Research Article

Self-Adaptive Prediction of Cloud Resource Demands Using Ensemble Model and Subtractive-Fuzzy Clustering Based Fuzzy Neural Network

Zhijia Chen, Yuanchang Zhu, Yanqiang Di, and Shaochong Feng

Department of Electronic and Optics, Mechanical Engineering College, Shijiazhuang 050003, China

Correspondence should be addressed to Zhijia Chen; youshenshui@163.com

Received 28 September 2014; Accepted 25 December 2014

Academic Editor: Justin Dauwels

Copyright © 2015 Zhijia Chen et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In IaaS (infrastructure as a service) cloud environment, users are provisioned with virtual machines (VMs). To allocate resources for users dynamically and effectively, accurate resource demands predicting is essential. For this purpose, this paper proposes a self-adaptive prediction method using ensemble model and subtractive-fuzzy clustering based fuzzy neural network (ESFCFNN). We analyze the characters of user preferences and demands. Then the architecture of the prediction model is constructed. We adopt some base predictors to compose the ensemble model. Then the structure and learning algorithm of fuzzy neural network is researched. To obtain the number of fuzzy rules and the initial value of the premise and consequent parameters, this paper proposes the fuzzy *c*-means combined with subtractive clustering algorithm, that is, the subtractive-fuzzy clustering. Finally, we adopt different criteria to evaluate the proposed method. The experiment results show that the method is accurate and effective in predicting the resource demands.

1. Introduction

In cloud computing [1, 2], high accurate and efficient resource provisioning is an important aspect for maximizing the utility. In IaaS mode cloud computing [3], resources are allocated in the form of virtual machines which are composed of virtual hardware virtualized by hypervisor [4]. Users send requests to cloud center and try to obtain the resources in accordance with their demands. However, before cloud center provisioning these resources for users, some time may be needed to prepare and initialize the instance, i.e. the VMs. On the other hand, when the VM is running, resources dynamic adjusting is also needed to guarantee the QoS (quality of service). Some schemes do not consider the customer driven management, where resources have to be dynamically rearranged based on customers' demands [5]. The rearrangement of resources cannot take effect instantly and some time is needed, which leads to insufficiently providing the elastic management of resources [6]. Moreover, if the resources are not allocated properly, the performance of VMs may be restricted, or resources may be idle and wasted. This severely

decreases the utility and meanwhile the QoS of cloud computing.

It stands to reason that resource provisioning in the cloud environment is influenced directly by performance predictions [7]. In order to know how to allocate resources beforehand, it is important to characterize users' demands and preferences accurately. To make an accurate prediction, this paper analyses the main factors that affect the prediction performance and proposes a prediction method that proves to be more accurate and effective.

The key contributions of this paper are listed as follows:

- (1) We make analysis of the characters of user demands and preferences. Corresponding models and solutions are researched.
- (2) Self-adaptive cloud resource demands prediction algorithm using fuzzy neural network is proposed. Besides historical data, the base predictors' output results are adopted by FNN with different weight.
- (3) Fuzzy *c*-means combined with subtractive clustering (i.e., subtractive-fuzzy clustering) algorithm is

adopted to optimize the convergence features and learning speed.

- (4) The learning algorithm of fuzzy neural network is optimized with self-adjusting learning rate and momentum weight, which improves the robustness and the real-time performance.
- (5) To evaluate the prediction algorithm, some statistic indexes are introduced to compare with other algorithms, MSE (mean squared error), MAE (mean absolute error), and PRED(x).

2. Literature Review

2.1. Cloud Resource Demands Prediction. Researches on resource demand prediction are mainly focused on how to save energy [8], improve performance [7, 9], increase profit [5, 10, 11], and so on. To optimize resource management and task scheduling, Ramezani et al. [12] introduce a prediction method of VM workload patterns and VM migration time using fuzzy expert system. However, only a simple prediction model is depicted and the details are not explicated. By contrast, a fuzzy prediction method is given in [13] to model the uncertain workload and the vague availability of virtualized server nodes, using type-1 and type-2 fuzzy logic systems. Adopting fuzzy algorithm, the performance of prediction method is more robust but the accuracy is decreased. This method needs to be combined with another prediction method to realize high performance.

There are some methods predicting resource demands based on duration time. Reference [14] proposes an approach for long-term trend prediction using moving average method. To control jitter in a small range, it further improves the conventional moving averages method using standard deviations. This method mainly aims at long-term prediction and the short-term prediction is not mentioned.

To balance the performance and the system cost, some researches make efforts to maximize the system utility. In [15], fast up and slow down algorithm is introduced to maximize the performance while maintaining the stability.

As workload has an obvious nonlinear feature, many machine-learning algorithms have also been used to support its prediction. Neural network (NN) is introduced as a prediction method [16–19]. Combining with the typical predicting methods such as sliding window method [20], auto regression model [21], and exponential smoothing model [22], they have worked well in predicting the workload. To improve the inference ability, fuzzy neural network is introduced for predicting.

2.2. Fuzzy Neural Network. Fuzzy neural network is a combination of a fuzzy logic system and a neural network. It keeps the merit of each [23]. The algorithm is widely used in various applications such as pattern recognition, prediction, and system control [24–26]. In prediction area, it has been discovered to be more accurate than other conventional or soft computing techniques. In [27], neurofuzzy and neural network techniques are adopted to forecast the sea level in Darwin Harbor. It is proved that adaptive neurofuzzy

inference system (ANFIS) is more effective in predicting than autoregressive moving average (ARMA) algorithm. References [28, 29] introduce FNN approach into energy consumption demands predicting. Results of [28] even show that the hybrid ANFIS model has better performance than ANN in terms of prediction accuracy. Some works [30, 31] use FNN algorithm in hydrological time series prediction. In the machine condition maintenance area, FNN algorithm is used to predict condition of the machine or the components [32, 33].

According to the literature reviewed above, FNN is used in many areas for predicting and the performance is fine. Unfortunately, in cloud resource demands predicting area, few researches have adopted FNN as predicting method. In this paper, we adopt FNN with self-adjusting learning rate and momentum weight as the core of the prediction system. To improve the performance, we use ensemble model and clustering algorithm to optimize FNN prediction system. Before the data is sent to the predictor, some preparation work needs to be done.

3. System Overview and Preparation for Prediction

3.1. System Overview. Before prediction of user demands, we firstly analyze the user requests, including the utilization data structure, content, and number of historical resources. By analyzing the historical data, we may draw conclusions about user preference, demand description, and so forth. The short-term versus long-term and the fluctuation period versus flat period prediction are separately treated. Fluctuation-threshold (th_u) and flat-threshold (th_a) are defined to distinguish the flat period and fluctuation period. In different periods, different base prediction methods are adopted according to the characters, such as second moving average model (SMA), exponential moving method (EMA), autoregression model (ARM), and trend seasonality model (TSM). The output of the base predictors is sent to the fuzzy neural network as input. Fuzzy neural network uses the historical data and the base prediction value as training data, which improves the accuracy of the results. The output of fuzzy neural network is used to instruct the resource allocation in IaaS cloud center. Prediction results and the actual resource demands are evaluated using statistical analysis and different criteria. The evaluation results are fed back to the historical database to improve the prediction performance. The overview of resource demands prediction system is depicted in Figure 1.

3.2. Overall Situations of User Demands. As depicted in Figure 1, there are some opposite circumstances to be concerned, such as long-term demands and short-term demands and fluctuation period and nonfluctuation period. Fluctuation period is an abnormally violent vibration on a cloud resource over a period of time.

In user long-term resources requirements, there are some characters, as we summarize in the following: (1) the regularity is more obvious than in short-term. It is understandable that the long-term users may show some repetition regularity;

FIGURE 1: The overview of the prediction system.

FIGURE 2: The model of sliding window.

(2) in the long running of the system, there may be some fluctuation periods and some flat periods. Coming to short-term requirements, the regularity may be not much obvious, and the fluctuation feature is more noticeable. Therefore, the long-term or short-term may be not conflicted with fluctuation or flat periods. For long-term data, we can on the one hand summarize the regularity. On the other hand, the flat and fluctuant periods should be distinguished. For short-term data, the regularity is not easy to figure out and the fluctuation should be processed. The prediction speed should be ensured as the short-term resources provision gives first place to quick response than other performance. The difference between short-term and long-term processing mainly lies in the fluctuation period processing. Hereby we discuss the fluctuation period and flat period, respectively.

3.2.1. Flat Period Procession. Based on the characters of flat period, the second moving average (SMA) [34] is adopted, it can effectively reduce the lag deviation between prediction value and actual value. In this method, we define a sliding window whose input size is N , that is, $X = [x_t, x_{t-1}, \dots, x_{t-(N-1)}]$ over the historical time interval $[t - (N - 1), t]$. Figure 2 depicts the method model of sliding window with a window size N .

The predicted output value $x_{t+\tau}$ after a prediction time interval τ is the dependent variable set X , where $X =$

$[x_1, x_2, \dots, x_N]$. Then the relationship can be abstracted as follows:

$$x_{t+\tau} = f(X, \tau). \quad (1)$$

The i th user's resource requirement prediction at time $t + \tau$ can be expressed as follows:

$$x_{t+\tau}(i) = a_t(i) + \tau b_t(i). \quad (2)$$

In function (2), $x_{t+\tau}(i)$ is the prediction value in time $t + \tau$, τ is the time sequence number to be predicted. $a_t(i)$ and $b_t(i)$ satisfy the following constraint equation:

$$a_t(i) = 2M_t^{(1)}(i) - M_t^{(2)}(i), \quad (3)$$

$$b_t(i) = \frac{2}{N-1} [M_t^{(1)}(i) - M_t^{(2)}(i)]. \quad (4)$$

In formula (3), $M_t^{(1)}(i)$ and $M_t^{(2)}(i)$ separately represent the first moving average value and the second moving average value at the t th time of the resource the i th user requested. N is the time period of moving average period. In addition, $M_t^{(1)}(i)$ and $M_t^{(2)}(i)$ can be expressed as follows:

$$M_t^{(1)}(i) = \frac{x_t(i) + x_{t-1}(i) + \dots + x_{t-(N-1)}(i)}{N}, \quad (5)$$

$$M_t^{(2)}(i) = \frac{M_t^{(1)}(i) + M_{t-1}^{(1)}(i) + \dots + M_{t-(N-1)}^{(1)}(i)}{N}.$$

```

For prediction value series  $\{x_t, x_{t-1}, \dots, x_{t-(n-1)}\}$  do
  Calculate the differences  $d_t, d_{t-1}, \dots, d_{t-(n-1)}$  of the adjacent value
End
Set  $d_u$  and  $d_l$  as the threshold of fluctuation
For the differences  $d_t, d_{t-1}, \dots, d_{t-(n-1)}$  do
  Compare them with benchmark  $d_u$  and  $d_l$ 
  If they are greater than  $d_u$  then
     $th_u$  is reached
  Else If they are less than  $d_l$  then
     $th_a$  is reached
End
For each type resource  $i$  do
  Compare the fluctuation degree with  $th_u$  and  $th_a$ 
  If it is greater than  $th_u$  then
    Resource Demands are experiencing a fluctuation period
  Else if it is less than  $th_a$  then
    Resource Demands are experiencing a flat period
  Else
    Resource demands is intervenient flatness and fluctuation
End

```

PSEUDOCODE 1

Then the total number of the resources requested by all the m users is

$$x_{t+\tau} = \sum_{i=1}^m x_{t+\tau}(i). \quad (6)$$

From the analysis above, we can see that the prediction value at the time $t + \tau$ is decided only by the values of the N periods' values at the time t and the total number of the users. And the prediction result can be calculated at time t .

3.2.2. Fluctuation Period Procession. Exponential moving method (EMA) is an effective method for short-term prediction and particularly suitable for time series prediction of the nonseasonal effect owing to its quick responsiveness and weight decreases with time passed. Predicted values are calculated using smoothing constant α . The exponential moving average is expressed as follows:

$$x_{t+1} = \alpha X(t, N) + (1 - \alpha) x_t. \quad (7)$$

In (7), $X(t, N)$ is the moving average value between the past time $t - (N - 1)$ and the current time t . Then the time interval is N . α is the smoothing constant that can be calculated by $\alpha = 2/(N + 1)$. We can see that α confines to $[0, 1]$.

The EMA method gives a higher weight to the later measure value and lower weight to the earlier measure value. So the EMA method is able to response rather quickly to the fluctuations in a short-term demand and workload conditions [35]. However, there will be some delay introduced as the window size increases. Based on Andreolini and Casolari [36], in nonlinear load trackers, the polynomial orders should be properly selected. If the order is low (degree ≤ 2), then the function will not react quickly enough to load changes. If the order is high (degree ≥ 4), the function will be unnecessarily complex and some undesirable sparks will be introduced. The cost may be too expensive for a run-time context.

3.2.3. The Identification of Flat Period and Fluctuation Period. Though we give different prediction methods according to the wiggle levels, it is difficult to know or identify the boundary of different wiggle levels in the overall situation. In this section, we define "fluctuation-threshold (th_u)" and "flat-threshold (th_a)" to distinguish the fluctuation and flat periods. Fluctuation-threshold is defined as the upper limit of the degree of vibration demand on cloud resource, while flat-threshold is the lower limit. In the last n period of time, if the difference d_t of prediction value x_t and x_{t-1} in series $\{x_t, x_{t-1}, \dots, x_{t-(n-1)}\}$ is greater than a certain value d_u , then the th_u is reached. If the difference of x_t and x_{t-1} is less than a certain value d_l , then the th_a is reached. For resource type i , the demands are experiencing a fluctuation period if the demand data in last k time satisfies the condition $Deg_i \geq th_u$, where Deg_i means the fluctuation degree of the prediction trend. th_u is the upper limit value. Type i resource demands are experiencing a flat period if the demand data in last k time satisfies $Deg_i \leq th_a$. If $th_a \leq Deg_i \leq th_u$, the demands of resource i are intervenient flatness and fluctuation.

The above procedure can be illustrated by Pseudocode 1.

4. Resource Demands Prediction Using Optimized Fuzzy Neural Network

In order to predict the resource demands accurately and effectively, a prediction method with different individual base prediction models ensemble and fuzzy neural network is proposed in this section. With different base prediction models, different demands occasions can be estimated accordingly, and the most likely future outcome is able to be predicted. With the results of base predictors, the fuzzy neural network tends to present better predicting performance. To improve the learning ability, we use self-adjusting learning rate and

FIGURE 3: The structure of ensemble model and FNN based prediction model.

momentum weight to optimize the learning procedure. Clustering algorithm is adopted to initialize the fuzzy inference rules of the FNN. The introduction of fuzzy neural network promises the robustness and accuracy of the prediction system.

The core of the prediction model adopts a two-level structure, as shown in Figure 3. The first level is an ensemble model that contains different base predictor models. The output of the first level is sent to the second level, fuzzy neural network, which is responsible for optimizing the precision and the robustness of the prediction results.

4.1. Base Prediction Models. As we know, diversity is necessary for the survival and evolution of species ensemble model. So as for the performance of the prediction models, it is important to introduce the diversity to the prediction ensemble model. To guarantee the prediction performance; the base prediction models should be firstly selected. Besides the prediction models mentioned in Section 3, some other models are introduced. The guideline of choosing is based on the capacity and overheads.

4.1.1. Autoregression Model. Autoregression model (ARM) is one of the linear models that are used for estimating the relationships between one dependent output variable y and one or more independent variables x_i . It represents how the dependent value changes along with each independent variable changing. The fundamentals of the method are to treat the historical measurement data as a stochastic process which can be treated as a white noise driven filter. It is proved effective for predicting host load. The form of an AR model is

$$y = \alpha_1 x_{t-1} + \alpha_2 x_{t-2} + \dots + \alpha_p x_{t-p} + e_t, \quad (8)$$

where e_t is white noise signal that contains all the unpredictable information in the past.

4.1.2. Trend Seasonality Prediction Model. Trend seasonality model (TSM) represents a regularity that repeats periodically, which can be modeled by low order polynomials. To measure

how the general cycle affects data value, we calculate a series of periodic indicators. Seasonality indicator demonstrates the offset between certain period average value and the overall average value. To get an accurate estimation of the indicator, each periodic value is calculated and compared with the total average value. The seasonal indicator Ind_i can be calculated by the equation $Ind_i = avg_i / avg$, where avg_i is the average value of the i th period and avg is the total value. The periodic data are generated by the cloud users' resource requirements. Based on the indicators, the future resource requirements are predicted by two steps: (1) compute the future trend level by using a polynomial equation with order two; (2) introduce the seasonal influence by multiplying the trend level by indicators.

4.1.3. Moving Average Method. By judging whether the monitoring data crosses over the moving average, moving average method (MAM) predicts the future trend. $MA(N)$ is a moving average of monitoring data series R_i with length of N , expressed as equation $MA(N) = \sum_{i=(N-1)}^i (R_i / N)$, s.t. $N \leq i$. If the monitoring data cross over the moving average upward, it indicates that an ascending trend is coming. While if the monitoring data cross over the moving average downward, it indicates that a descending trend is coming.

4.2. Fuzzy Neural Network. In fuzzy neural network, we use neural network to evolve the fuzzy inference rules. We consider a multi-input single-output (MISO) fuzzy model which consists of l rules. The i th if-then rules of fuzzy inference system can be expressed as follows.

Rule i . If x_1 is $A_1^{(i)}$ and x_2 is $A_2^{(i)}$ and $\dots$ x_n is $A_n^{(i)}$, then y is $B^{(i)}$.

In the rules, A_i , $B^{(i)}$, and M_i are the linguistic labels (e.g., high or low) associated with the node functions. The "if part" (antecedent) is fuzzy in nature, while the "then part" (consequent) is a crisp function of an antecedent variable.

4.2.1. The Structure of FNN. To give a full description to the FNN without loss of generality, we construct the FNN as

FIGURE 4: The architecture of FNN.

shown in Figure 4. $\text{net}_j^{(k)}$ means the net input of j th node in the k th layer. $y_j^{(k)}$ is the output of the j th node in the k th layer. $x_j^{(k)}$ is the input of the j th node in the k th layer. Here the relation between x and y is $y_j^{(k)} = x_j^{(k+1)}$. The meanings and function relations of each layer are discussed in the following.

Layer 1 is known as the input layer. Nodes in this layer transmit the input data to the next layer directly. Input data is abstracted as $X = \{x_1, x_2, \dots, x_n\}$. The relationship between input and output can be expressed as follows:

$$\text{net}_j^{(1)} = x_i^{(1)}, \quad i = j, \quad y_j^{(1)} = \text{net}_j^{(1)}. \quad (9)$$

Layer 2 is known as the fuzzification layer. In this layer, every node performs the calculation of a Gaussian membership function and specifies the degree to which the given input x_i satisfies the quantifier A_i . Consider

$$\text{net}_j^{(2)} = -\frac{1}{2} \left(\frac{x_i^{(1)} - m_{ij}}{\sigma_{ij}} \right)^2, \quad (10)$$

$$y_j^{(2)} = \mu_{A_j}(x_j) = \exp(\text{net}_j^{(2)}).$$

Here, m_{ij} and σ_{ij} are the center and the width of the Gaussian function of the j th term of the i th input variable, respectively. Both m_{ij} and σ_{ij} are adjustable parameters.

Layer 3 is the fuzzy inference layer. The fuzzified results of the individual scalar functions of each input data are aggregated. All potential rules of the input data are formulated by applying fuzzy intersection, which means product of data.

Thus, a product operation denoted as $\prod$ is performed to obtain the output of this layer. Consider

$$\text{net}_j^{(3)} = \prod_i x_{ij}^{(3)} = \prod_j y_j^{(2)} \quad y_j^{(3)} = \text{net}_j^{(3)}. \quad (11)$$

The output of layer 3 represents the firing strength of the rules.

Layer 4 is the normalization layer. Each node in layer 4 is labeled with N which denotes “normalization.” The ratio of the i th rule firing strength to the total firing strengths is calculated in this layer. The relationship between input data and output data is expressed as follows:

$$\text{net}_j^{(4)} = \frac{x_i^{(4)}}{\sum_i x_i^{(4)}} \quad y_j^{(4)} = \text{net}_j^{(4)}. \quad (12)$$

The output of this layer is known as normalized firing strength.

Layer 5 and layer 6 are known as the defuzzification layers. In layer 5, every node in this layer is labeled with θ_i . The node function is as follows:

$$\text{net}_j^{(5)} = \theta_i x_i^{(4)} \quad y_j^{(5)} = \text{net}_j^{(5)}. \quad (13)$$

Here θ_i is the adjustable weight parameter. Parameters in this layer are known as consequent parameters.

In layer 6, there is only one node labeled with Σ . It is used to compute the output of the fuzzy neural network. The output is the summation of all the incoming data from layer 5. Consider

$$\text{net}_1^{(6)} = \sum_i x_i^{(6)} \quad y_1^{(6)} = \text{net}_1^{(6)}. \quad (14)$$

4.2.2. The Learning Procedure of FNN. After the structure of FNN is constructed, we use error back propagation method to adjust the parameters θ_i , m_{ij} , and σ_{ij} . The objective function is defined as

$$E = \frac{1}{2} e^2 = \frac{1}{2} (\hat{y} - y_1^{(6)})^2, \quad (15)$$

where $\hat{y}$ is the instructor signal. The error propagates from the output layer back to the input layer. To clearly analyze the back propagation procedure, we combine layer 5 and layer 6 as one.

In the defuzzification layer (layer 5), the evolution of error is expressed as follows:

$$\delta_1^{(5)} = -\frac{\partial E}{\partial \text{net}_1^{(5)}} = \hat{y} - y_1^{(5)} = e, \quad (16)$$

$$\begin{aligned} \Delta \theta_i &= -\eta \frac{\partial E}{\partial \theta_i} = -\eta \frac{\partial E}{\partial \text{net}_1^{(5)}} \cdot \frac{\partial \text{net}_1^{(5)}}{\partial \theta_i} = \eta \cdot \delta_1^{(5)} \cdot x_i^{(5)} \\ &= \eta \cdot \delta_1^{(5)} \cdot y_i^{(4)} = \eta \cdot \delta_1^{(5)} \cdot \prod_i x_i. \end{aligned} \quad (17)$$

In layer 4, the differential error of node j is

$$\delta_j^{(4)} = -\frac{\partial E}{\partial \text{net}_j^{(4)}} = -\frac{\partial E}{\partial \text{net}_j^{(5)}} \cdot \frac{\partial \text{net}_j^{(5)}}{\partial \text{net}_j^{(4)}} = \delta_1^{(4)} \cdot \theta_j. \quad (18)$$

In layer 3, $\delta_j^{(3)}$ is

$$\begin{aligned}\delta_j^{(3)} &= -\frac{\partial E}{\partial \text{net}_j^{(3)}} = -\frac{\partial E}{\partial \text{net}_j^{(5)}} \cdot \frac{\partial \text{net}_j^{(5)}}{\partial \text{net}_j^{(4)}} \cdot \frac{\partial \text{net}_j^{(4)}}{\partial \text{net}_j^{(3)}} \\ &= \delta_j^{(4)} \cdot \frac{\sum_j \text{net}_j^{(3)} - \text{net}_j^{(3)}}{(\sum_j \text{net}_j^{(3)})^2}.\end{aligned}\quad (19)$$

In layer 2, $\delta_j^{(2)}$ is

$$\begin{aligned}\delta_j^{(2)} &= -\frac{\partial E}{\partial \text{net}_j^{(2)}} \\ &= -\frac{\partial E}{\partial \text{net}_j^{(5)}} \cdot \frac{\partial \text{net}_j^{(5)}}{\partial \text{net}_j^{(4)}} \cdot \frac{\partial \text{net}_j^{(4)}}{\partial \text{net}_j^{(3)}} \cdot \frac{\partial \text{net}_j^{(3)}}{\partial \text{net}_j^{(2)}} \\ &= \delta_j^{(3)} \cdot \prod_{\substack{j=1 \\ j \neq i}} \mu_{ij}, \\ \Delta m_{ij} &= -\eta \frac{\partial E}{\partial m_{ij}} = -\eta \frac{\partial E}{\partial \text{net}_j^{(2)}} \cdot \frac{\partial \text{net}_j^{(2)}}{\partial m_{ij}} \\ &= 2\eta \delta_j^{(2)} \cdot \exp\left(-\frac{(x_i - m_{ij})^2}{\sigma_{ij}^2}\right) \cdot \frac{x_i - m_{ij}}{\sigma_{ij}^2}, \\ \Delta \sigma_{ij} &= -\eta \frac{\partial E}{\partial \sigma_{ij}} = -\eta \frac{\partial E}{\partial \text{net}_j^{(2)}} \cdot \frac{\partial \text{net}_j^{(2)}}{\partial \sigma_{ij}} \\ &= 2\eta \delta_j^{(2)} \cdot \exp\left(-\frac{(x_i - m_{ij})^2}{\sigma_{ij}^2}\right) \cdot \frac{(x_i - m_{ij})^2}{\sigma_{ij}^3}.\end{aligned}\quad (20)$$

The update parameters are

$$\begin{aligned}\theta_l(t+1) &= \theta_l(t) + \eta_1 \cdot \Delta \theta_l(t), \\ m_{ij}(t+1) &= m_{ij}(t) + \eta_2 \cdot \Delta m_{ij}(t), \\ \sigma_{ij}(t+1) &= \sigma_{ij}(t) + \eta_3 \cdot \Delta \sigma_{ij}(t).\end{aligned}\quad (21)$$

4.2.3. Self-Adjusting Learning Method and Momentum Weight. In (17), the learning rate η determines the convergence speed of the neuron network. If η is small, the changes of synapse weight in the iterative computation procedure will be small and the locus of the weight space becomes smooth. However, the learning rate is decreased. If η is too large, the learning rate will increase but the network may become unstable and may cause wobble of the weights. To optimize the convergence speed and stability of the neural network, a momentum term can be included in (17), and it is expressed as follows:

$$\Delta \theta_k^{(l)} = \eta \delta_k^{(l)} \bar{y}_k^{(l)} + \lambda \Delta \theta_k^{(l-1)}, \quad (22)$$

where λ is the momentum constant. Equation (22) introduces the preceding $\Delta \theta_k^{(l-1)}$ into the procedure of calculating $\Delta \theta_k^{(l)}$.

The use of the momentum constant is a minor revise for refreshing the weight. However, it makes some advantages for the learning speed of the algorithm.

In addition, we adopt a “progressive-increase” and “conservative-decrease” method to adjust the learning rate η . If the error declines in the training procedure, we consider the modification direction to be right and a larger adjusting variable k is used. If the error is becoming bigger, we regard that the modification is excessive and the adjusting step needs to be slowed down and a smaller value is assigned to variable k . Meanwhile, the former modification should also be abandoned. The method is shown in the following function:

$$\eta(i+1) = \begin{cases} k_{\text{inc}} \eta(i) & E(i+1) < E(i) \\ k_{\text{dec}} \eta(i) & E(i+1) > E(i). \end{cases} \quad (23)$$

The variable i means the learning steps. k_{inc} and k_{dec} are, respectively, the increase factor and the decrease factor.

4.3. Optimized Clustering Method

4.3.1. Fuzzy c -Means Clustering. In FNN, the construction of fuzzy if-then rules is difficult. The improper rule set may result in bad prediction results. Recently, a number of different approaches have been used for designing fuzzy if-then rules based on clustering, gradient algorithms [37], genetic algorithms [38], fuzzy c -means clustering [39], and subtractive clustering [40]. Fuzzy clustering is an efficient technique for constructing the antecedent structures. The aim of clustering methods is to identify a certain group of data from a large data set, such that a concise representation of the behavior of the system is produced. Each cluster center can be translated into a fuzzy rule for identifying the class. In this paper, the fuzzy c -means clustering technique is used for structuring the premise part of the fuzzy system.

By analyzing the membership degree of sample data, the fuzzy c -means algorithm clusters partitions data set of different classes. Consider that there are n objects $X = \{x_1, x_2, \dots, x_n\}$. Fuzzy c -means partitions them into c fuzzy clusters, where c confines to $1 < c < n$. The centroids of the clusters are $Z = \{z_1, z_2, \dots, z_c\}$. The form of fuzzy clustering of objects is a fuzzy rule set matrix μ with n rows and c columns, where n and c are the total number of data objects and the number of clusters separately. μ_{ij} indicates the degree of association or membership function of the i th object with the j th cluster. The characters of μ are shown in the following:

$$\begin{aligned}\mu_{ij} &\in [0, 1] \quad 0 < \sum_{i=1}^n \mu_{ij} < n \quad \sum_{j=1}^c \mu_{ij} = 1 \\ i &= 1, 2, \dots, n; \quad j = 1, 2, \dots, c.\end{aligned}\quad (24)$$

The optimization objective function of FCM algorithm is

$$J(\mu_{ij}, z_i) = \sum_{i=1}^n \sum_{j=1}^c \mu_{ij}^m d_{ij}^2 \quad d_{ij} = \|x_i - z_j\|. \quad (25)$$

In the above equation, m is the exponent weight and it controls the fuzziness of the clusters and d_{ij} is the Euclidean distance between objective x_i and the centroid z_j . By

minimizing $J(\mu_{ij}, z_i)$, the centroid of the j th cluster can be calculated using the following equation:

$$z_j = \frac{\sum_{i=1}^n \mu_{ij}^m x_i}{\sum_{i=1}^n \mu_{ij}^m}. \quad (26)$$

The membership degree matrix can be calculated by the following equation:

$$\mu_{ij} = \frac{1}{\sum_{k=1}^c (d_{ij}/d_{ik})^{2/(m-1)}}. \quad (27)$$

4.3.2. Subtractive Clustering. From the discussion above, we can see that FCM is sensitive to isolated data. As the sum of the membership degree has to be 1, the results may be not good if the sample data is not ideal. Besides, the cluster centroids of FCM are initialized stochastically. If the initial value is not properly selected, the convergence may be affected and local convergence may happen. Thus, FCM relies on initial centroids greatly. Moreover, the diversity of membership function may lower the convergence speed.

To improve the FCM algorithm, we introduce the subtractive clustering as a complement. Subtractive clustering is unsupervised clustering, in which the number of clusters for input data points is determined by the clustering algorithm. The subtractive clustering does not need to define the number of the clusters. The results may be used for initializing the centroids of FCM algorithm. It assumes that each data is a potential cluster centroid. Based on the data density index of the potential centroid data, we select the data that has the highest density as the centroid. The procedure is concluded as follows.

- (1) Calculating the density index of each data x_i

$$D_i = \sum_k \exp \left[-\frac{\|x_i - x_k\|^2}{(r_a/2)^2} \right]. \quad (28)$$

The clustering radius r_a is determined by the following equation:

$$r_a = \frac{1}{2} \min_k \left\{ \max_i \|x_i - x_k\| \right\}. \quad (29)$$

Data beyond the radius affects little to the density index. We firstly choose the data x_{c1} that has the highest density index D_{c1} as the first cluster centroid. Then the data in the radius r_a is removed from the potential centroid data set.

- (2) We use the following equation to modify the density index of each data:

$$D_i = D_i - D_{c1} \exp \left[-\frac{\|x_i - x_{c1}\|^2}{(r_b/2)^2} \right]. \quad (30)$$

s.t. $r_b > r_a$. A neighborhood with lower density index is defined by the above equation. The aim is to keep one centroid away from others so that the clusters may be distinct from others. As the density index of the data that is close to the first cluster centroid x_{c1} is much lower, the potential to be the centroid is also decreased.

- (3) By calculating the density index of the remaining data, the next centroid is obtained. If the constraint equation (31) is achieved, we regard D_{ck} as the centroid of cluster C_k :

$$\frac{D_{ck}}{D_{c1}} < \delta. \quad (31)$$

Here, $0 < \delta < 1$ is the constraint parameter that decides the number of the cluster centroid. Through (31), we can see that the number of clusters is inversely proportional to δ . Furthermore, the identification sequence of centroids is decided by density index. The higher the density index is, the earlier the centroid emerges, and the proper centroid probability becomes greater.

4.3.3. The Combination of Fuzzy c-Means and Subtractive Clustering. This section introduces a clustering algorithm that combines FCM with subtractive clustering method. We obtain the cluster centroids and number through subtractive cluster. This can effectively improve the convergence speed of FCM and the probability of local convergence is decreased.

The procedure can be described as follows.

Step 1. Set the parameters, including neighborhood radius r_a and r_b , fuzzy exponent weight m , and comparison parameter δ .

Step 2. Calculate the number and centroids of clusters through subtractive method.

Step 3. Use (25) and (27) to calculate the objective function and the membership degree.

Step 4. Verify if the termination constraints are achieved. If $\|J^{(K+1)} - J^{(k)}\| < \varepsilon$ or the maximum iteration is achieved, the operation process terminates. Otherwise, turn to Step 5.

Step 5. Update k with $k + 1$ and turn to Step 3. Use (25) and (27) to calculate new cluster centroids and membership degree.

4.4. Validation Criteria. To evaluate the performance of the prediction system, we use a series of metrics [20] including MAE (mean absolute error), MSE (mean squared error), and PRED(x).

4.4.1. MAE and MSE. MAE is the criterion of measuring the mean deviation between the prediction output and the actual output. MAE can be calculated by the following function:

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |\hat{y}_i - y_i|, \quad (32)$$

where $\hat{y}_i$ is the actual output and y_i is the prediction value. n is the number of the data series. The smaller the value of MAE is, the more accuracy the prediction method is.

FIGURE 5: The training results and error of ESFCFNN.

MSE represents the energy of the mean error. MSE can be expressed as in the following:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2. \quad (33)$$

4.4.2. $\text{PRED}(x)$. $\text{PRED}(x)$ is the proportion of the prediction data number whose relative error falls within $\pm x \times 100\%$ to the whole data number. Take $\text{PRED}(5)$ as an example, according to formula [14], we define the relative error as

$$\tilde{e}_i = \frac{y_i - \hat{y}_i}{\hat{y}_i}, \quad i = 1, 2, \dots, n. \quad (34)$$

i represents the series number of the output data series. The number of all the relative errors that meet the condition $-5\% \leq \tilde{e}_i \leq 5\%$ is supposed as $k_{(5)}$. The whole number is n . Then $\text{PRED}(5)$ is defined as

$$\text{PRED}(5) = \frac{k_{(5)}}{n}. \quad (35)$$

The index $\text{PRED}(5)$ represents the fitness of the prediction model. If the value is close to 1.0, it indicates a good fit of the prediction model.

4.5. Feedback Control. To optimize the performance of the resource demands prediction system, we introduce the feedback control [41] into the system. In each prediction cycle, the feedback controller sends the actual resource demands and prediction results to historical database. Specially, the demands value is specified in fine-grained form, including the elements in data structure vector P_i . In addition, the validation indexes of MAE, MSE, $\text{PRED}(10)$, and so forth are also processed in the controller. The feedback controller sends corresponding value to the historical database.

5. Experimental Evaluation

In this section, experiments are conducted to validate the proposed prediction method. When we predict the fine-grained resource demands, the method of each kind of resource is similar to others. Here we do not distinguish resource type, and we use network traffic as the representation. From [42], we sample 400 days network visit traffic data. We use anterior 350 days traffic data as training data and posterior 50 days traffic data as test data. The training effect is shown in Figure 5. In Figure 5(a), the blue curve represents the prediction value and the red curve represents the actual value. The two curves fit with each other. In Figure 5(b), we can see that the overall effect is promising. The maximum error is controlled within $-1.1\% \sim +1\%$. The training results are accurate.

The test data is used to verify the prediction method. The test results are shown in Figure 6. In Figure 6(a), the blue curve is the prediction value and the red curve is the actual value. We can see that the most of the two curves almost overlap with each other. Figure 6(b) shows the prediction error. We can see that the maximum normalized error is 8%. Most of the normalized error data falls within $-8\% \sim +8\%$. The prediction results are acceptable.

The prediction error using different methods is compared with each other and the results are depicted in Figure 7. In Figure 7, the performances of EMA, SMA, AR, and ESFCFNN methods are evaluated. The difference between different predicting methods is shown intuitively. We can see that the prediction error using ESFCFNN prediction method is apparently decreased. The results of ESFCFNN method are more accurate than other methods. The performance of ESFCFNN is optimized greatly.

According to the main criteria we defined from formulas (32) to (35), we test some base prediction methods including SMA, EMA, AR, and ESFCFNN. The results are shown

(a) The actual output and prediction output of ESFCFNN training procedure

(b) The normalized error curve of ESFCFNN training procedure

FIGURE 6: The test results and error of ESFCFNN.

FIGURE 7: Normalized prediction error using different methods.

in Figure 8. In Figure 8(a) we can see that both the mean average error in regularity and the max error of the ESFCFNN method are small. The performances of EMA and SMA are close to ESFCFNN. So as for the MSE and SSE, the ESFCFNN takes advantages of the other predictors' merits and realizes self-adaption and robustness. Finally, Figure 8(d) shows the number of errors that falls in 5%. The red columniation of Figure 8(d) is the total number, which are 50. The blue columniation is the number that falls in 5%. From the above analysis, we can conclude that the prediction results of ESFCFNN method are more accurate and this method is promising for predicting users' resource demands.

Figure 9(a) depicts the training procedure without self-adjusting learning rate. After 100 training cycles, the error

P_a of FNN is approximately 0.048. The error is very large. Figure 9(b) depicts the performance of FNN with self-adjusting learning rate and momentum weight. After 100 training cycles, the error P_b reaches 0.0015. The ratio of P_a to P_b is $r = P_a/P_b = 0.048/0.0015 = 32$.

The performance is much better after using self-adjusting learning rate and momentum weight. Moreover, the convergence speed after using self-adjusting learning rate is improved. The training error falls down to 0.05 within 10 steps when using self-adjusting learning rate. However, if the self-adjusting learning rate algorithm is not adopted, the convergence speed is slowed down. More than 85 steps are needed before error falls down to 0.05. From above analysis, we can see that the performance of FNN with self-adjusting learning rate and momentum weight is improved greatly.

In Figure 10, the performance of FNN without using clustering algorithm is depicted. Figure 10(a) shows the convergence procedure. We can see that the convergence speed is slowed down. After more than 30 steps, the error falls down to 0.05. While Figure 9(b) shows that, with clustering algorithm, this procedure only needs less than 10 steps. Figure 10(b) shows the training error of FNN without using clustering algorithm. Compared with Figure 5, the training error of FNN without using clustering algorithm is greater. From the above comparison, the performance is improved using clustering algorithm.

6. Conclusions

To improve the performance of resource provision and resource utilization, this paper proposed a cloud resource demands self-adaptive predicting method using ensemble model and subtractive-fuzzy clustering based fuzzy neural network, which is called ESFCFNN for short. We discuss the structure of the prediction system. Users' preferences are analyzed to reduce the amount of calculation. Then the base

FIGURE 8: Prediction error indexes comparison of different methods.

prediction model is introduced into the system. The results are sent to FNN with self-adjusting learning rate and momentum weight as the inputs. To optimize the convergence performance of FNN, fuzzy-subtractive clustering algorithm is proposed. Fuzzy-subtractive algorithm is composed with fuzzy c -means clustering algorithm and subtractive clustering algorithm. We evaluate the prediction system using statistic criteria including MAE, MSE, SSE, and PRED(5). The results show that ESFCFNN can effectively improve the prediction performance.

Though the method this paper proposes is promising in improving the performance, the system is complex. As we can see, there are two prediction layers. The time delay may be

increased. In future, the improvement of efficiency is the main point of the research. We would also test the method in the real cloud computing system in future.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This work is supported by the Weapon Equipment Pre-research Program under Grant no. 9140A04030213JB34001.

FIGURE 9: The comparison of the training performance of different learning algorithms.

FIGURE 10: The performance without clustering.

References

- [1] L. Qian, Z. Luo, Y. Du et al., "Cloud computing: an overview," in *Cloud Computing*, vol. 5931 of *Lecture Notes in Computer Science*, pp. 626–631, Springer, Berlin, Germany, 2009.
- [2] S. Marston, Z. Li, S. Bandyopadhyay, J. Zhang, and A. Ghalsasi, "Cloud computing—the business perspective," *Decision Support Systems*, vol. 51, no. 1, pp. 176–189, 2011.
- [3] Y. Naibo, "Investigation of IaaS mode," *Telecommunications Science*, no. 10A, pp. 39–43, 2011.
- [4] C.-Y. Yeh, C.-Y. Kao, W.-S. Hung et al., "GPU virtualization support in cloud system," in *Grid and Pervasive Computing*, vol. 7861 of *Lecture Notes in Computer Science*, pp. 423–432, Springer, Berlin, Germany, 2013.
- [5] L. Wu, S. K. Garg, and R. Buyya, "SLA-based resource allocation for software as a service provider (SaaS) in cloud computing environments," in *Proceedings of the 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid '11)*, pp. 195–204, Newport Beach, Calif, USA, May 2011.
- [6] H.-S. Wu, C.-J. Wang, and J.-Y. Xie, "Terascaler ELB—an algorithm of prediction-based elastic load balancing resource management in cloud computing," in *Proceedings of the 27th International Conference on Advanced Information Networking and Applications Workshops (WAINA '13)*, pp. 649–654, March 2013.
- [7] D. Hu, N. Chen, S. Dong, and Y. Wan, "A user preference and service time mix-aware resource provisioning strategy for multi-tier cloud services," *AASRI Procedia*, vol. 5, pp. 235–242, 2013, Proceedings of the AASRI Conference on Parallel and Distributed Computing and Systems.
- [8] Y. Shi, X. Jiang, and K. Ye, "An energy-efficient scheme for cloud resource provisioning based on CloudSim," in *Proceedings of the*

- IEEE International Conference on Cluster Computing*, pp. 595–599, September 2011.
- [9] H. Zhang, P. Li, Z. Zhou, X. Du, and W. Zhang, “A performance prediction scheme for computation-intensive applications on cloud,” in *Proceedings of the IEEE International Conference on Communication (ICC '13)*, pp. 1957–1961, 2013.
 - [10] G. Reig, J. Alonso, and J. Guitart, “Prediction of job resource requirements for deadline schedulers to manage high-level SLAs on the cloud,” in *Proceedings of the 9th IEEE International Symposium on Network Computing and Applications (NCA '10)*, pp. 162–167, July 2010.
 - [11] V. Manish, G. R. Gangadharan, V. Ravi, and N. C. Narendra, “Resource demand prediction in multi-tenant service clouds,” in *Proceedings of the IEEE International Conference on Cloud Computing in Engineering Markets*, pp. 1–8, IEEE, Bangalore, India, October 2013.
 - [12] F. Ramezani, J. Lum, and F. Hussain, “An online fuzzy decision support system for resource management in cloud environments,” in *Proceedings of the Joint IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS '13)*, pp. 754–759, Edmonton, Canada, June 2013.
 - [13] X. Kong, C. Lin, Y. Jiang, W. Yan, and X. Chu, “Efficient dynamic task scheduling in virtualized data centers with fuzzy prediction,” *Journal of Network and Computer Applications*, vol. 34, no. 4, pp. 1068–1077, 2011.
 - [14] C. Guang, B. Xiaoying, H. Xiaofei et al., “Cloud performance trend prediction by moving averages,” *Journal of Frontiers of Computer Science and Technology*, vol. 6, no. 6, pp. 495–503, 2012.
 - [15] Z. Xiao, W. Song, and Q. Chen, “Dynamic resource allocation using virtual machines for cloud computing environment,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1107–1117, 2013.
 - [16] D. Xu, S. Yang, and H. Luo, “A fusion model for CPU load prediction in cloud computing,” *Journal of Networks*, vol. 8, no. 11, pp. 2506–2511, 2013.
 - [17] Y.-C. Chang, R.-S. Chang, and F.-W. Chuang, “A predictive method for workload forecasting in the cloud environment,” in *Advanced Technologies, Embedded and Multimedia for Human-Centric Computing*, vol. 260 of *Lecture Notes in Electrical Engineering*, pp. 577–585, Springer, Amsterdam, The Netherlands, 2014.
 - [18] Q. Yang, C. Peng, Y. Yu et al., “Host load prediction based on PSR and EA-GMDH for cloud computing system,” in *Proceedings of the 3rd IEEE International Conference on Cloud and Green Computing (CGC '13)*, pp. 9–15, October 2013.
 - [19] J. J. Prevost, K. M. Nagothu, B. Kelley et al., “Prediction of cloud data center networks loads using stochastic and neural models,” in *Proceedings of the 6th International Conference on System of Systems Engineering*, pp. 276–281, 2011.
 - [20] S. Islam, J. Keung, K. Lee, and A. Liu, “Empirical prediction models for adaptive resource provisioning in the cloud,” *Future Generation Computer Systems*, vol. 28, no. 1, pp. 155–162, 2012.
 - [21] W. Fang, Z. Lu, J. Wu, and Z. Cao, “RPPS: a novel resource prediction and provisioning scheme in cloud data center,” in *Proceedings of the IEEE 9th International Conference on Services Computing (SCC '12)*, pp. 609–616, IEEE, Honolulu, Hawaii, USA, June 2012.
 - [22] J. Cao, J. Fu, M. Li, and J. Chen, “CPU load prediction for cloud environment based on a dynamic ensemble model,” *Software: Practice and Experience*, vol. 44, no. 7, pp. 793–804, 2014.
 - [23] M. Xia, X. Liang, and F. Han, “Water quality comprehensive assessment approach based on T-S fuzzy neural network and improved FCM algorithm,” *Computers and Applied Chemistry*, vol. 30, no. 10, pp. 1197–1202, 2013.
 - [24] R. Singh, V. Vishal, T. N. Singh, and P. G. Ranjith, “A comparative study of generalized regression neural network approach and adaptive neuro-fuzzy inference systems for prediction of unconfined compressive strength of rocks,” *Neural Computing and Applications*, vol. 23, no. 2, pp. 499–506, 2013.
 - [25] C.-H. Lu, “Wavelet fuzzy neural networks for identification and predictive control of dynamic systems,” *IEEE Transactions on Industrial Electronics*, vol. 58, no. 7, pp. 3046–3058, 2011.
 - [26] Y.-Y. Lin, J.-Y. Chang, and C.-T. Lin, “Identification and prediction of dynamic systems using an interactively recurrent self-evolving fuzzy neural network,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 2, pp. 310–321, 2013.
 - [27] S. Karimi, O. Kisi, J. Shiri, and O. Makarynskyy, “Neuro-fuzzy and neural network techniques for forecasting sea level in Darwin Harbor, Australia,” *Computers and Geosciences*, vol. 52, pp. 50–59, 2013.
 - [28] K. Li, H. Su, and J. Chu, “Forecasting building energy consumption using neural networks and hybrid neuro-fuzzy system: a comparative study,” *Energy and Buildings*, vol. 43, no. 10, pp. 2893–2899, 2011.
 - [29] P.-C. Chang, C.-Y. Fan, and J.-J. Lin, “Monthly electricity demand forecasting based on a weighted evolving fuzzy neural network approach,” *International Journal of Electrical Power and Energy Systems*, vol. 33, no. 1, pp. 17–27, 2011.
 - [30] A. K. Lohani, R. Kumar, and R. D. Singh, “Hydrological time series modeling: a comparison between adaptive neuro-fuzzy, neural network and autoregressive techniques,” *Journal of Hydrology*, vol. 442–443, pp. 23–35, 2012.
 - [31] S. Alvisi and M. Franchini, “Fuzzy neural networks for water level and discharge forecasting with uncertainty,” *Environmental Modelling and Software*, vol. 26, no. 4, pp. 523–537, 2011.
 - [32] C. Chaochao, B. Zhang, G. Vachtsevanos, and M. Orchard, “Machine condition prediction based on adaptive neuro-fuzzy and high-order particle filtering,” *IEEE Transactions on Industrial Electronics*, vol. 58, no. 9, pp. 4353–4364, 2011.
 - [33] C. Chen and G. Vachtsevanos, “Bearing condition prediction considering uncertainty: an interval type-2 fuzzy neural network approach,” *Robotics and Computer-Integrated Manufacturing*, vol. 28, no. 4, pp. 509–516, 2012.
 - [34] G. Zhao, H. Yu, T. Ji, and H. Song, “Adaptive resource provisioning for cloud computing,” *Telecommunications Science*, no. 1, pp. 31–37, 2012.
 - [35] P. Saripalli, G. V. R. Kiran, R. R. Shankar, H. Narware, and N. Bindal, “Load prediction and hot spot detection models for autonomic cloud computing,” in *Proceedings of the 4th IEEE/ACM International Conference on Cloud and Utility Computing (UCC '11)*, pp. 397–402, Victoria, Australia, December 2011.
 - [36] M. Andreolini and S. Casolari, “Load prediction models in web-based systems,” in *Proceedings of the 1st International Conference on Performance Evaluation Methodologies and Tools*, October 2006.
 - [37] R. H. Abiyev, “Fuzzy wavelet neural network based on fuzzy clustering and gradient techniques for time series prediction,” *Neural Computing and Applications*, vol. 20, no. 2, pp. 249–259, 2011.
 - [38] S.-D. Fan and H.-Y. Zhao, “Short-term load forecast based on fuzzy neural network,” *Mechanical Manufacture and Automation*, no. 2, pp. 182–184, 2013.

- [39] T. Chaira, “A novel intuitionistic fuzzy C means clustering algorithm and its application to medical images,” *Applied Soft Computing Journal*, vol. 11, no. 2, pp. 1711–1717, 2011.
- [40] J. Jiang, X. Meng, H. Li, and X. Zhuang, “Study on application of subtractive clustering and adaptive network-based fuzzy inference system in network fault diagnosis,” *Computer Engineering and Applications*, vol. 47, no. 8, pp. 76–79, 2011.
- [41] Y. Wen, D. Meng, and J.-F. Zhan, “Adaptive virtualized resource management for application’s SLO guarantees,” *Journal of Software*, vol. 24, no. 2, pp. 358–377, 2013.
- [42] Data Flow Statistics and Analysis, 2014, <http://tongji.cnzz.com/>.