Multi-resource predictive workload consolidation approach in virtualized environments

Mirna Awad[a], Aris Leivadeas[a], Abir Awad[b]

[a] *Department of Software Engineering and Information Technology, École de Technologie Supérieure, Montreal, Canada*
[b] *Computing department at Bournemouth University, Bournemouth, England*

***Abstract*— The revolution of virtualization technologies and Cloud computing solutions has emphasized the need for energy-efficient and Service level agreement (SLA)-aware resource management techniques in cloud data centers. Workload consolidation in Infrastructure-as-a-Service (IaaS) providers allows for efficient utilization of hardware resources and reduced energy consumption by consolidating workloads onto fewer physical servers. To ensure successful workload consolidation, it is crucial for IaaS providers to carefully estimate the host state and identify overloaded and underloaded hosts, thereby avoiding overly aggressive consolidation. Existing proposals determine the host state depending on its current resource utilization or a single anticipated resource utilization value, and often consider only a single resource type of the host, such as CPU. These limitations may lead to unreliable host state estimations, resulting in excessive and needless service migrations between physical machines (PMs). This, in turn, can lead to extra delays in service execution, degraded performance, increased power consumption, and SLA violations. To address these challenges, we propose a workload consolidation approach that leverages a multi-resource and multi-step resource utilization prediction model. Based on this model, our overload and underload decision-making algorithms consider the forecasted future trend (sequence of future value) of each host resource's utilization, including CPU, memory, and bandwidth. Through extensive experimentations conducted with two real-world datasets, we demonstrate that our approach can significantly reduce power consumption, SLA violation rate, and the number of migrations compared to existing benchmarks.**

***Index Terms*— Multi-resource, workload prediction, Kalman filter, Support vector regression, consolidation approach, Cloud computing.**

I. Introduction

The evolution of cloud computing solutions, progressing from Virtual Machines (VMs) to containers and serverless platforms, has revolutionized the way services are hosted and managed, ushering in an era of unprecedented flexibility and efficiency. However, amidst this remarkable progress, cloud providers grapple with complex challenges, especially in the realm of resource management [1][2][3]. One of these challenges is handling the dynamic workload fluctuations in cloud environments while meeting the Service Level Agreements (SLA) requirements to avoid penalties. Additionally, the heterogeneity of services or applications hosted within these environments, with varying resource demands and characteristics, adds another layer of complexity to the problem. Furthermore, cloud providers face the intricate task of reconciling conflicting optimization objectives. On one hand, there is a pressing need to decrease the energy consumption in these expansive data centers, aligning with global efforts to promote green clouds and reduce environmental impact. On the other hand, the imperative is to enhance resource utilization while minimizing SLA violations, which often entails keeping additional servers operational to handle peak workloads. Balancing these objectives is a challenging task.

Workload consolidation emerges as a resource management technique aimed at efficiently reallocating resources while simultaneously reducing energy consumption. By consolidating workloads onto a minimal number of physical servers, Infrastructure-as-a-Service (IaaS) providers can optimize the utilization of their hardware resources while reducing energy consumption through the powering off of under-utilized servers[4]. In virtualized data centers, this consolidation is possible through the live migration of VMs or containers hosting the running services, between physical machines (PMs) [5]. However, it is crucial to strike a delicate balance between resource consolidation and the preservation of application performance as defined in the Service Level Agreements (SLAs).

For instance, implementing an excessively aggressive consolidation approach can lead to violations and performance degradation for the applications running on the servers. Accordingly, many concerns should be addressed to tackle such a research problem. First, to achieve an effective workload consolidation, it is crucial to carefully estimate the host state and identify overloaded and underloaded hosts in order to make informed decisions about workload redistribution. Underloaded hosts that have excess available resources can be utilized for consolidating additional workloads from overloaded hosts to improve overall resource utilization and mitigate the risk of SLA violations. Alternatively, they can be turned off for minimizing energy consumption. The second concern involves selecting the right virtual resources (VMs or containers) to migrate from a server [6][7]. Lastly, a placement strategy is needed to select the best destination servers that can handle these migrated workloads [8]. Each of these concerns poses its own challenges.

Regarding overload and underload detection concern, existing workload consolidation approaches often rely on current resource utilization of servers to determine their state and trigger the required migrations [9][10]. However, such proposals may result in unreliable host state estimations and excessive needless migrations. An increase in the current server utilization does not necessarily reveal an overloading state, as the load may rapidly decrease in the next time slot. To make accurate estimations and limit the frequency of migrations, it is crucial to consider not only the current host workload but also its future resource utilization. Moreover, relying on a single future resource utilization value to judge the host state, may be also insufficient to perform reliable estimations. Therefore, it is important to anticipate and consider the future trend (as a sequence of multiple future values) of the host's resource usage.

Furthermore, some existing consolidation schemes focus on the utilization of a single resource type (i.e., CPU) while deciding whether a server is overloaded or underloaded [11][12]. Due to the sheer multiplicity and heterogeneity of running applications (e.g., IoT-based applications, 5G applications, web etc.) that may be hosted in the cloud, and the variability of their workloads, considering only one resource type can lead to a non-efficient decision-making strategy. Different applications may have different resource requirements, such as CPU-intensive, memory-intensive, or bandwidth-sensitive workloads, and so on. Therefore, it is necessary to consider all these resource types in order to build a technique able to correctly detect overloaded and underloaded servers across different application types and workloads.

In this article, we present a predictive workload consolidation mechanism based on prediction model called MSPR. This mechanism incorporates overload detection and underload detection algorithms, which take into account the current and predicted trend (instead of a single future value) of resource utilization to determine if a host is experiencing overloading or underloading issue. Unlike some existing techniques, our approach considers many resource types, including CPU, memory, and bandwidth, when making host state estimation. For overload detection, we calculate adaptive Mean Absolute Deviation (MAD) thresholds tailored to each resource type. Moreover, our approach offers flexibility by allowing the specification of distinct underload thresholds and prediction window sizes for each resource type. Our evaluation involves comparing our approach with an optimized multi-resource versions of benchmark consolidation algorithms integrated into Cloudsim, including Mean Absolute Deviation MAD-based, Interquartile Range IQR-based, Static threshold THR-based, and Local Regression LR-based approaches, in addition to our alternative consolidation approach that uses Autoregressive Integrated Moving Average (Arima) multi-resource prediction model, replacing MSPR model. In summary, our contributions include:

- A multi-resource and multi-step workload prediction model called MSPR is proposed, to anticipate the future resource utilization trends of servers in terms of CPU, memory, bandwidth received, and bandwidth transmitted. This model combines the well-known algorithms Support Vector Regression (SVR) and Kalman Filter. Kalman Filter is used as a data filtering pre-processing step to enhance the accuracy of SVR prediction process.
- A workload consolidation approach is combined with the MSPR predictive model to reduce the total energy consumption in data center, limiting the frequency of virtual resource migrations, and decreasing SLA violations. This approach includes OD-MSPR and UD-MSPR algorithms for overload and underload detection, considering both current and predicted multi-resource utilization trends of servers. Our approach also offers the flexibility to specify different overload and underload thresholds for each resource type. Adaptive thresholds for overload detection, based on MAD, are calculated for each resource type based on historical utilization data.
- Extensive experiments are conducted on Cloudsim using Bitbrains [13] and Materna [14][15] datasets to validate the effectiveness of our proposed approach compared to optimized multi-resource versions of state-of-the-art consolidation techniques, in addition to an alternative consolidation approach that replaces the MSPR model with an ARIMA multi-resource prediction model. Moreover, a detailed time complexity analysis of our predictive workload consolidation approach is provided

The rest of the article is organized as follows: Section II discusses the existing workload consolidation approaches in the state-of-the-art and highlights their limitations. Section III describes the proposed resource utilization prediction technique. Section IV explains in detail the predictive workload consolidation approach proposed and analyzes its time complexity. Section V presents the experimental setup and compares the results obtained by our proposal with existing benchmarks. Section VI summarizes the main insights and the future directions for this work.

II. Related Work

Workload consolidation and resource utilization prediction constitute two major research problems that can be tackled separately. In this section, we review some related proposals that address workload prediction and/or energy-efficient resource management problems.

A. *Workload Prediction*

Numerous studies about workload prediction has been provided in the pertinent literature [16]. Qui et al. [17] present a novel method for predicting the CPU utilization of VMs in a cloud computing environment. The proposed approach utilizes a deep learning model consisting of a Deep Belief Network (DBN) and a logistic regression layer. The parameters of the entire model are fine-tuned using the Backpropagation (BP) algorithm. The proposed approach is compared to other prediction methods, using PlanetLab dataset in Cloudsim. Experimental results demonstrate that the proposed method outperforms existing prediction approaches in terms of prediction accuracy. Malik et al. [18] present a multi-resource utilization prediction technique based on Functional Link Neural Network (FLNN). A hybrid model, that combines genetic algorithm (GA) with particle swarm optimization (PSO) algorithm, is used to train the neural network and thus, improve its prediction accuracy. Mean Absolute Error (MAE) is calculated as fitness function for GA. Their experiments are carried out using a Google cluster workload and are focused mainly on CPU and memory utilization of VMs. Xie et al.[19] propose a hybrid model of ARIMA and triple exponential smoothing to accurately predict both linear and nonlinear relationships in container resource load sequence. The weighting values of the two single models in the hybrid model are chosen according to the sum of squares of their predicted errors for a period of time. They also introduce a real-time resource prediction system for Docker container that optimizes CPU and memory resource usage based on predicted values. Khan et al. [20] propose an intelligent prediction model based on machine learning for workload prediction and energy state estimation for VMs in cloud data centers. The model explores different Machine Learning (ML) algorithms for workload prediction including Linear Regression (LR), Ridge Regression (RR), ARD Regression (ARDR), ElasticNet (EN) and deep learning algorithm called Gated Recurrent Unit (GRU). The obtained results show that the GRU achieved the most negligible root mean square error (RMSE) value compared to other ML algorithms. In addition to workload prediction, the authors propose four different clustering algorithms including semi-supervised affinity propagation based on transfer learning (TSSAP), CLA based on transfer learning (TCLA), k-means based on transfer learning (TKmeans), and P-teda based on transfer learning (TP-teda), for identifying similar groups of VMs with different energy-consuming states. Based on their experiments, the TSSAP outperformed other methods by achieving the highest accuracy in clustering.

In this article, we present a prediction approach called MSPR, for forecasting the utilization of server resources, encompassing CPU, memory, received bandwidth, and transmitted bandwidth. The proposed model leverages a combination of Support Vector Regression (SVR) and Kalman Filter algorithms to accurately predict future resource utilization. Kalman Filter acts as a pre-processing step to improve the accuracy of SVR predictions. This predictive model can be employed to anticipate workload of diverse systems, including servers, virtual machines (VMs), and containers etc. In our work, we concentrate on utilizing this model to forecast incoming workload for servers, thereby enabling reliable estimation of their future states.

B. Energy-Efficient Resource Management

In the state of the art, Researchers have explored different energy-efficient resource management approaches [21][22][23]. For instance, Nath et al. [8] propose EASY, an energy-efficient approach for container consolidation in cloud data centers. EASY utilizes a Bayesian optimization-based algorithm for container placement, to minimize energy consumption while considering trade-offs with service response time. The authors compare the performance of the EASY algorithm with baseline methods, including Consecutive Allocation, Best Fit, and First Fit Decreasing mechanisms. Simulation results demonstrate the effectiveness of EASY approach in reducing energy consumption, although slightly increasing average response time. Hariharan et al. [24] develop an adaptive beetle swarm optimization (ABSO) algorithm that combines the strengths of particle swarm optimization and beetle swarm optimization to optimize the placement and consolidation of virtual machines in a cloud environment. The fitness function considers energy consumption, migration cost, and utilization metrics. These proposals focus on the placement problem of VMs or containers. They do not address the overload and underload detection issue that may trigger re-placement actions and directly impact the performance and efficiency of the consolidation system.

Yadav et al. [25] suggest GradCent, an algorithm based on Stochastic Gradient Descent technique, for detecting overloaded hosts in cloud data centers. It determines an upper CPU utilization threshold based on CPU utilization history. They also introduce the Minimum Size Utilization (MSU) Algorithm, which prioritizes VMs with high CPU utilization and small sizes for migration from overloaded hosts. This proposal considers a single resource type (CPU). Songara et al. [26] propose a multi-resource VM consolidation approach called MRA-VC. The underloaded hosts are classified into different categories: severe load, moderate load, or low load based on their current multi-resource utilization score and predefined thresholds. The overload detection algorithm assigns dynamic weights to each resource based on their importance in the decision-making process. If the calculated weighted score exceeds an upper threshold (80%), the host is considered overloaded. Regarding VM selection and placement, the authors propose a modified VM selection and placement algorithm based on a particle swarm optimization. The approach utilizes predefined static thresholds to determine overloading scenarios. Additionally, the aforementioned approaches rely on actual resource utilization data only to assess the current state of hosts, determining if they are overloaded or underloaded.

Hieu et al. [27] propose a VM consolidation technique based on a multiple resource usage prediction model. The prediction technique employs multiple linear regression to forecast k future resource utilization of servers. A server is detected as overloaded in a resource d if its multiple predicted resource utilization U_{t+k}^{d} exceeds a hot threshold h. Whereas, it is defined as under-utilized in a resource d when its multiple predicted resource utilization U_{t+k}^{d} is less or equal to its current utilization U_t. Minarolli et al. [28] address the challenge of detecting overloaded hosts in cloud computing by making long-term predictions of resource demands for VMs. The authors employ Gaussian processes as a machine learning approach for time series forecasting. The approach constructs a probability distribution model of the prediction error, to quantify the uncertainty associated with the long-term predictions. Based on this model, a decision-theoretic approach utilizing a utility function is introduced to address the impact of live migration overheads in VMs. This approach selectively initiates live migration actions only when the anticipated penalty associated with SLA violations outweighs the utility value attributed to live migration overhead. Arshad et al. [29] propose a scheduling mechanism called Energy Efficiency Heuristic using VM Consolidation (EEHVMC), which consolidates VMs on host machines. By setting two thresholds T_{high} and T_{low}, EEHVMC mechanism classifies hosts in a cloud data center into three main categories: Host Over-Loaded, Host Medium-Loaded, and Host Under-Loaded. The mechanism identifies host state by comparing its CPU and memory utilizations to defined thresholds. Once overloaded and underloaded hosts are identified, specific VMs are selected for migration to medium-loaded hosts using a method called Maximum ratio of CPU utilization to memory utilization (MRCU). If a VM is identified as being CPU-bound or memory-intensive, it is considered as a candidate for migration. Sayadnavard et al. [30] present a multi-objective approach for dynamic VM consolidation in cloud data centers. The main objectives are to reduce energy consumption, improve system reliability, and minimize resource wastage. The proposed consolidation approach includes: a model that combines Discrete Time Markov Chain (DTMC) and Continuous Time Markov Chain (CTMC) for physical machines (PMs) categorization, a heuristic VM selection algorithm considering the task completion time, and a VM placement strategy using ε-dominance-based multi-objective artificial bee colony (ε-MOABC) algorithm. The approach is compared to traditional consolidation approaches integrated into CloudSim simulation platform. The comparison demonstrates the superiority of the proposed approach in achieving the defined objectives. Banerjee et al. [31] present a framework for efficient resource utilization in cloud environments by utilizing a multi-step-ahead workload prediction technique. The framework encompasses three key components: workload characterization, where agglomerative hierarchical clustering is used to identify VMs with similar resource usage patterns; workload prediction, employing and comparing a set of supervised machine learning models to forecast future resource consumption (CPU and memory); and VM placement, which utilizes a modified Best Fit Decreasing algorithm to allocate VMs based on predicted resource consumption values. Farahnakian et al. [32] present a utilization prediction-aware VM consolidation approach in cloud data centers called UP-VMC. The problem is formulated as bi-dimensional vector packing problem as two types of resource are considered: CPU and memory. Two regression-based prediction models are used for resource utilization prediction: Linear Regression (LR) and K-Nearest Neighbor Regression (K-NNR). A PM is considered as overloaded if its current or its predicted CPU or memory utilization exceeds its resource capacity. Their approach identifies the underloaded PM by comparing the current load of the PMs and selecting the PM with the lowest load. Most of the approaches previously discussed consider CPU and/or memory resources ignoring bandwidth usage and/or use predefined thresholds.

In our previous work [33], we proposed a consolidation approach based on a multi-step-ahead workload prediction model that combines Support Vector Regression with Kalman filter. However, our proposed algorithms focused solely on CPU utilization forecast and on static thresholds to make underload and overload decisions. In our current work, we have made significant enhancements to address these limitations. Firstly, we have optimized the prediction technique to forecast future utilization trends of multiple resources, including CPU, memory, and bandwidth. Additionally, we have revamped the underload and overload techniques to consider all these resource types when estimating the host state. Moreover, instead of using static thresholds, we now calculate adaptive Median Absolute Deviation (MAD) thresholds for overload detection, which vary for each resource type. These thresholds are determined based on historical utilization data for each specific resource. Another different aspect of our work is the redefinition of underloaded hosts. In our previous approach, hosts with the minimum CPU utilization were considered underloaded when there was insufficient data for prediction. With the integration of multi-resource considerations, we have redefined underloaded hosts to include those with actual resource utilizations falling below the underloaded thresholds for all resources. Furthermore, we have refined our objective metrics to include all considered resource types in the calculation of energy consumption and SLA violation. To ensure a formal comparative study, we have also improved the implementation of other approaches such as LR, THR, MAD, IQR, and Arima-based methods. Similar to our approach, these approaches now consider multi-resources in their overload and underload decisions. Lastly, for testing purposes, we have utilized two datasets, namely Materna and bitbrains, instead of the previously used planetlab dataset [34].

III. Workload Prediction Model

This section explains our **M**ulti-**S**tep **P**redictive Multi-**R**esource Utilization model, called **MSPR,** for forecasting host resource utilizations based on Support vector regression (SVR) and Kalman. SVR [35] is a well-known machine learning technique derived from Support Vector Machine (SVM) specifically to solve regression problems. It is suitable for the complex and dynamic cloud environment and is mainly used in our work to proactively predict future host resource utilization. Kalman Filter [36] is also a famous algorithm originally built to estimate the time-varying states in dynamic systems, which makes it suitable for the dynamic load estimation of cloud applications. Our prediction model integrates Kalman Filter as a data pre-processing step which aims to filter data, eliminate noises, and enhance the SVR prediction accuracy. In the following, the working principles of the aforementioned techniques are explained.

1) Kalman Filter

Kalman filter aims to estimate the state x of a discrete-time controlled process using a set of measurements observed over time. The following linear stochastic difference equation shows the evolution of the state x from time k-1 to time k:

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \tag{1}$$

The above equation can be combined with a measurement z, as follows:

$$z_k = Hx_k + v_k \tag{2}$$

Where A is the state transition matrix from time k-1 to time k. B is a control matrix that relates the control vector u to x. H is a matrix that illustrates the relation between x_k and z_k. In our work, there is no control input (B=0), and the measurement z is the state directly (H=1). Assuming that the state does not change from time step to another, A is set to 1. w_{k-1} and v_k represent the process and measurement noises respectively. They are random variables assumed to be white and independent of each other, with $w_{k-1} \sim N(0, Q)$ and $v_k \sim N(0, R)$. Q and R represent the process noise covariance matrix and the measurement noise covariance matrix respectively. In our approach, we integrate Kalman Filter as a data pre-processing step, to benefit essentially from its filtering technique that may eliminate noises from resource usage data, whatever these noises are coming from the measurements technique or other factors, while still discovering the main load fluctuations.

To estimate a process, Kalman filter iteratively applies two computation steps: (a) the prediction step that projects the state estimation ahead of time, and (b) the correction step that adjusts the projected estimate based on an actual measurement value at that time. The equations used in each of the mentioned steps are as follows:

Prediction phase

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1} \tag{3}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{4}$$

Correction phase

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{5}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{6}$$

$$P_k = (I - K_k H)P_k^- \tag{7}$$

Where $\hat{x}_k^-$ denotes the priori state estimate and $\hat{x}_k$ denotes the posteriori estimate at time k. Similarly, P_k^- is the priori estimate error covariance matrix, while P_k is the posteriori estimate error covariance matrix. K_k represents the Kalman Gain matrix. A high gain means that the filter mostly depends on the accurate measurements to estimate $\hat{x}_k$. Conversely, a low gain means that the state estimation mostly depends on the model predictions $\hat{x}_k^-$ calculated in the prediction phase.

2) *Support Vector Regression*

SVR is a statistical learning method that estimates a function $f(x)$ by training a SVM model using observed data. In our case, the observed data represent the historical host resource utilizations. By performing time series forecasts, the workload data are first divided into input and output datasets (X and Y respectively). Each combination of input/output vectors (x_i, y_i) represents a training point. Eq. (8) defines both linear and non-linear regression prediction functions:

$$f(x) = w\emptyset(x) + b \tag{8}$$

Where $\emptyset$ is a mapping function that non-linearly maps x from “input space” to higher dimension feature space. To simplify the mapping, a Radial Basis Function (RBF) is employed for its easier computation and fewer parameters compared to other functions. $f(x)$ denotes the predicted value, w is a weight coefficient, and b is a bias. The main goal is to find the optimal weights and thresholds according to two essential criteria. The first is the flatness of the weights, which is defined in terms of minimum Euclidean norm (e.g., minimize $\frac{1}{2} \parallel w \parallel^2$). The second is the empirical risk R_{emp} minimization, which denotes the error generated by the prediction process of the value. R_{emp} is computed using the ε-insensitive loss function L^ε. Combining the mentioned two sub-objectives, the overall objective is to minimize the regularized risk R_{reg} defined in Eq. (9) in order to find the flattest function that allows the error to remain within a threshold epsilon ε.

$$\begin{aligned} Minimize\ R_{reg} &= \frac{1}{2} \parallel w \parallel^2 + R_{emp} \\ &= \frac{1}{2} \parallel w \parallel^2 + C\sum_{i=1}^{N} L^\varepsilon\big(y_i, f(x_i)\big) \end{aligned} \tag{9}$$

Where

$$L^\varepsilon\big(y_i, f(x_i)\big) = \begin{cases} |\, y_i - f(x_i) - \varepsilon| & if\ |\, y_i - f(x_i)| \geq \varepsilon \\ 0 & otherwise \end{cases} \tag{10}$$

Where ε and C are user-defined constants. C determines the trade-off between the empirical and regularized risk. Finally, Slack variables, ξ_i and ξ_i^* should be added to estimate the error for underestimation and upper estimation of the actual value. In other terms, slack variables allow regression errors to exist up to the value of ξ_i and ξ_i^*, yet still satisfying the required conditions. Consequently, the equations are updated as follows:

$$Minimize\ R_{reg} = \tfrac{1}{2} \parallel w \parallel^2 + C\textstyle\sum_{i=1}^{N}(\xi_i^* + \xi_i) \tag{11}$$

$$Subject\ to\ \begin{cases} f(x_i) - w\phi(x_i) - b \leq \varepsilon + \xi_i^* \\ w\phi(x_i) + b - f(x_i) \leq \varepsilon + \xi_i \\ \xi_i, \xi_i^* \geq 0 \end{cases} \tag{12}$$

3) *MSPR Algorithm*

In our work, the MSPR workload forecasting model is multi-resource in the sense that it is used to predict multiple types of resources including CPU and memory host utilizations, and the network bandwidth received and transmitted. Monitoring tools can return a vast variety of other resources, such as requests per second, delay, disk read/write throughput, etc. However, all these resources are related with the typical resources considered in a Cloud, such as CPU, memory, and bandwidth. To this end, in this work, we have considered these three types of resources since they are the most representative Cloud resources. In the prediction process, we do not consider the correlation between the resources. Since we have performed various predictions scenarios in the past, that allowed us to infer that the single-resource prediction yields better accuracies than the multi-source prediction [37]. Hence, the utilization of each resource type $r \in R$ is forecasted according to its historical data and can have different window size WS for prediction and a number of prediction steps n. MSPR model performs also multi-step-ahead predictions, meaning that it forecasts multiple future resource utilization values instead of just one value [33]. As shown in Algorithm 1, for each resource r, MSPR takes as input the host's historical data $History_s^r$, the pre-specified WS and n. The data are first filtered by Kalman filter and then divided into X and Y to train the SVM model as explained previously. Finally, the trained model is used to predict the future n values of the host's resource usage.

Algorithm 1: MSPR prediction algorithm

```
1:  Input: $History_s^r$, $WS_r$, $n$
2:  Output: $\widehat{U_{t+n}^r(s)}$ /* set of n predicted values */
3:  /* Preprocess data */
4:  Filtered_history = Kalman_filter ($History_s^r$)
5:  /* extract and divide training dataset into X and Y*/
6:  Training_data = extract (Filtered_history, $WS_r$)
7:   Xtrain, Ytrain = divide (Training_data)
8:  /* Train SVM model */
9:  svmTrain(Xtrain, Ytrain)
10: /* extract testing data */
11: Xtest = extract (Filtered_history, n)
12: $\widehat{U_{t+n}^r(s)}$ = svmPredict (Xtest)
13: Return $\widehat{U_{t+n}^r(s)}$
```

IV. Workload consolidation approach

In this section, we present our workload consolidation approach based on the multi-resource and multi-step prediction model discussed in the previous section. We first describe all algorithms that constitute this approach, namely, the proposed **O**verload **D**etection algorithm based on MSPR prediction model (OD-MSPR), and **U**nderload **D**etection algorithm based on MSPR prediction model (UD-MSPR), and the placement strategy. Then, we analyze the overall time complexity of this proposal.

A. Overload detection algorithm

In our approach, a server s is considered overloaded, if one of the following situations is met:

- It is overloaded in both current and future utilizations of at least one of its resources R. Precisely, if its current utilization $U_t^r(s)$ and the average of its n predicted utilizations $U_{t+n}^r(s)$ of any of its resources $r \in R$ exceed the upper threshold T^{up}.

  $for\ any\ r \in R$:

$$U_t^r(s) > T_r^{up}\ and\ Avg\left(\widehat{U_{t+n}^r(s)}\right) > T_r^{up}$$

- It is currently working normally but will be overloaded in the future time slots in at least one resource type $r \in R$.

  $for\ any\ r \in R$:

$$Avg\left(\widehat{U_{t+n}^r(s)}\right) > T_r^{up}$$

T_r^{up} is an upper adaptive threshold based on the Median Absolute Deviation technique. We assumed that a different upper threshold for each resource type r may be needed. The resources considered in our decision-making process are: CPU, memory, and bandwidth received by the server and transmitted. An overloading situation is detected when a server is overloaded in at least one of these resources (e.g., ***if*** *Avg* ($U_{t+n}^{\widehat{CPU}}(s)$) > T_{CPU}^{up} *or Avg* ($U_{t+n}^{\widehat{mem}}(s)$) > T_{mem}^{up} *or Avg* ($U_{t+n}^{\widehat{bwR}}(s)$) > T_{bwR}^{up} *or Avg* ($U_{t+n}^{\widehat{bwT}}(s)$) > T_{bwT}^{up}).

We argue that considering all resource types is important to estimate the server's state. For instance, in a datacenter, heterogeneous applications can be hosted including CPU-intensive, memory-intensive, and bandwidth-intensive applications. Accordingly, a server hosting memory-intensive application may be overloaded in memory but not in other resources. Checking the current server utilization for all these resources and anticipating the future values will help to better detect the various overloading situations in a more holistic manner. The pseudocode of the proposed overload detection algorithm (OD-MSPR) is given in Algorithm 2. It takes an active server $s \in S_{active}$ as input, and then decides whether it is overloaded or not. In particular, for each resource $r \in R$, it verifies if there is sufficient historical data for prediction. If data is not sufficient, the decision is made by comparing the current resource utilization with the threshold T_r^{up} (Steps 5-8). Otherwise, it predicts the future n utilizations of this resource using MSPR prediction model and compares the average of these predicted values with the threshold (Steps 9-14). At the end, the algorithm returns true if an overloading problem is detected in one of the server's resources.

Algorithm 2: OD-MSPR

```
1:  Input: $s \in S_{active}$
2:  Output: Boolean decision if s is overloaded or not
3:  For $\forall r \in R$ do
4:      Get $History_s^r$, $T_r^{up}$, $WS_r$ , $n$
5:      $T_r^{up} = getHistoryMAD(History_s^r)$
6:      if Length ($History_s^r$) < $WS_r$ then
7:          if $U_t^r(s) > T_r^{up}$ then
8:              return true
9:          end
10:     else
11:         $\widehat{U_{t+n}^r(s)}$ = MSPR ( $History_s^r$, $WS_r$, n)
12:         if Avg ( $\widehat{U_{t+n}^r(s)}$) > $T_r^{up}$ then
13:             return true
```

```
14:         end
15:     end
16: End
17: Return $false$
```

B. Underload detection algorithm

In this algorithm, a server s is defined as underloaded, if one of the following conditions is satisfied:

- It is underloaded in both current and future utilizations of all its resources R. In particular, if its current utilization $U_t^r(s)$ and the average of its multiple predicted utilization values $U_{t+n}^r(s)$ for each of its resources $r \in R$ are below the lower thresholds T_r^{Under}.

  $for\ all\ r \in R$:

  $U_t^r(s) <= T_r^{Under}\ and\ Avg\big(\widehat{U_{t+n}^r(s)}\big) <= T_r^{Under}$

- It is predicted to be underloaded in the future in all its resources R.

  $for\ all\ r \in R$:

  $$Avg\big(\widehat{U_{t+n}^r(s)}\big) <= T_r^{Under}$$

In our approach, R includes CPU, memory, and bandwidth received and transmitted. Thus, an underloading state is detected, if and only if the server is underloaded in all these resources (e.g., ***if*** *Avg*($\widehat{U_{t+n}^{CPU}(s)}$) <= T_{CPU}^{under} *and Avg* ($\widehat{U_{t+n}^{mem}(s)}$) <= T_{mem}^{under} *and Avg* ($\widehat{U_{t+n}^{bwR}(s)}$) <= T_{bwR}^{under} *and Avg* ($\widehat{U_{t+n}^{bwT}(s)}$) <= T_{bwT}^{under})). We assumed that a different lower threshold may be required for each resource. If an underloaded server is found, all VMs or containers hosted on this server should be migrated to other hosts if possible, and consequently, it will be switched to a low-power mode to save energy. The detailed pseudocode of the underload detection algorithm based on MSPR prediction model is illustrated in Algorithm 3. It iterates through the set of active servers S_{active} and searches if there is any underloaded host. Note that the overloaded servers detected by Algorithm 2 are excluded from S_{active}. To check a server state, it iterates through each of its resources $r \in R$ and verifies if the available historical data are enough for prediction. If no sufficient data are available to predict the resource, the algorithm compares the server's current utilization with the threshold. Otherwise, it compares the average of the future utilizations predicted by MSPR model, with the pre-specified threshold. The server is not considered underloaded if one of its resources exceeds its lower threshold.

<u>Algorithm 3:</u> UD-MSPR

```
1:  Input: $S_{active}$
2:  Output: An underloaded server
3:  For $\forall s \in S_{active}$ do
4:      IsUnderloaded = true
1:      For $\forall r \in R$ do
5:          Get $History_s^r$, $T_r^{Under}$, $WS_r$, $n$
2:          if Length ($History_s^r$) < $WS_r$ then
3:              if $U_t^r(s) > T_r^{Under}$ then
6:                  IsUnderloaded = false
7:                  Break
8:              end
4:          else
9:              $\widehat{U_{t+n}^r(s)}$ = MSPR ($History_s^r$, $WS_r$, $n$)
10:             if Avg ($\widehat{U_{t+n}^r(s)}$) > $T_r^{Under}$ then
11:                 IsUnderloaded = false
12:                 Break
13:             end
14:     End
15:     if (IsUnderloaded) then
16:         Return s
17:     end
18: End
19: Return Null
```

C. Migration and placement

After detecting overloaded and underloaded servers, the next step is to perform some migrations for the VMs or containers hosting the running applications on these servers. To perform a formal comparison between our approach and the techniques proposed in [10], we have re-used their VM selection approach and placement strategy. In particular, we have used the Minimum Migration Time (MMT) to select the VMs to migrate from overloaded hosts, and the Power Aware Best Fit Decreasing (PABFD) strategy to find destination servers for migrated VMs. However, these algorithms are modified to use our OD-MSPR algorithm.

MMT-MSPR algorithm (Algorithm 4) iterates through the list of VMs V^s hosted on an overloaded server s and then selects for migration the VMs that have the least migration time. Migration time is measured by dividing the RAM utilized by a VM v by the available network bandwidth: $D_v^{migration} = \frac{RAM_v}{B_v}$. A set of VMs may be selected until the overloading issue is solved. Thus, after each VM selection, the algorithm verifies if the server will remain overloaded after deallocating the selected VM or not (Steps 14-17). This verification is done by calling our OD-MSPR algorithm explained in section IV.A.

PABFD-MSPR strategy (Algorithm 5) iterates through the list of VMs to migrate and tries to find a destination server for each that meets certain criteria. First, the destination host should have sufficient capacity to meet the VM resource requirements in terms of CPU, memory, bandwidth, and disk (Step 7). Second, the candidate server should not become overloaded after hosting the VM (Steps 9-11). To verify the host state, the algorithm simulates the VM allocation and then uses OD-MSPR algorithm (Algorithm 2) to check the server state. Third, the selected server should have the least increase in its power consumption caused by this allocation because energy consumption is one of our main objectives in this work (Steps 12-17). At the end, the algorithm returns the migration map that includes the suitable destination hosts for the target VMs. Once the VMs to be migrated and their destinations are selected, a pre-copy live migration is applied to move them from their current hosts to the chosen ones.

Note that, in this work, we have focused mainly on resource prediction, overload detection, and underload detection parts. We have combined our proposed techniques (MSPR model, OD-MSPR, UD-MSPR) with simple VM selection and placement strategies to conduct our test experiments. However, our techniques can be combined with other advanced strategies, and be employed to consolidate workloads for other types of applications or services (e.g., containerized applications by using container migration strategies, virtual network functions consolidation by considering some migration constraints related to their service function chain requirements, etc.).

Algorithm 4: MMT-MSPR algorithm

```
1:  Input: $s \in S_{over}$
2:  Output: List vmsToMigrate
3:  While (true) do
4:      Set min_time = MAX
5:      CandidateVM = NULL
6:      foreach $v \in V^s$ do
7:          $D_v^{migration} = \frac{RAM_v}{B_v}$
8:          if $D_v^{migration}$ < min_ time then
9:              min_ time = $D_v^{migration}$
10:             CandidateVM = $v$
11:         end
12:     end
13:     vmsToMigrate.add (CandidateVM)
14:     /* implicity call OD-MSPR(s) *
15:     if overloadedAfterDeallocation (s, v) = false then
16:         Break
17:     end
18: end
19: Return vmsToMigrate
```

Algorithm 5: PABFD-MSPR

```
1:  Input: $S_{active}$, List vmsToMigrate
2:  Output: Migration Map
3:  For $\forall v \in vmsToMigrate$ do
4:      minPower = Max
5:      destinationServer = NULL
6:      For $\forall s \in S_{active}$ do
7:          if s.hasSufficientCapacity(v) then
8:              /* implicity call OD-MSPR(s) */
9:              if overloadedAfterAllocation(v, s) then
10:                 Continue
11:             end
12:             oldPower = s.getPower()
13:             newPower =
    estimatePowerAfterAllocation(v, s)
14:             powerDiff = newPower – oldPower
15:             if powerDiff < minPower then
16:                 minPower = powerDiff
17:                 destinationServer = s
18:             end
19:         end
20:     end
21:     if destinationServer is not NULL then
22:         migrationMap.add(v, destinationServer)
23:     end
24: end
25: Return migrationMap
```

D. Overall approach

The overall predictive workload consolidation approach is presented in Algorithm 6. It is executed periodically to manage the cloud resources in two sequential procedures: (a) Overload Avoidance Phase (OAP) (Steps 2-10); and (b) Resource Wastage Avoidance Phase (RWAP) (Steps 11-27). OAP aims to release some resources from overloaded servers to avoid SLA violations. It starts by checking the hosts' states and detecting overloaded ones using Algorithm 2 (OD-MSPR). Then, it selects the virtual resources to migrate from these servers using Algorithm 4 (MMT) and chooses the destination hosts for the migrated VMs by executing Algorithm 5 (PABFD-MSPR). To start RWAP, the list of active servers S_{active} is first updated to exclude the overloaded servers list S_{over} and the destination hosts $S_{destinations}$ selected in OAP phase, because these servers should not be turned off (Step 11). RWAP aims to switch off the underloaded servers to optimize resource utilization in the data center and save energy. Through continuous iterations, the algorithm checks if there is any underloaded server in the data center using Algorithm 3 (UD-MSPR). If an underloaded server is detected, it tries to find destination hosts for all virtual resources running on this server using Algorithm 5 (PABFD-MSPR). If and only if all hosted VMs can be migrated to other destinations, the underloaded server can be turned off. Otherwise, the server remains active and all migrations planned from this server are canceled. In the following, the time complexity of the overall approach is detailed.

Algorithm 6: workload consolidation approach

```
1:  Input: $S_{active}$, V
2:  /* start of OAP Phase */
3:  for $\forall s \in S_{active}$ do
4:      if OD-MSPR(s) then
5:          $S_{over}$.add (s)
6:          vmsToMigrate.add (MMT-MSPR( s ))
```

TABLE I
POWER CONSUMPTION OF HOSTS ACCORDING TO THEIR CPU USAGE (IN WATTS)

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

```
7:      end
8:  end
9:  migrationMap = PABFD-MSPR ($S_{active}$,
    vmsToMigrate)
10: $S'_{active} = S_{active} - (S_{over} \cup S_{destinations})$
11: /* Start of RWAP Phase */
12: While (true) do
13:     U_server = UD-MSPR ($S'_{active}$)
14:     if U_server = NULL then
15:         Break
16:     end
17:     Exclude U_server from $S'_{active}$
18:     $S_{under}$.add(U_server)
19:     MigrationMap2 = PABFD-MSPR ($S_{active}$, $V^s$)
20:     if migrationMap2 is complete then
21:         migrationMap.addAll(migrationMap2)
22:         U_server can be turned off after migrations
23:     else
24:         Discard migrationMap2
25:         U_server will remain active
26:     end
27: End
```

E. Complexity Analysis

In this section, we analyze step by step the time complexity of the overall predictive workload consolidation approach and its main phases (OAP and RWAP phases) described in section IV.D and Algorithm 6. The following notation is used to facilitate the complexity analysis: A is the number of active servers in the system; N denotes the total number of virtual resources (VMs or containers), N_{V^s} represents the number of virtual resources running on a server s; $N_{V^{mig}}$is the number of virtual resources selected for migrations; and H_s^r is the historical data length of each resource $r \in R$ of a server s; D is the dimensions or the number of considered resources R.

1) Complexity – OAP Phase

Starting with line 3 of Algorithm 6, the time complexity of the for loop is equal to the number of active servers $O(A)$. Inside the loop, Algorithm 2 (OD-MSPR) is called. Its time complexity depends mainly on MSPR algorithm (Algorithm 1). The time complexity of Kalman Filter is analyzed in [38] as $O(4n^3)$ with n is the state vector size. In our approach, kalman is used to filter historical data before proceeding with SVR prediction, and so its complexity is $O(4\,{H_s^r}^3)$. In [39], the time complexity of SVM in LibSVM library which was used to complete our implementation is discussed. According to their analysis, the worst complexity for svmPredict and svmTrain is $O(n^3)$ where n is the amount of data used in training and in prediction respectively. In our prediction model, WS represents the prediction window size and n is the number of prediction steps. Thus, the complexity of MSPR is $O(4\,{H_s^r}^3 + WS^3 + n^3)$. Going back to Algorithm 2 (OD-MSPR), the algorithm iterates through the R resources of the server (D dimensions) and calls MSPR to predict each of them. Its complexity is then $O(D \cdot (4\,{H_s^r}^3 + WS^3 + n^3))$. In line 6, MMT algorithm (Algorithm 4) is used to choose the VMs to migrate from an overloaded server. This algorithm also calls (OD-MSPR) to verify if the server remains overloaded after the deallocation of each selected VM. Its complexity is $O(N_{V^s} \cdot D \cdot (4\,{H_s^r}^3 + WS^3 + n^3))$. Therefore, the total complexity of the for loop (lines 3-8) is $O(A \cdot N_{V^s} \cdot D^2 \cdot (4{H_s^r}^3 + WS^3 + n^3)^2)$.

After the loop, Algorithm 5 (PABFD-MSPR) at line 9 is called to select destination hosts for migrated VM. Its complexity is $O(N_{V^{mig}} \cdot A \cdot \mathrm{D} \cdot (4{H_s^r}^3 + WS^3 + n^3))$ because it also uses OD-MSPR algorithm to check the status of the potential destination server after allocating the target VM. Hence, the total complexity of OAP phase is $O(A \cdot N_{V^s} \cdot D^2 \cdot (4{H_s^r}^3 + WS^3 + n^3)^2) + O(N_{V^{mig}} \cdot A \cdot \mathrm{D} \cdot (4{H_s^r}^3 + WS^3 + n^3))$. However, H_s^r, WS, n, and D are typically small numbers and the complexity can be represented by $O(A \cdot N_{V^s}) + O(N_{V^{migrate}} \cdot A)$.

2) Complexity – RWAP Phase

In RWAP phase (lines 12-27), Algorithm 3 (UD-MSPR) is called to find underloaded hosts. Its complexity is based also on MSPR model and can be illustrated by $O(A \cdot D \cdot (4{H_s^r}^3 + WS^3 + n^3))$. Then, Algorithm 5 (PABFD-MSPR) is executed to find destination hosts for the VMs running on the detected underloaded host. In this case, $N_{V^{mig}} = N_{V^s}$ and the complexity of the latest algorithm is $O(N_{V^s} \cdot A \cdot \mathrm{D} \cdot (4{H_s^r}^3 + WS^3 + n^3))$. Therefore, the complexity of this phase is $O(A^2 \cdot D^2 \cdot (4{H_s}^3 + WS^3 + n^3)^2 \cdot N_{V^s})$, and can be simplified to $O(A^2 \cdot N_{V^s})$.

3) Overall Complexity

The overall complexity can be calculated by the summation of the complexities of OAP and RWAP phases. Consequently, it is equal to $O(A \cdot N_{V^s}) + O(N_{V^{migrate}} \cdot A) + O(A^2 \cdot N_{V^s})$. Again, to simplify it, the total number of active servers can be approximated by dividing the total number of VMs by the number of VMs that can be allocated to a server ($A \approx \frac{N}{N_{V^s}}$). The modified complexity will be $O(N) + O\left(N_{V^{migrate}} \cdot \frac{N}{N_{V^s}}\right) + O(\frac{N^2}{N_{V^s}})$. Finally, the worst-case complexity is $O(N^2)$.

F. Performance Metrics

In this study, our main objective is to decrease energy consumption and minimize the violations rate of service level agreements (SLAs). To assess the effectiveness of our algorithms, we utilize the following metrics.

- SLA violation

SLA represents a contractual agreement between a cloud service provider and its customers, defining the desired quality of service (QoS). Within the SLA, Service Level Objectives (SLOs) specify the QoS measurements and constraints. Meeting these requirements is crucial for evaluating the quality of the cloud service and avoiding penalties. We evaluate SLA violations based on two SLO parameters: SLA violation due to host overloading issue ($SLOH$) and SLA violation due to resource under-provisioning per VM ($SLOVM$).

$SLOH$ measures the average ratio of time during which a host is fully utilized. When a host's resource utilization (e.g., CPU, memory, bandwidth) reaches 100%, it may fail to provide VMs with the necessary resources, resulting in degraded performance. $SLOH$ can be calculated using the following formula:

$$SLOH = \frac{1}{A}\sum_{i=1}^{A}\frac{\max_{R}(T_{O_i}^{r})}{T_{a_i}} \quad \forall r \in R \qquad (13)$$

Where A is the number of hosts; $T_{O_i}^{r}$ is the total time during which the host i experiences 100% utilization of resource r; T_{a_i} is the total time in which host i is active.

$SLOVM$ measures the average violation caused by resource under-provisioning to VMs. It is calculated by comparing the allocated amount of each resource r to the requested amount. The formula is as follows:

$$SLOVM = \sum_{r=1}^{R}\frac{1}{N}\sum_{j=1}^{N}\frac{requested^{r} - allocated^{r}}{requested^{r}} \qquad (14)$$

Where N represents the number of VMs.

- Energy consumption

We evaluate the total energy consumed by the physical machines in a data center. The energy consumed by each server i is calculated by summing the power consumption of each resource type $r \in R$ including CPU, memory, and bandwidth (equation 16). The power consumed by each resource type is calculated using the formulas 17-22.

$$\text{Total Power} = \sum_{i=1}^{A} P_i \qquad (15)$$

$$P_i = \sum_{r=1}^{R} P_i^{r} \quad \forall r \in R \qquad (16)$$

$$P_i^{CPU} = specPower(u^{CPU}) \qquad (17)$$

$$P_i^{RAM} = u^{RAM} \cdot P_{max}^{RAM} \qquad (18)$$

$$P_{max}^{RAM} = \frac{RAM_i}{x} \qquad (19)$$

$$P_i^{BW} = P_i^{static} + P_i^{dynamic} \qquad (20)$$

$$P_i^{static} = P_{NIC}^{idle} + \sum_{l=1}^{L} P_l \qquad (21)$$

$$P_i^{dynamic} = u^{BW} \cdot P_{max}^{BW} \qquad (22)$$

$$P_{max}^{BW} = \frac{BW_i}{y} \qquad (23)$$

The processing power measurements (consumed by CPU) are derived from real data obtained from SPECpower benchmark results [40]. Table I provides the power consumption of the servers HP G4 and G5 at various loads. The power consumed by the RAM is calculated by multiplying the RAM utilization of the server by the maximum potential power consumption of this resource (equation 18). Equation 19 is used to calculate this maximum power, where RAM_i represents the total memory of server i, and x is an input value that can be easily updated [41]. To provide a specific input value, we assume that each 3 GB of RAM consumes 1 watt. The power consumed by the bandwidth consists of static and dynamic power components. The static power is considered constant and is calculated by summing the idle power of the network card with the power increase in relation to the number of active links L (equation21). In our testing, we assume that the utilized NIC is an intel Multiport (4*1G) with only one active link, and the idle power consumption is 9 watts [42]. Whereas, the dynamic power is associated to the bandwidth utilization of the server and is calculated using equation 22. The maximum potential power for this resource is calculated by dividing the total bandwidth of the server BW_i by an input value y. To specify this input value, we consider that the max active power of intel multiport (4*1G) NIC is 1 watt for 0.45 Gbps [42] .

- Number of migrations

Minimizing the number of VM migrations is important to avoid negative impacts on application performance. Live migrations incur additional costs, including increased resource utilization on the source host, network bandwidth usage, service delay due to downtime during migration, and total migration time.

- Execution Time

We also compare the algorithms based on their execution time. Specifically, we measure the average time required to complete an entire consolidation cycle, including the steps of overloaded host detection, underloaded host detection, VM selection for migration, and VM placement.

TABLE II
VM INSTANCES CHARACTERISTICS

VM Instance Type	CPU (MIPS)	RAM (GB)	Bandwidth (Mbits/s)
High-CPU medium instance	2500	0.85	100
Extra-large instance	2000	3.75	100
Small instance	1000	1.7	100
Micro instance	500	0.613	100

V. EXPERIMENTS

A. Setup

1) Environment

We have conducted simulations using the CloudSim toolkit [43] to test our proposed algorithms. Our testing environment consists of 800 heterogeneous servers divided as follows: 400 HP ProLiant ML110 G4 machines with dual-core processors, each having 1860 MIPS, and 400 HP ProLiant ML110 G5 machines with dual-core, each having 2660 MIPS. Both server types are equipped with 4 GB of memory and 1 GB/s of bandwidth. The characteristics of VM instances are provided in Table II. To implement our *MSPR* prediction model, LibSVM library [44] is used in Java. We have conducted tests using different threshold values (ranging from 20% to 30%) for underload detection, and different windows sizes (8, 12, 16, 20, 24, and 28) for prediction. The underload threshold T_r^{Under} is set to 30% and the prediction window size WS_r to 20 for all resources. However, it is possible to set different threshold and window size for each resource r. Additionally, we have set the number of prediction steps n to 3, to predict three future utilizations of each resource type for the host. Nevertheless, our implementation is not limited to 3, and the value of n can be easily adjusted. All testing parameters are summarized in Table III. These input values can be easily modified according to specific requirements.

TABLE III
TESTING PARAMETERS

Kalman filter	A	H	Q	R
	1	1	0.01	1
SVR	ε	kernel	γ	C
	0.1	RBF	0.0625	1
Consolidation and prediction	T_r^{Under}	WS_r	n	
	30%	20	3	
Arima	p	d	q	
	1	0	1	

T_r^{Under}= Underload threshold for each resource r ; WS_r = Windows Size; n = number of prediction steps; Gamma (γ) = parameter of the RBF.

2) Datasets

Our simulation utilizes two publicly available real-world datasets: Bitbrains [13] and Materna [14][15].

The Materna dataset is derived from a well-established full-service provider with a significant history of successful ITC projects for top-tier German companies and public sector organizations. Importantly, the dataset includes VMs hosting highly critical business applications of internationally recognized companies, making it particularly relevant for assessing VM performance in real-world scenarios. It encompasses performance metrics from a distributed datacenter, making it reflective of a diverse set of workloads. It includes three distinct traces, each representing one month of data collection. The first trace comprises 520 VMs, the second trace consists of 527 VMs, and the third trace encompasses 547 VMs. The presence of three traces, each representing a different month's data, offers insights into potential variations over time. To ensure an adequate number of VMs for testing purposes, we have combined certain traces, as indicated in Table IV.

The Bitbrains dataset provides a different perspective. Bitbrains specializes in managed hosting and business computation for enterprises, including major banks, credit card operators, and insurers. This diverse clientele implies a wide range of workloads hosted in their datacenter. The dataset is divided into two traces, fastStorage and Rnd, reflecting VMs connected to different storage devices. The composition of VMs in these traces, including application servers, compute nodes, and management machines, underscores the dataset's diversity and its relevance to various VM performance scenarios. The fastStorage trace comprises 1,250 VMs connected to high-speed storage area network (SAN) devices. The Rnd trace consists of 500 VMs that are connected to either fast SAN devices or slower Network Attached Storage (NAS) devices. The Rnd trace is further divided into three sub-traces, each corresponding to a specific month when the metrics were recorded. While we acknowledge that no dataset can fully encapsulate all possible scenarios, we believe that Materna and Bitbrains datasets provide valuable insights into VM performance in real-world environments.

Before conducting the testing process, we have carried out a pre-processing phase on the datasets. This phase involves converting the datasets into a suitable format. Furthermore, by using the following linear transformation formula, we have normalized the resource utilization data.

$$x_k^n = \frac{x_k - x_{min}}{x_{max} - x_{min}} \tag{24}$$

where x_k^n represents the normalized resource utilization value calculated based on the original data x_k. x_{max} and x_{min} are the minimum and maximum values of x_k, respectively. Furthermore, we have performed data cleaning procedures to remove histories with insufficient data or null values. After

TABLE IV
DATASETS CHARACTERISTICS

Workloads	Datasets	Traces	Number of VMs	Number of servers
W1	Bitbrains	Trace Fast storage	1237	800
W2	Bitbrains	Rnd (3 traces)	1500	800
W3	Materna	Traces 1-3	1063	800
W4	Materna	Traces 1-2	1043	800
W5	Materna	Traces 3-2	1074	800

completing the pre-processing phase, the obtained resource utilization dataset is used to conduct our testing.

3) *Benchmarks comparison*

To demonstrate the efficiency of our approach, we have conducted a comparison with modified versions of consolidation techniques integrated into the Cloudsim toolkit [10]. Specifically, we consider four consolidation strategies where overload detection depends on: Static Threshold (THR), InterQuartile Range (IQR), Median Absolute Deviation (MAD), and Local Regression (LR). These approaches have been adapted to be multi-resource and to consider all resources in their overload and underload decisions. Similar to our approach, they detect an overloading situation when a server is overloaded in at least one of its resources, including CPU, memory, and bandwidth. An underloading state is detected only if the server is underloaded in all of these resources. Two different versions of underload detection are considered for these benchmarks. In the first experiment, underloaded hosts are identified as those whose actual resource utilizations are lower than the underload thresholds for all resources. In the second experiment, our predictive underload detection algorithm, UD-MSPR, discussed in section IV.B., is incorporated into these benchmarks. This integration allows a stronger comparison between different overload detection techniques and a clearer interpretation of the obtained results. To perform a formal comparison, all algorithms use a common VM selection strategy, Minimum migration time (MMT), and employ the same VM placement method explained in sub-section IV.C. All evaluations are made based on the performance metrics described in section IV.F.

In addition to the benchmark algorithms mentioned, we test our proposed consolidation approach against an Arima-based multi-resource consolidation technique. Instead of using the Kalman-SVR combination for resource utilization prediction, our consolidation technique is updated to employ Arima. The purpose of implementing this alternative technique is to evaluate our proposed approach against another predictive consolidation mechanism. Details of the experimental results are presented in the following sub-section.

B. *Results and discussion*

1) *Experiment 1*

Figures 1-5 illustrate a comparison of our MSPR-based consolidation approach with an alternative Arima-based consolidation approach and optimized multi-resource versions of Cloudsim benchmarks discussed in subsection V.A.3. In this experiment, Cloudsim benchmarks identify underloaded hosts as those whose actual resource usage are below the underload thresholds for all resources.

From the results depicted in Figure 1, our MSPR-based consolidation approach outperforms all approaches in term of energy consumption reduction. It demonstrates a noteworthy reduction in total energy consumed in the datacenter, averaging 9.384%, 14.126%, 19.270%, and 23.223% lower compared to LR, MAD, IQR, and THR, respectively. Arima-based approach ranks second in minimizing energy consumption. Among the benchmark algorithms, LR performs optimally in terms of energy optimization. Notably, our UD-MSPR and OD-MSPR algorithms effectively identify underloaded and overloaded hosts, enabling the migration of VMs from these hosts to alternate machines and transitioning idle hosts into sleep mode to conserve energy.

Moreover, by estimating the trend of future resource utilizations, our approach allows for proactive measures to prevent overloading situations and potential SLA violations. Consequently, our approach significantly reduces SLA violations resulting from host overloading ($SLOH$) compared to the other Cloudsim benchmarks, as shown in Figure 2. MSPR-based approach yields the best outcomes in reducing $SLOH$, with the Arima-based approach ranking second. LR is the worst among the benchmarks in reducing $SLOH$ in the first two workloads, but THR is the worst in the others. In terms of the average SLA violation per VM caused by resource under-provisioning ($SLOVM$), Figure 3 demonstrates that both the MSPR and Arima-based approaches outperform the other methods across all tested datasets. The MSPR-based approach reduces the $SLOVM$ by an average of 95.871%, 77.953%, 81.075%, and 87.188% compared to LR, MAD, IQR, and THR, respectively. Moreover, it achieves a 15.278% average reduction in $SLOVM$ compared to the Arima-based approach.

Once overloaded and underloaded hosts are identified, effective VM migration plans can be applied to readjust resource allocations and alleviate the issue. Figure 4 presents the comparison results in terms of the number of migrations. Remarkably, our MSPR-based approach dramatically minimizes the number of migrations across all datasets compared to the Cloudsim benchmarks. For example, for dataset W1, the MSPR-based technique initiates 6378 migrations, while the other approaches perform 18695 (LR), 16940 (MAD), 18027 (IQR), 17944 (THR), and 7273 (Arima) migrations. Minimizing the number of migrations is desirable as it reduces system overhead, extra expenses, and potential violations. Furthermore, avoiding unnecessary migrations contributes to a more efficient reallocation process, resulting in reduced runtime encompassing VM selection and destination host determination. Figure 5 provides insights into the runtime performance, indicating that MSPR-based technique achieves lower execution time compared to other approaches. Arima-based technique occupies the second-best runtime. Overall, MSPR-based approach strikes a favorable balance between energy consumption and SLA violation, outperforming the compared algorithms in almost all studied metrics.

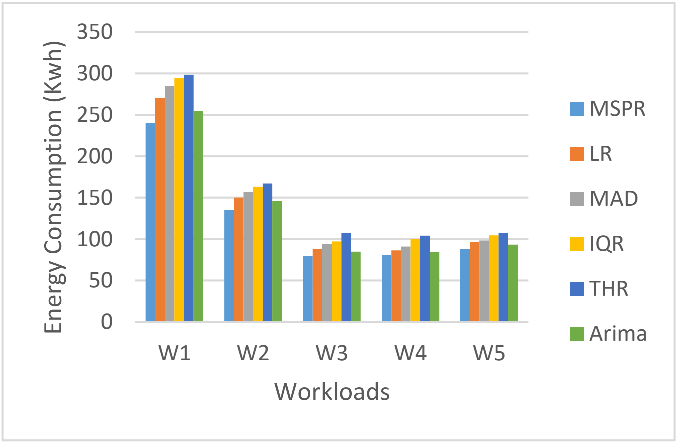

Figure 1. Comparison of energy consumption for 5 workloads- experiment 1.

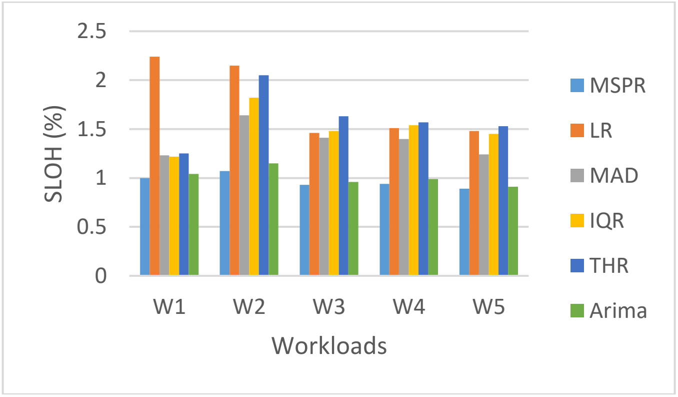

Figure 2. Comparison of SLOH metric for 5 workloads- experiment 1

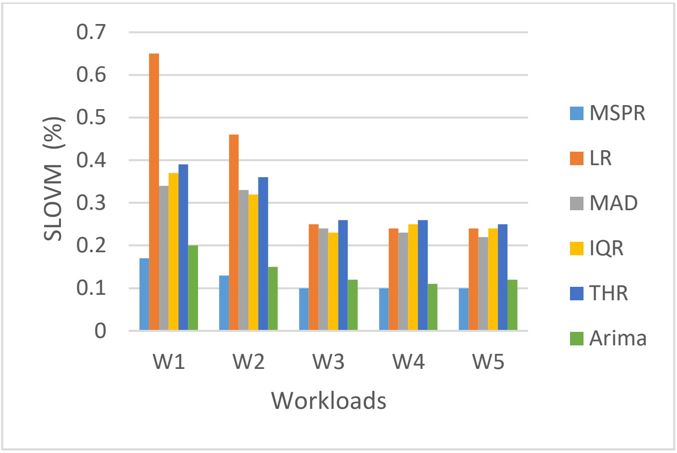

Figure 3. Comparison of the SLOVM metric for 5 workloads- experiment 1.

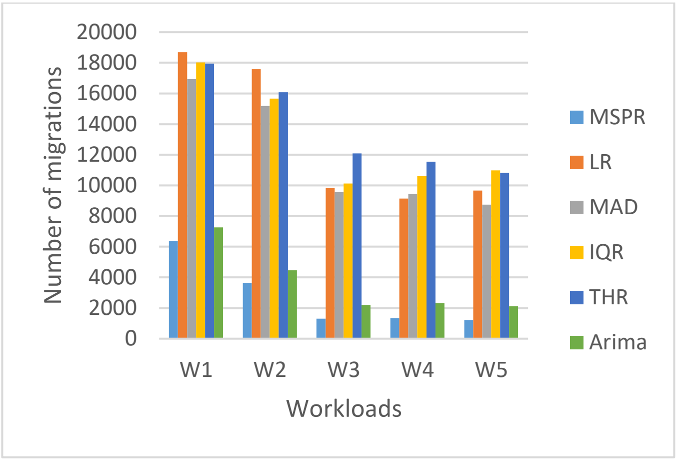

Figure 4. Comparison of number of migrations for 5 workloads- experiment 1.

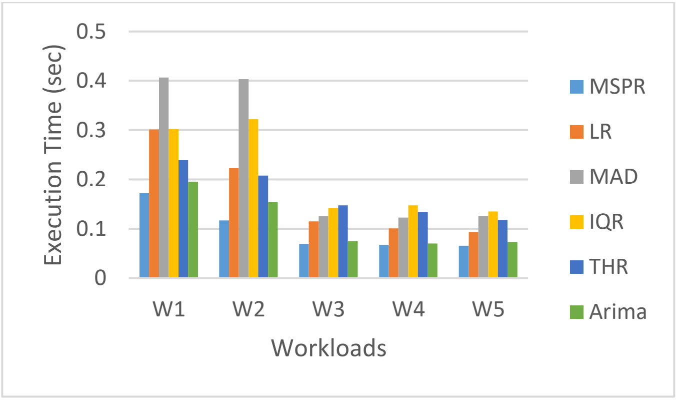

Figure 5. Comparison of execution time for 5 workloads- experiment 1.

2) *Experiment 2*

Figures 6-10 provide a comparison of our consolidation approach with another modified versions of Cloudsim benchmark algorithms. In this experiment, these benchmark algorithms incorporate our proposed underload detection algorithm UD-MSPR explained in subsection IV.B. As a result, the differentiation lies only in the overload detection part, allowing us to focus on comparing the performance in handling overloading and verifying if the benchmark algorithms combined with our predictive underload detection can outperform our approach. It is important to note that the testing results for the MSPR and Arima-based approaches remain the same as in experiment 1.

Figure 6 reveals that MSPR-based approach obtains the best results in terms of energy reduction. It also exhibits competitive performance, between Arima-based and LR-based techniques in the first two workloads (W1 and W2), and between Arima-based and MSPR-based approaches in the last three workloads (W3, W4 and W5). Threshold techniques (THR, IQR, and MAD) consume the highest amount of energy, with THR having the worst results.

Figure 7 highlights the superior performance of MSPR-based technique in minimizing $SLOH$ across most datasets. Figure 8 also indicates that MSPR achieves the lowest $SLOVM$ in all datasets. However, it is worth noting that the average difference in results between our approach and the other techniques is relatively smaller compared to experiment 1. This is primarily because the combination of our underload detection algorithm (UD-MSPR) with other approaches has reduced their $SLOH$ and $SLOVM$ violation rates.

In Figure 9, the number of VM migrations is compared among the different techniques. Our MSPR-based approach consistently performs the lowest number of migrations for resource reallocation. The difference in the average number of migrations is substantial between our approach and the other techniques: **76.486**% compared to LR, **84.113**% compared to MAD, **89.664**% compared to IQR, and **91.111**% compared to THR. It is worth noting that the benchmark algorithms also exhibit a reduction in the number of migrations compared to the results of experiment 1. Arima-based approach also performs significantly fewer migrations than these benchmarks.

Figure 10 reveals that LR and THR have competitive execution time and they outperform MSPR in terms of runtime. MSPR ranks the third. It is important to note that combining our UD-MSPR algorithm with the benchmark approaches results in reduced runtime compared to the results of experiment 1. This combination enables the benchmark algorithms to outperform also Arima-based approach in terms of execution time.

In summary, incorporating our underload detection algorithm into the benchmark approaches leads to improved performance results, significant reduction in SLA violation rates, the number of VM migrations, and runtime. However, our approach achieves the best overall results and outperforms these approaches, in terms of SLA violations, number of migrations, and power consumption.

It is worth mentioning that according to [45], Arima's time complexity is $O(x^2T)$ where x is the number of parameters (i.e. x = p + q + P + Q), and T is the historical data length. This proves that the time complexity of Arima's prediction process is indeed lower than that of MSPR. Nevertheless, MSPR is shown to be more effective in identifying overloaded and underloaded hosts compared to Arima. This accuracy in host state estimation has significant implications for cost savings and overall consolidation system efficiency.

Our complexity analysis in section IV.E reveals that the time complexity of the overall consolidation technique, is $O(N^2)$, where N represents the number of virtual resources (VMs or containers). This indicates that the computational cost of consolidation, irrespective of the prediction method employed, is substantial and grows quadratically with the number of resources. Thus, while Arima may have a lower time complexity for its prediction component, our findings emphasize the importance of accurate host load detection in the overall performance of workload consolidation techniques.

While our analysis demonstrates the effectiveness of the proposed consolidation approach within the contexts of the Materna and Bitbrains datasets, it's important to further investigate the broader generalizability of our findings. VM performance is subject to various influencing factors, such as hardware configurations, network conditions, and workload characteristics, which can vary significantly between organizations and data centers environments. Our results may serve as a foundation for future research aimed at expanding the applicability of our approach to a wider range of real-world contexts.

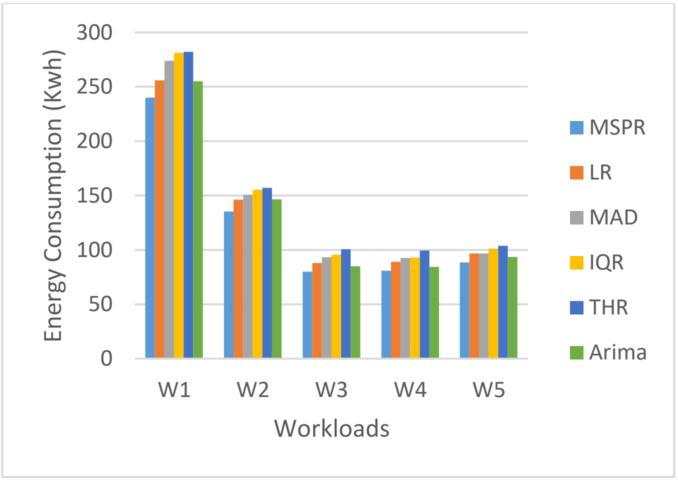

Figure 6. Comparison of energy consumption for 5 workloads-experiment 2.

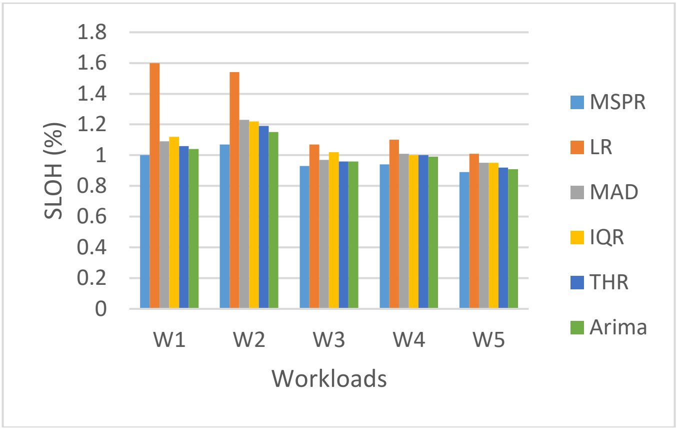

Figure 7. Comparison of SLOH metric for 5 workloads- experiment 2.

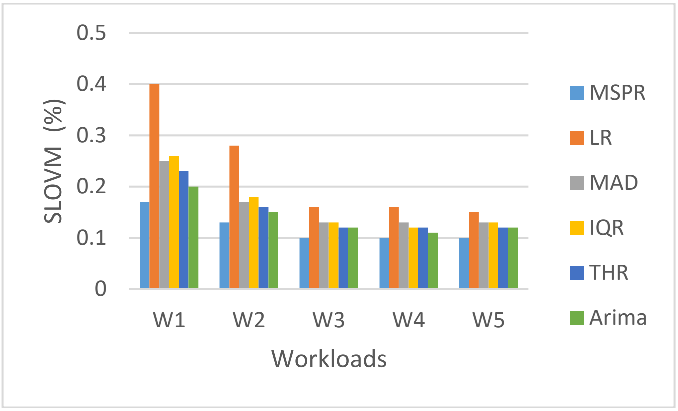

Figure 8. Comparison of SLOVM metric for 5 workloads- experiment 2.

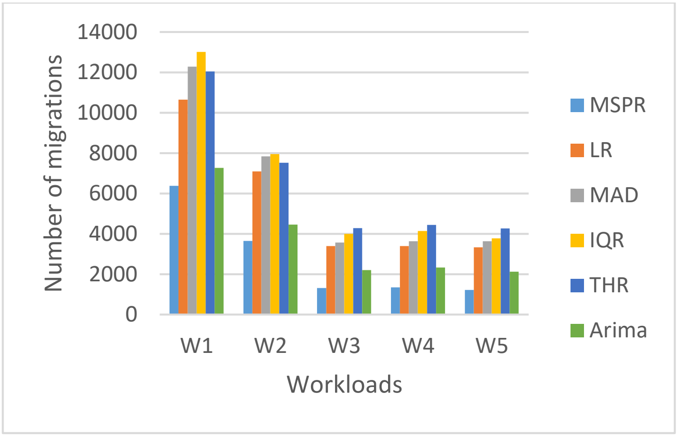

Figure 9. Comparison of number of migrations for 5 workloads- experiment 2.

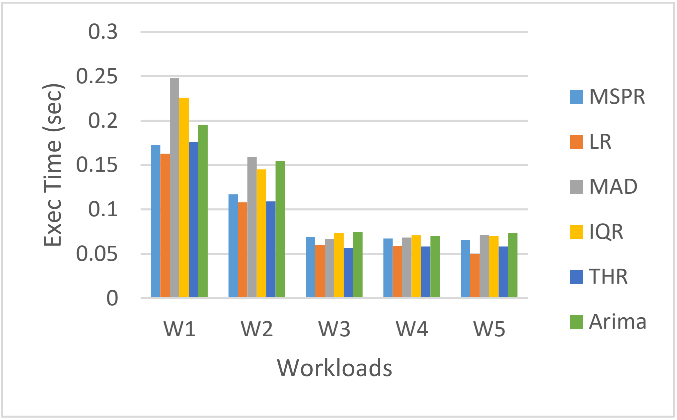

Figure 10. Comparison of execution time for 5 workloads- experiment 2.

VI. CONCLUSION

In this paper, we propose a predictive workload consolidation mechanism that aims to reduce energy consumption, minimize SLA violations, and optimize the resource re-allocation process. Our approach consists of several components. Firstly, we introduce a multi-step-ahead and multi-resource prediction model, called MSPR that combines the Kalman Filter with Support Vector Regression (SVR). This model allows us to forecast the future resource utilization of hosts, including CPU, memory, and bandwidth. By leveraging historical data and utilizing the strengths of both Kalman Filter and SVR, we can accurately predict resource demands. Secondly, we present novel techniques based on MSPR model, for detecting underload and overload states of hosts. These techniques consider the current and predicted resource utilization trend across all resource types (CPU, memory and bandwidth) for each host to proactively estimate its state. To make informed decisions about host states, we calculate an adaptive upper-threshold for overload detection using Median Absolute Deviation (MAD) based on historical data for each resource type. Additionally, we provide the flexibility to specify different underload thresholds and prediction window sizes for each resource type. To pursue testing experiments, we combined our proposed techniques with existing VM selection and VM placement strategies. It is worth noting that our techniques are not limited to VMs and can be combined with other selection and placement methods, such as those designed for containers. Although VM placement is re-used from Cloudsim platform, we updated it to incorporate our overload detection technique for the identification of potential overloading issues on candidate destination hosts after migrating a VM.

To evaluate our proposed techniques, we conducted simulations using real-world workload traces from Bitbrains and Materna. We compared our approach against modified and optimized versions of benchmark algorithms integrated into Cloudsim, including MAD-based, IQR-based, THR-based, and LR-based approaches. These benchmarks are updated to consider multi-resource aspects in their host state estimation. In addition to these approaches, we implemented another predictive consolidation technique by combining our overload and underload techniques with a multi-resource Arima prediction model, as an alternative to MSPR model. This allowed us to compare the performance of our proposal against another predictive consolidation mechanism. Our experimental results demonstrated the effectiveness of the MSPR-based consolidation approach in minimizing defined cost metrics and outperforming the other algorithms. For future work, we aim to combine our proposed algorithms with more advanced selection and placement strategies for VMs or containers. These strategies play a crucial role in resource reallocation decisions and can affect the overall system performance. Future research endeavors can also involve conducting further experiments in varied datacenter environments, and on other datasets from diverse sources and domains. This would allow us to explore the generalizability of our approach and its potential benefits across a broader spectrum of real-world scenarios.

Acknowledgment

This work is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), Grant no. RGPIN-2019-05250.

References

[1] T. Khan, W. Tian, G. Zhou, S. Ilager, M. Gong, and R. Buyya, "Machine learning (ML)-centric resource management in cloud computing: A review and future directions," *J. Netw. Comput. Appl.*, vol. 204, no. March, p. 103405, 2022, doi: 10.1016/j.jnca.2022.103405.

[2] M. Awad, N. Kara, and C. Edstrom, "SLO-aware dynamic self-adaptation of resources," *Futur. Gener. Comput. Syst.*, vol. 133, pp. 266–280, 2022, doi: 10.1016/j.future.2022.03.018.

[3] G. McGrath and P. R. Brenner, "Serverless Computing: Design, Implementation, and Performance," *Proc. - IEEE 37th Int. Conf. Distrib. Comput. Syst. Work. ICDCSW 2017*, pp. 405–410, 2017, doi: 10.1109/ICDCSW.2017.36.

[4] S. S. Panwar, M. M. S. Rauthan, and V. Barthwal, "A systematic review on effective energy utilization management strategies in cloud data centers," *J. Cloud Comput.*, vol. 11, no. 1, 2022, doi: 10.1186/s13677-022-00368-5.

[5] F. Zhang, G. Liu, X. Fu, and R. Yahyapour, "A Survey on Virtual Machine Migration: Challenges, Techniques, and Open Issues," vol. 20, no. 2, pp. 1206–1243, 2018, doi: 10.1109/COMST.2018.2794881.

[6] S. B. Melhem, A. Agarwal, N. Goel, and M. Zaman, "Selection process approaches in live migration: A comparative study," *2017 8th Int. Conf. Inf. Commun. Syst. ICICS 2017*, pp. 23–28, 2017, doi: 10.1109/IACS.2017.7921940.

[7] S. M. Moghaddam, S. F. Piraghaj, M. O'Sullivan, C. Walker, and C. P. Unsworth, "Energy-efficient and SLA-aware virtual machine selection algorithm for dynamic resource allocation in cloud data centers," *Proc. - 11th IEEE/ACM Int. Conf. Util. Cloud Comput. UCC 2018*, pp. 103–113, 2018, doi: 10.1109/UCC.2018.00019.

[8] S. B. Nath, S. K. Addya, S. Chakraborty, and S. K. Ghosh, "Green Containerized Service Consolidation in Cloud," *IEEE Int. Conf. Commun.*, vol. 2020-June, 2020, doi: 10.1109/ICC40277.2020.9149173.

[9] H. Xiao, Z. Hu, and K. Li, "Multi-objective vm consolidation based on thresholds and ant colony system in cloud computing," *IEEE Access*, vol. 7, pp. 53441–53453, 2019, doi: 10.1109/ACCESS.2019.2912722.

[10] 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. Comput. Pract. Exp.*, vol. 24, no. 13, pp. 1397–1420, 2012, doi: 10.1002/cpe.

[11] L. Li, J. Dong, D. Zuo, and J. Wu, "SLA-Aware and Energy-Efficient VM Consolidation in Cloud Data Centers Using Robust Linear Regression Prediction Model," *IEEE Access*, vol. 7, pp. 9490–9500, 2019, doi: 10.1109/ACCESS.2019.2891567.

[12] 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 Distrib. Comput.*, vol. 139, pp. 99–109, 2020, doi: 10.1016/j.jpdc.2019.12.014.

[13] S. Shen, V. Van Beek, and A. Iosup, "Statistical characterization of business-critical workloads hosted in cloud datacenters," *Proc. - 2015 IEEE/ACM 15th Int. Symp. Clust. Cloud, Grid Comput. CCGrid 2015*, pp. 465–474, 2015, doi: 10.1109/CCGrid.2015.60.

[14] A. Kohne, M. Spohr, L. Nagel, and O. Spinczyk, "FederatedCloudSim: A SLA-aware federated cloud simulation framework," *Proc. 2nd Int. Work. Cross-Cloud Syst. CrossCloud Brokers 2014 - Held conjunction with 15th ACM/IFIP/USENIX Int. Middlew. Conf. Middlew. 2014*, 2014, doi: 10.1145/2676662.2676674.

[15] A. Kohne, D. Pasternak, L. Nagel, and O. Spinczyk, "Evaluation of SLA-based decision strategies for VM scheduling in cloud data centers," *3rd Work. CrossCloud Infrastructures Platforms, CrossCloud 2016 - Coloca. with EuroSys 2016*, vol. 1, no. 212, 2016, doi: 10.1145/2904111.2904113.

[16] M. Masdari and A. Khoshnevis, "A survey and classification of the workload forecasting methods in cloud computing," *Cluster Comput.*, vol. 23, no. 4, pp. 2399–2424, 2020, doi: 10.1007/s10586-019-03010-3.

[17] F. Qiu, B. Zhang, and J. Guo, "A deep learning approach for VM workload prediction in the cloud," *2016 IEEE/ACIS 17th Int. Conf. Softw. Eng. Artif. Intell. Netw. Parallel/Distributed Comput. SNPD 2016*, pp. 319–324, 2016, doi: 10.1109/SNPD.2016.7515919.

[18] S. Malik, M. Tahir, M. Sardaraz, and A. Alourani, "A Resource Utilization Prediction Model for Cloud Data Centers Using Evolutionary Algorithms and Machine Learning Techniques," *Appl. Sci.*, vol. 12, no. 4, p. 2160, 2022, doi: 10.3390/app12042160.

[19] Y. Xie *et al.*, "Real-Time Prediction of Docker Container Resource Load Based on a Hybrid Model of ARIMA and Triple Exponential Smoothing," *IEEE Trans. Cloud Comput.*, vol. 10, no. 2, pp. 1386–1401, 2022, doi: 10.1109/TCC.2020.2989631.

[20] T. Khan, W. Tian, S. Ilager, and R. Buyya, "Workload forecasting and energy state estimation in cloud data centres: ML-centric approach," *Futur. Gener. Comput. Syst.*, vol. 128, pp. 320–332, 2022, doi: 10.1016/j.future.2021.10.019.

[21] N. Chaurasia, M. Kumar, R. Chaudhry, and O. P. Verma, "Comprehensive survey on energy-aware server

consolidation techniques in cloud computing," *J. Supercomput.*, vol. 77, no. 10, pp. 11682–11737, 2021, doi: 10.1007/s11227-021-03760-1.

[22] L. Helali and M. N. Omri, "A survey of data center consolidation in cloud computing systems," *Comput. Sci. Rev.*, vol. 39, p. 100366, 2021, doi: 10.1016/j.cosrev.2021.100366.

[23] R. Zolfaghari and A. M. Rahmani, *Virtual Machine Consolidation in Cloud Computing Systems: Challenges and Future Trends*, vol. 115, no. 3. Springer US, 2020. doi: 10.1007/s11277-020-07682-8.

[24] B. Hariharan, R. Siva, S. Kaliraj, and P. N. S. Prakash, "ABSO: an energy-efficient multi-objective VM consolidation using adaptive beetle swarm optimization on cloud environment," *J. Ambient Intell. Humaniz. Comput.*, vol. 14, no. 3, pp. 2185–2197, Mar. 2023, doi: 10.1007/s12652-021-03429-w.

[25] R. Yadav, W. Zhang, K. Li, C. Liu, and A. A. Laghari, "Managing overloaded hosts for energy-efficiency in cloud data centers," *Cluster Comput.*, vol. 24, no. 3, pp. 2001–2015, 2021, doi: 10.1007/s10586-020-03182-3.

[26] N. Songara and M. K. Jain, "MRA-VC: multiple resources aware virtual machine consolidation using particle swarm optimization," *Int. J. Inf. Technol.*, vol. 15, no. 2, pp. 697–710, 2023, doi: 10.1007/s41870-022-01102-9.

[27] N. T. Hieu, M. Di Francesco, and A. Yla-Jaaski, "Virtual Machine Consolidation with Multiple Usage Prediction for Energy-Efficient Cloud Data Centers," *IEEE Trans. Serv. Comput.*, vol. 13, no. 1, pp. 186–199, 2020, doi: 10.1109/TSC.2017.2648791.

[28] D. Minarolli, A. Mazrekaj, and B. Freisleben, "Tackling uncertainty in long-term predictions for host overload and underload detection in cloud computing," *J. Cloud Comput.*, vol. 6, no. 1, 2017, doi: 10.1186/s13677-017-0074-3.

[29] U. Arshad, M. Aleem, G. Srivastava, and J. C. W. Lin, "Utilizing power consumption and SLA violations using dynamic VM consolidation in cloud data centers," *Renew. Sustain. Energy Rev.*, vol. 167, no. July, p. 112782, 2022, doi: 10.1016/j.rser.2022.112782.

[30] M. H. Sayadnavard, A. Toroghi Haghighat, and A. M. Rahmani, "A multi-objective approach for energy-efficient and reliable dynamic VM consolidation in cloud data centers," *Eng. Sci. Technol. an Int. J.*, vol. 26, p. 100995, 2022, doi: 10.1016/j.jestch.2021.04.014.

[31] S. Banerjee, S. Roy, and S. Khatua, "Efficient resource utilization using multi-step-ahead workload prediction technique in cloud," *J. Supercomput.*, vol. 77, no. 9, pp. 10636–10663, 2021, doi: 10.1007/s11227-021-03701-y.

[32] F. Farahnakian, T. Pahikkala, P. Liljeberg, J. Plosila, N. T. Hieu, and H. Tenhunen, "Energy-aware VM consolidation in cloud data centers using utilization prediction model," *IEEE Trans. Cloud Comput.*, vol. 7, no. 2, pp. 524–536, Apr. 2019, doi: 10.1109/TCC.2016.2617374.

[33] M. Awad, N. Kara, and A. Leivadeas, "Utilization prediction-based VM consolidation approach," *J. Parallel Distrib. Comput.*, vol. 170, pp. 24–38, 2022, doi: 10.1016/j.jpdc.2022.08.001.

[34] K. Park and V. S. Pai, "CoMon: A Mostly-Scalable Monitoring System for PlanetLab," *ACM SIGOPS Oper. Syst. Rev.*, vol. 40, no. 1, pp. 65–74, 2006, doi: 10.1145/1113361.1113374.

[35] 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, 2020, doi: 10.1109/ACCESS.2020.2984056.

[36] E. Kalyvianaki, T. Charalambous, and S. Hand, "Adaptive resource provisioning for virtualized servers using kalman filters," *ACM Trans. Auton. Adapt. Syst.*, vol. 9, no. 2, 2014, doi: 10.1145/2626290.

[37] X. Zheng and A. Leivadeas, "Network Assurance in Intent-Based Networking Data Centers with Machine Learning Techniques," *Proc. 2021 17th Int. Conf. Netw. Serv. Manag. Smart Manag. Futur. Networks Serv. CNSM 2021*, pp. 14–20, 2021, doi: 10.23919/CNSM52442.2021.9615580.

[38] A. Valade, P. Acco, P. Grabolosa, and J. Y. Fourniols, "A study about kalman filters applied to embedded sensors," *Sensors (Switzerland)*, vol. 17, no. 12, pp. 1–18, 2017, doi: 10.3390/s17122810.

[39] A. Abdiansah and R. Wardoyo, "Time Complexity Analysis of Support Vector Machines (SVM) in LibSVM," *Int. J. Comput. Appl.*, vol. 128, no. 3, pp. 28–34, 2015, doi: 10.5120/ijca2015906480.

[40] "The SPECpower Benchmark," http://www.spec.org/power_ssj2008/.

[41] W. Lin, S. Xu, L. He, and J. Li, "Multi-resource scheduling and power simulation for cloud computing," *Inf. Sci. (Ny).*, vol. 397–398, pp. 168–186, 2017, doi: 10.1016/j.ins.2017.02.054.

[42] R. Sohan, A. Rice, A. W. Moore, and K. Mansley, "Characterizing 10 Gbps Network Interface Energy Consumption Abstract—This paper quantifies the energy consumption in six 10 Gbps and four 1 Gbps interconnects at a fine-grained level, introducing two metrics for calculating the energy efficiency of a netw," *IEEE Local Comput. Netw. Conf.*, pp. 268–271, 2010, [Online]. Available: https://www.cl.cam.ac.uk/~acr31/pubs/sohan-10gbpower.pdf

[43] R. N. Calheiros, R. Ranjan, A. Beloglazov, R. Buyya, and C. A. F. De Rose, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw. - Pract. Exp.*, vol. 41, no. 1, pp. 23–50, 2011, doi: 10.1002/spe.

[44] C. C. Chang and C. J. Lin, "LIBSVM: A Library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 1–39, 2011, doi: 10.1145/1961189.1961199.

[45] X. Wang, Y. Kang, R. J. Hyndman, and F. Li, "Distributed ARIMA models for ultra-long time series," *Int. J. Forecast.*, vol. 39, no. 3, pp. 1163–1184, 2023, doi: 10.1016/j.ijforecast.2022.05.001.