Robust approach for host-overload detection based on dynamic safety parameter

Imene El-Taani, Mohand-Cherif Boukala, Samia Bouzefrane, Anissa Imen Amrous

Robust approach for host-overload detection based on dynamic safety parameter

Imene El-Taani
MOVEP Lab
USTHB
Algiers, Algeria
eltaani.imene@usthb.dz

Mohand-Cherif Boukala
MOVEP Lab
USTHB
Algiers, Algeria
mboukala@usthb.dz

Samia Bouzefrane
CEDRIC Lab
Cnam
Paris, France
samia.bouzefrane@cnam.fr

Anissa Imen AMROUS
USTHB
Algiers, Algeria
aamrous@usthb.dz

***Abstract*—Host-overloading detection is an important phase in the dynamic Virtual Machines (VMs) consolidation process. Using machine learning to predict the future workload on a host, is a very promising technique to avoid the overload host situation. In this work, we propose a novel approach for overloaded hosts detection, based on neural network and Markov model. The neural network is trained on a workload data set composed of VMs CPU-utilization history. The trained model is then used to predict the future usage for a given Physical Machine(PM), by summing up the predicted utilization of all its VMs. The confidence of this prediction is measured through a dynamic safety parameter, based on Markov model. The obtained results show that our approach outperforms the state of the art algorithms such as: MAD, IQR and LRR.**

***Index Terms*—VM consolidation, host-overload detection, neural network prediction, Markov model, energy-efficiency, SLA violation.**

I. Introduction

Finding efficient energy-aware strategies to manage resources on data centers become an important challenge for cloud providers. The energy inefficiency is due mainly to poor management of servers' resources. It has been proven [1] that a server that runs at 10% of CPU usage, consumes more than 50% of the maximum of its power. To deal with this problem, the virtualization technology is proposed to improve resource utilization, by allowing a PM (Physical Machine) to be shared by multiple VMs (Virtual Machines).

Dynamic VM consolidation is an effective approach to improve energy efficiency while assuring an acceptable level of QoS (Quality of Service) based on SLA (Service Level Agreement). The dynamic VM consolidation process can be divided into four sub-problems:

- Overloaded-host detection: The VMs demands, in this case, exceed the available resources on the hosts. To avoid such situation, some VMs must be migrated to other hosts.
- Underloaded-host detection: The host is considered as underloaded if its resource usage falls below some underload threshold. To address this problem, all VMs from uderloaded hosts must be migrated, and then the hosts are switched off to save energy.
- VMs selection: Once an overloaded host is detected, some of its VMs must be selected for migration. This migration will cost an extra consumption of CPU and network resources from source and destination hosts. So, the selection process is crucial for resource management. Different parameters can be considered to select the migrated VMs [1]. One of the most utilized policy is MMT (Minimum Migration Time) which selects the VM that requires the minimum migration time between all the host's VMs. The migration time is calculated as a ratio of the memory assigned to the VM by the available network bandwidth.
- VMs placement : The goal is to find a better allocation mapping VMs-to-PMs based on the current resources of the selected VMs. In our previous work [2], we presented a VM placement algorithm based on intra-balanced resource allocation, using the cosine and Jaccard distances.

VM migration is a resource-intensive procedure that causes performance degradation of the running applications. This degradation is due to the extra consumption of CPU and bandwidth to ensure the in-cycle consolidation process. To minimize the number of migrations, it is necessary to avoid the detection of unreliable overloaded/underloaded hosts. Starting from this idea, we design a new approach for overloaded-host detection based on neural network and Markov models. The neural network is used to predict the future host CPU-requirements. The confidence of this prediction is measured through a dynamic safety parameter based on Markov model.

The remaining of this article is structured as follows : In Section II, we give an overview of the research works that deal with host's overload. In section III, we describe the principle of our new approach and its steps. Section IV presents the performance evaluation of our proposed model. Finally, conclusions and future works are presented in Section V.

II. related work on host-overload detection

In the literature, we identify two categories that are related to host-overload detection: adaptive utilization threshold and workload prediction methods.

A. Adaptive utilization threshold methods

The idea is to set an upper threshold, and consider the hots, for which the total CPU utilization exceeds this threshold, as overloaded. This upper threshold can be static as in [3] or dynamic [1]. The problem with static thresholds is that they are manually adjusted, and this is unsuitable for the dynamic nature of workload. Therefore, Beloglazov and Buyya in [1] propose a statistical uto-adjustment of the upper threshold based on historical VMs utilization.

They designed two robust statistic thresholds based on the Median Absolute Deviation (MAD) and the InterQuartile Range (IQR). The upper threshold T_u is calculated as follows [1]:

$$T_u = 1 - s \times StatisticMeasure \tag{1}$$

Where s is a safety parameter which defines a trade-off between energy consumption and SLA violation. The higher is s, the lower is the SLA violation, but the higher is the energy consumption.
$StatisticMeasure$ is the MAD or the IQR method.

In [4], authors propose an intelligent and adaptive upper threshold based on Dynamic Fuzzy Q-learning (DFQL). They design an algorithm that interacts with some characteristics of host environment to learn when to decide that this host is overloaded. The main drawback of this method is that it takes a long time until the convergence of the learning procedure.

B. Workload prediction methods

Workload prediction methods attempt to anticipate the resource demands by applying time series and machine learning approaches to predict the future resource usage. Effective prediction of resources can allow a load balancing and will help to avoid the SLA violation.

The predictive model extracts patterns from actual workload and analyzes the n previous workload values in order to predict upcoming workload on the data center at time instance $n + 1$.

Many researches are conducted in the literature. In [1], a host is considered as overloaded if:

$$PFRU \times s = 1 \tag{2}$$

Where s is the safety parameter similarly to the one of equation(1) and PFRU is the Predicted Future Resource Utilisation.

Markov-model prediction method is proposed in [5] where the authors designed a MadMCHD (Median Absolute Deviation Markov CHain Detection) algorithm that detects the future load state among : Overloaded, Normal and Under-loaded states. If the overloaded or underloaded state is detected for the future load, the migration process is triggered.

The ARIMA (Autoregression Integrated Moving Average) model and its improved versions are one of the most widely used technique for resource demand prediction in cloud computing [6][7][8]. Chen and Wang in [9] proposed a hybrid method for short-term host usage prediction that combines the ensemble empirical mode decomposition (EEMD), runs test (RT) and ARIMA. The non-stationary host usage is decomposed into relatively stable intrinsic mode fonctions, using the EEMD method, and the ARIMA is used to predict the future value of each component. The result model involves a better time and error prediction compared to the ARIMA baseline.

Another relevant type of techniques for host workload prediction includes neural and deep learning methods [10][11]. The authors in [12] proposed an adaptive two-stage multi-neural network model (ATSMNN), based on artificial neural networks (ANNs), which is composed of one classification model that is trained to classify the workload into two categories. Then, two prediction NN models are used to predict the workload according to the identified category from the first stage. The obtained results show a more accurate prediction compared to ARIMA, LR, and NN.

III. Proposed model

As it was mentioned in equation (2), a host-overload detection decision depends on two elements: the predicted future resource utilization and the safety parameter. In contrast to the existing workload prediction methods which rely on a fixed safety parameter, our model considers a dynamic one.

The workload prediction is performed using a neural network (NN) trained on each VM historical utilization. Then, the predicted host usage is estimated by summing up the predicted values of its VMs. To make this prediction more confident, the safety parameter is designed, based on Markov Model, to take into account only the historical data of the current host.

A. Neural network model for CPU-usage prediction

Several steps are involved in using neural networks for CPU-usage prediction. The neural network is trained and evaluated in an off-line mode, and then is used in a live prediction. The training and evaluation steps include:

1) The generation of the training data base: based on the sliding window technique, the VMs historical CPU-usage is fragmented into input and target data. For a given, T-length VM historical vector, $v_h = \{cpu_1, cpu_2, ..., cpu_n, cpu_{n+1}, ...cpu_T\}$ and a window slide of size n, the neural network input and target data are structured as follows:

$$X = \begin{bmatrix} Cpu_1 & Cpu_2 & ... & Cpu_n \\ Cpu_2 & Cpu_3 & ... & Cpu_{n+1} \\ . & & & \\ . & & & \\ Cpu_{T-n} & Cpu_{T-n} & ... & Cpu_{T-1} \end{bmatrix} \tag{3}$$

$$Y = \begin{bmatrix} Cpu_{n+1} \\ Cpu_{n+2} \\ . \\ . \\ Cpu_T \end{bmatrix} \quad (4)$$

The window slide size is fixed through a set of experiments.

2) The training of the neural network: our model is composed of 4 layers, an input layer, two hidden layers, and an output layer. The number of neurons is fixed to one for the output layer, while it is adapted to windows size in the input layer, and fixed to 200 in the case of the hidden one. The ReLu (Rectified Linear Unit activation) activation function [13] is used in each node, and the Adam optimizer [13] is employed to adapt the estimation of gradient descent training algorithm.

3) The evaluation of the trained neural network : the prediction accuracy is evaluated using the root of mean squared error (RMSE) metric:

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2} \quad (5)$$

Where y_i and $\hat{y}_i$ are the actual and predicted workloads. As the RMSE value is close to zero, the predicted values are close to the real values.

Once the neural network has been trained and evaluated, it will be used within the overloaded-host detection algorithm (Algorithm.1). This latter will be run periodically to analyze the workload of all the hosts across the data center. The algorithm uses the neural network model (line 3) to predict the future utilization for each VM on the input host. This prediction is then normalized (line 4) relatively to the total CPU capacity of the current VM and the target host. The algorithm gets the predicted host utilization by summing up the normalized prediction of all its VM (line 5), and then multiplies the sum by the safety parameter s (line 7) to be more prudent about the SLA violation. Finally, the overloaded decision is taken by checking if the future host utilization exceeds the total CPU on the host (line 8).

Algorithm .1: Overload host detection

Input: host, ANN_Model, window_size
Output: host overloaded decision: True or False

1 **foreach** $vm \in host.getVMsList()$ **do**
  /* get the required vm history vector */
2 $HistVect = getVmHistory(vm, windows\_size)$
  /* get the predicted vm usage, using the trained neural netwok */
3 $PredVmUsg = ANN\_Mdodel.getPred(HistVect)$
  /* Normalize the predicted vm usage */
4 $PredVmNorm = \frac{PredVmUsg \times TotalCPUCap(vm)}{TotalCPUCap(host)}$
  /* Calculate the predicted host usage */
5 $PredHostUsg += PredVmNorm$
6 **end**
7 $FinalPredHostUsg = PredHostUsg \times s$
8 **return** $FinalPredHostUsg > 1$

B. Markov Model for safety parameter prediction

In this section, we will explain our dynamic estimation of the safety parameter based on previously observed host state using Markov model technique.

Our Markov model is defined as a discrete stochastic variable X_t that can take one of two host state values, namely, overloaded(O) and Normal load (N). According to the Markov memoryless propriety, the evolution of state in the future depends only on the present state and does not depend on the past history as it is announced by the following formula :

$$P(X_{t+1}|X_t, X_{t-1}, ..., X_1) = P(X_{t+1}|X_t) \quad (6)$$

We assume that the transition probabilities between states are time-homogeneous, which means that they are independent of time t:

$$P(X_{t+1} = s_i|X_t = s_j) = P(X_{t+T+1} = s_i|X_{t+T} = s_j) \quad (7)$$

Where T is positive integer and $s_j, s_j \in \{O, N\}$.

For a given hot state, the goal is to compute the probability to be on an overloaded or not overloaded state in the next time step. To compute these probabilities, we start by tagging the host history as it is illustrated in Fig. 1.

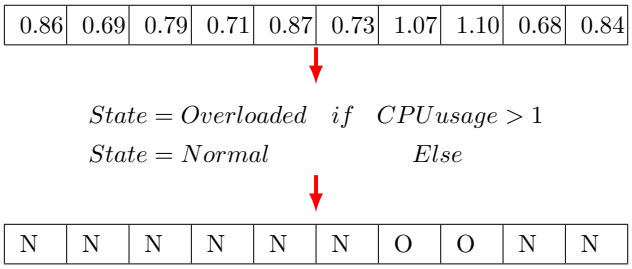

Fig. 1. Tagging the host history

The probability that the next host state will be overloaded is calculated as follows:

$$\begin{aligned} ProbaNextOver &= P(X_{t+1} = O|X_t = s_j) \\ &= \frac{P(X_{t+1} = O, X_t = s_j)}{P(X_t = s_j)} \end{aligned} \quad (8)$$

Where $s_j \in \{O, N\}$

TABLE I
THE ENERGY CONSUMPTION AT DIFFERENT LOAD LEVELS.

Host type \ Host CPU usage	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
HP ProLiant ML110 G4	86	89.4	92.6	96	99.5	102	106	108	112	114	117
HP ProLiant ML110 G5	93.7	97	101	105	110	116	121	125	129	133	135

Based on the illustrated example of Fig 1, the future host state is calculated as:

$$\begin{aligned} P(X_{t+1} = O | X_t = N) &= \frac{P(X_{t+1} = O, X_t = N)}{P(X_t = N)} \\ &= \frac{P(X_{t+1} = O, X_t = N)}{P(X_t = N, X_{t-1} = N) + P(X_t = N, X_{t-1} = O)} \\ &= \frac{1}{9} \end{aligned} \tag{9}$$

Similarly, the $P(X_{t+1} = O | X_t = O) = \frac{1}{9}$
and the safety parameter is then calculated, based on the following formula :

$$s = 1 + ProbaNextOver \tag{10}$$

The value of s in equation 10 is designed to support or adjust the neural network prediction. This adjustment will take into account only the historical data of the current host. The higher is the ProbaNextOver, the greater is the s value, and this will make the overload host detection more prudent.

It is worthy to mention that our Markov probabilities are calculated after collecting 10 historical observations. If it is not the case, a fixed safety parameter is considered.

IV. PERFORMANCE EVALUATION

This section describes the simulation setup of our cloud computing environment and the evaluation of the proposed overloaded-host detection algorithm.

A. Simulation setup

Our simulation setup has been configured with CloudSim toolkit [14], An open source and flexible Java library that allows the simulation of VM placement and consolidation. Our environment is modeled by a a data center consisting of 800 physical machines. Each machine, is modeled as one of those two types:

- HP ProLiant ML110 G4 with 2 cores of 1860 MIPS.
- HP ProLiant ML110 G5 with 2 cores of 2660 MIPS.

Table I illustrates the energy consumption of the aforementioned physical machines.

We consider five types of virtual machines which correspond to Amazon EC2, all of which are single-core instances. Their specifications are shown in Table II.

For all our experiments, the real workloads provided by the CoMon project PlanetLab are used [15]. The VM's traces provide the CPU utilization by thousands of VMs located at different servers around the world.

TABLE II
VMS CARACTERISTIQUES

VM Type	Large	Med	Small	Micr	Nano
Processor (MIPS)	2500	2000	1000	500	250
Memory (MB)	2048	2048	1024	1024	512
Bandwidth (GB/s)	1	1	1	1	1

The neural network is trained in off-line mode using Python language and Keras library [16]. The obtained model is then used for live prediction in CloudSim based on DeepLearning4j [17] library. The neural network model is trained with traces of 1024 VMs.

B. Performance Metrics

The performance of our proposed approach is evaluated through the following metrics:

1) Total energy consumption (E) : in our work, this metric is computed from realistic values provided by the SPECpower benchmark implemented in CloudSim. Table I illustrates the energy consumption of the aforementioned physical machines used in our simulation.
2) SLA Violations(SLAV): it measures the performance degradation experienced by each VM. This degradation is due to overloaded hosts, and to VM migration process. So, this metric is performed based on the SLA violation Time per Active Host (SLATAH) combined to the Performance Degradation due to Migrations (PDM) as defined in the following:

$$SLAV = SLATAH \times PDM \tag{11}$$

where SLATH is computed as in Formula (12) :

$$SLATAH = \frac{1}{N} \sum_{i=1}^{N} \frac{T_{si}}{T_{ai}} \tag{12}$$

Where N is defined as the total number of hosts, T_{si} is the total time during which host i has experienced 100% and caused the SLA violation, and T_{ai} is defined as the total time where a host i is in an active mode.

$$PDM = \frac{1}{M} \sum_{j=1}^{M} \frac{C_{dj}}{T_{rj}} \tag{13}$$

Where M is the total number of VMs, C_{dj} is the estimated performance degradation of the VMj due to migrations. This estimation is set, in our experiments, to 10% of CPU utilization. C_{rj} is the total CPU capacity required by the VM_j over its lifetime.

TABLE III
EVALUATING THE ESV METRIC FOR SEVERAL METHODS AT DIFFERENT STATIC SAFETY PARAMETER VALUES

	Safety parameter																	
	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	2	2.1	2.2	2.3	2.4	2.5	2.6	2.7	2.8
MAD	1,13	1,04	0,97	0,92	0,83	0,80	0,73	0,68	0,65	0,60	0,60	0,55	0,55	0,54	0,53	0,48	0,50	0,50
IQR	0.57	0,53	0,47	0,48	0,51	0,48	0,51	0,50	0,49	0,53	0,54	0,55	0,56	0,56	0,59	0,61	0,59	0,64
LRR	1.04	0.54	0,49	0,52	0,54	0,62	0,70	0,78	0,85	0,92	0,89	0,91	0,93	0,91	0,91	0,93	0,94	0,98
NN	0.49	0.42	0.33	0,30	0.35	0,33	0,39	0,38	0,42	0.39	0,42	0,37	0,37	0,37	0.34	0,34	0,33	0,32

3) Energy Service Level Agreement Violations (ESV): is a trad-off metric between Energy and SLAV. The ESV calculation is as follows:

$$ESV = E \times SLAV \tag{14}$$

4) Number of Migrations: in order to avoid the additional SLAV caused by migration process, a good consolidation approach must reduce the total number of VM migrations.

C. Results and discussions

To investigate the prediction performance of the proposed neural network model, we compute the RMSE for different historical windows length: 6, 8, 10, 12, 14. The best RMSE value was 8,61 and it corresponds to history window length of 10.

It is worthy to mention that for all our experiments, the Minimum Migration Time (MMT) algorithm [1] is used for VM selection phase whereas the VM placement is based on the Power-Aware Best Fit Decreasing (PABFD) algorithm [1]. Our proposed algorithm didn't deal with the underloaded host detection, where we kept the simple strategy proposed by [1] and implemented in CloudSim [14], that selects the host with the minimum CPU utilization as underloaded.

To evaluate our proposed approach, we compare it to the state-of-the-art algorithms proposed in [1]:

- MAD (Median Absolute Deviation)
- IQR (Inter Quartile Range)
- LRR (Local Robust Regression).

We start by evaluating the ESV metric for the aforementioned algorithms at different static safety parameter values. Table III depicts the obtained results from which we can retain the best static safety parameter values for each algorithm that are 2.6, 1.3, 1.3, 1.4 for MAD, IQR, LRR, NN respectively. These values will be fixed for next experiments. Moreover, we can conclude that the NN model provides the best ESV value, with 0.30, compared to the other algorithms. This result proves that a good prediction of the near workload future yields to a better trade-off between saving energy and avoiding the SLA violation.

In the second part of experiments, we introduce our proposed dynamic safety parameter based on Markov Model. This parameter is combined to the NN model as it is described in Section III. We call the combination result as NN_MARK model.

Fig.2 reports the resulting energy consumption, SLA violation, the number of VM migration and the ESV metric for each of the considered algorithms. The proposed NN_MARK model can reduce energy consumption by 22%, 28.49%, 34.16% and 33.55% compared to NN, LR, IQR and MAD methods, respectively, as it is depicted by Fig. 2.a. This reduction is due to the fact that NN_MARK model does not consider a fixed safety parameter which causes in some situation an unnecessary migration of VMs and consequently maximizes the number of active PMs. Fig.2.b clearly depicts that the NN_MARK approach minimizes the number of VMs migration by about twice compared to the state-of-the art ones. The safety parameter in the NN_MARK model is dynamically adjusted to the workload history on the current host. This adjustment allows to decrease the safety parameter according to the probability that the current host will be overloaded. Therefore, it is expected that it minimizes the energy consumption but it will cause some extra SLA violation. As it can bee seen from Fig.2.c, our proposed method results in slight more SLA violation compared to the method based on NN (0.0025 vs. 0.0029). However, is still more robust if we compare it to LR, IQR and MAD. Finally in Fig.2.d, we compare the ESV value amongst all the methods. It is obvious that our proposed method outperforms in term of ESV, which proves that it presents the best trade-off between energy minimization and QoS maximization under a specified SLA.

V. CONCLUSION AND FUTURE WORKS

In this article, we proposed a new host-overload detection algorithm based on combining neural network and Markov model (NN_MARK). Our proposed approach uses the neural network to predict the near future workload of PMs, and the Markov model to measure the confidence of this prediction through a dynamic safety parameter. The obtained results show that our proposed approach outperforms the existing solutions in terms of the total energy consumption, SLA violation, the number of VM migration, and the ESV metrics. As future works, we aim to explore other machine learning techniques to get better RMSE precision in the prediction phase. Furthermore, prediction can be extended to integrate multiple resources such as : CPU, memory and I/O bandwidth.

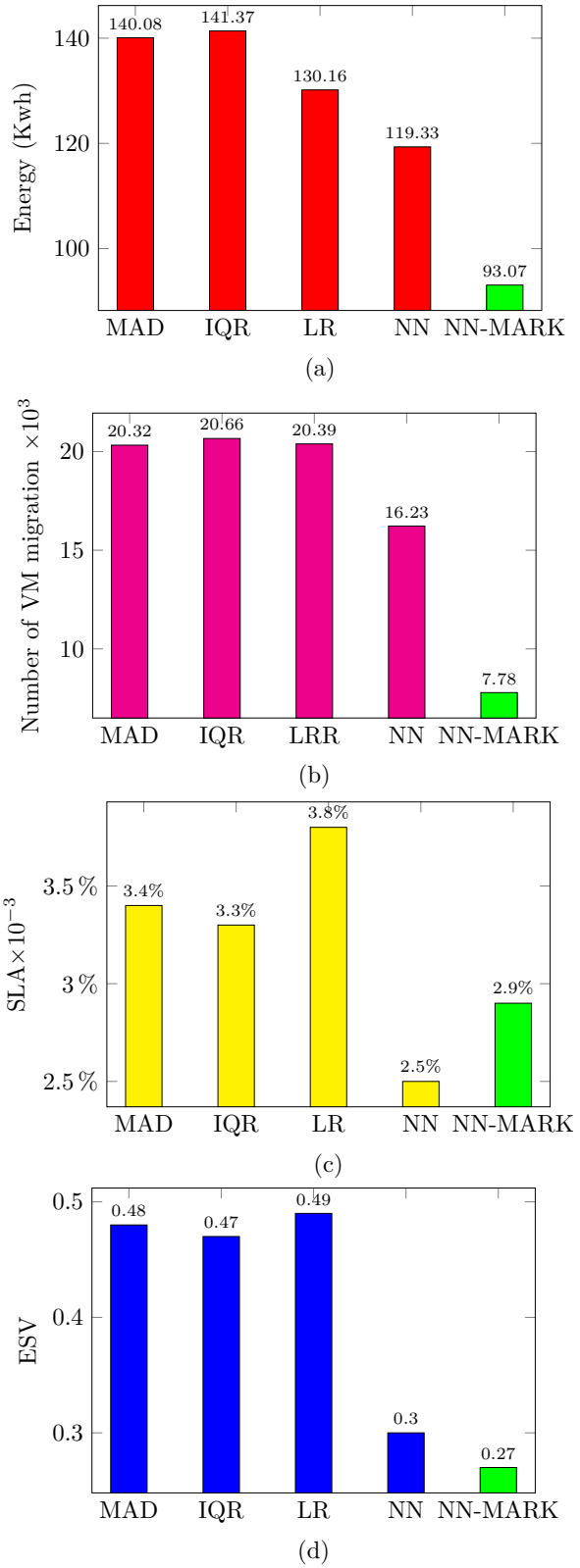

Fig. 2. Simulation results

References

[1] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers", Concurrency Comput.: Pract. Experience, vol. 24, no. 13, pp. 1397-1420, 2012.

[2] I. El-Taani, M.C Boukala, and S. Bouzefrane, "Energy-Aware VM placement based on intra-balanced resource allocation in data centers", 8th International Conference on Future Internet of Things and Cloud (FiCloud),IEEE, pp. 400-405, 2021.

[3] A. Beloglazov, J. Abawajy and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing", Future Gener. Comput. Syst., vol. 28, no. 5, pp. 755-768, 2012.

[4] S. S. Masoumzadeh and H. Hlavacs, "An intelligent and adaptive threshold-based schema for energy and performance efficient dynamic VM consolidation", Proc. Eur. Conf. Energy Efficiency Large Scale Distrib. Syst., pp. 85-97, 2013.

[5] M.S Bani et al., "Markov Prediction Model for Host Load Detection and VM Placement in Live Migration", IEEE Access, vol. 6, pp. 7190-7205, 2018.

[6] M.C Samani and F.S Esfahani, "PCVM.ARIMA: predictive consolidation of virtual machines applying ARIMA method", The Journal of Supercomputing, Jun. 2020.

[7] K. Dmytro, T. Sergii and P. Andiy, "Arima forecast models for scheduling usage of resources in it-infrastructure", Proc. 12th Int. Sci. Techn. Conf. Computer. Sci. Inf. Technol. (CSIT), vol. 1, pp. 356-360, 2017.

[8] A. Nadjar, S. Abrishami and H. Deldari, "Hierarchical VM scheduling to improve energy and performance efficiency in IaaS Cloud data centers". 5th International Conference on Computer and Knowledge Engineering (ICCKE), pp. 131-136, 2015.

[9] J. Chen and Y. Wang, "A Hybrid Method for Short-Term Host Utilization Prediction in Cloud Computing", Journal of Electrical and Computer Engineering, pp. 1-14, 2019.

[10] Z. Huang, J. Peng, H. Lian, J. Guo and W. Qiu, "Deep recurrent model for server load and performance prediction in data center", Complexity, vol. 2017, no. 99, pp. 1-10, 2017.

[11] M. S. Ricardo, N. Goel, M. Zaman, R. Joshi, M. Daraghmeh and A. Agarwal, "Developing Machine Learning and Deep Learning Models for Host Overload Detection in Cloud Data Center", 12th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON) IEEE ,pp. 0619-0626,2021.

[12] L. Lei, et al. "Two-stage adaptive classification cloud workload prediction based on neural networks." International Journal of Grid and High Performance Computing (IJGHPC), 2019.

[13] J. Pomerat, A. Segev and R. Datta, "On Neural Network Activation Functions and Optimizers in Relation to Polynomial Regression", Proc. - IEEE Int. Conf. Big Data Big Data 2019, no. 2, pp. 6183-6185, 2019.

[14] R. N. Calheiros, R. Ranjan, C. A. F. De Rose, and R. Buyya. . "CloudSim: A novel framework for modeling and simulation of cloud computing infrastructures and services".arXiv preprint arXiv:0903.2525. 2009

[15] K. Park and V. S. Pai. CoMon"A Mostly-Scalable Monitoring System for Planet Lab". Operating Systems Review, vol. 40, pp. 65-74, 2006.

[16] F. Chollet " Keras: The python deep learning library", Available: https://keras.io/.

[17] L.Steven, et al. "Wekadeeplearning4j: A deep learning package for weka based on deeplearning4j" Knowledge-Based Systems,vol. 178, pp. 48-50. 2019.