, and time-cost analysis. The results show that our method is a cost-effective method and is more suitable for short-term host utilization prediction in cloud computing.

1. Introduction

Cloud computing assembles a large number of computing, storage, and network resources into a data center. These resources are cut and allocated efficiently to satisfy users' resource demands through virtualization technology. In addition to rich resources, cloud computing also provides a pay-as-you-go model. Users can rent various resources as they demand, which reduces their costs. These characteristics of rich resources, on-demand resource provision, and low costs prompt cloud computing to be widely applied in various domains. However, it is still a challenge to allocate and schedule resources effectively to improve resource utilization and guarantee QoS.

The general process of resource allocation and scheduling in cloud computing is shown in Figure 1. When a new virtual machine (VM) request is initiated, the cloud data center selects a suitable physical host to allocate resources for this VM according to a specified resource allocation policy. This policy can maximize resource utilization per host to minimize the number of active hosts or balancing resource utilization of all active hosts. Whichever policy you use, it is important to know the future host utilization for the selection of a suitable host. Additionally, VM migration is also an effective method for resource scheduling. When the host utilization exceeds a predefined threshold, the performance of VMs running on this host will decrease. It will not

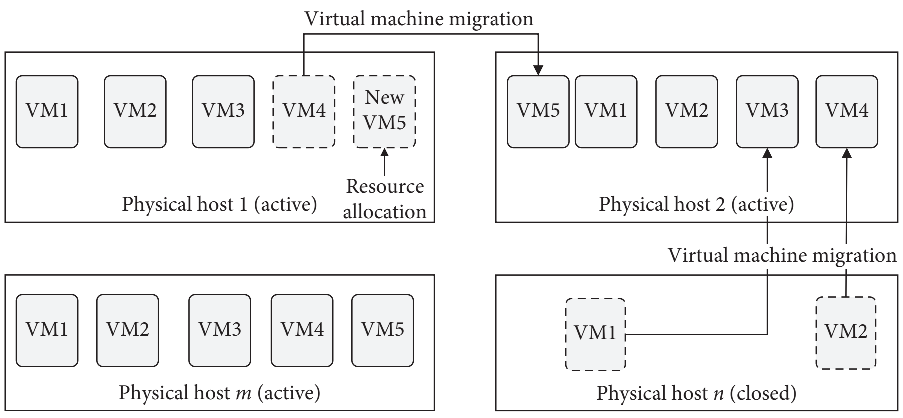

Figure 1: Resource allocation and scheduling process.

guarantee the QoS of applications running on these VMs. Therefore, it is necessary to migrate some hotspot VMs from one overloaded host to other nonoverloaded hosts. Similarly, if the host utilization is below a predefined threshold, all VMs on this host will be migrated to other hosts. Thus, this host can be closed to reduce energy consumption.

VM migration is a reactive method that cannot be initiated until the host is overloaded or underloaded. Therefore, it is very important to detect when the host is overloaded or underloaded. Most existing approaches monitor host utilization to determine its state. If its resource utilization exceeds a predefined threshold during an observation period, this host is overloaded. If its resource utilization is always below a predefined threshold during an observation period, it is declared underloaded. Basically, it usually takes some time to migrate VMs from an overloaded host to other hosts. If the host utilization changes faster than the provision time of the resources, users will suffer poor QoS until resources are available. In addition, host underload detection based on a single host utilization value also leads to unnecessary VM migration and stability problems. These problems can be addressed via proactive methods that actively predict short-term host utilization to allocate resources in advance. For example, if the host utilization within the future 15 minutes is always over 80%, this host will be overloaded. Therefore, the VMs should be migrated in advance from this host to other hosts to ensure QoS. If the host utilization within the current and future 1 hour is always below 15%, this host is underloaded and should be closed to save energy after VM migration. However, a large number of random resource demands and concurrent access to applications cause stochastic volatility of host utilization. They change very fast and exhibit strong instability with many bursts. It is difficult to predict short-term host utilization in a timely and accurate manner based on such data.

Although some machine learning methods, such as a neural network (NN) [1, 2], support vector regression (SVR) [3], and backpropagation neural networks (BPNN) [4], achieve good prediction accuracy in cloud computing, they require too much time to train a model to allocate resources rapidly. Line regression (LR) can implement prediction more quickly than ARIMA, but it demands that the training data have simpler behaviors. ARIMA is a prediction model for nonstationary time series, but it is not suitable if a large amount of random variation exists in the data. In our previous work [5], we proposed a resource demand prediction method EEMD-ARIMA that combines the EEMD method and ARIMA model to predict future resource demands. This method first uses the EEMD method to decompose the original resource demand sequence into multiple IMF components and the residual (R) component. Next, we forecast the future values of each component by the ARIMA model. Finally, the overall forecasting results are obtained by superposing the forecasting results of each component. Although this method alleviates random variation of resource demands and improves prediction accuracy by combining EEMD and ARIMA methods, two problems arise. One is the prediction error accumulation caused by the superposition of ARIMA prediction of all components. The ARIMA prediction of each component decomposed by the EEMD method generates a certain prediction error. The superposition of the prediction results of all components leads to the prediction error accumulation. Another is the high time cost due to EEMD decomposition and ARIMA prediction of too many decomposition components. The ARIMA prediction of each component takes some time. Thus, the total time of the ARIMA prediction of multiple components greatly increases compared with a single ARIMA prediction of the original sequence.

To solve these problems of the EEMD-ARIMA method, this paper further proposes a hybrid method, EEMD-RT-ARIMA, for short-term host utilization prediction that not only further improves prediction accuracy by combining the EEMD method with the ARIMA model but also reduces time cost by selecting and reconstructing efficient IMF components. The comparison and evaluation are made among our EEMD-RT-ARIMA method, ARIMA model, and EEMD-ARIMA method in terms of error, effectiveness, and time-cost analysis.

2. Related Works

Many studies have been conducted on various predictions in cloud computing. From the perspective of research objectives, some researchers have studied server load prediction

[6–10], VM load prediction [11, 12], VM utilization prediction [13, 14], host utilization prediction [15], web application workload prediction [16], cloud service workload prediction [17–19], workflow workload prediction [20], service quality prediction [21], and workload characterization [22–24]. Toumi et al. [6] described a server load according to the submitted task types and the submission rate and applied a stream mining technique to predict server loads. Jheng et al. [11] proposed a VM workload prediction method based on the gray forecasting model, which determines the migrated VMs according to power savings and workload balance. Dabbagh et al. [13] proposed a prediction approach that uses Wiener filters to predict the future resource utilization of VMs. Mason et al. [15] predicted host CPU utilization for a short time using evolutionary neural networks, which showed a high prediction accuracy and a high degree of generality. In this paper, we focus on host utilization prediction using EEMD and ARIMA methods to not only improve prediction accuracy but also reduce prediction time as much as possible.

From the perspective of approaches, prediction methods are usually divided into two categories. One is based on machine learning methods. Tseng et al. [25] proposed a prediction method for CPU and memory utilization of VMs and physical machines based on a genetic algorithm (GA), which precedes the gray model under stable tendency and unstable tendency in terms of prediction accuracy. Shyam and Manvi [26] proposed a short- and long-term prediction model of virtual resource requirements for CPU/memory-intensive applications based on Bayesian networks, where the relationships and dependencies between variables are identified to facilitate resource prediction. Lu et al. [27] proposed a workload prediction model RVLBPNN (Rand Variable Learning Rate Backpropagation Neural Network) based on BPNN algorithm, which achieves higher prediction accuracy than the hidden Markov model and the naive Bayes classifier. This method not only predicts CPU-intensive and memory-intensive workloads but also improves prediction accuracy by using the intrinsic relations among the arriving cloud workloads. Rajaram and Malarvizhi [28] compared the prediction accuracies of a few machine learning methods, such as LR, SVR, and multiplayer perceptron. Li and Zhang [29] proposed an optimal combination prediction method for resource demands, which combines the induced ordered weighted geometry averaging operator and the generalized dice coefficient with the improved Elman neural network and gray model to enhance the prediction accuracy. Minarolli and Freisleben [30] presented a cross-correlation prediction approach based on support vector machine (SVM), which considers the cross relation of VMs running the same application to improve prediction accuracy. Zhang et al. [31] proposed a deep belief network- (DBN-) based prediction approach of cloud resource requests in which orthogonal experimental design and analysis of variance are used to enhance the prediction accuracy. Compared with the ARIMA model, this method greatly reduces mean square error (MSE) by over 60% for CPU and RAM request predictions. Although machine learning methods are effective in improving prediction accuracy, they are complex and usually demand a large number of data to extract features and train a model. It requires too much time for the prediction to guarantee QoS of the running applications. Cloud computing requires a simple and rapid host utilization prediction method to support resource allocation and scheduling.

Another method is based on statistical methods, such as Brown's quadratic exponential smoothing method [32], autoregressive integrated moving average (ARIMA) model [33–35], and the kernel canonical correlation analysis [36]. Tran et al. [37] applied the ARIMA model in the long-term prediction of server workload, while our method aims to predict short-term host utilization. It is more difficult because host utilization can be extremely random and nonstationary in a short time. Calheiros et al. [33] proposed a short-term prediction model of cloud workload using the ARIMA model and evaluated the prediction accuracy and its impact on user applications' QoS. They suggested that users' behaviors must be considered to reflect real conditions in workload simulation. Our method combines the ARIMA model with EEMD and RT methods to improve prediction accuracy and reduce prediction time as much as possible. It is compared with EEMD-ARIMA and ARIMA methods in terms of error, effectiveness, and time-cost analysis.

Moreover, some studies combine the ARIMA model with other techniques to improve prediction accuracy. Xu et al. [38] constructed a model GFSS-ANFIS/SARIMA combining the seasonal ARIMA model with the generalized fuzzy soft sets and adaptive neuro-fuzzy inference system. This model improves the prediction accuracy of resource demands. Li et al. [39] proposed a workload predictor combined with ARIMA and dynamic error compensation to reduce the service-level agreement (SLA) default rate. Fu and Zhou [40] proposed a predicted affinity model to implement VM placement, which uses the resource demands predicted by the ARIMA model to calculate a VM-host affinity value. Jiang et al. [41] presented a self-adaptive ensemble prediction method for cloud resource demands, which uses a two-level ensemble method to predict VM demands based on a historic time series. This method not only combines multiple prediction methods: moving average (MA), autoregressive (AR), artificial neural network (ANN), gene expression programming (GEP), and SVM but also adjusts the weight of each method adaptively to obtain the best average performance according to the relative errors. In contrast, our method uses the EEMD method to deal with the nonstationary host utilization and then selects and reconstructs efficient components to improve prediction accuracy and reduce the time cost. The EEMD proposed by Wu and Huang [42] is an effective noise-aided method that can handle nonlinear and nonstationary time series. It has been widely used in wind speed forecasting [43, 44], aircraft auxiliary power unit (APU) degradation prediction [45], turbine fault trend prediction [46], and rolling bearing fault diagnosis [47]. It has shown a good effect on enhancing the prediction accuracy. Our method also uses EEMD to decompose the nonstationary host utilization for improving the prediction accuracy and further uses correlation coefficients, RT values, and average periods to select and reconstruct efficient components for reducing prediction error accumulation and prediction time.

3. Background

3.1. Empirical Mode Decomposition (EMD). EMD is a method of signal processing that can decompose a signal into multiple IMFs and an R trend item [48]. Two conditions must be satisfied for an IMF:

(1) The number of extrema and zero-crossings must either be identical or differ by at most one
(2) The mean value of the envelopes of the local maxima and the local minima must be zero

EMD includes the following steps:

Step 1. Make $f(t) = x(t)$, where $x(t)$ is given as the original data.

Step 2. Find all the local maxima and minima of $f_i(t)$, where i is the loop times and its initial value is 1. Interpolate between the local maxima and minima to obtain an upper envelope and a lower envelope and then compute the mean value $m_i(t)$ of these envelopes.

Step 3. Compute the new component $h_i(t) = f_i(t) - m_i(t)$.

Step 4. Verify whether $h_i(t)$ satisfies the abovementioned two conditions for an IMF. If it does not, make $f_{i+1}(t) = h_i(t)$ and repeat steps 2 and 3. If it satisfies the condition, $h_i(t)$ is regarded as the first IMF component $p_1(t)$, where $p_1(t) = h_i(t)$. Then, compute the R component by the formula $r_1(t) = x(t) - p_1(t)$.

Step 5. Repeat step 1–4 with $r_1(t)$ as the new data until the R is a monotonic function. Thus, $x(t)$ is decomposed into n IMFs and an R as follows:

$$x(t) = \sum_{i=1}^{n} p_i(t) + r(t). \tag{1}$$

3.2. Ensemble Empirical Mode Decomposition (EEMD). The EMD method has a noticeable drawback of mode mixing that can cause signal intermittency. Wu and Huang proposed a new method named ensemble empirical mode decomposition (EEMD) to solve this problem. Compared with the EMD method, the EEMD method first executes the decomposition process k times. Each time, it adds a different white noise to the signal and then decomposes the new signal. Generally, the k iterations are set as an integer in the range [50, 100], and the standard deviation d of the white noise is set as a value in the range [0.1, 0.2]. Next, k groups of decomposition results are obtained. Each group includes n IMFs $p_{mi}(t)\,(i = 1, \ldots, n)$ and an R $r_m(t)$, where m denotes the group number. Finally, the mean values of these groups of IMFs and Rs are calculated as the final IMFs $\overline{p}_i(t)\,(i = 1, \ldots, n)$ and the R $\overline{r}(t)$:

$$\begin{aligned}\overline{p}_i(t) &= \frac{\sum_{m=1}^{k} p_{mi}(t)}{k},\\ \overline{r}(t) &= \frac{\sum_{m=1}^{k} r_m(t)}{k}.\end{aligned} \tag{2}$$

The IMF components have three main characteristics.

(1) Completeness: the total of all IMFs and the R have the same feature as the original data.
(2) Orthogonality: each IMF with a certain physical meaning is independent and has no effect on other IMFs. The product of any two IMFs equals 0 in mathematics.
(3) Adaptability: an IMF with a higher frequency is decomposed from the original data faster than those with low frequencies. The frequencies of IMFs reflect the features of the original data.

3.3. Runs Test (RT). RT is a nonparametric test method that checks the randomness of a sequence with only two symbols or two values, such as + and – and 0 and 1. An RT is defined as a sequence with successive symbols (0 or 1). For example, a data sequence "111100000111110001100 10" includes 8 runs, 4 of which involve successive "1" and the others involve successive "0." RT can also be used to test a time series.

Assume that $M = <m_1(t), \ldots, m_i(t), \ldots, m_n(t)>$ denotes a time series, where $m_i(t)$ is an element of this time series and n is the total number of elements. The mean value of these elements is calculated by the following formula:

$$\overline{M} = \frac{1}{n}\sum_{i=1}^{n} m_i(t). \tag{3}$$

Then, the element of this time series can be denoted as follows:

$$G_i = m_i(t) - \overline{M} = \begin{cases} 1, & m_i(t) \geq \overline{M},\\ 0, & m_i(t) < \overline{M}. \end{cases} \tag{4}$$

Thus, this time series is transformed into a sequence with a series of 0 and 1, in which the elements are independent and identically distributed. The total number of RT reflects the fluctuation of the sequence.

4. A Hybrid Method for Short-Term Host Utilization Prediction

To improve prediction accuracy and reduce prediction time of the EEMD-ARIMA method, we propose a hybrid method, EEMD-RT-ARIMA, for short-term host utilization prediction as shown in Figure 2. First, the host utilization sequence is decomposed into multiple IMF components and the R component using the EEMD method. Next, we calculate the correlation coefficients between IMF components and the original data sequence to select the efficient IMF components. Then, we use RT values and average periods to reconstruct these efficient IMF and R components into three new components: high-frequency and strong-

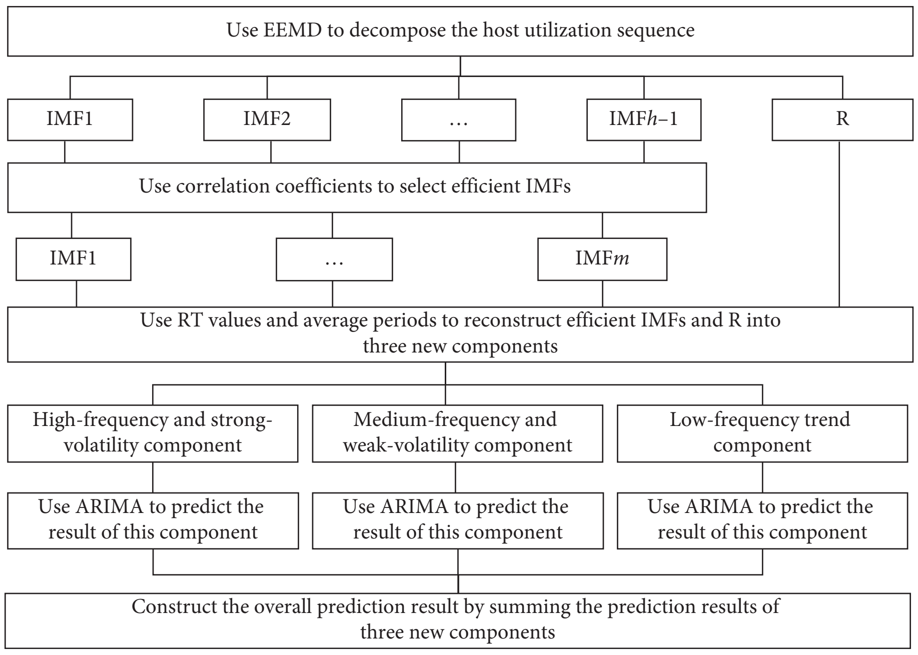
Use EEMD to decompose the host utilization sequence
IMF1
IMF2
...
IMFh−1
R
Use correlation coefficients to select efficient IMFs
IMF1
...
IMFm
Use RT values and average periods to reconstruct efficient IMFs and R into three new components
High-frequency and strong-volatility component
Medium-frequency and weak-volatility component
Low-frequency trend component
Use ARIMA to predict the result of this component
Use ARIMA to predict the result of this component
Use ARIMA to predict the result of this component
Construct the overall prediction result by summing the prediction results of three new components

Figure 2: EEMD-RT-ARIMA method.

volatility component, medium-frequency and weak-volatility component, and low-frequency trend component. Then, we use the ARIMA method to predict the results of three new components. Finally, the overall prediction results are achieved by summing the prediction results of the three new components.

The key to our EEMD-RT-ARIMA method is to select and reconstruct efficient components. Compared with the EEMD-ARIMA method, the number of its components involving in ARIMA prediction is reduced. Thus, the EEMD-RT-ARIMA method can reduce the prediction error accumulation and the total prediction time by reducing the number of components. Obviously, both the EEMD-RT-ARIMA method and EEMD-ARIMA method have a higher prediction time than the ARIMA model from their implementation processes. However, our EEMD-RT-ARIMA method focuses on cost-effectiveness, which has a trade-off among prediction accuracy, effectiveness, and time cost.

4.1. Use of EEMD to Decompose the Host Utilization Sequence. A host utilization sequence is classified into different categories according to the CPU, memory, and disk, such as CPU utilization sequence $T_{\text{cpu}} = \{c_1, \ldots, c_i, \ldots, c_n\}$. The CPU utilization sequence is usually random and unstable owing to random and sudden resource demands in cloud computing. It is necessary to transform such data into relatively stationary data to improve prediction accuracy. The EEMD method appears to be more effective in processing nonlinear and nonstationary data sequences than other decomposition algorithms. Therefore, we use the EEMD method to decompose the host utilization sequence and obtain a series of the IMF$_i$ components and the R component.

A running example shows the nonstationary CPU utilization trace of a physical host from our cloud platform. We divide it into the training set (673 data points) and the testing set (24 data points) in Figure 3. Then, we use the EEMD method to decompose the training set and obtain IMF1-IMF8 components and the R component. They are shown from the high frequency to low frequency in Figure 4.

4.2. Calculation of the Correlation Coefficients to Select Efficient IMF Components. A correlation coefficient measures the correlation between two sequences. We calculate the correlation coefficient $P_j(X,Y)$ between the IMF$_j$ component and the original training set based on the following formula, where $\text{cov}(X,Y)$ is a covariance between the sequences X and Y and $\text{Var}(X)$ and $\text{Var}(Y)$ are the variances of the sequence X and the sequence Y:

$$P_j(X,Y) = \frac{\text{Cov}(X,Y)}{\sqrt{\text{Var}(X)}\sqrt{\text{Var}(Y)}}. \quad (5)$$

Then, the correlation coefficient $P_j(X,Y)$ is checked to determine whether it is negative. If it is negative, the IMF$_j$ component is inefficient and dropped. If it is not negative, the IMF$_j$ component is efficient and reserved.

We calculate the correlation coefficient between each IMF component and the original training set. Only IMF6 and IMF7 have negative correlation coefficients of −0.08 and −0.15. Hence, they are dropped. IMF1–IMF5 and IMF8 are selected as efficient IMF components.

4.3. Reconstruction of Efficient IMFs and R into New Components. Each IMF component actually reflects a certain physical feature of the original data. If some IMF components are closer in terms of frequency and amplitude fluctuation, then they have similar features. Thus, they can be reconstructed into a new component with these typical

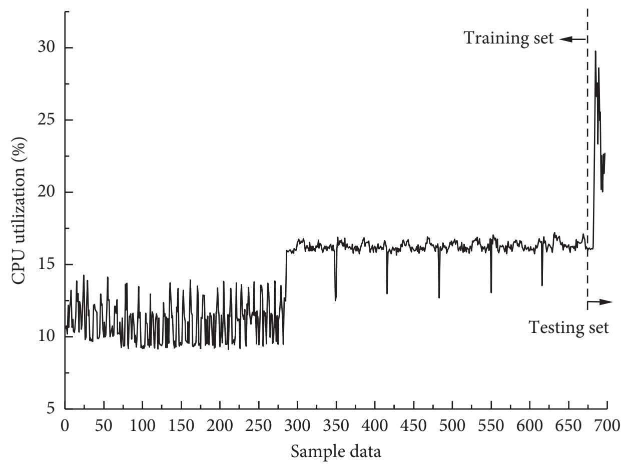

Figure 3: CPU utilization trace of a physical host.

features. Thus, the prediction error accumulation and the prediction time of the EEMD-ARIMA method can be reduced by reducing the number of components.

The average period reflects the frequency of host utilization variation. There exists a reciprocal relation between them. The smaller the average period, the higher the frequency. If the average periods of IMF components are closer, they are closer in frequency. The average period is calculated by the following formula, in which n is the number of the training set and l_j is the number of extrema:

$$T_j = \frac{n}{l_j}. \quad (6)$$

Similarly, the RT value reflects the trend of amplitude fluctuation. If the RT value is larger, the amplitude volatility is stronger. If the RT values of the two IMFs are closer, the overall trend of the two IMFs is similar in amplitude volatility.

To enhance the prediction accuracy and reduce the prediction time of the EEMD-ARIMA method, we reconstruct the IMF components and the R component into three new components according to their average periods and RT values in the EEMD-RT-ARIMA method. Because the average period and RT value have different units, we normalize the average period T_j as follows:

$$T_{nj} = \frac{T_j - T_{\min}}{T_{\max} - T_{\min}}, \quad (7)$$

where T_{nj} denotes the normalized average period of the IMF$_j$ component. $T_{\max}$ and $T_{\min}$ represent the maximum and minimum of the average periods of all IMF components. Similarly, the RT value R_j can be normalized as follows:

$$R_{nj} = \frac{R_j - R_{\min}}{R_{\max} - R_{\min}}, \quad (8)$$

where R_{nj} is the normalized value of R_j. $R_{\max}$ and $R_{\min}$ are the maximum and minimum of all RT values. Thus, the reconstruction factor (RF) is defined as follows:

$$F_j = \alpha \cdot \left(1 - T_{nj}\right) + \beta \cdot R_{nj}. \quad (9)$$

An IMF component is higher in frequency and stronger in volatility, and its RF value is greater. If the RF values of the two IMF components are closer, their overall trends are more similar. Thus, they can be reconstructed into a new component. All efficient IMF and R components are reconstructed into three new components: high-frequency and strong-volatility component, medium-frequency and weak-volatility component, and low-frequency trend component. The high-frequency and strong-volatility component reflects the strong volatility and randomness of the high-frequency part of the original host utilization sequence. The medium-frequency and weak-volatility component shows the detailed features of the volatility of the original host utilization sequence. The low-frequency trend component depicts the overall trend of the volatility of the original host utilization sequence.

Table 1 shows the RT values, average periods, and RF values of efficient IMF and R components. The RF values of IMF1 and IMF2 are large and relatively close, while the RF values of IMF8 and R are equal to 0. The RF values of IMF3–IMF5 are close. Therefore, we reconstructed IMF1-IMF2, IMF3–IMF5, and IMF8-R into three new components, as shown in Figures 5(a)–5(c). They separately reflect the randomness, the fluctuation details, and the overall trend of the original host utilization sequence.

4.4. Use of the ARIMA Model to Predict the Future Host Utilization.

We use the ARIMA model to predict the future results for each new component. Then, the overall prediction results are obtained by superposing the prediction results of each new component. The ARIMA prediction is described as follows (Algorithm 1).

For example, we assume that three new components C_{h}, C_{m}, and C_{l} are obtained, which represent the high-frequency and strong-volatility component, medium-frequency and weak-volatility component, and low-frequency trend component, respectively. Then, we use the ARIMA method to predict the future 24-point values for each new component. The prediction results P_{h}, P_{m}, and P_{l} of three new components can be described in the following formulas, each of which includes the values of the predicting 24-point data:

$$\begin{aligned} P_{\mathrm{h}} &= \left(f_{\mathrm{h}}^{1}, f_{\mathrm{h}}^{2}, \ldots, f_{\mathrm{h}}^{24}\right), \\ P_{\mathrm{m}} &= \left(f_{\mathrm{m}}^{1}, f_{\mathrm{m}}^{2}, \ldots, f_{\mathrm{m}}^{24}\right), \\ P_{\mathrm{l}} &= \left(f_{\mathrm{l}}^{1}, f_{\mathrm{l}}^{2}, \ldots, f_{\mathrm{l}}^{24}\right). \end{aligned} \quad (10)$$

Finally, we calculate the overall prediction result P by superposing the prediction results of each new component as follows:

$$P = P_{\mathrm{h}} + P_{\mathrm{m}} + P_{\mathrm{l}}. \quad (11)$$

From this process of the EEMD-RT-ARIMA method, we find that the number of components decreases from 9 to 3, which can reduce the total prediction time and the error accumulation of the component prediction compared with the EEMD-ARIMA method.

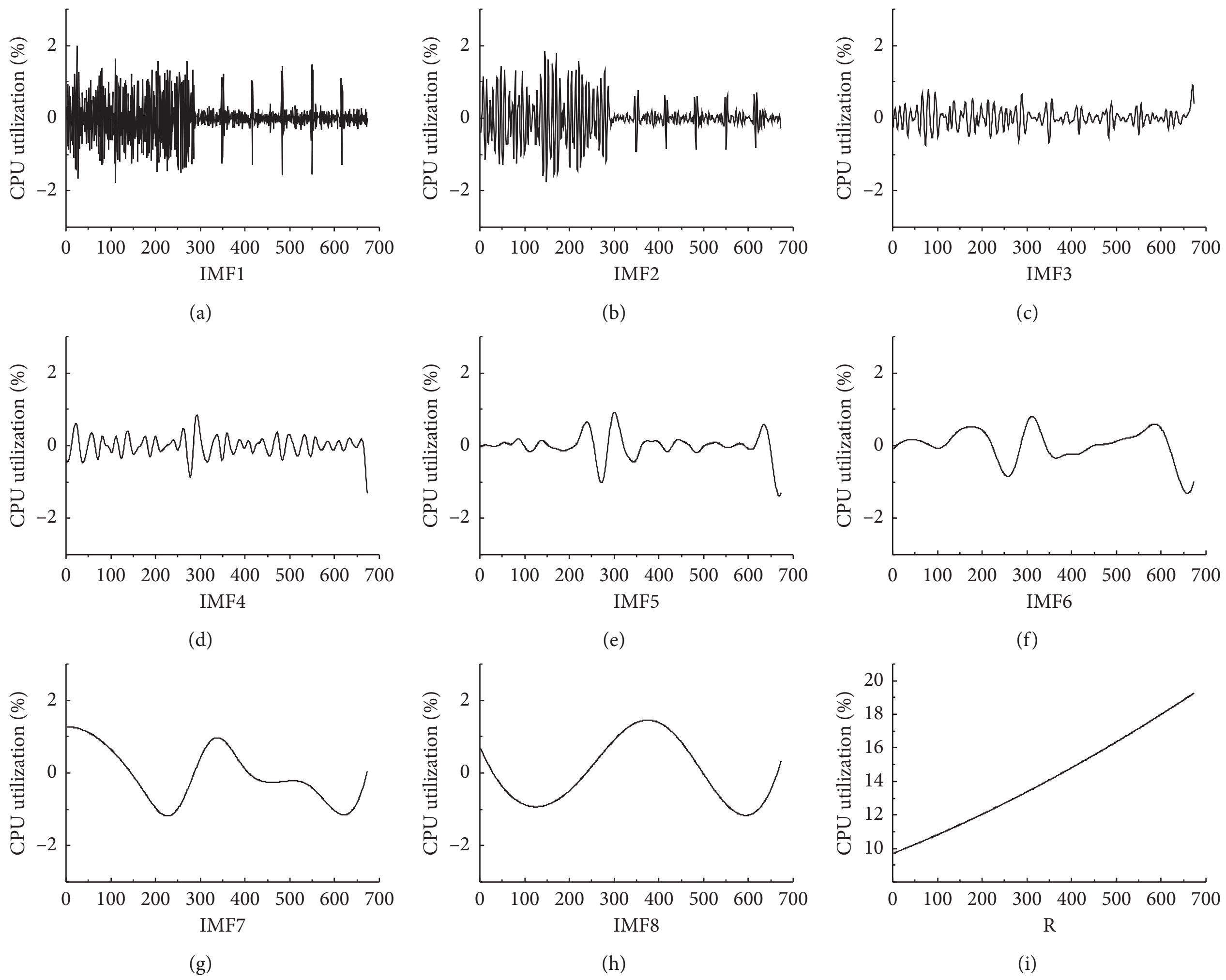

Figure 4: Decomposition results of EEMD.

Table 1: RT, average periods, and RF of efficient IMF and R components.

	IMF1	IMF2	IMF3	IMF4	IMF5	IMF8	R
RT	431	181	96	47	21	5	2
Average period	2.12	8.31	16.41	30.59	67.3	673	673
RF	1	0.70	0.60	0.53	0.47	0	0

5. Experimental Setup

We conduct an experiment to evaluate our method. The experimental dataset and measurement metrics are introduced as follows.

5.1. Experimental Dataset. Host utilization mainly involves in CPU utilization, memory utilization, network utilization, and disk utilization. In this paper, we mainly focus on host CPU utilization. We randomly select CPU utilization traces of 7 physical hosts from the dataset released by Alibaba in August 2017 [49], each of which includes 144 points (5 minutes per point). These traces are all time-dependent sequences as shown in Figures 6(a)–6(g).

Each sequence is divided into a training set and a testing set. We first use the training set to predict the future data, and then, these predicting data are compared with those actual data in the testing set to evaluate our method. In this paper, each training set is set as the first 120 points, and the testing set is defined as the subsequent points, such as 6 points, 12 points, and 24 points. We set the number of iterations $k = 50$ and the standard deviation $d = 0.2$ in EEMD decomposition.

5.2. Measurement Metrics. We evaluate our method in terms of error, effectiveness, and time-cost analysis as follows.

5.2.1. Error Analysis. To evaluate our method, we use the mean absolute percentage error (MAPE) to reflect the prediction accuracy. MAPE is defined as follows:

$$\text{MAPE} = \frac{1}{m}\sum_{i=1}^{m}\left(\frac{x_i^f - x_i^t}{x_i^t}\right) * 100\%, \quad (12)$$

where x_i^f denotes the value of the prediction point, x_i^t denotes the actual value in the testing set, and m denotes the

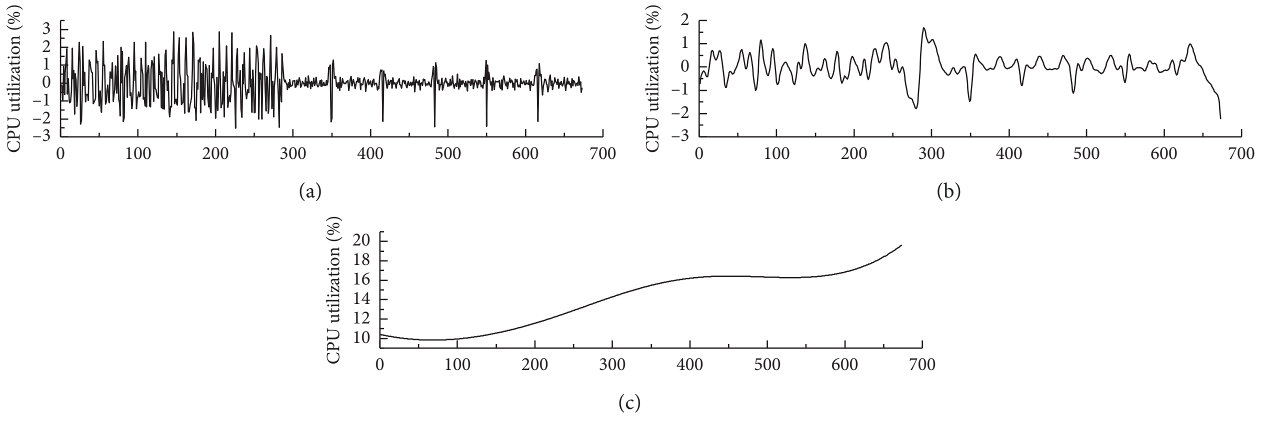

Figure 5: New components. (a) High-frequency and strong-volatility component. (b) Medium-frequency and weak-volatility component. (c) Low-frequency trend component.

number of prediction points. It is obvious that the prediction accuracy is higher when the MAPE is lower.

5.2.2. Effectiveness Analysis. Host utilization underprediction or overprediction can lead to resource underprovision or overprovision. Resource underprovision cannot guarantee applications' QoS, while resource overprovision can cause resource waste and low resource utilization. Therefore, a good prediction method should avoid underprediction and overprediction. In particular, underprediction should be avoided as much as possible because it results in a lower QoS to users.

We set up the positive and negative errors to reflect the overprediction and underprediction and then use them to evaluate the effectiveness of our method. A good prediction method should have a smaller negative error to avoid underprediction. The positive and negative prediction errors are calculated by the following formula, where p_i is the predicting data, r_i is the actual data and m is the number of underprediction data (i.e., negative deviation) or overprediction data (i.e., positive deviation):

$$e_i = \frac{1}{m}\sum_{i=1}^{m}\left|p_i - r_i\right|. \tag{13}$$

5.2.3. Time Cost Analysis. Host utilization varies very quickly in a cloud data center. If host utilization prediction is slower than the determination of VM migration, resource provision will be delayed, which can cause poor QoS. Thus, host utilization prediction must be completed in a timely manner. To investigate the time cost of our proposed method, we test the running time of the EEMD-RT-ARIMA method and compared it with other prediction methods according to the following index t_c:

$$t_c = \frac{t_{\text{our}} - t_{\text{other}}}{t_{\text{other}}} * 100\%, \tag{14}$$

where t_{our} indicates the running time of our method EEMD-RT-ARIMA and t_{other} represents the running time of other methods, such as the ARIMA model or the EEMD-ARIMA method. t_c denotes the percent of the reduced or increased time cost.

6. Experimental Results and Analysis

To validate the prediction effectiveness of our EEMD-RT-ARIMA method, we conduct experiments on ARIMA, EEMD-ARIMA, and EEMD-RT-ARIMA methods and compare their predictive results. All experiments were performed on a PC with 2.5 GHz Intel (R) i7 CPU running MATLAB. To make three methods comparable, we use the same original dataset to execute it 5 times for each method. The mean values of the prediction results are shown in the following tables and figures.

6.1. Error Analysis. Table 2 shows the MAPE values of host utilization predictions for 7 physical hosts. We can see that EEMD-ARIMA and EEMD-RT-ARIMA methods have lower MAPE values than ARIMA models for 6-point and 12-point predictions. For example, EEMD-ARIMA and EEMD-RT-ARIMA methods achieve MAPE values of 6.06% and 5.05% for the 6-point prediction of host 109, while the ARIMA model has a far higher MAPE value (up to 16.85%). They obtain MAPE values of 10.13% and 5.46% for the 12-point prediction of host 109, while the ARIMA model obtains 11.08%. For host 22, both the EEMD-ARIMA and EEMD-RT-ARIMA methods obtain far lower MAPE values of 5.31% and 5.42% than the 10.66% of the ARIMA model for 6-point prediction. Similarly, they also obtain better effectiveness on 12-point prediction. The same situation also exists in 6-point and 12-point predictions of other hosts. This indicates that both EEMD-ARIMA and EEMD-RT-ARIMA methods have higher prediction accuracy than ARIMA models in 6-point and 12-point predictions for host utilization. EEMD reduces the inherent volatility of the host utilization sequence, which improves the prediction accuracy of the EEMD-ARIMA and EEMD-RT-ARIMA methods. However, the situation changes in 24-point prediction. The MAPE values of hosts 1162, 424, 1060, and 237 are all over 30% using these three methods. Although the EEMD-RT-ARIMA method has lower MAPE values than

```
(1) For each new component
(2) Set the order of difference $d = 0$
(3) Execute the augmented Dickey-Fuller (ADF) test. If it is a stationary time series, go to step 5; else, go to step 4 until it is stationary
(4) Difference the time series and set $d = d + 1$, go to step 3
(5) Determine the order of the ARIMA model using Bayesian information criterion (BIC)
(6) Estimate the parameters of the ARIMA model using the maximum likelihood
(7) Forecast the future $n$ values of this new component using the ARIMA model
(8) End
(9) Obtain the overall prediction results by superposing the prediction results of each new component
```

Algorithm 1: ARIMA prediction.

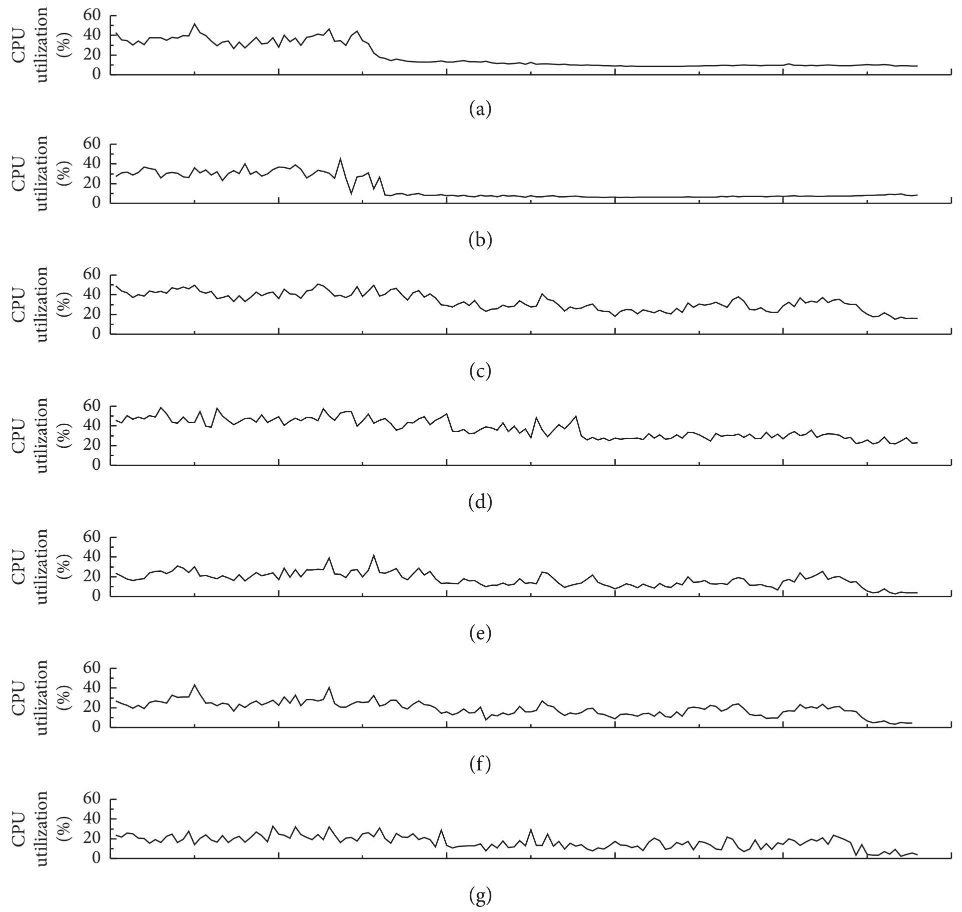

Figure 6: CPU utilization traces of 7 physical hosts. (a) Host 839. (b) Host 109. (c) Host 1162. (d) Host 22. (e) Host 424. (f) Host 1060. (g) Host 237.

the ARIMA and EEMD-ARIMA methods for hosts 839 and 109, it has far higher MAPE values in the 24-point prediction than those of 6-point and 12-point predictions. This shows that the EEMD-ARIMA and EEMD-RT-ARIMA methods are not suitable for long-term but suitable for short-term prediction.

For further analysis, we find that the EEMD-RT-ARIMA method achieves lower prediction error than the EEMD-ARIMA method for the 6-point and 12-point predictions of hosts 839, 109, and 1162, although the EEMD-RT-ARIMA method only selects efficient IMF components. However, it is the opposite for hosts 22, 424, 1060, and 237. The original CPU utilization sequences of all physical hosts are identical in frequency, so we calculate the RT value of each CPU utilization sequence shown in Table 3. Hosts 839, 109, and 1162 achieve lower RT values under 10, which shows that their CPU utilization is more stationary than other hosts. Smaller RT values indicate more stationary host utilization sequences. This phenomenon can also be seen in Figures 6(a)–6(c). From Tables 2 and 3, it can be found that the EEMD-RT-ARIMA method achieves a lower MAPE value than the EEMD-ARIMA method if the RT value is smaller. Conversely, the EEMD-RT-ARIMA method has a higher MAPE value than the EEMD-ARIMA method if the

Table 2: MAPE values of host utilization prediction.

Host ID	Prediction length	ARIMA (%)	EEMD-ARIMA (%)	EEMD-RT-ARIMA (%)
839	6-point	9.77	4.73	4.61
	12-point	13.78	10.48	5.03
	24-point	23.10	21.53	8.82
109	6-point	16.85	6.06	5.05
	12-point	11.08	10.13	5.46
	24-point	41.34	24.68	8.36
1162	6-point	24.45	7.76	6.45
	12-point	17.81	13.14	8.95
	24-point	21.93	41.41	33.45
22	6-point	10.66	5.31	5.42
	12-point	8.22	5.33	5.37
	24-point	11.41	18.44	14.77
424	6-point	37.54	16.12	17.65
	12-point	21.74	16.59	17.46
	24-point	88.68	77.39	115.7
1060	6-point	35.71	10.05	13.94
	12-point	37.60	17.44	19.40
	24-point	74.72	54.82	91.93
237	6-point	22.29	7.85	11.78
	12-point	25.51	11.30	15.93
	24-point	126.11	158.43	189.51

Table 3: RT values of each host utilization.

Host ID	839	109	1162	22	424	1060	237
RT value	2	6	10	16	22	24	38

RT value is larger. For instance, hosts 839, 109, and 1162 with smaller RT values obtain lower MAPE values using the EEMD-RT-ARIMA method than the EEMD-ARIMA method, while hosts 22, 424, 1060, and 237 with larger RT values obtain higher MAPE values using the EEMD-RT-ARIMA method than the EEMD-ARIMA method.

Furthermore, the difference in the MAPE values between EEMD-RT-ARIMA and EEMD-ARIMA methods decreases with the increase in the RT values from host 839 to host 1162. Their difference changes to negative from host 22, which indicates that the EEMD-ARIMA method has higher prediction accuracy than the EEMD-RT-ARIMA method. Then, their difference becomes larger as the RT values increase. The 12-point host CPU utilization prediction illustrates this situation. For example, host 839, with an RT value of 2, has a MAPE of 5.03% for 12-point prediction by using the EEMD-RT-ARIMA method, which is 5.45% lower than the 10.48% of the EEMD-ARIMA method. The MAPE value of the EEMD-RT-ARIMA method is only 4.19% lower than that of the EEMD-ARIMA method for host 1162, with an RT value of 10. For host 22 with an RT value of 16, the EEMD-RT-ARIMA method has a slightly higher MAPE of 5.37% than 5.33% of the EEMD-ARIMA method. With the increase of the RT value, the differences of MAPE values between EEMD-RT-ARIMA and EEMD-ARIMA methods further increase to 0.87%, 1.96%, and 4.63% for hosts 424, 1060, and 237, respectively. This indicates that the EEMD-RT-ARIMA method is less effective than the EEMD-ARIMA method in CPU utilization prediction for these hosts. Undoubtedly, the ARIMA prediction of each component decomposed by the EEMD method generates a certain error. The superposition of the prediction results of each component causes error accumulation. The EEMD-RT-ARIMA method reduces the error accumulation by selecting and reconstructing the efficient IMF components into fewer components, so it achieves better prediction accuracy than the EEMD-ARIMA method for hosts 839, 109, and 1162. Certainly, the selection and reconstruction of efficient IMF components also cause a certain prediction error due to the absence of nonefficient components, especially for nonstationary host utilization sequences. When this kind of prediction error exceeds the error accumulation of ARIMA prediction of all components in the EEMD-ARIMA method, the EEMD-RT-ARIMA method is no more effective than the EEMD-ARIMA method for the nonstationary CPU utilization prediction of some hosts, such as hosts 22, 424, 1060, and 237.

6.2. Effectiveness Analysis. To verify the effectiveness of our method in short-term prediction, we select the experimental results of hosts 839, 22, and 237 with the minimum, middle, and maximum RT values for further analysis. Figure 7 shows the prediction results of the EEMD-RT-ARIMA, ARIMA, and EEMD-ARIMA methods. We find that the future resource utilization of host 839 decreases below 11%. According to a predefined policy, host 839 is underloaded and can be closed to save energy. Figure 7 shows that our method is more accurate and effective than the ARIMA model. In particular, our method tends to change with the trend of data variation, while the ARIMA model cannot keep up with it. Our method is more suitable for handling nonstationary time series than the ARIMA model. Additionally, the predicting data using the EEMD-RT-ARIMA method are closer to the testing data than those of the EEMD-ARIMA method for host 839. These

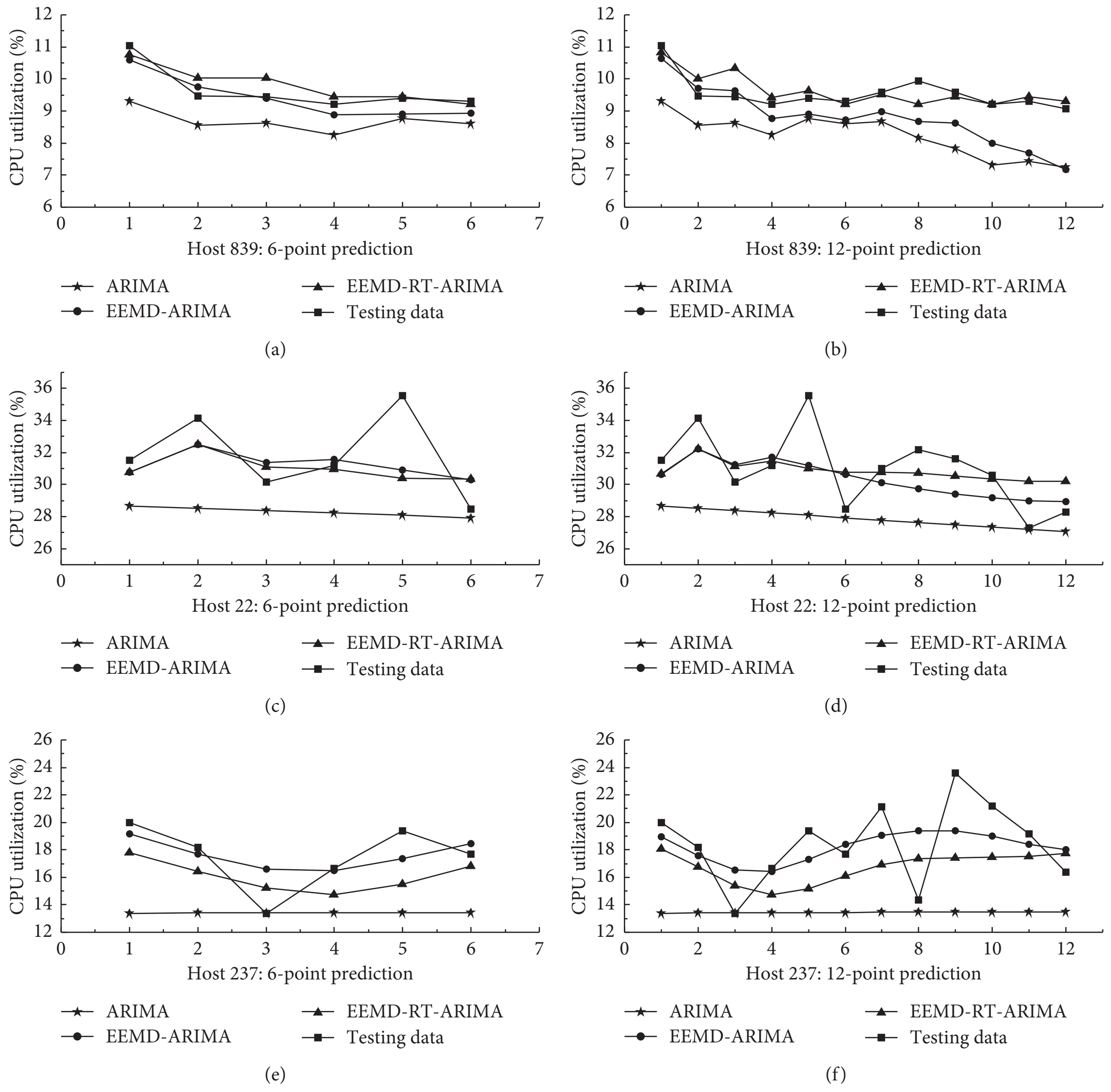

Figure 7: Prediction of the results of different methods.

results show that the EEMD-RT-ARIMA method is more effective than the EEMD-ARIMA method for CPU utilization sequences with weak fluctuations. When the host utilization sequence shows stronger fluctuation, the absence of non-efficient IMF components will greatly influence the prediction results. The EEMD-RT-ARIMA method is no more effective than the EEMD-ARIMA method for CPU utilization prediction of host 237.

To further analyze the effectiveness of our method, we calculate the positive and negative errors for 6-point and 12-point predictions of these hosts shown in Table 4. When the negative error is smaller, the prediction method is more suitable for cloud resource provision because of avoiding underprediction. It can be observed that most of the prediction results of the ARIMA model are underpredicted (the cells of positive error are all “null” for hosts 839 and 22). Furthermore, the negative errors of the ARIMA model are all far higher than those of other methods for host 237. For instance, the ARIMA model has a high negative error of up to 27.51% for the 12-point prediction of host 237, while the EEMD-ARIMA and EEMD-RT-ARIMA methods only have negative errors of 8.00% and 8.92%, respectively. If the ARIMA model is used to predict future host utilization, it can cause resource underprovision, which cannot ensure applications’ QoS. The EEMD-RT-ARIMA method achieves smaller negative errors than the EEMD-ARIMA method for hosts 839 and 22, while it has a larger negative error for hosts 237. For instance, the EEMD-ARIMA method achieves the negative error of 10.71% for the 12-point prediction of host 839, while EEMD-RT-ARIMA only achieves the negative error of 4.74%. Similarly, the EEMD-ARIMA method obtains a negative error of 6.09% for the 12-point prediction of host 22, while the EEMD-RT-ARIMA method achieves a lower value of only 5.05%. However, the situation changes

Table 4: Positive and negative error analysis.

Host ID	Prediction length	ARIMA		EEMD-ARIMA		EEMD-RT-ARIMA	
		Positive error (%)	Negative error (%)	Positive error (%)	Negative error (%)	Positive error (%)	Negative error (%)
839	6-point	Null	9.77	3.16	5.85	4.92	4.15
	12-point	Null	13.78	3.43	10.71	4.80	4.74
22	6-point	Null	10.66	3.91	6.70	4.12	6.34
	12-point	Null	9.72	4.26	6.09	5.39	5.05
237	6-point	0.39	26.68	14.31	4.62	13.81	11.37
	12-point	0.39	27.51	17.44	8.00	17.90	8.92

for host 237. The EEMD-ARIMA method has a smaller negative error than the EEMD-RT-ARIMA method. For instance, the EEMD-RT-ARIMA method obtains negative errors of 11.37% and 8.92% for 6-point and 12-point predictions, while the EEMD-ARIMA method only has negative errors of 4.62% and 8.00%.

6.3. Time-Cost Analysis. To verify the applicability of our method, we further compare the time cost of these methods in Figure 8. The running time of the EEMD-ARIMA method is the largest by over 180 s while the ARIMA model takes the least time at less than 50 s. The EEMD-RT-ARIMA method time cost is between 70 s and 117 s, which decreases the time cost by 40%–80% compared with the EEMD-ARIMA method. For example, the running time of the EEMD-RT-ARIMA method is 69.37 s, far less than the 337.20 s of the EEMD-ARIMA method for the 6-point prediction of host 22. Our method saves up to 80% of the time cost. For the CPU utilization sequence of host 237 with strong variability, it requires 190.46 s to predict the future 6-point values using the EEMD-ARIMA method, while it only takes 113.64 s using the EEMD-RT-ARIMA method. The running time is reduced by approximately 40%. Considering the prediction accuracy, effectiveness and time cost, our EEMD-RT-ARIMA method is more cost-effective for short-term host utilization prediction in cloud computing.

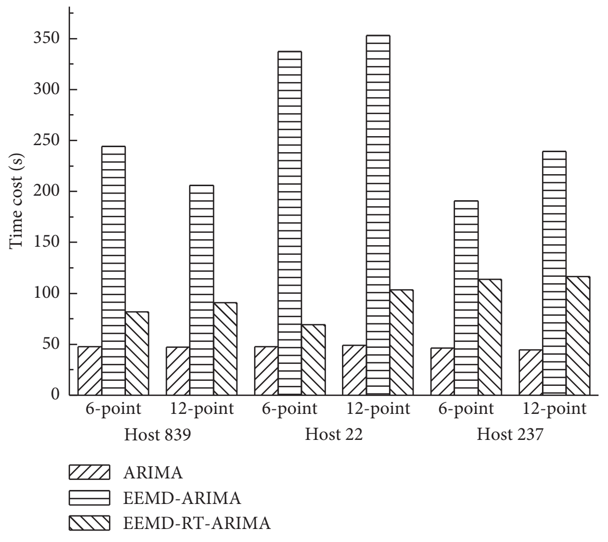

Figure 8: Time cost of different prediction methods.

7. Conclusions

Host utilization is an indicator of host performance, whose prediction can promote effective resource scheduling in cloud computing. However, host utilization demonstrates strong randomness and instability caused by users' random and various resource demands. It is difficult to improve prediction accuracy. In this paper, we propose a hybrid and cost-effective method, EEMD-RT-ARIMA, for short-term host utilization prediction in cloud computing. The EEMD method is first used to decompose the nonstationary host utilization sequence into a few relatively stationary IMF components and an R component. Then, we calculate the correlation coefficient between each IMF component and the original data to select efficient IMF components and use RT values and average periods to reconstruct these components into three new components to reduce error accumulation and time cost. Finally, three new components are predicted by the ARIMA model, and their prediction results are superposed to form the overall prediction results. We use the real host utilization traces from a cloud platform to conduct the experiments and compare our EEMD-RT-ARIMA method with the ARIMA model and EEMD-ARIMA method in terms of error, effectiveness, and time-cost analysis. The results show that our method is cost-effective and is more suitable for short-term host utilization prediction in cloud computing.

Data Availability

The running example and experimental data used to support the findings of this study have been deposited in the Figshare repository (https://doi.org/10.6084/m9.figshare.7679594).

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the Shandong Provincial Natural Science Foundation (ZR2016FM41).

References

[1] J. J. Prevost, K. Nagothu, B. Kelley, and M. Jamshidi, "Prediction of cloud data center networks loads using stochastic and neural models," in *Proceedings of 2011 6th International Conference on System of Systems Engineering*, pp. 276–281, IEEE, Albuquerque, NM, USA, June 2011.

[2] M. Borkowski, S. Schulte, and C. Hochreiner, "Predicting cloud resource utilization," in *Proceedings of 9th International Conference on Utility and Cloud Computing (UCC)*, pp. 37–42, IEEE, Shanghai, China, December 2016.

[3] M. Barati and S. Sharifian, "A hybrid heuristic-based tuned support vector regression model for cloud load prediction," *Journal of Supercomputing*, vol. 71, no. 11, pp. 4235–4259, 2015.

[4] Z. Chen, Y. Zhu, Y. Di, S. Feng, and J. Geng, "A high-accuracy self-adaptive resource demands predicting method in IaaS cloud environment," *Neural Network World*, vol. 25, no. 5, pp. 519–540, 2015.

[5] J. Chen and Y. Wang, "A resource demand prediction method based on EEMD in cloud computing," *Procedia Computer Science*, vol. 131, pp. 116–123, 2018.

[6] H. Toumi, Z. Brahmi, Z. Benarfa, and M. M. Gammoudi, "Server load prediction using stream mining," in *Proceedings of 2017 International Conference on Information Networking (ICOIN)*, pp. 653–661, IEEE, Da Nang, Vietnam, January 2017.

[7] S. Di, D. Kondo, and W. Cirne, "Host load prediction in a Google compute cloud with a Bayesian model," in *Proceedings of 2012 International Conference on High Performance Computing, Networking, Storage and Analysis (SC'12)*, pp. 1–11, IEEE, Salt Lake City, UT, USA, November 2012.

[8] N. K. Gondhi and P. Kailu, "Prediction based energy efficient virtual machine consolidation in cloud computing," in *Proceedings of 2015 Second International Conference on Advances in Computing and Communication Engineering*, pp. 437–441, IEEE, Dehradun, India, May 2015.

[9] A. Verma, G. Dasgupta, T. K. Nayak, P. De, and R. Kothari, "Server workload analysis for power minimization using consolidation," in *Proceedings of the 2009 Conference on USENIX Annual Technical Conference*, p. 28, USENIX Association, San Diego, CA, USA, June 2009.

[10] B. Song, Y. Yu, Y. Zhou, Z. Wang, and S. Du, "Host load prediction with long short-term memory in cloud computing," *Journal of Supercomputing*, vol. 74, no. 12, pp. 6554–6568, 2018.

[11] J.-J. Jheng, F.-H. Tseng, H.-C. Chao, and L.-D. Chou, "A novel VM workload prediction using grey forecasting model in cloud data center," in *Proceedings of International Conference on Information Networking 2014 (ICOIN2014)*, pp. 40–45, IEEE, Phuket, Thailand, February 2014.

[12] A. Beloglazov and R. Buyya, "Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 7, pp. 1366–1379, 2013.

[13] M. Dabbagh, B. Hamdaoui, M. Guizani, and A. Rayes, "An energy-efficient VM prediction and migration framework for overcommitted clouds," *IEEE Transactions on Cloud Computing*, vol. 6, no. 4, pp. 955–966, 2018.

[14] D. Minarolli, A. Mazrekaj, and B. Freisleben, "Tackling uncertainty in long-term predictions for host overload and underload detection in cloud computing," *Journal of Cloud Computing*, vol. 6, no. 1, p. 4, 2017.

[15] K. Mason, M. Duggan, E. Barrett, J. Duggan, and E. Howley, "Predicting host CPU utilization in the cloud using evolutionary neural networks," *Future Generation Computer Systems*, vol. 86, pp. 162–173, 2018.

[16] D. Magalhães, R. N. Calheiros, R. Buyya, and D. G. Gomes, "Workload modeling for resource usage analysis and simulation in cloud computing," *Computers & Electrical Engineering*, vol. 47, pp. 69–81, 2015.

[17] C. Tian, Y. Wang, Y. Luo et al., "Minimizing content reorganization and tolerating imperfect workload prediction for cloud-based video-on-demand services," *IEEE Transactions on Services Computing*, vol. 9, no. 6, pp. 926–939, 2016.

[18] M. Verma, G. R. Gangadharan, V. Ravi, and N. Narendra, "Resource demand prediction in multi-tenant service clouds," in *Proceedings of 2013 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, pp. 1–8, IEEE, Bangalore, India, October 2013.

[19] W. Zhang, Y. Shi, L. Liu, L. Cui, and Y. Zheng, "Performance and resource prediction at high utilization for n-tier service systems in cloud an experiment driven approach," in *Proceedings of 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, pp. 843–848, IEEE, Liverpool, UK, October 2015.

[20] G. Kecskemeti, A. Kertesz, and Z. Nemeth, "Cloud workload prediction by means of simulations," in *Proceedings of the Computing Frontiers Conference on ZZZ (CF'17)*, pp. 279–282, ACM, Siena, Italy, May 2017.

[21] Y. Chen and Z.-A. Jiang, "Dynamically predicting the quality of service: batch, online, and hybrid algorithms," *Journal of Electrical and Computer Engineering*, vol. 2017, Article ID 9547869, 10 pages, 2017.

[22] A. Khan, X. Yan, S. Tao, and N. Anerousis, "Workload characterization and prediction in the cloud: a multiple time series approach," in *Proceedings of 2012 IEEE Network Operations and Management Symposium*, pp. 1287–1294, IEEE, Maui, HI, USA, April 2012.

[23] A. K. Mishra, J. L. Hellerstein, W. Cirne, and C. R. Das, "Towards characterizing cloud backend workloads," *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, no. 4, pp. 34–41, 2010.

[24] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper, "Workload analysis and demand prediction of enterprise data center applications," in *Proceedings of 2007 IEEE 10th International Symposium on Workload Characterization*, pp. 171–180, IEEE, Boston, MA, USA, September 2007.

[25] F.-H. Tseng, X. Wang, L.-D. Chou, H.-C. Chao, and V. C. M. Leung, "Dynamic resource prediction and allocation for cloud data center using the multiobjective genetic algorithm," *IEEE Systems Journal*, vol. 12, no. 2, pp. 1688–1699, 2018.

[26] G. K. Shyam and S. S. Manvi, "Virtual resource prediction in cloud environment: a Bayesian approach," *Journal of Network and Computer Applications*, vol. 65, pp. 144–154, 2016.

[27] Y. Lu, J. Panneerselvam, L. Liu, and Y. Wu, "RVLBPNN: a workload forecasting model for smart cloud computing," *Scientific Programming*, vol. 2016, Article ID 5635673, 9 pages, 2016.

[28] K. Rajaram and M. P. Malarvizhi, "Utilization based prediction model for resource provisioning," in *Proceedings of 2017 International Conference on Computer, Communication and Signal Processing (ICCCSP)*, pp. 1–6, IEEE, Chennai, India, January 2017.

[29] L. Li and A. Zhang, "Resource demand optimization combined prediction under cloud computing environment based on IOWGA operator," *International Journal of Grid and Distributed Computing*, vol. 8, no. 3, pp. 77–86, 2015.

[30] D. Minarolli and B. Freisleben, "Cross-correlation prediction of resource demand for virtual machine resource allocation in clouds," in *Proceedings of 2014 Sixth International Conference on Computational Intelligence, Communication Systems and Networks*, pp. 119–124, IEEE, Tetova, Macedonia, May 2014.

[31] W. Zhang, P. Duan, L. T. Yang et al., "Resource requests prediction in the cloud computing environment with a deep belief network," *Software: Practice and Experience*, vol. 47, no. 3, pp. 473–488, 2017.

[32] H.-B. Mi, H.-M. Wang, G. Yin, D.-X. Shi, Y.-F. Zhou, and L. Yuan, "Resource on-demand reconfiguration method for virtualized data centers," *Journal of Software*, vol. 22, no. 9, pp. 2193–2205, 2011.

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

[34] Y. Meng, R. Rao, X. Zhang, and P. Hong, "CRUPA: a container resource utilization prediction algorithm for auto-scaling based on time series analysis," in *Proceedings of 2016 International Conference on Progress in Informatics and Computing (PIC)*, pp. 468–472, IEEE, Shanghai, China, December 2016.

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

[36] A. Ganapathi, Y. Chen, A. Fox, R. Katz, and D. Patterson, "Statistics-driven workload modeling for the cloud," in *Proceedings of 2010 IEEE 26th International Conference on Data Engineering Workshops (ICDEW 2010)*, pp. 87–92, IEEE, Long Beach, CA, USA, March 2010.

[37] V. G. Tran, V. Debusschere, and S. Bacha, "Hourly server workload forecasting up to 168 hours ahead using seasonal ARIMA model," in *Proceedings of 2012 IEEE International Conference on Industrial Technology*, pp. 1127–1131, IEEE, Athens, Greece, March 2012.

[38] D. Xu, S. Yang, and H. Luo, "Research on generalized fuzzy soft sets theory based combined model for demanded cloud computing resource prediction," *Chinese Journal of Management Science*, vol. 23, no. 5, pp. 56–64, 2015.

[39] S. Li, Y. Wang, X. Qiu, D. Wang, and L. Wang, "A workload prediction-based multi-VM provisioning mechanism in cloud computing," in *Proceedings of 2013 15th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pp. 1–6, IEEE, Hiroshima, Japan, September 2013.

[40] X. Fu and C. Zhou, "Predicted affinity based virtual machine placement in cloud computing environments," *IEEE Transactions on Cloud Computing*, vol. 99, p. 1, 2017.

[41] Y. Jiang, C. Perng, T. Li, and R. Chang, "ASAP: a self-adaptive prediction system for instant cloud resource demand provisioning," in *Proceedings of 2011 IEEE 11th International Conference on Data Mining*, pp. 1104–1109, IEEE, Vancouver, BC, Canada, December 2011.

[42] Z. Wu and N. E. Huang, "Ensemble empirical mode decomposition: a noise-assisted data analysis method," *Advances in Adaptive Data Analysis*, vol. 1, no. 1, pp. 1–41, 2009.

[43] H. Zang, L. Fan, M. Guo, Z. Wei, G. Sun, and L. Zhang, "Short-term wind power interval forecasting based on an EEMD-RT-RVM model," *Advances in Meteorology*, vol. 2016, Article ID 8760780, 10 pages, 2016.

[44] N. Safari, C. Y. Chung, and G. C. D. Price, "Novel multi-step short-term wind power prediction framework based on chaotic time series analysis and singular spectrum analysis," *IEEE Transactions on Power Systems*, vol. 33, no. 1, pp. 590–601, 2018.

[45] X. Chen, H. Wang, J. Huang, and H. Ren, "APU degradation prediction based on EEMD and Gaussian process regression," in *Proceedings of 2017 International Conference on Sensing, Diagnostics, Prognostics, and Control (SDPC)*, pp. 98–104, IEEE, Shanghai, China, August 2017.

[46] C. Yan, C. Yi, X. Wu et al., "Turbine fault trend prediction that based on EEMD and ARIMA models," *Journal of Gansu Sciences*, vol. 28, no. 4, pp. 100–106, 2016.

[47] M. Jin, P. Li, L. Zhang et al., "A signal feature method and its application based on EEMD fuzzy entropy and GK clustering," *ACTA Metrologica Sinica*, vol. 26, no. 5, pp. 501–505, 2015.

[48] E. H. Norden, Z. Shen, R. L. Steven et al., "The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis,"in *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* vol. 454, no. 1971, pp. 903–995, Royal Society, London, UK, March 1998.

[49] Alibaba cluster-trace-v2017, https://github.com/alibaba/clusterdata.