Cluster Computing
https://doi.org/10.1007/s10586-023-04001-1

A machine learning-based optimization approach for pre-copy live virtual machine migration

Raseena M. Haris[1] · Khaled M. Khan[1] · Armstrong Nhlabatsi[1] · Mahmoud Barhamgi[1]

Received: 4 February 2023 / Revised: 4 March 2023 / Accepted: 30 March 2023

Abstract
Organizations widely use cloud computing to outsource their computing needs. One crucial issue of cloud computing is that services must be available to clients at all times. However, the cloud services may be temporarily unavailable due to maintenance of the cloud infrastructure, load balancing of services, defense against cyber attacks, power management, proactive fault tolerance, or resource usage. The unavailability of cloud services impacts negatively on the business model of cloud providers. One solution to tackle the service unavailability is Live Virtual Machine Migration (LVM), that is, moving virtual machines (VMs) from the source host machine to the destination host without disrupting the running application. Pre-copy memory migration is a common LVM approach used in most networked systems such as the cloud. The main difficulty with this approach is the high rate of frequently updating memory pages, referred to as "dirty pages. Transferring these updated or dirty pages during the pre-copy migration approach prolongs the total migration time. After a predefined iteration, the pre-copy approach enters the stop-and-copy phase and transfers the remaining memory pages. If the remaining pages are huge, the downtime or service unavailability will be very high -resulting in a negative impact on the availability of the running services. To minimize such service downtime, it is critical to find an optimal time to migrate a virtual machine in the pre-copy approach. To address the issue, this paper proposes a machine learning-based method to optimize pre-copy migration. It has mainly three stages (i) Feature selection (ii) Model generation and (iii) Application of the proposed model in pre-copy migration. The experiment results show that our proposed model outperforms other machine learning models in terms of prediction accuracy and it significantly reduces downtime or service unavailability during the migration process.

Keywords Cloud computing · Live virtual machine migration · Pre-copy · Dirty page rate · Machine learning · Feature selection

1 Introduction

Virtualization [1, 2] enables cloud computing to create and run multiple virtual machines (VMs) on the same physical server at the same time. Virtual Machines (VMs) are the virtualization of the computing layer of data center resources, allowing physical servers, CPU, cache, memory, and other hardware to be shared by several VMs. Virtualization is one of the most cost-effective hardware and energy-saving approaches deployed by cloud providers.

Due to the high demand for cloud computing, services must be available without interruption. From time to time, cloud services need routine or emergency system maintenance that involves temporarily suspending or taking services offline. An absence of continuous services may have

✉ Raseena M. Haris
rm1704837@qu.edu.qa

Khaled M. Khan
k.khan@qu.edu.qa

Armstrong Nhlabatsi
armstrong.nhlabatsi@qu.edu.qa

Mahmoud Barhamgi
mbarhamgi@qu.edu.qa

[1] Department of Computer Science and Engineering, Qatar University, Doha, Qatar

a negative impact on clients. Live virtual machine migration; virtual machines are migrated or relocated from one physical host to another without impacting the running applications [3, 4]. The purpose of the live machine migration is to address issues related to fault tolerance, load balancing, maintenance, tackling cyber-attacks, etc.

There are mainly three types of live virtual machine migration: pre-copy, post-copy, and hybrid. At the initial stage of the pre-copy migration, the complete memory content is copied from the source to the destination. The updated or dirty memory pages from the previous iteration are then transferred to the destination host in the subsequent iteration until a predefined stopping condition is satisfied. The VM from the source is stopped when the stopping condition is met, and the remaining memory pages and CPU states are copied to the destination host. Then the VM resumes execution at the destination host. In contrast to pre-copy, post-copy [5] suspends VM activity on the source host and transfers the minimum required processor states to the destination, which are required to run the VM. After the memory pages are copied from source to destination via page requests, active pushing, or pre-paging after the VM is executed at the destination. This process is repeated until the destination machine has received all of the pages. Hybrid [6] as the combination of pre-copy and post-copy approaches. To reduce the number of page faults/network faults, it initially copies memory data with a minimum number of iterations in a pre-copy manner. Then the migration process transfers the VM execution to the destination server, and the remaining pages will be copied in a post-copy manner.

In this paper, we primarily focus on the optimization of pre-copy migration. Unless there are stop criteria, the iterative pre-copy stage can continue indefinitely. As a result, defining stop conditions is crucial to completing this step on schedule and efficiently. These requirements vary depending on the hypervisor and the live migration subsystem design. But they are generally intended to limit the amount of data moved between physical hosts while minimizing VM downtime. For example, in the Xen pre-copy migration [3, 7–9] the stopping conditions are: (i) During the last pre-copy iteration, less than 50 pages were dirty; (ii) There have been 29 pre-copy iterations; and (iii) The total amount of RAM allocated to the VM has been copied to the destination host more than three times. The first condition ensures minimal downtime because only a few pages need to be transferred. On the other hand, the other two conditions force migration into the stop-and-copy phase, which may still require numerous updated pages to be moved across, resulting in significant downtime. These predefined stopping conditions significantly impact migration performance and may result in non-linear trends in overall migration time and VM downtime.

The other parameters that influence the performance of pre-copy migration are VM size, network bandwidth, working set size, and dirty page rate [3, 7, 10–13]. The migration may take too long or even fail in some cases due to a high dirty page rate and a low network transmission rate. So the key obstacles to minimizing downtime and total migration time during the pre-copy migration are the varying rates of dirty pages in each iteration, memory page size, different workloads running on the VM, size of the VM, available bandwidth, and the predefined stopping condition.

Some analytical models such as in [10, 14, 15], and probabilistic models in [7, 16, 17] have already been proposed for predicting the downtime and total migration time of the pre-copy algorithm using several parameters. However, these models do not achieve good prediction accuracy due to the many parameters used in the models. To overcome the problems in analytical and probabilistic models, some machine learning-based models have been proposed for predicting the performance parameters of different migration algorithms [18–20]. To forecast the performance parameters of different migration algorithms, this research selected many input features without considering the most relevant features for the migration algorithm to compute. Input feature selection is essential in machine learning because it affects the model's prediction accuracy. Building a model with fewer features can also reduce the complexity in terms of space and time. Therefore it is crucial to find out a machine-learning model with relevant features to determine the optimal downtime for live virtual machine migration. The main objective of our paper is to develop a machine learning-based pre-copy optimization method with a set of significantly fewer input features.

The main contributions of the paper are:

- *A feature selection algorithm:* We developed an algorithm to identify the set of relevant features that influence migration performance, thereby reducing computational overhead and enhancing learning accuracy.
- *A KNN-based model to predict the optimal time for live migration:* We developed a machine-learning model using identified features to predict the optimal time for pre-copy migration, with high accuracy and adaptability.
- *Validation through a case study:* We evaluated the proposed model's prediction accuracy using a case study, obtained results show an error rate of less than 5%.
- *Application of the model in pre-copy migration:* We proposed a machine learning-based method for optimizing pre-copy migration, reducing downtime by 36% compared to existing algorithms.

The remainder of this paper is organized as follows: Sect. 2 discusses the background and related works. Section 3 describes the overview of the approach. Section 3.1 describes feature selection. Section 4 describes a machine learning model to determine the optimal time for VM migration. Evaluation of the proposed model is outlined in Sect. 5. Section 7 concludes this paper with some pointers to further research.

2 Background and related work

This section explains the preliminaries of the topics and approaches related to live virtual machine migration presented in this paper.

2.1 Live virtual machine migration

Virtual Machine (VM) Migration is the process of moving a running virtual machine [21] from a physical host to other physical machines without disconnecting the client or the application. The virtual machine's memory, storage, and network connectivity are transferred from the source machine to the destination machine. The simplest way to migrate a virtual machine is to shut down the source computer and move the whole state from the source to the destination machine. After completing a successful migration, the VM resumes at the destination machine. But this stop-and-copy technique interrupts client activity and cloud services for a long time and is impractical for all application environments. This is not a good option for cloud providers from a business perspective. To minimize downtime, the most commonly used approach is migrating VMs while they are running [3, 4, 22, 23].

In pre-copy migration, the total migration time and downtime are two important metrics that are often used to evaluate the effectiveness of the migration process. These are the following:

- *Total migration time (TMT):* Total migration time [24] refers to the elapsed time between the initiation of the migration process and the final switch over of the VM to the destination server. This metric is crucial because it determines the length of time during which the VM is unavailable to its users. The longer the total migration time, the more likely it is that users will experience disruptions or delays in their work, which can lead to dissatisfaction, lost productivity, and even financial losses. Therefore, minimizing total migration time is a key goal of any pre-copy migration strategy.
- *Downtime (DT):* Downtime [3] refers to the period of time during which the VM is completely unavailable to its users, either because it is still running on the source server or because it has not yet fully started up on the destination server. Downtime is a particularly important metric in live environments where VMs must remain operational to support mission-critical applications or services. Any disruption to the VM's availability during the migration process can cause serious problems, such as data loss, service interruption, or system crashes. By monitoring and minimizing downtime during pre-copy migration, businesses can ensure that the migration process does not negatively impact their operations or customer experience. As a result, minimizing downtime is a primary objective in precopy migration.

Overall, both total migration time and downtime are important metrics to monitor during pre-copy migration, as they provide valuable insights into the efficiency and effectiveness of the migration process, as well as its impact on business operations.

2.2 Machine learning algorithms

In the last few years, machine learning [18, 19, 25–27] Patel 2016 machine, Jo 2017 machine) has been widely used for accurately predicting the performance parameters of different migration algorithms. In the research reported in this paper, we use some machine learning algorithms to find the optimal time for migration. These are briefly introduced in this section.

Regression is a standard statistical approach for finding out the relationship between one or more input variables to the output variable. Simple regression contains only one input variable, whereas multiple regression has two or more input variables. The regression function can be linear or non-linear. Linear regression [19, 28, 29] is a simple regression approach that uses a straight line to fit the given data with the least amount of error. If the dataset and the output value have a clear linear relationship, then linear regression is a good option.

In *non-linear regression*, observational data are represented by a function that is a nonlinear combination of model parameters and is dependent on one or more independent variables. *Support Vector Regression (SVR)* [30, 31] is a non-linear regression technique for predicting a target value from input features. To improve the model performance, parameter tuning is an effective approach in machine learning algorithms. The important tuning parameters in SVR are 'kernel', 'gamma', and 'C'. Kernel parameters are 'rbf', 'poly', 'sigmoid', and 'linear. Bagging, also known as *Bootstrap Aggregation*, creates numerous submodels from a portion of the whole dataset and then overfits the model to the dataset. The average prediction of all submodels is utilized as the final value after submodel training.

The use of labeled datasets to train algorithms for accurately identifying data or predicting outcomes is known as supervised learning [32]. *K-Nearest Neighbors (KNN)* [19, 33–35] is a supervised learning. It is simple, more popular, and can be used both in regression and classification. It was first proposed by Fix et al. [36]. KNN algorithm's working is based on finding the **K**(K=1,2,3,4,.. n) nearest neighbors in input training data of *n* examples for a specific query instance. Different distance metrics have been utilized to compute the nearest neighbors in the KNN method. Euclidean distance, Manhattan distance, Minkowski distance, and Hamming distance are the popular distance metrics used in the KNN algorithm. Selecting a specific distance metric and number of neighbors for training data can be achieved by optimizing the hyperparameter of the KNN algorithm using the input training data. The main steps for the KNN algorithm are: (i) For a test example **i**, compute the distance from **i** to all the training examples; (ii) Find the k-nearest training examples of **i**; (iii) Compute the mean of the numerical target (value) of k-nearest neighbors to determine the numerical target of test example **i**.

Artificial Neural Networks (ANN) [37] are made up of layers of neurons. These neurons are the core processing units of the network. Each of these consists of an input layer that takes the input to the model, an output layer for predicting the final output, and in between, there are hidden layers that perform most of the computation required by the network. Neurons in one layer communicate with neurons in the other layer via channels. A weight value is assigned to each channel. The inputs are multiplied by the weight value assigned to them, and the result is the hidden layer's input value. The activation function is the sum of the hidden values associated with each neuron in the hidden layer, which is added to the preceding sum value of the input layer neurons. It determines whether or not a specific neuron is active. This activated neuron transmits data across the channel to the next neuron in the hidden layer is called forward propagation. Data is propagated over the network and higher-valued neurons in the hidden layer fire to the output neuron. Then the predicted output is compared with the actual value to find out the error. If the error is high, then this information is sent backward to the neurons; this is called back-propagation. Based on this information, the weights are adjusted. This process continues until the neurons predict the value more accurately. The expected output is then compared to the actual result to determine the degree of inaccuracy. If the error is high, the information is sent back to the neurons, a process known as backward propagation. The weights are adjusted based on this information. This process is repeated until the neurons can more precisely forecast the value.

2.3 Related work

Several research works have been reported on live migration and optimization of this. Some key research works are discussed in this subsection. Sherif Akoush et al. [7] proposed two simulation models: AVG (average page dirty rate) and HIST (history-based page dirty rate) for predicting the performance (total migration and downtime) of pre-copy migration to within 90% accuracy in both synthetic and real-world benchmarks. The AVG model is used to predict the migration performance of a VM with a constant memory dirtying rate. In contrast, the HIST model is used to predict the migration performance of a VM with identical memory characteristics across different workloads. The work also classified the parameters as static (i.e., memory size, VM resumption time) and dynamic (bandwidth, dirty page rate) based on their impact on migration performance. However, they did not consider some critical features, such as working set size, that impact migration performance. This prediction model is also only applicable to the LAN environment.

Nathan et al. [10] proposed an analytic model to predict the total migration time, the downtime, and the total traffic of a live migration after analyzing the problems in different existing analytical models [7, 38–44]. Due to the large number of factors that need to be considered, extending these analytic models to different methodologies or metrics is impracticable.

Hundreds of servers are used in modern data centers to service millions of clients worldwide. Computers in a data center create a large amount of data from VM performance logs and hardware sensors. This expands the scope of data center management solutions. Machine learning is a powerful tool to automatically generate models for various metrics and live migration techniques using data collected from data centers. Using 200,000 training samples collected over two years in Google data centers, Ferdaus et al. [45] proposed a machine learning model to forecast the power usage effectiveness of data centers. The model takes into account 43 different input factors. Creating an analytical model with that many parameters would be impossible. An analytical study of the performance of live migration based on different states of the virtual machine and the underlying physical host is less suitable. If there are n live virtual machine migration algorithms and m performance metrics, creating $n * m$ models with each set of parameters is difficult for the analytical model. This structure also makes it simple to add new algorithms or measurements.

Several studies have addressed the challenge of VM live migration in a data center. Machine learning is a sophisticated tool for solving complex issues in real-world scenarios using data. Because the intricacy of the site's operation and the volume of available monitoring data are both great, it's a well-suited solution for the data center environment. Scientists have recently deployed machine learning-based models to handle challenges in the live migration process [19, 20, 46–49].

The work in [50] proposed a Working Set Prediction using Machine Learning approaches (WSPML) to reduce the total migration time during the migration process. Experimentally, they showed that the M5 model tree (M5P) provides a more accurate result than linear regression for different workload types and varying network bandwidth. They concluded that WSPML reduces overall migration time more than the traditional pre-copy approach. The critical disadvantage of this prediction model is that it only predicts memory pages that will be required in the near future as a working set rather than frequently updated memory pages during the migration process. In addition, they only consider the input features of page dirty rate and transmission rate. Furthermore, this approach is ineffective in predicting the working set when the workload changes.

Nehra et al. [51] proposed a Support Vector Regression (SVR) based methodology to predict host utilization in the cloud environment with input features such as CPU, memory, and bandwidth usage. They proposed a radial basis function and a polynomial kernel function for accurate prediction. The numerical findings indicate that the proposed model's accuracy is better than other models. This model is applicable only for predicting host utilization, not live migration performance.

To predict CPU utilization and network bandwidth usage for live virtual machine migration, Duggan et al. [52] used an artificial neural network (ANN) and proposed a multi-time-ahead prediction model. The model aims to improve the performance of the data center by minimizing bandwidth utilization. Experimentally, they showed that the proposed methodology reduces bandwidth utilization during critical times and improves the data center's overall efficiency. This model is applicable for predicting the CPU utilization and network bandwidth for live virtual machine migration, but not for predicting performance parameters such as total migration time and downtime for the pre-copy approach.

An ML-based technique has been suggested in the paper [18] to automatically generate reliable models that can predict essential performance parameters of VM live migration under various resource restrictions and workloads for all generally accessible migration algorithms. They examined various supervised techniques for modeling an adaptive process in order to determine the best policies for migrating virtual machines (VMs) between hosts while meeting service level agreement (SLA) requirements. The results of their experiments revealed a considerable improvement in migration performance. They have shown that the suggested model outperforms existing work by a factor of 2-5 when compared to the state-of-the-art. However, without considering the critical features of each migration algorithm, they used all the input features included in the dataset to predict the performance metrics of all five live migration algorithms. Alrajeh et al. [53] employed three supervised learning algorithms to develop prediction models for VM live migration decision-making to determine which VMs could be migrated or not. The techniques used were stochastic gradient boosted, random forest, and bagging tree. The results of this analysis show that some VMs can be relocated in a short amount of time, while others can be migrated over a long period, and some cannot be transferred while the workload is running. However, to build this model, they do not consider the different job scheduling algorithms with other workloads to identify which job types are running.

Arif et al. [25] proposed a machine learning-based downtime optimization (MLDO) approach to reduce downtime during live migration over wide area networks based on predictive mechanisms for standard workloads. They compared the proposed technique with existing strategies and observed improvements of up to 15%. This prediction model is only applicable for migration over the WAN environment. Hassan et al. [54] proposed a two-step model based on local regression to predict SLA violation. For migration decisions, different classification algorithms such as support vector machine (SVM) and K-nearest neighbors (KNN) are suggested considering the input features of CPU usage, inter-VM bandwidth usage, and memory usage. In comparison to SVM and KNN, the obtained results demonstrated the importance of regression trees in terms of accuracy. This approach is primarily intended for applications with strict SLA requirements.

Motaki1 et al. [19] proposed an ML model for predicting six live migration performance metrics for each live migration algorithm. They proved that the proposed model reduces the service level agreement violation rate by 31% and 60% and considers CPU time requirements. The input features that affect the particular migration must be considered while building a machine learning model for a different migration algorithm. Apart from selecting the critical features for each migration algorithm, they considered some common features for building the model. It reduces the model's forecast accuracy. Althahat et al. [20] proposed a neural network-based machine learning model to predict the performance parameter in the pre-copy and post-copy approaches. For building the model, they used the dataset and all features mentioned in the paper [18]. Compared to the result in the paper [18], they only got better performance in the downtime model for the pre-copy approach. The input feature dirty rate and working set size mainly impact the pre-copy approach's performance;

they do not affect the performance issue of the post-copy approach. Rather than considering input features separately for pre-copy and post-copy migration, they used all features mentioned in the dataset, lowering the prediction accuracy. Table 1 summarizes the comparison of related work.

In general, VM live migration modeling based on machine learning has been a significant research focus in recent years. Each model described in the literature has its own goal, migration algorithms, relevant resources, and impacting parameters. The main focus of these papers [18–20] is predicting the performance parameter of live virtual machine migration. To build a different model for each migration algorithm, they selected some common features instead of considering the parameters affecting the performance of each live migration approach. So in their work, selecting the relevant features for each migration algorithm is missing. Compared to their work, our primary focus is to find the best ML model for predicting the performance parameter, i.e., downtime and total migration time in the pre-copy approach with a minimum number of relevant features. Our methodology for selecting the best ML model to determine the optimal time for a pre-copy migration is discussed in Sect. 3.

3 Overview of the approach

We propose a three-stage approach to determine the optimal time for a pre-copy migration, as depicted in Fig. 1. These are namely *f*eature selection, *g*enerate ML model, and *a*pply model in pre-copy migration.

The proposed approach involves three stages: two offline stages and one online stage. The offline stages consist of activities that do not require real-time interaction, such as feature selection, data pre-processing, model training, optimization, and validation. The online stage, on the other hand, involves real-time interaction. The model generated during the offline stage is leveraged to enhance the performance of the model applied during the online phase. Therefore, the online and offline phases are related, and they work together to achieve the desired outcome.

Input feature selection is a crucial stage for generating a better ML model. It needs domain knowledge to select a set of relevant and important features from the available features. After selecting the input features, we simulate a pre-copy migration to identify the impact of each feature in the output metrics. Section 3.1 discusses the feature selection process in more detail. The output of the *f*eature selection stage is fed to *g*enerate a model. This phase generates various ML models with the identified features and verifies their accuracy using different metrics. This process is repeated until a better ML model with a minimum number of features is obtained. These processes are further explained in section 4. After the model generation, in the final stage, we apply this model in pre-copy migration to determine the optimal time for migrating the VM from source to destination with minimal impact on downtime or service delay. The final stage is explained in section 6.

Table 1 Comparison of related work

Study	Methodology	Input features	Prediction targets	Key findings
[7]	AVG and HIST simulation models	Memory size, VM resumption time, bandwidth, dirty page rate	Total migration and downtime	Predict performance with high accuracy. Does not consider working set size and is only applicable to LAN
[10]	Analytic model	Large number of factors	Total migration time, downtime, and total traffic	Due to a large number of input factors it is impractical to extend to new metrics
[50]	Working Set Prediction using Machine Learning approaches (WSPML)	Bandwidth, workloads	Total migration time	Reduces overall migration time compared to pre-copy. Only predicts memory pages required in near future
[52]	Artificial neural network	Large number of factors	CPU utilization, and bandwidth usage	Improves data center efficiency. Not applicable for predicting pre-copy migration performance metrics
[51]	Support Vector Regression	CPU, memory, bandwidth	Host utilization	Higher accuracy than other models.Only applicable for predicting host utilization, not migration performance
[25]	Machine learning-based downtime optimization (MLDOM)	CPU, memory, and network utilization	Downtime	Improvements in reducing downtime. Only applicable for migration over the WAN environment

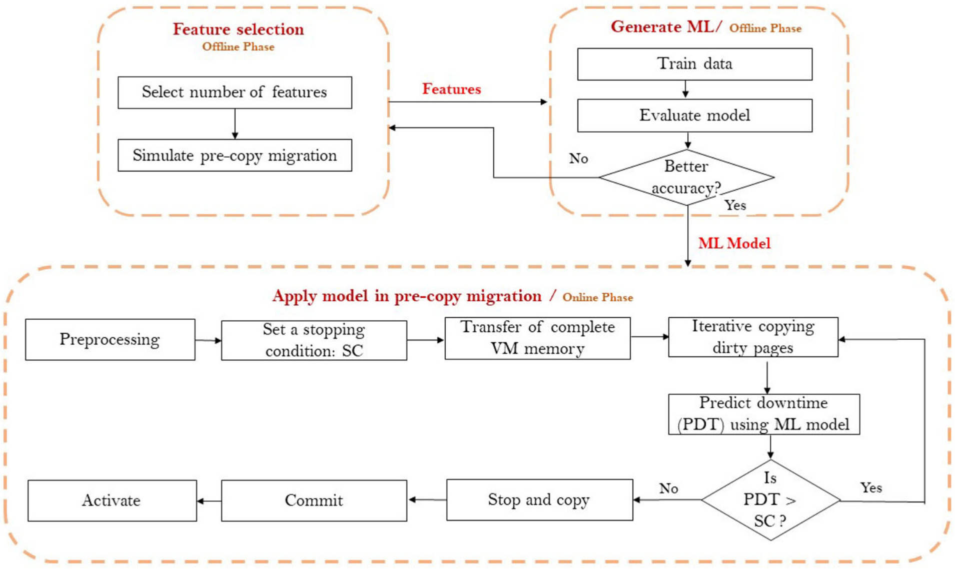

Feature selection
Offline Phase
Select number of features
Simulate pre-copy migration
Features
Generate ML/ Offline Phase
Train data
Evaluate model
No
Better accuracy?
Yes
ML Model
Apply model in pre-copy migration / Online Phase
Preprocessing
Set a stopping condition: SC
Transfer of complete VM memory
Iterative copying dirty pages
Predict downtime (PDT) using ML model
Is PDT > SC ?
Yes
No
Stop and copy
Commit
Activate

Fig. 1 Overview of the approach

3.1 A systematic approach to select features using simulation

Feature selection [55–57] is the process of obtaining a set of relevant features of the data set according to a feature selection criteria. Effective feature selection can enhance learning accuracy, minimize learning time, minimize computational overhead (time and space complexity), and simplify learning outcomes.

The main goal of feature selection is to improve the model's performance by reducing overfitting, decreasing the computational cost, and increasing the interpretability of the model. The main criteria for feature selection depend on the specific machine learning problem, high input and output correlation, and the nature of the data. Generating a model for predicting the performance of pre-copy migration requires domain expertise as well as a thorough examination of which input features are most relevant to the predicted output parameters. The entire memory of the VM from a host is copied to another host during the migration. As a result, the total migration time and downtime are dependent on the size of the VM's memory and bandwidth available for migration. Several studies [52, 58–63] were conducted for analyzing the correlation between bandwidth and performance parameters. Those studies have highlighted that the total migration time is reduced when high-bandwidth resources are available.

In the pre-copy method, each iteration copies the updated or dirty memory pages from source to destination. If the dirty page rate in each iteration is high, the total data transfer time will increase in each iteration, as will the amount of remaining updated memory pages in the stop and copy phase. It may cause an increase in downtime. So the VM page dirty rate and the VM's working set (it is a collection of recently referenced segments or memory pages) size [7, 10, 12, 12, 64–71] are relevant parameters for the pre-copy migration. To select these features are the critical parameters for the pre-copy migration, we have developed a feature selection algorithm and it is shown in Algorithm 1.

Algorithm 1 Feature Selection

```
1: Input: $X = \{x_1, x_2, ...x_n\}$
2: repeat
3:     initialize i=1
4:     for $x_i$ in $X$ do
5:         $x_c \subseteq X$ and $x_i \notin x_c$
6:         Create an array $PerformanceMetrics$ for storing the performance metrics
7:         initialize $v = R$, $R$ denotes the set of real numbers
8:         for $v_i$ in $v$ do
9:             $x_i = v_i$
10:            Simulate pre-copy migration $x_i \cup x_c$
11:            Compute performance metrics $TMT, DT$
12:            Store $TMT, DT, x_i, j$ in $PerformanceMetrics$
13:        end for
14:    end for
15: until all $x_i$ in $X$ is processed
16: for $x_i$ in $PerformanceMetrics$ do
17:     initialize $X_S = \emptyset$
18:     Plot graph $x_i$ against $TMT$ and $DT$
19:     if $(f(\Delta x_i) \Rightarrow \Delta TMT)$ or $(f(\Delta x_i) \Rightarrow \Delta DT)$ then
20:         $X_S$=$X_S \cup x_i$
21:     end if
22: end for
23: Output: $X_S$
```

Algorithm 1 Feature Selection is a feature selection method to identify the most significant features from a given set of features *X*. The input is a set of features *X*, and the algorithm works by iteratively selecting each feature x_i and comparing its performance against a subset of the remaining data points, x_c. The algorithm simulates pre-copy migration of each feature x_i in combination with the other features x_c in *X* to compute the performance metrics TMT and DT and stores them along with the feature x_i in an array called *PerformanceMetrics*. The detail of the pre-copy migration will be explained in the following paragraph. The algorithm then plots the performance metrics TMT and DT against each data point x_i in *PerformanceMetrics* and checks if a function $f(\Delta x_i)$ is true for either ΔTMT or ΔDT. If an input feature xi is found to have a significant impact on the performance metrics, it is added to the final set of selected features x_S. The algorithm repeats this process until all input features x_i in X have been processed and returns the final set of selected features x_S as output. In summary, this algorithm selects the most relevant features by evaluating their impact on performance metrics and selecting the ones that have the most significant impact.

To validate the impact of selected features for predicting the performance parameter in the pre-copy approach, we have conducted simulation experiments using CloudSim simulator [72–74]. We used CloudSim simulation to analyze the relationship between VM size, dirty rate, and bandwidth for downtime, as well as the overall migration time for the pre-copy method. To transfer dirty pages in the iterative phases, we use historical bitmap data. It is a two-dimensional bitmap array with *n* number of pages and iterations. In this array, bit 1 indicates that the page is dirtied in the corresponding iteration.

In the first iteration, we transfer all memory pages to the destination machine. In the following iterations, we transmit either updated or dirty pages. To avoid repeatedly sending the frequently produced dirty pages in this iteration, we categorize the memory pages into two classes: frequently dirty pages and less frequently dirty pages, based on a calculated threshold value. We use an equation available from [74] to find the threshold value.

$$T_1 = \lfloor [(\max[\text{page modification rate}] + \min[\text{page modification rate}]) \div 2] \rfloor$$

This threshold value is calculated in each iteration using the information in the bitmap array. If the page dirty rate of a memory page is higher than the calculated threshold value, these memory pages are saved in a separate array for transfer only in the stop-and-copy phase. It helps to reduce the repeated transfer of the frequently produced dirty pages in each iteration. This iterative phase will continue until the stop condition is reached, i.e., 29 iterations. We repeated the simulation with different VM sizes, page dirty rates, and bandwidth. We record the downtime and total migration for each condition.

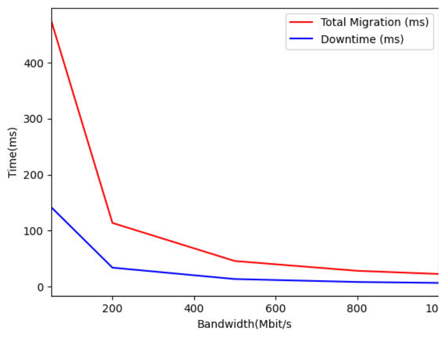
We set the number of iterations to 29 based on the default stopping condition of the Xen pre-copy approach [7]. Accordingly, the page size is set to 4 KB, the page dirty rate is 0.63, and the number of pages is 1000. We then use varied bandwidth and measure total migration and downtime to see how the bandwidth impacts these two parameters. Based on the obtained values, we plot a graph which is depicted in Fig. 2a.

Figure 2a shows a linear relationship between downtime and total migration time for bandwidth. The entire migration time and downtime are significantly reduced when the bandwidth is very high. This indicates that if adequate bandwidth is available throughout the migration process, the total migration time and downtime might be reduced.

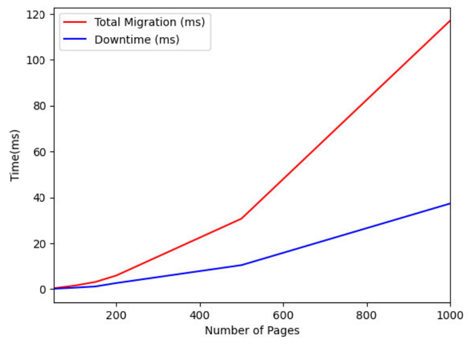
We repeat the simulation in 29 iterations with a 4KB page size and 200 MBit/s bandwidth. In this case, the page dirty rate varied with the page size. We also change the number of pages from 20 to 1000 and measure total migration time and downtime with a fixed bandwidth size. we plot graphs using the observed values and it is shown in Fig. 2b–d.

Figure 2b–d show the downtime and total migration time increased when the number of pages, page dirty rate per iteration, and working set size increased. The number of pages in the virtual machine's memory determines the amount of data that needs to be transferred during pre-copy live migration. The larger the number of pages, the longer it takes to migrate the virtual machine. In addition, if the virtual machine is actively using all its memory pages, pre-copy live migration may not be practical as the copying process can never complete. Therefore, it's important to consider the number of pages in the virtual machine's memory when planning a pre-copy live migration. If the number of pages increases, the dirty pages per iteration will increase and the higher the dirty page rate indicate that the virtual machine is highly active and more memory pages need to be transferred during pre-copy live migration. This can increase the time it takes to complete the migration and may result in some pages being transferred multiple times. Higher numbers of pages and dirty page rates can increase

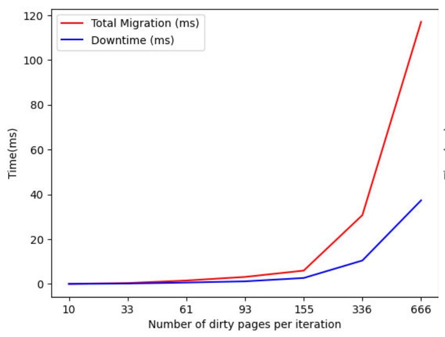
Fig. 2 Simulation Results



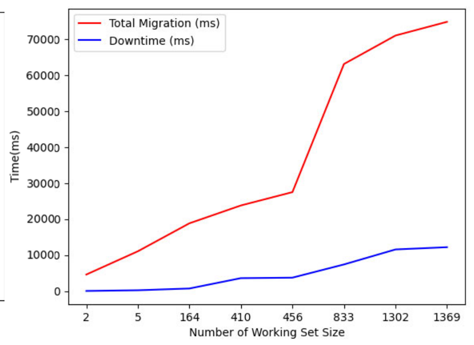
(a) Total Migration Time(ms) and Downtime(ms) vs Bandwidth



(b) Total Migration Time (ms) and Downtime (ms) vs the Number of Pages



(c) Total Migration Time (ms) and Downtime (ms) vs Page Dirty Rate



(d) Total Migration Time (ms) and Downtime (ms) vs Working Set Size migration time, while lower numbers may result in faster migrations.

The working set size represents the subset of the VM's memory that is actively being used. If the working set size is small, pre-copy live migration can be very efficient. This is because only a small subset of the VM's memory needs to be transferred during each iteration. However, if the working set size is large, pre-copy live migration may be less efficient, as more pages will need to be transferred during each iteration. In summary, the number of pages, bandwidth, dirty page rate, and working set size can all affect the efficiency and effectiveness of pre-copy live migration. If these factors are carefully considered, pre-copy live migration can be a very effective way to migrate a running VM from one physical host to another.

We also noticed from this experiment that if we can predict downtime or total migration time during the iterative phase using the above-mentioned parameters, we can set the stopping condition dynamically rather than using a predefined value. It will reduce the overall total migration time and downtime of the pre-copy approach. This simulation experiment motivates us to develop a stronger machine-learning prediction model to address the performance issue of the pre-copy approach.

Based on the feature selection Algorithm 1, we selected four relevant input features: (Virtual Machine size (VM_Size), Page Dirty Rate (PDR), Working Set Size (WSS), Page Transfer Rate (PTR) or bandwidth) to develop a better ML model for predicting Downtime (DT) and Total Migration Time (TMT) in pre-copy approach. Reducing the number of features can be beneficial for optimizing migration because it can simplify the process and reduce computational complexity. In addition, having a smaller set of features can make it easier to interpret and understand the results.

3.1.1 Feature selection using known techniques

This section discusses different feature selection techniques [56, 75] that are commonly available for selecting the best features for generating a machine learning model. We have selected 14 features from the dataset [18] and have done a Chi-square Test [76] and ANOVA test [77] in python with scikit-learn for the feature selection. We selected four features based on the test result and they are given in Table 2.

Comparison of selected features using proposed Algorithm 1 and known feature selection techniques are discussed in the section 5.

4 Generate a machine learning model to determine the optimal time for VM migration

The main steps for generating a model are *Data Preparation*, *Feature Extraction*, *Data Splitting*, *Training*, and *Testing*. These steps are shown in Fig. 3.

4.1 Data preparation

In our experiment, the data set we used for building and evaluating a model is provided by a research team at the National University of Seoul [18]. The dataset contains 40,000 records of various types of virtual machine migrations (i.e. pre-copy migrations, post-copy migrations, and modifications to pre-copy migrations, such as processor throttling (THR), delta compression (DLTC), and data compression (DTC)) collected over a period of several months in the CSAP lab cluster. The hardware setup they used for constructing the cluster is four identical servers with quad-core processors with a varying clock rate and 8-32GB of memory, three dedicated 1Gbit networks connected the machines for shared storage, public networking, and migration traffic with installed Ubuntu server 14.04 LTS on host PCs and the virtual machines. The performance of the live migration algorithm strongly depends on the workload running inside the VM [78, 79]. To examine the characteristics and performance metrics of several live migration strategies, 37 unique applications, and benchmark workloads were executed. The important workloads included are: SPECWeb to emulate a web server for e-commerce and banking services, OLTPBench [80] as a database applied to process online transactions, Mplayer that constitutes a multimedia workload, Memcached, Dacapo, parsec, Gzip, and idle. We filtered 8000 records from this data set based on migration type pre-copy migration and resized the distribution of values using StandardScaler.

4.2 Feature extraction

The data set containing the input features are VM size, page dirty rate, working set size, working set entropy, modified words per page, instructions per second, page transfer rate, CPU utilization of VM, network utilization of VM, CPU utilization on the host, CPU utilization on the destination, memory utilization on the host, and memory utilization on the destination. From these features, we selected four i.e. *V*M size, *d*irty page rate, *w*orking set size, and *b*andwidth for building a new ML model for predicting performance metrics i.e. downtime, and the total migration time in the pre-copy approach. The feature selection is

Table 2 Features selected using known techniques

Feature selection technique	Selected features
ANOVA	Working set entropy, Page dirty rate, Virtual machine size, and Working set size
Chi-square	Page dirty rate, Network utilization of VM, Working set size, Modified words per page

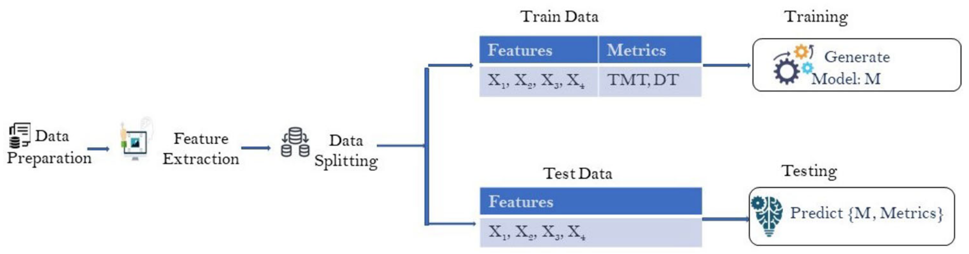


Fig. 3 Steps for generating a model

explained in Sect. 3.1. The description of the selected four features for creating the model is shown in Table 3.

In Table 3, the feature is described in the first two columns, and the third column is used to show where the parameter is analyzed. VM. Size in the first row refers to the amount of memory that has been allocated to the VM, not the maximum memory size that has been assigned. The relationship between page dirty rate (PDR) and working set size (WSS) [10, 81] is that WSS is the total number of pages dirtied during the entire period, whereas the dirty rate is the number of pages dirtied a given time.

4.3 Data splitting and generate machine learning model

To create the training and test data, we used 10-fold cross validation [82]: divided the data set into ten equal-sized subsets. Then, independently, 10 regression tests are run, with each of the ten sub-sets serving as testing data and the remaining nine as training data. This process is performed ten times, with the final evaluation result being the average of the results. The training data consists of the selected four features (discussed in the previous section) and two performance metrics for generating the model, whereas the test data is the input for predicting the model. The scikit-learn v0.17 [83] toolbox is used to train and evaluate the models for the prediction metrics.

Supervised machine learning techniques are used to generate machine learning models for predicting downtime and total migration time in the pre-copy migration. The different techniques we used for generating the model are linear regression, support vector regression (SVR) with linear kernels, SVR with bootstrap aggregation, ANN, and KNN.

Hyperparameter tuning or optimization [84] is important in the machine learning model. The process of selecting a set of optimal hyper-parameters for a learning algorithm is known as hyper-parameter tuning or optimization. A hyper-parameter is a value for a parameter that is used to influence the learning process. We used the optimal tuning parameters for SVR are C=100, gamma=.1, and kernel=-linear, which we found out using the grid search technique [85]. The penalty parameter, C, represents the difference in predicted and actual values. All input features are also standardized using the standard scalar method.

We used a grid search technique with the input features twenty, fourteen, and four and the output values downtime and total migration time to find out the best K values in the KNN approach. The value that we used in each model is shown in Table 4.

Table 3 The input features for creating proposed model

Feature	Description	Source
Virtual Machine size (VM_Size)	Number of memory pages allocated in VM	Virtual Machine Monitor (VMM)
Page Dirty Rate (PDR)	Average number of memory pages updated per second	VMM
Working Set Size (WSS)	Number of memory pages changed during VM initialization period	VMM
Page Transfer Rate (PTR)	Network bandwidth reserved for live migration	Source Host

Table 4 K value for KNN

Number of features	Downtime	Total migration time
20	5	7
14	6	8
4	6	2

We tested with numerous parameters to develop the optimal model using ANN, including two hidden layers with densities 32 and the number of neurons 16 or 32, batch sizes 5 or 25, epochs: 100, 200, or 300, and three hidden layers with densities 32 and the number of neurons 16 or 32. We create a distinct model for each of the twenty, fourteen, and four features using all of these factors and choose the best one. The best model comprises three hidden layers, each with 32 densities, 32 neurons, batch size 5, and 300 epochs. The performance of the generated models are discussed in Sect. 5.

5 Evaluation of the proposed machine learning model using a case study

After generating a model, the next step is to evaluate the performance of the model. For this, we conducted a case study using twenty features, fourteen features, and four features to show that the four features selected using feature selection Algorithm 1 are relevant to generate a better model to forecast the performance parameter of the precopy approach. To compare the performance of the model with different features, we used the performance metrics such as geometric Mean Absolute Error (gMAE) and geometric Mean Relative Error (gMRE) because these metrics are used to evaluate the model performance in the literature [18, 20], and we need to compare our results with theirs. The details about these are explained in this section.

5.1 Evaluation metrics

To compare the prediction accuracy of different machine learning models the following performance metrics are used.

geometric Mean Absolute Error (gMAE) geometric Mean absolute error is the geometric mean (n^{th} root of multiplication of n values) of the absolute difference between the predicted value and the actual value. The gMAE tells us how big of an error we can expect from the forecast. The equation is shown below

$$gMAE = \left(\prod\_i = 1^n |y\_i - x\_i|\right)^{\frac{1}{n}} = \sqrt[n]{|y\_1 - x\_1| * |y\_2 - x\_2 * *| \cdots |y\_n - x\_n|} \quad (1)$$

where y_i means the predicted value; x_i means the actual value in testing data set; Between the test data and the predicted score, n is the number of prediction pairs.

geometric Mean Relative Error (gMRE)

The difference between the actual value and the predicted value of data is called absolute error. The ratio of the absolute error of a predicted value and the actual value of the data is known as a relative error. gMRE is the geometric mean of the average relative error of the prediction.

$$gMRE = \left(\prod\_i = 1^n \frac{|y\_i - x\_i|}{x\_i}\right)^{\frac{1}{n}} = \sqrt[n]{\frac{|y\_1 - x\_1|}{x\_1} * \frac{|y\_2 - x\_2|}{x\_2} * \cdots \frac{|y\_n - x\_n|}{x\_n}} \quad (2)$$

5.2 Results and discussion

To validate the accuracy of the proposed ML model with the four identified influential features, we build a model with 14 features and 20 features (14 input features + composed features) and compare each model in terms of gMAE, and gMRE.

5.2.1 Model with 20 features

We selected 14 input features from the dataset and used six combined features from the paper [18] to build the model with twenty features. The twenty input features are listed in Table 5.

Using these twenty features we generate a different model for predicting total migration time and downtime using linear regression, SVR, ANN, and KNN. The prediction accuracy of each model is shown in Table 6.

The linear regression result shown in Table 6 does not reach adequate accuracy because the average prediction error for the model exceeds 10%. The complicated correlation of the features is the primary cause of the high inaccuracy. A simplistic method fails to grasp the complexities and fails to successfully train the model. When comparing the accuracy of the linear and SVR models, the ANN and KNN models show a substantial improvement. In the total migration model, ANN provides better accuracy with a 4% error, whereas KNN provides better accuracy for the downtime model with a 10% error. Neural networks can contain a large number of free parameters (weights and

Table 5 Twenty input features used for building our machine learning model

Feature	Description	Source
Virtual machine size (VM_Size)	Number of memory pages allocated in VM	Virtual Machine Monitor (VMM)
Page dirty rate (PDR)	Average number of memory pages modified per second	VMM
Working set size (WSS)	Number of memory pages changed during VM initialization period	VMM
Page transfer rate (PTR)	Network bandwidth reserved for live migration	Source Host
Working set entropy (WSE)	Working set memory entropy	VMM
Non-working set entropy (NWSE)	Non-working set memory entropy	VMM
Modified words per page (MWPP)	Number of altered words in updated pages	VMM
Instructions per second (IPS)	The number of instructions retired per second	Source Host
CPU utilization of VM (CPU.UTIL)	CPU usage of the VM process	Source Host
Network utilization of VM (NET.UTIL)	Network usage of the VM process	Source Host
CPU utilization on host (SRC.DST.CPU)	CPU core usage on the participated hosts	Source + destination host
Memory utilization on host (SRC.DST.MEM)	Memory utilization on the engaged hosts	Source + destination host
Weighted relative page transfer rate (R.PTR)	Weighted relative page transfer rate to page dirty rate	Composed
Non-working set size (NWSS)	Number of unmodified pages during the initiation period	Composed
Benefit of delta compression (DLTC.BF)	Anticipated pros of delta compression technique	Composed
Benefit of CPU throttling (THR.BF)	Anticipated advantage of CPU throttling technique	Composed
Compressed size of WSS (E.WSS)	The size of WSS expected after compression	Composed
Compressed size of NWSS (E.NWSS)	The size of NWSS expected after compression	Composed

Table 6 Accuracy of the different machine learning algorithms with 20 features for Pre-copy approach

Model	Performance metric	gMRE	gMAE
Linear regression	Total Migration Time (TMT)(ms)	3222	0.16
	Downtime (ms)	185	0.32
SVR	Total Migration Time (TMT)(ms)	1983	0.10
	Downtime (ms)	78	0.13
SVR with bootstrap aggregation	Total Migration Time (TMT)(ms)	3016	0.15
	Downtime (ms)	119	0.21
ANN	**Total Migration Time (TMT) (ms)**	**818**	**0.04**
	Downtime (ms)	67	0.12
KNN	Total Migration Time (TMT)(ms)	1376	0.07
	Downtime (ms)	**60**	**0.10**

biases across interconnected units), and they can fit highly intricate data that conventional models cannot.

5.2.2 Model with fourteen features

Then, in the dataset [18], we explored again with fourteen features excluded six combined features that are listed in Table 5 to see how the impact of fewer characteristics relative to more features differed. The model result of fourteen features is shown in Table 7.

When compared to the accuracy of other models such as Linear, SVR, and KNN, the results presented in Table 8 show that ANN performs quite well for both total migration and downtime models, with less error.

5.2.3 Model with four features selected using ANOVA and Chi-test

After generating different ML models with fourteen and twenty features, we generate an ML model with four features that were selected using ANOVA, and Chi-test

Table 7 Accuracy of the different machine learning algorithms for 14 features for Pre-copy approach

Model	Performance metric	gMAE	gMRE
Linear regression	Total Migration Time (TMT)(ms)	3889	0.20
	Downtime (ms)	480	0.85
SVR	Total Migration Time (TMT) (ms)	2795	0.14
	Downtime (ms)	313	0.56
SVR with bootstrap aggregation	Total Migration Time (TMT) (ms)	312	0.55
	Downtime (ms)	163	0.29
ANN	**Total Migration Time (TMT) (ms)**	**1390**	**0.07**
	Downtime (ms)	**50**	**0.08**
KNN	Total Migration Time (TMT) (ms)	1607	0.08
	Downtime (ms)	78	0.13

Table 8 Accuracy of the different machine learning algorithms for 4 features selected using ANOVA test

Model	Performance metric	gMAE	gMRE
Linear regression	Total Migration Time (TMT)(ms)	3315	0.18
	Downtime (ms)	531	1.01
SVR	Total Migration Time (TMT)(ms)	3593	.19
	Downtime (ms)	385	.68
SVR with bootstrap aggregation	Total Migration Time (TMT)(ms)	3881	.20
	Downtime (ms)	117	.21
ANN	Total Migration Time (TMT)(ms)	3770	.19
	Downtime (ms)	72	.13
KNN	**Total Migration Time (TMT)(ms)**	**3102**	**.16**
	Downtime (ms)	**70**	**.12**

explained in the subsection 3.1.1. The different model results of these selected features are shown in Table 8 and Table 9.

5.2.4 Proposed model with four features

Then we repeated the experiment using four relevant features selected using the Algorithm 1, namely, VM size, page dirty rate, working set size, and page transfer rate that explained in Sect. 3.1 to ensure that the selected features are sufficient to forecast pre-copy migration performance. Table 10 shows the results of different models with four relevant features selected using Algorithm 1.

Table 10 shows that SVR, KNN, and ANN do very well with lower error rates when compared to ML models created with twenty (Table 6), fourteen (Table 7), and four features (Tables 8 and 9). Also, linear regression shows better results with four features compared to fourteen features. This indicates that the four features selected using Algorithm 1 are sufficient to develop a better model for predicting the performance parameter of the pre-copy approach. In addition, when compared to other models, the

Table 9 Accuracy of the different machine learning algorithms for 4 features selected using Chi-Square test

Model	Performance metric	gMAE	gMRE
Linear regression	Total Migration Time (TMT)(ms)	5298	.29
	Downtime (ms)	532	1.08
SVR	Total Migration Time (TMT)(ms)	4238	.23
	Downtime (ms)	327	.46
SVR with bootstrap aggregation	Total Migration Time (TMT)(ms)	3903	0.20
	Downtime (ms)	155	.27
ANN	Total Migration Time (TMT)(ms)	5401	.29
	Downtime (ms)	74	.13
KNN	**Total Migration Time (TMT)(ms)**	**3916**	**.21**
	Downtime (ms)	**57**	**.10**

Table 10 Accuracy of the different machine learning algorithms for 4 features selected using Algorithm 1 for Pre-copy approach

Model	Performance metric	gMAE	gMRE
Linear regression	Total Migration Time (TMT)(ms)	3479	0.18
	Downtime (ms)	317	0.56
SVR	Total Migration Time (TMT)(ms)	1914	0.10
	Downtime (ms)	111	0.19
SVR with bootstrap aggregation	Total Migration Time (TMT)(ms)	2416	0.12
	Downtime (ms)	102	0.18
ANN	Total Migration Time (TMT)(ms)	1122	0.05
	Downtime (ms)	48	0.08
KNN	**Total Migration Time (TMT)(ms)**	**476**	**0.02**
	Downtime (ms)	**33**	**0.05**

KNN model has a very good performance with less than 5% error.

To validate our selected four features, the results demonstrate that they are more accurate and relevant to our proposed model. We measured the coefficient of determination(R^2) [86] for each model. The coefficient of determination (R^2) reflects how well the forecast fits the measured value; an (R^2) of 1 implies that the prediction fits the target value perfectly. This is shown in Table 11.

The R^2 value in Table 11 shows that the selected four features are more sufficient for predicting the performance parameters of the pre-copy approach.

5.2.5 Performance evaluation

We next compared the performance of our proposed model with other known outcomes to determine that our study produced better results. This is shown in Table 12.

Comparing the accuracy of the proposed work with two migration performance metrics, we selected other known works that used the same dataset. In these papers [18, 20], the authors selected fourteen features with four performance metrics, and twenty features (fourteen features + six derived features) with six performance metrics to build their models. So we have generated a machine learning model for other performance metrics (the total amount of transferred data, performance degradation, host CPU utilization, and host memory utilization) with four features and compared their results in Table 12. Table 12 suggests that our proposed machine learning model with KNN algorithms is more accurate than other known models with an error rate of less than 5% with four features. Furthermore, the results and the comparison confirm that the four identified features such as *V*M size, *d*irty page rate, *b*andwidth, and *w*orking set size are sufficient for developing an accurate model for predicting the total migration time and downtime for the pre-copy approach. Also, these four features are enough to determine other performance metrics mentioned in the paper [18].

In this study, regression, SVR, ANN, and KNN models with four, fourteen, and twenty features are trained to forecast the best time for live migration. With four features, KNN outperforms the rest of the models. KNN is simple, requires less training time, and is adaptable compared to the other machine learning models applied in this paper. The main reasons for this result are: (i) there is no need to tune several parameters to generate a better model; (ii) it is a non-parameterized algorithm that uses information acquired from the observed data to anticipate the amount of predicted variable in real-time without establishing a predefined parametric relationship between the predictor and the predicted variables. The fundamental advantage of KNN is that every variable is considered when determining whether or not an instance is a neighbor. It doesn't require any unique data distribution characteristics, and it can handle enormous data sets efficiently. Compared to KNN, neural networks require a significant amount of training data and many hyperparameter adjustments to reach appropriate accuracy. The critical issue in KNN is determining the ideal K value, which we overcame via hyperparameter tuning and selected a reasonable K value for greater performance.

Table 11 R^2 value for input features

ML model	TMT	DT
Linear Regression	0.87	0.83
SVR	0.87	0.83
SVR Bagging	.72	.90
ANN	.95	.98
KNN	**.95**	**.98**

Table 12 Performance comparison of proposed model with other known works

Model	Number of features	Performance Metric	gMAE	gMRE
Proposed Model	**4**	**Total Migration Time (TMT)(ms)**	**476**	**0.02**
		Downtime(ms)	**33**	**0.05**
		Total amount of data transferred (MB)	**36.13**	**0.01**
		Performance (%)	**0.03**	**0.04**
		Host CPU utilization (%)	**3**	**0.68**
		Host memory utilization (%)	**15**	**1.71**
SVR with Bagging [18]	20	Total Migration Time (TMT)(ms)	1053	0.06
		Downtime (ms)	96	0.23
		Total amount of data transferred(MB)	70.7	0.04
		Performance(%)	3.0	0.03
		Host CPU utilization(%)	2.7	4.8
		Host memory utilization(%)	16.4	1.79
ANN [20]	14	Total Migration Time (TMT)(ms)	3279	Not mentioned
		Downtime (ms)	114	Not mentioned

6 Applying proposed model to pre-copy migration

After the model generation, the next step is to apply the model in the iterative phase of the pre-copy migration and find out the optimal time for migrating VMs from one place to another. For this, we set up a simulation environment using CloudSim. The entire memory is transferred from source to destination during the initial stage of pre-copy migration. The updated or dirty pages are transferred from source to destination in a subsequent iteration. To apply our proposed model in the iterative phase to determine the stopping condition, first we set a predefined threshold value, which is downtime. We assume the downtime is zero or will be less than 100 ms. For the simulation experiment, we assume the VM size is 1024 MBit/s and the bandwidth is 200 MB. As per phase 1 of pre-copy migration, we transferred all the pages from the source to the destination. Then, in the iterative phase, we forecast the downtime in each phase with our proposed machine-learning model. Then we compared the obtained downtime with the previously defined threshold value. If the predicted downtime is less than the stopping condition (SC), we stop the iteration and enter the final stage, where we stop and copy the remaining pages, and activate the VM at the new destination. We repeated the experiment with different VM sizes and bandwidths as shown in Table 13 to monitor the performance of live migration. Finally, we compared the outcomes of our experiments and proved that the proposed method performs better than the existing pre-copy approach [74]. This is shown in Table 13.

The Table 13 values show that our proposed machine learning-based method to optimize pre-copy migration reduces 36%downtime in the case of page size 512 (KB) and BW 200 MBits/s, 9.5 % downtime in the case of page size 1024 (KB) and BW 200 Mbits/s compared to the existing pre-copy approach [74].

In practice, live virtual migration using machine learning models can be injected into cloud platforms to improve resource utilization, reduce downtime, and minimize costs. For example, a cloud provider can use machine learning models to predict the best time to migrate a VM based on the current load on the physical host, network traffic, and other factors.

To exploit this solution, cloud providers will need to modify their existing infrastructure to incorporate machine learning models. This will involve collecting and processing data from various sources, including VMs, physical hosts, network devices, and other monitoring tools.The cloud provider will also need to train machine learning models using historical data to predict the optimal time and destination for VM migration. The trained models can then be deployed in the cloud platform to automatically migrate VMs based on real-time data.

From the cloud provider side, the expected benefits of live virtual migration using machine learning models include:

- *Improved resource utilization* Machine learning models can help identify underutilized physical hosts and migrate VMs to these hosts, improving overall resource utilization.

Table 13 Simulation Results

Method	Page size 1024 (KB) BW 200 MBits/s DT (ms)	BW 100 MBits/s DT (ms)	Page Size 512 (KB) BW 200 MBits/s DT (ms)	BW 100 MBits/s DT (ms)
Proposed model	54.5	59.4	33.3	36.3
Existing Pre-copy approach [74]	59.7	63.1	45.3	47.1

- *Reduced downtime* Live virtual migration can help minimize service disruption by allowing VMs to be migrated without disrupting the service being provided.
- *Cost savings* By optimizing resource utilization and reducing downtime, cloud providers can reduce costs associated with running and maintaining their infrastructure.

In summary, live virtual migration using machine learning models can be a valuable tool for cloud providers looking to improve resource utilization, reduce downtime, and minimize costs. However, implementing this solution will require modifications to the existing infrastructure and a significant investment in data collection, processing, and model training.

7 Conclusion and future work

LVM is crucial in virtualized environments for migrating a virtual machine from one host to another with minimum service interruption. One of the most prevalent and reliable LVM approaches is pre-copy. However, the key obstacles to this strategy are the high dirty page rate in each iteration and the predefined stopping conditions. This could result in a longer overall migration time, downtime, or system unavailability. In this paper, to overcome the problem, we have proposed an optimal time prediction model with a smaller set of significant features. To select the model's input feature, we conducted a simulation experiment using CloudSim. When compared to the state of the art, our model has better prediction accuracy with less than 5% error.

The outcomes of this research show that we can use the machine learning method to predict downtime and total migration time for a pre-copy live migration approach. However, there are different types of live virtual migration, and various performance metrics need to be considered to select the best live migration algorithm. In our future work, we plan to extend this research work with feature selection for building different types of migration algorithms and performance metrics. Moreover, we plan to develop a framework for implementing an efficient pre-copy approach using this proposed model and conduct a real-time experiment to test the framework in a cloud environment.

Acknowledgements We acknowledge that we have used the dataset (CSAP Virtual Machine Live Migration Dataset) kindly provided by Dr.Changyeon Jo, which was obtained over several months in the CSAP lab cluster. This paper was made possible by the Grant NPRP 8-531-1-111 from Qatar National Research Fund (QNRF). The statements made herein are solely the responsibility of the authors.

Funding Open Access funding provided by the Qatar National Library. The authors have not disclosed any funding.

Data availability Enquiries about data availability should be directed to the authors.

Declarations

Competing interests The authors have not disclosed any competing interests.

References

1. Singh, M.: Virtualization in cloud computing-a study. In: 2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), pp. 64–67. IEEE (2018)
2. Rashid, A., Chaturvedi, A.: Virtualization and its role in cloud computing environment. Int. J. Comput. Sci. Eng. **7**(4), 1131–1136 (2019)
3. Clark, C., Fraser, K., Hand, S., Hansen, J.G., Jul, E., Limpach, C., Pratt, I., Warfield, A.: Live migration of virtual machines. In: Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation, vol. 2, pp. 273–286 (2005)
4. Gupta, A., Dimri, P., Bhatt, R.: An optimized approach for virtual machine live migration in cloud computing environment. In:

Evolutionary Computing and Mobile Sustainable Networks, pp. 559–568. Springer (2021)
5. Hines, M.R., Deshpande, U., Gopalan, K.: Post-copy live migration of virtual machines. ACM SIGOPS Oper. Syst. Rev. **43**(3), 14–26 (2009)
6. Sahni, S., Varma, V.: A hybrid approach to live migration of virtual machines. In: 2012 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM), pp. 1–5. IEEE (2012)
7. Akoush, S., Sohan, R., Rice, A., Moore, A.W., Hopper, A.: Predicting the performance of virtual machine migration. In: 2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, pp. 37–46. IEEE (2010)
8. De Maio, V., Kecskemeti, G., Prodan, R.: An improved model for live migration in data centre simulators. In: Proceedings of the 9th International Conference on Utility and Cloud Computing, pp. 108–117 (2016)
9. Nathan, S., Kulkarni, P., Bellur, U.: Resource availability based performance benchmarking of virtual machine migrations. In: Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering, pp. 387–398 (2013)
10. Nathan, S., Bellur, U., Kulkarni, P.: Towards a comprehensive performance model of virtual machine live migration. In: Proceedings of the Sixth ACM Symposium on Cloud Computing, pp. 288–301 (2015)
11. Breitgand, D., Kutiel, G., Raz, D.: {Cost-Aware} live migration of services in the cloud. In: Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE 11) (2011)
12. Denning, P.J.: Working set analytics. ACM Comput. Surv. (CSUR) **53**(6), 1–36 (2021)
13. Wu, T.Y., Guizani, N., Huang, J.S.: Live migration improvements by related dirty memory prediction in cloud computing. J. Netw. Comput. Appl. **90**, 83–89 (2017)
14. Salfner, F., Tröger, P., Polze, A.: Downtime analysis of virtual machine live migration. In: The Fourth International Conference on Dependability (DEPEND 2011), pp. 100–105. IARIA (2011)
15. Altahat, M.A., Agarwal, A., Goel, N., Kozlowski, J.: Dynamic hybrid-copy live virtual machine migration: Analysis and comparison. Procedia Comput. Sci. **171**, 1459–1468 (2020)
16. Bashar, A., Mohammad, N., Muhammed, S.: Modeling and evaluation of pre-copy live vm migration using probabilistic model checking. In: 2018 12th International Conference on Signal Processing and Communication Systems (ICSPCS), pp. 1–7. IEEE (2018)
17. Melhem, S.B., Agarwal, A., Goel, N., Zaman, M.: Markov prediction model for host load detection and vm placement in live migration. IEEE Access **6**, 7190–7205 (2017)
18. Jo, C., Cho, Y., Egger, B.: A machine learning approach to live migration modeling. In: Proceedings of the 2017 Symposium on Cloud Computing, pp. 351–364 (2017)
19. Motaki, S.E., Yahyaouy, A., Gualous, H.: A prediction-based model for virtual machine live migration monitoring in a cloud datacenter. Computing **103**(11), 2711–2735 (2021)
20. Altahat, M.A., Agarwal, A., Goel, N., Zaman, M.: Neural network based regression model for virtual machines migration method selection. In: 2021 IEEE International Conference on Communications Workshops (ICC Workshops), pp. 1–6. IEEE (2021)
21. Goldberg, R.P.: Survey of virtual machine research. Computer **7**(6), 34–45 (1974)
22. Keller, E., Ghorbani, S., Caesar, M., Rexford, J.: Live migration of an entire network (and its hosts). In: Proceedings of the 11th ACM Workshop on Hot Topics in Networks, pp. 109–114 (2012)
23. Baker-Harvey, M.: Google compute engine uses live migration technology to service infrastructure without application downtime
24. Wu, Y., Zhao, M.: Performance modeling of virtual machine live migration. In: 2011 IEEE 4th International Conference on Cloud Computing, pp. 492–499. IEEE (2011)
25. Arif, M., Kiani, A.K., Qadir, J.: Machine learning based optimized live virtual machine migration over wan links. Telecommun. Syst. **64**(2), 245–257 (2017)
26. Elsaid, M.E., Abbas, H.M., Meinel, C.: Virtual machines pre-copy live migration cost modeling and prediction: a survey. Distrib. Parallel Databases **40**, 1–34 (2021)
27. Patel, M., Chaudhary, S., Garg, S.: Machine learning based statistical prediction model for improving performance of live virtual machine migration. J. Eng. (2016). https://doi.org/10.1155/2016/3061674
28. Weisberg, S.: Applied Linear Regression, vol. 528. Wiley, Hoboken (2005)
29. Uyanik, G.K., Nece, G.: A study on multiple linear regression analysis. Procedia Soc. Behav. Sci. **106**, 234–240 (2013)
30. Shannon, C.E.: A mathematical theory of communication. ACM Sigmobile Mob. Comput. Commun. Rev. **5**(1), 3–55 (2001)
31. Awad, M., Khanna, R.: Support vector regression. In: Awad, M., Khanna, R. (eds.) Efficient Learning Machines, pp. 67–80. Springer, Cham (2015)
32. Caruana, R., Niculescu-Mizil, A.: An empirical comparison of supervised learning algorithms. In: Proceedings of the 23rd International Conference on Machine Learning, pp. 161–168 (2006)
33. Farahnakian, F., Pahikkala, T., Liljeberg, P., Plosila, J.: Energy aware consolidation algorithm based on k-nearest neighbor regression for cloud data centers. In: 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing, pp. 256–259. IEEE (2013)
34. Peterson, L.E.: K-nearest neighbor. Scholarpedia **4**(2), 1883 (2009)
35. Song, Y., Liang, J., Lu, J., Zhao, X.: An efficient instance selection algorithm for k nearest neighbor regression. Neurocomputing **251**, 26–34 (2017)
36. Fix, E., Hodges, J.L.: Discriminatory analysis. nonparametric discrimination: consistency properties. Int. Stat. Rev./Revue Internationale de Statistique **57**(3), 238–247 (1989)
37. Yegnanarayana, B.: Artificial Neural Networks. PHI Learning Pvt. Ltd., New Delhi (2009)
38. Aldhalaan, A., Menascé, D.A.: Analytic performance modeling and optimization of live vm migration. In: European Workshop on Performance Engineering, pp. 28–42. Springer (2013)
39. Deng, L., Jin, H., Chen, H., Wu, S.: Migration cost aware mitigating hot nodes in the cloud. In: 2013 International Conference on Cloud Computing and Big Data, pp. 197–204. IEEE (2013)
40. Li, J., Zhao, J., Li, Y., Cui, L., Li, B., Liu, L., Panneerselvam, J.: imig: Toward an adaptive live migration method for kvm virtual machines. Comput. J. **58**(6), 1227–1242 (2015)
41. Liu, H., Xu, C.-Z., Jin, H., Gong, J., Liao, X.: Performance and energy modeling for live migration of virtual machines. In: Proceedings of the 20th International Symposium on High Performance Distributed Computing, pp. 171–182 (2011)
42. Mann, V., Gupta, A., Dutta, P., Vishnoi, A., Bhattacharya, P., Poddar, R., Iyer, A.: Remedy: Network-aware steady state vm management for data centers. In: International Conference on Research in Networking, pp. 190–204. Springer (2012)
43. Zhang, J., Ren, F., Lin, C.: Delay guaranteed live migration of virtual machines. In: IEEE INFOCOM 2014-IEEE Conference on Computer Communications, pp. 574–582. IEEE (2014)

44. Xu, F., Liu, F., Liu, L., Jin, H., Li, B., Li, B.: iaware: Making live migration of virtual machines interference-aware in the cloud. IEEE Trans. Comput. **63**(12), 3012–3025 (2013)
45. Gao, J.: Machine learning applications for data center optimization (2014)
46. Jian, C., Bao, L., Zhang, M.: A high-efficiency learning model for virtual machine placement in mobile edge computing. Clust. Comput. **25**(5), 3051–3066 (2022)
47. Khodaverdian, Z., Sadr, H., Edalatpanah, S.A.: A shallow deep neural network for selection of migration candidate virtual machines to reduce energy consumption. In: 2021 7th International Conference on Web Research (ICWR), pp. 191–196. IEEE (2021)
48. Alrajeh, O., Forshaw, M., Thomas, N.: Using virtual machine live migration in trace-driven energy-aware simulation of high-throughput computing systems. Sustain. Comput. Inform. Syst. **29**, 100468 (2021)
49. Ouacha, A., El Ghmary, M.: Virtual machine migration in mec based artificial intelligence technique. IAES Int. J. Artif. Intell. **10**(1), 244 (2021)
50. Zaw, E.P.: Machine learning based live vm migration for efficient cloud data center. In: International Conference on Big Data Analysis and Deep Learning Applications, pp. 130–138. Springer (2018)
51. Nehra, P., Nagaraju, A.: Host utilization prediction using hybrid kernel based support vector regression in cloud data centers. J. King Saud Univ. Comput. Inf. Sci. **34**, 6481–6490 (2021)
52. Duggan, M., Shaw, R., Duggan, J., Howley, E., Barrett, E.: A multitime-steps-ahead prediction approach for scheduling live migration in cloud data centers. Softw. Pract. Exp. **49**(4), 617–639 (2019)
53. Alrajeh, O., Forshaw, M., Thomas, N.: Machine learning models for predicting timely virtual machine live migration. Lecture Notes in Computer Science, vol. 10497. Springer, Cham (2017)
54. Hassan, M., Babiker, A., Amien, M., Hamad, M.: Sla management for virtual machine live migration using machine learning with modified kernel and statistical approach. Eng. Technol. Appl. Sci. Res. **8**(1), 2459–2463 (2018)
55. Dhal, P., Azad, C.: A comprehensive survey on feature selection in the various fields of machine learning. Appl. Intell. (2021). https://doi.org/10.1007/s10489-021-02550-9
56. Cai, J., Luo, J., Wang, S., Yang, S.: Feature selection in machine learning: A new perspective. Neurocomputing **300**, 70–79 (2018)
57. Kirov, D.E., Toutova, N.V., Vorozhtsov, A.S., Andreev, I.: Feature selection for predicting live migration characteristics of virtual machines. T-Comm **15**(7), 62–70 (2021)
58. Li, H., Xiao, G., Zhang, Y., Gao, P., Lu, Q., Yao, J.: Adaptive live migration of virtual machines under limited network bandwidth. In: Proceedings of the 17th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, pp. 98–110 (2021)
59. Wood, T., Ramakrishnan, K., Shenoy, P., Van der Merwe, J., Hwang, J., Liu, G., Chaufournier, L.: Cloudnet: dynamic pooling of cloud resources by live wan migration of virtual machines. IEEE/ACM Trans. Netw. **23**(5), 1568–1583 (2014)
60. Mandal, U., Habib, M.F., Zhang, S., Chowdhury, P., Tornatore, M., Mukherjee, B.: Heterogeneous bandwidth provisioning for virtual machine migration over sdn-enabled optical networks. In: Optical Fiber Communication Conference, pp. 3–2. Optical Society of America (2014)
61. Yazidi, A., Ung, F., Haugerud, H., Begnum, K.: Effective live migration of virtual machines using partitioning and affinity aware-scheduling. Comput. Electr. Eng. **69**, 240–255 (2018)
62. Bhardwaj, A., Krishna, C.R.: Performance evaluation of bandwidth for virtual machine migration in cloud computing. Int. J. Knowl. Eng. Data Min. **5**(3), 139–152 (2018)
63. He, T., Toosi, A.N., Buyya, R.: Performance evaluation of live virtual machine migration in sdn-enabled cloud data centers. J. Parallel Distrib. Comput. **131**, 55–68 (2019)
64. Shi, B., Shen, H.: Memory/disk operation aware lightweight vm live migration across data-centers with low performance impact. In: IEEE INFOCOM 2019-IEEE Conference on Computer Communications, pp. 334–342. IEEE (2019)
65. Denning, P.J.: Working sets past and present. IEEE Trans. Software Eng. **1**, 64–84 (1980)
66. Kumar, A.V., Krishnakumar, V., Kumar, A.N.: Efficient performance upsurge in live migration with downturn in the migration time and downtime. Clust. Comput. **22**(5), 12737–12747 (2019)
67. Chanchio, K., Yaothanee, J.: Efficient pre-copy live migration of virtual machines for high performance computing in cloud computing environments. In: 2018 3rd International Conference on Computer and Communication Systems (ICCCS), pp. 497–501. IEEE (2018)
68. Bitchebe, S., Mvondo, D., Tchana, A., Réveillère, L., De Palma, N.: Intel page modification logging, a hardware virtualization feature: study and improvement for virtual machine working set estimation. arXiv preprint arXiv:2001.09991 (2020)
69. Jain, P.: Optimized pre-copy live virtual machine migration for memory-intensive workloads. PhD thesis, Dublin, National College of Ireland (2021)
70. Katal, A., Bajoria, V., Sethi, V.: Simulated annealing based approach for virtual machine live migration. In: 2021 8th International Conference on Smart Computing and Communications (ICSCC), pp. 219–224. IEEE (2021)
71. Naga Malleswari, T.Y.J., et al.: Resumptionof virtual machines after adaptive deduplication of virtual machine images in live migration. Int. J. Electr. Comput. Eng. **11**(1), 654–663 (2021)
72. Gupta, A.: A modelling & simulation via cloudsim for live migration in virtual machines. IOP Conf. Ser. Mater. Sci. Eng. **1116**, 012138 (2021)
73. Calheiros, R.N., Ranjan, R., De Rose, C.A., Buyya, R.: Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services. arXiv preprint arXiv:0903.2525 (2009)
74. Sharma, S., Chawla, M.: A three phase optimization method for precopy based vm live migration. Springerplus **5**(1), 1–24 (2016)
75. Chandrashekar, G., Sahin, F.: A survey on feature selection methods. Comput. Electr. Eng. **40**(1), 16–28 (2014)
76. Lancaster, H.O., Seneta, E.: Chi-square distribution. In: Armitage, P. (ed.) Encyclopedia of biostatistics, 2nd edn. Wiley, Hoboken (2005)
77. Faraway, J.J.: Practical Regression and ANOVA Using R, vol. 168. University of Bath, Bath (2002)
78. Moghaddam, M.J., Esmaeilzadeh, A., Ghavipour, M., Zadeh, A.K.: Minimizing virtual machine migration probability in cloud computing environments. Cluster Comput. (2020). https://doi.org/10.1007/s10586-020-03067-5
79. Li, C., Feng, D., Hua, Y., Qin, L.: Efficient live virtual machine migration for memory write-intensive workloads. Futur. Gener. Comput. Syst. **95**, 126–139 (2019)
80. Katsarakis, A., Ma, Y., Tan, Z., Bainbridge, A., Balkwill, M., Dragojevic, A., Grot, B., Radunovic, B., Zhang, Y.: Zeus: locality-aware distributed transactions. In: Proceedings of the Sixteenth European Conference on Computer Systems, pp. 145–161 (2021)
81. Nathan, S., Bellur, U., Kulkarni, P.: On selecting the right optimizations for virtual machine migration. ACM SIGPLAN Not. **51**(7), 37–49 (2016)
82. Kohavi, R., et al.: A study of cross-validation and bootstrap for accuracy estimation and model selection. IJCAI **14**, 1137–1145 (1995)

83. Hao, J., Ho, T.K.: Machine learning made easy: a review of scikit-learn package in python programming language. J. Educ. Behav. Stat. **44**(3), 348–361 (2019)
84. Yang, L., Shami, A.: On hyperparameter optimization of machine learning algorithms: Theory and practice. Neurocomputing **415**, 295–316 (2020)
85. Ngoc, T.T., Le Van Dai, C.M.T., Thuyen, C.M.: Support vector regression based on grid search method of hyperparameters for load forecasting. Acta Polytech. Hung. **18**(2), 143–158 (2021)
86. Cameron, A.C., Windmeijer, F.A.: An r-squared measure of goodness of fit for some common nonlinear regression models. J. Econom. **77**(2), 329–342 (1997)

Raseena M. Haris Raseena M. Haris received a Master of Technology (M. Tech) in Computer Science and Engineering from Calicut University, Kerala, India, and Bachelor of Technology (B. Tech) in Computer Science and Engineering from Mahatma Gandhi University, Kottayam, India. She is currently pursuing a Ph.D. degree with the Department of Computer Science and Engineering, Qatar University. She worked as a Software Engineer, Programmer, Lecturer, and Research Assistant. She is currently a Teaching Assistant at Qatar University. Her research interests include cloud computing, virtualization, machine learning, cloud security, and live virtual machine migration.

Khaled M. Khan is an associate professor of Computer Science and Engineering at Qatar University. Prior to these, he served Western Sydney University (Australia) and was the Head of Graduate Programs. He received his Ph.D. from Monash University, and BS and MS degrees from the Norwegian University of Science and Technology. He is a senior member of IEEE.

Armstrong Nhlabatsi is a Research Associate at KINDI Computing Research Centre, Qatar University. He received his PhD degree in Computer Science from the Open University, United Kingdom. His research interests include security requirements engineering, security risk evaluation, requirements traceability, and the feature interaction problem for information security. Dr Nhlabatsi has contributed to security research and innovation in adaptive security for the cloud, threat-specific security risk evaluation, quantification of satisfaction of security requirements, and information security for sports accreditation systems. He is currently working on approaches to analysing emergent security properties of systems of socio-technical systems, and visualization of security policies.

Mahmoud Barhamgi is an Associate Professor at the Computer Science and Engineering Department of Qatar University. He received his Ph.D. and HDR (Accreditation to Direct Research) in Computer Science from Claude Bernard Lyon-1 University (France) in 2010 and 2020, respectively. His current research work is focused on engineering solutions that would make the cyber space safer for people and more respectful of their rights such as the right to privacy, transparency and fairness.