A HIGH-ACCURACY SELF-ADAPTIVE RESOURCE DEMANDS PREDICTING METHOD IN IAAS CLOUD ENVIRONMENT

Z. Chen[*], *Y. Zhu*[*], *Y. Di*[*], *S. Feng*[*], *J. Geng*[†]

Abstract: In IaaS (Infrastructure as a Service) cloud environment, users are provisioned with virtual machines (VMs). However, the initialization and resource allocation of virtual machines are not instantaneous and usually minutes of time are needed. Therefore, to realize efficient resource provision, it is necessary to know the accurate amount of resources needed to be allocated in advance. For this purpose, this paper proposes a high-accuracy self-adaptive prediction method using optimized neural network. The characters of users' demands and preferences are analyzed firstly. To deal with the specific circumstances, a dynamic self-adaptive prediction model is adopted. Some basic predictors are adopted for resource requirements prediction of simple circumstances. BP neural network with self-adjusting learning rate and momentum is adopted to optimize the prediction results. High-accuracy self-adaptive prediction is realized by using the prediction results of basic predictors with different weights as training data besides the historical data. Feedback control is introduced to improve the whole operation performance. Statistic validation of the method is conducted adopting multiple evaluation criteria. The experiment results show that the method is promising for effectively predicting resource requirements in the cloud environment.

Key words: *IaaS cloud environment, user demand prediction, high-accuracy self-adaptive prediction, BP neural network, self-adjusting learning rate and momentum, statistic evaluation*

Received: June 5, 2014 **DOI:** 10.14311/NNW.2015.25.026
Revised and accepted: January 6, 2015

1. Introduction

In cloud computing [12,10], high efficient resource provision is important for maximizing the utility. Especially in IaaS mode cloud computing, users are provided with virtual machines that are composed of virtual hardware including virtual CPU, virtual GPU, virtual memory, virtual storage, etc. These virtual hardware resources are virtualized by hypervisor [23]. The virtual resources are dynamically

[*]Zhijia Chen – Corresponding author, Yuanchang Zhu, Yanqiang Di, Shaochong Feng, Department of Electronic and optics, Mechanical Engineering College, Shijiazhuang, 050003, China, E-mail: `youshenshui@163.com`

[†]Jingtao Geng, Hebei chemical and pharmaceutical College, Shijiazhuang, 050003, China

assembled as a virtual machine. According to their requirements, users send requests to the cloud center and try to obtain the resources. Cloud center allocates resources in the form of VMs to users based on the received requests. However, before provisioning these resources for users, some time is needed to prepare and initialize the instances, i.e. the VMs. In the other hand, when the VM is running, resources dynamic adjusting is also needed to guarantee the QoS (Quality of Service) of cloud computing. In some cases, resources have to be dynamically rearranged based on customers' demands [20]. However, the rearrangement of resources cannot take effect instantly and leads to insufficient elastic management of resources [19]. This severely influences the utility and the QoS of cloud computing.

It stands to reason that resource provisioning in the cloud environment is influenced directly by demands predictions [7]. In order to know how to allocate resources beforehand, it is important to characterize workload fluctuations accurately. The workload is related with user demands closely, as the load of the VM is decided mainly by users' tasks and demands. In addition, users' preferences contribute to workload variants. 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 main contributions of this paper are listed in the following:

1. We make a full analysis of the characteristics of resource requirements. Different resource requirement models based on time and degree of fluctuations are analyzed.

2. We propose a high-accuracy self-adaptive prediction method based on BP neural network. "Self-adaptive" means that the method is adaptive to any kind of circumstances and the parameters of the neural network are self-adaptive. We achieve "high-accuracy" by not only using historical data for neural network training, but also the base predictors' output results with different weights are adopted as the training data.

3. BP Neural network is optimized with momentum and self-adjusting learning rate, which improves the robustness and the accuracy performance.

4. To evaluate the prediction algorithm, some statistic indexes are introduced to compare with other algorithms, including MSE (Mean Squared Error), MAE (Mean Absolute Error), SSE (Sum Squared Error), etc.

The remainder of the paper is organized as follows. Section 2 compares some related work and analyzes the merits and demerits. Section 3 introduces the main structure of the self-adaptive online prediction system. Some preparing work including user preference data and different resource demands circumstances are analyzed. The idea and method using optimized neural network to improve the prediction performance are proposed in Section 4. Experiments are conducted in Section 5 and the results are analyzed. Section 6 concludes the paper and figures out the future research emphasis.

2. Related work

Researches on resource demand prediction are mainly focused on how to save energy [16], improve performance [7,24], increase profit [20,14,18] and so on. To optimize resource management and task scheduling, Fahimeh Ramezani et al. [13] introduce a prediction method for predicting VM workload pattern and VM migration time using fuzzy expert system. However, only a simple prediction model is depicted and the details are not explicated. Xiangzhen Kong et al. [9] use type-1 and type-2 fuzzy logic systems to model the uncertain workload and vague availability of virtualized server nodes. By adopting fuzzy algorithm, the performance of prediction method is more robust but the accuracy is decreased. This method needs to be combined with other prediction methods to realize high performance.

There are also some methods predicting resource requirements according to the lasting time. Guang Chen et al. [4] propose an approach for long-term trend prediction using moving averages 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, but the short-term prediction is not mentioned.

To balance the performance and the system cost, some researchers make efforts to maximize the system utility. Zhen Xiao et al. [21] introduce fast up and slow down algorithm to maximize the performance while maintain the stability. As workload has an obvious nonlinear feature, many machine-learning algorithms have also been used to support its prediction. Neural network is introduced for workload prediction. Dayu Xu et al. [22] propose a genetic algorithm optimized wavelet Elman neural network prediction model to predict the CPU load with current load sequence. In order to improve the resource allocation efficiency, a neural network model with learning algorithm is adopted to predict the workload of the cloud server [3]. The algorithm prevents cloud center from the problem of inadequate resources. To minimize energy consumption of the cloud systems while maintaining performance levels, John J. Prevost et al. [11] propose stochastic and neural models for predicting cloud data center network load. The results provide a framework that can be utilized to allocate the resources in the way of maintaining both an optimal power consumption level as well as all existing SLAs. Combining with the typical predicting methods, such as sliding window method [8], auto regression model [5], exponential smoothing model [2] and so on, neural network works well in predicting the workload. However, the self-adaptive online prediction method is not mentioned, real-time performance is hardly to achieve. Moreover, there is still much room for improving the prediction accuracy.

As mentioned above, there have been many studies on the resource demand and workload prediction. Unfortunately, those methods are hardly to adapt various circumstances and provide accurate prediction results. The robustness and the real-time performance also need to be improved. Thus, a dynamic and accurate prediction algorithm is necessary to be researched.

3. System overview and preparation for prediction

3.1 System overview

Before the prediction of the demand and workload, we firstly analyze the user requests, including the utilization data structure, content and amount of historical resource. By analyzing the historical data, we may draw some conclusions such as user preference and so on. To realize effective and accurate prediction, the short-term and long-term prediction needs to be specified. The fluctuating period and the flat period also need to be separately treated with. In addition, we define fluctuation-threshold (t_{u}) and flat-threshold (t_{a}) to distinguish the flat period and fluctuating period. In different periods, different basic prediction methods including second moving average model (SMA), exponential moving method (EMA), auto regression model (AR), trend seasonality model (TSM), etc are adopted according to the characters. The output of the base predictors is sent to the BP neural network. BP Neural network takes historical data and the base prediction value as training data, which improves the accuracy of the results. Besides, self-adjusting learning rate with momentum is introduced in the neural network. The output of neural network is used to instruct the resource allocation in IaaS cloud center. Prediction results and the actual resource demands are evaluated using statistic analysis and different criteria are measured. The evaluation results are returned to the historical database as feedback to improve the prediction performance. The overview of resource demands prediction system is depicted in Fig. 1.

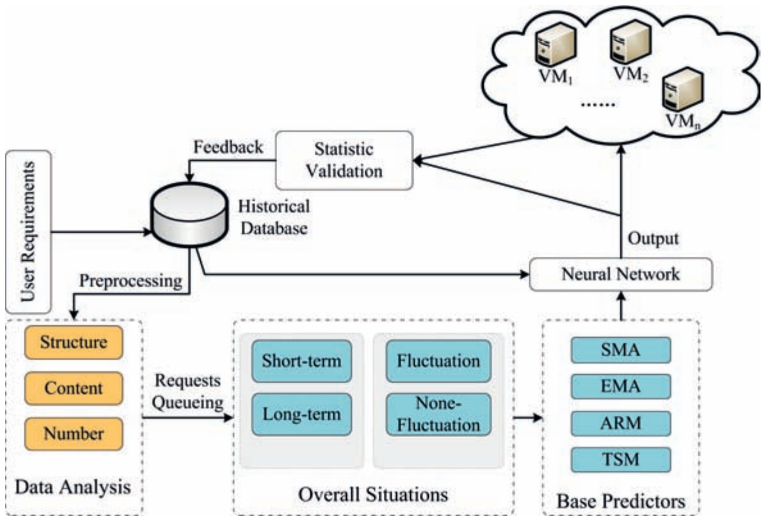


Fig. 1 *The overview of the prediction system.*

3.2 User demands analysis and description

User demands prediction is the base of resource provisioning. In IaaS mode cloud environment, users are provided with VMs that are composed of virtual CPU, virtual GPU, etc. A user who has certain applications running on VMs usually does specific operations more frequently, and more corresponding resources are need. This circumstance can be defined as preference. In cloud center, if only we can continuously monitor the resources utilization of users' VMs and effectively predict the resources demands, we will be able to dynamically schedule resources for users in according with their preferences and realize high QoS of cloud computing. Thus, we collect users' VMs information in data center and we need not know what kind of applications are running on their VMs exactly. The information data is classified into fine-grained structure. The data structure of n different users' resources requests is described as $\mathbf{p}_i\,\{\text{CPU, MEMORY, GPU, NETWORK}\}$, $i = 1, 2, \ldots$ For different element p_{ij} in the vector $\mathbf{p}_i$, to express different preference degree of each element, corresponding weight w_{ij} is assigned to p_{ij}:

$$\begin{cases} w_{ij} = \dfrac{p_{ij}}{\sum\limits_{i=1}^{n} p_{ij}} \\ \sum\limits_{i} w_{ij} = 1 \end{cases}. \tag{1}$$

3.3 Overall situations of user demands

As depicted in Fig. 1, there are some opposite circumstances needed to be considered, such as long-term demands and short-term demands, fluctuations and non-fluctuations. Fluctuating period is an abnormally violent vibration on a cloud resource over a period of time.

In long-term resource requirements, there are some characters, as we summarize in the following: (1) the regularity is more obvious than short-term as the long-term users may show some repetition regularity; (2) in the long running of the system, there may be some fluctuating periods and some flat periods. While for short-term requirements, the regularity may not be much obvious, but the fluctuating feature is more noticeable. Therefore, long-term or short-term resource requirements are not conflicted with fluctuating or flat periods. For long-term data, the regularity can be summarized; the flat and fluctuant periods also need to be distinguished. For short-term, the regularity is not easy to figure out and the fluctuating should be processed. The prediction speed needs to be ensured as the short-term resource provision gives first place to quick response than other performance. The difference between short-term and long-term processing mainly lies in the fluctuating period processing. Hereby we discuss the fluctuating period and flat period respectively.

1) Flat period procession

   In most time, resource demands in IaaS cloud computing do not vary much frequently. Take private cloud in a corporation as an example, workers use the virtual machines that locate in cloud center to complete their daily tasks. As the tasks do not change frequently, the resources used to process these tasks will not change too quickly. The most time may be flat period. So is the education cloud in school. As the course arrangement of a school is always fixed, the same virtual machine may mainly run the fixed tasks in one class, and the resource demands will not change frequently and the period could be considered as flat period.

Based on the characters of flat period, second moving average (SMA) [25] algorithm 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, i.e. $\mathbf{x} = [x_t,\, x_{t-1}, \ldots,\, x_{t-(N-1)}]$ over the historical time interval [t-(N-1), t]. Fig. 2 depicts the sliding window model with a window size N.

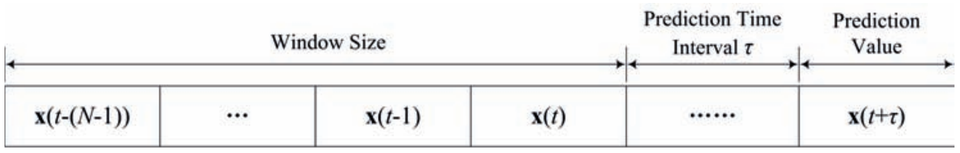


Fig. 2 *The model of sliding window.*

The predicted output value $x_{t+\tau}$ after a time interval τ is the dependent variable of set $\mathbf{x}$, where $\mathbf{x} = [x_1, x_2, \cdots, x_N]$. The relationship can be abstracted as follows:

$$x_{t+\tau} = f(\mathbf{x},\, \tau). \tag{2}$$

The i-th user's resource requirement at time t+τ can be expressed as follows:

$$x_{t+\tau}(i) = a_t(i) + \tau b_t(i). \tag{3}$$

In Eq. (3), $x_{t+\tau}$ is the prediction value in time $t + \tau$, τ is the time sequence number to be predicted. Variables $a_t(i)$ and $b_t(i)$ satisfy the following constraint equations:

$$a_t(i) = 2M_t^{(1)}(i) - M_t^{(2)}(i), \tag{4}$$

$$b_t(i) = \frac{2}{N-1}[M_t^{(1)}(i) - M_t^{(2)}(i)]. \tag{5}$$

In Eq. (4) and Eq. (5), $M_t^{(1)}(i)$ and $M_t^{(2)}(i)$ represent the first moving average value and the second moving average value of the i-th user requested resources at the t-th time. In addition, $M_t^{(1)}(i)$ and $M_t^{(2)}(i)$ can be expressed as

$$M_t^{(1)}(i) = \frac{x_t(i) + x_{t-1}(i) + \cdots + x_{t-(N-1)}(i)}{N}, \tag{6}$$

$$M_t^{(2)}(i) = \frac{M_t^{(1)}(i) + M_{t-1}^{(1)}(i) + \cdots + M_{t-1}^{(1)}(i)}{N}. \tag{7}$$

Then the total amounts of the resources requested by all the m users are:

$$x_{t+\tau} = \sum_{i=1}^{m} x_{t+\tau}(i). \tag{8}$$

From the analysis above, we can see that the prediction value at 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.

2) Fluctuating period procession

Although resource requirements in IaaS cloud computing do not change dramatically at most time, the fluctuating period deserves more attention as the fluctuating period may cause resource shortage or other problems that affect the QoS of cloud computing. Let us take education cloud in school as an example as before. Resource demands will fluctuate between different classes as these virtual machines could be used to process different tasks. For example, from 9:00 to 10:00, the virtual machine may be used to do computing tasks and the main resources demands are virtual CPU resources. However, from 10:00 to 11:00, the same virtual machine may be used to play teaching video and the main resources demands will be virtual GPU resources rather than virtual CPU resources. Thus, it is important to process the demands fluctuation period duly to guarantee the QoS of cloud computing.

Exponential moving average (EMA) algorithm is an effective method for short-term prediction, and particularly suitable for time series prediction of the non-seasonal 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:

$$x_{t+1} = \alpha \mathbf{x}(t, N) + (1 - \alpha) x_t. \tag{9}$$

In Eq. (9), $\mathbf{x}(t, N)$ is the moving average value between the past time t-(N-1) and the current time t. The time interval is N. α is the smoothing constant that can be calculated by $\alpha = \frac{2}{N+1}$. We can see that confines to [0, 1].

The EMA method gives a higher weight to the latest 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 [15]. However, there will be some delay as the window size increases. Based on Mauro Andreolini[1], in non-linear load trackers, the polynomial orders should be properly selected. If the order is low (degree ≤ 2), then the algorithm will not react quickly enough to load changes. If the order is high (degree ≥ 4), the algorithm will be unnecessarily complex and some undesirable sparks will be introduced in and the cost may be too expensive for a run-time context.

3) The identification of flat period and fluctuating period

We give different prediction methods according to the fluctuation levels, however, it is difficult to know or identify the boundary of different fluctuation levels in the overall situation. In this section, we define "fluctuation-threshold" (t_{u}) and "flat-threshold" (t_{a}) to distinguish the fluctuating 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 time intervals, if the difference $d_t$of prediction value x_t and x_{t-1}

in series $\left\{x_t, x_{t-1}, \cdots, x_{t-(n-1)}\right\}$ is greater than a certain value d_u, then t_{u} is reached. If the difference of x_t and x_{t-1} is less than a certain value d_l, then t_{a} is reached. For resource type i, the demands are experiencing a fluctuating period if the demand data in last k time interval satisfies the condition $g_i \geq t_{\mathrm{u}}$, where g_i means the fluctuating degree of the prediction trend, t_{u} is the upper limit value. Type i resource demands are experiencing a flat period if the demand data in last k time interval satisfies condition $g_i \leq t_{\mathrm{a}}$. If $t_{\mathrm{a}} \leq g_i \leq t_{\mathrm{u}}$, the demands of resource i is intervenient flatness and fluctuation.

The above procedure can be illustrated by the pseudo code:

```
for prediction value series $\left\{x_t, x_{t-1}, \cdots, x_{t-(n-1)}\right\}$ do
  calculate the differences $d_t, d_{t-1}, \cdots, d_{t-(n-1)}$ of the adjacent value
end for
set $d_u$ and $d_l$ as the threshold of fluctuation
for the differences $d_t, d_{t-1}, \cdots, d_{t-(n-1)}$ do
  compare them with benchmark $d_u$ and $d_l$
  if they are greater than $d_u$ then
    $t_{\mathrm{u}}$ is reached
  else if they are less than $d_l$then
    $t_{\mathrm{a}}$ is reached
  end if
end for
for each type resource $i$ do
  compare the fluctuating degree with $t_{\mathrm{u}}$ and $t_{\mathrm{a}}$
  if it is greater than $t_{\mathrm{u}}$ then
    resource demands are experiencing a fluctuating period
  else if it is less than $t_{\mathrm{a}}$ then
    resource demands are experiencing a flat period
  else
    resource demands is intervenient flatness and fluctuation
  end if
end for
```

4. Self-adaptive prediction using BP neural network

In order to predict the resource demands accurately and effectively, a self-adaptive prediction method with different base prediction models ensemble and BP neural network is proposed in this section. By adopting different base prediction models, various workload occasions can be estimated accordingly, and the most likely future outcome can be predicted effectively. The introduction of BP neural network guarantees the robustness of the prediction system and accuracy of prediction results. With the results of base predictors, the neural network will have a better predicting performance. The core of the self-adaptive prediction model adopts a two-level structure, as shown in Fig. 3. The first level is an ensemble that contains different base predictor models. The output of the first level is sent to the second level– neural network level, which is responsible for optimizing the precision and the robustness of the prediction results.

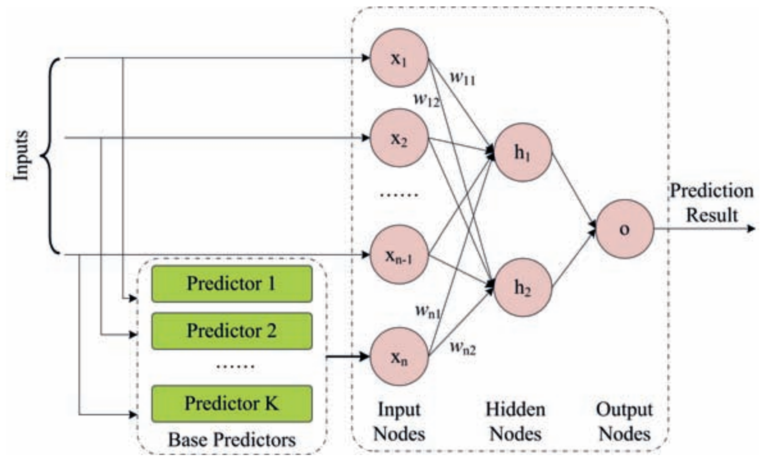


Fig. 3 *Self-adaptive prediction model.*

4.1 Base prediction models

As we know, diversity is necessary for the survival and evolution of species ensemble model. So as to the performance of the prediction models, it is important to introduce the diversity in 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.

1) Auto regression model

   Auto regression model (ARM) is one of the linear models used for estimating the relationships between one dependent output variable y and one or more independent variables x_i. It shows how the dependent value changes along with the independent variables. The fundamental of the method is 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:

$$y = \alpha_1 x_{t-1} + \alpha_2 x_{t-2} + \cdots \alpha_p x_{t-p} + e_t, \tag{10}$$

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

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 d_i can be calculated by the equation:

$$d_i = \frac{a_i}{\sum\limits_i a_i},$$

   where a_i is the average value of the ith period. 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.

3) Moving average method

   By judging whether the monitored data crosses over the moving average, moving averages (MA) method predicts the future trend is descending or ascending. m_N is a moving average of monitoring data series R_i with length of N, expressed as equation: $m_N = \sum\limits_{i-(N-1)}^{i} \frac{R_i}{N}$, s.t. $N \leq i$. If the monitoring data cross over the moving average upward, then it indicates that an ascending trend is coming. While if the monitoring data cross over the moving average upward, it indicates that a descending trend is coming.

4.2 Neural network model

A neuron is capable of reflecting a simple nonlinear intrinsic attribute. By self-organization of these basic units, the constructed neural network can reconstruct any nonlinear functions [6]. The self-adaptive and anti-interference capacity improves the prediction system robustness. We construct a three-layer neural network as shown in Fig. 3. The input layer has n input neurons, which are abstracted as vector $\mathbf{x} = [x_1, x_2, \cdots, x_n]$. The output layer only has one output node $\mathbf{o}$. The hidden layer is abstracted as $\mathbf{h} = [h_1, h_2, \cdots h_m]$. The number of nodes in the hidden layer can be calculated by the equation $m = \sqrt{n+1} + a$, where a is an adjusting constant confined with [1, 9]. The main idea of prediction is concluded in Eq. (2). The use of neural network is to fit function f. The connection between two neurons in different layer is called a synapse. A synapse is correspondence with a weight w_{ij}. The number i and j are the serial number of the two connected neurons in different layers. The weight needs to be calculated in the training procedure.

1) The output and error of a neuron We introduce a time series based back-propagation neural network model to predict the resource demands. In this model, the network is trained by a series of historical data. In the training

period, the synapses are assigned with random weights, and then the weights are optimized by using the back propagation error. To explain the model clearly, the data process procedure is figured out in Fig. 4. The input-output relationship of a neuron is:

$$y = f(\mathbf{w} \cdot \mathbf{x} + b), \tag{11}$$

where $\mathbf{w}$ is the weight vector and $\mathbf{x}$ is the input vector.

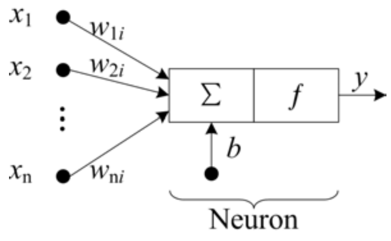


Fig. 4 *The sketch map of data procession in a neuron.*

Suppose the number of the nodes in the $l-$th layer is n_l, $y_k^{(l)}$ is the output of the node k in level l. According to Eq. (11), the output $y_k^{(l)}$ can be expressed as follows:

$$\begin{cases} \bar{y}_k^{(l)} = \sum\limits_{j=1}^{n_{l-1}} w_{jk}^{(l)} y_j^{(l-1)} \\ y_k^{(l)} = f(\bar{y}_k^{(l)}), \quad k = 1, 2, \cdots, n_l \end{cases}, \tag{12}$$

where $w_{jk}^{(l)}$ is the weight of the synapsis that connects the nodes in the (l-1)$-$th layer and the nodes in the l-th layer. Define the output error of the k-th output node as

$$e_k = y_k - \hat{y}_k, \tag{13}$$

where $\hat{y}_k$ is the actual output of the k-th node. The instantaneous error energy is:

$$E = \frac{1}{2} |y_k - \hat{y}_k|^2 = \frac{1}{2} e_k^2. \tag{14}$$

As there is only one output node in the output layer, so E is also known as the total instantaneous error energy. The goal of the network is to minimize the error energy E by readjusting the weights.

2) Adjustment of the sigmoid function

In the error back propagation procedure, the steepest descend method is adopted to adjust the network weights. The correction of weight $w_k^{(l)}$ can be calculated by

$$\Delta w_k^{(l)} = -\eta \frac{\partial E}{\partial w_k^{(l)}} = \eta \delta_k^{(l)} \bar{y}_k^{(l)}, \tag{15}$$

where η is the back propagation learning rate, local gradient δ_k^l is the partial derivative of error energy for the input $\bar{y}_k^{(l)}$:

$$\delta_k^{(l)} = \frac{\partial E}{\partial \bar{y}_k^{(l)}} = \frac{\partial E}{\partial e_k}\frac{\partial e_k}{\partial y}\frac{\partial y}{\partial \bar{y}_k^{(l)}} = e_k^{(l)} f'(\bar{y}_k^{(l)}). \tag{16}$$

Eq. (16) also indicates that the local gradient δ_k^l is the product of error signal $e_k^{(l)}$ and the derivative of corresponding transfer function $f'(\bar{y}_k^{(l)})$. Therefore, Eq. (15) can also be written as

$$\Delta w_k^{(l)} = \eta e_k^{(l)} f'(\bar{y}_k^{(l)}) y_k^{(l-1)}. \tag{17}$$

According to Eq. (17), one of the key factors for the adjustment of weight $\Delta w_k^{(l)}$ is the error signal $e_k^{(l)}$. In addition, we need to consider different occasions according to the different locations of neuron k. If the neuron is in the output layer, the error can be calculated by Eq. (13) as each of the output nodes provides the expected response signal. Then the local gradient can be calculated as

$$\delta_k^{(l)} = (y_k - \hat{y}_k) f'(\bar{y}_k^{(l)}). \tag{18}$$

If the neuron is in the hidden layer, the back propagation equation of local gradient $\delta_k^{(l)}$ is:

$$\delta_k^{(l)} = \sum_{j=1}^{n_{k+1}} \delta_j^{(l+1)} w_{kj}^{(l+1)} f'(\bar{y}_k^{(l)}). \tag{19}$$

To predict the complex and sudden changes of the demand curve effectively, the transfer function $f(\bar{y}_k^{(l)})$ in Eq. (12) usually adopts the form of nonlinear function:

$$f(\bar{y}_k^{(l)}) = \frac{1}{1 + \exp(-\bar{y}_k^{(l)})}. \tag{20}$$

From Eq. (20) we can see that the output of transfer function f confines to [0,1]. The region that approaches to 0 or 1 is called a saturation region, the region that apart from 0 and 1 is called an unsaturation region, as shown in Fig. 5.

In the saturation region, the dependent variable is not sensitive to the independent variables, which causes the connected weights unable to modulate the neuron output effectively. If the current output is not the expected value and the changes of the weights are small, then the output of the neuron is hardly adjusted to the best, and the convergence rate is influenced. Therefore, we modify the form of the transfer function as

$$f_m(\bar{y}_k^{(l)}) = \frac{1}{a + b\exp(-c\bar{y}_k^{(l)})}. \tag{21}$$

In Eq. (21), variables a, b and c are the adaption parameters for changing the saturation area of the sigmoid function and improving the convergence

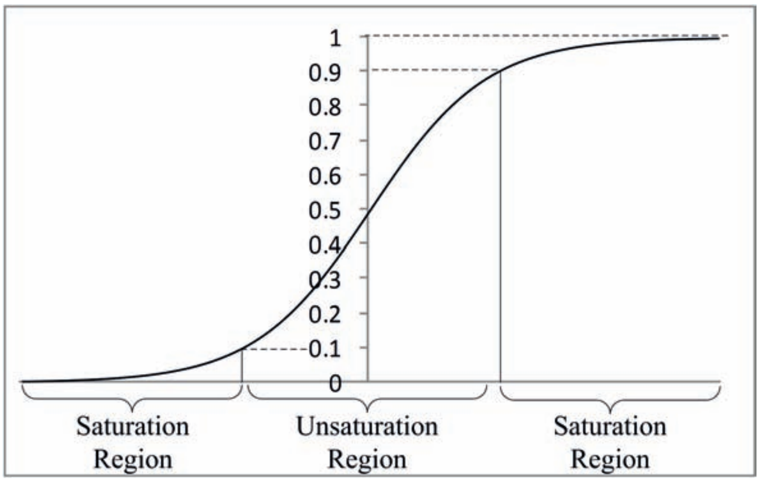


Fig. 5 *The saturation and unsaturation region of the sigmoid function.*

speed. Calculate the differential of Eq. (21):

$$f'(\bar{y}_k^{(l)}) = \frac{bc\exp(-c\bar{y}_k^{(l)})}{(a+b\exp(-c\bar{y}_k^{(l)}))^2}. \tag{22}$$

According to Eq. (12), Eq. (21) and Eq. (22), $f'(\bar{y}_k^{(l)})$ is:

$$f'(\bar{y}_k^{(l)}) = cf(\bar{y}_k^{(l)})(1 - af(\bar{y}_k^{(l)})) = cy_k^{(l)}(1 - ay_k^{(l)}). \tag{23}$$

According to Eq. (18) and Eq. (23), if the neuron is in the output layer, the local gradient $\delta_k^{(l)}$ is:

$$\delta_k^{(l)} = c(y_k - \hat{y}_k)\hat{y}_k(1 - a\hat{y}_k). \tag{24}$$

According to Eq. (19) and Eq. (23), if the neuron is in the hidden layer, the local gradient $\delta_k^{(l)}$ is:

$$\delta_k^{(l)} = \sum_{j=1}^{n_{k+1}} \delta_j^{(l+1)} w_{kj}^{(l+1)} c\hat{y}_k(1 - a\hat{y}_k). \tag{25}$$

3) Self adjusting learning rate with momentum

As we can see from Eq. (15), 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 weight space becomes smooth. However, the learning rate is decreased. If η is too large, the learning rate will improve 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 Eq. (15), and it is represented as follows:

$$\Delta w_k^{(l)} = \eta \delta_k^{(l)} \bar{y}_k^{(l)} + \lambda \Delta w_k^{(l-1)}, \tag{26}$$

where λ is the momentum constant. The using of the momentum constant is a minor revise for refreshing the weight. However, it brings advantages for the learning speed of the algorithm.

In addition, we introduce a "progressive-increase" and a "conservative-decrease" method to adjust the learning rate η. If the error declines in the training procedure, we may draw the conclusion that the modification direction is 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 slow down and a smaller value is assigned to variable k. Meanwhile, the former modification should also be abandoned. The method is shown as follows:

$$\eta(i+1) = \begin{cases} k_{\text{inc}}\eta(i) & E(i+1) < E(i) \\ k_{dec}\eta(i) & E(i+1) > E(i) \end{cases}, \tag{27}$$

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

4.3 Validation criteria

To evaluate the performance of the prediction system, we use a series of metrics[8] including MAE (Mean Absolute Error), MSE (Mean Squared Error), SSE(Sum Squared Error), PRED(x), etc. MSE and SSE represent the energy of the error.

1) MAE

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

$$e_{\text{MAE}} = \frac{1}{n} \sum_{i=1}^{n} |\hat{y}_i - y_i|, \tag{28}$$

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

2) Error energy

MSE represents the mean energy of error, while SSE represents the energy of the total error. MSE and SSE can be calculated by the following equations respectively:

$$\begin{cases} e_{\text{MSE}} = \frac{1}{n} \sum\limits_{i=1}^{n} (\hat{y}_i - y_i)^2 \\ e_{\text{SSE}} = \sum\limits_{i=1}^{n} (\hat{y}_i - y_i)^2 \end{cases}. \tag{29}$$

3) PRED(x)

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 PRED(5) as an example, according to formula (13), we define the relative error as

$$\tilde{e}_i = \frac{y_i - \hat{y}_i}{\hat{y}_i},$$

where $i = 1, 2, \cdots, n$ represents the series number of the output data series. The number of all the relative errors that meet the condition $-5\% \le \tilde{e}_i \le 5\%$ is supposed as $k_{(5)}$. The whole number is n. Then PRED(5) is defined as

$$\mathrm{PRED}(5) = \frac{k_{(5)}}{n}. \tag{30}$$

The measurement 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.

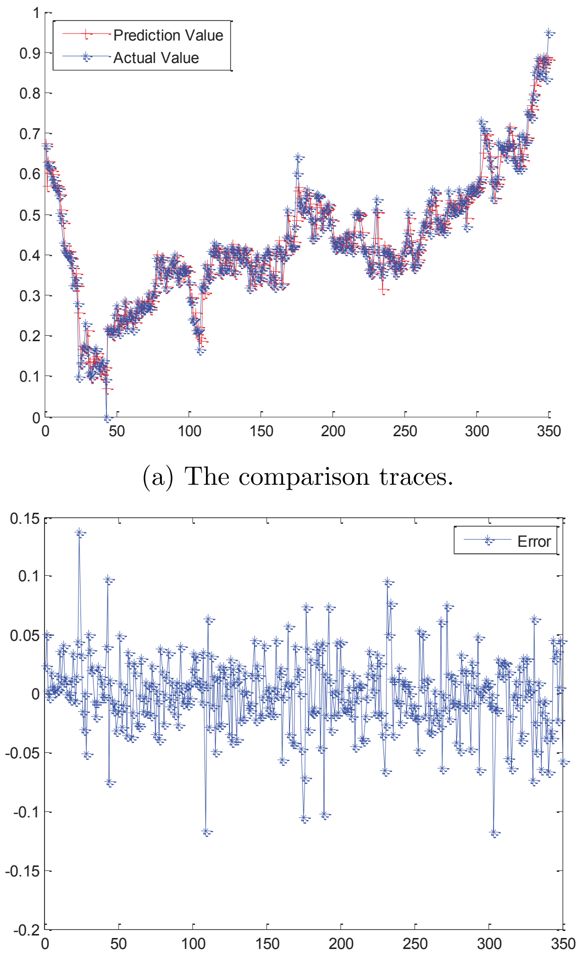
4.4 Feedback control

To optimize the performance of the resource demands prediction system, we introduce feedback control [17] into the system. In each prediction cycle, the feedback controller sends the actual resource demands and prediction results to historical database. The demands value is specified in fine-grained form, including the elements in data structure vector $\mathbf{p}_i$. In addition, the validation indexes of MAE, MSE, PRED(5), etc. 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. We use 350 days traffic data as the test reference. The prediction results of the self-adaptive real-time neural network (SRNN) this paper proposed are shown in Fig. 6. In Fig. 6(a), the "+" line represents the prediction value and the "*" line represents the actual value. The two curves fit with each other. In Fig. 6(b), we can see that the overall effect is promising. The maximum error is controlled in 15% and most of the error falls in $\pm 10\%$.

According to the main criteria we defined from Eq. (28) to Eq. (30), we test some base prediction methods including SMA, EMA, AR, basic neural network (BNN) and the self-adaptive real-time neural network (SRNN). The results are shown in Fig. 7. In Fig. 7(a) we can see that both the mean average error in regularity and the max error of the SRNN method are small. The performance of EMA and SMA are close to SRNN. However, the basic neural network is not very dedicated in prediction. So as to the MSE and SSE, the SRNN takes advantages of the other predictors' merits and realizes self-adaption and robustness. Finally, Fig. 7(d) shows the number of error that falls in 5%. The total numbers are all



(a) The comparison traces.

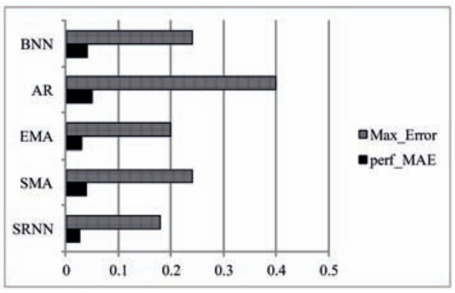
(b) The error trace.

Fig. 6 *The prediction results and error of SRNN.*

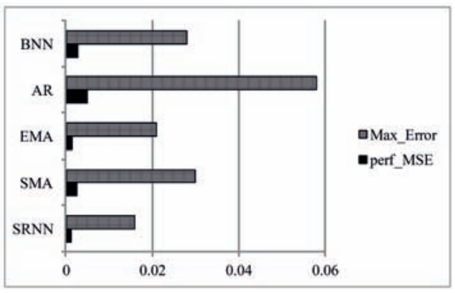
350. The black columniation represents the number that falls in 5%. The PRED(5) performance of SRNN is optimized and is superior to other predicting algorithms.

As we can see in Fig. 7, the performance of basic neural network is not good. There are some reasons: (1) the learning rate is not optimized and the expected training error cannot be reached. (2) The contradiction between convergence and learning rate is not well treated with. Therefore, we introduce the self-adjusting learning rate BP neural network with momentum.

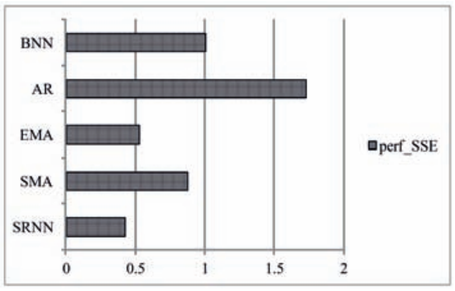
Fig. 8(a) depicts the performance of basic BP neural network. The curve shows the training procedure. After 2000 training cycles, the performance P_a of basic BP neural network is approximate 0.0087. Fig. 8(b) depicts the performance of the self-adjusting learning rate with momentum BP neural network. After 2000



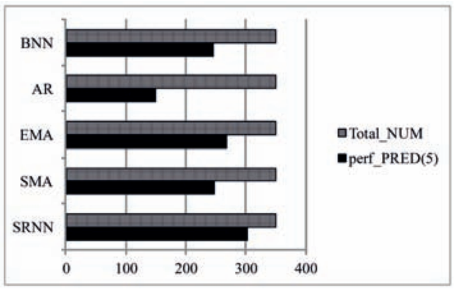
(a) MAE



(b) MSE



(c) SSE



(d) PRED(5)

Fig. 7 *Prediction error indexes comparison of different methods.*

training cycles, the performance P_b reaches 0.0031. The ratio of P_a and P_b is: $r = \frac{P_a}{P_b} = \frac{0.0087}{0.0031} = 2.8$. The learning rate of the proposed method is better than that of basic BP neural network. From above analysis, we can see that the performance of self-adjusting learning rate with momentum BP neural network is better than the performance of BP neural network.

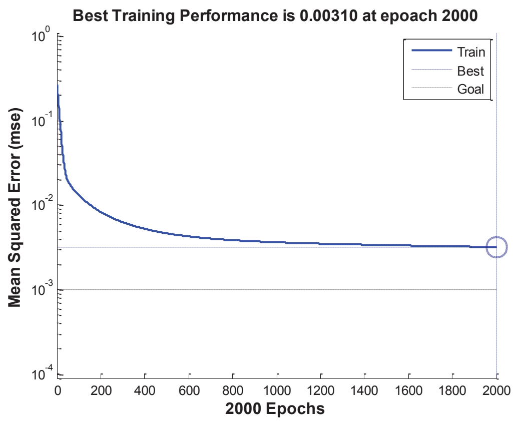
In traditional training, the training data always adopts the historical actual data of last n cycles. It is somehow conservative and the performance cannot reach optimization. Here we add the processed prediction results of base prediction algorithms such as EMA, AR and so on. For example, the traditional training data is:

$$\mathbf{p}(:, i) = [p_0(i); p_0(i+1); p_0(i+2); p_0(i+3)],$$

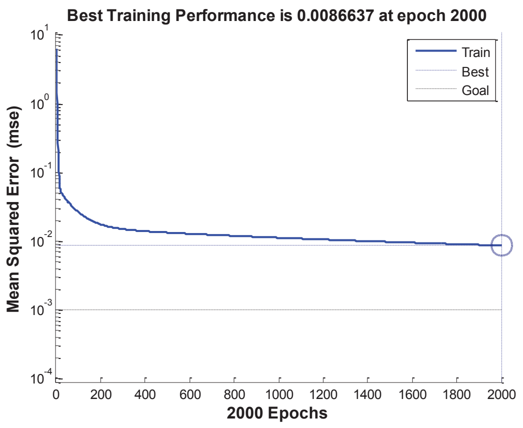
where $p_0\,(i)$ is the historical actual data. We add the prediction data to the training data vector:

$$\mathbf{p}(:, i) = [p_0(i); p_0(i+1); p_0(i+2); p_0(i+3); v(i)],$$

where v(i) is the weighted average of the prediction results of the base predictors. After the change of the training data structure, the training efficiency and performance are optimized greatly, as shown in Fig. 9. Compared with Fig. 8(b), only 1600 training cycles passed before reaching the goal, which is not reached in basic BP neural network or the self-adjusting learning rate with momentum BP neural network. The convergence speed is greatly improved after basic predictors ensemble is introduced.



(a) Training procedure of basic BP neural network.



(b) Training procedure of self-adjusting learning rate with momentum BP neural network.

Fig. 8 *The comparison of the training performance of different learning algorithms.*

To clearly understand the effect after adding the predictors' outputs, we compare the main statistic criteria of MAE, MSE, SSE and PRED(5), as shown in Fig. 10. From Fig. 10, we can see that after the base predictors' outputs are introduced into the training data, the performance is improved. Fig. 10(a) shows the performance of MAE. Although the absolute value only decreases 0.0031, the MAE has improved 13.9%. Fig. 10(b) and (c) represent the mean error energy and total error energy. The MSE is 0.001 versus 0.0012 and the max squared error is 0.011 versus 0.016. The SSE is 0.033 versus 0.042. The number that falls within ±5% is 320 versus 302, and the total number is both 350. From the data comparison, we can conclude that the introduction of the results of base predictors is helpful to improve the performance of prediction.

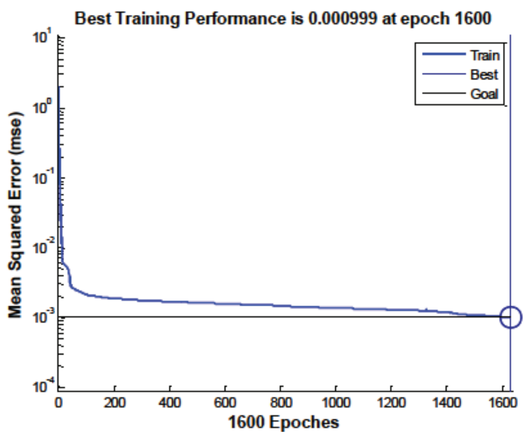


Fig. 9 *The prediction performance with prediction results of base predictors.*

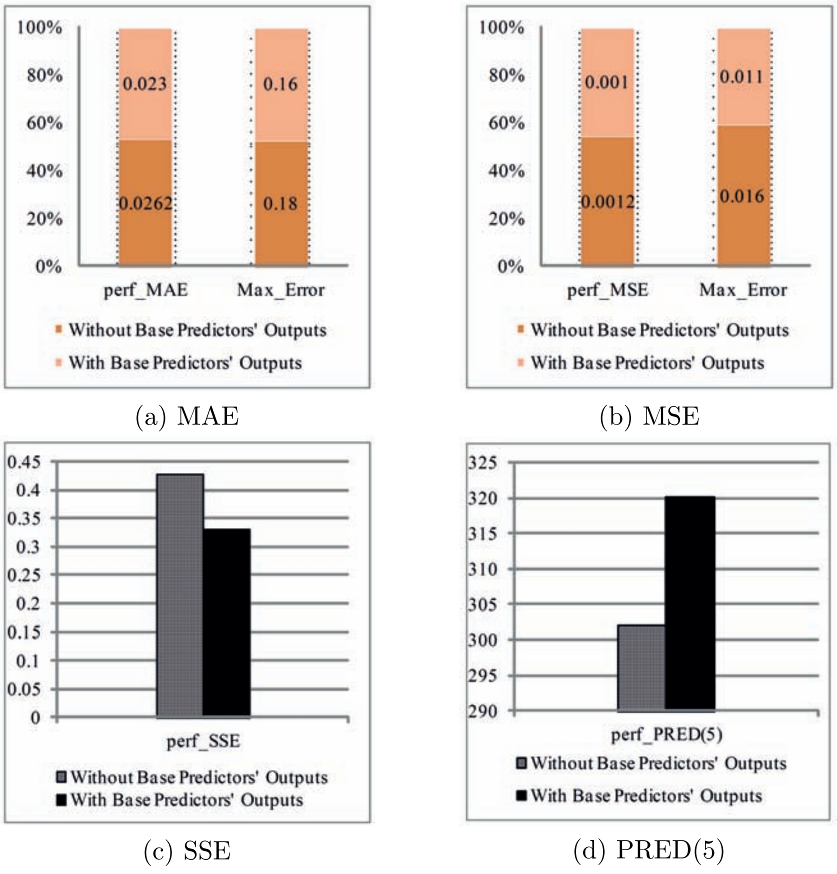


Fig. 10 *Prediction error indexes comparison of different methods.*

6. Conclusions

To solve the problem of resource waste or shortage in resource provision procedure effectively, a self-adaptive real-time resource demands prediction method called SRNN has been presented this paper. The structure of the prediction system is discussed. Users' preferences are analyzed firstly to reduce the amount of calculation. Then the base prediction models and methods are introduced into the system. The results are sent to the self-adjusting learning rate with momentum BP neural network as the inputs. With the results of base predictors, the BP neural network is able to improve the prediction performance. Statistic criteria including MAE, MSE, SSE, PRED(5), etc are adopted to evaluate the method. The results show that the proposed method can effectively improve the prediction accuracy.

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 using the real cloud workloads for traffic analysis in the real cloud computing system in future.

Acknowledgement

The authors would like to thank the anonymous reviewers for their invaluable feedback. This work was supported by the Weapon Equipment Pre-research Fund of China (Grant No. 9140A04030214JB34058).

References

[1] ANDREOLINI M., CASOLARI S. Load prediction models in Web-based systems. In: *Proceedings of the 1st International Conference on Performance Evaluation Methodologies and Tools*, Pisa, Italy. New York: ACM, 2006, p. 27.

[2] CAO J., FU J.W., LI M.L., CHEN J.J. CPU load prediction for cloud environment based on a dynamic ensemble model. *Software: Practice and Experience.* 2014, 44(7), pp. 783–804, doi: 10.1002/spe.2231.

[3] CHANG Y.C., CHANG R.S., CHUANG F.W. A Predictive Method for Workload Forecasting in the Cloud Environment. *Advanced Technologies, Embedded and Multimedia for Human-centric Computing.* 2014, 260, pp. 577–585, doi: 10.1007/978-94-007-7262-5_65.

[4] CHEN G., BAI X.Y., HUANG X.F., LI M.Y, ZHOU L.Z. Cloud Performance Trend Prediction by Moving Averages. *Journal of Frontiers of Computer Science and Technology.* 2012, 6(6), pp. 495–503, doi: 10.3778/j.issn.1673-9418.2010.06.002.

[5] FANG W., LU Z.H., WU J., CAO Z.Y. RPPS: A Novel Resource Prediction and Provisioning Scheme in Cloud Data Center. In: *Proceedings of 2012 IEEE Ninth International Conference on Services Computing*, Honolulu, America. New Jersey: IEEE, 2012, pp. 609–616.

[6] HAYKIN S. *Neural Network and Learning Machines.* Harlow: Pearson Education Inc, 2011.

[7] HU D.D., CHEN N.J., DONG S.L, WAN Y.M. A User Preference and Service Time Mix-aware Resource Provisioning Strategy for Multi-tier Cloud Services. In: *AASRI Conference on Parallel and Distributed Computing Systems*, Singapore. Amsterdam: Elsevier B.V., 2013, pp. 235–242.

[8] ISLAM S., KEUNG J., LEE K., LIU A. Empirical Prediction Models for Adaptive Resource Provisioning in the Cloud. *Future Generation Computer Systems.* 2012, 28(1), pp. 155–162, doi: 10.1016/j.future.2011.05.027.

[9] KONG X.Z., LIN C., JIANG Y.X., YAN W., CHU X.W. Efficient Dynamic Task Scheduling in Virtualized Data Centers with Fuzzy Prediction. *Journal of Network and Computer Applications.* 2011, 34(4), pp. 1068–1077, doi: 10.1016/j.jnca.2010.06.001.

[10] MARSTON S., LI Z., BANDYOPADHYAY S., ZHANG J.H. Cloud Computing-The Business Perspective. *Decision Support Systems.* 2011, 51(1), pp. 176–189, doi: 10.1016/j.dss.2010.12.006.

[11] PREVOST J.J., NAGOTHU K.M., KELLEY B., JAMSHIDI M. Prediction of Cloud Data Center Networks Loads Using Stochastic and Neural Models. In: *Proceedings of the 6th International Conference on System of Systems Engineering*, Albuquerque, America. New Jersey: IEEE, 2011, pp. 276–281.

[12] QIAN L., LUO Z.G., DU Y.J., GUO L.T. Cloud Computing: An Overview. *Lecture Notes in Computer Science.* 2009, 5931(1), pp. 626–631, doi: 10.1007/978-3-642-10665-1_63.

[13] RAMEZANI F., LU J., HUSSAIN F. An Online Fuzzy Decision Support System for Resource Management in Cloud Environments. In: *Proceedings of 2013 Joint IFSA World Congress and NAFIPS Annual Meeting*, Edmonton, Canada. New Jersey: IEEE, 2013, pp. 754–759.

[14] REIG G., ALONSO J., GUITART J. 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*, Cambridge, America. New Jersey: IEEE, 2010, pp. 162–167.

[15] SARIPALLI P., KIRAN G., SHANKAR R., NARWARE H, BINDAL N. Load Prediction and Hot Spot Detection Models for Autonomic Cloud Computing. In: *Proceedings of the 4th IEEE International Conference on Utility and Cloud Computing*, Victoria, Australia. New Jersey: IEEE, 2011, pp. 397–404.

[16] SHI Y.X., JIANG X.H., YE K.J. An Energy-Efficient Scheme for Cloud Resource Provisioning Based on CloudSim. In: *Proceedings of 2011 IEEE International Conference on Cluster Computing*, Texas, America. New Jersey: IEEE, 2011, pp. 595–599.

[17] WEN Y., MENG D., ZHAN J.F. Adaptive Virtualized Resource Management for Application's SLO Guarantees. *Journal of Software.* 2013, 24(2), pp. 358–377, doi: doi: 10.3724/SP.J.1001.2013.04216.

[18] VERMA M., GANGADHARAN G.R., RAVI V., NARENDRA N.C. Resource Demand Prediction in Multi-tenant Service Clouds. In: *Proceedings of 2013 International Conference on Cloud Computing in Engineering Markets*, Bangalore, India. New Jersey: IEEE, 2013, pp. 1–8.

[19] WU H.S., WANG C.J., XIE J.Y. 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*, Barcelona, Spain. New Jersey: IEEE, 2013, pp. 649–654.

[20] WU L.L., GARG S.K., BUYYA R. 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*, Newport Beach, Canada. New Jersey: IEEE, 2011, pp. 195–204.

[21] XIAO Z., SONG W.J., CHEN Q. Dynamic Resource Allocation using Virtual Machines for Cloud Computing Environment. *IEEE Transaction on Parallel and Distributed Systems.* 2013, 24(6), pp. 1107–1117, doi: 10.1109/tpds.2012.283.

[22] XU D.Y., YANG S.L., LUO H. A Fusion Model for CPU Load Prediction in Cloud Computing. *Journal of Networks.* 2013, 8(11), pp. 2506–2611, doi: 10.4304/jnw.8.11.2506-2511.

[23] YEH C.Y., KAO C.Y., HUNG W.S., LIN C.C, LIU P.F, WU J.J, LIU K.C. GPU Virtualization Support in Cloud System. *Grid and Pervasive Computing.* 2013, 7861, pp. 423–432, doi: 10.1007/978-3-642-38027-3_45.

[24] ZHANG H.L., LI P.P., ZHOU Z.G., DU X.J, ZHANG W.Z. A Performance Prediction Scheme for Computation-Intensive Applications on Cloud. In: *Proceedings of 2013 IEEE International Conference on Communications*, Budapest, Hungary. New Jersey: IEEE, 2013, pp. 1957–1961.

[25] ZHAO G.S., YU H., JI T.K., SONG H. Adaptive Resource Provisioning for Cloud Computing. *Telecommunication Science.* 2012, 28(1), pp. 31–37, doi: 10.3969/j.issn.1000-0801.2012.01.007.s.