Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2018) 000–000

Procedia
Computer Science

www.elsevier.com/locate/procedia

22nd International Conference on Knowledge-Based and
Intelligent Information & Engineering Systems

A Multivariate Fuzzy Time Series Resource Forecast Model for Clouds using LSTM and Data Correlation Analysis

Nhuan Tran[a], Thang Nguyen[a], Binh Minh Nguyen[a,*], Giang Nguyen[b]

[a] *School of Information and Communication Technology, Hanoi University of Science and Technology, Hanoi, Vietnam*
[b] *Institute of Informatics, Slovak Academy of Sciences, Bratislava, Slovakia*

Abstract

Today, almost all clouds only offer auto-scaling functions using resource usage thresholds, which are defined by users. Meanwhile, applying prediction-based auto-scaling functions to clouds still faces a problem of inaccurate forecast during operation in practice even though the functions only deal with univariate monitoring data. Up until now, there are still very few efforts to simultaneously process multiple metrics to predict resource utilization. The motivation for this multivariate processing is that there could be some correlations among metrics and they have to be examined in order to increase the model applicability in fact. In this paper, we built a novel forecast model for cloud proactive auto-scaling systems with combining several mechanisms. For preprocessing data phase, to reduce the fluctuation of monitoring data, we exploit fuzzification technique. We evaluate the correlations between different metrics to select suitable data types as inputs for the prediction model. In addition, long-short term memory (LSTM) neural network is employed to predict the resource consumption with multivariate time series data at the same time. Our model thus is called multivariate fuzzy LSTM (MF-LSTM). The proposed system is tested with Google trace data to prove its efficiency and feasibility when applying to clouds.

Keywords: Cloud computing; proactive auto-scaling; multivariate fuzzy time series; long-short term memory; LSTM; data correlation; symmetrical uncertainty measure; Google trace dataset

1. Introduction

One of the main advantages of cloud computing is the ability to provide flexible and fast computational resources on demand. Currently, most of cloud providers offer users a certain auto-scaling mechanism, which operates based on predefined resource usage thresholds. However, this approach creates a delay during provisioning resources process [7]. For example, an user sets a scale-out threshold of 80% memory utilization for a virtual machine (VM) deployed

* Corresponding author. Tel.: +84-24-3869-6124 ; fax: +84-24-3869-2906.
E-mail address: minhnb@soict.hust.edu.vn

on cloud infrastructure. In theory, when the VM memory usage reaches 80% capacity, the cloud will automatically add one more VM for the user's system. However, the scaling function needs to spend a few moments to complete the deployment of second VM. During this time, the memory usage has already been changed from the set threshold to another that can be bigger or smaller than 80%. This causes not only the decrease of service quality but also the increase of operation cost for both users and vendors while adopting cloud technologies.

Although the threshold approach has limitation as presented above, it is still widely used in clouds today. Meanwhile, applying prediction methods to provision resources is quite uncommon. Several reasons can explain for this. First, the prediction accuracy always is a concern of cloud vendors and even users, who mainly focus on efficiency of scaling functions in the manner of improving service quality. Hence, if prediction has low precision, the resources over or missing provision could occur. This leads to decrease user reliance in services provided by cloud vendors. Second, almost all existing forecast model designed for cloud computing just have dealt with single resources utilization. In other word, the recent proposed algorithms were designed to predict future resource consumption for single metrics. Nonetheless, to make auto-scaling decision efficiently and exactly, the forecast model must consider many resource metrics such as CPU, memory, disk I/O and so on at the same time. Until now, for cloud computing, there are not any proactive auto-scaling solutions, which have analyzed with the relationships to select appropriately metrics to put into the forecast model.

The works presented in this paper concentrate on proposing a new prediction system for proactive auto-scaling services based on analyzing multivariate time series data (i.e. multiple metrics) at the same time. Several mechanisms are designed to increase the accuracy of the forecast model. Firstly, because monitored utilization data is collected continuously by time, the data could be very noisy. To reduce effect of the data fluctuation on prediction accuracy, fuzzy technique is used to pre-process data. Secondly, the correlations among different metrics are examined to choose suitable data types before putting into prediction model. This helps the proposed forecast system to determine metrics that have high correlation for the prediction process. This mechanism could bring better performance for our forecast model when dealing with multivariate data. Finally, long-short term memory (LSTM) neural network is employed to predict the future resource consumption. In order to evaluate effectiveness of the proposed system, we carry out experiments with Google trace dataset [8]. The obtained outcomes demonstrate the feasibility of our approach for multivariate cloud resource prediction system in practice with high efficiency.

The rest of document is organized as follows. In section 2, we give our motivations for the mechanisms that are designed to the proposed system. We discuss some related studies to highlight the differences between our work and existing efforts in Section 3. In the next part, we present designs in detail for our proactive auto-scaling system. Section 5 describes the experiments, gained results, observations, and evaluations of the system with real dataset. Finally, Section 6 concludes and defines some future directions for our research.

2. Research motivation

Our work is come into being based on the following motivations:

Multivariate time series data analytics. The cloud monitoring data often is considered as time series data, which contains many different metric types. Hence, there is a need of having a certain solution to analyze these metrics at the same time to approach cloud auto-scaling problem in practice. For example, a Database Management System (DBMS) is deployed on a virtual machine (VM) with a virtual disk (VD). Due to huge number of writing and reading operations, the VD reaches I/O speed limitation, but the utilizations of CPU and memory are still low. In this case, the system has to scale out to ensure the ability of processing data for the DBMS service.

Fuzzy time series. Time series data collected from clouds usually is quite noisy with continuous fluctuation by time (even second or millisecond). For instance, we extract and illustrate the CPU utilization of an application from Google trace dataset [8] via Figure 1. It can be observed that there are many data fluctuations in a short time period. This phenomenon also impacts on efficiency of prediction as well as auto-scaling functions. Hence, there is a great significance if having a solution that can decrease the data fluctuation but keep the accuracy for the prediction-based auto-scaling model.

Correlation of different data type. There may be implicit correlations between monitored metric types. For instance, CPU and memory usages or CPU, memory utilization and disk I/O, which are collected from one VM may have relationships each other. In terms of analyzing many data types simultaneously, having a mechanism to evaluate the data type correlation will bring valuable information for prediction models in the manner of increasing its accuracy.

Proactive auto-scaling and prediction accuracy. In fact, existing cloud monitoring services provide the abilities of infrastructure cost control, SLA violation prevention and QoS guarantee via measuring all resource metrics. For example, CloudWatch [1] enables customers to collect resource utilization usages of AWS cloud such as CPU utilization, network in/out bytes, disk read/write operations and so on. If there is a certain solution, which can analyze the data to forecast resource requirement in advance, system administrators may make early decisions to allocate/release a right amount of resources to/from the applications without any delays, thus improve provided resource QoS. Although a lot of prediction models have been proposed, enhancing the accuracy for them still is the main target of researchers and developers. The reason is that the more accurately predicted, the lower resource cost.

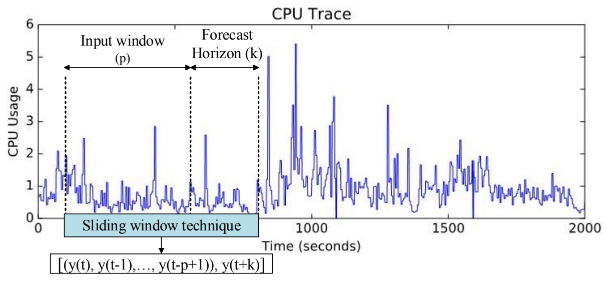

Fig. 1: CPU utilization trace of a job on Google cluster

3. 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 [13], several forecasting methods including autoregressive (AR), moving average (MA), exponential smoothing, error-trend-seasonal (ETS), automated autoregressive integrated moving average (ARIMA) and back-propagation neural networks (BPNN) were evaluated and compared in predicting cloud workloads. Tran et al. [10] put forward PD-GABP algorithm, which combines data periodic detection and neural network to improve the forecasting accuracy. Recently, deep learning has emerged as an effective solution for prediction problem. In which, Recurrent Neural Network (RNN) is a kind of folded neural network that is considered as a useful for time series data analytics because there are dependencies among points in this data type and RNNs are designed to model temporal dependencies and trained by back-propagation through time. RNN method distinguishes from feed-forward neural networks (FFNNs) by its operation. Thus, information flows backwards through feedback loops. However, a vanilla RNN suffers from exploding and vanish gradients problem. To improve memorization of RNN and control better the information flow, Hochreiter et al. [5] proposed Long Short Term Memory (LSTM), which is the most popular blocks inside RNN layers. With the improvement, RNN has the ability of recognizing and learning long-term dependencies. In term of applying to clouds, Jitendra et al. [6] proposed the LSTM-RNN model for workload prediction on cloud data-centers using NASA dataset. In [11], the authors built an auto-scaling cloud system, which exploits multivariate monitoring resources, genetic algorithm, back-propagation, and BPNN to forecast workload. Although several deep learning efforts have been proposed to use for clouds, those works still considered the problem with univariate time series.

As presented in the previous Section, for *multivariate data analytics*, the prediction outcomes would gain better if the correlations among different metrics are also considered [2]. Although in theory, multivariate dataset would provide more prediction accuracy, in practice, the irrelevant or non-correlated features could cause the problem of distorting forecast model. Several studies have proposed solutions for correlation determination. Sabine et al. [12] built a framework for grouping variables via correlation characteristic using diverse methods. The authors of [16] also created a framework of efficient feature selection via redundancy and relevant analysis using non-linear correlation methods. Vergara et al. [14] developed a unifying solution to select theoretic features with the algorithms for sequential forward selection and sequential backward elimination. However, there are not many efforts that have dealt with discovering correlations in cloud workload metrics. Otherwise, there is no mechanism yet, which enables prediction systems to determine the appropriate number of metrics in order to put together into forecast model at the same time today.

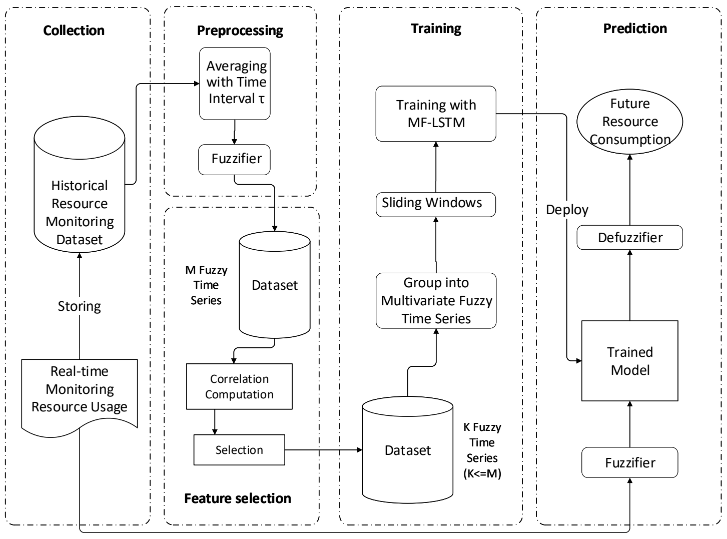

Fig. 2: Architecture of Multivariate Fuzzy LSTM for time series forecasting

The concept of *fuzzy time series* has been put forward firstly in [9], which addressed only one factor first-order model (i.e. univariate). To deal with multivariate data, Erol et al. [3] proposed a neural network that can be used for determining fuzzy relationships between two factors having high order fuzzy time series. Tran et al. [11] introduced the k-factors high order fuzzy time series using genetic algorithm-based artificial neural network (GABPNN) in order to discover hidden fuzzy relationship as well as to take advantage of fuzzy technique to reduce fluctuation in real dataset. In this study, we also propose a fuzzy time series model to deal with k-factor and noisy problem of datasets collected from cloud resources. However, for pre-processing, and learning processes, we evaluate and select only high correlated metrics and use LSTM to increase the prediction accuracy respectively.

4. Designing system architecture

The overall proposed architecture for our system is shown by Figure 2 that consists five modules including collection, preprocessing, feature selection, training and prediction. The collection module plays the role of storing and providing monitoring data for our system. There are two data types covering: historical and real-time. The first type is data collected in the past and it is used for the preprocessing, feature selection, and training modules to create a trained forecast model in the prediction module. The second data type is resource utilization gathered in real-time, which is immediately fuzzified and put to the trained model to predict consumptions in advance. The real-time data will be stored in the historical resource monitoring database for process of updating the prediction model. The preprocessing module is designed to convert raw data into a transformed and clean dataset. In this stage, the historical data in long time period is averaged with each time interval τ. After that the obtained time series will be fuzzified. The outcomes of fuzzifier operation are fuzzy time series of all monitoring metrics. The feature selection module's goal is to choose metrics which have high relevant for the learning phase. First, we compute the nonlinearity correlation of all metrics. Then an algorithm is run with a specific threshold δ in order to select the significant relevant metrics. In the model training module, data is grouped together into a multivariate time series and then the sliding window technique (time-step) is applied to create input-output pairs, which are fed to LSTM neural network. As mentioned above, the proposed forecast model is called MF-LSTM. In the prediction module, the trained model and fuzzy time series data of real-time usage are used to predict future resource consumption. At the final step, these forecast values are defuzzified.

4.1. Preprocessing

Through the preprocessing module, a suitable dataset for the learning part in our system is constructed. In the first phase, values of each cloud monitoring metric are transformed into the corresponding time series $r_i(t)$ $(i = 1, 2, \ldots, M)$ with time interval τ. Each point in time series $r_i(t)$ is calculated by averaging all monitored values of a resource metric with the following formula $r_i(n) = \frac{\sum_{(n-1)\tau<t<n\tau} D_i(t)}{n_\tau}$, where $D_i(t)$ is the monitored value of type-i resource metric at time t, and n_τ is the number of observation $D_i(t)$ in the interval τ. That time interval τ must be chosen reasonably in accordance with input data as well as cloud system characteristics because the interval directly affects sliding window of learning model generated by the next modules. The window size and forecast horizon (as depicted by Figure 1) of sliding window technique impact on prediction performance. For example, assuming the cloud infrastructure needs 10 minutes to finish resource provision process. Hence, a proactive auto-scaling system must predict the next 10 minutes of resource usages in order to trigger the scaling actions in advance. In the case of $\tau = 1$(s), the forecast horizon of sliding window is $k = 10 * 60/1 = 600$(s), which is too large and not suitable in time-series forecasting. It is clear that τ should be larger to reduce the forecast horizon.

The next phase is fuzzification process, which transforms real-valued dataset into a corresponding fuzzy time series. Definitions of fuzzy time series were presented in [11]. Thus, the fuzzifier component maps each real value into a certain fuzzy set using its maximum value. The fuzzification process also was presented in [3] with three steps, including: defining universe of discourses and intervals, defining fuzzy sets, and mapping each crisp value into maximum-membership fuzzy set. In our system, we use these steps for the fuzzification phase.

4.2. Feature selection

In the feature selection module, Algorithm 1 is applied to select metrics that have high correlation. Thus, firstly, we compute the symmetrical uncertainty (SU) [16]of all metrics. Then, we choose the metric pair which have the maximum correlation. For the others, we continue to evaluate the correlation of each metrics with the selected pair. If the achieved correlation is better than a certain threshold, the third metric is chosen together with the pair that we already have had before. The process is repeated until for all to evaluate and find out a suitable metric set. In the case of gained correlation does not meet threshold, the evaluated metric is eliminated. With this approach, we have entropy, which is an uncertainty measure of random variable X as follows $H(X) = -\sum_i P(x_i)\log[2]P(x_i)$. The entropy of X after observing values of another variable Y is reckoned by the following formula $H(X|Y) = -\sum_j P(y_j)\sum_i P(x_i|y_j)\log[2]P(x_i|y_j)$, where $P(x_i)$ is the prior probabilities for all values of X and $P(x_i|y_i)$ is the posterior probabilities of X given the values of Y. Thus, information gain $IG(X/Y)$ is the amount of uncertainty in X minus the amount of uncertainty in X after Y is known, given by the following formula $IG(X|Y) = H(X) - H(X|Y)$.

According to the measurement, if the correlation of Y and X is higher than Y and Z, the information gain among three features X, Y, and Z is expressed by formula: $IG(X|Y) > IG(Z|Y)$. However, information gain is a symmetrical measure, that means $IG(X, Y)$ equals $IG(Y, X)$. In addition, the information gain has a bias in favor of features with more number of values. In other word, if attributes having great numbers of values will obtain more information than those with fewer values even if they are not much correlation. To overcome this problems, the symmetrical uncertainty of X and Y [15] compensates for information gain bias with the following equation $SU(X, Y) = 2[\frac{IG(X|Y)}{H(X)+H(Y)}]$. Thus, $SU(X, Y)$ always is in range of 0 and 1, where 1 is equivalent to the meaning of X and Y have the absolute correlation. Meanwhile, 0 is equivalent to X and Y are independent. In this case, $SU(X, Y)$ still preserves the symmetric property.

Through the processing of feature selection module, nonlinearity correlation of all metrics is calculated. Based on that, the module chooses at most k ($k \leq$ n) metrics, which have maximum nonlinearity. At the end of selection phase, a dataset containing k fuzzy time series with maximum nonlinear correlations is created for the training module.

4.3. Training

Our learning module is carried out with three stages, namely grouping into multivariate fuzzy time series, sliding window, and training of LSTM neuron network. The future values of fuzzy time series could be predicted when hidden fuzzy logic relationship is known. The relationship could be discovered through historical fuzzy time series. In our proposal, long-short term memory network (LSTM) also is used to determine the fuzzy relationships in learning process from historical data. Our LSTM model of fuzzy relationships is depicted by Figure 3.

Algorithm 1: Selecting high correlation metrics

```
1  Input Threshold $\delta$, and a multivariate fuzzy dataset F = $\{F_1, ..., F_n\}$
2  Output Subset SS
3  //Compute symmetrical uncertainty among all metrics
4  Initialization: S is bidirectional array with default value = 0 with length=$n$
5  for $i = 1$ to $n$ do
6      for $j = 1$ to $n$ do
7          $S(i, j) = \mathrm{SU}(F_i, F_j)$
8      end
9  end
10 // Pick two metrics of the maximum of S
11 $(F_i, F_j)=argmax(S(.., ..))$
12 SS=$\{F_i, F_j\}$
13 for $k = 1$ to $n$ do
14     if ($k!=i$ and $k!=j$ and $SU(F_i, F_k) >= \delta$ and $SU(F_j, F_k) >= \delta$) then
15         $SS = SS \cup F_k$
16     end
17 end
18 Return SS
```

After grouping values into multivariate fuzzy time series, the training phase of LSTM network will be carried out. During the process, each input/output pair is constructed by sliding window technique before feeding to the neural network. In terms of operation, LSTM unit has a memory cell and three gates: input, forget and output gate. The input gate controls how much new information will be updated in memory cell, the forget gate allows how much information should be forgot/remembered in the cell and the output gate decide how much cell state affects the hidden state output. LSTM model is defined by 10.40, 10.41, 10.42, 10.43 and 10.44 equations in [4].

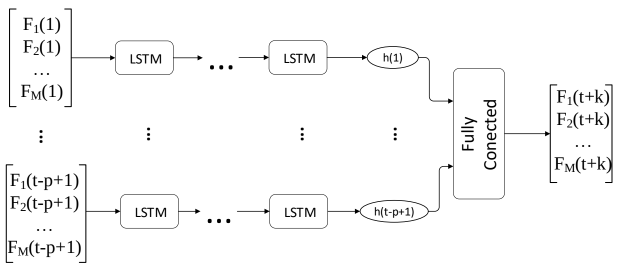

Fig. 3: LSTM model of fuzzy relationships

As presented above, sliding window technique is exploited to create pairs of the inputs (a fixed number p of previous observed values $y(t), y(t1), ..., y(tp + 1)$) and output (k-step ahead forecast $y(t + k)$). The sliding window technique in time series forecasting also is expressed by Figure 3, where p is sliding window size and k is forecast horizon. For multivariate fuzzy time series, we employ sliding window technique in the same way described above. In which the inputs are $F(t), F(t-1), F(t-2), \ldots, F(t-p+1)$, and the output is $F(t+k)$. A common condition in time series forecasting is that p is not less than k. The operation of MF-LSTM model is described by Algorithm 2.

4.4. Prediction

As shown in Figure 2, the real time monitoring resource usage will be fuzzified and become input of the model trained by MF-LSTM model. The output value of trained model will be defuzzified into the real values. This is also the final step of prediction process. For defuzzification centroid method (also called as center of area of center of gravity) is adopted. The defuzzification stage for fuzzy time series forecasting is presented in [11] that also is applied to our design. The operations of prediction module are described by Algorithm 3.

Algorithm 2: MF-LSTM

```
1 Normalizing and scaling all historical resource consumptions of M metrics into $r_1(t), r_2(t), r_3(t), ..., r_n(t)$
2 Fuzzifying time series of $n$ resource types into the $n$ corresponding fuzzy time series:
    $F_1(t), F_2(t), \ldots, F_i(t), \ldots, F_M(t)$
3 Choosing top $k$ relevant metrics: $SS = F_1(t), F_2(t), ..., F_k(t)$
4 Grouping all fuzzy time series into an unique multivariate $F(t)$: $\begin{bmatrix} F_1(1) \\ F_2(1) \\ \dots \\ F_M(1) \end{bmatrix}, \begin{bmatrix} F_1(2) \\ F_2(2) \\ \dots \\ F_M(2) \end{bmatrix}, \ldots, \begin{bmatrix} F_1(n) \\ F_2(n) \\ \dots \\ F_M(n) \end{bmatrix}$
5 Creating a stacked multiple LSTM layers, initializing a sliding window with $p$ consecutive points as the inputs
    $F(t), F(t-1), F(t-2), \ldots, F(t-p+1)$ and the next observation $F(t+k)$ as the outputs $(t = 1, 2, \ldots, n)$
6 Training the learning model with back propagation through time. The trained model will capture the logical
    fuzzy relationship: $F(t), F(t-1), F(t-2), \ldots, F(t-p+1)) \rightarrow F(t+k)$
7 Deploying the model to forecasting component.
8 Repeating step 1 to 6 for every time period $T$
```

Algorithm 3: Prediction

```
1 At each point of time $t$, collecting new resource consumption data $r_1(t), r_2(t), \ldots, r_i(t)$, fuzzifying them into
    $F_1(t), F_2(t), \ldots, F_i(t), \ldots, F_M(t)$. Calculating $F_1(t+k), F_2(t+k), \ldots, F_i(t+k), \ldots, F_M(t+k)$ by the trained
    model.
2 For each fuzzy forecast $F_1(t+k), F_2(t+k), \ldots, F_i(t+k), \ldots, F_M(t+k)$, applying centroid defuzzification
    method to obtain the final crisp result, which is resource consumption at the next point of time:
    $r_i^{pred}(t+1), i = 1, 2, \ldots, M$
3 Repeating from step 1 to 2 at the next point of time.
```

5. Experiments

5.1. Experimental setup

We use Google cluster trace dataset [8] to evaluate our proposed system. The dataset is a collection of many jobs and their resource consumptions submitted to Google cluster in 2011. Each job contains many tasks that are run simultaneously on multiple machines. The task resource utilizations have 20 metrics which are measured by several metrics such as CPU, memory usage, disk I/O mean time, and so forth. The data is recorded in over one billion and two hundred million data items (i.e. 1232799308). According to the analyses described in [8], only less than 2% of jobs that run longer than one day, even though such jobs contribute to over 80% of the dataset records. In addition, only jobs containing at least hundreds of tasks consume a significant resource amount. To achieve the generality in evaluating the efficiency of our model with the Google dataset, we select a long-running job with ID 6176858948 that consists of 25954362 monitoring data items during the 29-day period. We used dataset from 1st to 20th day to train our model and from 21st to 29th to evaluate the model performance.

We assume that required time to instantiate a new VM is 10 minutes. Therefore, the resource usages in the next 10 minutes must be predicted to enable cloud system makes scaling decisions in advance. In this way, τ is set to 10, forecast horizon $k = 1$ and adaptation length $l = 10$ in all our experiments. The forecast accuracy is assessed by mean absolute error (MAE) defined as follows: $MAE = \frac{1}{n}\sum_{i=1}^{n}\left|\widehat{y_i} - y_i\right|$, where $\widehat{y_i}$ is predicted value and y_i is real measured value. The fuzzy sets are selected based on the following formula [3]: $A_i = \frac{0}{u_1} + \frac{0}{u_2} + \ldots + \frac{0.5}{u_{i-1}} + \frac{1.0}{u_i} + \frac{0.5}{u_{i+1}} + \ldots + \frac{0}{u_n}$.

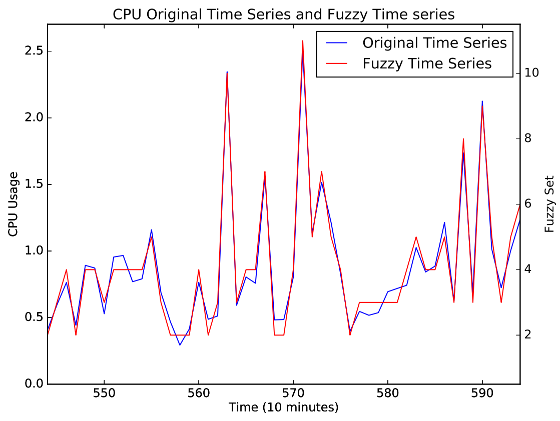

Fig. 4: Original Time series and Fuzzy Time series of CPU usage

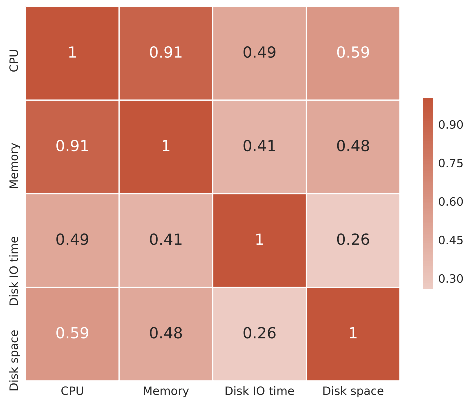

Fig. 5: Nonlinearity correlation

	$\delta = 0.8$	$\delta = 0.45$	$\delta = 0.3$
CPU	Disk io time	Disk io time	Disk io time
	Local disk space	Local disk space	Local disk space
Memory	Disk io time	Disk io time	Disk io time
	Local disk space	Local disk space	Local disk space

Table 1: Feature selection results

5.2. *Fuzzification*

We carry out the fuzzification test to show the effectiveness of fuzzy mechanism in preprocessing phase in order to reduce the data fluctuation. Figure 4 shows an example for the original time series of CPU utilization (without normalizing) comparing with corresponding fuzzy time series produced by fuzzifier. In several ranges such as (551, 554), (577, 581) and so on, the data fluctuation is reduced through transforming into straight line (these fluctuation values are transformed into a unique fuzzy set).

Because there are two phases including fuzzification and defuzzification, which are added into our system. The extra performance cost of transformation could be occurred, and the forecast accuracy may decrease in some case. However, there is also a worth note here: with respect to scaling decisions, which use our system, the extra cost of transformations will not affect significantly on efficiency of whole model. For example, in CPU time series, let a fuzzy set $A_2 = 0.0/u_1 + 1.0/u_2 + 0.0/u_3$, $u_1 = (0, 100\%), u_2 = (100\%, 200\%), u_3 = (200\%, 300\%)$. If prediction of fuzzy time series is A_2 and actual value also belongs to the fuzzy set A_2, in this way, both defuzzified and actual values will be in the same range (100%, 200%). Consequently, the fuzzy approach has the same scaling decision with the number of CPU core is 2. This phenomenon is like the other resource measurements like memory, Disk I/O, and so forth.

5.3. *Feature selection*

The goal of this test is to prove the operability of feature selection algorithm to choose the high correlative metrics for the next learning phase. In this direction, we reckon the correlation for four main metrics of Google cluster machines, including mean CPU usage, canonical memory usage, mean disk I/O time and mean local disk space using Algorithm 1.

The achieved correlation outcomes with δ threshold is expressed by Table 1. With each δ in the table, the selected metrics are filled by gray color. For example, with $\delta = 0.8$, we do not choose any metrics because the correlations of disk io time and local disk space metrics with CPU-memory pair are not satisfy the following condition at line 14 in

Metrics	CPU				Memory			
Models	p = 2	p = 3	p = 4	p = 5	p = 2	p = 3	p = 4	p = 5
Multivariate with $\delta = 0.8$	**0.3221**	0.3318	0.3383	0.3259	**0.0303**	0.0305	0.0309	0.0307
Multivariate with $\delta = 0.45$	0.3546	0.3745	0.3673	0.3866	0.0341	0.0339	0.0328	0.0361
Multivariate with $\delta = 0.3$	0.4244	0.4075	0.4058	0.4335	0.0377	0.0368	0.0423	0.0419
Univariate	0.3510	0.3316	0.3528	0.3278	0.0357	0.0346	0.0406	0.0362

Table 2: MAE prediction evaluation of CPU and memory

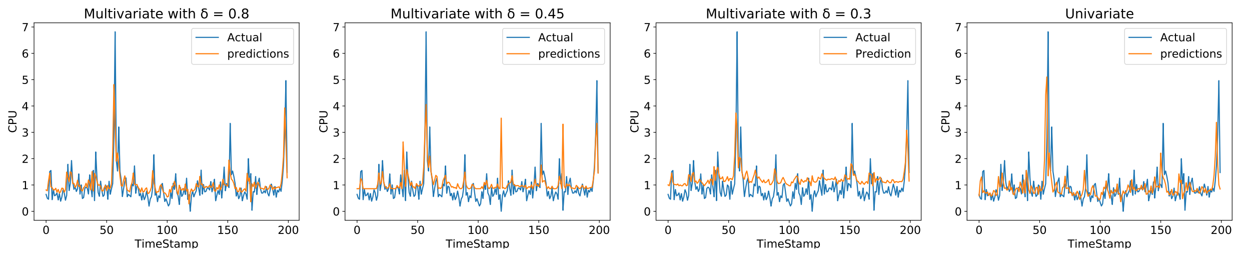

Fig. 6: CPU prediction with sliding size = 2

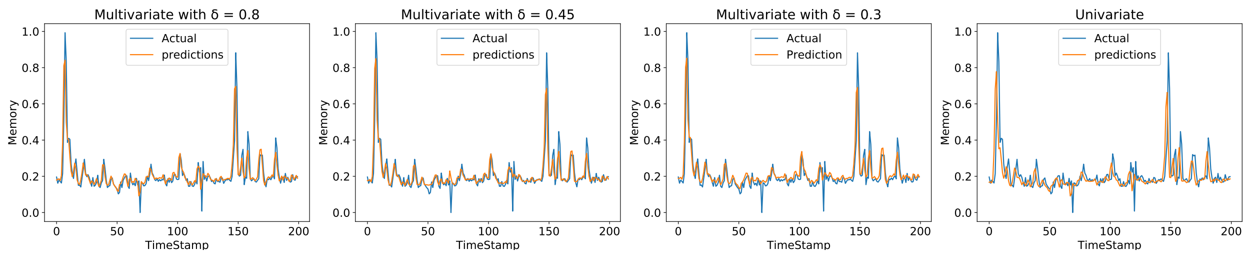

Fig. 7: Memory prediction with sliding size = 2

Algorithm 1. Similarly, with $\delta = 0.45$, we select only the local disk space and with $\delta = 0.3$, we pick all of four metrics for the training module.

5.4. *Forecasting*

This experiment focuses on comparing prediction accuracy between MF-LSTM and univariate learning model to demonstrate effectiveness of the proposed prediction model. To do that, for the univariate model, each metric is predicted separately using LSTM model and for the multivariate model, we use the selected metrics by difference thresholds δ in the feature selection phase. With the achieved forecast results of each metrics using both MF-LSTM and univariate model, we can compare the accuracy of them. We also test those models with diverse sliding windows values. The experimental results with MAE measurement are shown in Table 2.

Through the outcomes, there is no model, which can get the outstanding effect. However, it can be made a key observation here. In both CPU and memory prediction tests, MAE of the multivariate with $\delta = 0.8$ has a little better as compared with others. Specifically, the best MAE of CPU is 0.3221 for MF-LSTM model. Similarly, for memory usage, using MF-LSTM model, the best MAE gains 0.0303 with same δ value. The CPU and memory prediction outcomes using MF-LSTM with thresholds δ and univariate model are illustrated by Figure 6, and 7.

6. Conclusion and future work

In this research, we proposed a new prediction system for cloud proactive auto-scaling service focusing on exploiting multivariate monitoring data. This is practical requirement of cloud systems today. In this direction, we proposed several mechanisms to improve accuracy as well as effectiveness of the resource usage prediction model. To deal with fluctuation problem of monitored data, we employed fuzzification technique to collected time series data. We also addressed the need of analyzing correlations among different metrics based on the symmetrical uncertainty measure. In this way, we can select suitable data for our prediction model, which uses the long-short term memory (LSTM) neural network. This model is considered as an effective learning model for analyzing time series data type. We tested the proposed system with real Google dataset and the achieved results express the applicability of our solution in real cloud systems. In the near future, we plan evaluate the efficiency of different learning techniques for our prediction model. Besides, we will build a scaling module for the proposed system.

Acknowledgment

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 project 2/0167/16 "Methods and algorithms for the semantic processing of Big Data in distributed computing environment", and EU H2020-777435 DEEP-HybridDataCloud project "Designing and Enabling E-infrastructures for intensive Processing in a Hybrid DataCloud".

References

[1] Amazon CloudWatch-Cloud & Network Monitoring Services. https://aws.amazon.com/cloudwatch/. Accessed: 2018-04-13.

[2] Kanad Chakraborty, Kishan Mehrotra, Chilukuri K. Mohan, and Sanjay Ranka. Forecasting the behavior of multivariate time series using neural networks. *Neural Networks*, 5(6):961 – 970, 1992.

[3] 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.

[4] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[5] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[6] Jitendra Kumar, Rimsha Goomer, and Ashutosh Kumar Singh. Long short term memory recurrent neural network (lstm-rnn) based workload forecasting model for cloud datacenters. *Procedia Computer Science*, 125:676 – 682, 2018. The 6th International Conference on Smart Computing and Communications.

[7] B Nguyen Minh, Viet Tran, and Ladislav Hluchy. Abstraction layer for development and deployment of cloud services. *Computer Science*, 13(3)):79–88, 2012.

[8] 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.

[9] Qiang Song and Brad S. Chissom. Fuzzy time series and its model. 54:269–277, 03 1993.

[10] D. Tran, N. Tran, B. M. Nguyen, and H. Le. Pd-gabp 2014; a novel prediction model applying for elastic applications in distributed environment. In *2016 3rd National Foundation for Science and Technology Development Conference on Information and Computer Science (NICS)*, pages 240–245, Sept 2016.

[11] Dang Tran, Nhuan Tran, Giang Nguyen, and Binh Minh Nguyen. A proactive cloud scaling model based on fuzzy time series and sla awareness. *Procedia Computer Science*, 108:365 – 374, 2017. International Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland.

[12] A. Tucker, S. Swift, and X. Liu. Variable grouping in multivariate time series via correlation. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 31(2):235–245, Apr 2001.

[13] 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.

[14] Jorge R. Vergara and Pablo A. Estévez. A review of feature selection methods based on mutual information. *Neural Computing and Applications*, 24(1):175–186, Jan 2014.

[15] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[16] Lei Yu and Huan Liu. Efficient feature selection via analysis of relevance and redundancy. *J. Mach. Learn. Res.*, 5:1205–1224, December 2004.