2018 IEEE 11th International Conference on Service-Oriented Computing and Applications

A Resource Usage Prediction System Using Functional-link and Genetic Algorithm Neural Network for Multivariate Cloud Metrics

Thieu Nguyen, Nhuan Tran, Binh Minh Nguyen*
School of Information and Communication Technology
Hanoi University of Science and Technology
Hanoi, Vietnam
nguyenthieu2102@gmail.com, tranducnhuan1994@gmail.com,
minhnb@soict.hust.edu.vn*

Giang Nguyen
Institute of Informatics
Slovak Academy of Sciences
Bratislava, Slovakia
giang.ui@savba.sk

***Abstract*—Designing prediction-based auto-scaling systems for cloud computing is an attractive topic for scientists today. However, there are many barriers, which must be solved before applying these systems to practice. Some challenges include: improving accuracy for prediction models, finding a simple and effective forecast method instead of complex techniques, and processing multivariate resource metrics at the same time. So far, there are no existing proactive auto-scaling solutions for clouds that have addressed all those challenges. In this paper, we present a novel cloud resource usage prediction system using functional-link neural network (FLNN). We propose an improvement for the FLNN by exploiting genetic algorithm (GA) to train learning model in order to increase forecast effectiveness. To deal with multivariate input data, several mechanisms also are combined together to enable the ability of processing simultaneously different resource types in our system. This enables to discover implicit relationship among diverse metrics and based on that realistic scaling decisions can be made closer to reality. We use Google trace dataset to evaluate the proposed prediction system and data preprocessing mechanisms introduced in this work. The gained outcomes demonstrated that our system can work effectively under practical situations with good performance as compared with traditional techniques.**

***Index Terms*—Cloud computing, Functional-link neural network, Functional-link and Genetic algorithm neural network, FL-GANN, Proactive auto-scaling, Multivariate time series data, Genetic algorithm, Google trace dataset.**

I. Introduction

Pay-as-you-go fashion is one of the prominent advantage features of cloud computing. With the usage model, users always expect their applications, which operate on cloud infrastructures are provided computational resources adequately but not redundantly in real-time without any delays [1]. This helps reduce maximally the renting resource costs. However, it is quite difficult and complicated to meet the requirement because clouds must have a new scaling mechanism that can be tuned immediately and precisely offered resources on demand of these appliances to replace the current approach with resource usage thresholds.

*Corresponding author.

Indeed, although the setting threshold mechanism enables to define scaling rules easily, it cannot supply accurately resources as compared with the changeful requirements of applications [2]. For example, assuming CPU percentage threshold of virtual machine (VM) is set at 80% to scale out. When CPU utilization reaches the threshold, cloud will automatically add one more VM for the users system. However, the scaling function needs to spend a few moments to complete the deployment of second VM. During this time, the CPU percentage usage has already been changed from the set threshold to another that can be bigger or smaller than 80%. This makes the inaccuracy issue for current cloud auto-scalers as presented above.

Besides, due to quality of service (QoS) tie with users (which is defined in service-layer agreements - SLA), cloud vendors often accept the resource over-provisioning (despite sometime the cloud users cannot acquire enough resources like) rather than offering resource accurately in real-time [3]. This leads to the increase of infrastructure costs in general for both cloud users and providers. It can be seen that there is a long distance among the resource supply requirements of users and the vendors' capabilities with the cloud auto-scaling solution today. Hence, proposing a better auto-scaler to resolve the problem has significant for cloud computing adoption at this time.

Recently, many prediction-based auto-scaling systems have been proposed for cloud computing. While increasing prediction accuracy for resource consumptions is a very attractive challenge, the problem of processing multiple metric types (e.g. CPU, and memory usages, disk capacity and I/O, network throughput/throughout and so forth) at the same time have received less attention from researchers when they develop prediction systems. Among the resource metrics could have some implicit relationships (e.g. between CPU and memory usage, and between disk I/O and memory I/O) and forecasting separately each resource type cannot discover the relationships. In this way, prediction outcomes will not fit practice requirements. To resolve the issue, cloud prediction-based

978-1-5386-9133-5/18/$31.00 ©2018 IEEE
DOI 10.1109/SOCA.2018.00014

auto-scalers has to the ability of processing different metrics simultaneously to make the scaling decision precisely. Until now, there are very few studies dealt with that multivariate problem for clouds. In this paper, we focus on proposing a new method for prediction stage using functional-link neural network (FLNN) and genetic algorithm (GA). Our forecast system also is designed to process multiple resource metrics simultaneously. We use real dataset produced from Google cluster to test and evaluate our designed model. The gained results show that our proposals bring positive effectiveness in predicting resource consumption.

The organization of this paper is as follows. In Section II, we categorize and analyze existing works to highlight our contributions. Section III presents our design cloud resource consumption prediction system with FN-GANN - an improvement of traditional FLNN using GA to train model. We also introduce several proposed data preprocessing mechanisms in this section. In the next Section, we describe our tests, evaluations for the proposed system to prove the proposal effectiveness. Section V concludes this paper and defines some our future works.

II. Related Work

The main goal of proactive resource provision techniques is to predict precisely the resource consumptions in advance. There are a huge research number that have dealt in forecast models for cloud computing. In [5], several prediction methods including autoregressive (AR), moving average (MA), autoregressive-moving average (ARMA), nonstationary, long memory, three families of seasonal, multiple input-output, intervention and multivariate ARMA models were evaluated and compared in predicting cloud workloads.

Recently, deep learning has emerged as an effective solution for prediction problem. In terms of applying to clouds, Prevost et al. [17] used the Multi-layer neural network (MLNN) model for prediction URL resource requests of a WWW server at NASA and WWW server at EPA. Although deep learning brings significant effectiveness for the cloud proactive auto-scaling issue, these learning models often have very complex structure and require a long time for training process. In this way, few other artificial networks have been proposed to overcome the disadvantages. Functional Link Neural Network is a representation for that simple neural network group.

FLNN has a single neuron is proposed by Pao [6] in pattern-recognition task. The author pointed out that FLNN has less computational cost and can be easily implemented in hardware applications. The reason is that hidden layer in this network is eliminated. In this way, nonlinear relationship between the inputs and outputs is processed via a set of functional expansions (i.e. polynomials). FLNN has been used in several applications such as stock market [7], and exchange rate prediction [8]. In their work, Khandelwal et al. [9] used 4 datasets from 4 different areas to test the forecasting ability of FLNN and in comparison with MLNN, the authors' obtained results indicate that FLNN model provides better outcomes for all 4 datasets. In [10], Sahoo et al used efficient Chebyshev and Legendre polynomials in combination as Functional-Link, the FLNN's outcomes gives better accuracy and takes less computation time as compared to MLNN's outcomes.

However, the proposed prediction systems using FLNN listed above which still use back-propagation technique with gradients descent to train learning model. S. Dehuri at el. in [11] figured out that the main drawbacks of back-propagation are slow training speed and local minimum traps. In comparison with those existing works, the main difference given in this study is that we exploit a herd evolution algorithm to solve back propagation disadvantages in traditional FLNN.

Genetic algorithm is one of optimization mechanisms for large and complex spaces in order to find values, which are close to the global optimum. Hence, GA suits the problem of training feed-forward networks [12]. In this direction, applying GA to diverse neural network variants have been proposed long time ago. In [15], Blanco et al. used GA-based approach to selected the features and optimized the appropriate classifier parameters of neural network and vector machines for bearing fault detection from time-domain vibration signals. In their work [4], Dehuri et al. employed GA to choose an optimal subset of input features in FLNN and radial basis function neural network (RBFNN). Until now, there are no works, which use GA to train FLNN instead of back-propagation algorithm.

In the aspect of dealing with multivariate time series problem in cloud computing, in [13], the authors proposed MF-GABPNN model to mine the relationship among different metric types in forecast process. Thus, the authors choose CPU and memory as multivariate time-series to test their proposed model. Going further, in [14], the authors proved that there is a relation among different metric types and through analyzing data correlation, the authors select appropriate time series data metrics to put into LSTM-RNN prediction model. However, so far, there are no any studies that use FLNN for multivariate time series data prediction for cloud computing. As compared with the existing works presented above, our differences and contributions include:

1) Proposing a new approach for proactive auto-scaling problem in clouds using FLNN.
2) Proposing a novel improvement for FLNN, in which the network is trained by GA instead of back-propagation mechanism called FL-GANN.
3) Proposing a prediction resource forecasting system that can be processed multivariate data. This is an important feature of cloud computing while designing auto-scalers in practice.
4) Testing the proposed system performance with real dataset produced by Google cluster.

III. Functional-Link Generic Algorithm Neural Network

A. Designing Prediction System

Our designs for cloud resources forecasting system are shown by figure 1 with four main modules, including Collector, Preprocessor, Trainer, and Forecaster.

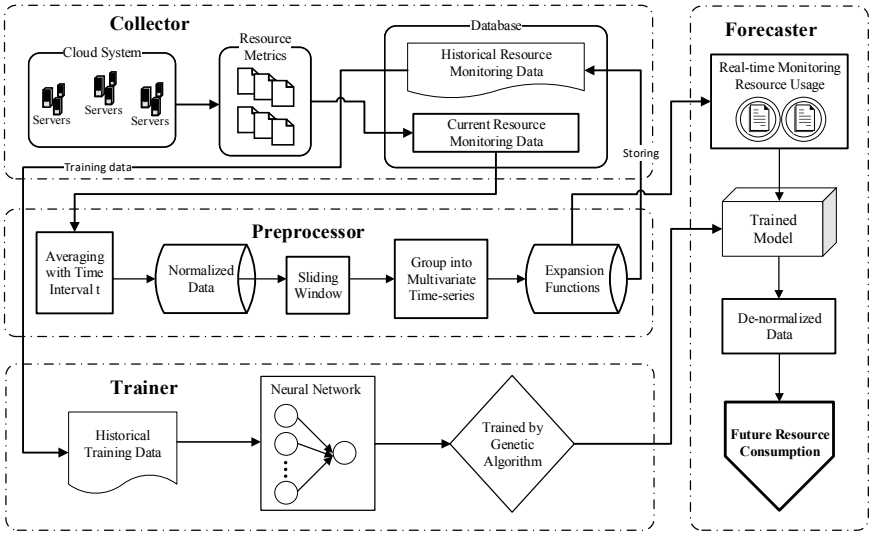

Fig. 1. Multivariate Resources Forecasting System

Through Collector module, raw resource monitoring data is collected from VMs and stored in a repository. Currently, there are a lot of available monitoring services for public clouds such as CloudWatch of Amazon Web Services, IBM cloud monitoring, and Rackspace Monitoring and so forth. In addition, users can deploy and configure monitoring tools like Nagios, Prometheus and Zabbix by themselves on their VMs. The database in this module is used to store current resources monitoring data, which is produced in form of time-series and historical resources monitoring data, which is made by Preprocessor module.

Preprocessor module plays the role of transforming the collected raw time-series data to supervised data to fit the input of neural network. There are several mechanisms that are deployed to process cloud workload data, covering: averaging data in long time period, normalized data, sliding window, group into multivariate time series and expansion functions. After preprocessing in this module, the output data is put into database in Collector as historical resources data, which is used to create prediction model in Trainer module. The data also is provided for Forecaster to predict consumption resources.

A novel learning method is proposed in our Trainer module using FLNN that is a variant of traditional neural network with expansion functions. Furthermore, the network also is trained by GA to speed up the convergence and increase the forecast accuracy. Due to combination of those mechanisms, the proposed learning method is called by FL-GANN.

After the training process finishes, trained model is used to predict future resource consumption in Forecaster module.

B. Preprocessing Monitoring Data

As presented before, the goal of Preprocessor is to prepare data for Trainer and Forecaster. There are five mechanisms deployed in this component. Firstly, the current raw data gathered in a long period is transformed into the corresponding time series $r_i(t)(i = 1, 2, ..., M)$ with time interval ρ. Each point in time series $r_i(t)$ is calculated by averaging all the values of a resource metric usage in the period of ρ as follows: $r_i(n) = \frac{\sum_{i=(n-1)\rho<t<n\rho}^{n} D_i(t)}{n_\rho}$, where $D_i(t)$ is the value of type-i resource at time t that is monitored from cloud system, $n\rho$ is the number of observation $D_i(t)$ in the interval ρ.

The next phase is normalization, which scales a time series in the range of [0, 1]. Then, time-series data is transformed to supervised data by using sliding method with window width k that is the number of values before time t to predict value at the time t. Then all resource metric types are grouped into single multivariate data. Finally, the gained multivariate data undergoes expansion functions such as Chebyshev, Legendre or Power to enable the ability of catching the nonlinear relationship between the inputs and outputs for our neural network. The reason is there are no a hidden layers in the network architecture. Concept of multivariate time series is defined by Definition 1. Meanwhile, functions link that is used in expansion functions component is introduced by Definition 2.

Definition 1: Let $X_1(t), X_2(t), ..., X_i(t), ..., X_M(t)$ are M time series (M metrics resource types), the single multivariate time series is:
$X(t) = [X_1(t), X_2(t), ..., X_M(t)], t = 1, 2, ..., n.$

Definition 2: A functional-link is a function $f \in R^D \rightarrow R$ that transforms X to a scalar (X: vector input patterns). If the input layer of the FLNN consists of functional-link f_i, $i = 1, 2, ..., n$ then input patterns X will be transformed to $Y \in R^n$: $Y = (f_1(X), ...f_n(X))$.

There are many expansion functions, however the most popular are Chebyshev, Legendre, Laguerre, Power Series, Trigonometric.

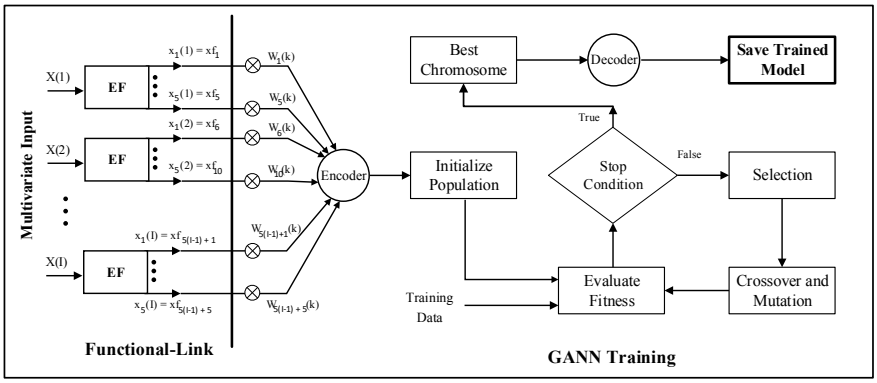

Fig. 2. FL-GANN training process

C. Trainer Module

This module consists of two main components, covering a neural network with single unit of neuron and GA which is used to train the network. The functional-link of FLNN are deployed in Preprocessor presented in Subsection III-B. Figure 2 describes general training process of our proposed FL-GANN model, in which the encoder component is used to encode the weights and bias of network into a chromosome (real-value vector). Otherwise, the decoder component decodes the chromosome into the weights and bias of network. Because the requirement of GA is a fitness function, we calculate Mean Absolute Error (MAE) based on error of training data according to formula (1). The fitness function is reckoned by equation (2). Operations of GA in FL-GANN model are introduced via Algorithm 1.

$$MAE = \frac{\sum_{i=0}^{N}(forecast(i) - y_i)^2}{N} \tag{1}$$

$$Fitness = \frac{1}{MAE} \tag{2}$$

D. Forecaster Module

Our Forecaster module uses values of real-time monitoring data (after preprocessing process) as inputs of the trained model to predict new values (i.e. resource consumption) in advance. Before can be used, the obtained outputs are unnormalized into the real values.

IV. Experiment

In this section, we present experiments as well as evaluations for our proposed system. The tests cover:

1) Comparing prediction accuracy between MLNN, traditional FLNN and FL-GANN;
2) Evaluating influence of GA Hyper-parameters on FL-GANN performance;
3) Evaluating influence of FLNN expansion functions on FL-GANN effectiveness.

Algorithm 1 GANN Training Algorithm

Input: p_s - the population size
d - problem size / d-dimension vector of each individual,
p_c - the probability of 2 individual exchanging crossovers,
p_m - the probability of individual mutation,
g_{max} - the maximum number of generations
Output: The best chromosome.

Initializing population $P = \{C_1, ..., C_{p_s}\}$, each individual is a d-dimension vector $C_i = (c_{i1}, , c_{id})$, $c_{ij} \in$ [-1, 1]
g $\leftarrow 1$
while $g \leq gmax$ **do**
  Calculating fitness of population based on formula (2)
  Finding the best chromosome C_{best} according to the achieved fitness value.
  repeat
    Using Routtele Wheel Selection to choose two individuals (parents p_1, p_2) based on the fitness value to cross-over with probability p_c to make 2 offspring (CH_1, CH_2) based on:
    $CH_{1i} \leftarrow \lambda p_{1i} + (1 - \lambda)p_{2i}$
    $CH_{2i} \leftarrow \lambda p_{2i} + (1 - \lambda)p_{1i}$, $\lambda \sim U(0, 1), i = \overline{1, d}$
    Then offspring undergo the mutation process by replacing each $CH_{ij}, i = \overline{1, 2}, j = \overline{1, d}$, with probability p_m by a random uniform value $\sim U(0, 1)$.
    Appending offspring into a new generation.
  until new generation has p_s individuals
  Replacing the old generation by the new generation.
  g $\leftarrow g + 1$
end while
Return C_{best}

A. Experimental Setup

In our experiments, we use a real workload dataset provided by Google [16] in one month of 2011. In the dataset, each job is a collection of many tasks that are run simultaneously on multiple machines. Resource utilization of tasks are measured by several metrics such as CPU, and memory usage, disk

TABLE I
MAE COMPARISON OF MLNN, TRADITIONAL FLNN AND FL-GANN MODELS

Input Type	Model	CPU			RAM		
		k = 2	k = 3	k = 5	k = 2	k = 3	k = 5
Univariate	MLNN	0.3327	0.3514	0.3570	0.0288	0.0265	0.0273
	FLNN	0.2944	0.2999	0.3054	0.0210	0.0201	0.0215
	FL-GANN	**0.2829**	**0.2812**	**0.2843**	**0.0195**	**0.0197**	**0.0202**
Multivariate	MLNN	0.3314	0.3387	0.3448	0.0266	0.0271	0.0289
	FLNN	0.2971	0.2903	0.3201	0.0218	0.0202	0.0226
	FL-GANN	**0.2815**	**0.2814**	**0.2902**	**0.0194**	0.0207	**0.0212**

I/O mean time, and so on. According to the analyses also presented in [16], 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 Google cluster. In order to evaluate the generalization of our prediction model, we select a long running job with ID 6176858948. The job consists of 60171 divergent tasks during the 20-day period (from 1st to 20th day). We set average time t = 5 minutes, forecast horizon k = 1 in all experiments. The data from 1st to 15th day is used to train neural networks. Meanwhile, the data from 16th to 20th day is employed to test the prediction performance of these networks. For the multivariate input case, both CPU and memory usage data is used simultaneously for the learning models. Meanwhile, in the case of univariate, only one metric (CPU or memory) is put into the prediction models. To show the effectiveness of our proposed model, we compare performance of MLNN, traditional FLNN and FL-GANN in the tests below.

The MLNN thus is configured with 5 layers (1 input, 1 output and 3 hidden). The neuron number for layers is set in succession as follows: k, 10, 15, 5, and 1. Traditional FLNN and FL-GANN have only one input and output layer with structure $(k, 1)$. Here, k is the sliding window value used in the Preprocessor module. Activation function used for all three networks are Exponential Linear Unit (ELU).

B. Forecasting Resource Consumption with Multivariate Input Data

In this test, we evaluate the efficiency of FL-GANN against MLNN, and traditional FLNN in forecasting resource consumption. For each model, we also compare univariate (single input metric) and multivariate (multiple input metrics) data. We change sliding windows size k from 2 to 5 (k = 2, 3, 5) in this test. Our achieved MAE outcomes are given in Table I.

The results point out that MAE accuracy of FL-GANN are almost smaller than FLNN and MLNN model with different sliding window values as well as input types. Concretely, in the case of univariate data input, FL-GANN brings the best results in comparison with traditional FLNN and MLNN. For multivariate data, there is only one case of k = 3 and memory consumption prediction, MAE of FL-GANN is lower than traditional FLNN. Otherwise, the FL-GANN has better performance as compared with other models.

Figure 3 and 4 show the predicted results between FLNN and FL-GANN models for CPU and memory with multivariate input data. While, the blue line is the actual usage, orange line is the predicted values. It can be seen that prediction curve of FL-GANN tends to closer to the actual value curve in comparison with forecast FLNN curve for both CPU and memory with multivariate input data. Also, for the weirdo points (both high and low), the results also show that the FL-GANN model is better than FLNN and MLNN.

C. Influence of GA hyper-parameter on FL-GANN

In this experiment, we focus on evaluating the influence of different GA hyper-parameters on our FL-GANN with multivariate input data. The parameters are pre-set for this test as follows.

1) For population size p_s changing experiment: p_s value is put out in turn from $\{50, 100, 200, 300, 400, 500, 600\}$, sliding window = 3, expansion functions is Power Series, $g_{max} = 650$, $p_c = 0.95$, and $p_m = 0.025$.
2) For probability of two individual exchanging crossover p_c changing experiment: p_c value is put out in turn from $\{0.70, 0.75, 0.80, 0.85, 0.90, 0.95, 0.98\}$, sliding window = 3, expansion functions is Power Series, $g_{max} = 650$, $p_s = 500$, and $p_m = 0.025$.
3) For probability of individual mutation p_m changing experiment: p_m value is put out in turn from $\{0.005, 0.01, 0.025, 0.05, 0.065, 0.80, 0.10\}$, sliding window = 3, expansion functions is Power Series, $g_{max} = 650$, $p_s = 400$, and $p_c = 0.95$.

Population size changing experiment. For each p_s, we carry on 20 times of MAE and RMSE measurements then calculate average values, which are shown in Table II. The evident observation is that with p_s = 50, MAE's and RMSE's average values reach the biggest numbers at 0.2988, and 0.4913, respectively. When p_s = 600, the both average error accuracy values are the smallest number at 0.2899, and 0.4821, respectively. Moreover, the MAE results are produced by p_s at 400, and 500 that approximate the value is produced by p_s at 600 although population size is increased by 100 and 200, respectively. Based on figure 5, we also recognize that MAE

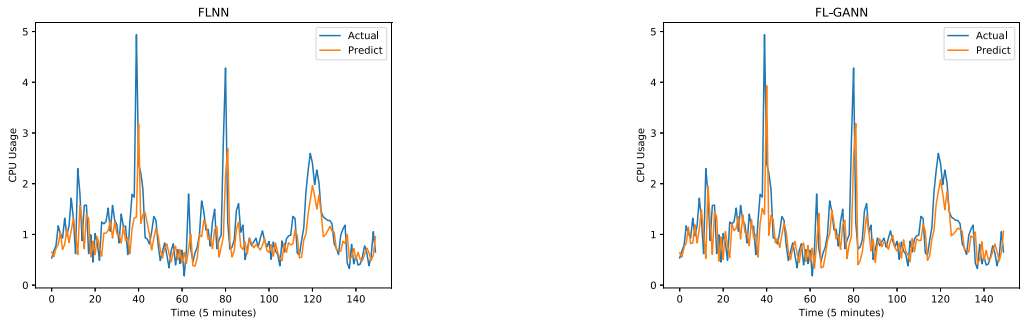

Fig. 3. CPU prediction outcomes of FLNN and FL-GANN with multivariate input data

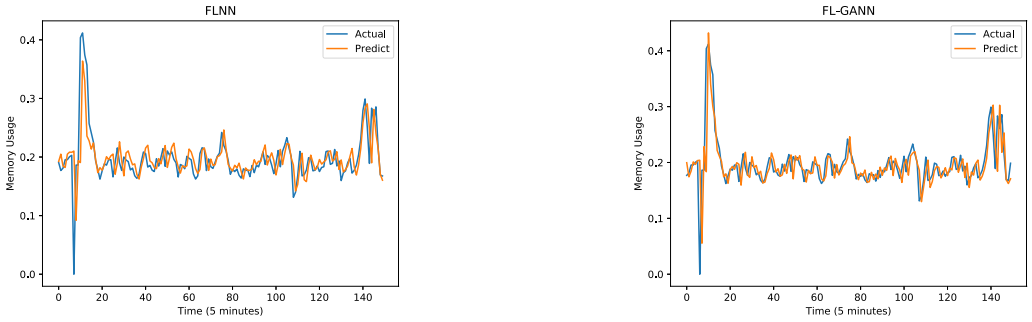

Fig. 4. Memory prediction outcomes of FLNN and FL-GANN with multivariate input data

values of p_s at 400, and 500 are smaller than MAE values of p_s at 600. This infers that the smaller population, the worse of outcomes and if the population is large, the results may be better. However, in that case, the proposed model will take time to learn. Hence, the population size is from 300 to 600 producing acceptable performance for the tested data.

Figure 5 indicates the MAE range of population sizes with diverse p_s parameters using box-and-whisker plots. While the intermittent lines in the middle of boxes represent the mean values, the upper and lower boundaries of the boxes represents upper and lower quantiles of the distributions. Via those results shown by this figure, a remark could be brought forward as follow. The population size has the smallest average MAE accuracy at p_s = 600 (average MAE is equal to 0.2898 in this case).

Probability of two individual exchanging crossover changing experiment. For this test, we change p_c from 0.7 to 0.98 to evaluate error accuracy via MAE and RMSE measures. Table III gives the average accuracy values of the measurements after 20 running times with each p_c. An observation can be made from those outcomes. When p_c = 0.95, average MAE accuracy gets 0.2897, which is minimum value as compared to others. The better p_c range for FL-GANN is 0.9 to 0.98. This means the probability of two individual exchanging crossovers to create offspring gets higher value, the results may get better. More specifically, figure 6 presents the achieved MAE ranges with diverse p_c. Through the figure, it can conclude

TABLE II
COMPARISON OF AVERAGE MAE WITH DIFFERENT POPULATION SIZES

Population size	**Error Accuracy**	
	MAE	RMSE
p_s = 50	0.2988	0.4913
p_s = 100	0.2940	0.4846
p_s = 200	0.2925	0.4860
p_s = 300	0.2924	0.4833
p_s = 400	0.2907	0.4834
p_s = 500	0.2908	0.4823
p_s = 600	**0.2899**	**0.4821**

that the probability of two individual exchanging crossover in the range of [0.9, 0.98] producing acceptable results for the tested data.

Probability of individual mutation changing experiment. Table IV shows the MAE and RMSE average values obtained after 20 running times with each p_m. The average MAE accuracy is 0.2911 when p_m = 0.025. In the case of p_m value is increased to 0.08 or even higher, the average MAE and RMSE values are bigger. This proves that the higher probability of individual mutation gets, the results may worse. Figure 7 is used to illustrate MAE range with different p_m. Via this

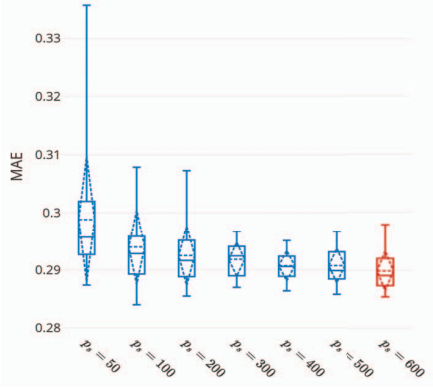

Fig. 5. MAE accuracy fluctuation comparison of with different population sizes

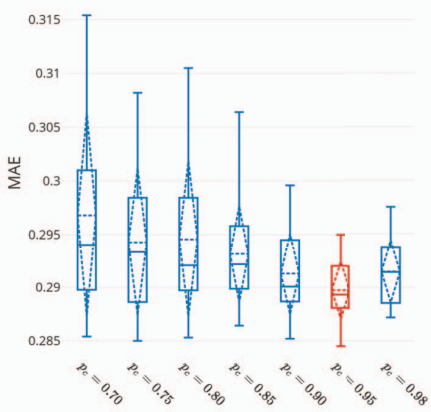

Fig. 6. MAE accuracy fluctuation comparison with different probability of two individuals exchanging crossovers

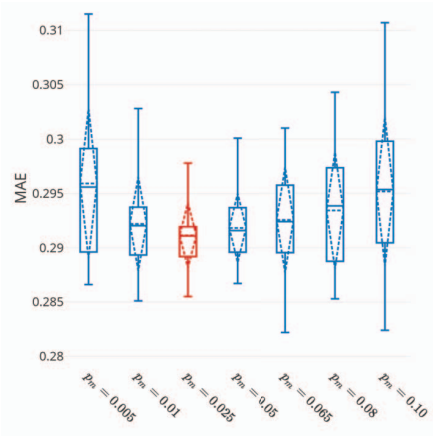

Fig. 7. MAE accuracy fluctuation comparison with different probability of individual mutation

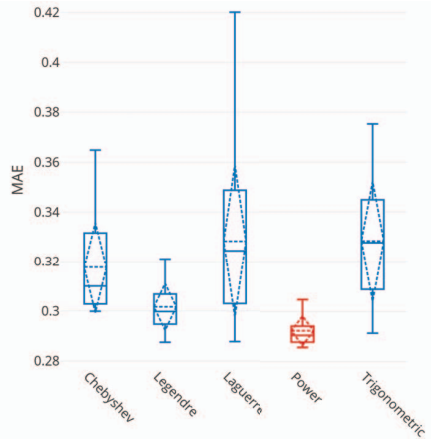

Fig. 8. Comparison of expansion functions influence on FL-GANN

TABLE III
COMPARISON OF AVERAGE MAE AND RMSE WITH DIFFERENT PROBABILITY OF TWO INDIVIDUALS EXCHANGING CROSSOVERS

Crossover Probability	**Error Accuracy**	
	MAE	RMSE
p_c = 0.70	0.2968	0.4818
p_c = 0.75	0.2943	0.4854
p_c = 0.80	0.2946	0.4852
p_c = 0.85	0.2931	0.4858
p_c = 0.90	0.2913	0.4855
p_c = **0.95**	**0.2897**	**0.4836**
p_c = 0.98	0.2915	0.4869

TABLE IV
COMPARISON OF AVERAGE MAE AND RMSE WITH DIFFERENT PROBABILITY OF INDIVIDUAL MUTATION

Mutation Probability	**Error Accuracy**	
	MAE	RMSE
p_m = 0.005	0.2959	0.4921
p_m = 0.01	0.2922	0.4847
p_m = **0.025**	**0.2911**	**0.4829**
p_m = 0.05	0.2918	0.4830
p_m = 0.065	0.2926	0.4843
p_m = 0.08	0.2938	0.4847
p_m = 0.10	0.2952	0.4868

experiment, it can remark that MAE and RMSE results which are produced by the probability of individual mutation in the range of [0.01, 0.08] is acceptable with the tested data.

D. Influence of Expansion Functions on FL-GANN

The goal of this experiment is to evaluate the influence of different expansion functions on our FL-GANN in the

TABLE V
COMPARISON OF EXPANSION FUNCTION INFLUENCE ON FL-GANN

Expansion Functions	CPU			RAM		
	k = 2	k = 3	k = 5	k = 2	k = 3	k = 5
Chebyshev	0.3208	0.3654	0.3424	**0.0205**	0.0244	0.0288
Legendre	0.2932	**0.2908**	0.3347	0.0211	0.0227	**0.0211**
Laguerre	0.4102	0.4889	0.4485	0.0283	0.0333	0.0342
Power	**0.2860**	0.2915	**0.2914**	0.0215	**0.0206**	0.0228
Trigo*	0.3069	0.3257	0.4388	0.0346	0.0233	0.0261

*Trigonometric function

learning phase. In this way, we test five popular expansion functions including Chebyshev, Legendre, Laguerre, Power, and Trigonometric. The input data is multivariate. The sliding window k in this experiment is set to 2, 3, and 5. The gained MAE outcomes with CPU and memory are given through Table V. In which, Power Series function produces the smallest MAE values, which are 0.2860 with k = 2, 0.2914 with k = 5 (CPU prediction), and 0.0206 with k = 3 (memory forecast). On the other hand, Legendre function brings two minimum MAE values, which are 0.2908 with k = 3 (CPU prediction) and 0.0211 with k = 5 (memory forecast). Unfortunately, Laguerre and Trigonometric functions do not yield any the smallest MAE in our evaluations.

In order to more objectively evaluate Power Series and Legendre functions with FL-GANN, we carry on a small additional experiment for CPU prediction with k = 3. Thus, we run 20 times for each the function, obtained MAE values of these all runs are shown by the box-and-whisker diagram 8. It can be conclude that MAE fluctuation of the Power Series function is the smallest, also the average MAE (intermittent line in the box) is also the smallest as compared to Legendre function.

V. CONCLUSION AND FUTURE WORK

In this paper, we presented our designs for a novel cloud resource prediction system using functional-link neural network. We also proposed the use of genetic algorithm instead of back-propagation technique with gradients descent to overcome drawbacks of the training process for traditional FLNN. Besides, in our system, the multiple resource metrics are processed simultaneously using several proposed data preprocessing mechanisms. This helps the prediction system can forecast more precisely resource consumptions in advance because the metrics often have relationship each other. We tested our FL-GANN model with a real dataset provided by Google. The achieved results show that although the forecast method is simple but it can bring good performance in prediction process even though the system exploits multivariate data. The outcomes also prove our proposals can be applied to practice. In the near future, we plan to evaluate several other variants of FLNN and to develop an auto-scaler service with decision phase for clouds using this prediction technique.

ACKNOWLEDGMENTS

This research is supported by the Vietnamese MOET's project No. B2017-BKA-32 "Research on developing software framework to integrate IoT gateways for fog computing deployed on multi-cloud environment", Slovak VEGA 2/0167/16 project "Methods and algorithms for the semantic processing of Big Data in distributed computing environment", and EU H2020-777533 project PROCESS "PROviding Computing solutions for ExaScale ChallengeS".

REFERENCES

[1] Nguyen Minh, B., Tran, V., and Hluchy, L. (2012). Abstraction layer for development and deployment of cloud services. Computer Science, 13, 79-88.
[2] Nguyen, B. M., Tran, D., and Nguyen, G. (2016). Enhancing service capability with multiple finite capacity server queues in cloud data centers. Cluster Computing, 19(4), 1747-1767.
[3] Hluch, L., Nguyen, G., Astalo, J., Tran, V., ipkov, V., and Nguyen, B. M. (2017). Effective computation resilience in high performance and distributed environments. Computing and Informatics, 35(6), 1386-1415.
[4] Dehuri, Satchidananda, Bijan Bihari Mishra, and Sung-Bae Cho. "Genetic feature selection for optimal functional-link artificial neural network in classification." International Conference on Intelligent Data Engineering and Automated Learning. Springer, Berlin, Heidelberg, 2008.
[5] Hipel, Keith W., and A. Ian McLeod. Time series modelling of water resources and environmental systems. Vol. 45. Elsevier, 1994.
[6] Pao, Yohhan. "Adaptive pattern recognition and neural networks." (1989).
[7] Majhi, Ritanjali, Ganapati Panda, and Gadadhar Sahoo. "Development and performance evaluation of FLANN based model for forecasting of stock markets." Expert systems with Applications 36.3 (2009): 6800-6808.
[8] Majhi, Babita, Minakhi Rout, Ritanjali Majhi, Ganapati Panda, and Peter J. Fleming. "New robust forecasting models for exchange rates prediction." Expert Systems with Applications 39, no. 16 (2012): 12658-12670.
[9] Khandelwal, Ina, Udit Satija, and Ratnadip Adhikari. "Forecasting seasonal time series with functional link artificial neural network." Signal Processing and Integrated Networks (SPIN), 2015 2nd International Conference on. IEEE, 2015.
[10] Sahoo, Deepti Moyi, and Snehashish Chakraverty. "Functional link neural network learning for response prediction of tall shear buildings with respect to earthquake data." IEEE Transactions on Systems, Man, and Cybernetics: Systems 48.1 (2018): 1-10.
[11] Dehuri, Satchidananda, and Sung-Bae Cho. "A comprehensive survey on functional link neural networks and an adaptive PSOBP learning for CFLNN." Neural Computing and Applications 19.2 (2010): 187-205.
[12] Montana, David J., and Lawrence Davis. "Training Feedforward Neural Networks Using Genetic Algorithms." IJCAI. Vol. 89. 1989.
[13] Tran, D., Tran, N., Nguyen, G., and Nguyen, B. M. (2017). A proactive cloud scaling model based on fuzzy time series and SLA awareness. Procedia Computer Science, 108, 365-374.
[14] Tran, N., Nguyen, T., Nguyen, B. M., and Nguyen, G. (2018). A Multivariate Fuzzy Time Series Resource Forecast Model for Clouds using LSTM and Data Correlation Analysis. Procedia Computer Science, 126, 636-645.
[15] Blanco, Armando, Miguel Delgado, and Maria C. Pegalajar. "A real-coded genetic algorithm for training recurrent neural networks." Neural networks 14.1 (2001): 93-105.
[16] Reiss, Charles, et al. "Heterogeneity and dynamicity of clouds at scale: Google trace analysis." Proceedings of the Third ACM Symposium on Cloud Computing. ACM, 2012.
[17] Prevost, J. J., Nagothu, K., Kelley, B., & Jamshidi, M. (2011, June). Prediction of cloud data center networks loads using stochastic and neural models. In System of Systems Engineering (SoSE), 2011 6th International Conference on (pp. 276-281). IEEE.