Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 108C (2017) 365–374

Procedia Computer Science

International Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland

A Proactive Cloud Scaling Model Based on Fuzzy Time Series and SLA Awareness

Dang Tran[1], Nhuan Tran[1], Giang Nguyen[2], and Binh Minh Nguyen[1]*

[1] School of Information and Communication Technology,
Hanoi University of Science and Technology, Vietnam
dangtv18@gmail.com, tranducnhuan1994@gmail.com, minhnb@soict.hust.edu.vn
[2] Institute of Informatics, Slovak Academy of Sciences, Slovakia
giang.ui@savba.sk

Abstract
Cloud computing has emerged as an optimal option for almost all computational problems today. Using cloud services, customers and providers come to terms of usage conditions defined in Service Agreement Layer (SLA), which specifies acceptable Quality of Service (QoS) metric levels. From the view of cloud-based software developers, their application-level SLA must be mapped to provided virtual resource-level SLA. Hence, one of the important challenges in clouds today is to improve QoS of computing resources. In this paper, we focus on developing a comprehensive autoscaling solution for clouds based on forecasting resource consumption in advance and validating prediction-based scaling decisions. Our prediction model takes all advantages of fuzzy approach, genetic algorithm and neural network to process historical monitoring time series data. After that the scaling decisions are validated and adapted through evaluating SLA violations. Our solution is tested on real workload data generated from Google data center. The achieved results show significant efficiency and feasibility of our model.

Keywords: Fuzzy time series, neural network, genetic algorithm, adaptive resource management, decision model, SLA, autoscaling, cloud computing

1 Introduction

Thanks to virtualization, clouds can allocate computational resources quickly and dynamically. From the view of software-as-a-service (SaaS) developers, they must ensure the quality of service (QoS) for their end-users. However, the developers are depended on resource QoS provided by cloud infrastructure vendors. This QoS relation among cloud resource providers, software developers and end-users leads to appear Service Layer Agreement (SLA), which is a contract specifying the quality expectation of provided cloud services. Therefore, resource QoS guarantee and SLA violation prevention are crucial points for SaaS developers.

*Corresponding author

In fact, most of Infrastructure-as-a-Service (IaaS) clouds offer at least one resource monitoring solution for customers, who can rely on collected data and thresholds to decide amount of resources and scaling moments themselves. The main drawback of the approach is that cloud systems often response quite slowly in comparison with actual requirements from applications, especially with sudden demands. Besides, wasting and lacking resources problems will occur because it is difficult to determine exactly the scaling moments using the threshold technique. In recent years, extensive efforts have been conducted in the area of resource QoS improvement and SLA audit for cloud systems. In order to enhance QoS of resource provision, there are many studies that have dealt with building prediction models [17], [8] for application resource consumption. However, while the studies concern with the forecasting problem in the manner of improving accuracy among models together, they lack methods to evaluate the model effectiveness when making resource increase or decrease decisions based on achieved predictive results. In other words, they do not provide any solution to validate the prediction models in scaling process. In the aspect of SLA, many languages and frameworks [2] and [1] have been developed to keep close control of SLA violations. Unfortunately, these related SLA proposals only focus on auditing QoS based on traditional resource monitoring. In the scenario of applying prediction models to scale resources in cloud systems, there is still a need of having a solution to prevent SLA violations. In those directions, our work described in this paper has the following contributions: (1) Building a novel proactive autoscaling model for clouds includes two main components: prediction and scaling decision; (2) Proposing a novel prediction approach that exploits simultaneously multiple monitoring utilization data such as CPU, memory to forecast the future system usages; (3) Applying fuzzy time series approach in the prediction model to improve forecast effect; (4) Proposing a novel approach to make scaling decisions based on multiple parameters including predictions of multi-resource utilization and SLA violation estimation.

The rest of this paper is organized as follows. In section 2, we discuss some related studies to highlight the differences between our work and existing researches. In section 3, we present the model design of our proactive autoscaling system. Section 4 presents our experiments, gained results and observations with our proposals using real Google cluster data [13] to demonstrate the efficiency and feasibility of the model in practice. Finally, section 5 concludes and figures out some future directions.

2 Related work

Recently, problem of dynamic resource provisioning for clouds has been studied extensively. A lot of efforts deals with trade-off between minimizing resource consumption and meeting SLA. These provisioning techniques could be classified roughly into two categories: reactive and predictive provisioning.

The first category focuses on resource allocation methods [7], which react to immediate demands. In this way, Nguyen et al. [10, 11] propose a three-state model for server management that adjust resources by turning on or off servers according to job arrival and departure. In [4], Dutreilh and et al. apply threshold-based policies for autoscaling actions to adapt resources according to requirements. Thus, resources are allocated or deallocated to applications if performance metrics pass the upper or lower thresholds. Authors of [6] use a set of four thresholds and two duration of multiple performance metrics to trigger resource scaling actions. However, using these techniques, due to delay during adding or removing resources, scaling actions might not achieve the desired effect as compared with actual application requirements. Moreover, this also may lead to SLA violation phenomenon.

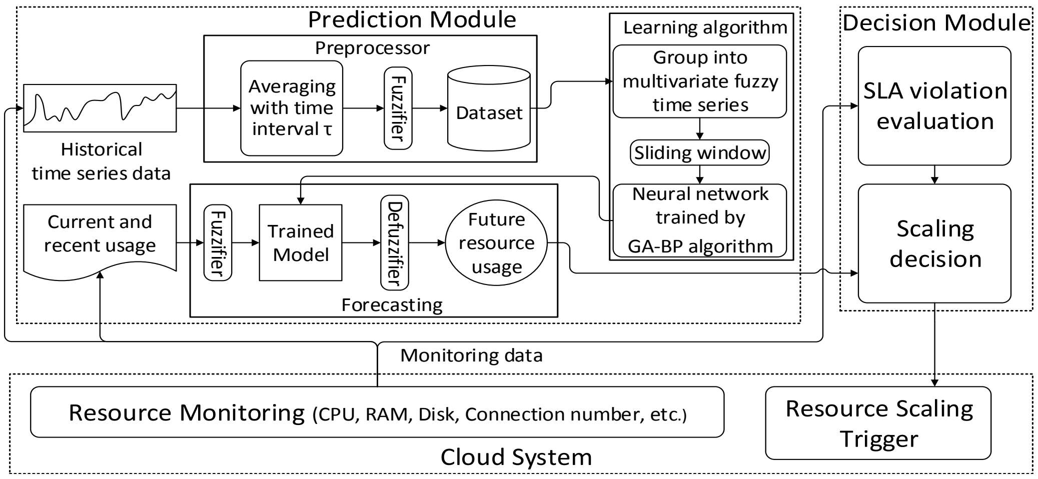


Figure 1: Proposed proactive scaling model for clouds

The goal of second category is to bring forward prediction models to scale in advance. Here, accuracy is crucial point for the approach. In [16], several forecasting methods including AR, MA, exponential smoothing, ETS, automated ARIMA and BPNN are employed to predict cloud workloads. Roy and et al. Authors of [8] use BPNN and multiple linear regression methods to predict the CPU utilization. Tran and et al. [15] put forward PD-GABP algorithm, which combines data periodic detection and neural network to improve the forecasting accuracy. Although a lot of prediction models are proposed but these works treat the prediction problem as univariate time series data and predict each metric separately. In this way, they may skip relationship among different metric types in the scaling problem. Hence it is difficult to apply research outcome as whole. Although fuzzy time series technique is referred in [5], where the authors map real data values into a corresponding fuzzy set. However, until now, fuzzy time series has not been applied to process monitoring resource usage in the cloud autoscaling problem.

In another direction, one of essential requirements when developing resource provisioning techniques is to guarantee QoS and prevent SLA violation. The authors of [14] use SLA to estimate necessary resource amount applying to the autoscaling problem. The authors demonstrate that their proposed algorithm can reduce the number of SLA violations up to 95%. In [12], a scheduler and prediction architecture built based on machine learning techniques are designed to forecast required resources in order to meet SLA. In comparison with our work, operations of these existing models only rely on a simple assumption, which SLA evaluation uses a single performance metric such as job execution deadline, response time, CPU usage, and so on.

3 Designing Model

The overall model of the proactive scaling system for clouds is shown in Figure 1 that consists of two modules including *Prediction* and *Decision*. The *Prediction* module is composed from three sub-components: data pre-processor, learning algorithm, and forecasting. The role of the first component is to transform all performance metrics into fuzzy time series, which are input for learning algorithm. For this phase, we propose a novel learning approach based on high order fuzzy time series and neural network with GA-BP method [3], called Multivariate Fuzzy GABP Neural Network (MF-GABPNN). When the training process finishes, the trained model is employed by the forecasting component to predict future resource usages, which are used to make decisions of allocated resources in the *Decision* module. The scaling decisions are made not only by predicted values but also by recent SLA violations, which are estimated by the feedback of monitoring data from cloud systems.

3.1 Preprocessor

Preprocessor's goal is to construct a suitable dataset for the learning part. The values of each metric monitored in cloud system will be collected and transformed into the corresponding time series $r_i(t)$ $(i = 1, 2, \dots, M)$ with time interval τ. Each point in time series $r_i(t)$ is calculated by averaging all the values of a resource usage in the period of τ: $r_i(n) = \frac{\sum_{(n-1)\tau<t<n\tau} D_i(t)}{n_\tau}$, where $D_i(t)$ is the value of type-i resource usage at time t that is monitored from cloud system, n_τ is the number of observation $D_i(t)$ in the interval τ. The next phase after averaging is fuzzification, which transforms a time series into a corresponding fuzzy time series. Definitions of fuzzy time series are presented below.

Definition 1. The definition of fuzzy time series is based on time series [5]. Let U is the universe of discourse, a fuzzy set A_i of U is defined as $A_i = f_{A_i}(u_1)/u_1 + f_{A_i}(u_2)/u_2 + \cdots + f_{A_i}(u_b)/u_b$, where u_a is an element of fuzzy set A_i, f_{A_i} is the membership function of the fuzzy set A_i: $f_{A_i} : U \to [0, 1]$. Let y(t) is a time series, the fuzzy time series defined on y(t) is a series of $F(t)$, where $F(t)$ is a collection of $f_{A_1}(y(t)), f_{A_2}(y(t)), \dots$

The fuzzifier defines all fuzzy sets for observations and then maps each real value into a certain fuzzy set, where its membership degree has a maximum value. Detail of fuzzification is presented by step 1, 2 and 3 described in [5] including: defining universe of discourses and intervals, defining fuzzy sets, and mapping each crisp value into maximum-membership fuzzy set. We use these steps for our fuzzification phase by repeating them on each time series data. At the end of preprocessor phase, a dataset that contains all the fuzzy time series will be stored in a database management system.

3.2 Learning algorithm

The learning algorithm consists of three stages: grouping into multivariate fuzzy time series, sliding window, and training of neural network. The concept of multivariate fuzzy time series is described in Definition 2. Multivariate fuzzy logic relationship is presented in Definition 3.

Definition 2. Let $F_1(t)$, $F_2(t)$,..., $F_i(t)$,..., $F_M(t)$ are M fuzzy time series, the multivariate fuzzy time series is: $F(t) = \left[F_1(t), F_2(t), \dots, F_M(t)\right], \quad t = 1, 2, \dots, n$

Definition 3. Let $F_1, F_2, \dots, F_M$ be M fuzzy time series. If $(F_1(t+k), F_2(t+k), \dots, F_M(t+k))$ is caused by $(F_1(t), F_2(t), \dots, F_M(t)), \dots, (F_1(t-k), F_2(t-k), \dots, F_M(t-k))$ then this multivariate fuzzy logical relationship is represented by:

$(F_1(t), F_2(t), \dots, F_M(t)), \dots, (F_1(t-p+1), F_2(t-p+1), \dots, F_M(t-p+1)) \to (F_1(t+k), F_2(t+k), \dots, F_M(t+k))$

The future values of fuzzy time series can be predicted by discovering the hidden fuzzy logic relationship in historical fuzzy time series. For this task, we employ a fully-connected three-layer neural network with the training method combining genetic and back-propagation algorithm [3], called GA-BPNN. The data from Pre-processor component are grouped into multivariate fuzzy time series, which are used for the training phase of neural network. During the training process, the *sliding window technique* [8] is exploited to construct input/output pairs and feed them to the network. In each point of time t, the inputs are $\{F(t), F(t-1), \dots, F(t-p+1)\}$, and the output is k-step ahead forecast $F(t+k)$. A common condition in time series forecasting is that p is not less than k.

Algorithm 1 MF-GABPNN

```
1: Normalize all historical time series of M type of resource consumption:
   $r_1(t), r_2(t), \dots, r_i(t), \dots, r_M(t)$
2: Fuzzify M time series into M corresponding fuzzy time series:
   $F_1(t), F_2(t), \dots, F_i(t), \dots, F_M(t)$
3: Group all fuzzy time series into a unique multivariate one $F(t)$:
   $[F_1(1), F_2(1), \dots, F_M(1)], [F_1(2), F_2(2), \dots, F_M(2)], \dots, [F_1(n), F_2(n), \dots, F_M(n)]$
4: Create a three-layer neural network, initialize sliding window with $p$ consecutive points as
   the inputs $(F(t), F(t-1), F(t-2), \dots, F(t-p+1)$ and the next observation $F(t+k)$ as
   the outputs $(t = 1, 2, \dots, n)$.
5: Training neural network by GA-BP algorithm. The trained network will represent fuzzy
   logical relationship: $(F(t), F(t-1), F(t-2), \dots, F(t-p+1) \to F(t+k)$
6: Pass the trained model to Forecasting component
7: Repeat step 1 to step 5 every time period $T$
```

Algorithm 2 SLA-based proactive scaling algorithm

```
1: At each point of time $t$, collect the new resource consumption data $r_1(t), r_2(t), \dots, r_i(t)$,
   fuzzify them into $F_1(t), F_2(t), \dots, F_i(t), \dots, F_M(t)$. Calculate $F_1(t+k), F_2(t+k), \dots, F_i(t+k), \dots, F_M(t+k)$ by the trained neural network.
2: For each fuzzy forecast $F_1(t+k), F_2(t+k), \dots, F_i(t+k), \dots, F_M(t+k)$, apply centroid
   defuzzification method to get the final crisp result of forecasting, which is resource con-
   sumption at the next point of time: $r_i^{pred}(t+1), i = 1, 2, \dots, M$
3: for $z$ in $(t-p+1, t-p+2, \dots, t)$ do evaluate $r_{violation}(z)$ by equation (1)
4: for $i$ in $(1, 2, \dots, M)$ do calculate $r_i^{decis}(t+1)$ by equation (2)
5: Calculate number of allocated VMs $(n_{VM}^{alloc}(t+1))$ by equation (3)
6: if $n_{VM}^{alloc}(t+1) > n_{VM}^{alloc}(t)$ then Make decision to instantiate $\left(n_{VM}^{alloc}(t+1) - n_{VM}^{alloc}(t)\right)$ VMs
7: else Make decision to destroy $\left(n_{VM}^{alloc}(t) - n_{VM}^{alloc}(t+1)\right)$ VMs
8: Repeat from step 1 to 13 at the next point of time
```

3.3 Forecasting

The forecasting component uses current and recent resource usage values as the inputs of the trained model from learning algorithm to predict new values in advance. The inputs must be fuzzified before calculating outputs of the trained network. Output values will be defuzzified into the real values, which are final step of forecast process. In contrast to fuzzification, defuzzification is the process of producing real values from given fuzzy set. For defuzzificating, centroid method (also called as center of area or center of gravity) is adopted. The defuzzification stage for fuzzy time series forecasting is presented in step 6 at [5] that is applied to our design. The operations of Prediction module are described by Algorithm 1 and the first two steps of Algorithm 2, where step 1, 2 and 3 of Algorithm 1 are intended for preprocessor subcomponent, step 4 and 5 of Algorithm 1 are learning algorithm, step 1 and 2 of Algorithm 2 are forecasting. In every time period T, learning algorithm is repeated to update the neural network with new monitoring data.

3.4 SLA violation evaluation

Because error rate of scaling decisions causes the increase of SLA violation, therefore a SLA violation estimation mechanism is used as a surveying ruler to adapt the decisions in our proposal. We assume that SLA requires the cloud system not to allocate less amount of allocated resources than cloud-based application's demands. The SLA violation is evaluated based on calculating the differences between actual resource usage requirement and the amount of allocated resource by the following formula:

$$r_{violation}(t) = max\{0, r_i^{actual}(t) - r_i^{alloc}(t)\} \tag{1}$$

where $r_i^{alloc}(t)$ is the amount of type-i resource allocated to application at the point of time t and $r_i^{actual}(t)$ is the actual resource usage required by the application. If the allocation amount is greater than actual utilization, there is no SLA violation and $r_{violation}(t) = 0$. By contrast, if $r_i^{alloc}(t) < r_i^{actual}(t)$, the more required usage is, the more $r_{violation}(t)$ is.

3.5 Scaling decision

The Scaling decision module uses the predicted values and SLA violation estimations to make final resource allocation decisions. To perform scaling tasks in this module, formula (3) is proposed to calculate the number of VMs that need to be provided to cloud-based application. The resource amount at the next point of time $(t+1)$ (assuming that t is current point of time) is estimated for each data type: CPU, memory usage, disk I/O, and so forth as follow:

$$r_i^{decis}(t+1) = s * r_i^{pred}(t+1) + \frac{1}{l}\sum_{z=t-l+1}^{t} r_{violation}(z) \tag{2}$$

where $r_i^{decis}(t+1)$ is the amount of type-i resource that is decided to allocate to application, $r_i^{pred}(t+1)$ is resource usages that are predicted in advance, s is the scaling coefficient and l is adaptation length. $r_{violation}(z)$ is defined in the formula 1.

It can be seen that the right-hand side of equation 2 consists of two terms. In the first term, the prediction of resource usage is multiplied by s $(s \geq 1)$ in order to ensure the system always allocates a larger resource amount than demand. This helps reduce the impact of forecasting error on on scaling decision accuracy (in other word, this prevents QoS decrease). The bigger value of s leads to the more resources allocated for the application and decreases SLA violation rate. The second term is the total estimation of SLA violation in recent period with length l. Therefore, if the rate of SLA violation increases (or decreases), the resource amount decided for allocation will increase (or decrease). In this way, the scaling decision is adapted by SLA violation.

After the processes above, the final number of VM is determined to allocate for appliance. We also assume that cloud systems provide a lot of homogeneous VMs with the same capacity and total capacity of all VMs can be utilized by applications. In addition, we do not consider scheduling policies among VMs in our scaling strategy. In this way, the number of allocated VMs is determined by all the metrics of resource consumption as follows:

$$n_{VM}^{alloc}(t+1) = max\left\{\frac{r_1^{decis}(t+1)}{C_1}, \frac{r_2^{decis}(t+1)}{C_2}, \ldots, \frac{r_i^{decis}(t+1)}{C_i}, \ldots, \frac{r_M^{decis}(t+1)}{C_M}\right\} \tag{3}$$

where C_i is the VM capacity of type-i resource. For example, if a VM has 4 CPU cores, 4GB of memory, 1024 Mbps of bandwidth, its capacity will be represented by $C_1 = 4$ (CPU), $C_2 = 4$ (RAM) and $C_3 = 1024$ (bandwidth). Our autoscaling strategy is expressed by Algorithm 2.

4 Experiments

4.1 Experimental Setup

To carry out experiments, we used Google cluster trace dataset [13]. In the dataset, each job is a collection of many tasks, which are run simultaneously on multiple machines. Resource utilization of tasks are measured by several metrics such as CPU usage, memory usage, disk I/O mean time, and so on, which are recorded in about 1233 million data items. According to the analyses described in [13], only less than 2% of jobs run for longer than one day, even though such jobs contribute to over 80% of the recorded resource utilization in the cluster. In addition, just jobs containing at least hundreds of tasks consume a significant resource amount. To achieve the generality in evaluating the efficiency of our model against the Google data, we select a job in the set of long-running jobs. The job consists of 60171 divergent tasks during the 20-day period (from 1th day to 20th day).

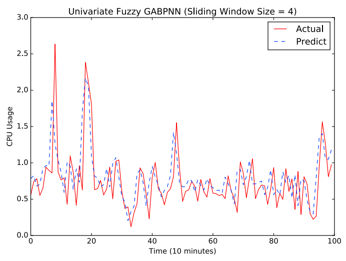
We simulate a cloud system with a lot of homogeneous VMs performing that job. The capacity of a VM equals the smallest capacity of the machines in Google cluster that is $C_{CPU} = 0.245$ of CPU and $C_{RAM} = 0.030$ of memory (normalized values). Both CPU and memory utilization measurements are used for scaling decisions. We also assume that required time to instantiate a new VM is 10 minutes. Therefore, the resource usage in the incoming 10 minutes is predicted to make scaling decisions in advanced. We set $\tau = 10$ minutes, forecast horizon $k = 1$ and adaptation length $l = 10$ in all experiments. The dataset from 1st to 15th day is used to train neural network, and the Proactive Scaling Decision workflow (Algorithm 2) starts from 16th day. The prediction accuracy is assessed by mean absolute error (MAE) defined as follows [15]: $MAE = \frac{1}{n}\sum_{i=1}^{n} |\widehat{y_i} - y_i|$, where $\widehat{y_i}$ is predicted value and y_i is real measured value. The fuzzy sets (described in Definition 1) for CPU and memory utilization measurement are selected by following previous work [5]: $A_i = \frac{0}{u_1} + \frac{0}{u_2} + \ldots + \frac{0.5}{u_{i-1}} + \frac{1.0}{u_i} + \frac{0.5}{u_{i+1}} + \ldots + \frac{0}{u_n}$. For CPU: $u_1 = [0; 245], u_2 = [0.245; 0.49], \ldots$ For memory: $u_1 = [0; 03], u_2 = [0.03; 0.06], \ldots$

4.2 Multivariate Fuzzy GABPNN (MF-GABPNN)

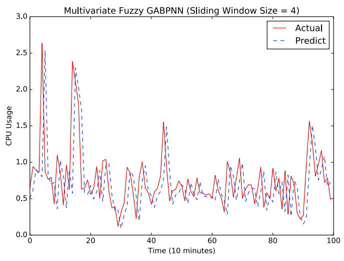
In this test, we aim at evaluating the efficiency of MF-GABPNN in comparison with other forecasting methods: back-propagation neural network (BPNN), genetic algorithm BPNN [3], Fuzzy BPNN, and Fuzzy GABPNN, in predicting CPU and memory usages using Google cluster data. In this direction, we also perform the tests in the manner of comparing univariate (predicting separately each metric and combining achieved outcomes at the end of forecast pro-

Methods		CPU				Memory			
		$p = 2$	$p = 3$	$p = 4$	$p = 5$	$p = 2$	$p = 3$	$p = 4$	$p = 5$
Univariate	BPNN	1.0519	1.0523	0.8647	0.8805	0.1232	0.1426	0.1028	0.0864
	GABPNN	1.0528	1.0487	0.8731	1.0519	0.1173	0.0991	0.1387	0.1222
	Fuzzy BPNN	1.0527	0.8679	0.8914	1.0528	0.1034	0.1416	0.0874	0.1113
	Fuzzy GABPNN	1.0597	0.8701	1.0519	1.0576	0.0991	0.1387	0.1232	0.1034
Multivariate	BPNN	0.8848	0.9380	0.8973	0.9127	0.0639	0.0587	0.0534	0.0726
	GABPNN	0.9465	0.8875	0.7959	0.7867	0.0723	0.0602	0.0524	0.0561
	Fuzzy BPNN	1.0328	0.9157	0.8909	0.8842	0.0731	0.0547	0.0589	0.0615
	MF-GABPNN	0.8806	**0.7847**	0.8364	0.8297	0.0704	0.0528	**0.0523**	0.0593

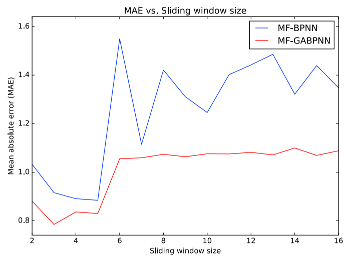
Table 1: MAE of CPU and memory prediction accuracy against window sizes (p)



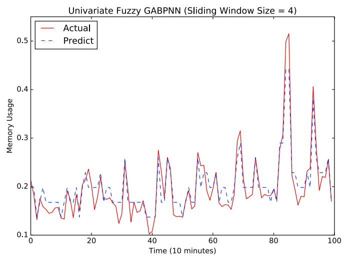
(a) Univariate Fuzzy GABPNN



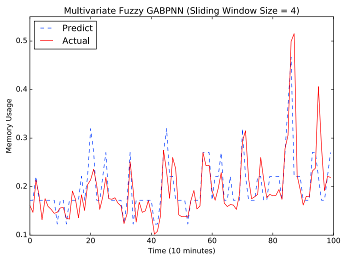
(b) MF-GABPNN



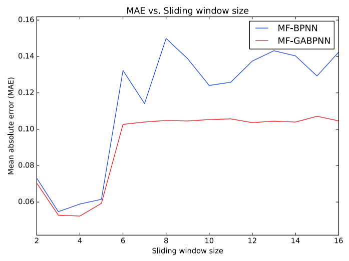
(c) MAE



(d) Univariate Fuzzy GABPNN



(e) MF-GABPNN



(f) MAE

Figure 2: CPU forecasting (2a, 2b, 2c) and Memory Forecasting (2d, 2e, 2f)

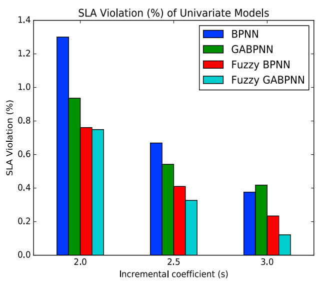


Figure 3: SLA violation of univariate approach

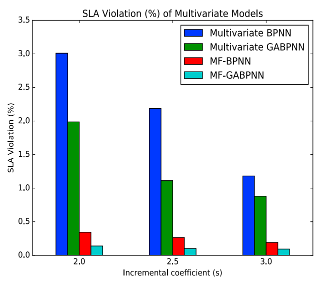


Figure 4: SLA violation of multivariate approach

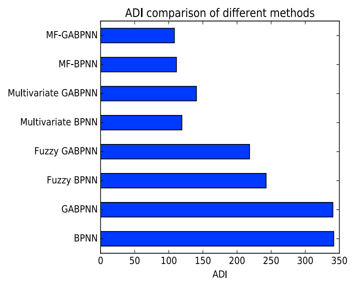


Figure 5: ADI evaluation (smaller is better)

cess) and multivariate time series data (predicting simultaneously metrics). The gained results are represented in Table 1. It is remarkable that the multivariate model produces better results as compared with univariate approach. Specifically, the forecast of univariate approach with $p = 4$ depicted in figure 2a and 2d, overestimation can be recognized easily at many diverse points. However, the best MAE of CPU and memory predicted by MF-GABPNN are equal 0.7847 and 0.0523 respectively, which are 50% and 11% smaller than CPU and memory forecast usages respectively as compared with the univariate experiment outcomes. In figure 2b and 2e, it is obvious that prediction lines of MF-GABPNN are quite identical with actual lines.

On the other hand, figure 2c and figure 2f compare the responses of the prediction accuracy with different sliding window sizes based on MAE metric. It can be seen that achieve MAE values of MF-GABPNN are lower than multivariate fuzzy BPNN (MF-BPNN). When sliding window size increases from 6 to 16, the MAE of MF-GABPNN is completely smaller than MF-BPNN, which demonstrates the efficiency of our model in improving prediction accuracy.

4.3 Scaling Decision

In the second test, we verify scaling decisions using prediction results achieved from the forecasting module. To do that we evaluate the SLA violations which occurs when making decisions. As presented in previous section, we assume that SLA will be violated if the amount of any resource type allocated to application is less than demands. We propose a metric for measuring the level of SLA violations called SLA Violation Time Percentage (SLAVTP), which is defined as follow: $SLAVTP = T_{under-provisioning}/T_{execution}$, where $T_{under-provisioning}$ is the total time, in which at least one type of resource allocated to application causes a under-provisioning, $T_{execution}$ is total time of the application running in cloud system. Figure 3 and 4 show the estimation of SLAVTP in resource allocation process with window size $p = 4$. In general, our MF-GABPNN evaluation outcomes are nearly 60% smaller than the other methods. Concretely, it reaches smallest SLAVTP at 0.093.

We also consider resource over-provisioning issue, which will reduce SLA violations but instead, it causes resource waste. In every time t, almost all cloud users would like to keep a reasonable utilization level. This level is defined by percentage of resource use in total resource allocation. In this way, the percentage must belong to a certain range denoted by L and U that are lower and upper bound respectively, with $0 \leq L \leq U \leq 1.0$. In [9], Autoscaling Demand Index (ADI) method is represented by σ variable, which is defined as follows: $\sigma = \sum_{t \in T} \sigma_t$, where $\sigma_t = L - u_t$ if $u_t \leq L$; $\sigma_t = 0$ if $L < u_t < U$; $\sigma_t = u_t - U$ if $u_t \geq U$.

ADI is the sum of distances between u_t and $[L, U]$. For each time t, according to ADI metric, optimal autoscaling strategy will delivers minimum σ. Figure 5 shows the ADI metric measurement of different prediction models, with the desired utilisation level [50%,80%]. It is easily to see that ADI of our MF-GABPNN is smaller than the others. Specifically, the obtained outcome is less near 42% than GABPNN and the approach reaches optimal value at 108.26. The results demonstrate the significant efficiency of our solution.

5 Conclusion

This work concentrates on designing a proactive autoscaling model for cloud computing. While most of all related studies intends to develop prediction methods with high accuracy, our proposed model offers a comprehensive solution, which consists of two components, namely prediction and scaling decision. For the prediction module, we take advantages of fuzzy approach to process multivariate monitoring resource, genetic algorithm, back-propagation, and neural network to forecast precisely and efficiently in comparison with other prediction methods. For the decision module, we propose formula to calculate SLA violations, then the SLA-aware data is sent back to system in order to integrate with predicted values to adapt our autoscaling model. The solution is tested on real cluster usage data published by Google. The achieved results prove the feasibility and efficiency of the approach.

Acknowledgments

This research is supported by Vietnamese MOET's project "Research on developing software framework to integrate IoT gateways for fog computing deployed on multi-cloud environment" No. B2017-BKA-32, Slovak VEGA 2/0167/16 project "Methods and algorithms for the semantic processing of Big Data in distributed computing environment", and EU Research and Innovation programme Horizon 2020 H2020-654142 EGI-Engage project "Engaging the Research Community towards an Open Science Commons".

References

[1] M. Alhamad, T. Dillon, and E. Chang. Conceptual sla framework for cloud computing. In *4th IEEE Int. Conf. on Digital Ecosystems and Technologies*, pages 606–610, April 2010.

[2] A. Andrzejak, D. Kondo, and S. Yi. Decision model for cloud computing under sla constraints. In *2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pages 257–266, Aug 2010.

[3] Shifei Ding, Chunyang Su, and Junzhao Yu. An optimizing bp neural network algorithm based on genetic algorithm. *Artificial Intelligence Review*, 36(2):153–162, 2011.

[4] X. Dutreilh, A. Moreau, J. Malenfant, N. Rivierre, and I. Truck. From data center resource allocation to control theory and back. In *2010 IEEE 3rd International Conference on Cloud Computing*, pages 410–417, July 2010.

[5] Erol Egrioglu, Cagdas Hakan Aladag, Ufuk Yolcu, Vedide R. Uslu, and Murat A. Basaran. A new approach based on artificial neural networks for high order multivariate fuzzy time series. *Expert Systems with Applications*, 36(7):10589 – 10594, 2009.

[6] M. Z. Hasan, E. Magana, A. Clemm, L. Tucker, and S. L. D. Gudreddi. Integrated and autonomic cloud resource scaling. In *2012 IEEE Network Operations and Management Symposium*, pages 1327–1334, April 2012.

[7] Ladislav Hluchý, Giang Nguyen, Ján Astaloš, Viet Tran, Viera Šipková, and Binh Minh Nguyen. Effective computation resilience in high performance and distributed environments. *COMPUTING AND INFORMATICS*, 35(6):1386–1415, 2017.

[8] Sadeka Islam, Jacky Keung, Kevin Lee, and Anna Liu. Empirical prediction models for adaptive resource provisioning in the cloud. *Future Generation Computer Systems*, 28(1):155 – 162, 2012.

[9] M. A. S. Netto, C. Cardonha, R. L. F. Cunha, and M. D. Assuncao. Evaluating auto-scaling strategies for cloud computing environments. In *2014 IEEE 22nd Int. Symposium on Modelling, Analysis Simulation of Computer and Telecommunication Systems*, pages 187–196, Sept 2014.

[10] B. M. Nguyen, D. Tran, and Q. Nguyen. A strategy for server management to improve cloud service qos. In *2015 IEEE/ACM 19th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, pages 120–127, Oct 2015.

[11] Binh Minh Nguyen, Dang Tran, and Giang Nguyen. Enhancing service capability with multiple finite capacity server queues in cloud data centers. *Cluster Computing - The Journal of Networks, Software Tools and Applications*, 19(4):1747–1767, 2016.

[12] G. Reig, J. Alonso, and J. Guitart. Prediction of job resource requirements for deadline schedulers to manage high-level slas on the cloud. In *Network Computing and Applications (NCA), 2010 9th IEEE International Symposium on*, pages 162–167, July 2010.

[13] Charles Reiss, Alexey Tumanov, Ganger, and et al. Heterogeneity and dynamicity of clouds at scale: Google trace analysis. In *Proceedings of the Third ACM Symposium on Cloud Computing*, SoCC '12, pages 7:1–7:13, 2012.

[14] A. A. D. P. Souza and M. A. S. Netto. Using application data for sla-aware auto-scaling in cloud environments. In *Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), 2015 IEEE 23rd International Symposium on*, pages 252–255, Oct 2015.

[15] D. Tran, N. Tran, B. M. Nguyen, and H. Le. PD-GABP - a novel prediction model applying for elastic applications in distributed environment. In *2016 IEEE 3rd National Foundation for Science and Technology Development Conference on Information and Computer Science (NICS)*, pages 240–245, 2016.

[16] Carlos Vazquez, Ram Krishnan, and Eugene John. Time series forecasting of cloud data center workloads for dynamic resource provisioning. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, 6(3):87–110, 2015.

[17] Z. Xiao and et al. Dynamic resource allocation using virtual machines for cloud computing environment. *IEEE Transactions on Parallel and Distributed Systems*, 24(6):1107–1117, June 2013.