Enhanced Linear Regression Models for Resource Usage Prediction in Dynamic Cloud Environments

Xiaoxiao Ma
School of Transportation, Chongqing Vocational College of Transportation, Chongqing 402247, China

***Abstract*—In response to the diverse resource utilization patterns observed across enterprises, this study proposes the utilization of adaptable cloud services. A novel system framework is presented, capturing and logging resource consumption at discrete intervals. Subsequently, this recorded data serves as input for a linear regression model, functioning as a machine learning tool to predict resource utilization in forthcoming intervals, leveraging historical data stored within the regression module. To bolster the resilience of the linear regression model, various effective meta-heuristic techniques are integrated alongside the conventional linear regression methodology, facilitating more accurate anticipation of overloaded or under-loaded resource conditions before their occurrence in real-world scenarios. Simulations demonstrate that the hybrid algorithm, named Whale Optimization Algorithm-based Linear Regression (WOA-LR), outperforms Genetic Algorithm-Linear Regression (GA-LR), Particle Swarm Optimization-Linear Regression (PSO-LR), JAYA-LR, and traditional Linear Regression (LR) in achieving desired objective functions and significantly reducing Mean Squared Error (MSE). This approach holds promise for more accurate resource utilization prediction and optimization in dynamic cloud environments.**

***Keywords*—Cloud computing; resource utilization; prediction; linear regression; metaheuristics**

I. Introduction

With recent advances in artificial intelligence, the Internet of Things (IoT) [1, 2], Wireless Sensor Networks (WSNs) [3], and cloud computing, research efforts are shifting towards simplifying communication across various devices. Cloud computing represents a novel computational paradigm for provisioning computing resources, catering to a wide spectrum of users, ranging from individuals to large-scale enterprises [4, 5]. Cloud computing ecosystems are dominated by the intricate and cost-intensive data centers (DCs) that significantly impact service providers' financial viability [6]. Cloud providers offer three primary service categories, namely Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS), leveraging web service technology [7, 8]. Notable examples include Amazon for IaaS [9], Google for PaaS [10], and Salesforce for SaaS [11], all renowned as leading cloud providers worldwide. On one front, cloud providers offer their computational resources to fulfill users' Quality of Service (QoS) requirements, and on the other, they must effectively manage their Total Cost of Ownership (TCO) to thrive in the increasingly competitive cloud market [12]. Virtualization technology is widely employed within DCs to optimize resource allocation and reduce overall power consumption, a pivotal component of TCO. Furthermore, power management aligns with sustainability objectives. In a virtualized environment, a hypervisor intervenes to multiplex the resources of Physical Machines (PMs) among Virtual Machines (VMs) [13]. Inefficient resource allocation has repercussions on both resource utilization and the overall power consumption of DCs [14, 15].

Given the dynamic and ever-changing nature of cloud infrastructures and platforms, live VM migration emerges as a practical strategy that aligns with the current state of DCs [16]. Two common occurrences in DCs are under-utilization and over-utilization events [17]. The former entails a high-power consumption rate, while the latter is characterized by a high SLA violation rate. To address the scenario where businesses deploy VMs with varying usage patterns and resource requirements over time, machine learning approaches prove invaluable in discerning near-precise usage patterns in the short-term future [18]. Consequently, live virtual machine migration is employed to meet requirements before the aforementioned unfavorable events materialize. To maintain the desired QoS for users, Service Level Agreements (SLAs) are established between users and providers [19]. For instance, if a user submits an application comprising 150,000 million Instructions (MIs) and requests a VM with processing power equivalent to 250 million Instructions Per Second (MIPS), the provider must ensure the VM operates continuously at 100% utilization to deliver the results within 10 minutes. Failure to do so results in an SLA violation and a penalty for the service provider. The risk of SLA violation escalates when PMs become overloaded in DCs. Hence, preemptive offloading of some VMs prior to this event can mitigate SLA violations. Conversely, the phenomenon of server sprawl significantly increases total power consumption, whereby numerous under-loaded active PMs run concurrently. Server consolidation, which consolidates VMs into the fewest active PMs, is an effective strategy for reducing overall power consumption [20].

The integration of machine learning, deep learning, and meta-heuristic algorithms significantly enhances resource usage prediction in dynamic cloud environments. These methodologies provide a key insight into complex resource utilization patterns, contributing to the efficient management of cloud infrastructures [21]. Machine learning models, particularly linear regression and its variants, empower predictions based on historical resource usage data, enabling proactive resource allocation and load balancing [22]. Deep learning techniques, with their ability to analyze vast amounts of unstructured data, facilitate the identification of intricate patterns within cloud workloads, leading to more accurate predictions [23, 24]. Moreover, the inclusion of meta-heuristic

algorithms, such as genetic algorithms, particle swarm optimization, and whale optimization, augments predictive accuracy by refining the traditional models, enabling them to adapt and evolve according to dynamic resource demands [25]. The synergy among these methodologies results in robust and adaptable prediction models vital for optimizing resource usage in dynamic cloud environments, ultimately leading to improved service quality, cost efficiency, and better user experiences within cloud services [26, 27].

In this paper, machine learning techniques are extensively leveraged to derive resource usage patterns from historical data. This empowers the hypervisor to make swift decisions regarding under-utilization and over-utilization events before they occur. To address this challenge, machine learning techniques are applied to historical data to predict near-future resource requests. Processing and memory resources are pivotal for each requested VM, with processing capacity holding greater significance among power consumers, which informs the focus of this study on CPU capacity requirements [28]. To forecast near-future resource requests, the linear regression algorithm, a branch of machine learning, is employed. This involves recording the average resource utilization at five-minute intervals, utilizing the data history from the previous hour to inform short-term predictions. The time interval for data collection can be tailored to specific requirements. To enhance the performance of traditional linear regression, several effective meta-heuristic approaches are incorporated, yielding promising results. The innovation in this paper centers on the following key aspects:

- Introduction of a resource usage prediction model based on historical data.
- Presenting a migration trigger model for timely decision-making to avert unforeseen events.
- Exploration and evaluation of customized meta-heuristic algorithms to determine the most efficient approach.

The structure of this paper is organized as follows: Section II provides an overview of related work in the field. Section III introduces the proposed system model. In Section IV, the problem is formally defined and elaborated upon. Section V presents the suggested algorithm to address the problem. The performance assessment of the proposed approach is outlined in Section VI. Section VII concludes the paper and outlines potential future directions for research in this domain.

II. BACKGROUNDS

Live VM migration is the intricate process of seamlessly transferring a VM's loads from one PM to another, ensuring uninterrupted service for the end user. The initiation of live VM migration and server consolidation is motivated by various factors, with power management being of paramount importance [29]. Other driving factors include mobile computing [30], reduction of communication costs [31, 32], system maintenance [33], and enhancing system failure reliability [34]. This leads to fundamental questions regarding when and where VM migration should be triggered, a topic that has been extensively explored in existing literature. One notable contribution in this domain was made by Martinovic, et al. [35], who introduced a server consolidation model aimed at minimizing power consumption. They approached the problem by transforming the VM placement challenge into a bi-packing problem with conflicts and modeled by an integer linear program to solve it while utilizing the minimum number of PMs required. Zhou, et al. [36] proposed a linear regression model for predicting the CPU demands of VMs and subsequently triggering live VM migration in anticipation of near-future overload. Although this approach holds promise, it suffers from relatively high prediction errors.

Zhao, et al. [37] introduced a communication-aware live VM migration algorithm based on the Ant Colony Optimization (ACO) algorithm to minimize overall costs. This algorithm consists of two phases: first, it identifies VMs with high affinity for migration, and second, it selects the destination PMs for relocating the VMs. An energy-aware VM migration model was presented by Patel, et al. [38] to achieve both load balancing and power conservation, involving a three-way decision-making process for heavy, medium, and light workloads. A combined forecasting and load-aware migration model, along with an automated algorithm, was proposed by Forsman, et al. [39] to address live VM migration. A similar approach was put forth by Paulraj, et al. [40], focusing on saving energy, enhancing system reliability against failures, and maximizing service availability. The VM migration process, being resource-intensive and potentially degrading performance, employs forecasting models to estimate the resource requirements of each VM. Optimal online deterministic VM placement and adaptive heuristic algorithms are employed to address server consolidation, contributing to efficient DC power management and performance maintenance [41]. In the proposed approach, a novel system framework is introduced, featuring both local and global managers. The global manager resides at the master node, while each PM is equipped with a local manager responsible for collecting information on resource utilization and transmitting it to the global manager. Subsequently, the global manager issues directives for optimal VM placement, considering user SLAs. To further enhance efficiency, a cost function is defined, linked to the time associated with the migration process. The migration process is carefully orchestrated to avoid violating predetermined SLAs.

The review of existing literature reveals promising contributions from the research community. However, there remains a challenge in achieving near-precise prediction models. This motivates the current article, which leverages machine learning methods to predict short-term resource requirements for each VM, thereby triggering relevant VM migration processes to reduce both power consumption and SLA violation rates significantly.

III. SYSTEM MODEL

This section introduces the proposed system model and its components, along with a schematic example to demonstrate how the model functions. The proposed system model, depicted in Fig. 1, consists of two main parts: the front end and the back end. In the front end, users request specific types and specifications of VMs encapsulated in SLA format. These

requests are forwarded to a broker module that possesses knowledge of the underlying infrastructure's capabilities. The requested resources for each VM are logged in a repository, accumulating as historical data. Subsequently, the forecaster module, a part of the live VM migration scheme, is activated based on the historical data in the repository. Its primary objective is to prevent both SLA violations resulting from overloaded conditions and high-power consumption due to the server sprawl phenomenon.

Fig. 2 provides a schematic example of a scenario involving the execution of different VMs within a data center with three PMs. Initially, three different users submit their requests denoted as R_1, R_2, and R_3, where R_1 entails a request for 2 VMs, R_2 for 1 VM, and R_3 for 2 VMs. The broker promptly dispatches these requests to the available PMs, as depicted in Fig. 2(a). At five-minute intervals, resource requests are recorded in a repository. The forecaster module extracts insights from this data history and anticipates that PM_3 will become overloaded in the near short-term future due to the surging resource request of VM_5. To prevent an overload event, the live VM migration module is activated, and PM_2 is chosen for offloading due to its surplus resources. After the live migration, the scenario is represented in Fig. 2(b). In this situation, when the five-minute time interval is reached, the forecaster module predicts that VM_2 and VM_4 will reduce their resource requirements. This prediction aims to decrease resource requests.

Consequently, both PM_1 and PM_3 are in an under-loaded condition in the near future. Therefore, the live VM migration scheme is initiated for both VM_2 and VM_4 to consolidate servers. Subsequently, the unused PM_3 is transitioned into hibernation mode to conserve energy that would otherwise be dissipated.

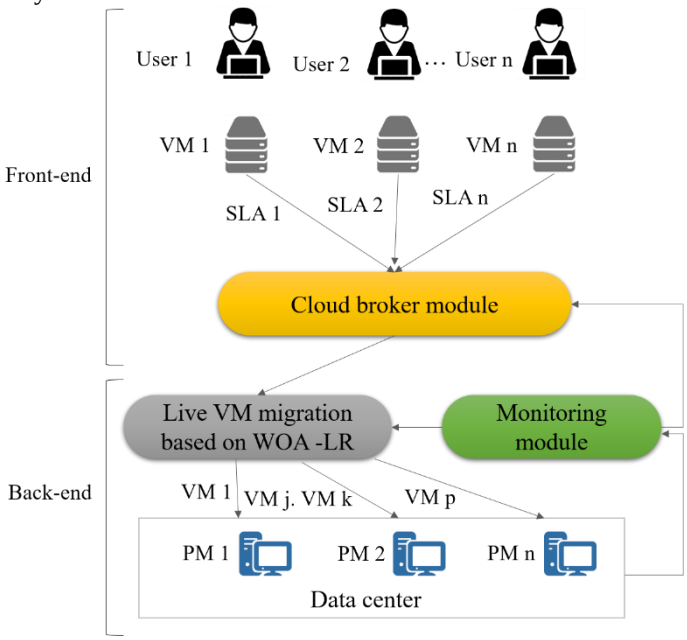

Fig. 1. System model.

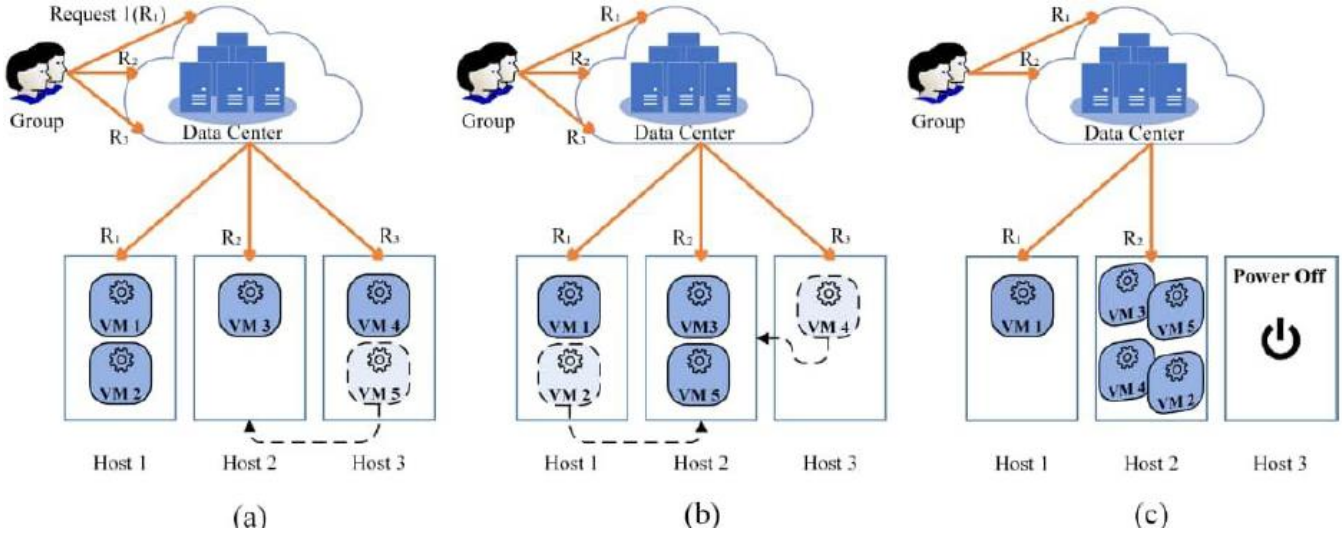

Fig. 2. A schematic example of a data center with deployed VMs.

IV. PROBLEM STATEMENT

The core problem at hand is the necessity to make prompt decisions to preempt unfavorable events before they materialize. Periodically, the forecaster module retrieves data from the data repository within the data center to predict the impending over-loaded and under-loaded states of each PM. Subsequently, the most appropriate decision is made based on these predictions. To accomplish this, an advanced linear regression model is employed with the objective of minimizing the model's Mean Squared Error (MSE). In this pursuit, the conventional linear regression method is fused with a meta-heuristic algorithm, resulting in a novel and advanced hybrid forecaster algorithm. The key aim is to reduce the MSE to the lowest extent, thereby enhancing the accuracy of the decision-making process. In essence, the decision becomes increasingly accurate as the error approaches its minimum value. To facilitate this, the data history stored in the repository pertaining to resource requests within the most recent hour is divided into 12 records, each representing a five-minute interval. Utilizing the information within these recorded data, a linear function is established. Eq. (1) represents this linear function, where *x* and *y* denote the input and forecast functions, respectively. The terms C_0 and C_1 signify the two constants that serve as the coefficients of the linear function and must be determined by the proposed model.

$$y = C_0 + C_1 x \tag{1}$$

V. PROPOSED ALGORITHM

To address the live VM migration, which is inherently an optimization challenge, various competitive algorithms have been proposed. These algorithms have been selected based on their demonstrated success in the existing literature for solving continuous optimization problems and their adaptability to the specific problem under consideration. In this section, we introduce the following algorithms for calculating both LR's parameters C_0 and C_1: Canonical Linear Regression (LR), Genetic Algorithm-based Linear Regression (GA-LR), Particle Swarm Optimization-based Linear Regression (PSO-LR), Whale Optimization Algorithm-based Linear Regression (WOA-LR), and JAYA Linear Regression (JAYA-LR). GA-LR predicts the utilization of each server for the short-term future based on previously recorded mean server CPU utilization. In this context, single-point crossover and random gene mutation operators are employed. Additionally, a tournament algorithm is formulated. These algorithms are designed to optimize the LR's parameters C_0 and C_1, thereby enabling more accurate predictions and decision-making in the context of the problem at hand.

In the proposed GA-LR, random individuals are generated to represent both populations associated with the two constants, C_0 and C_1, which serve as the coefficients in the linear regression function. Each record for each coefficient is incorporated into the linear regression function, and the difference between this value and the actual value in the dataset is regarded as the fitness value. This optimization problem is a straightforward minimization problem, where the goal is to refine the coefficients over successive rounds. For encoding, each chromosome consists of two segments: the first part encodes an integer value, and the second part encodes a real number. It's worth noting that binary genes are used in the encoding process. The Tournament selection procedure aims to increase the likelihood of selecting promising chromosomes. In this procedure, *K* chromosomes are randomly chosen from the populations, and the best-performing ones are returned. It is important to highlight that the proposed tournament selection approach does not directly select the best individuals from the entire population, as this would risk early convergence, potentially resulting in suboptimal performance. The algorithm iterates until a specified termination condition is met, ultimately returning the best-performing chromosomes that yield the minimum MSE value.

In the PSO-LR algorithm, two distinct swarms of particles are randomly generated, akin to the populations of individuals in genetic Algorithms. Each particle's future trajectory is determined by three key parameters: inertia, local best, and global best values. The first parameter, inertia, is responsible for the particle continuing in its previous direction. The second parameter directs the particle to adjust its direction based on its local best, which is recorded in its memory. The third parameter steers the particle towards changing its direction to align with the global best of the entire swarm. To weigh the effectiveness of each parameter, specific weights are assigned to them. The algorithm is executed over multiple iterations, and subsequently, the best-performing particle thus far is identified and returned as the optimal solution. This iterative process helps refine the solution and converge towards the most accurate values for C_0 and C_1 in the linear regression model. For another comparative approach, the WOA-LR algorithm is presented in Fig. 3.

Similar to other swarm-based meta-heuristic algorithms, the WOA-LR begins with the generation of random swarms of whales. In this algorithm, each pair of whales represents a solution, as each solution requires two coefficients. To this end, variables $PopC_0(i)$ and $PopC_1(i)$ pertain to the i^{th} whale. The loop encompassing lines 4 to 34 is executed for each whale, which is why there are two inner for-loops. In lines seven and eight, two sets of vectors, *a*, *A*, and *C*, are updated. It is important to note that *a* is a vector that gradually decreases from 2 to 0. Additionally, the vectors *A* and *C* consist of random real values within the [0..1] interval. The changes in r are determined by Eq. (2), and Eq. (3) defines the alterations in these vectors.

$$\vec{A} = 2\vec{a}.\vec{r} - \vec{a} \tag{2}$$

$$\vec{C} = 2\vec{r} \tag{3}$$

Furthermore, random real values *l* are stochastically selected from the interval [-1..1]. The essence of the WOA lies in the oscillation between exploration and exploitation phases at intervals during the algorithm's lifecycle. To this end, a random variable *P* is drawn to determine whether to explore or exploit the search space. The update process in line 15 is specifically designed to reflect the inclination towards exploitation or local search. In the context of exploration, the update is carried out using Eq. (4), and this operation is executed in line 18. In this scenario, the update is impartially performed based on the random position of the whale without any bias.

Algorithm 4. WOA-Regression

Input:
Swarm1 , Swarm2 : Swarm;
SwarmSize : integer;
MAXITER : integer;

Output:
Optimal C0, C1, and future server utilization

1. Initialize the two whale populations $PopC_0$ and $PopC_1$ including $PopC_0(i)$ and $PopC_1(i)$ (i=1,2,…,*PopSize*)
2. Calculate the fitness fucntion for all y(i)= PopC0[i]+ PopC1[i]*x(i)
3. let PopC0* and PopC1* to be the coefficient of the best whale recognized so far.
4. While iteration is not reach to *MaxIteration* Do
5. for i=1 To *PopSize* Do
6. for j=1 To *PopSize* Do
7. Update vectors a0, A0, and C0 based on Eqs.(6-7) ; and random variable *l*
8. Update vectors a1, A1, and C1 based on Eqs.(6-7) ; and random variable *l*
9. Draw new real random number 0≤P≤1;
10. if (P<0.5) then
11. if (|A| < 1) then
12. Update the whlale PopC0[i] position based on Eq. (8)
13. else if (|A| ≥ 1) then
14. Select a random whale PopC0[j] from population
15. Update the whale PopC0[i] position based on Eq.(9) incorporating PopC0[j]
16. end-if
17. else (* P ≥0.5 *)
18. Update the whale PopC1[i] position based on Eq.(10)
19. end-if
20. Draw new real random number 0≤P≤1;
21. if (P<0.5) then
22. if (|A| < 1) then
23. Update the whlale PopC1[i] position based on Eq. (9)
24. else if (|A| ≥ 1) then
25. Select a random whale PopC1[j] from population
26. Update the whale PopC1[i] position based on Eq.(11) incorporating PopC1[j]
27. end-if
28. else (* P ≥0.5 *)
29. Update the whale PopC1[i] position based on Eq.(13)
30. end-if
31. Call Clamping PopC0[i] and PopC1[i] to correct the whale position if it returns infeasible solution
32. end-for
33. find current best so far from population and put it to PopC0* and PopC1*
34. end-while
35. return PopC0*+PopC1*.CurrentUtilization

Fig. 3. Algorithm WOA-LR.

$$\vec{W_i}(t+1) = \vec{W_j}(t) - \vec{A}.\vec{D} \quad (4)$$

The algorithm uses Eq. (5) to update the chosen whale's location in order to imitate the distinctive circular movement of whales, which is sometimes referred to as a "spiral update position." This mechanism is designed to mimic the distinctive movement pattern of whales during the optimization process.

$$\vec{W_i}(t+1) = \vec{D'}.e^{bl}.Cos(2\pi) + \overrightarrow{W*}(t) \quad (5)$$

After the update is executed for all whales, in case any whale's solution becomes infeasible, the Clamping function is invoked to adjust the encoded solution within the appropriate and feasible space. Line 31 represents the application of the Clamping (.) function. The specific implementation of the Clamping function can vary based on the context. Ultimately, the best whale found so far, which represents an efficient solution, is returned as the final solution. Another successful optimization algorithm, which has been introduced recently, is the JAYA algorithm. The JAYA algorithm is specifically designed for addressing continuous optimization problems. Similar to other meta-heuristic algorithms, it commences with a limited number of randomly generated solutions. In each iteration, the best and worst solutions found so far are identified. The primary objective is to approach the best solution while distancing from the worst one. Throughout the evolution of each solution, if a new solution improves in terms

of fitness value, it is accepted; otherwise, the previous version is retained. Each solution is iteratively adjusted to converge toward the best solution gradually found thus far.

VI. RESULTS

This section is devoted to the performance evaluation of the proposed prediction model. To assess its effectiveness, various meta-heuristic-based algorithms have been put forward, and a comparative study has been conducted [42-44]. In this context, three distinct sets of datasets representing data from the last hour have been randomly generated. Table I provides an overview of these datasets. All of the selected algorithms operate on the same datasets, ensuring a level playing field for fair competition in the evaluation process. The simulation results are presented in Table II, with the reported values for the successful WOA-Regression algorithm highlighted. The key to the success of WOA-Regression lies in its ability to strike a balance between the exploration and exploitation phases during the optimization process, effectively optimizing the search process.

One of the most critical concerns in energy-intensive data centers is power consumption. Furthermore, the rate of SLA violations significantly affects users' decisions when it comes to adopting cloud services. In practice, users tend to abandon unreliable cloud providers that cannot meet the agreed SLA terms. Thus, ensuring a high-quality experience for users is of paramount importance. To address this issue, a power consumption model with a linear relationship to CPU utilization is introduced in Eq. (6).

$$P_{Current}^{PM_j} = \lambda_j \times P_{Full}^{PM_j} + (1 - \lambda_j) \times P_{Full}^{PM_j} \times U_{CPU}^{PM_j} \quad (6)$$

The term λ_j is employed to denote that an idle machine consumes a certain percentage of power compared to a fully loaded machine. Various research studies, including the current paper, commonly use λ_j as 70% of the power consumed by a fully utilized machine. To calculate the CPU utilization of a PM, the summation of the utilization of all co-hosted VMs processing requests is considered, which can be obtained using Eq. (7). Additionally, the memory utilization of each PM is determined by summing the requested memory of all co-hosted VMs, a calculation that can be performed using Eq. (8). These measurements are essential for assessing and managing the resource utilization of each PM in the data center.

$$U_{CPU}^{PM_j} = \sum_{i=1}^{n} U_{CPU}^{VM_i}.x_{ij} \quad (7)$$

$$U_{CPU}^{PM_j} = \sum_{i=1}^{n} U_{Mem}^{VM_i}.x_{ij} \quad (8)$$

The binary decision variable, denoted as x_{ij}, indicates whether a VM is placed on a PM or not. If a VM is placed on a PM, this variable is set to 1; otherwise, it is set to 0. Additionally, the terms $U_{CPU}^{VM_i}$ and $U_{Mem}^{VM_i}$ are used to represent the CPU and memory bandwidth requirements of a VM *i*. Similarly, the terms $U_{CPU}^{PM_j}$ and $U_{Mem}^{PM_j}$ are employed to denote the CPU and memory utilization of a PM *j*, respectively. These variables and terms play a vital role in optimizing the allocation and utilization of resources within the data center.

TABLE I. AN OVERVIEW OF THE GENERATED DATASETS

Server	CPU utilization recorded for different rounds											
	Round 1	***Round 2***	***Round 3***	***Round 4***	***Round 5***	***Round 6***	***Round 7***	***Round 8***	***Round 9***	***Round 10***	***Round 11***	***Round 12***
First server	82%	93%	73%	69%	33%	81%	94%	83%	39%	64%	27%	82%
Second server	69%	72%	37%	58%	17%	6%	62%	41%	40%	69%	53%	55%
Third server	66%	96%	49%	43%	81%	71%	6%	42%	17%	94%	41%	76%

TABLE II. SIMULATION RESULTS

Error calculation model	First PM	Second PM	Third PM
WOA-LR	MSE= 0.514	MSE= 0.352	MSE= 1.082
Regression coefficient	C0= 0.594, C1= -0.093	C0= 0.664, C1= -0.417	C0= -7.601, C1= 1.271
JAYA-LR	MSE= 1.128	MSE= 0.505	MSE= 1.282
Regression coefficient	C0= 0.628, C1= -0.16	C0= 0.784, C1= -0.511	C0= 0.654, C1= -0.191
PSO-LR	MSE= 1.0073	MSE= 0.441	MSE= 1.123
Regression coefficient	C0= 0.624, C1= -0.752	C0= 0.355, C1= 0.259	C0= 0.751, C1= -0.341
GA-LR	MSE= 0.684	MSE= 0.432	MSE= 1.108
Regression coefficient	C0= 0.722, C1= -0.0481	C0= 0.407, C1= 0.126	C0= 0.563, C1= -0.116
Conventional LR	MSE= 5.762	MSE= 4.128	MSE=7.361
Regression coefficient	C0= 0.653, C1= 0.024	C0= 0.397, C1= 0.154	C0= 0.622, C1= -0.164

The live VM migration technique allows the transfer of a VM's pages from the source PM to the destination PM with minimal interruption, resulting in a small downtime. However, VM live migration can have adverse effects on the overall system performance and potentially jeopardize SLA between users and service providers. This is due to the time required for page transfers during migration. For cloud-based web applications, on average, downtime may decrease CPU utilization by approximately 10%. In other words, this phenomenon can lead to SLA violations to some extent. On the flip side, it is desirable to reduce the number of aggressive live migrations. The Live Migration Time (LMT) is highly dependent on the size of the VM's pages being transferred and the underlying bandwidth capacity. Since data centers often use Storage Area Networks (SAN), there is no need to transfer VM storage data, as all PMs have uniform access to SAN. The LMT for a VM is calculated using Eq. (9). This information is crucial for managing VM migrations efficiently and minimizing potential SLA violations.

$$LMT(VM_i) = \frac{Msize(M_i)}{BW_i} \quad (9)$$

The term $Msize(M_i)$ represents the memory size of data being transferred via a shared link with a bandwidth capacity of BW_i. Furthermore, experimental results indicate that there is a 10% degradation in CPU utilization during the process of live migration. This performance degradation is quantified using Eq. (10), where the term $u_i(t)$ represents CPU utilization associated with VM i during the migration process. This equation provides a measure of the performance impact of live migrations, which is crucial for optimizing resource allocation and minimizing SLA violations.

$$PD(VM_i = 0.1. \int_{t_0}^{t_0+LMT(VM_i)} u_i(t)dt \quad (10)$$

The paramount issue that encourages users to remain loyal to specific cloud providers is the delivery of a high-quality user experience from the services provided. In this context, key points such as the minimum throughput and the maximum response time must be determined to meet the required QoS, which is outlined in the SLA. Web applications that leverage cloud infrastructure often exhibit fluctuations in resource utilization. As a result, an independent parameter reflecting the system's SLA violation rate is needed. To address this, two new parameters have been introduced: SLA violation length per active PM, denoted as a, and the total performance degradation due to VM migration, denoted as β. The parameter α represents the duration during which active PMs experience 100% CPU utilization, indicating the time span during which PMs are overloaded. This parameter is measured using Eq. (11) and plays a crucial role in quantifying SLA violations within the system.

$$a = \frac{1}{n}\sum_{i=1}^{n} \frac{T_{PM_i}}{T_{active(PM_i)}} \quad (11)$$

where, n represents the number of PMs, T_{PM_i} stands for the time during which PM_i experiences 100% CPU utilization, indicating when PM_i is overloaded. $T_{active(PM_i)}$ is the total time during which PM_i remains active, serving various VMs. The parameter β is calculated using Eq. (12). In this equation, m is the number of VMs, $PD(VM_i)$ is the performance degradation caused by VM migration, and $C(VM_i)$ represents the total CPU resource capacity associated with VM_i in terms of MIPS.

$$\beta = \frac{1}{m}\sum_{i=1}^{m} \frac{PD(VM_i)}{C(VM_i)} \quad (12)$$

Both parameters α and β independently influence SLA violations. To provide a comprehensive assessment of SLA violations, a new parameter called the SLA Violation Rate (SLAVR) is introduced in Eq. (13).

$$SLAVR = \alpha.\beta = \frac{1}{n}\sum_{i=1}^{n} \frac{T_{PM_i}}{T_{active\,(PM_i)}} . \frac{1}{m}\sum_{i=1}^{m} \frac{PD(VM_i)}{C(VM_i)} \quad (13)$$

With the inclusion of live migration costs, the performance of comparative algorithms is assessed in terms of energy consumption attributable to SLA violations, SLAVR, α (SLA violation length per active PM), β (total performance degradation due to VM migrations), and the number of VM migrations. Table III provides a comparison of the state-of-the-art algorithms based on these assessment metrics. All of the comparative algorithms were executed in the CloudSim environment, with 20 independent runs. The reported results represent the average outcomes obtained from these runs.

TABLE III. PERFORMANCE EVALUATION

Algorithm	VM migrations	β (%)	α (%)	SLAVR (%)	Total power consumption (Watt)	Power for SLAVR (Watt)
WOA-LR	189	0.11	1.31	14.76	1327	65.11
JAYA-LR	203	0.13	1.36	15.61	1672	77.05
PSO-LR	208	0.13	1.37	17.79	1463	79.19
GA-LR	211	0.14	1.48	19.62	1495	75.91
Conventional LR	306	0.16	1.42	21.47	1588	89.02

VII. CONCLUSION

This paper introduced a system framework for cloud data centers, comprising multiple modules designed to enable timely live VM migration for preventing SLA violations through the integration of machine learning tools. In this framework, a repository module is deployed within the DC, functioning as a data history repository where each physical machine records its average CPU utilization over time. Subsequently, a machine learning tool, specifically a linear regression-based model, is employed at regular intervals to predict near-future resource requirements. Based on these predictions, decisions are made to optimize resource allocation and avoid SLA violations.

There are still unanswered issues and unresolved problems in this field. Expanding the predictive capabilities to include important resources such as memory, storage, and network bandwidth might improve the overall effectiveness of the system. Furthermore, conducting performance evaluations on a range of real-world datasets and in different workload conditions would provide a more thorough understanding of the system's capacity to adjust and withstand challenges. Furthermore, it is necessary to do further research to determine the scalability, flexibility, and real-time responsiveness of the system when implemented and deployed in live cloud settings. The limitations of this study are its primary emphasis on CPU use, which may result in neglecting the complex interaction between various resources and their effect on adhering to SLA requirements. Moreover, the system's reliance on past data may provide difficulties in dynamic settings, requiring ongoing model training and adjustment methods. Future improvements may include integrating sophisticated machine learning methods, such as deep learning algorithms, to boost the accuracy of predictions and consider intricate resource linkages. Furthermore, investigating decentralized or distributed decision-making models for VM migrations, while also taking into account security and privacy concerns in a multi-tenant cloud environment, presents a promising direction for future study and improvement of the suggested framework.

REFERENCES

[1] B. Pourghebleh, V. Hayyolalam, and A. A. Anvigh, "Service discovery in the Internet of Things: review of current trends and research challenges," Wireless Networks, vol. 26, no. 7, pp. 5371-5391, 2020.

[2] B. Pourghebleh and N. J. Navimipour, "Data aggregation mechanisms in the Internet of things: A systematic review of the literature and recommendations for future research," Journal of Network and Computer Applications, vol. 97, pp. 23-34, 2017.

[3] J. Zandi, A. N. Afooshteh, and M. Ghassemian, "Implementation and analysis of a novel low power and portable energy measurement tool for wireless sensor nodes," in Electrical Engineering (ICEE), Iranian Conference on, 2018: IEEE, pp. 1517-1522, doi: 10.1109/ICEE.2018.8472439.

[4] S. Vinoth, H. L. Vemula, B. Haralayya, P. Mamgain, M. F. Hasan, and M. Naved, "Application of cloud computing in banking and e-commerce and related security threats," Materials Today: Proceedings, vol. 51, pp. 2172-2175, 2022.

[5] V. Hayyolalam, B. Pourghebleh, M. R. Chehrehzad, and A. A. Pourhaji Kazem, "Single - objective service composition methods in cloud manufacturing systems: Recent techniques, classification, and future trends," Concurrency and Computation: Practice and Experience, vol. 34, no. 5, p. e6698, 2022.

[6] A. H. A. Al-Jumaili, R. C. Muniyandi, M. K. Hasan, J. K. S. Paw, and M. J. Singh, "Big Data Analytics Using Cloud Computing Based Frameworks for Power Management Systems: Status, Constraints, and Future Recommendations," Sensors, vol. 23, no. 6, p. 2952, 2023.

[7] S. AlMuraytib, L. Alqurashi, and S. Snoussi, "Blockchain-based solutions for Cloud Computing Security: A Survey," in Proceedings of the 6th International Conference on Future Networks & Distributed Systems, 2022, pp. 338-342.

[8] B. Kruekaew and W. Kimpan, "Multi-objective task scheduling optimization for load balancing in cloud computing environment using hybrid artificial bee colony algorithm with reinforcement learning," IEEE Access, vol. 10, pp. 17803-17818, 2022.

[9] X. Song, L. Pan, and S. Liu, "An online algorithm for optimally releasing multiple on-demand instances in IaaS clouds," Future Generation Computer Systems, vol. 136, pp. 311-321, 2022.

[10] R. Kaviarasan, P. Harikrishna, and A. Arulmurugan, "Load balancing in cloud environment using enhanced migration and adjustment operator based monarch butterfly optimization," Advances in Engineering Software, vol. 169, p. 103128, 2022.

[11] E. Ahumada-Tello and R. Evans, "A Complexity-based Framework for Social Product Development," Procedia CIRP, vol. 119, pp. 1204-1209, 2023.

[12] S. S. Gill et al., "Transformative effects of IoT, Blockchain and Artificial Intelligence on cloud computing: Evolution, vision, trends and open challenges," Internet of Things, vol. 8, p. 100118, 2019.

[13] H. Vahideh, P. Behrouz, P. K. A. Asghar, and A. Ghaffari, "Exploring the state-of-the-art service composition approaches in cloud manufacturing systems to enhance upcoming techniques," The International Journal of Advanced Manufacturing Technology, vol. 105, no. 1-4, pp. 471-498, 2019.

[14] K. Saidi and D. Bardou, "Task scheduling and VM placement to resource allocation in Cloud computing: challenges and opportunities," Cluster Computing, vol. 26, no. 5, pp. 3069-3087, 2023.

[15] S. Pazouki and M. R. Haghifam, "Optimal planning and scheduling of smart homes' energy hubs," International Transactions on Electrical Energy Systems, vol. 31, no. 9, p. e12986, 2021.

[16] P. Verma et al., "Voltage Rise Mitigation in PV Rich LV Distribution Networks Using DC/DC Converter Level Active Power Curtailment Method," Energies, vol. 15, no. 16, p. 5901, 2022.

[17] S. Durairaj and R. Sridhar, "MOM-VMP: multi-objective mayfly optimization algorithm for VM placement supported by principal component analysis (PCA) in cloud data center," Cluster Computing, pp. 1-19, 2023.

[18] A. Belgacem, S. Mahmoudi, and M. A. Ferrag, "A machine learning model for improving virtual machine migration in cloud computing," The Journal of Supercomputing, pp. 1-23, 2023.

[19] J. Singh and M. S. Goraya, "An Autonomous Multi-Agent Framework using Quality of Service to prevent Service Level Agreement Violations in Cloud Environment," International Journal of Advanced Computer Science and Applications, vol. 14, no. 3, 2023.

[20] B. Pourghebleh, A. A. Anvigh, A. R. Ramtin, and B. Mohammadi, "The importance of nature-inspired meta-heuristic algorithms for solving virtual machine consolidation problem in cloud environments," Cluster Computing, pp. 1-24, 2021.

[21] S. Vairachilai, A. Bostani, A. Mehbodniya, J. L. Webber, O. Hemakesavulu, and P. Vijayakumar, "Body Sensor 5 G Networks Utilising Deep Learning Architectures for Emotion Detection Based On EEG Signal Processing," Optik, p. 170469, 2022.

[22] S. P. Rajput et al., "Using machine learning architecture to optimize and model the treatment process for saline water level analysis," Journal of Water Reuse and Desalination, 2022.

[23] V. Monjezi, A. Trivedi, G. Tan, and S. Tizpaz-Niari, "Information-Theoretic Testing and Debugging of Fairness Defects in Deep Neural Networks," presented at the 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE), 2023. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/ICSE48619.2023.00136.

[24] W. Anupong et al., "Deep learning algorithms were used to generate photovoltaic renewable energy in saline water analysis via an oxidation process," Water Reuse, vol. 13, no. 1, pp. 68-81, 2023.

[25] S. R. Abdul Samad et al., "Analysis of the Performance Impact of Fine-Tuned Machine Learning Model for Phishing URL Detection," Electronics, vol. 12, no. 7, p. 1642, 2023.

[26] M. Hajihosseinlou, A. Maghsoudi, and R. Ghezelbash, "Stacking: A novel data-driven ensemble machine learning strategy for prediction and mapping of Pb-Zn prospectivity in Varcheh district, west Iran," Expert Systems with Applications, vol. 237, p. 121668, 2024.

[27] M. Hajihosseinlou, A. Maghsoudi, and R. Ghezelbash, "A Novel Scheme for Mapping of MVT-Type Pb–Zn Prospectivity: LightGBM, a Highly Efficient Gradient Boosting Decision Tree Machine Learning Algorithm," Natural Resources Research, pp. 1-22, 2023.

[28] M. H. Shirvani, "An energy-efficient topology-aware virtual machine placement in Cloud Datacenters: A multi-objective discrete JAYA optimization," Sustainable Computing: Informatics and Systems, vol. 38, p. 100856, 2023.

[29] K. Kumar, K. Patange, P. Pete, M. Wankhade, A. Chatterjee, and M. Kurhekar, "Power and Energy-efficient VM scheduling in OpenStack Cloud Through Migration and Consolidation using Wake-on-LAN," IETE Journal of Research, pp. 1-13, 2022.

[30] Y. Kumar, S. Kaul, and Y.-C. Hu, "Machine learning for energy-resource allocation, workflow scheduling and live migration in cloud computing: State-of-the-art survey," Sustainable Computing: Informatics and Systems, vol. 36, p. 100780, 2022.

[31] S. Manjunatha and L. Suresh, "Optimal Min-Communication and Migration Cost Algorithm based Approach for Efficient Task Migration in Cloud Computing," in 2021 International Conference on Circuits, Controls and Communications (CCUBE), 2021: IEEE, pp. 1-6.

[32] S. Shahryari, F. Tashtarian, and S.-A. Hosseini-Seno, "CoPaM: Cost-aware VM Placement and Migration for Mobile services in Multi-Cloudlet environment: An SDN-based approach," Computer Communications, vol. 191, pp. 257-273, 2022.

[33] R. M. Haris, K. M. Khan, and A. Nhlabatsi, "Live migration of virtual machine memory content in networked systems," Computer Networks, vol. 209, p. 108898, 2022.

[34] M. Torquato, P. Maciel, and M. Vieira, "Availability and reliability modeling of vm migration as rejuvenation on a system under varying workload," Software Quality Journal, vol. 28, pp. 59-83, 2020.

[35] J. Martinovic, M. Hähnel, G. Scheithauer, and W. Dargie, "An introduction to stochastic bin packing-based server consolidation with conflicts," Top, vol. 30, no. 2, pp. 296-331, 2022.

[36] H. Zhou, Q. Li, K.-K. R. Choo, and H. Zhu, "DADTA: A novel adaptive strategy for energy and performance efficient virtual machine consolidation," Journal of Parallel and Distributed Computing, vol. 121, pp. 15-26, 2018.

[37] H. Zhao et al., "VM performance-aware virtual machine migration method based on ant colony optimization in cloud environment," Journal of Parallel and Distributed Computing, vol. 176, pp. 17-27, 2023.

[38] D. Patel, R. K. Gupta, and R. Pateriya, "Energy-aware prediction-based load balancing approach with VM migration for the cloud environment," Data, Engineering and Applications: Volume 2, pp. 59-74, 2019.

[39] M. Forsman, A. Glad, L. Lundberg, and D. Ilie, "Algorithms for automated live migration of virtual machines," Journal of Systems and Software, vol. 101, pp. 110-126, 2015.

[40] G. J. L. Paulraj, S. A. J. Francis, J. D. Peter, and I. J. Jebadurai, "A combined forecast-based virtual machine migration in cloud data centers," Computers & Electrical Engineering, vol. 69, pp. 287-300, 2018.

[41] 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 and Computation: Practice and Experience, vol. 24, no. 13, pp. 1397-1420, 2012.

[42] K. Ramana, R. Aluvalu, V. K. Gunjan, N. Singh, and M. N. Prasadhu, "Multipath Transmission Control Protocol for Live Virtual Machine Migration in the Cloud Environment," Wireless Communications and Mobile Computing, vol. 2022, 2022.

[43] A. Gupta, P. Dimri, and R. Bhatt, "An Optimized Approach for Virtual Machine Live Migration in Cloud Computing Environment," in Evolutionary Computing and Mobile Sustainable Networks: Springer, 2021, pp. 559-568.

[44] K. J. Naik, "An Adaptive Push-Pull for Disseminating Dynamic Workload and Virtual Machine Live Migration in Cloud Computing," International Journal of Grid and High Performance Computing (IJGHPC), vol. 14, no. 1, pp. 1-25, 2022.