A Fuzzy Virtual MachineWorkload Prediction Method for Cloud Environments

Fahimeh Ramezani
Centre for Artificial Intelligence
School of Software, Faculty of Engineering and IT
University of Technology Sydney
PO Box 123, Broadway, NSW 2007, Australia
Fahimeh.Ramezani@uts.edu.au

Mohsen Naderpour
Centre for Artificial Intelligence
School of Systems, Management and Leadership, Faculty of Engineering and IT, University of Technology Sydney
PO Box 123, Broadway, NSW 2007, Australia
Mohsen.Naderpour@uts.edu.au

***Abstract*—Due to the dynamic nature of cloud environments, the workload of virtual machines (VMs) fluctuates leading to imbalanced loads and utilization of virtual and physical cloud resources. It is, therefore, essential that cloud providers accurately forecast VM performance and resource utilization so they can appropriately manage their assets to deliver better quality cloud services on demand. Current workload and resource prediction methods forecast the workload or CPU utilization pattern of the given web-based applications based on their historical data. This gives cloud providers an indication of the required number of resources (VMs or CPUs) for these applications to optimize resource allocation for software as a service (SaaS) or platform as a service (PaaS), reducing their service costs. However, historical data cannot be used as the only data source for VM workload predictions as it may not be available in every situation. Nor can historical data provide information about sudden and unexpected peaks in user demand. To solve these issues, we have developed a fuzzy workload prediction method that monitors both historical and current VM CPU utilization and workload to predict VMs that are likely to be performing poorly. This model can also predict the utilization of physical machine (PM) resources for virtual resource discovery.**

Keywords—Cloud Computing; Virtual Machine; Fuzzy Systems.

I. INTRODUCTION

Cloud computing delivers scalable on-demand services over the Internet, including SaaS, PaaS, and infrastructure as a service (IaaS). A cloud provides these services through virtualized resources that are overlaid on physical resources, called virtualized cloud resources. Typically, a virtualized cloud resource is a set of specification and configuration files, called a VM [1, 2]. SaaS and PaaS providers are charged by IaaS providers on an hourly basis. They, therefore, need to determine the optimal number of VMs required in each IaaS cluster to provide their services, given a predicted workload, and guarantee service level agreement constraints at the same time. IaaS providers also need an estimation of the required capacity of VMs in their cloud clusters for optimal on-demand resource provisioning [3].

Nagothu et al. [4] proposed a method for load prediction by separating it into linear and non-linear algorithms. They believed that a linear prediction algorithm could either involve 1-Dim observation sequences or d-Dim observation space signals [5]. They also proposed an alternative method for load prediction that makes use of Burg's algorithm [6]. Saripalli et al. [4] demonstrated the use of load prediction algorithms for cloud platforms using a two-step approach, i.e., load trend tracking followed by load prediction. Their approach uses cubic spline interpolation and a hotspot detection algorithm for sudden spikes. Meng et al. [7] proposed an approach called joint-VM provisioning, which estimates the aggregate size of multiplexed VMs, then resource allocation (CPU and memory) is considered for a determined set of compatible VMs instead of allocating resources to an individual VM. This helps scaling for high-utilization VMs by using the spare resources of a co-located low-use VM. They also developed a workload prediction model to estimate the capacity required for a determined set of VMs by decoupling the VM workload into regular and irregular fluctuating components. The regular workload refers to deterministic patterns, such as trends, cycles, and seasonality. The irregular fluctuating workload is the residue after the regular workload has been removed. To forecast the regular workload, they simply assume that the regular patterns will be preserved in the future, i.e., that a steadily increasing trend will continue to increase at the same rate and that daily seasonality will continue to hold. To forecast an irregular workload, they performed a time series forecasting technique based on historic workload patterns. Yang et al. [8] developed a pattern fusion model for predicting multi-step-ahead CPU loads by categorizing historical CPU load-time series patterns into two sets: patterns that rarely occur and have the lowest possibility of occurrence, and patterns with almost similar trends that have the highest degree of likely occurrence. Islam et al. [9] proposed a prediction-based resource measurement method that predicts the CPU utilization pattern for a given application by applying two learning algorithms: an error correction neural network [10] and linear regression [11]. In addition, they used sliding windows [12] and cross-validation [13, 14] techniques in the training and prediction stages. Ardagna et al. [3] also developed a prediction-based resource measurement approach by applying an exponential smoothing prediction method. They believe this method is appropriate for predicting run-time and non-stationary behavior. They also performed dynamic load redirection [15, 16] periodically to predict short-term changes in the number of workload arrivals.

In short, most researchers have applied prediction methods, such as neural networks, pattern recognition and linear regression to estimate VM workloads [17]. These methods predict the future workload of VMs by applying previous workload patterns in time slot *t*, determined on the basis of related historical data [8]. And, most are designed for SaaS and PaaS resource prediction or VM remapping or in situations where cloud providers are aware of the types of applications and software being executed and can trace their behavior. However, for IaaS (where IaaS is not delivered to PaaS and SaaS providers), there is no information about upcoming executing applications on each VM. In these cases, application behavior is not applicable when estimating VM workloads and CPU usage. In addition, a VM's workload is affected by user behavior and the sudden decisions they make, so fluctuations in the workload could be independent of previous workloads and CPU load patterns. In other words, they could change dramatically on the basis of dynamic, unpredictable, and fluctuating resource user demand. Fig. 1 shows one example of this. Considering these facts, we propose a fuzzy workload prediction method that applies both historical and current VM CPU utilization and workload to predict probable poorly performing VMs.

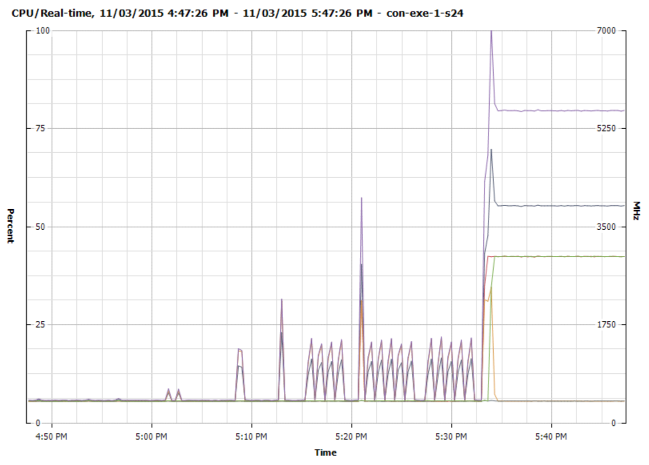


Fig. 1. CPU usage trend of a VM in a cloud cluster.

The rest of this paper is organized as follows. Section II provides a brief description of fuzzy logic and fuzzy logic systems. Section III presents the fuzzy workload prediction method. In Section IV, the proposed method is implemented in a cloud cluster, and Section V presents the conclusion and future works.

II. Fuzzy Logic Systems

Fuzzy logic mathematically emulates human reasoning and provides an intuitive way of designing function blocks for intelligent systems. It allows humans to express their knowledge in the form of related, but imprecise, inputs and outputs as linguistic variables, which simplifies knowledge acquisition and representation. The knowledge obtained is easy to understand and modify. In this regard, a fuzzy logic system (FLS) is technology that takes expert knowledge about a particular system into account when designing intelligent systems. Generally, an FLS, as shown in Fig. 2, includes three parts: fuzzification, a fuzzy inference engine, and defuzzification. In the fuzzification process, fuzzy sets are formed for all input variables. The fuzzy inference engine takes the input variables into account along with the logic relations between them. Fuzzy logic operations are used to generate the output. In the defuzzification process, the output fuzzy set is converted into a crisp value [18].

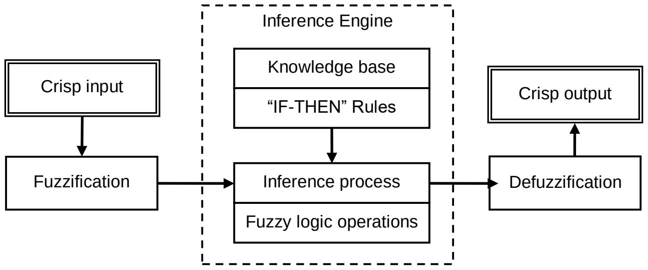


Fig. 2. A fuzzy logic system.

III. The Fuzzy Workload Prediction Method

Our fuzzy workload prediction (FWP) method not only applies neural networks (NN) to predict VM CPU usage patterns using historical data but also applies an FLS to control near future changes in CPU workload. It operates on every VM in use to deliver SaaS, PaaS, or IaaS. Based on FWP's results, the method is able to determine which VM's are performing poorly and, consequently, it can predict PM hotspots. To design the FWP, we first determined the conditions under which a VM might become overloaded in the near future. The corresponding input/output variables for the FWP algorithm are then defined on the basis of the determined conditions. These conditions and variables are taken into consideration to develop the FLS rules.

A. *The Poorly Performing VM: Conditions and Variables*

Given CPU utilization fluctuates with the potential for very sudden increases and decreases within a short period of time, checking CPU usage at regular intervals, as applied in [19], is not a reliable means of estimating a VM's upcoming CPU workload. Therefore, we monitor CPU usage trends and fluctuations over a small period of time (e.g., every two minutes) to forecast the VM's workload level for the next interval. Four definitions are used to explain a poorly performing VM based on its CPU workload. Each is detailed below, where *ct* is the current time, *t* is a given period of time (e.g., two minutes), and time slot $Ts = \{ct - t, ct\}$ in seconds:

Definition 1: If VM^k_{Ucpu} is the total amount of CPU utilization of VM_k, this VM will be overloaded if:

$$\lim_{s \to ct} VM^k_{Ucpu}(s) \geq 80\% \,, s \in Ts \tag{1}$$

Virtual CPUs determine how many physical cores can be used by a VM. The number of virtual CPUs, together with the scheduler credit, determine the total CPU resource allocated to a VM [20]. Based on this definition, VM_k has the potential to be overloaded under the following conditions:

Condition 1.1: The CPUs allocated to VM_k remain busy during the last minutes of time slot Ts.

To check this condition, the current value of CPU utilization by VM_k, i.e., $VM^k_{Ucpu}(ct)$, should be determined. If $VM^k_{Ucpu}(ct) \geq 80\%$, we calculate the average value of

VM^k_{Ucpu} in the last part of time slot Ts $(i.e. lt)$. Based on these assumptions, VM_k will probably be overloaded if:

$$VM^k_{Ucpu}(ct) \geq 80\% \; \& \qquad \underset{s_i \in lt}{\text{Avg}}\, VM^k_{Ucpu}(s_i) \geq 80\%, lt \subset Ts \tag{2}$$

<u>Condition 1.2:</u> The cumulative average of the CPU usage of VM_k (calculated every 20 seconds) has an increasing trend.

CPU usage usually fluctuates dramatically, and it is difficult to estimate its overall increasing or decreasing trend. Therefore, the cumulative average of CPU usage, as presented in [21], is used to estimate the trend during time slot Ts as follows:

$$CA^k_{cpu}(x) = \sum_{i=1}^{x*20} \frac{VM^k_{Ucpu}(s_i)}{x*20}, x \in \left\{1,2,3,..,div(\frac{Ts}{20})\right\} \tag{3}$$

where div is the integer division. The polynomial fitting tool in MATLAB is then applied to determine the overall trend of $CA^k_{cpu}(x)$. $CA^k_{cpu}(x)$ has an increasing trend if the derivative of its fitted line ($fCA^k_{cpu}(x)$) is positive in Ts:

$$fCA'^k_{cpu}(x) \geq 0\, , x \in \left\{1,2,3,..,div(\frac{Ts}{20})\right\} \tag{4}$$

<u>Fuzzy Variable 1:</u> We assume $VM^k_{Huti}(Ts)$ is a variable that is defined based on VM_k utilization to control whether or not Conditions 1.1 and 1.2 are satisfied:

$$VM^k_{Huti}(Ts) = \begin{cases} H & [VM^k_{Ucpu}(ct) \geq 80\%]\ and \\ & \left[\underset{s_i \in lt}{Avg}\, VM^k_{Ucpu}(s_i) \geq 80\%\right]\ and \\ & \left[fCA'^k_{cpu}(x) \geq 0\right] \\ L & otherwise \end{cases} \tag{5}$$

<u>Definition 2:</u> If $VM^k_{et}(s)$ are the number of tasks scheduled for VM_k , as a time series in time slot Ts, then VM_k will be overloaded if the time series $VM^k_{et}(s)$ has an increasing trend.

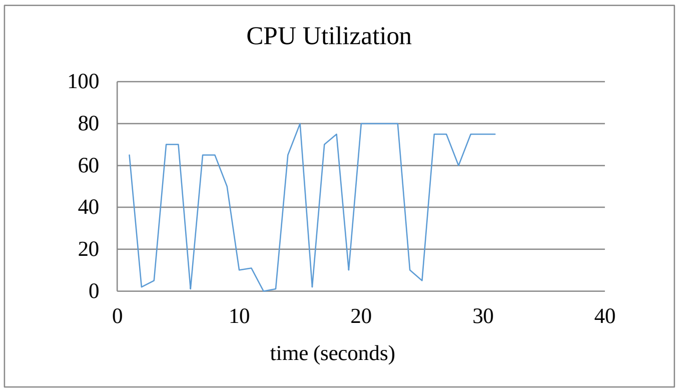
<u>Condition 2.1:</u> The cumulative average of time series $VM^k_{et}(s)$ – which is the number of tasks executed by the VM_k at time slot S – shows that the overall trend of $VM^k_{et}(s)$ has an increasing trend.

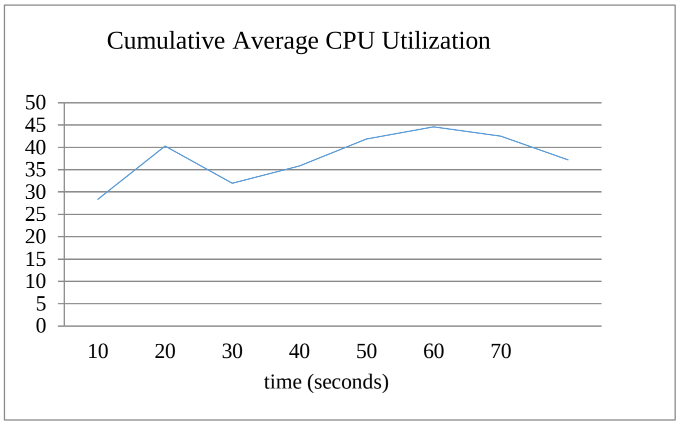
The cumulative average of $VM^k_{et}(s)$ is calculated every 20 seconds as follows:

$$CA^k_{et}(x) = \sum_{i=1}^{x*20} \frac{VM^k_{et}(s_i)}{x*20}, x \in \left\{1,2,3,..,div(\frac{Ts}{20})\right\} \tag{6}$$

where div is the integer division. $CA^k_{et}(x)$ has an increasing trend if its polynomial fitted line has a positive derivative in Ts , i.e.,

$$fCA'^k_{et}(x) \geq 0\, , x \in \left\{1,2,3,..,div(\frac{Ts}{20})\right\} \tag{7}$$





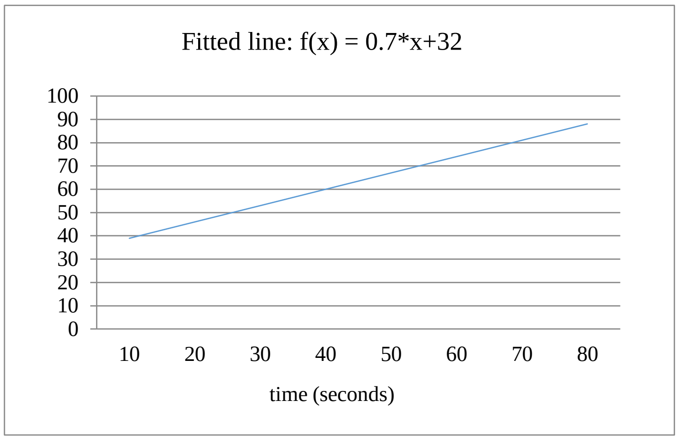


Fig. 3. Estimating CPU utilization trend in time slot Ts [21].

<u>Fuzzy Variable 2:</u> Variable $VM^k_{RisEt}(Ts)$ is defined based on Condition 2.1 to show whether or not the status of $VM^k_{et}(s)$ will lead to VM_k being overloaded:

$$VM^k_{RisEt}(Ts) = \begin{cases} H & fCA'^k_{et}(x) \geq 0 \\ L & \text{Otherwise} \end{cases} \tag{8}$$

<u>Definition 3:</u> Rao [20] proposed a metric called the productivity index (PI) to measure system processing capability. He defined PI as

$$PI^k\,(s) = \frac{CW_k(s)}{CC_k\,(s)}, s \subset Ts \tag{9}$$

where $CW_k(s_i)$ is the number of completed tasks and $CC_k(s)$ is the amount of resources consumed (CPU utilization) during the time slot Ts by VM_k. According to Rao's definition [20], VM_k will become overloaded if PI begins to drop, although Rao believes that for online identification, a single PI metric is not enough to identify the system state because any change in PI could be due to either system capacity or changes in the input load. Condition 3.1 is suggested to determine the PI trend and monitor a VM’s workload during *Ts* with a constant amount of resources.

Condition 3.1: The time series $PI^k(s)$ has a decreasing trend.

Fuzzy Variable 3: Variable $PI^k_{Dec}(Ts)$ is defined to indicate the increasing or decreasing trend of $PI^k(s)$ during *Ts* as follows:

$$PI^k_{Dec}(Ts) = \begin{cases} H & fPI'^k(s) \leq 0,\ s \in Ts \\ L & \text{Otherwise} \end{cases} \tag{10}$$

where $fPI^k(s)$ is the polynomial fitted line of $PI^k(s)$.

Definition 4: The workload status of VM_k is estimated based on the CPU usage prediction results of a designed NN.

In this approach, a three-layer NN is designed and then trained, based on historical data about the CPU usage of VM_k, to predict its CPU usage pattern. It has an input layer with three neurons $I = \{VM^k_{Ucpu}, VMm_k,\ VMc_k\}$, and a hidden layer with two hidden neurons $H = \{h_1, h_2\}$ inbetween. The output layer of the NN ($O = NN^k_R(Ts)$) has one neuron that represents its prediction results for the upcoming CPU usage by VM_k. The activation functions that are applied in the designed NN are sigmoidal in the hidden layer and linear in the output.

Condition 4.1: The prediction result of the designed NN indicates that VM_k will be overloaded.

Fuzzy Variable 4: Variable $NN^k_R(Ts)$ is defined to show the NN prediction results as:

$$NN^k_R\ (Ts) = \begin{cases} O & VM_k \text{is overloaded} \\ U & VM_k \text{is under} - \text{loaded} \end{cases} \tag{11}$$

However, none of these conditions by themselves indicate risk unless they happen when some of other conditions are also satisfied. Therefore, a combination of the fuzzy variables 1-4 are applied by FLS as inputs to estimate the value of the output variable, which is defined as follows:

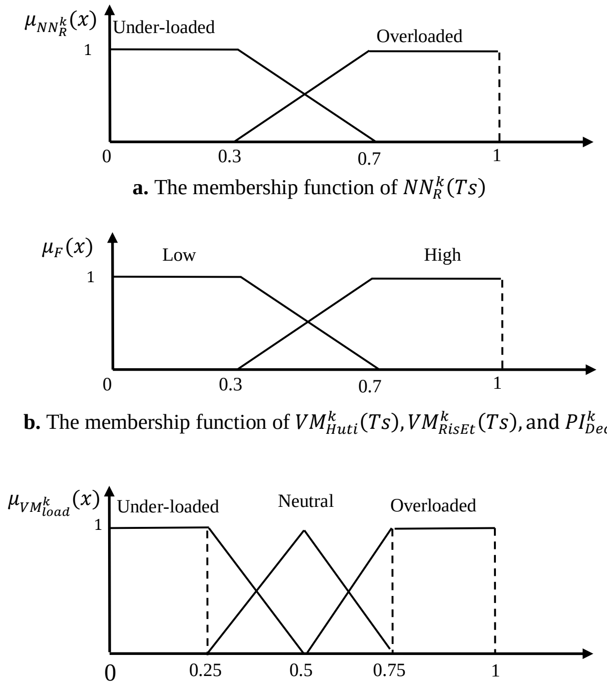
Output Fuzzy Variable: The variable VM^k_{load} is defined to show the predicted VM_kworkload situation by FLS as:

$$VM^k_{load}\ (Ts) = \begin{cases} O & VM_k \text{ is likely to be overloaded} \\ N & \text{Neutral condition} \\ U & VM_k \text{is likely to be under} - \text{loaded} \end{cases} \tag{12}$$

In some situations, the FLS answer is neutral and does not suggest an obvious result. In such cases, the workload situation of VM_k will be forecast in the next prediction round. The defined variables are summarized in Table I, and the membership functions of these fuzzy variables are illustrated in Fig. 4.

TABLE I. THE FWP VARIABLES

Symbol	Definition
$CW_k(Ts)$	The amount of work completed during the time slot *Ts* by VM_k
$CC_k(Ts)$	The amount of resource (CPU) consumed during the time slot *Ts* by VM_k
VM^k_{load}	VM_k workload situation
VM^k_{Ucpu}	The amount of CPU utilization by VM_k
VM^k_m	The amount of available memory of VM_k
VM^k_c	The number of CPUs allocated to VM_k
$CA^k_{cpu}(x)$	The cumulative average of time series $VM^k_{Ucpu}(s_i)$
$fCA'^k_{cpu}(x)$	The derivative of the fitted line to $CA^k_{cpu}(x)$
$VM^k_{et}(s_i)$	The number of executing tasks in the VM_k at time s_i
$CA^k_{et}(x)$	The cumulative average of time series $VM^k_{et}(s_i)$
$fCA'^k_{et}(x)$	The derivative of the polynomial fitted line to $CA^k_{et}(x)$
$VM^k_{RisEt}(Ts)$	Shows VM_k’s workload has had an increasing or decreasing trend during *Ts*
$VM^k_{Huti}(Ts)$	Shows CPU utilization of VM_k has had an increasing or decreasing trend during *Ts*
$PI^k(s)$	The PI for VM_k during *Ts*
$PI^k_{Dec}(Ts)$	Shows $PI^k(s)$ has had an increasing or decreasing trend during *Ts*
$NN^k_R(Ts)$	The neural network forecasting results for CPU usage by VM_k during *Ts*



a. The membership function of $NN^k_R(Ts)$

b. The membership function of $VM^k_{Huti}(Ts), VM^k_{RisEt}(Ts)$, and $PI^k_{Dec}(Ts)$

c. The membership function of $VM^k_{load}(Ts)$

Fig. 4. The Membership Function of Input and Output Variables.

B. *Poorly Performing VMs: Rules*

The FLS rules were determined based on expert knowledge and the aforementioned conditions. Variables related to monitoring VM CPU usage and workload changes are used to predict which VM is likely to highly utilize its allocated resources and become overloaded. Forty-eight (16*3) rules can be extrapolated from the different combinations of four input variables ($VM^k_{Huti}(Ts)$, $VM^k_{RisEt}(Ts)$, $PI^k_{Dec}(Ts)$ and $NN^k_R(Ts)$) and the output variable $VM^k_{load}\,(Ts)$. In this paper, we can control the system using eleven rules, as summarized in Table II.

TABLE II. FLS RULES

Based on historical data		Based on actual data in the past specified minutes						VM workload situation
		D1		D2		D3		
$NN^k_R(Ts)$		$VM^k_{Huti}(Ts)$		$VM^k_{RisEt}(Ts)$		$PI^k_{Dec}(Ts)$		VM^k_{load}
O	and	H	and	H	and	H	then	Overloaded
O	and	H	and	L	and	H	then	Overloaded
O	and	H	and	H	and	L	then	Overloaded
O	and	L	and	H	and	H	then	Overloaded
U	and	H	and	H	and	H	then	Overloaded
U	and	H	and	H	and	L	then	Overloaded
U	and	L	and	H	and	L	then	Neutral
U	and	H	and	L	and	L	then	Neutral
U	and	L	and	L	and	H	then	Under-loaded
O	and	L	and	L	and	L	then	Under-loaded
U	and	L	and	L	and	L	then	Under-loaded

C. *The FWP Algorithm*

The FWP algorithm responsible for predicting VMs with over- and under-utilization based on their CPU usage and workload status is summarized as follows. This algorithm plays a key role in cloud resource management and determines the origin and destination VMs for the extra workload in a cloud cluster.

FWP Algorithm

```
Input: All variables in Table I.
[Begin]
1.  Monitor cloud blackboard data to calculate the value of
    following variables:
      ➢ $VM^k_{Ucpu}(ct)$ and $VM^k_{Ucpu}(s_r)$
      ➢ $CA^k_{cpu}(x)$ and $fCA'^k_{cpu}(x)$
      ➢ $VM^k_{Huti}(Ts)$
      ➢ $CA^k_{et}(x)$ and $fCA'^k_{et}(x)$
      ➢ $VM^k_{RisEt}(Ts)$
      ➢ $PI^k_{Dec}(Ts)$ and $fPI'^k(s)$
2.  Calculate NN results about VM workload situations
    ($NN^k_R(Ts)$)
3.  Determine the value of $VM_{load}(Ts)$ for each VM based on
    FLS rules, then forecast the VMs' workload.
[End]
```

IV. IMPLEMENTATION RESULTS

A. *Environment Description*

The cloud environment used for implementation comprised: two data-stores; four PMs; 20 VMs; and 200 independent arrival computation, memory, and data-intensive tasks. Information about the VMs and the tasks is summarized in Tables III and IV, respectively. The PMs were homogenous and each had five different VMs (see Table V).

TABLE III. PROPERTIES OF VMS

VM Id	CPU speed in GHz	Available memory in MB	Bandwidth in Mb/s	Number of CPUs
1-4	2.6	4096	1024	4
5-8	2.6	4096	1024	2
9-12	1.3	2048	1024	2
13-16	1.3	1024	1024	1
17-20	1.3	512	1024	1

B. *Results*

The tasks listed in Table IV were randomly allocated to the VMs, and the FWP algorithm was implemented on every VM to check its performance every two minutes. According to the FWP algorithm results, all VMs were under-loaded in the first round. In the second round (after four minutes), the FWP algorithm results show that VM_{17}, located on PM_1, would become overloaded, based on the value of its input variables. These variables were calculated as follows, where $Ts = (2{,}4)$:

$$VM^{17}_{Ucpu}(ct) = 85\%\,,\ \ ct = 4$$
$$\operatorname*{Avg}_{s_i\in lt} VM^{17}_{Ucpu}(s_i) \geq 81\%, \qquad lt = (3{,}4)$$
$$fCA'^{17}_{cpu}(x)\ \geq 0$$

By applying the values in Eq. (5), the value of $VM^k_{Huti}(Ts)$ indicates that VM_{17} had a high CPU utilization level. In addition, $fCA'^{17}_{et}(x)\ \geq 0$, which means that VM_{17} had a rising workload pattern and the value of $VM^{17}_{RisEt}(Ts)$ was high (see Equation 8). Furthermore, $PI^{17}(Ts)$ showed a decreasing trend during *Ts*, therefore the value of $PI^{17}_{Dec}(Ts)$ was also high (see Equation 10). The designed NN prediction results ($NN^{17}_R(Ts)$) also suggest that this VM had a high level of CPU usage during time slot *Ts* and would become overloaded, thus exhibiting low performance (see Equation 11).

TABLE IV. PROPERTIES OF TASKS

Task Id	Required CPUs	CPU usage in GHz	Total memory usage in MB	Max level of memory usage in MB
Computing Intensive Tasks				
1-20	1	186.372	2055.06	125.828
21-40	1	21.186	985.616	62.912
41-60	1	67.166	754.924	62.912
61-80	1	8.261	471.848	73.4
81-100	1	21.759	524.26	62.912
101-120	1	2.263	125.82	41.94
Memory Intensive Tasks				
121-140	1	41.097	11534.304	943.716
141-160	1	38.367	9940.468	859.832

161-180	1	38.372	1992.268	167.772
Data Intensive Tasks				
181-200	1	2.833	377.484	125.828

These results satisfy the first rule of the developed FLS (see Table II), and therefore suggest VM_{17} as the poorly performing VM (see Equation 12). At the time, VM_{17} was executing t_5 while it had $T_{set}^{17} = \{t_7, t_{21}, t_{45}, t_{66}, t_{181}, t_{119}, t_{126}, t_{158}, t_{183}, t_{198}\}$ in its task queue. The FLS also nominated $VM_2, VM_3, VM_5, VM_8, VM_9$ and VM_{15} as VMs that were under-loaded as illustrated in Table VI.

TABLE V. RESOURCE ALLOCATION IN THE CLUSTER

PM Id	VMs				
1	VM_1	VM_5	VM_9	VM_{13}	VM_{17}
2	VM_2	VM_6	VM_{10}	VM_{14}	VM_{18}
3	VM_3	VM_7	VM_{11}	VM_{15}	VM_{19}
4	VM_4	VM_8	VM_{12}	VM_{16}	VM_{20}

TABLE VI. THE FWP RESULTS

VM	$NN_R^k(Ts)$		$VM_{Huti}^k(Ts)$		$VM_{RisEt}^k(Ts)$		$PI_{Dec}^k(Ts)$		VM_{load}^k
17	O	and	H	and	H	and	H	then	Overloaded
2	U	and	L	and	L	and	L	then	Under-loaded
3	U	and	L	and	L	and	L	then	Under-loaded
5	U	and	L	and	L	and	H	then	Under-loaded
8	U	and	L	and	L	and	H	then	Under-loaded
9	O	and	L	and	L	and	L	then	Under-loaded
15	U	and	L	and	L	and	L	then	Under-loaded

V. CONCLUSION AND FUTURE WORKS

Prediction methods, such as neural networks and linear regression, have been widely applied in previous works to forecast the resource utilization of VMs in cloud environments and estimate required resources and size for each VM. However, these prediction methods rely on related historical data in a time slot *t* to predict future resource utilization. Considering the fact that a VMs' future workload and resource utilization patterns could be independent of their previous patterns, the proposed prediction method in this study not only uses a neural network to predict resource VMs utilization patterns but also considers other variables that control VM performance. The method is based on the resource utilization required to execute the VM's allocated workload and applies a neural network along with defined fuzzy variables in an FLS to control near future changes in resource utilization for every VM. Through this approach, the time that a set of VMs are expected to begin performing poorly can be predicted. This method can also be used for physical and virtual resource discovery and predicting hot spots in PMs.

A simulated implementation environment was used to assess various factors in the method, such as the number of tasks executed by each VM. We intend to implement our method in a real life environment in future work. In addition, the performance of the proposed method will be compared with current technologies.

REFERENCES

[1] Celesti, A., Fazio, M., Villari, M., and Puliafito, A.: 'Virtual machine provisioning through satellite communications in federated cloud environments', Future Generation Computer Systems, 2012, 28, (1), pp. 85-93.

[2] Buyya, Rajkumar, James Broberg, and Andrzej M. Goscinski, eds. Cloud computing: Principles and paradigms. Vol. 87. John Wiley & Sons, 2010.

[3] Ardagna, D., Casolari, S., Colajanni, M., and Panicucci, B.: 'Dual time-scale distributed capacity allocation and load redirect algorithms for cloud systems', Journal of Parallel and Distributed Computing, 2012, 72, (6), pp. 796-808.

[4] Saripalli, P., Kiran, G., Shankar, R.R., Narware, H., and Bindal, N.: 'Load prediction and hot spot detection models for autonomic cloud computing', in 'Load prediction and hot spot detection models for autonomic cloud computing' (IEEE, 2011, edn.), pp. 397-402.

[5] Nagothu, K.M., Kelley, B., Prevost, J., and Jamshidi, M.: 'Ultra low energy cloud computing using adaptive load prediction', in 'Ultra low energy cloud computing using adaptive load prediction' (IEEE, 2010, edn.), pp. 1-7.

[6] Nagothu, K., Kelley, B., Prevost, J., and Jamshidi, M.: 'On prediction to dynamically assign heterogeneous microprocessors to the minimum joint power state to achieve ultra low power cloud computing', in 'On prediction to dynamically assign heterogeneous microprocessors to the minimum joint power state to achieve ultra low power cloud computing' (IEEE, 2010, edn.), pp. 1269-1273.

[7] Meng, X., Isci, C., Kephart, J., Zhang, L., Bouillet, E., and Pendarakis, D.: 'Efficient resource provisioning in compute clouds via VM multiplexing', in 'Efficient resource provisioning in compute clouds via VM multiplexing' (ACM, 2010, edn.), pp. 11-20.

[8] Yang, D., Cao, J., Fu, J., Wang, J., and Guo, J.: 'A pattern fusion model for multi-step-ahead CPU load prediction', Journal of Systems and Software, 2013, 86, (5), pp. 1257-1266.

[9] Islam, S., Keung, J., Lee, K., and Liu, A.: 'Empirical prediction models for adaptive resource provisioning in the cloud', Future Generation Computer Systems, 2012, 28, pp. 155–162.

[10] Theodoridis, S., and Koutroumbas, K.: 'Pattern Recognition' (Academic Press, 2008. 2008).

[11] Weisberg, S.: 'Applied Linear Regression': 'Wiley Series in Probability and Statistics' (John Wiley & Sons, 2005).

[12] Dietterich, T.: 'Machine learning for sequential data: A review, in 'Machine learning for sequential data: A review, in: T. Caelli, A. Amin, R. Duin, D. de Ridder, M. Kamel (Eds.), Structural, Syntactic, and Statistical Pattern Recognition' (Springer, 2002, edn.), pp. 227–246.

[13] Arlot, S., and Celisse, A.: 'A survey of cross-validation procedures for model selection', Statistics Surveys, 2010, 4, pp. 40–79.

[14] Efron, B., and Gong, G.: 'A leisurely look at the bootstrap, the jackknife, and crossvalidation', The American Statistician, 1983, 37, pp. 36–48.

[15] Zhu, X., Young, D., Watson, B.J., Wang, Z., Rolia, J., Singhal, S., Mckee, B., Hyser, C., Gmach, D., and Gardner, R.: '1000 islands: An integrated approach to resource management for virtualized data centers', Cluster Computing, 2009, 12, (1), pp. 45-57.

[16] Ardagna, D., Panicucci, B., Trubian, M., and Li, Z.: 'Energy-aware autonomic resource allocation in multitier virtualized environments', IEEE Transactions on Services Computing, 2012, 5, (1), pp. 2-19.

[17] Weingärtner, R., Bräscher, G.B., and Westphall, C.B.: 'Cloud resource management: A survey on forecasting and profiling models', Journal of Network and Computer Applications, 2015, 47, pp. 99-106.

[18] Naderpour, M., Lu, J., and Zhang, G.: 'An intelligent situation awareness support system for safety-critical environments', Decision Support Systems, 2014, 59, pp. 325-340.

[19] Islam, S., Keung, J., Lee, K., and Liu, A.: 'Empirical prediction models for adaptive resource provisioning in the cloud', Future Generation Computer Systems, 2012, 28, (1), pp. 155-162.

[20] Rao, J.: 'Autonomic management of virtualized resources in cloud computing', Wayne State University, 2011.

[21] Ramezani, F., Lu, J., Taheri, J., and Zomaya, A.Y.: ‘A Multi-Objective Load Balancing System for Cloud Environments’, The Computer Journal, 2017, in press.