The Journal of Supercomputing (2022) 78:5806–5855
https://doi.org/10.1007/s11227-021-04107-6

Deep learning-based multivariate resource utilization prediction for hotspots and coldspots mitigation in green cloud data centers

Yashwant Singh Patel[1] · Rishabh Jaiswal[2] · Rajiv Misra[3]

Accepted: 20 September 2021 / Published online: 7 October 2021

Abstract
Dynamic virtual machine (VM) consolidation is a constructive technique to enhance resource usage and is extensively employed to minimize data centers' energy consumption. However, in the current approaches, consolidation techniques are heavily relied on reducing the actively used physical servers (PMs) based on their current resource utilization without considering future resource demands. Also, many of the reported works for cloud workload prediction applied univariate time series-based forecasting models and neglected the dependency of other resource utilization metrics. Thus, resulting in inaccurate predictions, unnecessary migrations, high migration costs, and increased service level agreement violations (SLAVs) may nullify the consolidation benefits. To efficiently address this issue, we propose a multivariate resource usage prediction-based hotspots and coldspots mitigation approach that considers both the current and future usage of resources with $O(sk)$ time complexity, where s and k denote the number of PMs and VMs, respectively. The proposed technique uses a clustering-based stacked bidirectional (Long Short-Term Memory) LSTM deep learning network to predict the future memory and CPU usage of PMs and VMs with high accuracy and $O((Q(Q+W)*\Theta)$ computational complexity, where Q, W, and Θ represent the number of hidden layer cells, outputs, and training epochs, respectively. Through extensive simulations based on Google's cluster workload traces, we demonstrate that our proposed method obtains substantial improvements in terms of prediction performance, energy-efficiency, actively used PMs, VM migrations, and SLA violations over the benchmark approaches.

Keywords Green cloud computing · Deep learning · Dynamic VM consolidation · Energy-efficiency · SLA · Migration · Resources' utilization · Prediction approaches · Performance evaluation

✉ Yashwant Singh Patel
yashwant.patel@thapar.edu

Extended author information available on the last page of the article

1 Introduction

Cloud computing is rapidly emerging and evolving as a successful computing paradigm for providing on-demand and dynamic provisioning of IT infrastructure, services, and resources in a pay-per-use manner. The cloud provides not only readily available infrastructure but also the ability of rapid auto-scaling to support the massive and fluctuating big data workloads. In the cloud, the data centers are connected via networks and offer flexible and cost-effective services for individuals, organizations, and enterprises. To satisfy the diverse demands from different users, the commercial cloud Infrastructure-as-a-Service or IaaS providers, such as Amazon EC2, IBM, and Microsoft Azure, offer different VM instances with heterogeneous resources (or dimensions) (e.g., memory, CPU, storage and network bandwidth) based on pay-per-use business model [1]. In this regard, the virtualization technology such as VMware [2] and Xen [3] plays a vital role for efficient resource management in a multi-user environment. With a hypervisor or virtual machine monitor (VMM), it permits several VMs to share a single PM's resources. By adopting such virtualized infrastructure technology, several public cloud service providers (CSPs) such as Microsoft, Amazon, Yahoo, and Google build massive data centers geographically distributed all over the globe to offer cloud services with low cost, global coverage, low latency, and high application availability.

However, cloud infrastructures' rising demands are drastically increasing the global data center energy consumption [4]. As per the recent reports on world's servers power consumption, the electricity consumption of servers worldwide accounts for 3% of the global electricity production [5]. Such high data center energy consumption not only leads to considerable operation expenses, but also generates substantial carbon dioxide (CO_2) emissions. Gartner estimated that the information and communication technology (ICT) industry ecosystem accounts for greater than 2% of global CO_2 emissions, which is equal to the aviation industry's emissions from fuel [6, 7]. In the context of ICT's future energy demand, analysts also forecasted that the data center electricity usage would increase approximately 15 times by 2030, which may result in around 8% of the projected global demand of electricity [8]. Therefore, even though several phenomenal developments on cloud infrastructures are made, designing effective energy-aware resource provisioning strategies to enhance energy-efficiency of data centers have become essential, as depicted in Fig. 1. Additionally, supporting a good quality of service (QoS) level is necessary for CSPs to satisfy users' expectations concerning performance. Service level agreement (SLA) is the concrete implementation to define QoS requirements, which are contracts to describe service level details offered to consumers, such as downtime ratio, system throughput, and response time. Therefore, minimizing the overall energy consumption of data centers, while restricting the SLA violations (SLAVs), is the principle objective of this study.

The research on cloud data centers shows that the physical machines run at 10 to 50% of their maximum CPU usage [9]. Additionally, the majority of PMs are

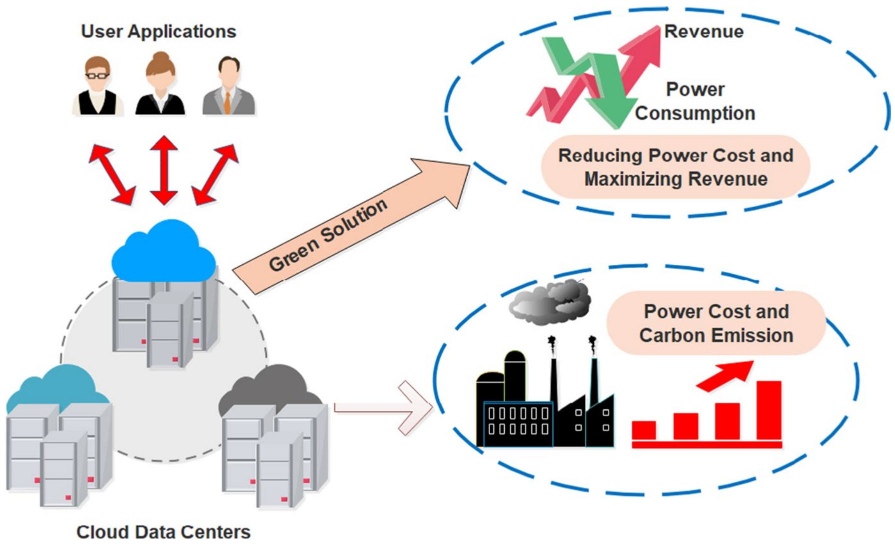


Fig. 1 Green Cloud Computing

idle PMs, which consume about 70% of their peak power [10]. To reduce energy waste and improve resource utilization, the CSPs apply dynamic VM consolidation approaches. By leveraging hardware virtualization technique [3], the VM instances are dynamically consolidated and packed in fewer PMs while considering their current resource demands. In addition, by using live migration technology [11], the virtualization enables the CSPs to dynamically allocate VMs in least active PMs and allows switching the idle PMs into sleep mode. Migration is transparent and beneficial when a physical server is highly overloaded (i.e., creating hotspot) or underloaded (i.e., creating coldspot) [12, 13]. However, consolidation policies reduce energy consumption significantly but live VM migration results in increased SLAVs. Consequently, an effective VM consolidation approach is crucial for reducing the energy expenses while meeting SLAs. In a cloud data center comprising thousands of heterogeneous PMs, the consolidation process comprises three phases [12]:

1. *Hotspots and coldspots detection* Determining the PM is become overloaded (i.e., creating hotspots) or underloaded (i.e., creating coldspots);
2. *VM selection* What VMs can be chosen for migration from the hotspots and coldspots while achieving the objectives such as reducing migration cost or total number of migrations;
3. *Destination PM selection* What are the target PMs for VMs that should be migrated to mitigate the hotspots and coldspots.

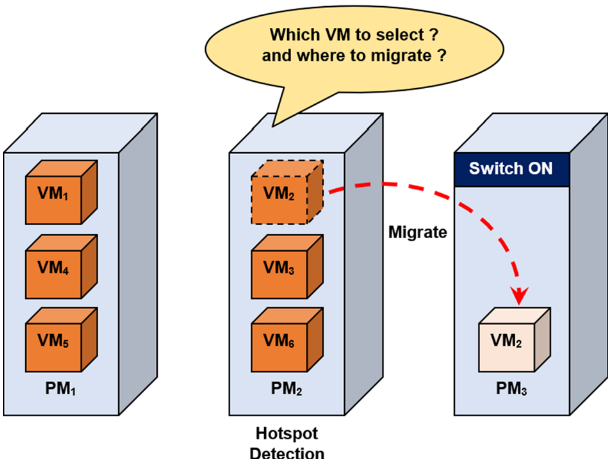


Fig. 2 Hotspot mitigation

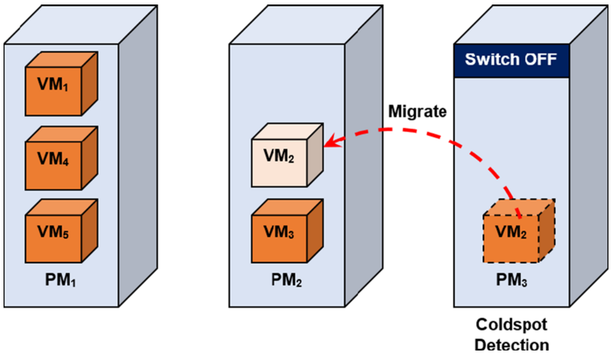


Fig. 3 Coldspot Mitigation

1.1 Example

Consider the examples represented in Figs. 2 and 3 to illustrate the basic operations of dynamic VM consolidation, including hotspot mitigation and coldspot mitigation, respectively. Figure 2 demonstrates a hotspot mitigation example with three PMs $< PM_1, PM_2, PM_3 >$ and six VMs $< VM_1, \ldots, VM_6 >$ are placed on them. Assume the hotspot is detected on PM_2 due to resource over-utilization. In this case, a few of the VM instances should be migrated to some other suitable PMs. Let's say, VM_2 is selected and migrated from PM_2 to PM_3.

In Fig. 3, we have presented an example of coldspot mitigation, where five VMs, i.e.,VM_1, VM_2, VM_3, VM_4, and VM_5 are hosted on three PMs. The coldspot mitigation process identifies the underloaded machines and tries to migrate all VMs from that PM to any other suitable PM without creating any other hotspots. In the given example, the consolidation process determines PM_3 as a least loaded PM and migrate VM_2 from PM_3 to PM_2. With the intent that the PM_3 could be switched to power-saving mode to maximize the energy-efficiency.

1.2 Motivation

In the context of the VM consolidation, many of the reported works have addressed the VM allocation and migration problem as a multi-dimensional bin-packing problem, known to be an NP-hard problem [14]. Similar to bin-packing, the physical servers would be considered as bins and VMs mapped with items. To apply in the VM placement and migration scenario, the classical bin-packing heuristics have to be modified in three aspects: first, unlike the bin-packing, the PMs are treated as bins with different sizes. Second, the PMs and VMs are characterized by varying dimensions of resource so that optimal balancing can be performed with respect to resource usage across several dimensions. Third, the objective of classical bin-packing algorithms is only to reduce the total number of bins. Still, in the case of a consolidation, there can be multi-objective such as minimizing SLA violations, actively used PMs and migrations. Thus, to find near-optimal solutions, it requires optimization algorithms. Also, for effective energy-efficient provisioning of resources in green data centers, several schemes have been provided. However, from the reported studies, we have found a few aspects of improvements:

1. Many of the existing solutions only consider the current resource utilization for hotspots and coldspots mitigation.
2. Reported works on resource utilization prediction in the cloud only consider the univariate time series-based classical methods. Such methods apply only the past trends of the target resource usage metric, i.e., CPU utilization, only to generate future predictions.
3. Assignment of VMs to PMs to reduce energy cost as much as possible without considering the SLA violations.
4. Live VM migration is necessary as the VMs arrive and depart periodically in highly dynamic data centers, but it results in high migration cost (e.g., network cost).
5. Migration decision to mitigate the hotspots and coldspots, when there are not sufficient resources on least loaded PMs.

1.3 Contributions

In this paper, our approach is to predict future utilization of resources on the basis of past workload traces of PMs and VMs. Current and predicted future utilization metrics are applied for efficient hotspots and coldspots mitigation to improve the data centers' energy-efficiency while preserving the QoS guarantee. The technical contributions of this work are highlighted as follows:

1. Demonstrate the advantages of using different feature selection methods to determine relevant features subset to predict future resource utilization.
2. Design of multivariate resource utilization prediction approach based on the clustering-based stacked bidirectional LSTM deep learning model to forecast future resource usage as per the past trace workload on the considered hosts.
3. Effective dynamic VM consolidation algorithms using current and future resource utilization is presented, namely prediction-based hotspots detection, prediction-based coldspots detection, energy and QoS based VM selection, and destination PMs selection for migrated VMs.
4. Through real workload traces, we investigate the relationship in respect of prediction accuracy, migration cost, energy consumption, actively used PMs, and SLA violations with resource utilization thresholds.

1.4 Article organization

The remaining of the current paper is structured as follows: Sect. 2 provides the related studies on the dynamic VM consolidation and prediction approaches for cloud data centers. The problem definition is presented in Sect. 3. The system model and performance models are provided in Sect. 4. In Sect. 5, we present different feature selection techniques, proposed clustering-based stacked bidirectional LSTM prediction model, the VM consolidation algorithms for hotspots and coldspots mitigation, including hotspots detection, coldspots detection, VM selection, and destination PMs selection, and complexity analysis. Simulation experiments with real-world traces are demonstrated in Sect. 6. At last, Sect. 7 concludes the work with some of the future directions.

2 Related works

We study the problem of VM consolidation in data centers. Several studies comprising VM consolidation and resource prediction in the cloud are discussed to address this problem.

2.1 VM consolidation

In recent years, there have been many methodologies reported in the literature on VM consolidation techniques under different settings [4, 15, 16]. In [13] Timothy wood et al. presented the sandpiper system that combines the product of three dimensions, i.e., CPU, network, and memory utilization, into a single dimension volume metric. Sandpiper has implemented gray-box and black-box approach (BG) to automate the cloud system for continuous monitoring of hotspots. To capture the degree of overload, it uses the volume metric to sort all the overloaded servers and apply the volume-to-size ratio (VSR) for VMs in each server. The BG strategy then selects the highest volume server first and the highest VSR VM for migration. To allocate these VMs, it sorts the destination PMs based on their increasing volume metric. The major limitation of sandpiper is that it does not provide good performance while characterizing underloaded and overloaded servers. It is also limited to homogeneous servers and does not enable the coldspot mitigation.

Beloglazov et al. [12] proposed thresholds (i.e., lower and upper) based on adaptive heuristics. For host overloading detection, they presented MAD: median absolute deviation, IQR: interquartile range, LR: local regression, and robust LR-based approaches that can be applied via the statistical investigation of the historical resource usage data. They also designed three policies for selecting VMs: maximum correlation, random selection, and MMT: minimum migration time. Furthermore, to identify the destination host, they have implemented a Power-aware Best Fit Decreasing (PBFD) technique, combining Best Fit Decreasing (BFD) bin-packing scheme and a power-aware model. They implemented these policies on CloudSim [17] with PlanetLab workload traces. Using experimental results, they showed that the LR-based method along with the MMT policy of selecting VMs performs better than existing approaches in regard to the number of migrations, energy consumption, SLAVs, and ESV metric (where ESV = SLAV $\times$ energy consumption).

Farahnakian et al. [18] provided LR-based method to predict CPU utilization of PMs and simulated it using CloudSim. In [19], researchers have applied the K-nearest neighbor regression approach to demonstrate that it could further enhance the SLAV and energy consumption more than the linear regression method. In [20], the authors have presented a metaheuristic algorithm named Ant Colony System (ACS) for VM consolidation. Through experimental results, they showed that it could improve the overall system performance in terms of VM migrations, energy consumption, and QoS requirements. F. Farahnakian et al. [21] designed a utilization prediction-based dynamic VM consolidation method. It applies a regression-based strategy to predict future usage of resources. Their approach uses both the current and future usage of resources to estimate the next time step memory and CPU usage of PMs and VMs. Through simulation results, they demonstrated that the consolidation process could improve the performance of SLA violations, migrations, and energy consumption.

Moghaddam et al. [22] proposed machine learning (ML)-based resource prediction models. They have tuned a forecasting framework for each VM independently, which uses the VM's past behavior for predicting their CPU utilization. Haghshenas et al. [23] designed a regression-based technique to predict the resource utilization of PMs and VMs. They have applied a linear regression-based algorithm for destination host selection during the migration of VMs. Furthermore, they proposed an improved version of the approach to predict the host usage for underutilized host selection. Hsieh et al. [24] suggested a VM consolidation technique for the host overload and underload detection. They have used a Gray-Markov-based model to estimate future resource utilization. Tarafdar et al. [25] applied a QoS and energy-aware consolidation technique. They have applied a Markov chain-based model to detect the underloaded and overloaded PMs. Additionally, they applied a VM selection and placement scheme using the linear weighted sum technique.

Chen et al. [26] proposed an Exponentially Weighted Moving Average or EWMA scheme to identify host overloading. They further evaluated their approach using local regression and identified that when the constant value is 0.025, the EWMA performs better with respect to SLAVs, energy-efficiency, and ESV. Donnell et al. [27] provided Gossip Contracts (GC)-based dynamic virtual machine consolidation approach for a decentralized solution. Their approach performs well in respect of power consumption and SLA Violations. Naeen [28] applied Markov chain models for VM consolidation with QoS constraints.

Li et al. [29] designed a quality and energy-aware VM consolidation (EQ-VMC) strategy to improve the service quality and energy-efficiency. They have developed a discrete differential evolution heuristic to explore the global optimum solution for VM assignment. El-Moursy et al. [30] designed a multi-dimension-based regression algorithms to identify overloaded PMs, which integrate memory, CPU, network BW usage. Ranjbari et al. [31] suggested a technique of learning automata to improve resource usage while reducing energy consumption.

By reviewing the previous works, we have observed that most of the current solutions apply a single prediction model for resource utilization prediction for all the VMs or PMs. In addition, most of the studies have meticulously focused on CPU utilization and neglected the computational complexity and optimization of trade-offs between SLA violation, migrations and energy consumption. Most of the reviewed studies have examined future resource utilization of PMs and migration time independently. While some of the current solutions only examine the current utilization of resources and address the single objective of energy cost minimization and may not be feasible to work in any arbitrary stage of the cloud data centers such as detecting and mitigating multiple hotspots and coldspots at any time instant. Therefore, in the next subsection, we explore the prominent prediction models applied for cloud resource usage prediction.

2.2 Prediction models for cloud workloads

In the cloud, the problem of resource utilization prediction can be framed as a time series forecasting model, where we can analyze the past data of cloud resource utilization to estimate future values. Several statistical models, such as Holt-winter [32], ARIMA (Autoregressive integrated moving average) [33], seasonal ARIMA (SARIMA) [34], Feed-forward neural network [35], and Markov models [36, 37], are proposed for utilization prediction. In contrast, Zhang et al. [33] applied the ARIMA method to estimate the future cloud resource utilization. This model considers the last t observations of past data RH_t to predict the resource usage RH_{t+1} at time $t+1$. It can be expressed as:

$$\mathrm{RH}_{t+1} = \rho_0\mathrm{RH}_t + .. + \rho_{m-1}\mathrm{RH}_{t-n+1} + \epsilon_{t+1} + \lambda_0\epsilon_t + \cdots + \lambda_{k-1}\epsilon_{t+1-k}, \tag{1}$$

where the notation $\rho + i$ and λ_j represent the constants. ϵ_t is the independent error term. The other parameters n and k show the number of lags or previously measured resource usage values and error terms correspondingly. ARIMA is one of the well-known time series prediction models. However, it is failed to detect the presence of nonlinear patterns present in the time series data. Caglar et al. [35] presented an iOverbook model, which applies a two-layer feed-forward artificial neural network

(ANN). It also generalizes the nonlinear and linear correlation between the input and output using a single hidden layer. It can be defined as:

$$\Delta\hat{z}^t = \sum_{q=1}^{n} \alpha_q \Upsilon\left(\sum_{p=1}^{m} \theta_{pq} \Delta z^{t-p}\right) \tag{2}$$

where $\Delta\hat{z}^t$ defines the estimated resource utilization value at time t. The term m denotes total past lags, and n shows the total neurons present in the hidden layer of network. The training weight parameters are denoted by α and θ. The term $\Upsilon(.)$ shows the activation function. However, most of the abovementioned studies have presumed that observations of resource usage in a long-time period are unassociated with each other. In addition, these techniques have also presumed that the time series data is memoryless and stationary. But Ghorbani et al. [38] identified the existence of long-range dependence (LRD) in the Google's cluster workload trace [39]. The LRD may occur during the time series analysis, where the past-time lags may affect the next step value. The LRD may exist if the dependence decays more gradually than any decaying exponential and mostly similar to a power-like decay. Intuitively, the LSTM (Long short-term memory) models are more suitable to handle this issue, as it can model the LRD in time series.

Song et al. [40] adopted univariate LSTM networks, where they have analyzed the past CPU resource utilization to forecast the future CPU usage trends. LSTMs are a robust kind of recurrent neural networks (RNNs) and work well on sequence-based tasks having long-term dependencies. The basic difference between the component of RNN architecture and standard LSTM architecture is the hidden layer. The LSTM's hidden layer is also termed as LSTM cell. The basic LSTM block is shown in Fig. 4. It comprises a memory cell C_t, an input gate i_t, an output gate o_t, and also a forget gate f_t. The weight matrices are W_{xc}, W_{hc}, W_{xi}, W_{hi}, W_{xo}, W_{ho}, W_{xf}, and W_{hf}. The activation functions are Sigmoid and *tanh* denoted by σ and *tanh*, respectively.

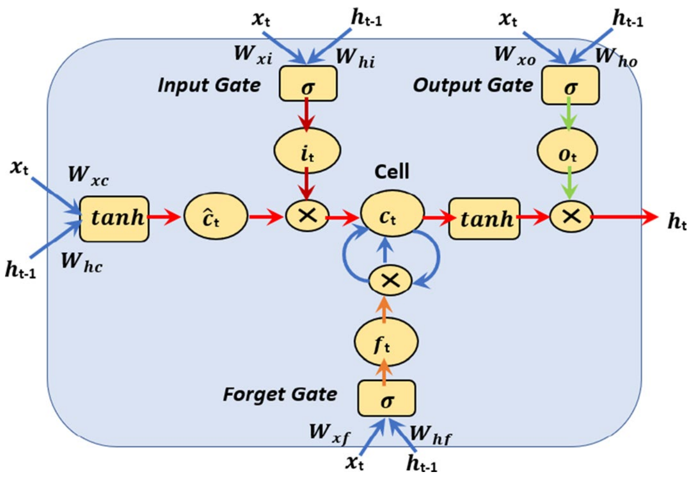


Fig. 4 LSTM memory block

Moreover, the x denotes the point-wise multiplication. At time instant t, x_t indicates the input and h_t shows the hidden state. The memory cell's candidate state is denoted by $\hat{C}_t$. It decides how much input information is received by the cell state. The following expresses the formulation for each gate including input candidate, hidden states, and cell state:

$$f_t = \sigma(x_t * W_{xf} + b_f + h_{t-1} * W_{hf}) \tag{3}$$

$$C_t = C_{t-1} * f_t + \hat{C}_t * i_t \tag{4}$$

$$\hat{C}_t = tanh(b_c + x_t * W_{xc} + h_{t-1} * W_{hc}) \tag{5}$$

$$i_t = \sigma(b_i + x_t * W_{xi} + h_{t-1} * W_{hi}) \tag{6}$$

$$o_t = \sigma(b_o + x_t * W_{xo} + h_{t-1} * W_{ho}) \tag{7}$$

$$h_t = o_t * tanh(C_t) \tag{8}$$

It is observed that most of the workload prediction networks only analyze the past resource utilization trends of the required resource usage metric to predict subsequent resource usage. But the other resource usage metrics are inter-related to each other, and it may impact the overall performance of desired resource metrics. Also, in [41, 42], it is shown that the multivariate approaches deliver high performance and promising results than the univariate approaches. Hence, it is inevitable to analyze the multivariate time series prediction to obtain correct behavior and correlation between the resource usage metrics. Several works have used feature selection methods to recognize the set of relevant features. Such as Dannecke et al. [43] applied the correlation-based feature selection method, where they have used the Pearson correlation technique to distinguish the relevant features set for energy demand prediction. Sun et al. [44] proposed a Granger causality-based model for multivariate time series investigation of Parkinson's telemonitoring and monitoring of water quality. In this work, we have used different features such as CPU usage, memory assigned, utilized memory, total page cache usage, unmapped page cache, utilization of maximum memory observed, disk I/O time, disk capacity space utilization, usage of maximum CPU (over an interval), maximum disk I/O time (over an interval), memory accesses per instruction, and cycles Per Instruction (CPI). By analyzing the diverse set of features, we can identify the correlation between different features.

Most of the studies in the cloud are limited to the unidirectional LSTM (i.e., forward dependencies) to predict future events, which only preserves the information of past resource usage. In [45], Zhao et al. proposed a bidirectional time series model that can acquire the long-range context in both dependencies (i.e., forward and backward) using the extreme learning machines approach. They have observed that the prediction accuracy is highly improved in the bidirectional approach. For the forecasting of laser data, Wakuya et al. [46] applied the bi-directional time series computation and concluded that the prediction model's quality is much better than the unidirectional model. Cui et al. [47] proposed a deep stacked bidirectional and unidirectional LSTM (SBU-LSTM) neural network, which applies both forward and backward dependencies of time series data, to predict the network wide traffic speed. Furthermore, Gupta et al. [48, 49] studied the multivariate LSTM and BLSTM models for cloud workload prediction using Google cluster trace. However, the application of stacked bidirectional LSTM (BiLSTM) deep learners has not been reported for the interpretation of cloud workload patterns. To the best of authors' knowledge, this study is a first attempt to use a clustering-based stacked BiLSTM based multivariate resource utilization prediction model for hotspots and coldspots mitigation in the cloud. Tables 1 and 2 highlight the key differences of existing techniques against our proposed approach.

3 Problem definition

To achieve maximum expected benefit in a virtualized data center, an efficacious consolidation strategy optimizes the VMs assignment for reducing the actively used PMs. This gain solely depends on two primary factors: minimize the SLA violations and migrations of VMs. We can realize these advantages in advance by placing the VMs to PMs on the basis of their future resource usages. Thus, to understand the limitation of the classical VM consolidation approach without predicting next time step resource utilization, let's consider two examples represented in Figs. 5 and 6, respectively. Assume that there are two PMs and three VMs are placed on PMs. Through Fig. 5a, we can observe the CPU utilization of PMs namely PM_1 and PM_2 as 55% and 40% separately. As PM_1 has sufficient resources to place VM_3, a classical consolidation approach will migrate VM_3 from PM_2 to PM_1 at time instant t and switch PM_2 to the low-power state or sleep state. At the time instant $t + 1$ in Fig. 5b, when the CPU utilization demand of VM_1 is increased from 30 to 38%. The PM_1 is become overloaded due to the lack of adequate resources, and some SLA violations occur. Thus, in Fig. 5c, VM_3 is migrated from PM_1 to PM_2 to eliminate the SLA violations further. In this scenario, a robust consolidation approach can avoid useless migrations and minimize SLA violations' rate if it could predict the future resource demand of any VM prior to migration.

Another scenario is depicted in Fig. 6. In this case, at time instant t in Fig. 6a, VM_3 is migrated from PM_2 to PM_1 since it has ample resource to fulfill the demand of VM_3. In Fig. 6b, PM_2 is switched to the low-power state or sleep mode. We noticed that the CPU usage of VM_3 is further increased from 40 to 50%, and a hotspot is created. To mitigate the hotspot and avoid the SLA violations further, VM_3 is migrated to PM_2 at time instant $t + 2$ as represented in Fig. 6c.

Through both the illustrated examples, we observe that the unnecessary migrations and potential SLA violations could be eliminated if a consolidation process considers the future resource usage of PMs and VMs during the assignment of VMs. In this article, the problem of dynamic VM consolidation aims towards robust

Table 1 Comprehensive review of existing VM consolidation approaches

Year & Work	Utilized technique	Performance metrics				Type of prediction		Deep learning	Type of mitigation		Workloads
		SLA	Energy	Migration	Prediction error	Univariate	Multivariate		Hotspots	Coldspots	
2020 [22]	Machine learning, best fit decreasing	✓	✓	✓	✓	✓	×	×	✓	✓	PlanetLab
2020 [23]	Regression-based	✓	✓	×	✓	✓	×	×	✓	✓	PlanetLab
2020 [24]	Gray-Markov-based	✓	✓	✓	×	✓	×	×	✓	✓	PlanetLab
2020 [25]	Markov chain-based	✓	✓	✓	×	✓	×	×	✓	✓	PlanetLab
2020 [27]	Gossip and contract net protocols	✓	✓	✓	×	×	×	×	✓	✓	Gossip and Contract Net protocols
2020 [28]	Markov chain models	✓	✓	✓	✓	✓	×	×	✓	✓	PlanetLab
2020 [29]	Discrete differential evolution algorithm	✓	✓	✓	×	×	×	×	✓	✓	Bitbrains trace, Google-ClusterTrace, Alibaba cluster data
2019 [30]	Regression-based	✓	✓	✓	×	✓	×	×	✓	×	PlanetLab, Gaia cluster data

Table 2 (Continued.) Comprehensive review of existing VM consolidation approaches

Work	Utilized Technique	Performance metrics				Type of prediction		Deep Learning	Type of mitigation		Workloads
		SLA	Energy	Migration	Prediction Error	Univariate	Multivariate		Hotspots	Coldspots	
2019 [21]	Regression-based	✓	✓	✓	✓	✓	×	×	✓	✓	PlanetLab, Google cluster
2018 [31]	Learning Automata	✓	✓	✓	×	✓	×	×	✓	✓	PlanetLab
2018 [26]	Exponentially weighted moving average	✓	✓	✓	×	✓	×	×	✓	×	PlanetLab
2015 [20]	Ant colony optimization	✓	✓	✓	×	✓	×	×	✓	✓	PlanetLab
2013 [18]	Linear regression	✓	✓	×	×	✓	×	×	✓	✓	PlanetLab
2013 [19]	k-nearest neighbor regression	✓	✓	×	×	✓	×	×	✓	✓	PlanetLab
2012 [12]	Best fit decreasing, regression	✓	✓	✓	×	✓	×	×	✓	✓	PlanetLab
2007 [13]	Black-box & Gray-box	✓	×	✓	×	✓	×	×	✓	×	SPECjbb 2005
Proposed	Stacked BiLSTM	✓	✓	✓	✓	✓	✓	✓	✓	✓	Google cluster-usage traces

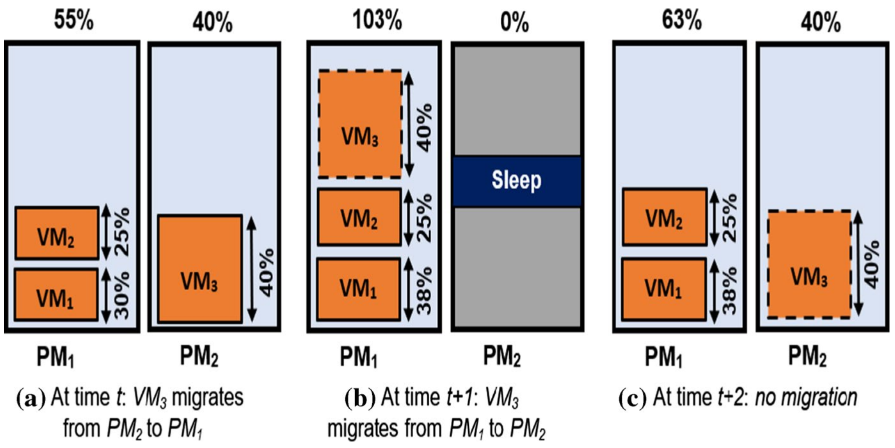

55%
40%
103%
0%
63%
40%
VM₂ 25%
VM₁ 30%
VM₃ 40%
VM₃ 40%
VM₂ 25%
VM₁ 38%
Sleep
VM₂ 25%
VM₁ 38%
VM₃ 40%
PM₁
PM₂
(a) At time t: VM₃ migrates from PM₂ to PM₁
(b) At time t+1: VM₃ migrates from PM₁ to PM₂
(c) At time t+2: no migration

Fig. 5 Mitigation example 1

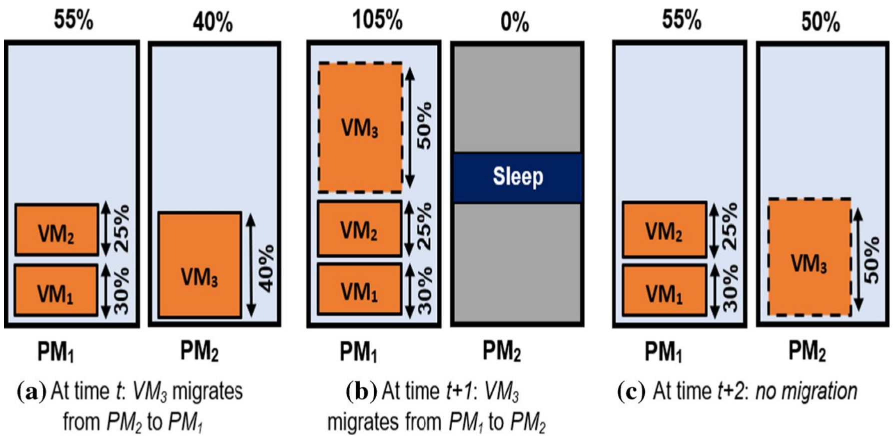

55%
40%
105%
0%
55%
50%
VM₂ 25%
VM₁ 30%
VM₃ 40%
VM₃ 50%
VM₂ 25%
VM₁ 30%
Sleep
VM₂ 25%
VM₁ 30%
VM₃ 50%
PM₁
PM₂
(a) At time t: VM₃ migrates from PM₂ to PM₁
(b) At time t+1: VM₃ migrates from PM₁ to PM₂
(c) At time t+2: no migration

Fig. 6 Mitigation example 2

decision-making during the detection of hotspots (when the PMs are overloaded) and coldspots (when the PMs are underloaded). Once the detection of hotspots and coldspots is performed, all potential VMs are migrated from such PMs to realize the goals of maximum QoS level and reducing energy consumption, respectively.

4 System model

We consider an IaaS cloud data center environment, where its users initiate VM requests as represented in Fig. 7. The data center contains a pool of m heterogeneous PMs denoted by $L_{PM} = \{\mathrm{pm}_1, \mathrm{pm}_2, \mathrm{pm}_3, \dots, \mathrm{pm}_m\}$. Each *PM* owns d types of resources namely memory, CPU, network bandwidth, and disk storage, etc., for

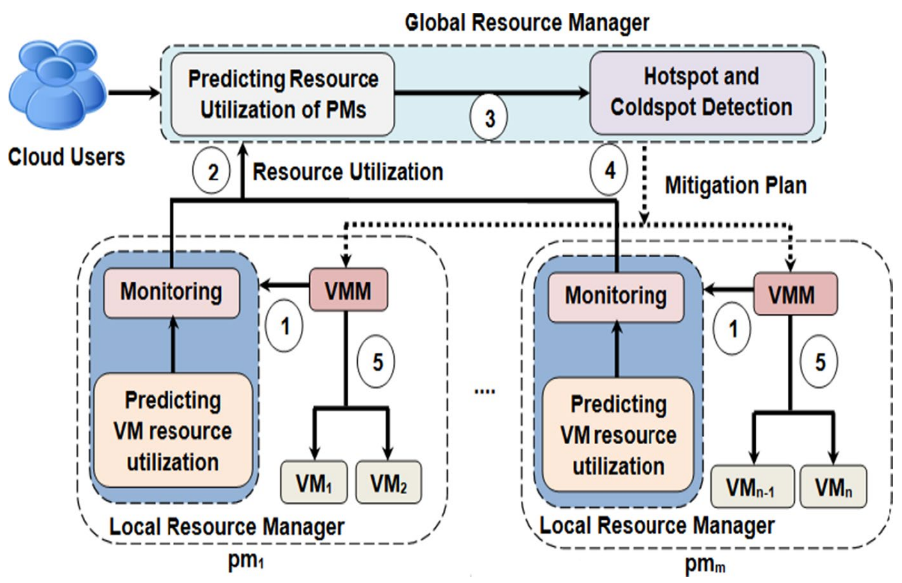


Fig. 7 The system model

which R_j^d depicts the availability of resource d in pm_j. The data center possesses ample capacity to serve all arriving and currently running requests. Next we assume a set of VMs denoted by $L_{\mathrm{VM}} = \{\mathrm{vm}_1, \mathrm{vm}_2, \mathrm{vm}_3, \ldots, vm_n\}$. Here, n indicates the number of VMs and r_i^d represents the resource demand d of vm_i. The VMs require to be allocated to PMs via a hypervisor or Virtual Machine Monitor (VMM). For initial resource assignment of VMs to PMs, the BFD algorithm is applied as it is one of the most prominent heuristic to solve the bin-packing problem. BFD algorithm always prefers a PM in such a way that the resources requested by the VM is closest to the number of available resources so that the resource wastage in the destination PM can be minimized [20]. However, due to dynamic workloads in the cloud, the requested usage of PMs and VMs fluctuate over time. Therefore, we need to optimize the initial VM placement using a consolidation process that could be applied periodically based on current and future workloads. In our proposed approach, the hotspots and coldspots mitigation algorithm is applied every 5 min in a cloud data center so that the actively used PMs and total energy consumption can be reduced based on the workload [20].

The proposed model comprises of two resource managers: (i) The Local Resource Manager (LRM) which resides on each PM as a component of the Hypervisor and (ii) The Global Resource Manager (GRM) which helps in acquiring data center view on a higher (global) level and in managing the VM requests in a data center [12]. The following steps illustrate the sequence of processes performed using these resource managers:

1. LRMs continuously monitors the current resource usage of its assigned PM and the state of its constituent (running) VMs. It then forecasts the future resource usages of all VMs placed in a PM on the basis of historical workload traces via the clustering-based stacked bidirectional LSTM method and periodically sends the information obtained to the GRM.
2. The responsibility of the GRM is to communicate with LRMs and collect information such as current and future resource usage of all VMs to acquire the global data center view.
3. Based on the information received, the GRM identifies hotspots and coldspots. It then builds a mitigation plan and issues the necessary commands to VMMs for its execution. The commands indicate the list of VMs for migration and their corresponding target PMs on the basis of mitigation algorithms.
4. After receiving the commands from GRM, the VMMs execute the actual migration of VMs.

Next, we discuss the power and energy modeling for PMs and the migration overheads of VMs to be migrated. Also, we describe the performance metrics to measure the QoS and SLA violations.

4.1 Power and energy modeling

In data centers, the energy consumption of computing nodes is mostly analyzed by the memory, CPU, power supplies, cooling systems, and disk storage. As reported in the literature [50], the CPU consumes maximum amount of energy, when compared to the other system resources of server power consumption. The SpecPower [51] benchmark defines the relationship between the server's utilization and power consumption. Based on this benchmark, the total power consumption of a cloud server increases linearly with the increase in its CPU usage [12]. Moreover, the server's power consumption is directly proportional to its CPU usage as written in Eq. (9).

$$\mathrm{RU}_j = \mathrm{RU}_j^{\mathrm{CPU}} \tag{9}$$

Therefore, using the server's CPU utilization, we can model the server's power consumption value ΔP_j as follows:

$$\Delta P_j = \begin{cases} P_{j,\mathrm{idle}} + (P_{j,\mathrm{peak}} - P_{j,\mathrm{idle}}) \times UT_j, & \text{if } \mathrm{UT}_j > 0 \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

Here, the idle power consumption $P_{j,\mathrm{idle}}$ denotes the half of the peak power consumption $P_{j,\mathrm{peak}}$ and UT_j is the percentage of utilization in pm_j. From Eq. (10), it is clear that the server's power consumption is linearly proportional to its CPU usage.

In a virtualized data center, the server's energy consumption is varied over time and depends on the incoming load or the VM's sizes [52].

Corresponding to the workload variation, the CPU usage of a cloud server also varies over time. Thus, we can represent the utilization UT_j as a function of time.

Server's total energy consumption ΔE_j is determined via the energy model [50] represented as:

$$\Delta E_j = \int_t \Delta P(\mathrm{UT}_j)\mathrm{d}t. \tag{11}$$

Determining analytical power consumption models for the modern multi-core and multi-thread processors is a very complicated research issue. Therefore, instead of adopting an analytical power model, we take advantage of real-time power consumption data rendered by the SPECpower benchmark results [51]. For this, we picked 2 cloud server configurations of dual-core CPUs as published in Feb 2011: HP ProLiant ML110 G5 (Intel Xeon 3075, (2 cores × 2660 MHz, 4 GB), and HP ProLiant ML110 G4 (Intel Xeon 3040, 2 cores × 1860 MHz, 4 GB). The selected servers' power consumption features are presented in Table 3 [12].

4.2 Overhead of live VM migration

Live VM migration permits to transfer of a VM between the PMs without suspending the running applications and with a short duration of downtime. It also assists in consolidating the VMs to a lesser number of PMs dynamically. Although, live VM migration affects the performance of running applications of migrating VMs [53]. Previous studies have shown that, due to migration, the average degradation of performance along with the downtime of the application running in that VM is almost 10% of its CPU usage [12]. Each migration phase may also produce an SLA violation. Thus, it is a very critical factor to reduce the VM migrations. The live VM migration time primarily relies on the VM's total memory and available network bandwidth. For interpreting the migration model, the overall delay and performance degradation of any VM vm_i [12, 54] can be estimated as:

$$\mathrm{Perf}_{\mathrm{deg}_i} = 0.1 \cdot \int_{t_0}^{t_0+T_{m_i}} \mathrm{UT}_i(t)\mathrm{d}t, \tag{12}$$

$$T_{m_i} = \frac{M_i}{B_i} \tag{13}$$

where $\mathrm{Perf}_{\mathrm{degr}_i}$ represents the performance degradation realized by vm_i ; T_{m_i} indicates the time taken to finish the migration of vm_i ; t_0 denotes the time when the migration

Table 3 Servers' energy consumption with respect to load levels in Watts [12]

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

of vm_i begins; $\mathrm{UT}_i(t)$ shows the CPU usage of vm_i. M_i shows the total memory utilized by vm_i ; and B_i indicates the available network bandwidth.

4.3 QoS and SLA violation model

Meeting the customers' QoS requirements is one of the primary objectives of cloud service providers (CSPs). QoS demands are generally determined through SLAs. Furthermore, the excessive VM consolidation may lead to the hotspots in PMs, VMs performance degradation, and increased response time while finishing the jobs executing inside the VMs, which, in result, produces SLA violations and also impacts the QoS performance. In [12], the researchers proposed performance metrics to evaluate the SLA violation levels in an IaaS cloud. The performance metrics are as follows: SLA violation time per active PM (SLATAPM) as determined in Eq. (14), performance degradation due to migrations (PERFDM) represented in Eq. (15), and the merged SLAV metric expressed in Eqs. (16) and (17).

$$\mathrm{SLATAPM} = \frac{1}{m}\sum_{j\in \mathrm{PM}} \frac{T_{\mathrm{max}_j}}{T_{\mathrm{act}_j}} \tag{14}$$

where T_{max_j} is defined as the total time, when the pm_j has encountered 100 % utilization in any type of d resource and T_{act_j} defines the total active state time for pm_j.

$$\mathrm{PERFDM} = \frac{1}{n}\sum_{i\in \mathrm{VM}} \frac{\mathrm{Perf}_{d_i}}{T_{r_i}} \tag{15}$$

where Perf_{d_i} denotes the performance degradation in vm_i due to migrations and T_{ri} is the total resource demand by vm_i in its lifetime.

As stated by [12], the SLATAPM and PERFDM metrics are both equally essential to characterize the SLA violation level of the infrastructure. The key reason behind the SLATAPM formulation is in the case when a hotspot is created in any PM, or the CPU utilization reaches to 100%, then the running applications' performance inside the VMs assigned to that PM is restricted by the PM's capacity. Consequently, the VMs executing in that PM may not gain their expected performance level throughout the peak workloads. It means the SLATAPM metric value rapidly increases with the PMs' over-utilization. On the other hand, the PERFDM metric is useful to measure the overall performance degradation due to migrations. The value of PERFDM rises with an increase in the VM migrations.

By combining the SLATAPM and PERFDM metrics, we can formulate the SLAV metric to determine the performance degradation and impact of QoS caused due to overloading of PM and also due to migrations. The SLAV metric can be stated as follows:

$$\mathrm{SLAV} = \mathrm{SLATAPM} \times \mathrm{PERFDM} \tag{16}$$

$$\mathrm{SLAV} = \frac{1}{m}\sum_{j\in \mathrm{PM}}\frac{T_{\mathrm{max}_j}}{T_{\mathrm{act}_j}} \times \frac{1}{n}\sum_{i\in \mathrm{VM}}\frac{\mathrm{Perf}_{d_i}}{T_{r_i}} \tag{17}$$

We can minimize the energy consumption at the risk of an increased SLAVs level. Thus, we combined both the SLA violations and Energy factors as ESV performance metric [12], which can be formulated as follows:

$$\mathrm{ESV} = \Delta E \times \mathrm{SLAV} \tag{18}$$

where ΔE signifies the overall energy consumption of all the PMs.

5 Proposed multivariate resource utilization prediction-based hotspots and coldspots mitigation

In this section, our proposed multivariate resource utilization prediction-based hotspots and coldspots mitigation approach is segmented into the following parts: In Sects. 5.1 and 5.2, we discuss the clustering-based stacked BiLSTM deep learning model and feature selection methods, respectively. The proposed algorithms for hotspots and coldspots detection, potential VM selection, and destination PM selection are described in Sect. 5.3.

5.1 Multivariate resource usage prediction

The time series data components can be decomposed into four integral parts: seasonality, trends, level, and noise or randomness. More often, the forecasting in time series applies the information in a time series to predict the given series's future values. A univariate time series contains only one single time-dependent variable. To predict cloud resource usage, the models interpret the temporal usage patterns present in the desired resource metric's historical data and forecast future trends of data. Nevertheless, for robust prediction performance in predicting new values of the target resource metric, the presence of temporal variations in other associated resource metrics can also be considered along with the target resource metric itself. A multivariate time series analysis considers more than one time-dependent variable simultaneously. During forecasting, each variable depends on its past values and has some dependency on other variables of the given dataset. Also, this dependency is utilized for predicting future resource values. Multivariate analysis for resource utilization prediction uses multiple resource usage metrics to investigate the relationships and their consolidated influence on the target resource usage metric. The multivariate time series analysis offers greater statistical power than exploring individual features and strictly model the existence. Each decision includes the study of multiple variables [41]. It applies various statistical techniques to identify the relationship between numerous features and also correlate how an individual metric is more critical while generating a final out-of-sample usage prediction of the target metric.

This paper analyzes the combined effect of different resource usage metrics for future utilization value of the target resource metric. Next, we assume Y is a multivariate time series to describe the system's state by applying other resource metrics y with different time intervals. Here, Y is characterized as:

$$Y = \left(y^1, y^2, ..., y^t, ..., y^T\right) \tag{19}$$

where $y^t = \left[y_1^t, y_2^t, ..., y_L^t\right]^T, \forall = 1, ..., T$.

were T represents total observations used in the analysis, and y_i^t indicates the value of any resource usage metric i, which is reported at t time period. The variable L signifies the number of resource metrics used for observation.

For multivariate time series prediction, the forthcoming value of any resource y_j is predicted using the combination of two individual functions $G_1(.)$ and $G_2(.)$ as:

$$\hat{y}_j^{\,t} = G_1\left(y_j^{t-1}\right) + \sum_{p=1, p\neq j}^{D} G_2\left(y_p^{t-1}\right) \tag{20}$$

where the function $G_1(.)$ is intended to learn the correlation of the target metric with itself, and function $G_2(.)$ is designed to learn the correlation of the target metric with the other associated metrics.

This paper suggests extending the classical univariate time series prediction models to deep learning-based multivariate time series prediction models.

5.1.1 BiLSTM (Bidirectional LSTM)

Bi-LSTM combines LSTM networks, and Bi-directional RNNs (BiRNN) [55], which processes the input sequence in both directions, i.e., forward and backward.

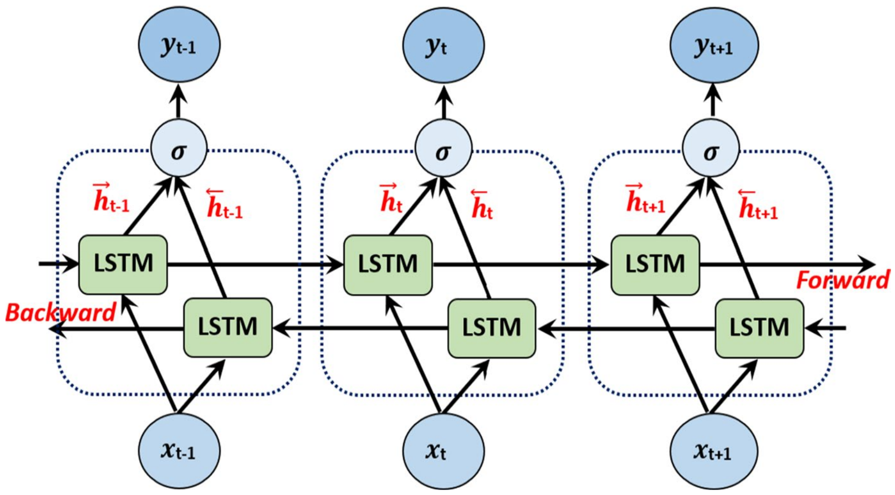


Fig. 8 Structure of BiLSTM

It has been shown that the bidirectional networks perform better than the unidirectional LSTM in several domains. Figure 8 shows the unfolded BiLSTM layer architecture comprising a forward and backward layer of LSTM. Here, the forward layer output sequence denoted by $\overrightarrow{\mathrm{h}}$, is repetitively computed by applying inputs in a positive order via time series $T-n$ to $T-1$, while the sequence of backward layer output, $\overleftarrow{h}$, is determined by operating the reversed inputs via time series $T-1$ to $T-n$. By using conventional LSTM equations given in Eqs. (3) to (8), we can find the forward layer and backward layer outputs. Finally, the BiLSTM layer produces Y_T processed output vector, where each element is determined by applying the following equation:

$$y_t = \sigma(\overrightarrow{h}, \overleftarrow{h}) \tag{21}$$

where σ function integrates the two output sequences. Equivalent to the LSTM layer, we can represent the final output of a BiLSTM layer through a vector, $Y_T = [y_{T-n}, ..., y_{T-1}]$, in which the y_{T-1} denotes the predicted next time step value.

5.1.2 Clustering-based stacked bidirectional LSTM

The deep LSTM model can build up with several stacked LSTM networks' hidden layers. The LSTM hidden layer output will be given as an input to the successive LSTM hidden layer. Such stacked layers of architecture improve the capability of neural networks. As discussed earlier, the BiLSTMs utilizes both forward and

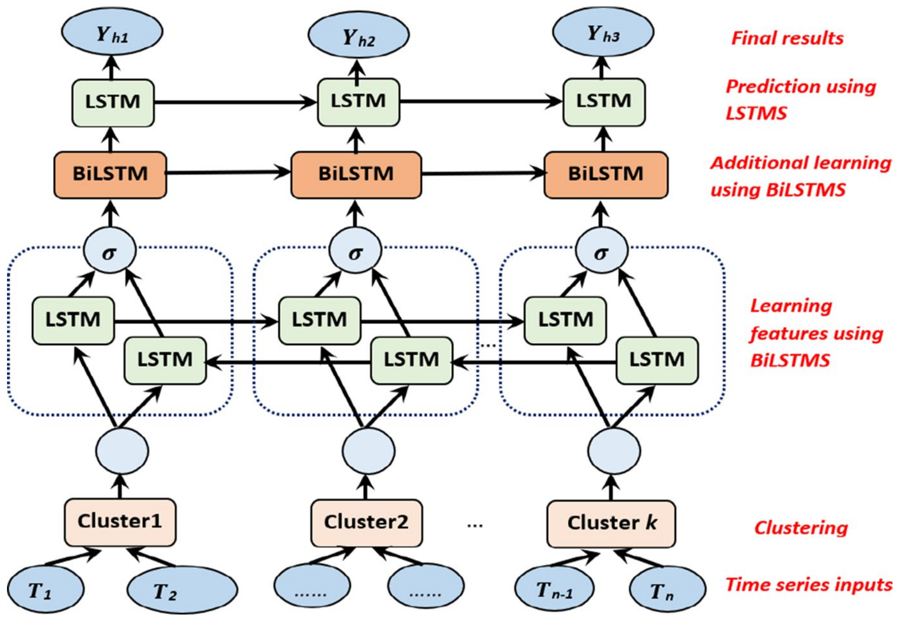


Fig. 9 Proposed architecture

backward dependencies. Therefore, in this study, we present a Stacked BiLSTM architecture, which uses the BiLSTMs to learn more useful information via the spatial-time series data and the LSTM layer as the top (last) layer of the architecture which only utilize learned features through lower layers, to calculate and generate the predicted values.

In Fig. 9, we have illustrated the proposed deep learning architecture. In input, it takes the spatial time series data and predicts the future resource utilization values for multiple future time steps. This network is based on the hypothesis that the cluster data contain a group of patterns, and recognizing those groups and training against them, an individual deep learning model for each will result in better predictions.

In Algorithm 1, we have discussed the step-by-step phases of the proposed deep learning network. To calculate the out-of-sample prediction, first, we classify our dataset into training and test sets. Moreover, the training set is classified into k number of clusters with the help of an unsupervised learning model, i.e., K-Means. Obtaining the optimal value of k is a critical factor. If the value of k is very high, then there might be some pattern redundancy, and also, a large number of network models will have to be trained, which would be computationally expensive. While, if the value of k is low then there might be some undetected patterns left in the data and scattered over several clusters. Therefore, the value of k is selected by the trial-and-error method. This model is saved and further used for classifying samples of the test dataset. By using the stacked BiLSTM deep learners, we train each of these clusters separately. Thus, there is a total k number of stacked BiLSTM deep learners for learning of k clustered datasets. For each stacked BiLSTM deep learner, there is a BiLSTM layer that acts as the first feature-learning layer, and, on top of that, an LSTM layer works as the final layer. For complex and exhaustive feature learning using input data, we apply additional BiLSTM layers. The clustering-based stacked BiLSTM deep learners are capable of the multi-step-ahead predictions using the historical resource usage values.

To capture the relevant features set, we combine the clustering-based stacked BiLSTM deep learning model with the feature selection approaches. In the next subsection, we discuss some feature selection schemes for the effective selection of resource metrics from the dataset.

Algorithm 1: Prediction Algorithm

```
1  Input: Number of clusters $n_c$; Multi-variate time series $S_{in} \leftarrow \{T_1, T_2, T_3, ....T_n\}$ where
   $T_i \leftarrow \{x_1, x_2, x_3, ....x_{n_f}\}$ and $n_f$ is number of features; Number of iterations $N$;
2  Output: New improvised prediction model $P_{final}$
3  Divide the time series into two parts:
4  Training Set $S_{train} \leftarrow \{T_i\}$: i $\epsilon$ (0,l) and $l = 0.8 * n$;
5  Test Set $S_{test} \leftarrow \{T_j\}$: $j \epsilon [l, n]$;
6  Let Set $Y_{train}$ be a set of training outputs/targets
7  Let Set $C \leftarrow \{C_1, C_2, C_3, ....C_{n_c}\}$ be a set of clusters where $C_i \leftarrow \{T_i\}$: $T_i \ \epsilon \ S_{train}$, and $C_i \cap$
   $C_j = \phi$ if i!=j
8  Let $P_{final} \leftarrow \{P_1, P_2, P_3 .....P_{n_c}\}$, where $P_i$ is a weak model.
9  while ($N$!=0):
10 begin
11     for $X$ in $S_{train}$:
12     begin
13         if $X \ \epsilon \ C_i$:
14         $Y_{h1} \leftarrow h1_i(W1, X)$, where $Y_{h1} = \{y_{11}, y_{12}, y_{13} .....y_{1m}\}$ : $m$ != $n$
15         $Y_{h2} \leftarrow h2_i(Y_{h1})$, where $Y_{h2} = \{y_{21}, y_{22}, y_{23} .....y_{2q}\}$ : $q$ != $m$
16         $Y_{h3} \leftarrow h3_i(Y_{h2})$, where $Y_{h3}$ is the target output.
17         Calculate difference between predicted and target value.
18         $\varepsilon = Y_{h3} - Y_{target,i}$
19         Adjust weights using $\varepsilon$ as $\Delta$w = $\alpha * \varepsilon * h3_i`(Y_{h2})$ $w_{new} = w_{old} + \Delta$w
20         Adjust intermediate weights for hidden layers using back-propagation
21     $N = N - 1$
```

5.2 Feature selection schemes for prediction of multivariate resource usages

5.2.1 Pearson's correlation

Pearson's correlation can be defined as the linear correlation between two variables A and B. Therefore, a subtle change in A will also result a similar change in B. This form of relation outputs a value from -1 to 1, where -1 represents a negative linear correlation and a $+1$ represents a positive linear correlation. Thus, change is proportional. If there is no linear correlation at all, the formula outputs 0. Pearson's correlation is given as:

$$p_r = \text{cor}(a_i, a_j) = \frac{\sum_{n=1}^{N}(a_i^n - \mu_i)(a_j^n - \mu_j)}{\sqrt{\sum_{n=1}^{N}(a_i^n - \mu_i)^2}\sqrt{\sum_{n=1}^{N}(a_j^n - \mu_j)^2}} \tag{22}$$

where A and B are represented as a_i and a_j, respectively. These two variables are resource matrices such that $a_i = \left[a_i^1, a_i^2, a_i^3, ...a_i^N\right]$ and $a_j = \left[a_j^1, a_j^2, a_j^3, ...a_j^N\right]$ whereas μ_i an μ_j are the mean value of the respective features. Thus, the target metric is correlated with every other resource metric to obtain a value between *-1* and $+1$. A threshold is primarily decided, and only those resource metrices are picked whose p_r value greater than the limit of threshold. The threshold limit stated as M, which is the median of all the scores evaluated by the correlation formula.

5.2.2 Spearman's correlation

This correlation is also called as a rank coefficient, is a measure of ranks between two variables (termed as features). This correlation indicates the direction between a_i and a_j. It is assumed that the two variables tend to change at different rates; therefore, they are numbered or ranked in order to achieve a value between − 1 to + 1. Here, + 1 indicates that if one variable increases, the other is bound to increase, and − 1 depicts the correlation that if one variable decreases, so will the other. The values in the metrices are ranked from high to low, meaning the highest value in the variable will be ranked 1 so on and so forth up until the last value is ranked N. The Spearman's correlation coefficient is denoted by r_s and is given by,

$$r_s = 1 - \frac{6\sum_{n=1}^{N}(d^n)^2}{N(N^2 - 1)} \tag{23}$$

where d^n is the difference between the ranks of the two resource metrices.

5.2.3 Granger causality

Granger's Causality is a statistical procedure which states that if a variable A "Granger causes" another variable B, then past values of A has some valuable details about B that will help predict B in a better way. Here, the two variables are denoted as time series. It is substantial to note that unlike other correlation techniques, Granger causality is asymmetric, meaning if A Granger causes B, it does not mean B will Granger cause A. Let A and B be two-time series denoted by the variables a_i and a_j. The causality is calculated by applying linear regression on the value we need to predict. Suppose we have to predict a_i; therefore, a_i will be formulated in two linear regressions. One, where $\hat{a}_i$ is predicted only using the past values of a_i (termed as $\hat{a}^n_{i\,(\text{restricted})}$) and secondly, a prediction using the past values of a_i as well as a_j (termed as $\hat{a}^n_{i\,(\text{unrestricted})}$).

$$\hat{a}^{\,n}_{i(\text{restricted})} = w_0 a_i^{n-1} + w_1 a_i^{n-2} + \cdots + w_{\rho-1} a_i^{n-\rho} \tag{24}$$

$$\begin{aligned}\hat{a}^{\,n}_{i(\text{unrestricted})} =& w_0 a_i^{n-1} + w_1 a_i^{n-2} + \cdots + w_{\rho-1} a_i^{n-\rho} + w'_0 a_j^{n-1} \\ &+ w'_1 a_j^{n-2} + \cdots + w'_{\rho-1} a_j^{n-\rho}\end{aligned} \tag{25}$$

where w and w' are the weights calculated by the linear regression on learning from the past values. ρ represents the number of steps taken back to predict the next value in the series. G-Test is the test that depicts the overall significance of the calculation and whether or not a_j's past will truly help us predict a_i or not.

$$\text{G-Test} = \frac{\delta_{\text{restricted}} - \delta_{\text{unrestricted}}}{\delta_{\text{unrestricted}}}\left(\frac{N - (2\rho + 1)}{\rho}\right)$$

where δ is denoted as sum of square error in prediction. Given as,

$$\delta_{\text{unrestricted}} = \sum_{n=1}^{N} \left(\hat{a}_{i_{\text{unrestricted}}}^{n} - a_i^n\right)^2 \tag{27}$$

$$\delta_{\text{restricted}} = \sum_{n=1}^{N} \left(\hat{a}_{i_{\text{restricted}}}^{n} - a_i^n\right)^2 \tag{28}$$

5.2.4 Mutual information

The mutual information technique measures the relationship between two random variables. It is described as the mutual dependencies between the two variables and how much information a random variable A has about another random variable B. It is given as the two variables' entropy combined subtracted from the sum of the uncertainty of the two variables separately. Here again, the variables are the two resource metrices. One is our candid resource metric (CPU rate), and the other is the resource metric that needs to be validated for acceptance or rejection.

$$E(a_i) = \sum_{n=1}^{N} \rho\left(a_i^n\right) log\left(\rho\left(a_i^n\right)\right) \tag{29}$$

where $\rho\left(a_i^n\right)$ is the probability distribution of the resource metric a_i^n

$$E\left(a_i, a_j\right) = \sum_{n=1}^{N} \sum_{n=1}^{N} \rho\left(a_i^n, a_j^n\right) log\left(\rho\left(a_i^n, a_j^n\right)\right) \tag{30}$$

Therefore, mutual information can be given as

$$M(a_i, a_j) = E(a_i) - E(a_i, a_j) + E(a_j) \tag{31}$$

Larger the value of mutual information depicts a higher dependency of the two resource metrices, and such a metric cannot be omitted. If $M(a_i, a_j)$ is higher than the threshold, then it is accepted by the algorithm. The range of the value is $[0, \infty]$.

5.2.5 ANOVA

Analysis of variance is the ratio between variability between the sample means and variability within the distributions.

$$F = \frac{\text{Variance Between}}{\text{Variance Within}} \tag{32}$$

$$\text{Variance Between} + \text{Variance Within} = \text{Total Variance} \tag{33}$$

The variability between the means in the numerator is large compared to the variance within the samples in the denominator, the ratio will be much larger than 1. Then, the samples most likely do not come from a common population. Therefore, we discard the null hypothesis, which states that the means are equal. The inputs will be taken in the format of (nsamples, nfeatures) and output a 1D array, including an F test score for each of the feature (1, nfeatures).

5.2.6 Kendall's Tau

Kendall's nonparametric Tau, more commonly known as Kendall's rank correlation coefficient, is the degree of consistency between two columns of ranked data. Kendall's Tau outputs a value from − 1 to + 1. This method deals with concordant and discordant pairs in two columns. A concordant pair represents the measure ranks lie below a certain rank that is greater than it. Therefore, a discordant pair shows the observed ranks lower than a stated rank whose values are lesser than the stated rank.

5.3 Prediction-based hotspots and coldspots mitigation

Prediction-based hotspots and coldspots mitigation perform 3 steps to accomplish an optimal dynamic VM consolidation (as shown in Algorithm 2). First, the algorithm distinguishes the PMs having hotspots (Sect. 5.3.1), then selects some VMs from each hotspot PM to perform migration (Sect. 5.3.3) and determines destination PMs for the VMs (Sect. 5.3.4). To detect coldspots (Sect. 5.3.2), the algorithm chooses one PM with the lowest CPU usage from the PM list not having hotspots and migrates its deployed VMs to a suitable destination PM.

Algorithm 2: Dynamic hotspots and coldspots mitigation

```
Input: $PM_{active}$
Output: MigrationPlan
for each $pm \in PM_{active}$ do
begin
    if(HotspotsDetection(pm)=true) then
    SelectedVM ← VM with minimum $EQV$;
    MigrateVMList ← pm.SelectedVM
    $PM_{hot} \leftarrow PM_{hot} \cup pm$
end
for each $pm$ not in $PM_{hot}$ do
begin
    if(ColdspotsDetection(pm)=true) then
    $PM_{cold} \leftarrow PM_{cold} \cup pm$
end
MigrateVMList ← All VMs of $PM_{cold}$;
$PM_{available}=PM_{active} - PM_{hot} - PM_{cold}$;
MigrationPlan ← Destination PMs from $PM_{available}$ for MigrateVMList;
return MigrationPlan
```

5.3.1 Prediction-based hotspots detection

In the process of dynamic VM consolidation, each PM must be identified whether having a hotspot or not. To detect the PMs with hotspots, we propose a prediction-based hotspots detection method as presented in Algorithm 3. It takes a set of active PMs as input and determines whether the PMs creating a hotspot or not. Algorithm 3 is explained step-by-step as follows: In line 3, we set the minimum window length of historical data as 20 using the cross-validation technique for best time window selection. In line 3, the utilization of a pm at time instant t is determined by using the current usage and total capacity of pm. For hotspot detection, we use a dynamic upper threshold method. If the utilization value of at least one resource (i.e., memory or CPU) for any pm is greater than the threshold value, then it has a hotspot. In this method, the threshold value is decided by using the median absolute deviation (MAD) as discussed in [12]. In lines 6–10, if the PMs have insufficient utilization data, then the hotspot decision is only made by comparing the current utilization of pm $\mathrm{UT}_{\mathrm{pm}}(t)$ with a static threshold. In lines 12–16, if the PMs have sufficient history data, then the algorithm uses the future value as well. For this, it first determines the future utilization of pm by applying the clustering-based stacked BiLSTM model (line 12). The pm is considered to be in the set of hotspots pm if the current and future usage value is higher than the upper threshold level of pm (line 13). This condition demonstrates that the pm is a potential candidate that executes the migration operation if and only if it has a hotspot in both the current time instant t and near future at time instant $t + 1$. It means Algorithm 3 analyzes not only the present situation but also the near-future situation. It can avoid useless migrations by performing necessary migration only. Moreover, a clustering-based stacked BiLSTM deep learning model can avoid an SLA violation rate by detecting a pm has a hotspot in advance.

Algorithm 3: Prediction Based Hotspots Detection

```
1  Input: $PM_{active}$
2  Output: Is PM having hotspot or not
3  $windowLength \leftarrow 20$; $UT_{pm}(t) = CU_{pm}(t)/Cap_{pm}(t)$; Set $Th_{hot}$ applying MAD =
     1-s*Mad;
4  for each $pm \in PM_{active}$ do
5  begin
6      if $Length.UTHistory_{pm}(t) < windowLength$ then
7      if($UT_{pm}(t) > Th_{static}$) then
8      return true;
9      else
10     return false;
11     else
12     predict $UT_{pm}(t+1)$ using clustering-based stacked BiLSTM model;
13     if(($UT_{pm}(t) > Th_{hot}$) && ($UT_{pm}(t+1) > Th_{hot}$)) then
14     return true;
15     else
16     return false;
17 end
```

5.3.2 Prediction-based coldspots detection

Once the hotspots are identified, the coldspots detection procedure is executed. To bring down the overall energy consumption, the actively used PMs are switched into a low-power state or sleep state. The proposed prediction-based coldspots detection algorithm works as follows: Algorithm 4 takes the active PM set's input and evaluates whether the PM is having a coldspot or not. The phases of Algorithm 4 is almost similar to the steps of Algorithm 3. The differences lie in lines 7 and 13. If the $\mathrm{UT}_{\mathrm{pm}}(t)$ is less than or equal to $\mathrm{Th}_{\mathrm{cold}}$, then the pm is having coldspot. After collecting the windowLength of 20 utilization historical data, the pm is considered to be in the set of coldspots pm if the current and future utilization values are less than or equal to the lower threshold.

Algorithm 4: Prediction Based Coldspot Detection

```
1  Input: $PM_{active}$
2  Output: Is PM having coldspot or not
3  $windowLength \leftarrow 20$; $UT_{pm}(t) = CU_{pm}(t)/Cap_{pm}(t)$; Set $Th_{cold} = 30\%$
4  for each $pm \in PM_{active}$ do
5  begin
6      if $Length.UTHistory_{pm}(t) < windowLength$ then
7      if($UT_{pm}(t) \leq Th_{cold}$) then
8      return true;
9      else
10     return false;
11     else
12     predict $UT_{pm}(t+1)$ using stacked BiLSTM model;
13     if(($UT_{pm}(t) \leq Th_{cold}$) && ($UT_{pm}(t+1) \leq Th_{cold}$)) then
14     return true;
15     else
16     return false;
17 end
```

5.3.3 Energy and quality aware VM selection

After detecting the PMs with hotspots, the next stage is to pick VMs to be migrated from that PM. It is obvious that the selection of VMs with lower CPU utilization will lead to a smaller rise in energy consumption. Thus, to improve the QoS level and migration time, we should select the VMs with smaller memory for migration. Therefore, whenever any PM creates hotspot, we compute the cost of energy and QoS (EQV) for each of the VMs assigned to it. The EQV value of VM stated as follows:

$$\mathrm{EQV}_{\mathrm{vm}} = w1 \cdot \frac{\mathrm{CU}_{\mathrm{vm}}^{\mathrm{CPU}}(t)}{\mathrm{Cap}_{\mathrm{vm}}^{\mathrm{CPU}}(t)} + w1 \cdot \frac{\mathrm{CU}_{\mathrm{vm}}^{\mathrm{MEM}}(t)}{\mathrm{Cap}_{\mathrm{vm}}^{\mathrm{MEM}}(t)} \tag{34}$$

In comparison to a VM with a higher EQV cost, we can achieve maximum energy-efficiency and minimum performance degradation by migrating the VM with a smaller EQV cost.

5.3.4 Destination PM selection

After building the list of selected VMs for migration, VM consolidation's last step is to obtain the target PMs for the selected VMs. VMs to PMs placement is an NP-hard problem and could be solved with the help of a multidimensional bin packing problem. In the bin packing, each PM is considered as a bin and each VM is mapped with an item of a different volume. The bin packing problem's objective is to pack all the items into bins such that the used number of bins can be minimized [13]. To achieve this goal, we designed a solution based on the BFD algorithm because it used a minimum number of bins compared to the First-Fit and Next-Fit schemes [56]. The step-by-step process of destination PM selection is illustrated in Algorithm 5.

Algorithm 5: Destination PM Selection

```
Input: $PM_{List}, VMList$
Output: Assignment of VMs
Set $windowLength \leftarrow 20$;
if($Length.UTHistory_{pm}(t) < windowLength$) then
return PBFD($PM_{List}, VMList$);
else
Set $DestinationPM \leftarrow \phi$;
Set $MigrationList \leftarrow \phi$;
SortedVMList ← Sort VMs of $VMList$ by their CPU usage prediction in desceneding order;
for each $vm \in VMList$ do
begin
    $Allocation \leftarrow false$;
    for each $pm \in PMList$ do
    begin
        if(($UT_{pm}(t) + UT_{vm}(t) \leq Cap_{pm}(t)$) && ($UT_{pm}(t+1) > maxUT$)) then
        DestinationPM ← $DestinationPM \bigcup pm$;
        $Allocation \leftarrow true$;
    end
    if($Allocation = true$) then
      $MigrationList = MigrationList \bigcup \{vm, DestinationPM\}$;
end
return $MigrationList$;
```

The destination PM selection algorithm sorts all chosen VMs in the decreasing sequence of their predicted CPU value. In next step, the algorithm assigns each VM to a PM having maximum CPU utilization after placement. This is because selecting the PM with higher CPU usage fulfills the consolidation goal effectively, which allows the packing of many VMs to a lesser number of PMs. In the initial phase, when the length of historical records is less than the window length, we apply the Power-aware BFD (PBFD) algorithm [12] as no predicted CPU utilization value is available yet.

5.4 Complexity analysis

To compute the time complexity of Prediction Algorithm (Algorithm 1), we observe the computational complexities of different neural network approaches analyzed in the work of Behera et al. [57]. Based on this study, we consider a basic neural network model having Q number of cells in the hidden layer, T inputs, and W outputs. In the case of simple multi-layer perceptron (MLP) with one hidden layer, we can formulate the total parameters as: $C_{\mathrm{MLP}} = \mathrm{TQ} + \mathrm{QW}$. For RNN, if we replace the hidden layer cells with RNN units, then the total parameters will be $C_{\mathrm{RNN}} = \mathrm{TQ} + Q^2 + \mathrm{QW}$, here Q^2 represents the recurrent connection. Similarly, in the case of LSTM cell with 3 gates and a cell state, the total parameters can be represented as $C_{\mathrm{LSTM}} = 4\mathrm{TQ} + 4Q^2 + 3Q + \mathrm{QW}$. While in the case of a GRU cell with two gates and a hidden state, the total parameters can be expressed as: $C_{\mathrm{GRU}} = 3\mathrm{TQ} + 3Q^2 + 3Q + \mathrm{QW}$. In the case of BiLSTM, we know that each BiLSTM layer is stacked with two BiLSTM layers. Thus, we can express the total parameters for a BiLSTM layer as: $C_{\mathrm{BiLSTM}} = 2 * C_{\mathrm{LSTM}}$. If we stack up P number of layers, the total parameters can be represented as: $C_{\mathrm{BiLSTM}} = P * C_{\mathrm{LSTM}}$. If we consider the input layer to be the same across all stacked networks, then the overall complexity of the proposed prediction model with Θ number of training epochs can be expressed as $C_{\mathrm{model}} \approx O((Q(Q + W) * \Theta)$.

In order to describe the time complexity of hotspots and coldspots mitigation algorithm, we analyze the time complexity of its sub-algorithms, respectively. The hotspots and coldspots mitigation algorithm (Algorithm 2) comprises Prediction-Based Hotspots Detection (Algorithm 3), Prediction-Based Coldspot Detection (Algorithm 4), and Destination PM Selection (Algorithm 5). Assuming s number of PMs and k number of VMs, the time complexity of hotspots and coldspots mitigation algorithm is $O(sk)$ in worst-case. The time complexity of both Prediction-Based Hotspots Detection (Algorithm 3), and Prediction-Based Coldspot Detection (Algorithm 4) is $O(k)$ in worst-case. Destination PM Selection (Algorithm 5) identifies each PM for MigrateVMList, so its worst-case time complexity is $O(sk)$. Thus, the total time complexity of hotspots and coldspots mitigation algorithm (Algorithm 2) is $O(sk)$.

6 Experimental results

First, we discuss workload data, the simulation environment with assumptions, performance metrics, and benchmark approaches. Then, we present the experimental evaluations for validation of our proposed method.

6.1 Workload data

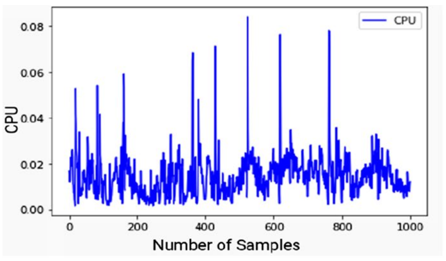
In this work, we apply the real trace data of a Google cluster [39]. This trace provides cluster's workload data containing 12,500 PMs recorded in May 2011. The Google cluster trace comprises running-time traces of over 650 thousand real-time jobs dynamically entering, executing, and leaving the Google cluster for 29 days, where resource utilization values are bundled at 10 seconds time-frame. Resource tables of Google cluster trace are broadly classified into three major categories, *Machines, Jobs & Tasks, and Resource Usage*. Each category has one or more than one table containing multiple features. In the present study, we have used the resource usage

Table 4 List of resource usage metrics

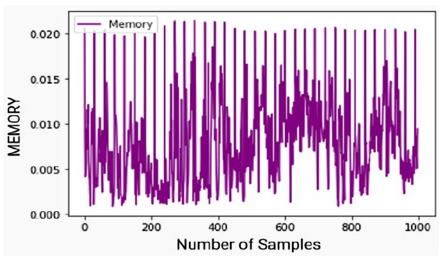
Resource metrics	Description
CPU	Usage of aggregated CPU
MEM	Usage of aggregated memory
MAXM	Maximum memory utilization observed
VM	Assigned memory (may or may not be used)
MAXC	Maximum CPU utilization (over interval)
UPC	Aggregate unmapped page cache
TPC	Usage of total page cache
CPI	Cycles per instruction over all nodes
MAXD	Maximum disk I/O time detected (over interval)
DSP	Disk capacity space utilization over all nodes
MAI	Memory accesses per instruction over all nodes
DIO	Disk I/O time over all disks

table, and from a total of 20 columns, only 12 different resource usage metrics are taken for CPU and memory usage prediction. Like the others, resource metrics such as start_time, end_time, job_ID, and tasks_ID, etc., do not provide any correlation with CPU or memory. The diverse resource utilization metrics applied in this work are described in Table 4.

During the data preparation, the rows with missing and incomplete information are dropped. Each row corresponds to data collected by the cluster for over 10 seconds. We have merged rows corresponding to 5 minutes' worth of data into a single row for prediction. A total of 10 days of resource workload with 86,880 samples is taken to train the deep learning models and predict CPU and memory usage. Both resource usage values are normalized, respectively. In Figures and , we represent the sample CPU and memory data. To show the characteristics of the workload traces, in Figures and , we present the rolling mean and standard deviation of CPU and memory data, respectively. Furthermore, we have applied different feature selection approaches to decide the most relevant set of features for prediction. For evaluation, we categorized the data into a validation, training, and test sets. For validation, we

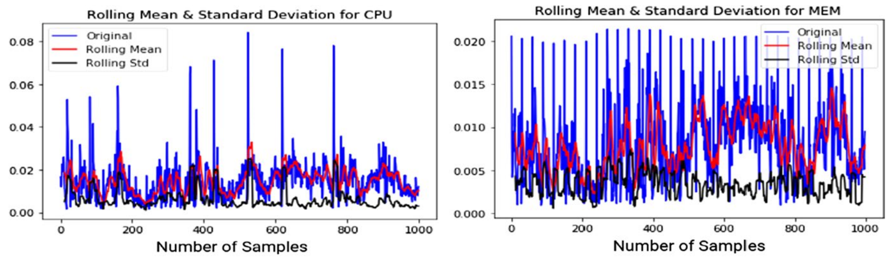


(a) Sample of CPU values



(b) Sample of memory values

Fig. 10 Sample values of Google cluster traces



(a) Mean and standard deviation for CPU **(b)** Mean and standard deviation for memory

Fig. 11 Analysis of rolling mean and standard deviation

perform the parameter tuning of different models. Then, we select the best fitting parameters for each of the corresponding models to generate future predictions for the next 90, 180, and 270 steps (Figs. 10 and 11).

6.2 Simulation environment

To validate the proposed method's performance, we use CloudSim 3.0.3 toolkit [17]. We have conducted our simulations on a compute server powered with Intel(R) Xeon(R) Processor E5-2630 v3 @ 2.40 GHz with 2 CPU sockets ($2 * 8 = 16$ core), 1 TB Hard disk, 128 GB RAM, and 64-bit Windows 10 OS. Moreover, a cloud data center is simulated with 800 heterogeneous PMs. Each PM of data center comprises a 2-core processor, 4 GB memory, 1GB/s network bandwidth, and MIPS is randomly generated between 1860 or 2660 MIPS for each core. In the PM list, half of the PMs are HP ProLiant ML110 G4 servers while the other half are HP ProLiant ML110 G5 servers. Based on the information available in Amazon EC2, 4 VM instances are used, i.e., small (1000 MIPS, 1.7 GB), micro (500 MIPS, 613 MB), High-CPU Medium (2500 MIPS, 0.85 GB) and Extra large (2000 MIPS, 3.75 GB)

Table 5 Hyper-parameter settings of the proposed model

Architecture details	
Parameter	Value
Default hidden layers	3
Units in hidden layers	60,120,60
Training step/epochs	70
Batch size	100
Loss function	Mean squared error
Optimizer	Adam
Activation function	tanh
Input layer neurons	12
BiLSTM layer neurons	60,120,60
Final layer neurons	1

[12]. The energy consumption value of all PMs is computed using the data represented in Table 3.

The list of hyper-parameters for the proposed deep learning model is shown in Table 5. To implement deep learning models' network architecture, we have used the Python language with the Keras library through the TensorFlow back-end via functional API.

6.3 Evaluation metrics

We have applied different performance metrics to validate the effectiveness of the proposed hotspots and coldspots mitigation approach. For analyzing the accuracy of prediction models, we have used the RMSE: Root Mean Square Error metric, which computes the standard deviation of the absolute errors as follows:

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{t=1}^{P} (y_i^t - \hat{y}_i{}^t)^2}{\Delta T}} \tag{35}$$

where ΔT indicates the length of prediction at 90, 180, and 270 steps. We use i to represent the desired resource usage metric. y_i^t shows the actual resource usage metric's value at particular time instant t, and $\hat{y}_i{}^t$ indicates the predicted value of resource metric at time instant t. We have used the iterative multi-step ahead resource usage predictions, which predict the CPU and memory resource utilization at 90, 180, and 270 steps ahead.

Then, we analyze the overall energy consumption across all PMs of data center. Each PM's energy consumption value is computed via the power and energy model, as discussed in Sect. 4.1. Next, we have considered the total migrations of all VMs during the mitigation of hotspots and coldspots in the data center. Another metric we have utilized to monitor the QoS and SLA violations. For this, we calculated the SLATAPM, PERFDM, and SLAV metrics, as explained in Sect. 4.3. Finally, the ESV metric is applied to measure the trade-off between QoS and energy consumption, as expressed by Eq. (18).

6.4 Benchmark approaches

For overall performance analysis of the proposed scheme, we have implemented the following algorithms present in the literature:

1. *Sercon* [58] This technique is based on the greedy bin-packing algorithm, which follows all-or-none migration property. In order to free the least loaded host, it tries to migrate all the VMs from the least loaded host to the most loaded host. During the migration process, if any of them is failed, then no migration is performed.
2. *ACSVMC (Ant Colony System based VM Consolidation)* [20] ACSVMC policy follows the meta-heuristic bin-packing approach. It applies an ant colony algo-

rithm to minimize SLA violations and energy consumption and produces a migration plan.

3. *MBFD (Modified Best Fit Decreasing)* [21, 50] In MBFD, the VMs are assigned to a most-loaded PM only if the total utilized capacity of the destination PM and migrated VM is not violating the hot threshold limit of the PM's total resource capacity. It means:

$$\mathrm{UT_{pm}}(t) + \mathrm{UT_{vm}}(t) \leq \mathrm{Th_{hot}} \times \mathrm{Cap_{pm}}(t) \tag{36}$$

Although the MBFD algorithm does not apply any prediction model, thus it can only migrate VMs of the PMs, creating hotspots to avoid SLA violations. Moreover, MBFD maximizes energy-efficiency via mitigating coldspots.

4. *Modified First Fit Decreasing (MFFD)* [21]
The MFFD algorithm applies the First Fit (FF) strategy and allocates a VM to the first PM that meets the current resource demands of the VM and ensures that VM and PM's total utilized capacity is less than the hot threshold limit of the PM's full resource vector. It means:

$$\mathrm{UT_{pm}}(t) + \mathrm{UT_{vm}}(t) \leq \mathrm{Th_{hot}} \times \mathrm{Cap_{pm}}(t) \tag{37}$$

5. *PUP-VMC (PM Usage Prediction-aware VM Consolidation)* [21] This technique only inspects the future resource usage of a target PM for assigning any VM. It also ensures that the PM's predicted total utilized capacity and the VM's current demand capacity should be lesser than the hot threshold limit of the PM's total resource capacity. It means:

$$\mathrm{UT_{pm}}(t+1) + \mathrm{UT_{vm}}(t) \leq \mathrm{Th}_{hot} \times \mathrm{Cap_{pm}}(t) \tag{38}$$

6.5 Simulation results

1. *Prediction performance using all features* We assess multivariate prediction models' accuracy for predicting future CPU usage and memory usage in Google

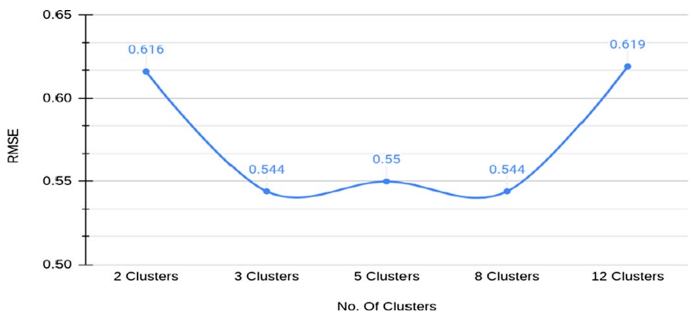


Fig. 12 Cluster analysis

Table 6 RMSE analysis for CPU and memory usage prediction using Google cluster's trace

RMSE of CPU usage prediction

Model	Univariate			Bivariate			All features		
	90 steps	180 steps	270 steps	90 steps	180 steps	270 steps	90 steps	180 steps	270 steps
ARIMA	0.00667	0.00674	0.0064	0.01087	0.01113	0.01122	0.00762	0.00773	0.00777
Linear Regression	0.00567	0.00571	0.00572	0.00555	0.00563	0.00568	0.00549	0.00555	0.00557
GRU	0.00560	0.00564	0.00566	0.00549	0.00553	0.00554	0.00539	0.00544	0.00547
LSTM	0.00558	0.00562	0.00565	0.00546	0.00550	0.00552	0.00537	0.00541	0.00542
BiLSTM	0.00555	0.00559	0.00561	0.00542	0.00546	0.00548	0.00534	0.00538	0.00540
Clustering-based Stacked BiLSTM	**0.00551**	**0.00555**	**0.00558**	**0.00540**	**0.00543**	**0.00546**	**0.00531**	**0.00535**	**0.00537**

RMSE of memory usage prediction

Model	Univariate			Bivariate			All features		
	90 steps	180 steps	270 steps	90 steps	180 steps	270 steps	90 steps	180 steps	270 steps
ARIMA	0.00542	0.00546	0.00544	0.00545	0.00556	0.00546	0.00545	0.00556	0.00546
Linear Regression	0.00475	0.00475	0.00476	0.00473	0.00473	0.00473	0.00424	0.00424	0.00425
GRU	0.00456	0.00457	0.00457	0.00456	0.00456	0.00457	0.00413	0.00415	0.00417
LSTM	0.00452	0.00452	0.00453	0.00455	0.00455	0.00455	0.00411	0.00413	0.00414
BiLSTM	0.00450	0.00450	0.00451	0.00451	0.00451	0.00451	0.00409	0.00411	0.00412
Clustering-based Stacked BiLSTM	**0.00447**	**0.00448**	**0.00450**	**0.00444**	**0.00445**	**0.00445**	**0.00405**	**0.00407**	**0.00409**

Here, the entries in bold shows the best performance of model

cluster workloads. In Fig. 12, we analyze the optimal size of clusters to be utilized for the proposed deep learning network model. With 3 clusters, the overall RMSE is minimal; hence, we set the cluster size to 3. The influence of all features (as discussed through Table 4) is investigated for out-of-sample CPU usage and memory usage predictions at 90 steps (15 min), 180 steps (30 min), and 270-steps-ahead (45 min). In Table 6, we present the RMSE of the multi-step ahead CPU and memory utilization predictions caused employing univariate, bivariate, and all features. From the table, it can be observed that the error produced by the multivariate models with all features is smaller than the error of univariate models. It is due to the fact that the multivariate methods analyze the correlations between the desired resource metric, and therefore, they can acquire the overall dynamics of the time series data. Moreover, the proposed prediction method performed bet-

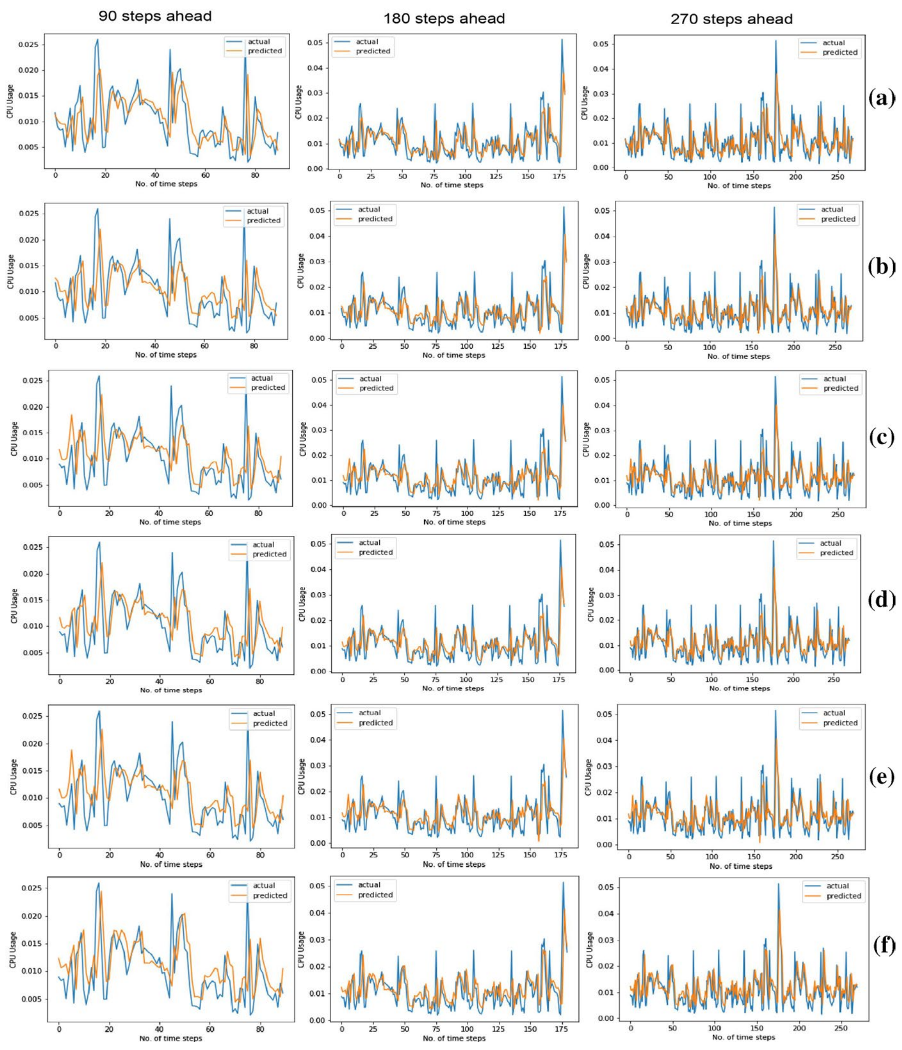


Fig. 13 Multi-steps-ahead CPU prediction with all features using **a** ARIMA, **b** Linear Regression, **c** GRU, **d** LSTM, **e** BiLSTM, **f** Clustering-based Stacked BiLSTM

ter than the other prediction models and generated minimum RMSE in all cases. Furthermore, in Figs. 13 and 14, we present the predictions of ARIMA, Linear regression, GRU, LSTM, BiLSTM, and proposed method for CPU and memory usage predictions, respectively, using all 12 features.

2. *Prediction performance using feature selection techniques* Figure 15 analyzes the Pearson and Spearman correlation schemes, Mutual information, Granger's causality, ANOVA, and Kendall's feature selection methods at different threshold values to distinguish the most suitable features from the given set. In the Pearson (Fig. 15a), we can see that features such as DIO, MAXC, MAXD, CPI, and MAI

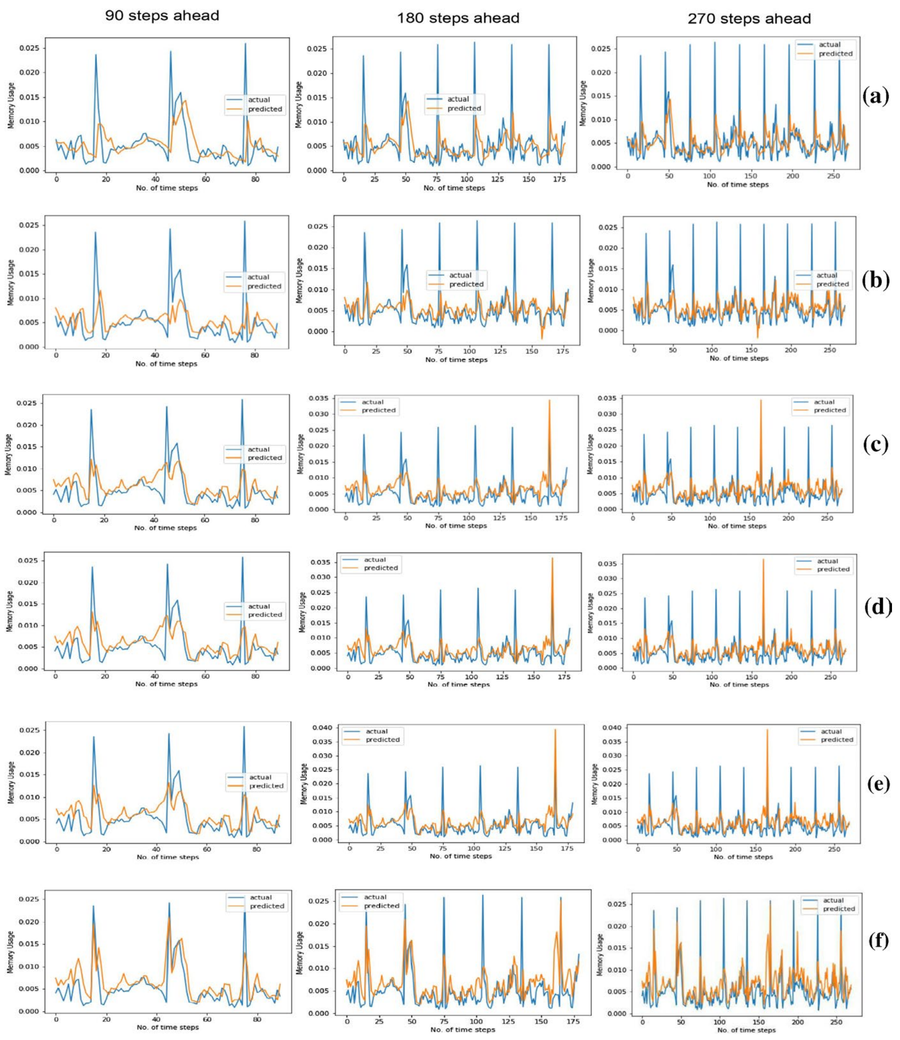


Fig. 14 Multi-steps-ahead memory prediction with all features using **a** ARIMA, **b** Linear Regression, **c** GRU, **d** LSTM, **e** BiLSTM, **f** Clustering-based Stacked BiLSTM

are clearly below the threshold and must not be selected as the prime features for training the model to predict the CPU usage. In the Spearman (Fig. 15b), values below the median rank such as MAXM, DIO, CPI, MAI are omitted from our selected features. In the Granger causality (Fig. 15c), after calculating the G-Test value for each of the resource metrics against that of our candid metric (CPU Rate), we may omit any feature whose G-Test value is lower than that of the threshold. In the Mutual information (Fig. 15d), values below the Median rate of mutual information are suggested to be removed for optimum result. In the ANOVA (Fig. 15e), values below the median rank such as MAXM, DIO, DSP,

Table 7 Selection of features using different feature selection methods

Method	Set of features										
	VM	MEM	UPC	MAXM	TPC	DIO	MAXC	DSP	MAI	CPI	MAXD
Kendall	✓	✓	✓	×	✓	×	✓	✓	×	×	✓
ANOVA	✓	✓	✓	×	✓	×	✓	×	×	×	✓
Pearson	✓	✓	✓	✓	✓	×	×	✓	×	×	×
Spearman	✓	✓	✓	×	✓	×	✓	✓	×	×	✓
Granger causality	×	✓	✓	×	✓	✓	×	✓	✓	×	×
Mutual information	✓	✓	✓	×	✓	✓	×	✓	✓	✓	×

CPI, and MAI are eliminated from the feature set. In Kendall's Tau (Fig. 15f), we have tested the feature selection with the nonparametric Tau B. Many of the methods we have encountered have clearly marked CPI and MAI as features with the least correlation to CPU utilization.

Table 7 presents the selection of features assuming a median *M* as a threshold to compare the various feature selection methods. A "Yes" entry in the table represents the selection of particular features, whereas a "No" entry shows the feature is not selected by the method. From the tabular results, it can be noted that applying Spearman, Kendall, and Pearson feature selection techniques produce almost an identical set of feature preferences. In Table 8, we represent the RMSE of all feature selection methods for multi-step ahead CPU and memory usage predictions. From the results, it can be perceived that multivariate models with a selected set of features deliver better performance than the models with all sets of features. The tabular results report that the feature selection approaches achieve better performance with other resource utilization prediction models. For example, CPU utilization prediction, ARIMA, and linear regression models perform better with the Kendall feature selection technique, while the GRU, LSTM, BiLSTM, and proposed clustering-based stacked BiLSTM model work better with Spearman correlation-based feature selection technique. Form memory usage prediction ARIMA model performs better with the Pearson feature selection technique. In comparison, the other models work better with Spearman correlation-based feature selection approach. Figures 16, 17, 18, 19, 20 and 21 depict resource usage prediction models' performance using the Pearson, Spearman, and Kendal methods, respectively, for different multivariate prediction models. We compare the prediction performance at a different step size of 90, 180, and 270 steps ahead. From the results, it is noted that with the increase in step-size, all methods' predictions degrade. It is due to the errors' accumulation for executing multiple steps ahead of out-of-sample predictions. It is also observed that the prediction techniques generate different predictions with different feature selection methods.

3. *SLA violations* In this experiment, we simulated many VMs and obtained the plots on different threshold values set within 30% to 100%. Figure 22a shows the SLA violations provoked by Sercon, MFFD, PUP-VMC, MBFD, ACS-VMC, and proposed methods at different threshold values. As a result, the proposed method

Table 8 RMSE of CPU and memory usage prediction for selected features in Google cluster trace

RMSE of CPU usage prediction

Model	Pearson			Spearman			Kendall		
	90 steps	180 steps	270 steps	90 steps	180 steps	270 steps	90 steps	180 steps	270 steps
ARIMA	0.00763	0.00774	0.00779	0.00763	0.00774	0.00778	**0.00761**	**0.00773**	**0.00776**
Linear Regression	0.00540	0.00544	0.00546	0.00539	0.00543	0.00549	**0.00538**	**0.00541**	**0.00546**
GRU	0.00536	0.00540	0.00543	**0.00535**	**0.00540**	**0.00542**	0.00535	0.00540	0.00543
LSTM	0.00535	0.00539	0.00541	**0.00533**	**0.00538**	**0.00540**	0.00534	0.00538	0.00540
BiLSTM	0.00534	0.00539	0.00541	**0.00532**	**0.00536**	**0.00538**	0.00533	0.00537	0.00539
Clustering-based stacked BiLSTM	0.00533	0.00537	0.00539	**0.00530**	**0.00535**	**0.00536**	0.00531	0.00536	0.00537

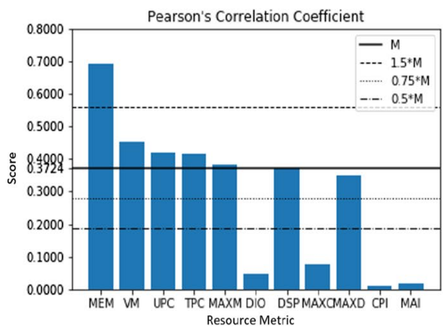
RMSE of memory usage prediction

Model	Pearson			Spearman			Kendall		
	90 steps	180 steps	270 steps	90 steps	180 steps	270 steps	90 steps	180 steps	270 steps
ARIMA	**0.00488**	**0.00488**	**0.00485**	0.00545	0.00546	0.00546	0.00535	0.00535	0.00536
Linear Regression	0.00434	0.00435	0.00435	**0.00430**	**0.00431**	**0.00431**	0.00431	0.00432	0.00432
GRU	0.00433	0.00434	0.00434	**0.00429**	**0.00428**	**0.00427**	0.00430	0.00430	0.00430
LSTM	0.00429	0.00430	0.00430	**0.00423**	**0.00424**	**0.00425**	0.00425	0.00425	0.00426
BiLSTM	0.00428	0.00429	0.00429	**0.00422**	**0.00423**	**0.00424**	0.00424	0.00424	0.00425
Clustering-based stacked BiLSTM	0.00426	0.00427	0.00428	**0.00422**	**0.00423**	**0.00423**	0.00423	0.00424	0.00424

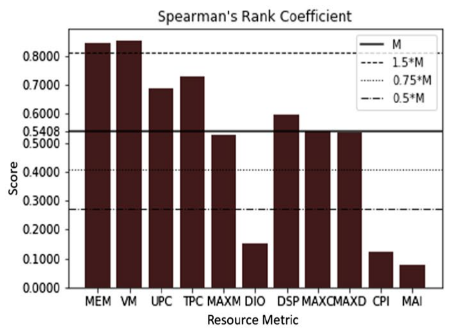
The bold entries represents the best performance of feature selection method in various networks

leads to fewer violations than the other existing approaches. The reason behind the high performance is that the proposed method applies prediction and assures that the destination PM does not create any hotspot whenever any VM is migrated on it. To avoid further SLA violations, the proposed method always tries to migrate VMs from the PMs that may create hotspots and predicted hotspots only. Thus, it prevents 100% usage of PMs and prevents the migration of VM instances to PMs, showing the potential of turning into hotspots by predicting the future utilization of PMs and VMs through the proposed prediction method.

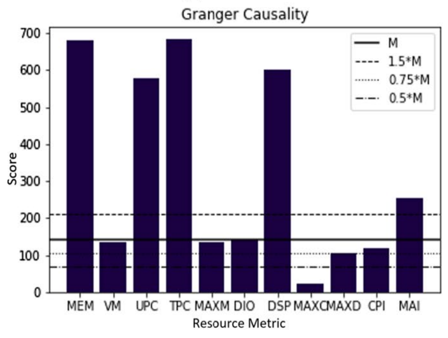
4. *Energy consumption analysis* PMs' energy consumption relies on the utilization of resources mainly memory, network bandwidth, CPU, etc. However, existing researches reveal that the CPU utilizes more power than other resources. Hence, we represent the PM's resource usage by its CPU usage. As compared to the



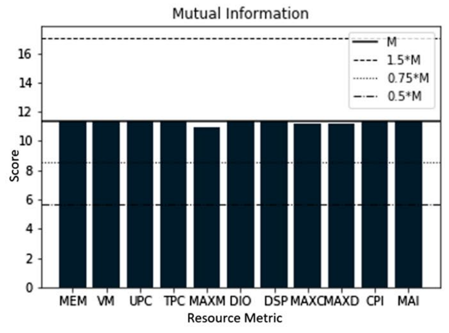
(a) Pearson



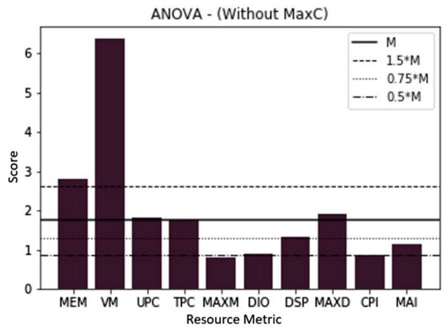
(b) Spearman



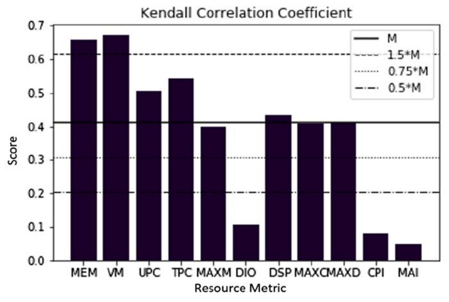
(c) Granger causality



(d) Mutual information



(e) ANOVA



(f) Kendall's Tau

Fig. 15 Feature selection with**a** Pearson's correlation coefficient **b** Spearman's correlation **c** Granger's causality **d** Mutual information **e** ANOVA and **f** Kendall's Tau

energy consumption caused by different algorithms, the suggested approach uses a lesser amount of energy, as represented in Fig. 22b. The reason being that the PMs having their usage lower than the threshold limit are switched to power-saving mode. The energy consumption of PMs is further optimized via packing the VMs into the most loaded PMs.

5. *Migration cost analysis* As displayed in Fig. 23a, the total migrations are less when compared with the existing algorithms. The depreciation of migrations due

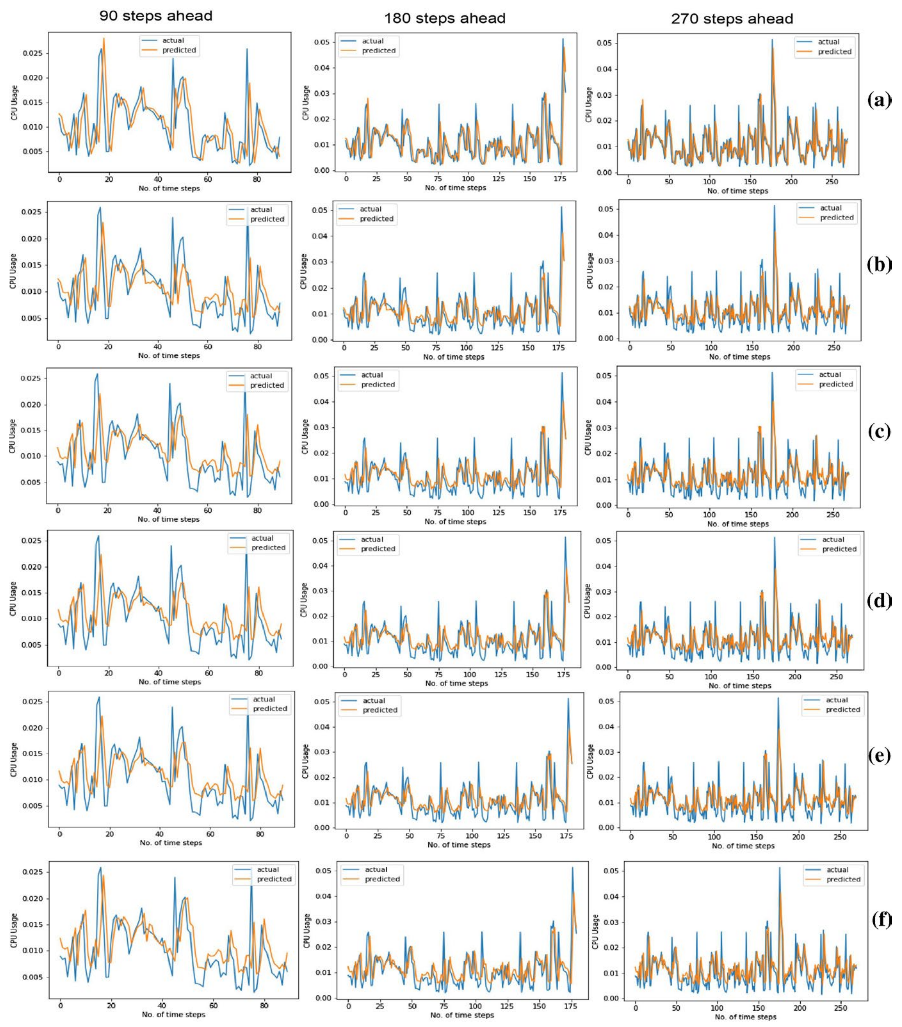


Fig. 16 Multi-steps-ahead CPU prediction with Pearson using **a** ARIMA, **b** Linear Regression, **c** GRU, **d** LSTM, **e** BiLSTM, **f** Clustering-based Stacked BiLSTM

to an explicit selection of VMs to mitigate hotspots and coldspots and prevent a VM's migration to a PM might result in the creation of new hotspots in the near future. Furthermore, the EQV-based VM selection increases energy savings and reduces the migration cost. The EQV metric always tries to assign VMs on PMs with less CPU utilization and smaller memory. Thus, it balances both the energy cost and migration cost.

6. *ESV* Figure 23b shows the performance comparison of ESV (product of Energy and SLAV) metric. The proposed scheme depicts significant improvement because it minimizes energy consumption and the SLAV violation rate. Thus, it maintains a trade-off between power cost and QoS guarantee.

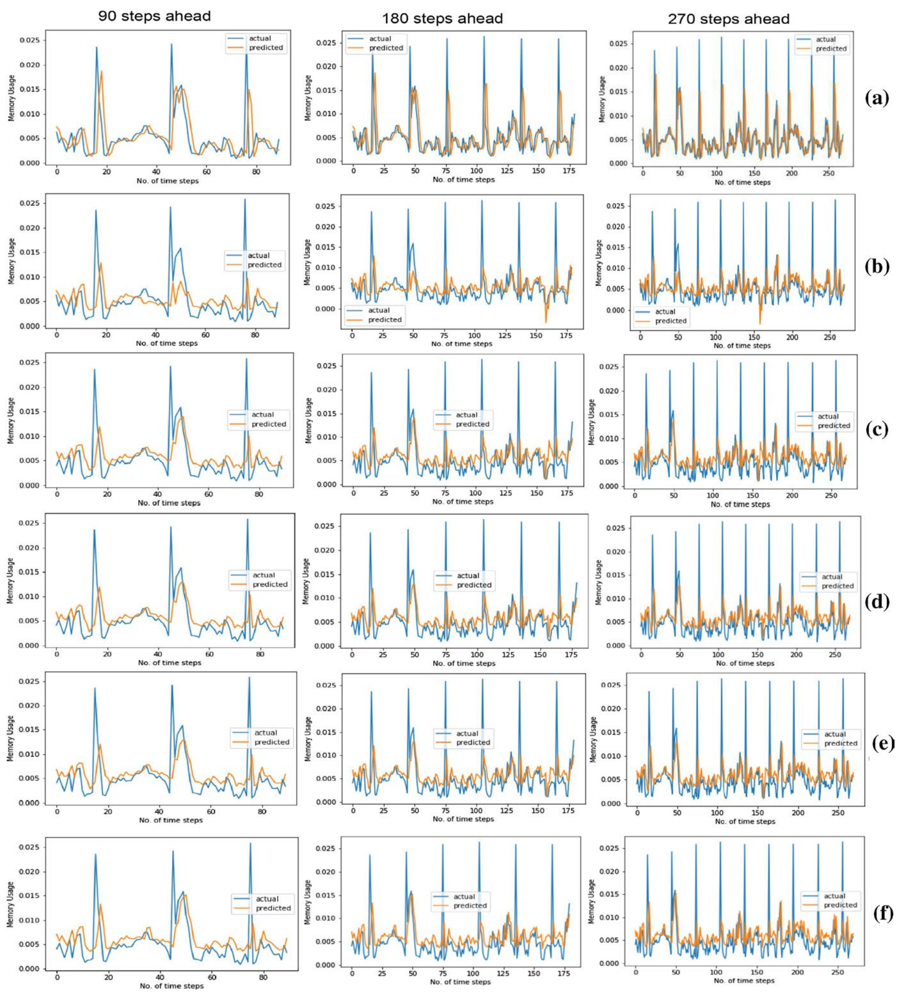


Fig. 17 Multi-steps-ahead memory prediction with Pearson using **a** ARIMA, **b** Linear Regression, **c** GRU, **d** LSTM, **e** BiLSTM, **f** Clustering-based Stacked BiLSTM

The results demonstrate that the proposed scheme remarkably outperforms the benchmarks in terms of prediction accuracy, energy consumption, migrations, SLAV, and ESV values.

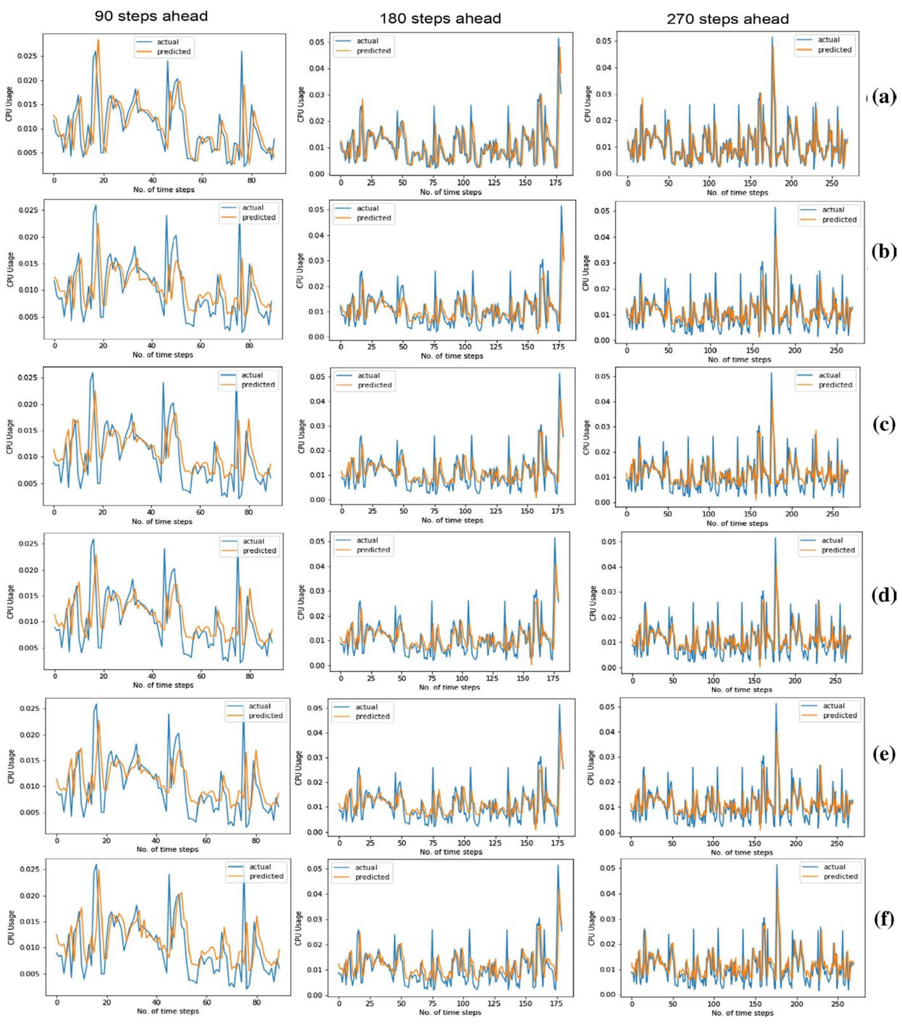


Fig. 18 Multi-steps-ahead CPU prediction with Spearman using **a** ARIMA, **b** Linear Regression, **c** GRU, **d** LSTM, **e** BiLSTM, **f** Clustering-based Stacked BiLSTM

7 Conclusions

This study presents deep learning-based multivariate resource utilization prediction for hotspots and coldspots mitigation in energy-efficient cloud data centers. We proposed a clustering-based stacked bidirectional LSTM deep learning network. We have divided our mitigation algorithm into different phases for hotspots and coldspots mitigation into various stages to generate an efficient mitigation plan. Firstly, it performs CPU and memory usage predictions using a deep learning model. Then, it considers both future and current resource usage to detect hotspots and coldspots formed in heterogeneous PMs. After their detection, we perform VM selection and

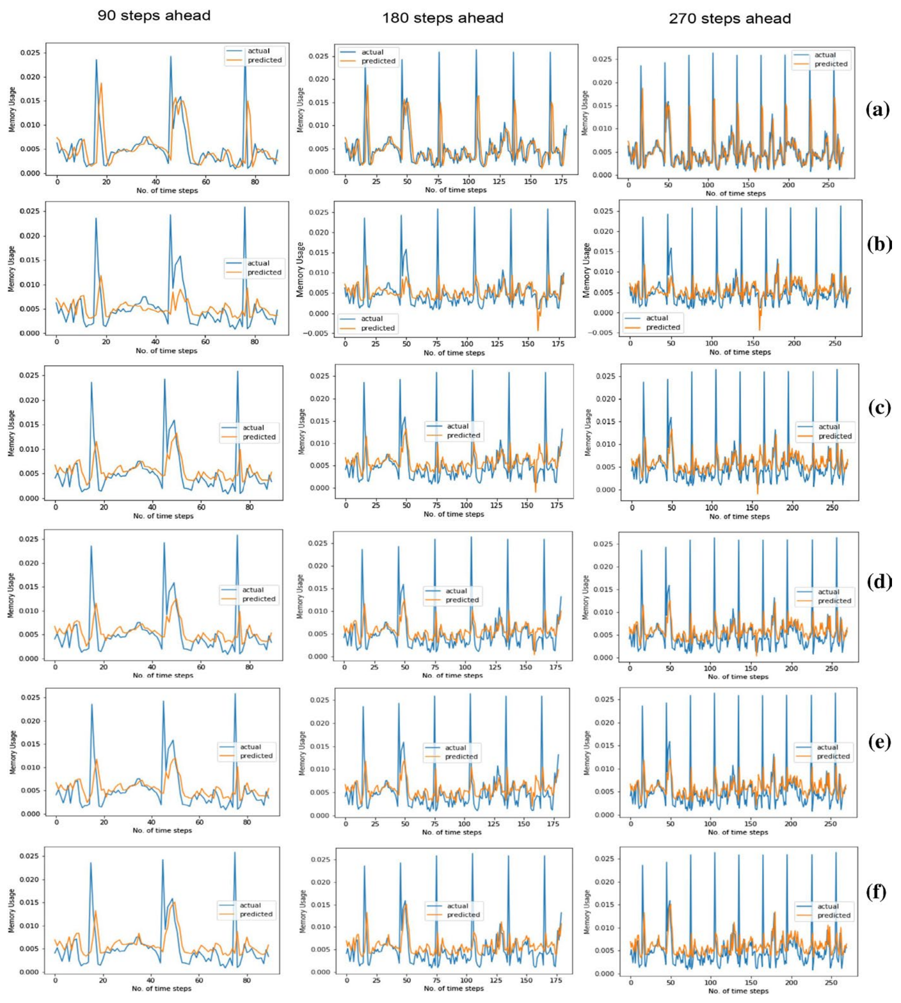


Fig. 19 Multi-steps-ahead memory prediction with Spearman using **a** ARIMA, **b** Linear Regression, **c** GRU, **d** LSTM, **e** BiLSTM, **f** Clustering-based Stacked BiLSTM

select the target PMs for VMs. Moreover, this study also presents trace-driven simulations using Google cluster workload traces. The obtained results show significant improvements over existing approaches concerning RMSE of prediction, energy-efficiency, migration cost, the number of actively used PMs, SLA violations, and ESV.

Future research might include considering network topology, communication cost, and implementation on the real-world cloud platform, such as OpenStack. Moreover, we have also observed some of the limitations of the proposed technique.

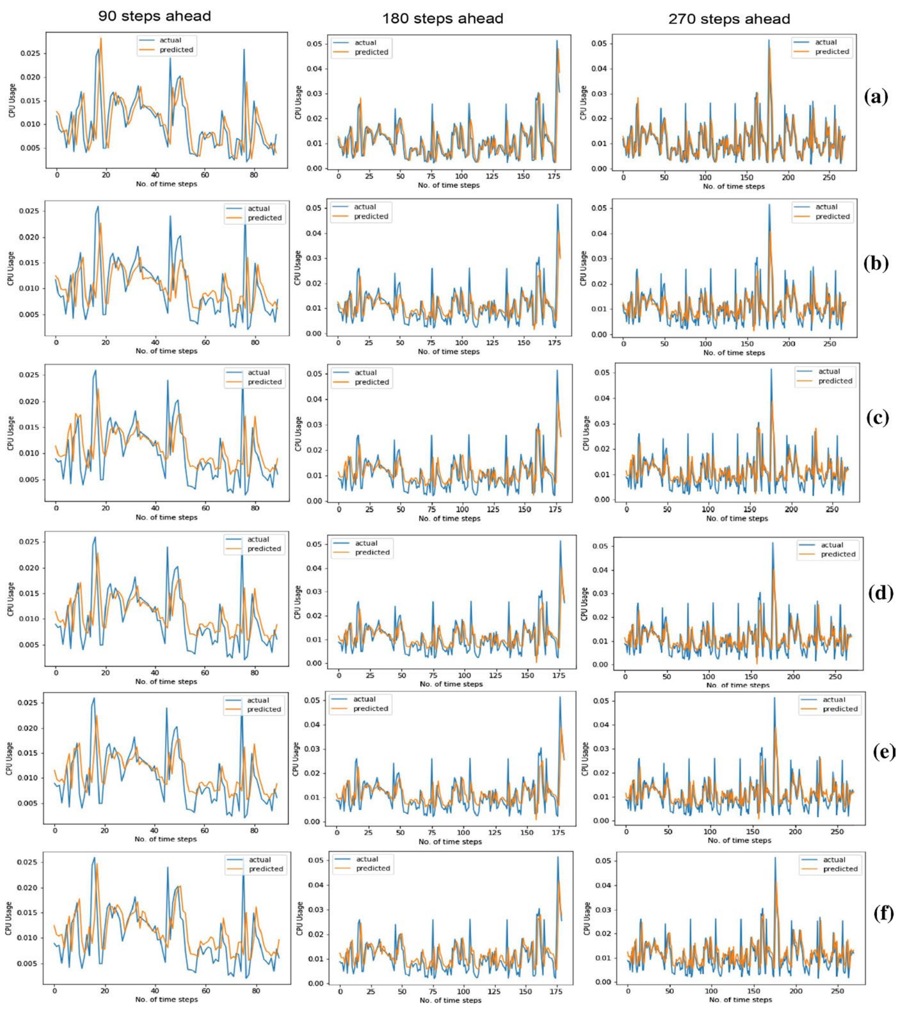


Fig. 20 Multi-steps-ahead CPU prediction with Kendall using **a** ARIMA, **b** Linear Regression, **c** GRU, **d** LSTM, **e** BiLSTM, **f** Clustering-based Stacked BiLSTM

For instance, the deep learning models are suitable for small- and medium-scale cloud data enters, but for large-scale data centers, global machine learning models are best suited for overall resource usage prediction of VMs. Also, the consideration of concurrent migrations and resource over-subscription could result in different variations in migration and energy cost analysis.

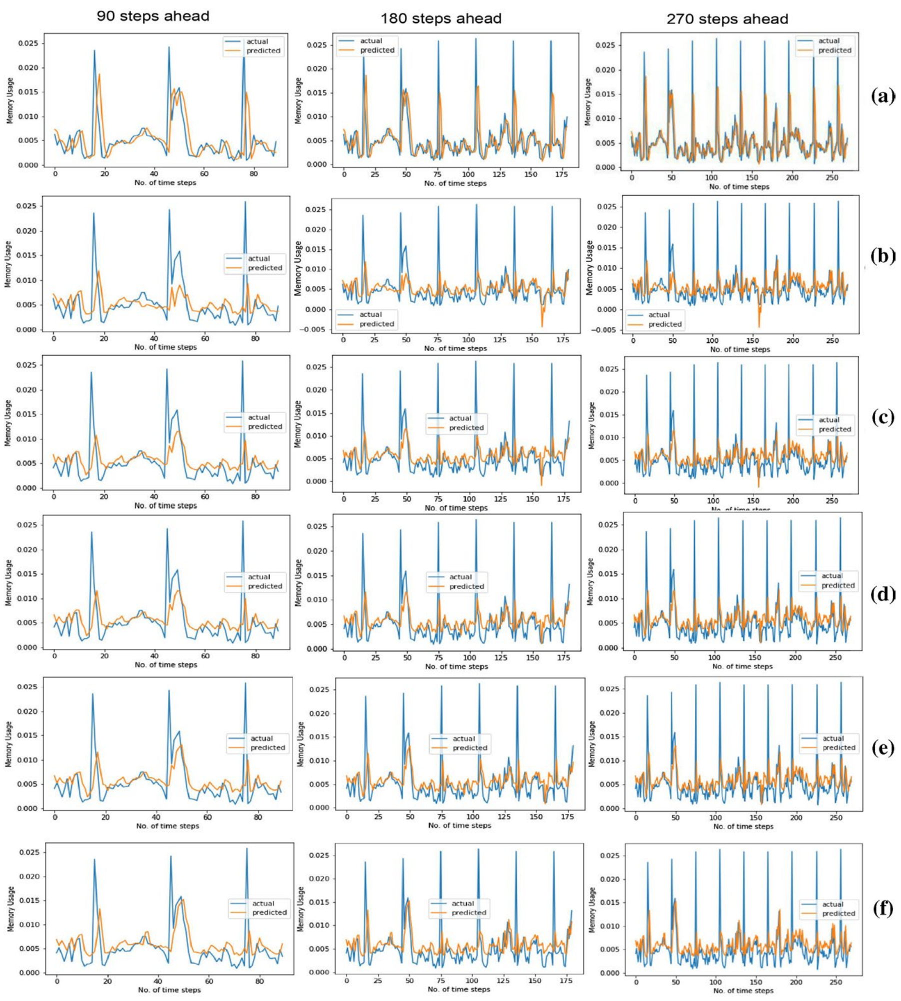
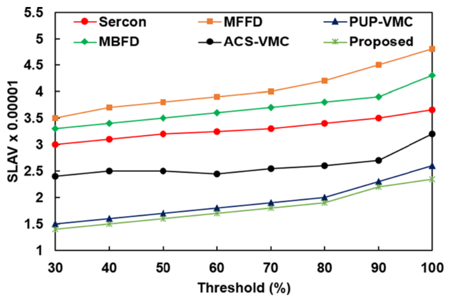
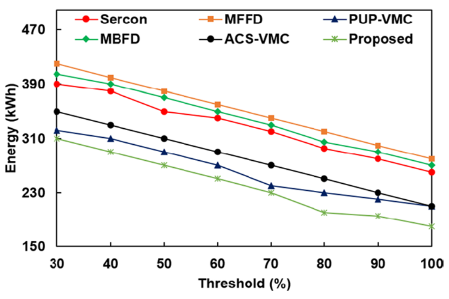


Fig. 21 Multi-steps-ahead memory prediction with Kendall using **a** ARIMA, **b** Linear Regression, **c** GRU, **d** LSTM, **e** BiLSTM, **f** Clustering-based Stacked BiLSTM

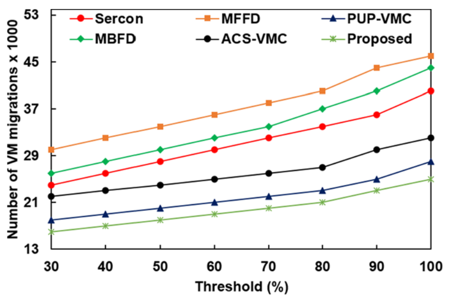


(**a**) Comparison of the SLAV metric

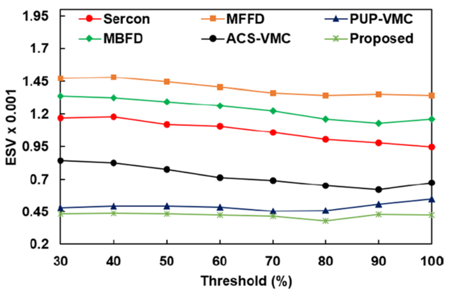


(**b**) Comparison of the energy consumption

Fig. 22 Comparison of **a** The SLAV metric **b** Energy consumption for GCD workload trace



(**a**) Migration comparison



(**b**) ESV metric comparison

Fig. 23 Analysis of **a** VM migrations, and **b** ESV metric

References

1. Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I (2009) Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. Future Gener Comput Syst 25(6):599–616
2. VMwareInc (2009) How VMware virtualization right sizes IT infrastructure to reduce power consumption. VMware Inc, Palo Alto, CA
3. Paul B, Boris D, Keir F, Steven H, Tim H, Alex H, Rolf N, Ian P, Andrew W (2003) Xen and the art of virtualization. SIGOPS Oper Syst Rev 37(5):164–177
4. Beloglazov A, Buyya R, Lee YC, Zomaya A (2011) A taxonomy and survey of energy-efficient data centers and cloud computing systems. Adv. Comput. 82:47–111
5. Asad Z, Chaudhry MAR (2017) A two-way street: green big data processing for a greener smart grid. IEEE Syst. J. 11(2):784–795
6. Gartner Inc (2007) Gartner estimates ICT industry accounts for 2 percent of global CO2 emissions. Gartner Press Release
7. Global warming: Data centres to consume three times as much energy in next decade, experts warn (2016) https://www.independent.co.uk/environment/global-warming-data-centres-consume-three-times-much-energy-next-decade-experts-warn-a6830086.html. Accessed 24 Aug 2020
8. How to stop data centres from gobbling up the world’s electricity (2018) https://www.nature.com/articles/d41586-018-06610-y. Accessed 27 Aug 2020
9. Barroso LA, Hölzle U (2007) The case for energy-proportional computing. Computer 12:33–37

10. Fan X, Weber WD, Barroso LA (2007) Power provisioning for a warehouse-sized computer. ACM SIGARCH Comput Archit News ACM 35:13–23
11. Christopher C (2005) Live migration of virtual machines. In: Proceedings of the 2nd conference on symposium on networked systems design & implementation. Berkeley, CA, USA, pp 273–286
12. Beloglazov A, Buyya R (2012) 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 24(13):1397–1420
13. Timothy W, Prashant S, Arun V, Mazin Y (2007) Black-box and gray-box strategies for virtual machine migration. In: Proc. 4th USENIX conference on networked systems design & implementation, NSDI'07, Cambridge, MA, USENIX Association, Berkeley, CA, USA, pp 229–242
14. Bin-packing (2006) In: Proc. combinatorial optimization, ser. Algorithms and combinatorics, vol 21. Springer Berlin Heidelberg, pp 426-441
15. Khan MA, Paplinski A, Khan AM, Murshed M, Buyya R (2018) Dynamic virtual machine consolidation algorithms for energy-efficient cloud resource management: a review. Sustainable cloud and energy services. Springer, New York, pp 135–165
16. Zhou Q, Xu M, Gill SS, Gao C, Tian W, Xu C, Buyya R (2020) Energy efficient algorithms based on VM consolidation for cloud computing: comparisons and evaluations. In: proc. 20th IEEE/ACM international symposium on cluster, cloud and internet computing (CCGRID). Melbourne, Australia, pp 489–498
17. Calheiros RN, Ranjan R, Beloglazov A, De Rose CAF, Buyya R (2011) CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw. Pract. Exp. 41(1):23–50
18. Farahnakian F, Liljeberg P, Plosila J (2013) LiRCUP: linear regression based CPU usage prediction algorithm for live migration of virtual machines in data centers. In: Proc. 39th euromicro conference on software engineering and advanced applications. Santander, pp 357–364
19. Farahnakian F, Pahikkala T, Liljeberg P, Plosila J (2013) Energy aware consolidation algorithm based on k-nearest neighbor regression for cloud data centers. In: Proc. IEEE/ACM 6th international conference on utility and cloud computing. Dresden, pp 256–259
20. Farahnakian F, Ashraf A, Pahikkala T, Liljeberg P, Plosila J, Porres I, Tenhunen H (2015) Using ant colony system to consolidate VMs for green cloud computing. IEEE Trans Ser Comput 8(2):187–198
21. Farahnakian F, Pahikkala T, Liljeberg P, Plosila J, Hieu NT, Tenhunen H (2019) Energy-aware VM consolidation in cloud data centers using utilization prediction model. IEEE Trans Cloud Comput 7(2):524–536
22. Moghaddam SM, O'Sullivan M, Walker C, Piraghaj SF, Unsworth CP (2020) Embedding individualized machine learning prediction models for energy efficient VM consolidation within Cloud data centers. Future Gener Comput Syst 106:221–233
23. Haghshenas K, Mohammadi S (2020) Prediction-based underutilized and destination host selection approaches for energy-efficient dynamic VM consolidation in data centers. J Supercomput 76:10240–10257
24. Hsieh S-Y, Liu C-S, Buyya R, Zomaya AY (2020) Utilization-prediction-aware virtual machine consolidation approach for energy-efficient cloud data centers. J Parallel Distrib Comput 139:99–109
25. Tarafdar A, Debnath M, Khatua S et al (2020) Energy and quality of service-aware virtual machine consolidation in a cloud data center. J Supercomput. 76:9095–9126
26. Lu S, Chen J (2018) Host overloading detection based on EWMA algorithm in cloud computing environment. In: Proc. IEEE 15th international conference on e-business engineering (ICEBE). Xi'An, China, pp 274–279
27. Donnell NM, Howley E, Duggan J (2020) Dynamic virtual machine consolidation using a multi-agent system to optimise energy efficiency in cloud computing. Future Gener Comput Syst 108:288–301
28. Monshizadeh Naeen H, Zeinali E, Toroghi Haghighat A (2020) Adaptive Markov-based approach for dynamic virtual machine consolidation in cloud data centers with quality-of-service constraints. Softw Pract Exp 50:161–183
29. Li Z, Xinrong Y, Lei Y, Guo S, Chang V (2020) Energy-efficient and quality-aware VM consolidation method. Future Gener Comput Syst 102:789–809
30. El-Moursy A, Abdelsamea A, Kamran R, Saad M (2019) Multi-dimensional regression host utilization algorithm (MDRHU) for host overload detection in cloud computing. J Cloud Comput 8:1–17

31. Ranjbari M, Torkestani JA (2018) A learning automata-based algorithm for energy and SLA efficient consolidation of virtual machines in cloud data centers. J Parallel Distrib Comput 113:55–62
32. Subramanian S, Kannammal A (2019) Real time non-linear cloud workload forecasting using the holt-winter model. In: Proc. 10th international conference on computing, communication and networking technologies (ICCCNT). Kanpur, India, pp 1–6
33. Qi Z, Mohamed FZ, Shuo Z, Quanyan Z, Raouf B, Joseph LH (2012) Dynamic energy-aware capacity provisioning for cloud computing environments. In: Proc. international conference on autonomic computing (ICAC). ACM, New York, NY, USA, pp 145–154
34. Podolskiy V, Jindal A, Gerndt M, Oleynik Y (2018) Forecasting models for self-adaptive cloud applications: a comparative study. In: Proc. IEEE 12th international conference on self-adaptive and self-organizing systems (SASO). Trento, Italy, pp 40–49
35. Caglar F, Gokhale A (2014) iOverbook: intelligent resource overbooking to support soft real-time applications in the cloud. In: Proc. IEEE 7th international conference on cloud computing (CLOUD). IEEE, Anchorage, AK, USA, pp 538–545
36. Gong Z, Gu X, Wilkes J (2010) PRESS: PRedictive elastic resource scaling for cloud systems. In: Proc. international conference on network and service management (CNSM). IEEE, Niagara Falls, ON, Canada, pp 9–16
37. Nguyen H, Shen Z, Gu X, Subbiah S, Wilkes J (2013) AGILE: elastic distributed resource scaling for infrastructure-as-a-service. In: Proc. 10th international conference on autonomic computing (ICAC). San Jose, CA, USENIX, pp 69–82
38. Ghorbani M, Wang Y, Xue Y, Pedram M, Bogdan P (2014) Prediction and control of bursty cloud workloads: a fractal framework. In: Proc. international conference on hardware/software codesign and system synthesis (CODES+ISSS), pp 1–9
39. Reiss C, Wilkes J, Hellerstein JL (2011) Google cluster-usage traces: format + schema. https://github.com/google/cluster-data
40. Song B, Yu Y, Zhou Y et al (2018) Host load prediction with long short-term memory in cloud computing. J Supercomput. 74:6554–6568
41. Chakraborty K, Mehrotra K, Mohan CK, Ranka S (1992) Forecasting the behavior of multivariate time series using neural networks. Neural Netw. 5(6):961–970
42. Aboagye-Sarfo P et al (2015) A comparison of multivariate and univariate time series approaches to modelling and forecasting emergency department demand in Western Australia. J Biomed Inf 57:62–73
43. Dannecker L (2015) Energy time series forecasting: efficient and accurate forecasting of evolving time series from the energy domain, 1st edn. Springer, Berlin
44. Sun Y, Li J, Liu J, Chow C, Sun B, Wang R (2015) Using causal discovery for feature selection in multivariate numerical time series. Mach Learn 101(1–3):377–395
45. Zhao Y, Ye L, Li Z, Song X, Lang Y, Su J (2016) A novel bidirectional mechanism based on time series model for wind power forecasting. Appl Energy 177:793–803
46. Wakuya H, Shida K (2001) Bi-directionalization of neural computing architecture for time series prediction. III. Application to laser intensity time record Data Set A. In: Proc. international joint conference on neural networks (IJCNN). IEEE, Washington, DC, USA, pp 2098–2103
47. Cui Z, Ke R, Pu Z, Wang Y (2020) Stacked bidirectional and unidirectional LSTM recurrent neural network for forecasting network-wide traffic state with missing values. Transp Res Part C Emerg Technol 118:102674
48. Gupta S, Dinesh DA (2017) Resource usage prediction of cloud workloads using deep bidirectional long short term memory networks. In: Proc. international conference on advanced networks and telecommunications systems (ANTS). IEEE, Bhubaneswar, pp 1–6
49. Gupta S, Dileep AD, Gonsalves TA (2018) A joint feature selection framework for multivariate resource usage prediction in cloud servers using stability and prediction performance. J Supercomput 74(11):6033–6068
50. Beloglazov A, Abawajy J, Buyya R (2012) Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future Gener Comput Syst 28(5):755–768
51. Standard Performance Evaluation Corporation (SPEC) (2008), http://www.spec.org
52. Blackburn M, Grid G (2008) Five ways to reduce data center server power consumption (white paper). The Green Grid
53. Boutaba R, Zhang Q, Zhani MF (2014) Virtual machine migration in cloud computing environments: benefits, challenges, and approaches. Communication infrastructures for cloud computing. IGI Global, Pennsylvania, pp 383–408

54. Zakarya M, Gillam L (2019) Managing energy, performance and cost in large scale heterogeneous datacenters using migrations. Future Gener Comput Syst. 93:529–547
55. Schuster M, Paliwal KK (1997) Bidirectional recurrent neural networks. IEEE Trans Signal Process 45(11):2673–2681
56. Wang M, Meng X, Zhang L (2011) Consolidating virtual machines with dynamic bandwidth demand in data centers. In: 2011 Proceedings IEEE INFOCOM, vol 201. pp 71–75
57. Behera S, Misra R, Sillitti A (2021) Multiscale deep bidirectional gated recurrent neural networks based prognostic method for complex non-linear degradation systems. Inf Sci 554:120–144
58. Murtazaev Aziz, Sangyoon Oh (2011) Sercon: server consolidation algorithm using live migration of virtual machines for green computing. IETE Tech Rev 28(3):212–231

Authors and Affiliations

Yashwant Singh Patel[1] · Rishabh Jaiswal[2] · Rajiv Misra[3]

Rishabh Jaiswal
rsj.rishabh@gmail.com

Rajiv Misra
rajivm@iitp.ac.in

[1] Department of Computer Science and Engineering, Thapar Institute of Engineering and Technology, Patiala, Punjab, India

[2] Department of Computer and Information Science and Engineering, University of Florida, 432 Newell Dr, Gainesville, FL 32611, USA

[3] Department of Computer Science and Engineering, Indian Institute of Technology Patna, Bihar, India