Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.050626

ARTICLE

A Prediction-Based Multi-Objective VM Consolidation Approach for Cloud Data Centers

Xialin Liu[1,2,3,*], Junsheng Wu[4], Lijun Chen[2,3] and Jiyuan Hu[5]

[1]School of Computer Science, Northwestern Polytechnical University, Xi'an, 710005, China

[2]School of Computer Science and Technology, Xi'an University of Posts and Telecommunications, Xi'an, 710021, China

[3]Shaanxi Key Laboratory of Network Data Analysis and Intelligent Processing, Xi'an University of Posts and Telecommunications, Xi'an, 710021, China

[4]School of Software and Microelectronics, Northwestern Polytechnical University, Xi'an, 710005, China

[5]School of Automation, Xi'an University of Posts and Telecommunications, Xi'an, 710021, China

*Corresponding Author: Xialin Liu. Email: liuxialin@xupt.edu.cn

Received: 12 February 2024 Accepted: 11 June 2024 Published: 18 July 2024

ABSTRACT

Virtual machine (VM) consolidation aims to run VMs on the least number of physical machines (PMs). The optimal consolidation significantly reduces energy consumption (EC), quality of service (QoS) in applications, and resource utilization. This paper proposes a prediction-based multi-objective VM consolidation approach to search for the best mapping between VMs and PMs with good timeliness and practical value. We use a hybrid model based on Auto-Regressive Integrated Moving Average (ARIMA) and Support Vector Regression (SVR) (HPAS) as a prediction model and consolidate VMs to PMs based on prediction results by HPAS, aiming at minimizing the total EC, performance degradation (PD), migration cost (MC) and resource wastage (RW) simultaneously. Experimental results using Microsoft Azure trace show the proposed approach has better prediction accuracy and overcomes the multi-objective consolidation approach without prediction (i.e., Non-dominated sorting genetic algorithm 2, Nsga2) and the renowned Overload Host Detection (OHD) approaches without prediction, such as Linear Regression (LR), Median Absolute Deviation (MAD) and Inter-Quartile Range (IQR).

KEYWORDS

VM consolidation; prediction; multi-objective optimization; machine learning

1 Introduction

Cloud computing is a computational paradigm that offers its users scalable services in an on-demand pattern. The most prominent feature of cloud computing is that it provides users with virtualized resources [1]. Cloud computing utilizes various technologies, such as virtualization, distributed computing, and storage. It eliminates front-end capital and customers' ongoing maintenance. Cloud computing permits customers to increase and decrease resource demands and pay accordingly [2]. Cloud providers (CPs) should allocate and deallocate resources on demand. As a result, operating costs are reduced [3].

VM consolidation can improve the utilization of resources by reducing EC in cloud data centers [4]. VM consolidation process consists of three phases in general [4]:

1. Detection of overload/underload PMs. Identifying overloaded PMs and migrating VMs to prevent potential PD or even Service Level Agreements (SLA) violations. Identifying underloaded PMs, migrating all VMs on the PMs, and turning the PMs off to reduce EC.
2. VM selection. The candidate VMs from overload PMs are selected to keep the PM under average load. Some policies, such as Maximum Correlation Coefficient (MCC), Minimum Migration Time (MMT), and Minimum Utilization (MU) are utilized.
3. PM selection. The destination PMs for the VMs coming from overload/underload PMs. Some objectives, such as EC, RW, and MC, can be considered.

Most works consolidate VMs based on PM/VM current resource utilization. In other words, all three phases of VM consolidation should be performed based on current resource utilization. However, due to the unsteadiness and high changeability of the workload, PM/VM resource utilization is unstable and highly variable. Improving these three phases of VM consolidation is essential to matching VM's dynamic change of workload and resource capacity. The first phase of VM consolidation requires an approach that predicts future resource usage. Otherwise, the detection results of overload/underload will soon become invalid, and a high number of needless VM migrations may be caused not only because the workload is highly variable but also because there are large numbers of PMs and VMs in the data center, it takes neglected time to make and implement migration decisions. For instance, according to a migration decision, a VM should be migrated from an overload PM to another underload PM. Unfortunately, this underload PM cannot accommodate the VM when implementing the migration decision due to real-time changes in workload. The unreliable detection result in the first phase may result in the selected VMs and destination PMs not being the optimal options in the second and third phases. The unreliable detection result increases the overheads, such as the EC for VM migration, PD, and network traffic [5,6].

On the other hand, the existing works consider a few optimization goals. The obtained consolidation solutions may need to be improved in other optimization goals and hinder their application in practice. It is essential to consider more goals while consolidating VMs, as more factors influence the quality of the consolidation solution in practice:

(1) Although VM migration can reduce EC, VM migration also gives rise to VM PD or SLA violation on new PMs because sharing hardware resources among more VMs increases resource contention.
(2) The MC of a VM in different cloud data centers varies greatly because it is heavily dependent on data center's configuration, such as network architecture and bandwidth usage, and the application executed on the VM.
(3) Maximizing resource utilization has always been an essential resource management goal in data centers.

The above considerations may be conflicting when combined. Therefore, finding an effective strategy that considers compromise among all the goals is essential.

This paper introduces a novel prediction-based multi-objective VM consolidation approach that stands out for its two key innovations. The first innovation is a hybrid method that combines ARIMA and SVR to predict future resource usage. The second innovation is the consideration of multiple objectives such as EC, PD, MC, and RW, and the optimization of these objectives using Nsga2 based

on future resource utilization. This approach allows for the search for the optimal solution in a more timely and practical manner.

The main contributions of this paper are summarized as follows:

1. We propose a prediction approach based on the ARIMA-SVR model to forecast future VM resource demand, which can capture both linear and nonlinear features of resource demand data. Specifically, we use the ARIMA model to predict the future resource demand of VM preliminarily. Because the ARIMA model can only capture linear time series features, we use the SVR model to assist the ARIMA model. The SVR model predicts future errors. Adding the forecasts of the ARIMA and SVR models obtains the final prediction.
2. We design a framework to predict future resource demands of VMs in a cloud data center. This framework depicts the process for predicting future VM resource demand. We propose an algorithm to estimate the parameters of the SVR model and train it.
3. Our proposed prediction-based multi-objective VM consolidation approach, which leverages future resource demands of VMs for multi-objective optimization, offers significant practical value. It enables the obtained consolidation solution to be both timely and practical. To validate the effectiveness of our approach, we have conducted a series of rigorous experiments.

We organize this paper as follows: Section 2 reviews the related works on resource usage prediction in cloud data centers and multi-objective VM consolidation. Section 3 provides our problem formulation. Section 4 provides the specifics of the proposed approach, including the prediction framework and algorithms. Section 5 is devoted to performance evaluation and shows the superiority of the proposed approach. We conclude the paper and describe future work in Section 6.

2 Related Works

2.1 Resource Usage Prediction in Cloud Environment

The VM's resource demand may change over time. Accurate prediction of VM's future resource demand improves resource utilization efficiency. Some approaches have been developed to predict cloud resource demand. Reference [7] used a gray model to predict the host's CPU and RAM resource usage. The gray model does not require a large number of training data. However, it cannot guarantee accurate prediction of workloads with frequent fluctuations. Exploiting a Gray-Markov (G-M)-based model predicts future usage of resources [8]. Reference [9] used a Discrete-Time Markov chain (DTMC) model to predict future resource usage. It presents a multi-objective VM placement approach to search for the optimal solution for the VM placement problem by exploiting the ε-dominance-based multi-objective artificial bee colony (ε-MOABC) algorithm.

Other researchers use machine learning (ML) technology to predict resource usage. Reference [10] proposed a Host States Naive Bayesian Prediction (HSNBP) model in cloud data centers. The HSNBP model can forecast overload hosts using The Naive Bayesian (NB) classifier. Reference [11] proposed an ARIMA model for forecasting workload. Due to the simplicity of implementation, LR was used to estimate future resource usage in many types of research. Reference [12–15] proposed employing LR methods to predict CPU usage. However, the LR technique considers only the linear features in time series data. The relationship between resource demand data and time appears more curved. In [16], the authors replaced the LR with a K-Nearest Neighbor Regression (KNNR). To avoid needless VM migration, reference [17] proposed a Bayesian network-based estimation model for live VM migration. Reference [18] presented a self-directed workload forecasting method (SDWF). SDWF utilizes a Multilayer Neural Network (MNN) to learn historical data and forecast the next possible

workload value. Reference [19] presented a Multi-Resource Feed-forward Neural Network (MR-FNN) to predict the multiple resource demands concurrently. MR-FNN uses a differential evolution algorithm to enhance its learning and prediction capability. Reference [2] employed SVR to predict the future usage of multi-attribute host resources. They applied a Sequential Minimal Optimization (SMO) algorithm to train the SVR model. Reference [20] proposed a framework for cloud resource allocation based on reinforcement learning (RL) mechanism and fuzzy logic (FL).

To enhance prediction accuracy, some researchers use hybrid or ensemble learning methods. Reference [21] presented a hybrid method that combines LR and RL techniques to handle changes in workload trace. In [22], host overload detection is based on an ARIMA model. Reference [23] proposed a new Swarm Intelligence Based Prediction Approach (SIBPA) to predict the resource demands of a cloud user. SIBPA combines the Multiple SVRs and ARIMA models to consider mapping the linear and nonlinear attributes in time series. It uses the Particle Swarm Optimization (PSO) approach to select the best features for prediction results. Reference [24] employed ARIMA and Auto-Regressive Neural Network (ARNN) to predict the usage of resources. Reference [3] developed different ML prediction models to perform a consolidation, including LR, Multilayer Perceptron (MLP), SVR, Decision Tree Regression (DTR), and Boosted Decision Regression (BDR). Reference [25] presented an ensemble learning-based workload forecasting method. The proposed framework employs multiple MNNs as base predictors. It used an algorithm illuminated by black hole theory to select the best weight. Reference [26] proposed an Intelligent Regressive Ensemble Approach for Prediction (REAP), which uses eight ML methods to predict CPU usage. Reference [27] proposed an ensemble prediction for forecasting the future needs of resources. The proposed predictor consists of four fundamental prediction models: MA, exponential smoothing (ES), LR, and double ESs. Other researchers use Deep Learning techniques to predict resource usage. Reference [28] proposed a host load predictive model based on a long short-term memory model in a recurrent neural network (LSTM-RNN). Reference [29] developed the workload prediction based on LSTM networks, which uses four LSTMs. Reference [30] proposed a deep RL-based resource management. Table 1 compares all the above prediction approaches regarding application domain, strengths, and weaknesses. These approaches do not consider prediction-based multi-objective optimization. Thus, the issued decision can be insignificant because real-world problems are often multi-objective optimization problems. In this paper, we propose a prediction model based on which we optimize multiple objectives simultaneously. In addition, the proposed prediction model can capture both linear and nonlinear features of resource demand data.

Table 1: Comparison of the existing prediction approaches

Paper	Technique uesd	Domain	Strength	Weakness
[7]	Gray	Resource management	Balances workload and lower power consumption.	Cannot guarantee exact prediction results.
[8]	G-M	VM consolidation	- Prediction accuracy increases. - Effectively reduces the number of migrations, EC, and SLA violations.	Suitable only for short-medium-term future resource utilization.

(Continued)

Table 1 (continued)

Paper	Technique uesd	Domain	Strength	Weakness
[9]	Markov-MOABC	VM consolidation	Minimizes EC, RW, and maximizes the reliability.	Suitable only for short-term future resource utilization.
[10]	NB	VM consolidation	Reduces SLA violation rates while keeping energy cost efficient.	Need to know the prior probability that depends on some hypothesis.
[11]	ARIMA	Resource provisioning	- Simple implementation. - Efficiency in resource utilization.	Suitable only for predicting stationary future resource utilization.
[12]	LR	VM live migration	- Simple implementation. - Reduces EC and SLA violation rates.	Suitable only for short-term future utilization and capturing linear relationship.
[13]	LR	VM consolidation	- Simple implementation. - Reduces the EC.	Suitable only for short-term future utilization and capturing linear relationship.
[14]	LR	VM consolidation	- Simple implementation. - Reduces both load and the power consumption.	Suitable only for short-term future utilization and capturing linear relationship.
[15]	LR	VM consolidation	- Simple implementation. - Saves energy by eliminating the repeated migration of the same VM.	Suitable only for short-term future utilization and capturing linear relationship.
[16]	KNNR	VM consolidation	Minimizes EC and maintains performance.	High complexity.
[18]	MNN-Black hole	Resource management	- Reduces the mean squared forecasting errors. - Captures nonlinear relationship.	Need optimization algorithm for training neurons.
[19]	SVR	Resource management	Reduces RW and saves energy.	Parameters must be discreetly set.
[2]	FNN	VM autoscaling and placement	Achieves the high data center energy efficiency and avoiding SLA violations.	- Require sufficient data. - Need optimization algorithm to train the weights of FNN.
[20]	RL-FL	Resource management	Reduces the total cost and increases the resource utilization.	Training result is unstable.
[21]	LR-RL	Resource provisioning	Improves resource utilization and response time.	Suitable only for predicting stationary future resource utilization.

(Continued)

Table 1 (continued)

Paper	Technique uesd	Domain	Strength	Weakness
[22]	Auto regressive-ARIMA	VM consolidation	Reduces EC, the number of VM migrations. QoS guarantees.	Parameters must be discreetly set.
[23]	ARIMA-MSVR-PSO	IT budget	Predicts dynamic behavior in a long term.	Parameters must be discreetly set.
[24]	ARIMA-AR-NN	Resource management	Minimizes EC, maintaining QoS and reducing cost.	Need optimization algorithm for training neurons.
[25]	MNN-Black hole	Resource management	Reduces the RW, EC and carbon footprints.	Need optimization algorithm for training neurons.
[26]	BRR-BRNN-SVM-DT-ELM-LM-NN-RF	Resource management	Improves the accuracy rate and reducing the execution time.	Suitable only for short-medium-term future utilization and capturing linear relationship.
[27]	MA-ES-LR-DES	VM placement	Reduces EC.	Time-consuming.
[28]	LSTM-RNN	Resource provisioning	Decreases EC while keeping a high QoS.	- Time-consuming. - Parameters need to be carefully set.
[29]	LSTM-RNN	Resource provisioning	Dynamic resource scaling, reduces the number of active machines.	- Time-consuming. - Parameters need to be carefully set.
[30]	DRL	Resource management	Improves resource utilization.	Very complicated and time-consuming.

2.2 Multi-Objective VM Consolidation

This section presents the optimization approaches to VM consolidation proposed in existing works. We divide these approaches into categories: (1) multi-objective approach transformed into mono-objective and (2) multi-objective approach.

(1) Multi-objective transformed into mono-objective approach

They combine multiple objectives into a mono-objective. This approach may be classified as I. the weighted sum method. The user gives the weight of each objective. II. the constraint method. Choosing only one objective as mono-objective and considering other objectives as constraints.

I. The weighted sum method

They combine all objectives into a mono-objective by attaching each weight to each objective. A multi-objective energy-efficient VM consolidation using adaptive beetle swarm optimization (ABSO) algorithm is proposed [31]. The proposed ABSO is a hybridization of PSO and BSO. Reference [32] studied consolidation problems of processor-intensive and disk-intensive workloads and formulated the problem as the four-objective functions. The objective functions are (1) total energy consumed by the processor, disk, VM migration and turning PMs ON and OFF, (2) consolidation fitness of multiple disk-intensive VMs, (3) processor utilization of PM on which at least one disk intensive VM is running and (4) number of PMs with too high processor utilization. To solve the presented multi-objective optimization problem, it first calculates the optimum point for each objective, then weights the difference ratios between each objective function and its optimum point and sums them

finally. The Simulated Annealing (SA) method obtains the optimal solution. Reference [33] proposed a method based on the Monarch Butterfly Optimization algorithm (MBO) for VM placement to maximize packaging efficiency and reduce the number of active physical servers. Reference [1] aimed at controlling manufacturing costs and treated MCs, the energy cost of servers, the cost of creating VMs, and the total penalty cost for tasks whose demands are not satisfied as four objective functions. The user determines the weights.

The approaches above are all priori methods, and they must clarify prior information's impacts on their solution. First, it is challenging to set weights accurately. Second, the metric used in calculating the objective function is diverse for diverse objectives, resulting in a significant error in the weighted objective function value.

II. The constraints method

They treat one of the objectives as one objective and consider the others as constraints. Reference [5] minimized the number of the required hosts for hosting VMs under SLA constraint. Reference [34] aimed at reducing the EC, composed of three parts: cost of assigning new VMs to PMs, MC, and cost of keeping PMs turned on, and treated the demands of CPU, memory, and network bandwidth as resource constraints. Reference [35] presented security-aware VM consolidation, which treats reducing the security risk of a PM as an objective, and the risk increase for each VM does not exceed the value of the proposed Risk Increase Threshold (RITH) as a constraint. Reference [36] proposed a failure-aware VM consolidation approach, which regards failures arising as a constraint before performing VM consolidation. Reference [37] presented an Interval-valued Fuzzy Logic (FL) mechanism aiming at power conservation while optimizing performance.

Constraints limit the variation range of independent variables on an objective. The solution obtained by this constraint may differ from the exact solution to that objective.

(2) Multi-objective method

A multi-objective optimization problem includes multiple objective functions and multiple constraints. To compare the two solutions, the concept of Pareto dominance is introduced. Reference [38] proposed a two-objective approach and made a tradeoff between these objectives using a Fuzzy Analytic Hierarchy Process (FAHP). Reference [39] aimed at energy saving and network bandwidth consumption and proposed a two-objective approach that treats the number of the required hosts for hosting VMs and the number of VM migrations as objective functions under SLA constraint. Reference [9] proposed a VM placement approach to obtain the optimal solution using the ε-MOABC algorithm, which can efficiently balance the overall EC, RW, and system reliability. Reference [40] proposed an approach that uses an assembly of sine cosine algorithm (SCA) and ant lion optimizer (ALO) for optimal VM assignment. The three objectives are EC, RW, and SLA levels. Reference [41] proposed a multi-objective ant colony (AC) system algorithm for VM consolidation. The goal is to minimize total RW and EC simultaneously. Reference [42] proposed an energy-aware and QoS-aware multi-objective AC Optimization (ACO) approach for VM consolidation.

The discussed multi-objective approaches have focused on only two or three factors and treat other factors as constraints, such as SLA. As for the VM consolidation problem, the factors may conflict with each other (i.e., minimizing EC and minimizing SLA violation, etc.). Therefore, each factor should be regarded as an objective instead of a constraint. Table 2 compares the approaches mentioned above.

Table 2: Comparison of the existing multi-objective VM consolidation approach

App	Mo	We	Con	Mul	Goals	Strategy	Limitation
[31]		√			EC, MC and resource utilization	Uses a hybridization of PSO and BSO.	DSWA/DOFD
[32]		√			EC, consolidation fitness, processor utilization and number of PMs with too high processor utilization	Uses SA algorithm.	DSWA/DOFD
[33]		√			EC, RW	Uses MBO.	DSWA/DOFD
[1]		√			Controls manufacturing costs	Uses discrete cuckoo optimization algorithm based on group technology (GT).	DSWA/DOFD
[5]			√		The number of the required PMs	Uses an LP formulation and heuristics to control VM migration.	SOES
[34]			√		EC	Uses a snapshot-based solution.	SOES
[35]			√		The security risk of a PM	Utilizes a three-dimensional security assessment model.	SOES
[36]			√		The occurrence of failures, the hazard rate of physical resources	Utilizes failure prediction technique based on exponential smoothing.	SOES
[37]			√		Energy savings, PD	Uses an Interval-valued FL.	SOES
[38]				√	EC, SLA	Uses a AHP.	FG
[39]				√	The number of the required PMs, the number of VM migration	Uses a mixed single-parent genetic algorithm.	FG
[9]				√	EC, RW, and the system reliability	Introduces a DTMC model to predict future resource usage and uses ε-MOABC algorithm.	MGI/DCPD
[40]				√	RW, power consumption and performance	Uses a combination of the SCA and ALO.	MGI/DCMC
[41]				√	RW and EC	Uses a multi-objective AC system algorithm.	FG
[42]				√	Energy efficiency, system performance and SLA-compliance	Uses multi-objective AC optimization approach.	FG

Note: Legend: App: Approach; Mo: mono-objective approach; We: The weighted sum approach; Con: The constraints approach; Mul: Multi-objective approach; DSWA: Difficult to set weights accurately; DOFD: The dimensions of the objective function are different; SOES: The solution obtained may not be the exact solution; FG: Few goals; MGI: The model for the goal is incomplete; HCT: Higher computing time; DCPD: Does not consider PD while VM consolidation; DCMC: Does not consider MC while VM consolidation.

3 Problem Formulation

3.1 Multi-Objective Optimization Formulation

To consolidate VMs dynamically when resource usage changes, we need to get an optimal allocation between VMs and PMs, which can optimize multiple system goals. Our work uses two-dimensional resources (i.e., CPU and memory) to characterize VMs and PMs because the VM's image file is stored in network-attached storage (SAN) in VM live migration technology. We propose a prediction-based multi-objective VM consolidation approach that minimizes a data center's total EC, RW, MC, and PD.

3.1.1 EC Modeling

Our previous work [43] introduced an EC model that considers EC in diverse states, EC of states switching, and EC of live migrations. The model we adopted in this paper significantly contributes to the VM consolidation field. Its inclusion in our research underscores the importance of considering EC in various scenarios, enhancing the robustness and applicability of our approach.

3.1.2 Performance Modeling

We can measure system performance in terms of such characteristics as turnaround time, maximum execution time, and maximum access capacity. It is challenging to evaluate and measure these characteristics directly. More importantly, these characteristics can vary with diverse applications. To assess and measure conveniently, we model PD when SLAV occurs and treat minimizing PD as one of the optimization objectives. We define that the SLA is delivered when 100% of the performance requested by an application inside a VM is provided at any time. As a result, SLA is violated when the system does not fully provide the performance requested by an application inside a VM at some point. There are two cases where the system cannot entirely perform an application request. One is that a PM serving applications is experiencing 100% utilization, and another is a live VM migration is in progress (i.e., it requires a specific MC such as migration latency and downtime). Therefore, we adopt two metrics proposed in [44] to calculate SLAV: SLA violation Time per Active Host (SLATAH) and the overall PD due to VM migrations (PDM).

$$SLATAH = \frac{1}{N}\sum_{i=1}^{N}\frac{T_{si}}{T_{ai}} \quad PDM = \frac{1}{M}\sum_{j=1}^{M}\frac{C_{dj}}{C_{rj}} \tag{1}$$

SLATAH and PDM are calculated with Eq. (1). Where N is the number of PMs, T_{si} is the total time during which the ith PM has experienced the utilization of 100%, T_{ai} is the total time during which the ith PM is in the active mode, M is the number of VMs, C_{dj} is the estimate of the PD of the jth VM caused by migration and C_{rj} is the total CPU capacity requested by the jth VM during its lifetime. From both of the above, SLAV is defined by Eq. (2).

$$SLAV = SLATAH \cdot PDM \tag{2}$$

The value of SLAV is a real number between 0 and 1, usually expressed as a percentage. A larger value indicates a more severe SLAV.

3.1.3 MC Modeling

MC is highly dependent on data center conditions, such as network architecture, bandwidth usage, the application itself (i.e., how many pages the application updates during migration), and VM memory usage (i.e., memory read or write intensity). MC includes migration latency and downtime.

Migration latency is related to the VM size, workload characteristics, and network transmission rate. All memory pages are copied to the target PM in the pre-copy phase while the VM remains running. During the phase, some pages will be modified many times and must be copied many times to ensure data consistency between source PM and target PM [45]. The continuous formation of dirty pages means that pre-copy should be carried out in several rounds, and the dirty pages to be transmitted during one round should be generated in the previous round. There are four conditions used to terminate iterations: (1) the number of iterations exceeds a pre-defined threshold ($n > n_{th}$), (2) the total amount of memory transmitted exceeds a given value ($m > m_{th}$), (3) the number of dirty pages in the previous round drops below a given value ($p < p_{th}$) and (4) the dirty page rate exceeds a given

bandwidth ($d > B_{th}$) [46]. At the stop-and-copy phase, the VM is hung, and all remaining dirty pages and CPU registers are copied onto the target PM.

Let variable d be the memory dirtying rate during migration and constant B be the memory transmission rate during migration. We observed that migration latency highly depends on (1) the total amount of memory V_{mig} that has to be transmitted from the source to the target PM and (2) the memory transmission rate B. It is proportional to V_{mig} and is inversely proportional to B, and we can calculate it as:

$$T_{mig} = \frac{V_{mig}}{B} \tag{3}$$

According to [47], the amount of memory transmitted in round i is the following as Eq. (4):

$$v_{mig,i} = \begin{cases} v_{mem} & i = 0 \\ d \cdot l \cdot t_{i-1} & 0 < i < n \end{cases} \tag{4}$$

where v_{mem} is the memory size of the VM when migration starts, l is the page size and t_{i-1} is the duration of round $i-1$. Consequently, V_{mig} is calculated as in Eq. (5).

$$V_{mig} = \sum_{i=0}^{n} v_{mig,i} \tag{5}$$

where n is the number of rounds in the pre-copying phase, and which is plus 1 due to that all remaining dirty pages are transmitted at the final stop-and-copy phase.

A subset of memory pages called writable working sets (WWS) will be updated, much faster than memory transfer speed. These pages are typically used to run process stacks, local variables, and buffers. The hottest page should be skipped until the VM hangs. Therefore, the amount of memory transmitted in each round should be subtracted from the size of WWS. Reference [45] believed that the number of the hottest pages is almost proportional to the number of dirty pages in the previous round, and proposes that the size of WWS in round i is calculated as:

$$W_i = \gamma \cdot v_{mig,i} \tag{6}$$

where γ is a correlation coefficient.

Downtime is another part of MC. The VM is hung throughout the stop-and-copy phase and duplicates the remaining dirty pages to the target PM. After the migration process is completed, the VM is resumed on target PM. According to [45], downtime T_{down} is defined as in Eq. (7).

$$T_{down} = \frac{d \cdot l \cdot t_n}{B} \tag{7}$$

where t_n is the length of time of the last pre-copy round. Both T_{mig} and T_{down} depend on t_i ($0 \le i \le n$), namely, how much time is spent in round i and the total number of rounds n before the final stop-and-stop phase. In addition, the value of n is affected by termination conditions of iteration. We cannot calculate them directly according to Eqs. (3)–(7). We propose a more practical model to calculate MCs. We describe the pseudo-code for the model in *Algorithm* 1.

We propose an overall formula that synthesizes both migration latency and downtime to direct migration choice as follows:

$$MC = a \cdot T_{mig} + b \cdot T_{down} \tag{8}$$

Algorithm 1: Performance model for migration

Input: $n_{th}, m_{th}, p_{th}, v_{mem}, B, d, l, \gamma$ ***Output:*** $T_{mig}, T_{down}, V_{mig}$
Initialize: $v_0 \leftarrow v_{mem}$, $V_{mig} \leftarrow 0$, $T_{mig} \leftarrow 0$, $T_{down} \leftarrow 0$
1 **for** $i = 0$ *to* n_{th} **do**
2 $t_i \leftarrow v_i / B$
3 $v_{i+1} \leftarrow d \cdot l \cdot t_i - \gamma \cdot v_i$
4 $T_{mig} + \leftarrow t_i$
5 $V_{mig} + \leftarrow v_i$
6 **if** $V_{mig} > m_{th}$ ***or*** $v_i / l < p_{th}$ ***or*** $d > B_{th}$
7 break
8 **end if**
9 **end for**
10 $T_{down} \leftarrow v_{i+1} / B$
11 $T_{mig} + \leftarrow T_{down}$

3.1.4 RW Modeling

If no VMs utilize the remaining resources available, we believe those resources are wasted. Minimizing resource wastage in VM consolidation is one of our objectives. Table 3 sums up the notations in our approach. As depicted in Eq. (9), we define w_j^r as a vector representing the RW of PM j in term of resource r.

Table 3: Notations used in RW modeling

Notation	Signification
N	Number of VMs.
M	Number of PMs.
R	Number of dimensions of resource.
x_{ij}	VM-to-PM assignment Boolean variable, which is equal to 1 if VM i is assigned to PM j, 0 otherwise.
y_j	PM selection Boolean variable, which is equal to 1 if PM j is active, 0 otherwise.
C_j^r	Non-negative capacity of PM j in term of resource r.
D_i^r	Non-negative demand of VM i in term of resource r.
w_j^r	RW of PM j in term of resource r.

$$w_j^r = y_j \cdot \frac{(C_j^r - \sum_{i=1}^{N}(x_{ij} \cdot D_i^r))}{C_j^r} \tag{9}$$

To quantify RW on multiple dimensions, we define the RW of PM j as the magnitude of w_j^r as Eq. (10). The total RW is defined in Eq. (11).

$$W_j = |w_j^r| \tag{10}$$

$$W = \sum_{j=1}^{M} W_j \tag{11}$$

3.1.5 Optimization Formulation

We aim to minimize EC, MC, and RW and maximize performance simultaneously. In the above model, we use the predicted values of VM resource demands (i.e., CPU demand prediction for EC, memory demand prediction for MC, and CPU and memory demand predictions for performance and RW) to calculate the objective function value.

Suppose $f_i(x)$ is the ith objective function entailed by a consolidation solution x. Let $f_1(x)$ be the function of EC described by [43], $f_2(x)$ be the function of performance defined by Eq. (2), $f_3(x)$ be the function of MC described by Eq. (8), and $f_4(x)$ be the function of RW described by Eq. (11). We can formulate VM consolidation problem as:

$Minimize: \ [f_1(x), f_2(x), f_3(x), f_4(x)]$

- Constraint 1: $\sum_{j=1}^{M} x_{ij} = 1$. This means that one VM can be on only one PM.
- Constraint 2: $\sum_{i=1}^{N} D_i^r \cdot x_{ij} \leq C_j^r y_j$. This guarantees that VM resource demands are not beyond PM capacity.
- Constraint 3: $y_j, x_{ij} \in \{0, 1\}$.

3.2 The Suggested Prediction Approach

3.2.1 ARIMA Prediction Method

This paper uses the ARIMA model to map linear patterns in time series. ARIMA is a prediction and analysis model of time series. It can provide fast predictions [48]. It is very suitable for the prediction of the scaling of cloud resources. Through difference, it can transform the nonstationary time series into a stationary time series, and then apply AR and MA techniques to the time series. Reference [23] gives ARIMA model as follows:

$$y_t = \alpha_0 + \beta_1 y_{t-1} + \beta_2 y_{t-2} + \cdots + \beta_p y_{t-p} + \epsilon_t - \alpha_1 \epsilon_{t-1} - \alpha_2 \epsilon_{t-2} - \cdots - \alpha_q \epsilon_{t-q} \tag{12}$$

where $\beta_i\,(i = 1, 2, \ldots, p)$ is AR coefficient at lag p, $\alpha_i\,(i = 0, 1, \ldots, q)$ is MA coefficient at lag q, and ϵ_t is the forecast error at period t, and y_t is the actual value. The integers p and q are the orders of the model. In this paper, the Box-Jenkins method [49] fits ARIMA (p, d, q) model.

3.2.2 SVR Prediction Method

SVR is strong at extracting nonlinear patterns of time series. It optimally maps nonlinear input data to a higher-dimensional feature space through the transformation of the kernel function and then performs LR in feature space.

We give the original input training set as: $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, where $x_i \in X \subset R^n$, $y_i \in Y \subset R$, $i = 1, 2, \ldots, n$, n is a total number of samples. We need to find a separating hyperplane, such as $f(x) = w^T x + b = 0$, where w is normal vector, and b is a bias. We can write the distance from point x to hyperplane w, b as:

$$r = \frac{|w^T x + b|}{||w||} \tag{13}$$

We write the margin width of the decision boundary as:

$$\gamma = \frac{2}{||w||} \tag{14}$$

We aim to minimize the number of samples that do not fulfill the constraint of decision boundary as much as possible. Thus, we write the optimization problem as:

$$\min_{w,b} \frac{1}{2}||w||^2 + C\sum_{i=1}^{n} l_\epsilon(f(x_i) - y_i) \tag{15}$$

where C is regularization constant, l_ϵ is the loss function, and defined as:

$$l_\epsilon(f(x_i) - y_i) = \begin{cases} 0 & |f(x_i) - y_i| \le \epsilon \\ |f(x_i) - y_i| - \epsilon & otherwise \end{cases} \tag{16}$$

To relax the samples' margin requirements, slack variables ξ_i and ξ_i^* are led into. We can display the optimization of SVR with slack variables below:

$$\min_{w,b,\xi_i,\xi_i^*} \frac{1}{2}||w||^2 + C\sum_{i=1}^{n} \xi_i + \xi_i^* \tag{17}$$

$$\text{s.t. } f(x_i) - y_i \le \epsilon + \xi_i, \tag{18}$$

$$y_i - f(x_i) \le \epsilon + \xi_i^*,$$

$$\xi_i \ge 0,\ \xi_i^* \ge 0,\ i = 1, 2, \ldots, n.$$

Using Lagrange multipliers forms the dual optimization problem given by:

$$\max_{\alpha,\hat{\alpha}} \sum_{i=1}^{n} y_i(\widehat{\alpha}_i - \alpha_i) - \epsilon(\widehat{\alpha}_i + \alpha_i) - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\widehat{\alpha}_i - \alpha_i)\left(\widehat{\alpha}_j - \alpha_j\right)x_i^T x_j \tag{19}$$

$$\text{s.t.} \sum_{i=1}^{n}(\widehat{\alpha}_i - \alpha_i) = 0, \tag{20}$$

$$0 \le \alpha_i, \widehat{\alpha}_i \le C,\ i = 1, 2, \ldots, n.$$

where α and $\hat{\alpha}$ are Lagrange multipliers. After solving α_i and $\widehat{\alpha}_i$, we express prediction function $f(x)$ as:

$$f(x) = \sum_{i=1}^{n}(\widehat{\alpha}_i - \alpha_i)x_i^T x + b \tag{21}$$

Let function $\varphi(x)$ map n-dimensional vector x into m-dimensional vector x, where $m > n$. We can write $f(x)$ as:

$$f(x) = \sum_{i=1}^{m}(\widehat{\alpha}_i - \alpha_i)\varphi(x_i)^T\varphi(x) + b \tag{22}$$

By using the kernel function, we can convert the calculation in the feature space into the calculation in the original space. The kernel function is defined as:

$$k\left(x_i, x_j\right) = <\varphi(x_i), \varphi\left(x_j\right)> = \varphi(x_i)^T\varphi(x_j) \tag{23}$$

Therefore, we can overwrite $f(x)$ as:

$$f(x) = \sum_{i=1}^{m}(\widehat{\alpha}_i - \alpha_i)k(x, x_i) + b \tag{24}$$

3.2.3 SMO

SMO is an algorithm usually used to solve optimization problems during the training process of SVM. This paper exploits it to solve the optimization problem in the training process of SVR. We

describe the dual problem of SVR in Eq. (19). Let $\lambda_i = \widehat{\alpha}_i - \alpha_i,\ i = 1, 2, \ldots, m$, consider feature mapping and define $k_{ij} = \phi(x_i)^T \phi(x_j) = k(x_i, x_j)$, and we can express this problem as a minimization problem as below:

$$\min_{\lambda} \sum_{i=1}^{m} \epsilon(|-\lambda_i| - (y_i\lambda_i) + \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \lambda_i\lambda_j k_{ij} \tag{25}$$

$$\text{s.t.} \sum_{i=1}^{m} \lambda_i = 0, \tag{26}$$

$$-C \le \lambda_i \le C,\ i = 1, 2, \ldots, m.$$

If variables $\lambda_3, \ldots, \lambda_m$ are regarded as constants, and then $\lambda_1 + \lambda_2 = -\sum_{i=3}^{m} \lambda_i = \varsigma$, assuming that variable $\lambda_i \le 0$ (otherwise, the result remains the same), we can simplify the objective function as:

$$W(\lambda_1, \lambda_2) = \frac{1}{2}\lambda_1^2 k_{11} + \frac{1}{2}\lambda_1^2 k_{22} + \lambda_1\lambda_2 k_{12} + \sum_{j=3}^{m} \lambda_1\lambda_j k_{1j} + \sum_{j=3}^{m} \lambda_2\lambda_j k_{2j} - \epsilon(\lambda_1 + \lambda_2) - y_1\lambda_1 - y_2\lambda_2 \tag{27}$$

On substituting $\lambda_1 = \varsigma - \lambda_2$ to Eq. (27), keep only the items with λ_2, we can overwrite Eq. (27) as:

$$W(\lambda_2) = \frac{1}{2}\lambda_2^2 k_{11} - \varsigma\lambda_2 k_{11} + \frac{1}{2}\lambda_2^2 k_{22} + \varsigma\lambda_2 k_{12} - \lambda_2^2 k_{12} - \sum_{j=3}^{m} \lambda_2\lambda_j k_{1j} + \sum_{j=3}^{m} \lambda_2\lambda_j k_{2j} + y_1\lambda_2 - y_2\lambda_2 \tag{28}$$

Find the partial derivative of function W concerning λ_2 and let the derivative be zero as below:

$$\frac{\partial W}{\partial \lambda_2} = (k_{11} + k_{22} - 2k_{12})\lambda_2 - \left(\varsigma k_{11} - \varsigma k_{12} + \sum_{j=3}^{m} \lambda_j k_{1j} - \sum_{j=3}^{m} \lambda_j k_{2j} + y_2 - y_1\right) = 0 \tag{29}$$

Let $v_i = \sum_{j=3}^{m} \lambda_j k_{ij} = f(x_i) - \sum_{j=1}^{2} \lambda_j k_{ij} - b, i = 1, 2, \ldots, m$, and substitute $\varsigma = \lambda_1 + \lambda_2$, there are:

$$(k_{11} + k_{22} - 2k_{12})\lambda_2 = (k_{11} + k_{22} - 2k_{12})\lambda_2 + E_2 - E_1 \tag{30}$$

$$E_i = f(x_i) - y_i,\ i = 1, 2, \ldots, m. \tag{31}$$

Consequently, suppose that the solution of the last round is λ_2^{old} and the new solution of this round without clipping is $\lambda_2^{new,unc}$, we get:

$$\lambda_2^{new,unc} = \lambda_2^{old} + \frac{E_2 - E_1}{k_{11} + k_{22} - 2k_{12}} \tag{32}$$

By clipping the original solution, we get:

$$\lambda_2^{new} \begin{cases} H & \lambda_2^{new,unc} > H \\ \lambda_2^{new,unc} & L < \lambda_2^{new,unc} < H \\ L & \lambda_2^{new,unc} < L \end{cases} \tag{33}$$

where $L = \lambda_1^{old} + \lambda_2^{old} - C, H = C$. Because of the linear relationship between variables λ_1 and λ_2, we can get:

$$\lambda_1^{new} = \lambda_1^{old} + \lambda_2^{old} - \lambda_2^{new} \tag{34}$$

After optimizing the two variables, we need recalculate threshold b and error E_i. According to Karush-Kuhn-Tucker (KKT) conditions, when $-C < \lambda_i < C$, there are $\xi_i = 0$ or $\xi_i^* = 0$ and Eq. (35)

holds:

$$\begin{cases} f(x_i) - y_i - \epsilon = 0 & -C < \lambda_i < 0 \\ y_i - f(x_i) - \epsilon = 0 & 0 < \lambda_i < C \end{cases} \tag{35}$$

If $-C < \lambda_1^{new} < 0$,

$$b = y_i + \epsilon - \sum_{j=1}^{m} \lambda_j k_{ij} \tag{36}$$

Therefore,

$$b_1^{new} = y_1 + \epsilon - \sum_{i=3}^{m} \lambda_i k_{i1} - \lambda_1^{new} k_{11} - \lambda_2^{new} k_{21} \tag{37}$$

Due to:

$$E_1 = f(x_1) - y_1 = \sum_{i=3}^{m} \lambda_i k_{i1} + \lambda_1^{old} k_{11} + \lambda_2^{old} k_{21} + b^{old} - y_1 \tag{38}$$

We get:

$$b_1^{new} = -E_1 + \epsilon + \left(\lambda_1^{old} - \lambda_1^{new}\right) k_{11} + \left(\lambda_2^{old} - \lambda_2^{new}\right) k_{21} + b^{old} \tag{39}$$

If $-C < \lambda_2^{new} < 0$, we get:

$$b_2^{new} = -E_2 + \epsilon + \left(\lambda_1^{old} - \lambda_1^{new}\right) k_{12} + \left(\lambda_2^{old} - \lambda_2^{new}\right) k_{22} + b^{old} \tag{40}$$

In the same way, if $0 < \lambda_1^{new} < C$ and $0 < \lambda_2^{new} < C$, we get:

$$b_1^{new} = -E_1 - \epsilon + \left(\lambda_1^{old} - \lambda_1^{new}\right) k_{11} + \left(\lambda_2^{old} - \lambda_2^{new}\right) k_{21} + b^{old} \tag{41}$$

$$b_2^{new} = -E_2 - \epsilon + \left(\lambda_1^{old} - \lambda_1^{new}\right) k_{12} + \left(\lambda_2^{old} - \lambda_2^{new}\right) k_{22} + b^{old} \tag{42}$$

We take the new value of b as:

$$b^{new} = \frac{b_1^{new} + b_2^{new}}{2} \tag{43}$$

We can calculate E_i as:

$$E_i = \sum_{i=1}^{m} \lambda_i k_{ij} + b^{new} - y_i \tag{44}$$

The SMO process is repeated until all variables λ_i $(i = 1, 2, \ldots, m)$ are obtained.

4 Prediction-Based Multi-Objective VM Consolidation

4.1 Resource Usage Prediction

Dealing with dynamic resource requirements is necessary to minimize the total EC, MC, PD, and RW in a cloud data center. Knowing resource requirements in advance and buying time to calculate the optimization solution requires predicting future resource demand based on historical resource demand. This paper proposes a hybrid prediction approach based on ARIMA and SVR (HPAS) to predict VM resource demand using available resource demand history. We depict the framework of our prediction approach in Fig. 1. We utilize the resource demand trace generated from Microsoft Azure. We divide the resource demand data into two parts at a ratio of 6 to 4. For ARIMA, we train and test

it to predict the future resource demand of VM. For SVR, we carry out the preprocessing phase. Then, we use the prepared and normalized resource demand data to train and test the SVR model. The SVR model predicts future errors. We depict the prediction model workflow in Fig. 2. We use an ARIMA model to predict the resource demand initially. As mentioned, the ARIMA model can only capture linear time series features. We use additional ARIMAs to calculate the errors but only consider linear attributes. History of residuals from multiple time slots (i.e., slots $t-1,\ldots,t-n+1$) generated by ARIMAs is used to predict future error (i.e., slot t). Therefore, we use the residuals as inputs of the SVR model. Adding the forecasts of the ARIMA model (i.e., X_t) and the SVR model (i.e., $\widehat{E}_t$) obtains the final prediction ($\widehat{O}_t$).

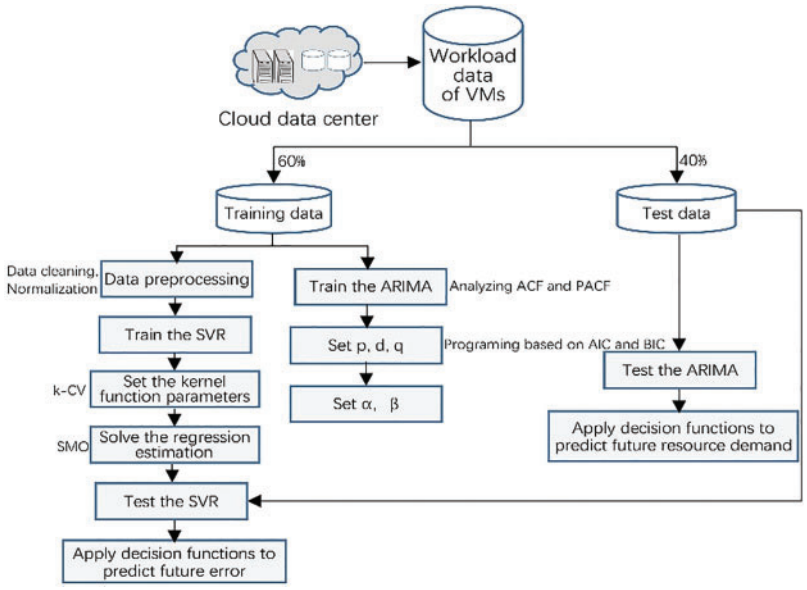


Figure 1: The prediction framework of resource demand

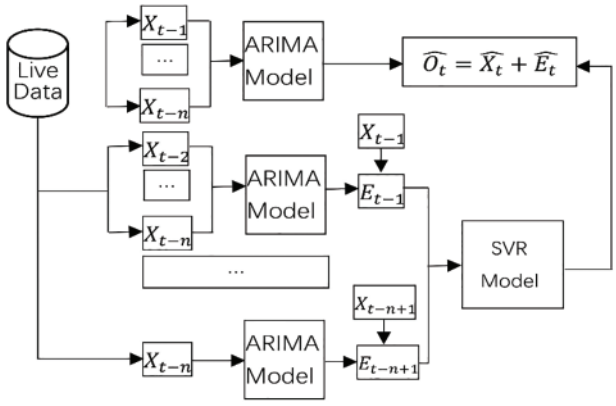


Figure 2: The prediction model workflow

From the above, we can see that in HPAS, ARIMA fully captures the linear features of live data from the past n time slots, but inevitably overlooks nonlinear features. We accumulate the error of using ARIMA only for prediction in the past $n-1$ time slots through $n-1$ additional ARIMAs. These errors

contain sufficient nonlinear features of live data and serve as inputs for SVR. Therefore, SVR can fully capture the nonlinear features of live data from the past $n - 1$ time slots. SVR predicts the nonlinear part of VM resource demand in time slot t and finally revises the initial ARIMA prediction of VM resource demand for time slot t. Consequently, HPAS results in better predictive performance than only one of ARIMA and SVR.

We present the pseudo-code for predicting resource demand per VM in *Algorithm* 2. Initially, we enter the dataset into the suggested model (line 1). We partition the dataset into two parts: the training part and the testing one (line 2). The training part estimates the ARIMA model by analyzing AC and partial PAC functions and programming (lines 3–4). Next, we preprocess the generated resource demand data for SVR (line 5). We use the boxplot method to detect outliers, remove them, and use interpolation to fill in missing values to implement regression smoothly. We then apply the K-fold Cross Validation (k-CV) strategy to select the best parameters of the kernel function (i.e., γ, C, and ε) used by SVR (line 6). We perform the training process with the normalized resource demand data and the suitable parameters (line 7). In the testing phase, the test evaluates the suggested model's prediction accuracy (line 8). Finally, we obtained the prediction result (lines 9–13).

Algorithm 2: HPAS

Input: s*tateHistory* // State history of VM
Output: prediction of VM resource demand in next time slot
1 Enter the workload dataset $x_i \in X \subset R^n$ (VM resource demand) to the model
2 Divide the dataset into training part and testing one
3 Determine p, d and q by analyzing ACF and $PACF$ on training set
4 Determine the parameters α and β of $ARIMA$ by programing on training set
5 Preprocess data (cleaning and normalizing) for SVR
6 Apply k-CV strategy to find the best parameters of the kernel function used by SVR
7 Execute training process of the SVR model on training dataset using Algorithm 3
8 Test the model on testing dataset
9 Predict $\widehat{y_t}$ according to Eq. (12) for s*tateHistory*
10 Predict $z_1, z_2, \ldots, z_{stateHistory.size()}$ according to Eq. (12) for each state in s*tateHistory*
11 Get error history *errHistory* according to s*tateHistory* and $z_1, z_2, \ldots, z_{stateHistory.size()}$
12 Predict $\widehat{E_t}$ according to Eq. (24) for *errHistory*
13 **return** $\widehat{y_t} + \widehat{E_t}$

We performed the training process to get the best parameters of the SVR model using *Algorithm* 3. The first line is initialization. We then pick two variables (line 4). The first variable λ_1^k selects the sample with the most severe violation of KKT conditions in the training set. If all samples meet the KKT condition, select a sample randomly. After selecting λ_1^k, the second variable λ_2^k selects the sample with enough change for $|E_1 - E_2|$. Next, we optimize λ_1^k and λ_2^k repeatedly using Eqs. (32)–(34), (39)–(44) until we reach maximum iteration or satisfy the termination condition (lines 5–10). Finally, we complete the optimization of all the variables $\lambda_i (i = 1, \ldots, n)$ and b.

4.2 VM Consolidation

The main goal of this work is to consolidate VMs into PMs based on the prediction result. We propose an efficient prediction-based multi-objective VM consolidation algorithm that aims to minimize the total EC, PD, MC, and RW in a cloud data center, depending on future resource demand.

Algorithm 3: The estimate of the parameters of SVR model

Input: Training data $T = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}, x_i \in X \subset R^n, y_i \in Y \subset R, i = 1, 2, \ldots, n$, *precision* ε

Output: The values of parameters λ and b

$i = 0$, $\lambda^0 = 0$, $b^0 = 0$
While ($i < n$ or stopping criterion is not met)
  $k = 0$
  Select two optimization variables as λ_1^k, λ_2^k
  While (No. iterations or stopping criterion is not met)
    Calculate $\lambda_2^{new,unc}$ according to Eq. (32)
    Calculate λ_2^{k+1} according to Eq. (33)
    Calculate λ_1^{k+1} according to Eq. (34)
    Calculate b^{k+1} and E_i according to Eqs. (39)–(44)
  $k = k + 1$
  $i = i + 2$
return λ and b

4.2.1 Overload/Underload PM Detection

Algorithm 4 represents the overload/underload PM detection procedure. We used a warm-up window to ensure computational tractability [44]. This paper's first *warmUpWindowSize* time slots were regarded as a warm-up phase (line 5). During the warm-up phase, we calculate the CPU usage of PM and choose overload/underload PM (lines 1–6, lines 11–12). After the warm-up phase, we use the prediction result of *Algorithm 2* to calculate the CPU usage of PM and choose over/lower utilized PMs (line 8).

Algorithm 4: Overload/underload host detection

Input: *host*, *upperThreshold*/*lowerThreshold*
Output: Boolean
1 *vmList* = *host.getVmList( )*
2 *totalMips* = *0*
3 **FOR** *v* **IN** *vmList*
4 *stateHistory* = *v.getStateHistory( )*
5 **IF** *(stateHistory.size( ) <= warmUpWindowSize)*
6 *totalMips* += *v. getCurrentMips( )*
7 **ELSE**
8 *totalMips* += *HPAS(stateHistory)*
9 **END**
10 **END**
11 *hostUtilization* = *totalMips*/*host.getTotalMips*()
12 **return** *hostUtilization* $\geq$ *upperThreshold* or *hostUtilization* $\leq$ *lowerThreshold*

All the VMs of underload hosts need to migrate, and we shut all the underload hosts down to save EC and decrease RW.

4.2.2 VM Selection

We chose the MMT policy for VM selection. MMT policy selects a VM that requires the minimum time to complete a migration. The migration time is the amount of RAM the VM utilizes divided by the spare network bandwidth available for the PM.

4.2.3 VM Placement

We perform multi-objective optimization VM placement to reduce EC, PD, MC, and RW simultaneously. We develop a multi-objective optimization VM placement (MOVP) approach based on the Nsga2 algorithm.

Pareto dominance is widely used in the search for a multi-objective optimization solution. Consider two solutions $u, v \in X$, u dominate v, denoted as $u > v$, if and only if $f(u)$ is superior to or equal to $f(v)$ in each objective function and strictly superior in at least one objective function. It is called solution x is non-dominant for set X, if there is no dominant member in X. The solution of multi-objective optimization is called the *Pareto optimal set*. No other solution dominates the solutions in the Pareto optimal set. The corresponding set of objective values defines the *Pareto frontier*.

Each placement solution is a vector x as described in Section 3.1.5. In MOVP, the objective functions are defined in Sections 3.1.1–3.1.4, as described in Section 3.1.5. It is worth noting that when calculating the values of these objective functions, we use the resource demand values (i.e., CPU utilization and Memory usage) of the VM predicted through HPVS. Using the predicted value of resource demands instead of the current value for MOVP can give the placement solution good timeliness and practical value. We present the pseudo-code for VM placement in *Algorithm* 5, which inputs the PM, VM sets, and initial placement solution x_0. Initially, we built a random parent population P_0 (line 1). We use selection, crossover, and mutation operators to create an offspring population Q_0 of size *Pop_size* (lines 3–5). In each generation, a merged population R_t of size $2 \cdot$*Pop_size* is formed to prevent individuals from being lost in the next generation in line 7. Next, we sort the population R_t in non-domination order and arrange individuals into different frontiers (line 9). Function *crowding − distance − assignment* is used to ensure the diversity of individuals (line 10). We choose individuals from several frontiers (frontier order from low to high, i.e., $\mathcal{F}_1, \mathcal{F}_2$, and so on) for new population P_{t+1}. To determine exactly *Pop_size* individuals, we sort the individuals of the last frontier $\mathcal{F}_i$ using the crowd-comparison operator in descending order (lines 14, 21) and select enough individuals (lines 15–22). Then, we generate offspring Q_{t+1} of P_{t+1} for the next iteration (lines 23–28). Function *decisionMaking* is an operator that chooses favorite individuals from current Pareto optimal set P_t (line 30).

Algorithm 5: MOVP

Input: $P = \{p_1, \ldots, p_M\}, V = \{v_1, \ldots, v_N\}, x_0$

Output: x

$P_0 \leftarrow initializePopulation\,(Pop\_size)$; $R_0, Q_0 \leftarrow \varnothing, t \leftarrow 0, x \leftarrow x_0$
for $i \leftarrow 1$ *to* $\dfrac{Pop\_size}{2}$ **do**
  $parents \leftarrow selection\,(P_0)$
  $offspring \leftarrow crossover\,(parents)$
  $offspring \leftarrow mutation\,(offspring)$
  $Q_0 \leftarrow Q_0 \cup offspring$
end for
while $(t \leq Max\_Generatios)$
  $R_t \leftarrow P_t \cup Q_t$
  $\mathcal{F} \leftarrow fast - non - dominated - sort(R_t)$
  $crowding - distance - assignment(F)$
  $P_{t+1} \leftarrow \varnothing$ *and* $i \leftarrow 1$
  if $|\mathcal{F}_i| > Pop\_size$
    $sort\,(\mathcal{F}_i)$
    $P_{t+1} \leftarrow P_{t+1} \cup \mathcal{F}_0[1 : Pop\_size]$
  else
    while $|P_{t+1}| + |\mathcal{F}_i| \leq Pop\_size$
      $P_{t+1} \leftarrow P_{t+1} \cup \mathcal{F}_i$
      $i \leftarrow i + 1$
  end if
  $sort\,(\mathcal{F}_i)$
  $P_{t+1} \leftarrow P_{t+1} \cup \mathcal{F}_i[1 : (Pop\_size - |P_{t+1}|)]$
  for $i \leftarrow 1$ *to* $\dfrac{Pop\_size}{2}$ **do**
    $parents \leftarrow selection\,(P_{t+1})$
    $offspring \leftarrow crossover\,(parents)$
    $offspring \leftarrow mutation\,(offspring)$
    $Q_{t+1} \leftarrow Q_{t+1} \cup offspring$
  end for
  $t \leftarrow t + 1$
$x \leftarrow decisionMaking(P_t)$
return x

Fast − non − dominated − sort and *crowding − distance − assignment* are the two fundamental operations of Nsga2, which will not be further elaborated here to save space.

5 Evaluation

This section evaluates the proposed approach based on the results obtained. First, the prediction accuracy of the suggested prediction model is evaluated, and then the prediction-based consolidation of the proposed approach is evaluated.

5.1 Experimental Setup

We performed a simulation on CloudSim version 5.0. We simulated a cloud data center with 50 heterogeneous PMs in five configurations. Table 4 describes the characteristics of these machines. VMs are supposed to correspond to Amazon EC2 instance types. The experiments use four types of VMs, as shown in Table 5. After each VM consolidation step, VM resource demand changes according to workload data.

Table 4: Configuration of PMs

Server	CPU model	Cores	Frequency (MHZ)	RAM (GB)
IBM system x3200 M3	Intel Xeon X3480	4	3067	8
IBM system x3250 M3	Intel Xeon X3480	4	3067	8
IBM x3450	Intel Xeon E5462	8	2800	16
IBM system x3550 M3	Intel Xeon X5675	12	3067	16
IBM system x3500 M4	Intel Xeon E5-2680	16	2700	24

Table 5: VM types (EC2 VM types)

VM type	CPU (MIPS)	RAM (GB)
High-CPU medium instance	2500	0.87
Extra large instance	2000	3.75
Small instance	1000	1.74
Micro instance	500	0.613

We use the parameters of HPAS, C, ε, and γ to adjust the tradeoff between model complexity and training error, define the accuracy requirement, and facilitate the mapping of input data, respectively [19]. We obtain a better performance of HPAS when C is 150, ε is 0.01, and γ is 0.0001.

Table 6 presents MOVP's parameters. The first seven are used by Algorithm 1, which MOVP invokes.

Table 6: The parameters of MVOP

Notation	Description	Value
N_TH	Pre-copy rounds threshold	20
M_TH	Amount of memory transmitted threshold	1000 MB
P_TH	Number of dirty pages threshold	10000
B_TH	Bandwidth threshold	0.6
B	Bandwidth	1 Gbps
l	Page size	4 KB
γ	Correlation coefficient	0.001

(Continued)

Table 6 (continued)

Notation	Description	Value
CrossoverRate	Crossover rate	0.6
MutationRate	Mutation rate	0.1
Max_Generatios	Max iteration generations	100
popSize	Population size	30

We performed each experiment 10 times for our proposed approach, and reported the mean values for different metrics.

5.2 Workload Model

It is essential to conduct experiments using workload traces from a real system. We used MicroSoft Azure trace as resource demand data. This data trace contains information about every VM running on Azure from 16 November 2016, to 16 February 2017. The data-trace corresponds to tens of millions of VMs from tens of thousands of first- and third-party users.

In Sections 5.3 and 5.4, we conducted comparative experiments between the proposed and benchmark algorithms using diverse datasets obtained from Azure Trace. These datasets' average CPU load rates are low to high, and their memories are relatively stable. We give the characteristics of all the datasets in Table 7.

Table 7: Dataset characteristic

Dataset	Average CPU load rate (%)	Memory usage (GB)
1	16.7	4.8
2	22.6	4
3	36	3.5
4	46.8	3.3
5	56.8	3.3
6	65.2	3.4
7	74	3.4
8	87	3.3

5.3 Evaluation of Prediction Accuracy

5.3.1 Evaluation Metrics

We use several metrics to evaluate prediction accuracy for the suggested prediction model. The evaluation metrics include Root Mean Square Error (RMSE), Mean Absolute Percentage (MAPE), Mean Absolute Error (MAE), and R-squared (R^2) [50].

5.3.2 Results

We compare HPAS' forecast accuracy with state-of-the-art approaches to its efficacy. ARIMA and SVR are two promising prediction methods for linear and nonlinear loads, respectively. The back propagation neural network (BP-NN) is one of the most widely used neural network models. Thus, we adopt them as baseline approaches.

We used 400 VMs and adopted Microsoft Azure data trace as the VM's resource demand. Table 8 shows the comparison in terms of evaluation metrics in four hours. The table indicates that ARIMA performs worst, followed by SVR and BP-NN. HPAS performs best. The low value of RMSE indicates high accuracy. The value of RMSE for HPAS is the lowest. In terms of MAPE and MAE, HPAS performs the best. As we all know, the value of R^2 close to 1 indicates a good fit of data. HPAS is closer to 1 than existing approaches.

Table 8: Comparison analysis of different prediction method

Performance parameter	HPAS	ARIMA	SVR	BP-NN
RMSE	13.8	41.51	24.36	27.02
MAPE	0.71	2.72	1.67	1.99
MAE	22.72	383	132.11	301.23
R^2	0.77	0.49	0.56	0.51

In summary, HPAS shows better prediction accuracy, followed by SVR, BP-NN, and ARIMA. HPAS' better prediction accuracy can be explained by the proposed approach's powerful capacity to capture linear and nonlinear data features simultaneously.

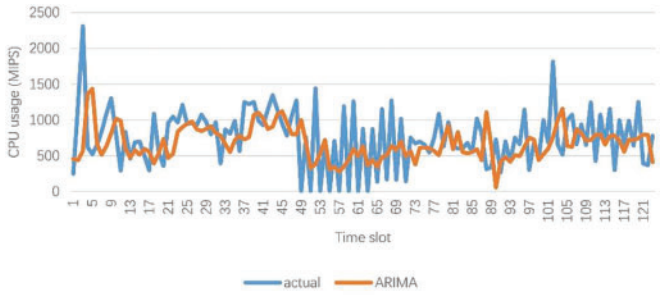
Table 9 compares diverse kernel functions used in HPAS. We want to see whether HPAS uses RBF kernel functions, which can produce better RMSE, MAPE, MAE, and R^2 results, compared to Linear, Polynomial, and Sigmoid kernel functions. If R^2 is negative (i.e., Linear and Sigmoid), it indicates that the fitting result is unreliable because the two kernel functions do not match data.

Table 9: Comparison using diverse kernel functions for HPAS

Performance parameter	RBF	Linear	Polynomial	Sigmoid
RMSE	13.8	148.85	16.56	20.45
MAPE	2.72	5.54	6.77	5.74
MAE	132.11	937.39	144.17	159.61
R^2	0.77	−2.15	0.42	−0.18

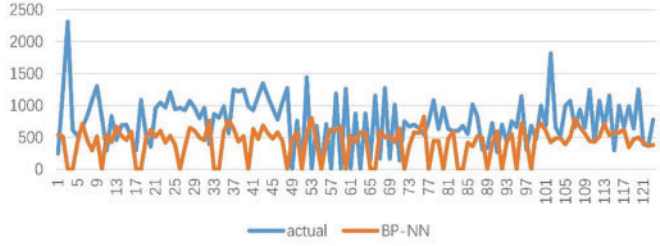
Fig. 3 provides prediction using different prediction methods. For ARIMA and SVR, the prediction is accurate for a few time slot points and cannot reflect the trend of changes in actual values, as shown in Fig. 3a,b. For BP-NN, The predicted value is generally smaller than the actual value and cannot reflect the trend of changes in actual values either, as shown in Fig. 3c. HPAS gives better prediction results, as shown in Fig. 3d. In the initial stage (the time slot before point 55), the relationship between actual and predicted values is quite chaotic due to the small number of samples in the prediction set. Afterward, as the number of samples in the prediction set increases, the predicted values reflect the trend of changes in actual values at many points, such as the sharp shapes near time

slot points 57, 61, 63, 71, 101, and 121. At some points, the predicted value almost coincides with the actual value, such as time slot points 58, 63, 67, 69, 80, 86, 88, 99, 107, 112, 118, and 124.

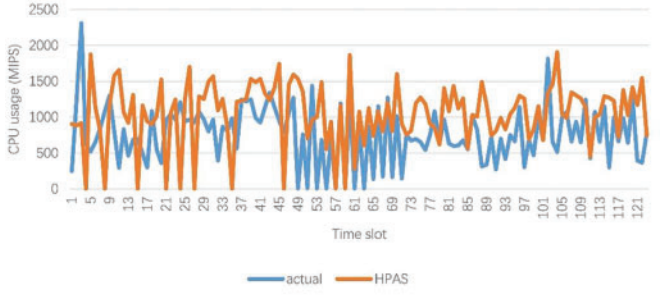


(a) Prediction based on ARIMA.

(b) Prediction based on SVR.



(c) Prediction based on BP-NN.



(d) Prediction based on HPAS.

Figure 3: Comparison of actual and predicted values using different prediction method (a–d)

We can conclude that HPAS' predictions will become more accurate as the prediction sample set grows. The results convey that HPAS produces more accurate forecasts than other prediction methods.

5.4 Evaluation of VM Consolidation

5.4.1 Evaluation Metrics

We evaluate our approach using metrics corresponding to the objectives proposed in Section 4, such as EC, SLA violation, MC, and RW. We also use the number of VM migrations and computation time as additional metrics.

5.4.2 Results

We first compare our approach with the multi-objective VM consolidation approach without prediction. We chose the VM consolidation approach adopting the Nsga2 algorithm for multi-objective optimization as a baseline approach.

Fig. 4 illustrates the metrics produced by the two approaches for six different datasets in four hours. Our consolidation approach consumed less power than Nsga2 except for datasets 4 and 6. EC was reduced by 3.1% in the worst case and by 37.7% in the best case compared to Nsga2. SLAV was significantly reduced for all the datasets. It was reduced by 11.3% in the worst case and by 88.5% in the best case compared to Nsga2. MC was reduced by 4% in the worst case and by 48.7% in the best case compared to Nsga2. The number of VM migrations was significantly reduced for all the datasets. It was reduced by 24.5% in the worst case and by 68.7% in the best case compared to Nsga2. There was a slight reduction in RW. In terms of computation time, our approach spent more time than Nsga2. It was increased by 431% in the worst case and 3% in the best case compared to Nsga2, which can be explained by the fact that our approach integrates two prediction models (i.e., ARIMA and SVR), and their training and prediction processes are time-consuming tasks, especially SVR. Another reason is that calling the ARIMA model multiple times to obtain error history data takes some time.

In summary, the proposed approach has an excellent performance in terms of EC, SLAV, MC, and number of VM migrations, which is attributed to the high predictive capacity of our approach. Our approach provides the foresight of overload and underload detection, avoiding the issue of detection results becoming invalid soon after, which also provides the foresight of VM selection, avoiding the issue of selecting too many or too few VMs, and the foresight of destination PM selection, avoiding the issue of target PM accommodating too many or too few VMs.

Next, we compare our approach with the renowned OHD approaches without prediction, such as LR, MAD, and IQR. LR predicts CPU utilizations using LR. MAD and IQR measure the workload stability by calculating the median absolute deviation and interquartile range of CPU utilization. Fig. 5 illustrates the metrics produced by all the approaches for six different datasets in two hours.

As shown in Fig. 5, our approach consumed less energy than any of the three approaches. Compared to the best of the three algorithms (i.e., LR), EC was reduced by 3.1% in the worst case and by 42.2% in the best case. SLAV was significantly reduced for all the datasets. Compared to the best of the three algorithms (i.e., MAD), SLAV was reduced by 52.2% in the worst case and by 88.8% in the best case. MC was greatly reduced except for dataset 5. Compared to the best of the three algorithms (i.e., MAD), it was reduced by 8% in the worst case and by 55.3% in the best case. The number of VM migrations was significantly reduced for all the datasets. Compared to the best of the three algorithms (i.e., MAD), it was reduced by 4.9% in the worst case and by 75.9% in the best case. The excellent performance of our approach can be explained by the fact that MAD and IQR provide adaptive utilization thresholds based on statistics and cannot learn, LR has weak predictive ability for nonlinear data, and our approach can better predict future resource demand for both linear and nonlinear data, enabling overload and underload detection more accurate.

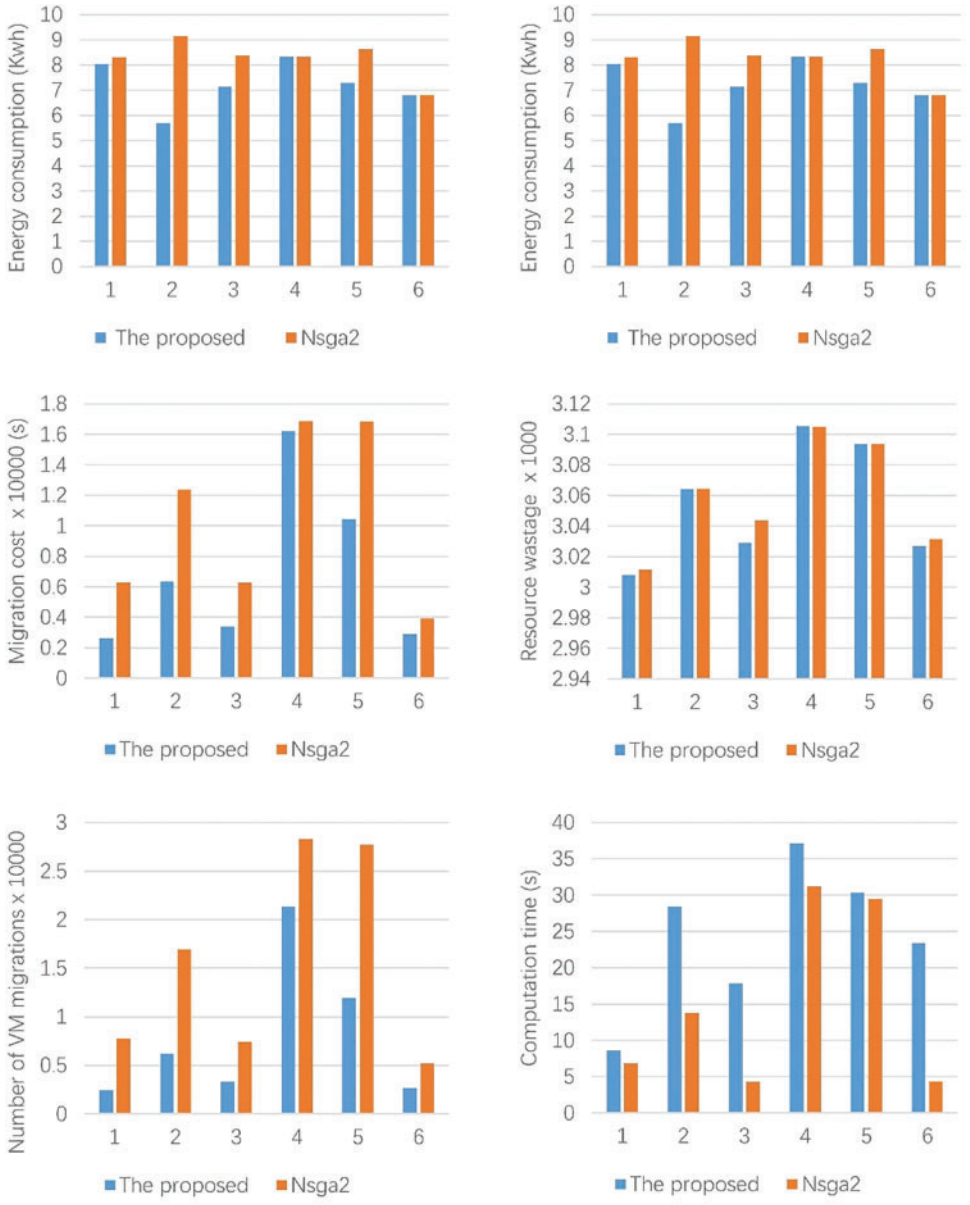


Figure 4: Comparison between the proposed approach and Nsga2 with different dataset

There is not much improvement in RW. Regarding computation time, our approach spent more time than the three approaches. This is explained by the fact that LR, MAD, and IQR only require simple calculations compared to our approach.

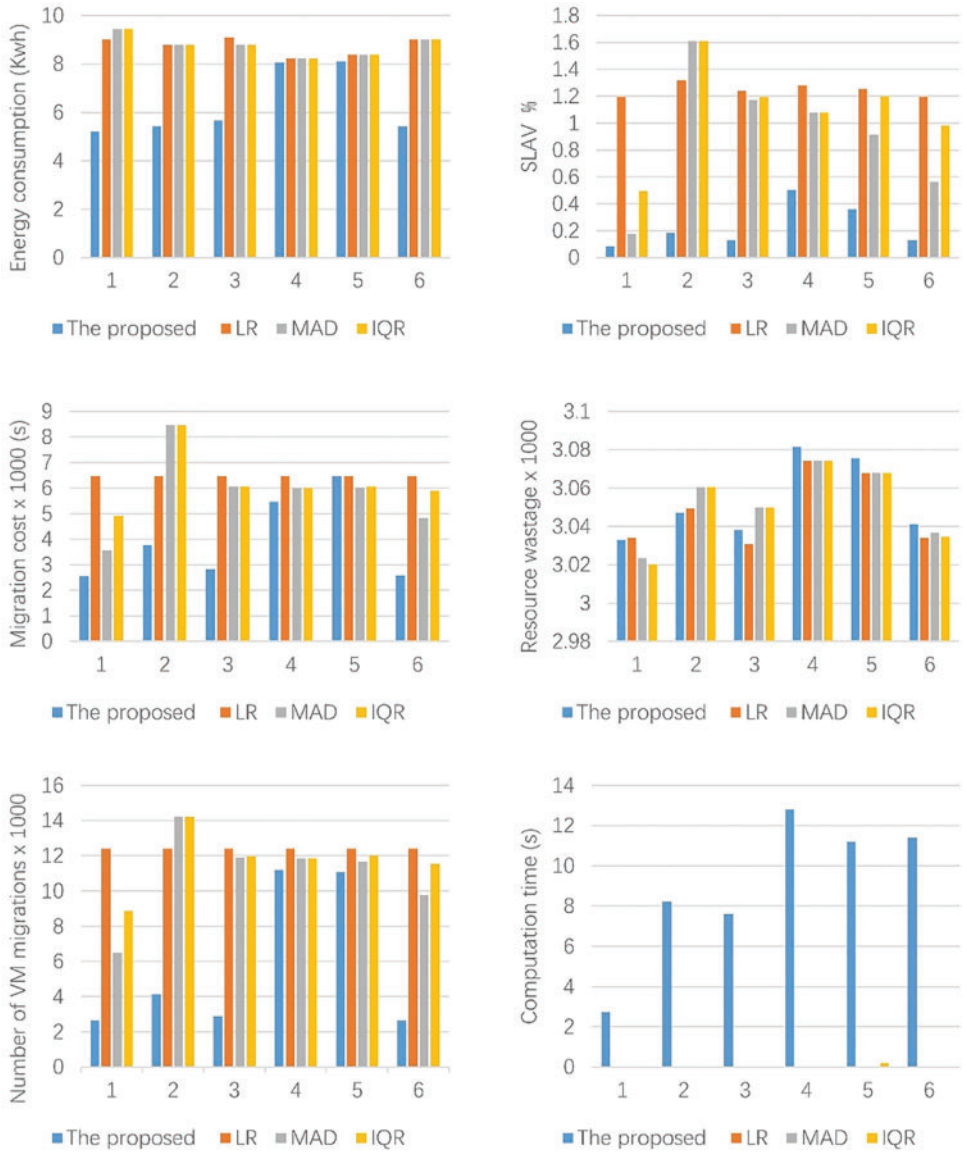


Figure 5: Comparison between the proposed approach and the existing OHD algorithms with different dataset

6 Conclusion and Future Work

In this paper, we have proposed a prediction-based multi-objective VM consolidation approach, which predicts future VM resource demand using HPAS, then consolidates VMs to PMs based on prediction results by HPAS, aiming at simultaneously minimizing the total EC, PD, MC, and RW. We have evaluated the proposed approach through simulation using real workload traces from Microsoft Azure. The results illustrate that the proposed prediction model shows better prediction accuracy, and the proposed consolidation approach overcomes the multi-objective consolidation approach without prediction and the renowned OHD approaches without prediction, such as LR, MAD, and IQR.

Our approach powerfully captures both linear and nonlinear features of data, enabling timely overload and underload detection, accurate VM and destination PM selections, and avoiding of unnecessary VM migrations.

As this work is the first step in developing the best VM consolidation approach, we only evaluated our approach in a simulation environment. In the future, we plan to study the effect of the proposed approach in a real cloud environment. In two hours, the time spent on training, predicting, and multi-objective optimization 2–13 s. The time cost is reasonable for small and medium-sized cloud data centers. However, combining grouping technology based on this work is more suitable for large-sized cloud data centers.

Acknowledgement: We sincerely appreciate Xinfeng Shu's valuable guidance and suggestions in this study. At the same time, we also appreciate the experimental equipment provided by Shaanxi Key Laboratory of Network Data Analysis and Intelligent Processing. In addition, we would like to thank all colleagues and collaborators who participated in this study for their hard work and selfless dedication, which enabled the successful completion of this study.

Funding Statement: This study was funded by Science and Technology Department of Shaanxi Province, Grant Numbers: 2019GY-020 and 2024JC-YBQN-0730. This organization did not influence the study design, data collection, analysis, interpretation, manuscript writing, or the decision to submit the manuscript for publication.

Author Contributions: Conceptualization: Xialin Liu; Methodology: Xialin Liu; Software: Xialin Liu; Validation: Jiyuan Hu; Formal Analysis: Xialin Liu; Investigation: All authors; Resources: Junsheng Wu; Data Curation: Lijun Chen; Writing–Original Draft: Xialin Liu; Writing–Review & Editing: All authors; Visualization: Lijun Chen; Funding Acquisition: Junsheng Wu. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: All data and materials included in this study are available upon request by contacting the corresponding author.

Conflicts of Interest: We declare that we have no financial and personal relationships with other people or organizations that can inappropriately influence our work, and there is no professional or other personal interest of any nature or kind in any product, service, or company that could be construed as influencing the position presented in, or the review of, the manuscript entitled "A Prediction-Based Multi-Objective VM Consolidation Approach for Cloud Data Centers".

References

[1] M. Tavana, S. Shahdi-Pashaki, E. Teymourian, F. J. Santos-Arteaga, and M. Komaki, "A discrete cuckoo optimization algorithm for consolidation in cloud computing," *Comput. Ind. Eng.*, vol. 115, no. 1, pp. 495–511, Jan. 2018. doi: 10.1016/j.cie.2017.12.001.

[2] D. Saxena and A. K. Singh, "A proactive autoscaling and energy-efficient VM allocation framework using online multi-resource neural network for cloud data center," *Neurocomputing*, vol. 426, no. 6, pp. 248–264, Feb. 2021. doi: 10.1016/j.neucom.2020.08.076.

[3] S. Mashhadi Moghaddam, M. O'Sullivan, C. Walker, S. Fotuhi Piraghaj, and C. P. Unsworth, "Embedding individualized machine learning prediction models for energy efficient VM consolidation within Cloud data centers," *Future Gener. Comput. Syst.*, vol. 106, no. 1, pp. 221–233, Jan. 2020. doi: 10.1016/j.future.2020.01.008.

[4] N. K. Biswas, S. Banerjee, U. Biswas, and U. Ghosh, "An approach towards development of new linear regression prediction model for reduced energy consumption and SLA violation in the domain of green cloud computing," *Sustain. Energy Technol.*, vol. 45, no. 4, pp. 101087, Jun. 2021. doi: 10.1016/j.seta.2021.101087.

[5] T. C. Ferreto, M. A. S. Netto, R. N. Calheiros, and C. A. F. de Rose, "Server consolidation with migration control for virtualized data centers," *Future Gener. Comput. Syst.*, vol. 27, no. 8, pp. 1027–1034, Oct. 2011. doi: 10.1016/j.future.2011.04.016.

[6] I. Takouna, E. Alzaghoul, and C. Meinel, "Robust virtual machine consolidation for efficient energy and performance in virtualized data centers," presented at the 2014 IEEE Int. Conf. Green. Comput., Taipei, Taiwan, Sep. 1–3, 2014.

[7] J. Jheng, F. Tseng, H. Chao, and L. D. Chou, "A novel VM workload prediction using grey forecasting model in cloud data center," presented at the 2014 Int. Conf. Inf. Net., Phuket, Thailand, Feb. 10–12, 2014.

[8] S. Y. Hsieh, C. S. Liu, R. Buyya, and A. Y. Zomaya, "Utilization-prediction-aware virtual machine consolidation approach for energy-efficient cloud data centers," *J. Parallel. Distr. Comput.*, vol. 139, pp. 99–109, May 2020. doi: 10.1016/j.jpdc.2019.12.014.

[9] M. H. Sayadnavard, A. Toroghi Haghighat, and A. M. Rahmani, "A multi-objective approach for energy-efficientand reliable dynamic VM consolidation in cloud data centers," *Eng. Sci. Technol.*, vol. 26, no. 2, pp. 100995, Feb. 2021. doi: 10.1016/j.jestch.2021.04.014.

[10] L. Li, J. Dong, D. Zuo, and J. Liu, "SLA-aware and energy-efficient VM consolidation in cloud data centers using host states naive bayesian prediction model," presented at the 2018 IEEE. Int. Conf. Ubiq. Comput., Melbourne, VC, AUS, Dec. 11–13, 2018.

[11] R. N. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, "Workload prediction using ARIMA model and its impact on cloud applications' QoS," *IEEE Trans. Cloud Comput.*, vol. 3, no. 4, pp. 449–458, Aug. 2015. doi: 10.1109/TCC.2014.2350475.

[12] F. Farahnakian, P. Liljeberg, and J. Plosila, "LiRCUP: Linear regression based CPU usage prediction algorithm for live migration of virtual machines in data centers," presented at the 2013 Euro. Conf. Soft. Eng. Adv. Appl., Santander, Spain, Sep. 4–6, 2013.

[13] F. Farahnakian *et al.*, "Using ant colony system to consolidate VMS for green cloud computing," *IEEE Trans. Serv. Comput.*, vol. 2, pp. 187–198, Oct. 2015. doi: 10.1109/TSC.2014.2382555.

[14] N. T. Hieu, M. D. Francesco, and A. Ylä-Jääski, "Virtual machine consolidation with usage prediction for energy-efficient cloud data centers," presented at the 2015 IEEE 8th Int. Conf. Clou. Comput., New York, USA, Jun. 27–Jul. 2, 2015.

[15] S. B. Shaw, J. P. Kumar, and A. K. Singh, "Energy-performance trade-off through restricted virtual machine consolidation in cloud data center," presented at the 2017 Int. Conf. Inte. Comput. Cont., Coimbatore, India, Jun. 23–24, 2017.

[16] F. Farahnakian, T. Pahikkala, P. Liljeberg, and J. Plosila, "Energy aware consolidation algorithm based on k-nearest neighbor regression for cloud data centers," presented at the 2013 IEEE/ACM Int. Conf. Util. Clou. Comput., Dresden, Germany, Dec. 9–12, 2013.

[17] Z. Li, C. Yan, X. Yu, and N. Yu, "Bayesian network-based virtual machines consolidation method," *Future Gener. Comput. Syst.*, vol. 69, no. 7, pp. 75–87, Apr. 2017. doi: 10.1016/j.future.2016.12.008.

[18] J. Kumar, A. K. Singh, and R. Buyya, "Self directed learning based workload forecasting model for cloud resource management," *Inform. Sci.*, vol. 543, no. 1, pp. 345–366, Jan. 2021. doi: 10.1016/j.ins.2020.07.012.

[19] L. Abdullah, H. Li, S. Al-Jamali, A. Al-Badwi, and C. Ruan, "Predicting multi-attribute host resource utilization using support vector regression technique," *IEEE Access*, vol. 8, pp. 66048–66067, Mar. 2020. doi: 10.1109/ACCESS.2020.2984056.

[20] T. Thein, M. M. Myo, S. Parvin, and A. Gawanmeh, "Reinforcement learning based methodology for energy-efficient resource allocation in cloud data centers," *J. King Saud Univ-Comput.*, vol. 32, no. 10, pp. 1127–1139, Dec. 2020. doi: 10.1016/j.jksuci.2018.11.005.

[21] M. Ghobaei-Arani, S. Jabbehdari, and M. A. Pourmina, "An autonomic resource provisioning approach for service-based cloud applications: A hybrid approach," *Future Gener. Comput. Syst.*, vol. 78, no. 3, pp. 191–210, Jan. 2018. doi: 10.1016/j.future.2017.02.022.

[22] Y. Liu, X. Sun, W. Wei, and W. Jing, "Enhancing energy-efficient and QoS dynamic virtual machine consolidation method in cloud environment," *IEEE Access*, vol. 6, pp. 31224–31235, May 2018. doi: 10.1109/ACCESS.2018.2835670.

[23] H. A. Kholidy, "An intelligent swarm based prediction approach for predicting cloud computing user resource needs," *Comput. Commun.*, vol. 151, no. 1, pp. 133–144, Feb. 2020. doi: 10.1016/j.comcom.2019.12.028.

[24] Z. U. Qazi, H. Shahzad, and K. G. Muhammad, "Adaptive resource utilization prediction system for infrastructure as a service cloud," *Comput. Intell. Neurosci.*, vol. 2017, pp. 4873459, Jul. 2017. doi: 10.1155/2017/4873459.

[25] J. Kumar, A. K. Singh, and R. Buyya, "Ensemble learning based predictive framework for virtual machine resource request prediction," *Neurocomputing*, vol. 397, no. 5, pp. 20–30, Jul. 2020. doi: 10.1016/j.neucom.2020.02.014.

[26] G. Kaur, A. Bala, and I. Chana, "An intelligent regressive ensemble approach for predicting resource usage in cloud computing," *J. Parallel. Distr. Comput.*, vol. 123, no. 1, pp. 1–12, Jan. 2019. doi: 10.1016/j.jpdc.2018.08.008.

[27] J. Subirats and J. Guitart, "Assessing and forecasting energy efficiency on cloud computing platforms," *Future Gener. Comput. Syst.*, vol. 45, no. 1, pp. 70–94, Apr. 2015. doi: 10.1016/j.future.2014.11.008.

[28] K. Mason, M. Duggan, E. Barrett, J. Duggan, and E. Howley, "Predicting host CPU utilization in the cloud using evolutionary neural networks," *Future Gener. Comput. Syst.*, vol. 86, no. 4, pp. 162–173, Sep. 2018. doi: 10.1016/j.future.2018.03.040.

[29] J. Kumar, R. Goomer, and A. K. Singh, "Long short term memory recurrent neural network (LSTM-RNN) based workload forecasting model for cloud datacenters," *Procedia Comput.*, vol. 125, no. 3, pp. 676–682, 2018. doi: 10.1016/j.procs.2017.12.087.

[30] Y. Zhang, J. Yao, and H. Guan, "Intelligent cloud resource management with deep reinforcement learning," *IEEE Cloud Comput.*, vol. 4, no. 6, pp. 60–69, Dec. 2017. doi: 10.1109/MCC.2018.1081063.

[31] B. Hariharan, R. Siva, S. Kaliraj, and P. N. Senthil Prakash, "ABSO: An energy-efficient multi-objective VM consolidation using adaptive beetle swarm optimization on cloud environment," *J. Amb. Intell. Hum. Comput.*, vol. 14, no. 3, pp. 2185–2197, Aug. 2021. doi: 10.1007/s12652-021-03429-w.

[32] M. Sharifi, H. Salimi, and M. Najafzadeh, "Power-efficient distributed scheduling of virtual machines using workload-aware consolidation techniques," *J. Supercomput.*, vol. 61, no. 1, pp. 46–66, Jul. 2011. doi: 10.1007/s11227-011-0658-5.

[33] M. Ghetas, "A multi-objective monarch butterfly algorithm for virtual machine placement in cloud computing," *Neur. Comput. Appl.*, vol. 33, no. 17, pp. 11011–11025, Jan. 2021. doi: 10.1007/s00521-020-05559-2.

[34] S. Mazumdar and M. Pranzo, "Power efficient server consolidation for cloud data center," *Future Gener. Comput. Syst.*, vol. 70, pp. 4–16, May 2017. doi: 10.1016/j.future.2016.12.022.

[35] M. A. Elshabka, H. A. Hassan, W. M. Sheta, and H. M. Harb, "Security-aware dynamic VM consolidation," *Egypt. Inform. J.*, vol. 13, pp. 277–284, Sep. 2020. doi: 10.1016/j.eij.2020.10.00.

[36] Y. Sharma, W. Si, D. Sun, and B. Javadi, "Failure-aware energy-efficient VM consolidation in cloud computing systems," *Future Gener. Comput. Syst.*, vol. 94, pp. 620–633, May 2019. doi: 10.1016/j.future.2018.11.052.

[37] B. M. P. Moura, G. B. Schneider, A. C. Yamin, H. Santos, R. H. S. Reiser and B. Bedregal, "Interval-valued fuzzy logic approach for overloaded hosts in consolidation of virtual machines in cloud computing," *Fuzz. Sets Syst.*, vol. 446, no. 4, pp. 144–166, Oct. 2022. doi: 10.1016/j.fss.2021.03.001.

[38] N. Kord and H. Haghighi, "An energy-efficient approach for virtual machine placement in cloud based data centers," presented at the 2013 5th Conf. Info. Know. Tech., Shiraz, Iran, May 28–30, 2013.

[39] Q. Liao and Z. Wang, "Energy consumption optimization scheme of cloud data center based on SDN," *Procedia Comput. Sci.*, vol. 131, no. 5, pp. 1318–1327, Jan. 2018. doi: 10.1016/j.procs.2018.04.327.

[40] S. Gharehpasha and M. Masdari, "A discrete chaotic multi-objective SCA-ALO optimization algorithm for an optimal virtual machine placement in cloud data center," *J Amb. Intell. Hum. Comp.*, vol. 12, Nov. 2020. doi: 10.1007/s12652-020-02645-0.

[41] Y. Gao, H. Guan, Z. Qi, Y. Hou, and L. Liu, "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing," *J. Comput. Syst Sci.*, vol. 79, no. 8, pp. 1230–1242, Dec. 2013. doi: 10.1016/j.jcss.2013.02.004.

[42] M. H. Malekloo, N. Kara, and M. E. Barachi, "An energy efficient and SLA compliant approach for resource allocation and consolidation in cloud computing environments," *Sustain. Comput-Infor.*, vol. 17, pp. 9–24, Mar. 2018. doi: 10.1016/j.suscom.2018.02.001.

[43] X. Liu, J. Wu, L. Chen, and L. Zhang, "Energy-aware virtual machine consolidation based on evolutionary game theory," *Concurr. Comp-Pract. E*, vol. 34, no. 10, pp. e6830.1–e6830.16, May 2022. doi: 10.1002/cpe.6830.

[44] 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," *Concurr. Comp-Pract. E*, vol. 24, no. 13, pp. 1397–1420, Oct. 2011. doi: 10.1002/cpe.1867.

[45] Z. Wei *et al.*, "Performance degradation-aware virtual machine live migration in virtualized servers," presented at the 2012 Int. Conf. Para. Dist. Comput., Appl. Tech., Beijing, China, Dec. 14–16, 2012.

[46] H. Liu, H. Jin, C. Z. Xu, and X. Liao, "Performance and energy modeling for live migration of virtual machines," *Cluster Comput.*, vol. 16, no. 2, pp. 249–264, Dec. 2011. doi: 10.1007/s10586-011-0194-3.

[47] Q. Huang, F. Gao, R. Wang, and Z. Qi, "Power consumption of virtual machine live migration in clouds," presented at the 2011 Int. Conf. Comm. Mob. Comput., Qingdao, China, Apr. 18–20, 2011.

[48] M. Valipour, M. E. Banihabib, and S. M. R. Behbahani, "Comparison of the ARMA, ARIMA, and the autoregressive artificial neural network models in forecasting the monthly inflow of Dez dam reservoir," *J. Hydrol.*, vol. 476, no. 10, pp. 433–441, Jan. 2013. doi: 10.1016/j.jhydrol.2012.11.017.

[49] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel, *Time Series Analysis: Forecasting and Control*. New Jersey, USA: Wiley, Jun. 2008. doi: 10.1002/9781118619193.

[50] T. Mandhi and H. Mezni, "A prediction-based VM consolidation approach in IaaS cloud data centers," *J. Syst. Softw.*, vol. 146, no. 12, pp. 263–285, Dec. 2018. doi: 10.1016/j.jss.2018.09.083.