Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

Analysis and prediction of virtual machine boot time on virtualized computing environments

Ridlo Sayyidina Auliya[1], Yen-Lin Lee[1], Chia-Ching Chen[1], Deron Liang[1] and Wei-Jen Wang[1*]

Abstract

Starting a virtual machine (VM) is a common operation in cloud computing platforms. In order to achieve better management of resource provisioning, a cloud platform needs to accurately estimate the VM boot time. In this paper, we have conducted several experiments to analyze the factors that could affect VM boot time in a computer cluster with shared storage. We also implemented four models for VM boot time prediction and evaluated the performance of the four models based on the datasets of four hosts and seven hosts in our environment, where the four models are the rule-based model, the regression tree model, the random forest regression model, and the linear regression model. According to our analysis, we found that host capability and maximal network bandwidth are two main factors that can influence VM boot time. We also found that VM boot time becomes harder to predict when booting VMs at different hosts concurrently due to competition between hosts to obtain resources. According to the experimental results, the proposed random forest regression is the best model for VM boot time prediction with an average accuracy of 94.76% and 96.59% in predicting VM boot time in two clusters with four and seven compute hosts, respectively.

Keywords Virtual machine, Boot time prediction, Virtual machine placement, Cloud computing

Introduction

Virtual Machine (VM) is a crucial component that serves as a virtualized computing resource to offer accessible services in cloud computing environments [1]. Virtualization technology enables multiple VMs to operate on a single physical server, thereby enhancing the cost-effectiveness and efficiency of resource utilization [2, 3]. VMs and virtualization technologies have been widely used in cloud computing. For example, many public cloud platforms, such as Amazon Elastic Compute Cloud (EC2) [4] and Microsoft Azure [5], allow users to rent VMs that host user applications. Many users also use several virtualization tools, such as KVM [6] and VMware vSphere [7], to build their private cloud platforms. In response to the computing requests from users, cloud platforms may need to provision VMs at runtime (VM provisioning) [8–11]. VM provisioning involves co-locating VMs in the same physical host to optimize resource utilization [12, 13]. Therefore, an efficient VM provisioning mechanism is crucial for cloud platforms to achieve optimal resource utilization.

VM boot time prediction plays a pivotal role in VM provisioning on cloud platforms. Each cloud platform needs to guarantee a certain quality of service (QoS), as stated in Service-Level Agreements (SLA) [14–16]. The violation of SLAs could have a serious impact on the cloud platform. Thus, ensuring the performance of the cloud environment and improving the quality of the provided services is essential [17–20]. Inaccurate VM boot

*Correspondence:
Wei-Jen Wang
wjwang@csie.ncu.edu.tw
[1] Department of Computer Science and Information Engineering, National Central University, Zhongda Road, Zhongli District, Taoyuan City 320, Taiwan

time prediction can have severe consequences in several cases, as follows:

- **VM Failover** [1]: Inaccurate VM boot time prediction can result in choosing the wrong hosts to boot the VMs, which leads to long VM boot times and consequently, adds more delay to the failover process. Delay in the failover process can be very dangerous in certain cases where high availability becomes a priority, such as in time-critical applications [21].
- **Cloud Simulation Tools** [22]: Inaccurate VM boot time prediction can affect the makespan in cloud scheduling.

In summary, inaccurate VM boot time prediction can have several adverse consequences, such as reducing the effectiveness and efficiency of resource management, increasing costs, and decreasing the performance of cloud platforms. Therefore, organizations should invest in more accurate prediction models for VM booting times to address these consequences. Despite its importance, VM boot time prediction has rarely been explored in literature, and only a few studies have focused on it [23].

Conventionally, VM boot time is assumed to be constant [24]. VM start-up process generally includes three stages: placement of VMs on physical machines, transmission of VM images for booting, and VM booting [24]. These stages typically take tens of seconds. However, the exact start-up duration can be affected by several factors [25–27]. VM boot time is measured as the duration to boot a VM on a selected host until the VM is ready for execution. While VM boot time is assumed to be constant and often ignored, the booting process consumes resources. Therefore, the previous assumption is incorrect. VM boot time can be affected by many factors, and these factors have been discussed in a study by Nguyen et al. [24], who found that co-located VMs could compete with CPU and I/O resources, resulting in varying VM boot times. Similarly, Nitu et al. [22] pointed out that VM boot times can be long and vary depending on the number of VMs that started.

There are two popular ways to boot VMs: from a VM image or a bootable volume [28, 29]. The boot-from-image approach transfers a VM image from the storage host to the compute host. Meanwhile, the boot-from-volume approach uses a bootable volume, the block storage created from a VM image containing the bootable operating system. The volume can be stored in a remote shared storage host and does not need to be transferred to a compute host. An illustration of the boot-from-volume approach for shared storage is presented in Fig. 1.

In this study, all VMs in our computer cluster are booted from the volumes. The bootable volume is created based on a VM image and stored in a shared storage host (Fig. 1). Then, these volumes are used to boot a VM instance on a compute host. In terms of VM boot time prediction, the boot-from-volume approach allows the

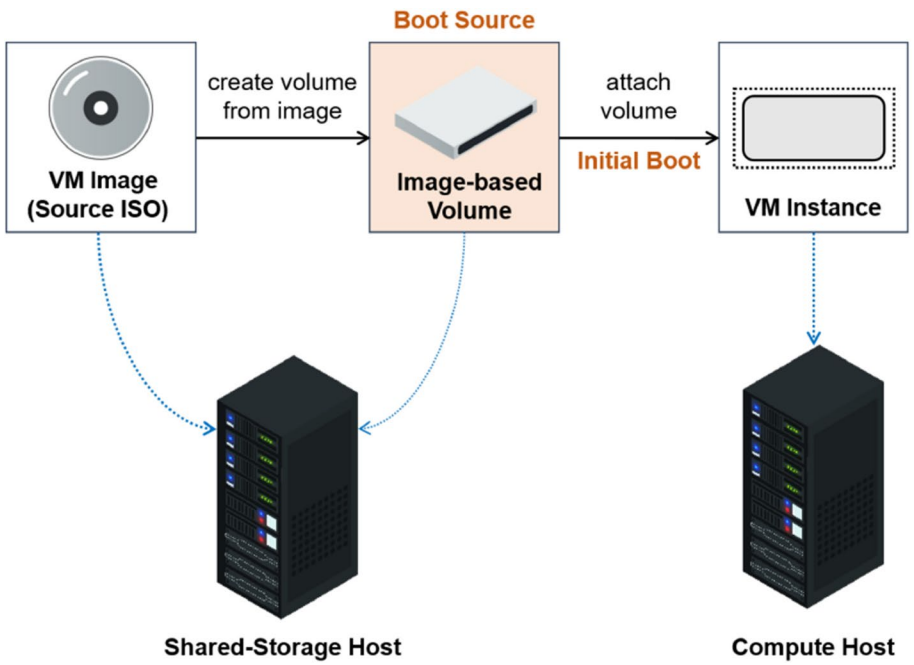

Fig. 1 Boot-from-volume approach in a shared storage host

second stage of the VM start-up process which transfers the VM image from the storage host to the compute host, to be ignored [24]. Thus, this approach can enhance the efficiency of VM boot time prediction by shortening the start-up process and directly focusing on measuring VM boot time. Moreover, the boot-from-volume approach also provides the ability to boot a VM from a remotely attached volume and improves the management and recovery in a computer cluster [30].

There are two types of VM boot time prediction models: Nguyen et al.'s rule-based model [24] and Govindaraju et al.'s Machine Learning (ML)-based model [1]. Nguyen et al. [24] identified several factors affecting VM boot times, such as I/O throughput and CPU capacity, and then proposed a rule-based VM boot time prediction model. They claimed the model can reproduce VM boot times under different resource contention. However, the model's major limitation is that it does not consider host competition for VM boot time or the number of CPU cores. Govindaraju et al. [1] proposed an ML-based VM boot time prediction model using a regression tree algorithm. Their model uses several features such as VM image size, CPU utilization, memory utilization, network utilization, and number of concurrent VM creation requests. The proposed model has two limitations. First, the model is only used to predict VM start-up time in an OpenStack platform and does not provide a factor analysis of the selected features. Second, the model is only used in a small-scale cluster of only four hosts. In summary, a more accurate VM boot time prediction model is necessary to address the limitations of the existing prediction models.

In this study, we aim to find a more accurate VM boot time prediction model in a computing platform with shared storage. We collected and analyzed experimental data to identify the factors that influence the boot time of a VM. Then, we built four prediction models: Nguyen et al.'s model [24], a regression tree model inspired by Govindaraju et al. [1], a random forest regression model, and a linear regression model. Notice that, to the best of our knowledge, the random forest regression and linear regression models have not been used for VM boot time prediction in the literature. In addition, we adopted the heterogeneous cloud computing architecture in our experiments. The heterogeneous architecture has been widely implemented [31–33], especially by data centers because VMs and hardware heterogeneity could serve varied customers [34]. We designed two experiments in two computer clusters with four and seven compute hosts to evaluate the performance of each model. Our experimental results show that the random forest regression outperformed the other models in both computer clusters, regardless of the training dataset size. In the second experiment, the model achieved an average accuracy of 94.76% and 96.59% in predicting VM boot times outside the range of known values, using 75% of the dataset for training.

This study has two major contributions as follows:

- We discovered that the number of CPU cores, number of VMs on a host, and concurrent booting of VMs on another host can affect the average boot time of the host. We found that VM boot time can be hard to predict due to the contention among physical hosts. This phenomenon only occurs in the shared-VM-image-storage environment and has not yet been discussed in previous studies. By focusing on the previous three factors, we can improve the VM boot time prediction accuracy. In addition, we released our experimental data for other researchers interested in VM boot time prediction.
- We propose to use random forest regression for VM boot time prediction. Random forest regression performs well in predicting VM boot times using the data within and outside the known dataset range. In addition, we have discovered that a small training dataset is sufficient to build an efficient prediction model with high accuracy. In our experiments, the proposed model can achieve the average accuracy of 94.76% and 96.59% in predicting VM boot time in two clusters with four and seven compute hosts by using only 1% case coverage.

The remainder of this paper is organized as follows. Related work section introduces the existing approaches in VM boot time prediction. VM boot time analysis section discusses the experiments designed to identify the factors determining the VM boot time. Data collection and model building section describes the data collection and prediction models' training. Performance evaluation section discusses the performance comparison of the existing VM boot time prediction models. Finally, this study's conclusion and future research directions are presented in Conclusions and future work section.

Related work

This section discusses related research on VM boot time predictions. First, we introduce the VM start-up process and the factors affecting the VM boot time. Then, we discuss the existing VM boot time prediction models, which can be divided into the rule-based model and the ML-based model.

Virtual machine start-up process and the influencing factors

VM start-up process generally includes three stages [24]:

1. The cloud/cluster scheduler determines an appropriate physical machine to place the VM based on the resource requirements and the optimization goals.
2. The VM image is transferred from the storage host to the compute host, and a disk for the VM is created from the image.
3. The VM is booted on the compute host.

The duration of each start-up stage can be different due to the influence of many factors. For example, the duration of the first stage was typically short. However, several factors such as user requests, the distribution of available computing resources, and the purpose of the scheduling algorithm (for example, to shorten the VM boot time, reduce network traffic, or ensure the quality of service) can determine the duration of the first stage [24].

VM image size and maximal network bandwidth are the main factors affecting the second stage's duration [24]. Researchers usually consider that the duration of the second stage accounts for most of the VM start-up time. Therefore, various acceleration methods [35, 36] have been proposed for this stage. However, the duration of this stage may be negligible in certain cases. For example, when a user boots a VM from a volume [37] in an OpenStack [38] environment, transferring the VM image is unnecessary because OpenStack allows users to create a bootable volume on shared storage and uses this volume to boot the VM.

In the third stage, the hypervisor boots the VM on the selected host until it is ready for execution. The duration of this stage is called the VM boot time [24]. VM boot time is essential for the start-up process and cannot be ignored. In the literature, many researchers and cloud simulation tools, such as CloudSim [39, 40], assume that the VM boot time is constant [24]. They assumed that the booting process consumes few resources based on the assumption that the environment for booting the VM is ready, and therefore, the VM boot time can be ignored. However, the fact is that the VM boot time is long and can be affected by many factors, as pointed out by several studies [22, 24].

A study by Nitu et al. [22] pointed out that the VM boot time is very long, and depends on the number of VMs that are booting. Moreover, server consolidation makes the workload on each host different, making it difficult to predict the VM boot time accurately. Nguyen et al. [24] confirmed the influence of various factors on VM boot time, such as CPU usage, memory usage, network pressure, and I/O throughput. Their experimental results found that CPU usage and I/O throughput were key factors affecting VM boot time. In summary, VM boot time is not constant and can be affected by various hardware specifications and configuration factors. Thus, regarding their influence, these factors should be considered in measuring VM boot time.

Research on VM boot time prediction models

VM boot time is different from the actual physical machine boot time since in the virtualized environment, the physical machines are usually assumed to be ready for running a VM at any time. Two types of VM boot time prediction models exist in the literature: Nguyen et al.'s rule-based model [24] and Govindaraju et al.'s ML-based [1] models. Both prediction models are discussed in the following subsections.

Rule-based prediction models

The rule-based prediction models are built using the expertise of researchers to create and maintain rules. In a study by Nguyen et al. [24], they identified several primary factors that affect VM boot times, such as I/O throughput and CPU capacity. Then, they proposed a rule-based VM boot time prediction model based on the identified factors. A VM boot time should integrate the I/O and CPU dimensions. Therefore, the idea of the proposed model can be written in Eq. 1, then rewritten in Eq. 2 as follows:

$$boot\_time = time_{I/O} + time_{CPU} \tag{1}$$

$$boot\_time = \frac{e^x \times \alpha}{1-x} + \frac{\beta}{1-y} \tag{2}$$

Each variable of Eqs. (1) and (2) can be described in Table 1 as follows:

According to their experimental results, their proposed rule-based model can reproduce VM boot time under different resource contentions. However, because the rule-based model relies on human experts to create and maintain rules, these rules may work differently in different configurations. Thus, further experimentation is required to validate the accuracy of the proposed model for different configurations. However, the major limitation of this study is that it does not consider host competition for VM boot time or the number of CPU cores.

Machine learning (ML)-based prediction models

The ML-based prediction models were built using ML techniques. Govindaraju et al. [1] proposed a model for predicting the average, minimum, and maximum VM start-up times using a regression tree algorithm. They collected the data for their model in a private OpenStack environment with one controller host and four compute hosts. They proposed three regression

Table 1 Attributes of Nguyen et al.'s rule-based model

Variables	Description
boot_time	The time from the execution of the VM boot instruction to the completion of the initialization of the VM
$time_{I/O}$	the total time required by a VM to perform I/O operations during the booting process
$time_{CPU}$	the total time required by the CPU to run boot operations without resource contention
x	utilization percentage of the total I I/O throughput
y	utilization percentage of the CPU resources
α	the total time required by a VM to perform I/O operations during the booting process
β	the total time required by the CPU to run boot operations without resource contention

tree-based prediction models for predicting average, minimum, and maximum VM start-up times. The predictive analysis workflow included (1) dataset preparation, (2) regression tree model training, (3) doing a pruning process, (4) testing the trained model, and (5) using the trained model for prediction. According to their experimental results, the proposed regression tree model achieved an accuracy of 91.51%. The advantage of the proposed model is the graphical representation that helps to understand the phenomenon modeled. However, the model has not been implemented to predict VM boot time. Moreover, the prediction model did not provide a factor analysis of the selected features. This study also used a small scale consisting of only four hosts.

Several ML-based prediction models, such as linear regression and random forest regression, were used in the study by Li et al. [41]. In their study, they used two ML-based models to predict the workload of a virtual machine. According to their experimental results, random forest regression has proven to be the best model for VM workload prediction. Other ML-based prediction models, such as Support Vector Machine (SVM) [42, 43], also have been implemented to predict VM workload for VM provisioning. Although the existing studies have proven that the ML-based method provides good prediction performance, these studies have not been implemented to predict the VM boot time. Thus, there exists an opportunity to observe the performance of ML-based models for VM boot time prediction.

Summary

In summary, according to studies by Nitu et al. [22], and Nguyen et al. [24], VM boot time is not constant. VM boot time can be affected by many factors, such as CPU usage, memory usage, network pressure, and I/O throughput. There are two prediction models for VM boot time prediction, which can be seen in Table 2.

There exist two types of VM boot time prediction models: Nguyen et al.'s rule-based [24] and Govindaraju et al.'s ML-based models [1], using a regression tree algorithm. The major limitation of Nguyen et al.'s rule-based model [24] is that it does not consider host competition for VM boot time or the number of CPU cores. Meanwhile, the existing ML-based model of Govindaraju et al. [1] has not been implemented to predict VM boot time. Moreover, the prediction model did not provide a factor analysis of the selected features. This study also used a small-scale cluster consisting of only four hosts. In summary, a more accurate VM boot time

Table 2 Summary of the related work

Attributes	VM Boot Time Prediction Models	
	Rule-based	**ML-based**
Prediction Methods	Nguyen et al. [24]	Regression Tree (Govindaraju et al. [1])
Overview	Use the expertise of researchers to create rules based on several features, including CPU time and I/O time to build a prediction model	Use several features, including VM image size, memory utilization, and network utilization to build prediction models
Advantages	1) easy to interpret	1) provide higher accuracy
	2) fast processing time	2) provide a better understanding of data and features
Limitations	1) does not consider competition between hosts	1) have not been applied for VM boot time prediction
	2) does not consider the number of CPU cores	2) only applied for a small-scale cluster of four hosts
		3) does not provide feature analysis

prediction model is necessary to address the limitations of the existing prediction models.

VM boot time analysis

In order to understand the factors that could affect VM boot time, we have performed several experiments on a cluster consisting of four compute hosts. In this section, we first explain the experimental protocol of those experiments and then analyze the following three experiment scenarios:

- The impact of the number of host CPU cores on the VM boot time.
- The impact of the number of VMs on the VM boot time.
- The impact between different hosts.

Experimental protocol

In this study, we used KVM to build a computing environment and start the VM from the volume. The KVM-based environment, such as OpenStack is widely used in cloud computing [44]. In addition, since heterogeneous cloud computing architecture has been widely implemented [31–33], we adopted the heterogeneous cloud computing architecture in this study. In other words, the specifications of the hosts used to construct the computing environment, such as the number of CPU cores, memory size, and hard disk capacity, are different. The architecture of the computing environment is shown in Fig. 2, and the specifications of the compute hosts are shown in Table 3.

Four compute hosts (Compute 1 to Compute 4) are responsible for running VMs, one controller host is responsible for managing all hosts and VMs in the environment, and one storage host with SSD

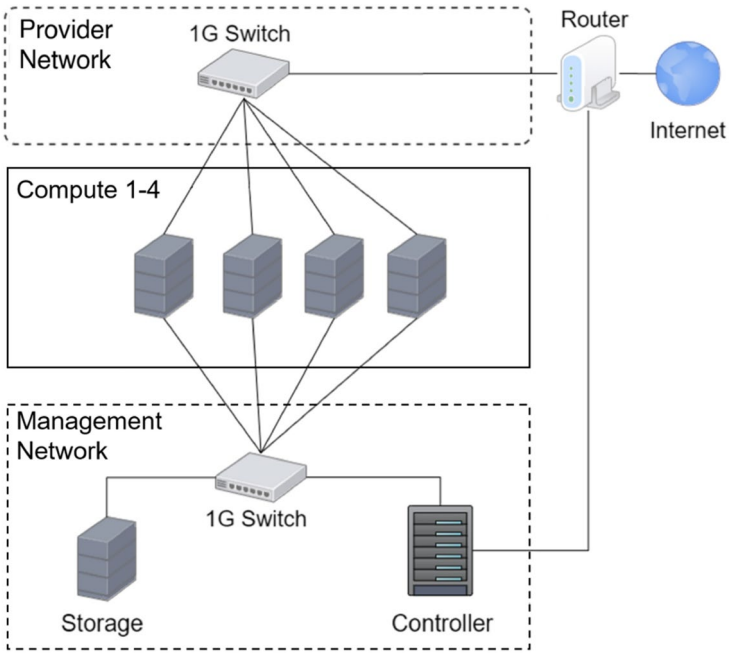

Fig. 2 Computing architecture for the experiments

Table 3 The specifications of each host of the computer cluster

Host role	CPU	Memory	Disks	Operation System
Compute 1	Intel i7-6700 (Quad-core)	32 GB	1024 GB HDD	Ubuntu 16.04 Server
Compute 2	Intel i5-8500 (Hex-core)	32 GB	512 GB HDD	Ubuntu 16.04 Server
Compute 3	Intel i7-9700 (Octa-core)	32 GB	1024 GB HDD	Ubuntu 16.04 Server
Compute 4	AMD A6-3670K (Quad-core)	32 GB	512 GB HDD	Ubuntu 16.04 Server
Controller	Intel i7-6700 (Quad-core)	16 GB	1024 GB HDD	Ubuntu 16.04 Server
Storage	Intel i7-3770 (Quad-core)	12 GB	1024 GB HDD & 480 GB SSD	Ubuntu 16.04 Server

is responsible for storing VM images and volumes. Each host and switch/router are connected by 1 Gigabit Ethernet. In addition, we set up all VMs with one vCPU and 1 GB of memory and used Ubuntu 16.04 as the host operating system. In the experiments, we did not consider the impact of changes in VM specifications on the VM boot time. An early study [27] showed that the capacity of a VM does not affect the VM boot time, which means that a VM with one core and 1 GB of memory and a VM with four cores and 16 GB of memory requires the same boot time. We collected VM boot time data from compute hosts 1, 2, 3, and 4 and analyzed the factors affecting VM boot time. Compute 4 is not used here because it is used for prediction accuracy in the next section (Performance evaluation section). The VM is launched by the libvirt API [45] on the specified host, and then booted from the volume stored in the storage host.

We use two types of VMs in our experiments: workload VMs (wVMs) and experimental VMs (eVMs). wVM is a VM that runs stress tests to compete for computing resources, especially CPU resources, to simulate a running VM in an actual cloud. eVM is a VM that is ready to be started and will not perform any load after booting, and its boot time is measured and analyzed in this study. The VM boot time is the time from the execution of the VM boot instruction to the completion of the initialization of the Getty service [46] in the eVM.

Impact of the number of CPU cores on the host

To understand how the number of physical CPU cores affects the VM boot time, we evaluated the following three factors in the experiments: (i) the number of eVMs (from 1 to 6), (ii) the number of wVMs (from 0 to 12), and (iii) the number of CPU cores (quad-core (Compute 1 and 4), hex-core (Compute 2), and octa-core (Compute 3)). The experiments included all the combinations of the above three factors, and each combination was performed at least 30 times. We run several wVMs on the compute host first, then boot several eVMs simultaneously. The experimental results are shown in Fig. 3 and are available in our git repository [47].

According to our experimental results, we found that the VM boot time is affected by the number of physical CPU cores. In Fig. 3, the VM boot time values in (b), (c), and (d) are almost the same when the total numbers of wVMs and eVMs are less than the number of CPU cores. However, in Fig. 3, the VM boot time values in (b), (c), and (d) become different when the total number of wVM and eVM approaches a multiple of the number of CPU cores. For example, the VM boot time in Fig. 3b significantly increases when the number of wVM changes from two to three, and the number of eVM is 1. This is because Compute 1 is a host with a quad-core CPU that can support four VMs.

Similarly, the VM boot time in Fig. 3a increases significantly at the same point because Compute 4 is also a host with a quad-core CPU. This situation may be related to the process scheduling of the host operating system.

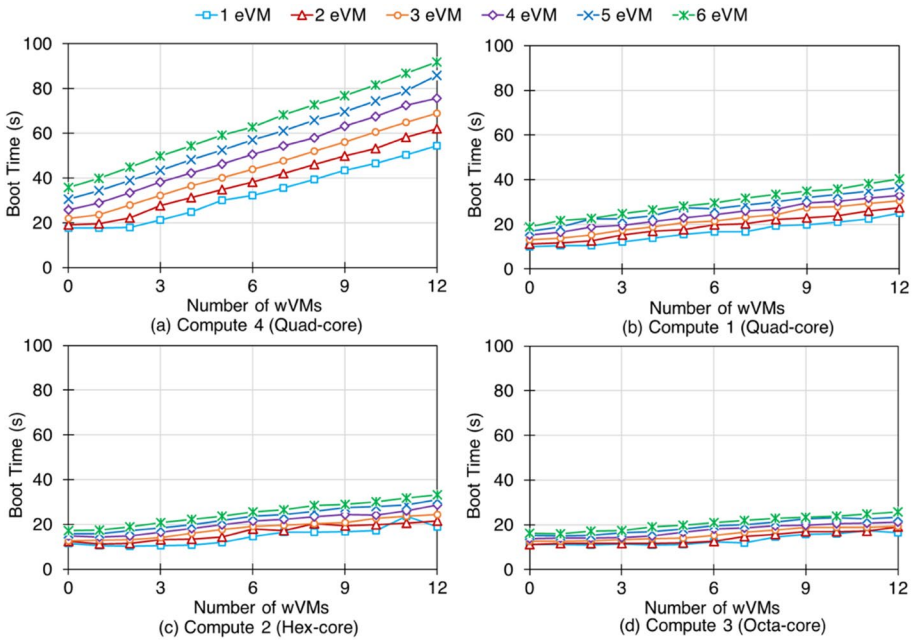

Fig. 3 Average VM boot time in different test cases on Compute 1, 2, 3, and 4 with 1G network

When the total numbers of wVM and eVM are less than the number of CPU cores, the host scheduler tends to allocate a separate CPU core to each VM; therefore, there is no CPU contention. However, when the total number of wVM and eVM approaches the number of CPU cores, the host scheduler must determine which process (VM) can use the CPU resources. In this case, the VM boot time increases significantly due to the CPU contention. According to our observations, the growth curve of VM boot time has a stair-like shape. Furthermore, as the total number of wVM and eVM increases, the VM boot time growth trend becomes slower (Fig. 3c and d).

Impact of the number of VMs on the host

In this section, we discuss the impact of factors related to the number of VMs, such as network transmission delay, number of eVMs, and number of wVMs, on the average VM boot time. Because all VM volumes are stored in the storage host, and the host needs to access the VM volumes to boot up a VM, the booting process should inevitably increase the network workload and encounter transmission delays. To verify this idea, we conducted experiments by changing the network card of each host from 1G to 10G. However, due to the lack of 10G switches, we use network cables to connect the compute and storage hosts to simulate the 10G network environment. We then performed the same experiments described in Impact of the number of CPU cores on the host section, and the experimental results are shown in Fig. 4. Experimental results are available in our git repository [47].

Comparing the results in Figs. 3 and 4, we find that when the total numbers of wVMs and eVMs on the same host are small, the VM boot time remains the same. However, as the total number of wVM and eVM increases, the VM boot time in a 10G network environment is shorter than in a 1G network environment. In addition, by observing the network traffic when booting an eVM on Compute 1 in a 1G network environment, we found that the eVM usually transmits a large amount of data in a short period, which can easily cause resource contention and increase transmission delays. In summary, the transmission delay of the network affects the VM boot time, and as the total number of wVMs and eVMs increases, the delay becomes more significant.

In addition, in both Figs. 3 and 4, if the number of wVMs is fixed, it can be observed that the VM boot time increases as the number of eVMs increases. However, if the number of eVMs is fixed, the VM boot time also increases as the number of wVMs increases. This is because the resource contention on the host becomes more frequent as the number of eVMs or wVMs increases.

Impact between hosts

Because all VM volumes are stored in the storage host, the average boot time in a host can be affected by the VM boot time in another host. Therefore, this experiment aimed to measure the impact of VMs (wVMs and eVMs)

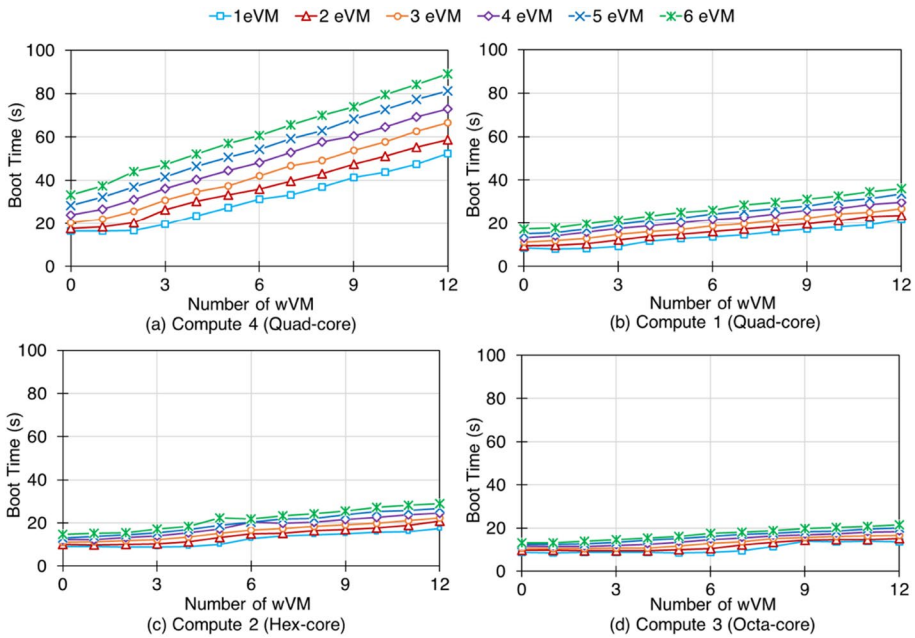

Fig. 4 Average VM boot time in different test cases on Compute 1, 2, 3, and 4 with 10G network

on other hosts during VM boot time. This experiment has two scenarios: First, we run 0 to 5 wVMs on Compute 3, then boot 1 to 3 eVMs on Compute 1 one minute later, and measure the VM boot time. Second, we booted 0 to 6 eVMs simultaneously on two of the three hosts (Compute 1, Compute 2, and Compute 3) and measured the VM boot time on the target host (one of the two). There were 18 cases in the first scenario and 252 in the second scenario. Each case was performed five times to measure the VM boot time. The experimental results of the above two scenarios are shown in Figs. 5 and 6, respectively, where the unit of VM boot time is seconds.

Figure 5 shows that the wVMs on other hosts do not affect the measured VM boot time. This is because wVM mainly competes for CPU resources on the compute host rather than for resources on the storage host. Figure 6

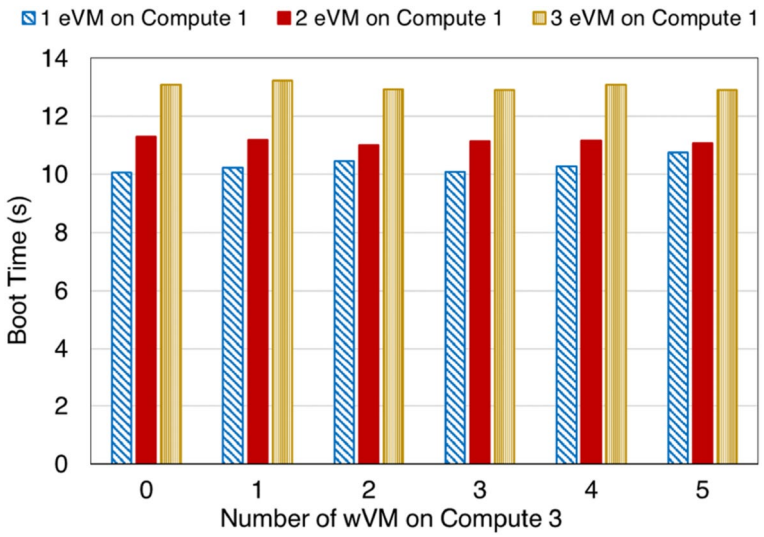

Fig. 5 Average boot time (in seconds) of 1 to 3 eVMs on Compute 1 when Compute 3 has 0 to 5 wVMs

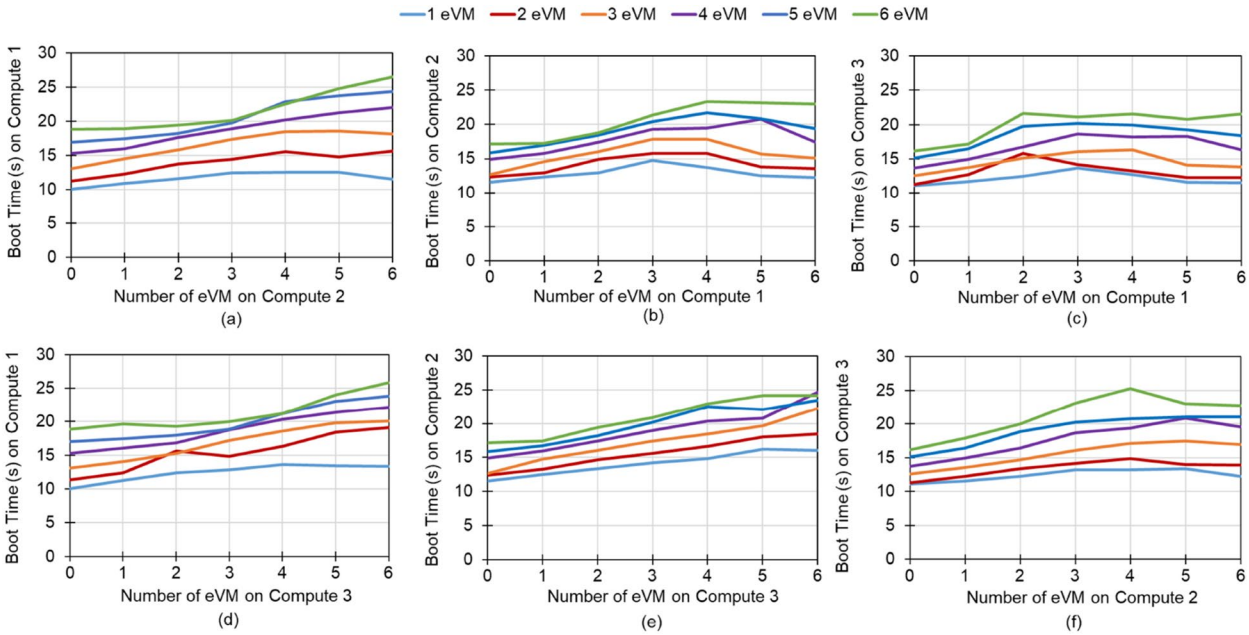

Fig. 6 **a**,**d** Average boot time of 1 to 6 eVMs on Compute 1 when booting 0 to 6 eVMs on Compute 2 and Compute 3, respectively. **b**,**e** Average boot time of 1 to 6 eVMs on Compute 2 when booting 0 to 6 eVMs on Compute 1 and Compute 3, respectively. **c**,**f** Average boot time of 1 to 6 eVMs on Compute 3 when booting 0 to 6 eVMs on Compute 1 and Compute 2, respectively

shows the effect of eVMs on other hosts during VM boot time in two situations, as follows:

- **Case 1:** The average VM boot time could increase as the number of the remote eVMs increases, such as starting up 4 to 6 eVMs at Compute 1 and 0-6 eVMs at Compute 2 (the 4, 5, 6 eVMs lines in Fig. 6a). This could be because network contention from the remote host increases and consequently delays VM booting.
- **Case 2:** As the number of remote eVMs increases, the average VM boot time can increase and suddenly either stop increasing or decrease at some turning point. For example, starting up one eVM at Compute 2 and 0-6 eVMs at Compute 1 (the 1 eVM line in Fig. 6c) has a turning point of three eVMs at Compute 1. We believe that the network contention ability of a host does not always increase as the number of eVMs increases. The number of physical CPU cores could limit the contention ability.

Summary

Based on our observations, host capability (workload) and maximal network bandwidth are the main factors affecting the VM boot time. The number of VMs can introduce the workload in a host. Meanwhile, the number of CPU cores represents the host's capability to handle the workload. The analysis of the factors affecting the average VM boot time can be summarized as follows:

- **The Number of CPU Cores on a Host**
  - There is no CPU competition in a host when the total number of VMs is less than the number of host CPU cores because each VM can use a separate CPU core.
  - In a host, the VM boot time of a VM could be delayed when the total number of VMs is greater than or equal to the number of CPU cores in that host.
- **The Number of VMs on a Host**
  - If the maximal network bandwidth is sufficient, the VM boot time of a VM becomes shorter, and the network competition has less impact on the VM boot time.
- **The Impact Between Hosts**
  - In a host, the VM boot time of a VM could be delayed due to the influence of network and I/O competition from other co-located VMs.
  - The average VM boot time could increase as the number of remote booting VMs increases. As the number of remote booting VMs increases, the average VM boot time could increase and then suddenly either stop increasing or decrease at some turning points.

Data collection and model building

This section discusses the data collection and model training of the four VM boot time prediction models. The prediction models to be used are a rule-based model by Nguyen et al. [24] and three ML-based prediction models: the regression tree model by Govindaraju et al. [1], linear regression model, and random forest regression model. Notice that, to the best of our knowledge, random forest regression and linear regression models have not been used for VM boot time prediction in the literature.

Environment setup

In our experiments, we used two types of VMs, which are eVMs and wVMs, as follows:

- eVM: the experimental VM (eVM) is the VM whose boot time is the prediction target.
- wVM: the workload VM (wVM) is the VM that emulates the VM running on the environment.

Those VMs used to demonstrate different VM booting situations and generate VM boot time datasets to be used in this study. The VM boot time prediction steps in this study can be described as follows (Fig. 7):

1. **Data Collection and Preprocessing**
   - **Data Collection:** VM boot times were collected by measuring the time from the execution of the VM boot instruction to the completion of the initialization of the VM. The collected data had several attributes based on the identified factors that can influence VM boot time prediction, including the number of CPU cores on a host, the number of booting VMs on a host, and the impact between hosts.
   - **Data Cleansing:** This step was performed after VM boot times were collected. The data cleansing step consists of cleansing and preprocessing by handling outliers, missing values, and inconsistencies.
   - **Data Splitting:** After the collected data were cleansed and preprocessed, the datasets were split into training and testing datasets. These datasets

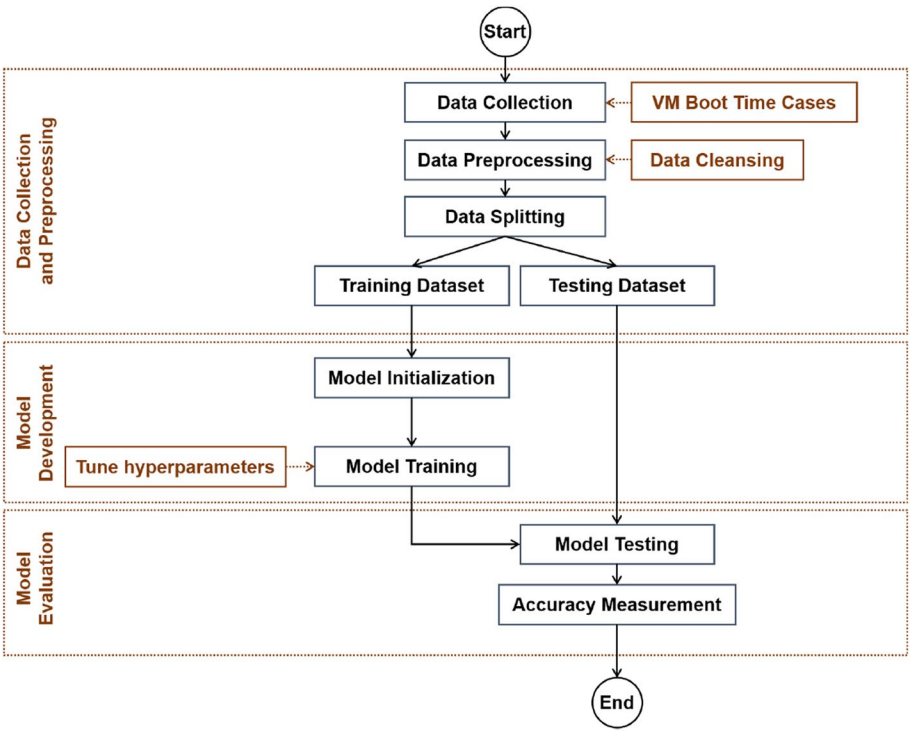

Fig. 7 VM boot time prediction with random forest regression

were used to train and evaluate the ML models, including random forest regression.

2. **Model Development**
   - **Model Initialization:** The random forest regression model was initialized using the preferred machine learning library. We used the "Statistics and Machine Learning Toolbox" of MATLAB.
   - **Model Training:** After the random forest regression model was initialized, we trained the random forest regression model using the training datasets. At this stage, we also tuned some hyperparameters, including the number of trees, tree depth, and minimum samples per leaf. In this study, we tuned the number of trees using "*n_estimators*" hyperparameter and set its value to 1000.
3. **Model Evaluation**
   - **Model Testing:** After the training phase was completed, the prediction models were tested with the test dataset.
   - **Accuracy Measurement:** The accuracy was measured as the percentage of prediction difference divided by the actual time. After completing the training and testing steps, we measured the accuracy of the four prediction models.
   - **Significance Test:** A statistical significance test can be performed to measure the significance of prediction models' accuracy values. In this case, we can use several methods, such as the Wilcoxon test.

Data collection and preprocessing

The data was collected on two clusters, each consisting of four and seven hosts, respectively. We measured the time from the execution of the VM boot instruction to the completion of the initialization of the VM. In addition, we did not consider any criteria for selecting a host for data collection. The hosts are selected based on each VM placement case, and each case is randomly chosen from all cases. For the cluster of four hosts, we used Compute 1, Compute 2, Compute 3, and Compute 4 to collect the data and calculate the average VM boot of 1 to 6 eVMs on a host (Compute 1 to Compute 4) when there are 0 to 12 wVMs on that host. For the cluster with seven hosts, we used Compute 1 to 7 to collect data.

There are two types of data to be collected as follows:

1. **Single-Host:** the average VM boot time of 1 to 6 eVMs on a host (Compute 1 to Compute 7) when there are 0 to 12 wVMs on the host.

2. **Multi-Host:** the average VM boot time of the eVMs on each host (Compute 1 to Compute 3, and Compute 5 to Compute 7) where each host supposes 0 to 6 eVMs and when the eVMs are booted concurrently.

In the cluster with four hosts, the first experiment had 312 cases and the second experiment had 7200 cases, with 18 cases in the two experiments being the same. In other words, the dataset consists of 7494 cases. In the cluster with seven hosts, the single-host dataset had 78 cases per host, and the multi-host dataset had 823,542 cases per host. The data collection process required a lot of time due to a large number of cases. There were 823,542 cases, each of which took 3 minutes and 30 seconds to collect. Because it takes thousands of days to collect all of them, it needs too much time and cost. Therefore, we used 1% of the overall cases, which is 8,235 cases. For the 8,235 cases, approximately 28 days were required for data collection.

Each record is the average result of each case performed at least five times, and each record consists of seven attributes selected based on the summary in VM boot time analysis section. The attributes can be seen in Table 5. In addition, we performed data cleansing to guarantee the accuracy of the data. The data cleansing was performed by discovering the cases in which the boot time was longer than the median VM boot time. Then, we re-measured the boot time on those cases. It is worth mentioning, that most of the data are clean. Data cleansing was performed only for data obtained from host failures and delayed executions of VM boot commands.

Configuration of Nguyen et al.'s model

Nguyen et al. [24] use Eq. (1) as a rule-based model to predict the VM boot time on a host. For each host, Eq. (1) coefficients must be measured and calculated. To do this, we need the I/O time, denoted by α, and CPU time, denoted by β. To calculate the value of α and β, we need the I/O throughput percentage, denoted by x, and the value of CPU resources utilization percentage, denoted by y. The value of y can be obtained from the monitoring process. However, the model only considers local storage; meanwhile, in this study, we use shared storage. Therefore, we calculate the value of α and β without the I/O throughput percentage, and the value of x is set to 0.

For this model, we use the single-host dataset of four hosts, as mentioned in Environment setup section. Using x and y, we calculate the coefficients α and β of the model for the single-host dataset. The coefficients are presented in Table 4. Finally, using the coefficient in Table 2 and the environment setup in Environment setup section, the model's accuracy is calculated.

Table 4 The I/O time (α) and CPU time (β) of Nguyen et al.'s model

compute Host *i*	α_i (sec)	β_i (sec)
Compute 1	2.82	7.75
Compute 2	3	8.43
Compute 3	2.98	8.28
Compute 4	2.4	8.23
Compute 5	1.56	7.60
Compute 6	1.46	8.02
Compute 7	1.46	7.99

Training and testing of ML-based models

This study used three ML-based prediction models: the regression tree model proposed by Govindaraju et al. [1], the random forest regression model, and the linear regression model. The linear regression model is implemented with the "*sklearn*" library of Python. Meanwhile, the regression tree model is implemented with the well-known "*Statistics and Machine Learning Toolbox*" of MATLAB.

Subsequently, the datasets were split into training and test datasets. The three ML-based prediction models were trained using the same datasets and attributes during the training phase. The datasets used were the single-host dataset of four hosts, the single-host dataset of seven hosts, and the multi-host dataset of seven hosts. The datasets' attributes and their descriptions are listed in Table 5. These datasets are available in our git repository [47].

Although the three ML-based prediction models were trained using the same datasets and attributes during the training process, the linear regression model did not use the "*core*" attribute. Meanwhile, we used the "*train_test_split*" function of the "*sklearn*" library to split the dataset for the random forest regression model. We did not normalize or scale the data because the random forest regression and regression tree are non-parametric methods. In addition, we did not perform any transformations of the collected data. For the random forest regression, we set the parameter *n_estimators* to 1000 and the random state to 42. *n_estimators* is the number of trees built by the random forest regression, and the *random_state* is the bootstrap randomness of the samples used when building trees. To observe the relationship between the training dataset size and model performance, we designed 19 cases in which the ratio of the training dataset to the original dataset increased every 5% from 5% to 95%.

The time complexity of random forest regression is measured in two ways: the overall time complexity for

Table 5 Attributes of the dataset

Attributes	Description
eVM	the boot time of the experimental VM (eVM) is the prediction target
wVM	the workload VM (wVM) runs a stress test to compete for computing resources
$c_{i,CPU}$	The number of CPU cores available on host *i*
α_i	I/O and network latency time spent during the boot process of an eVM when there is no resource contention on the selected host *i*
β_i	CPU time spent during the boot process of an eVM when there is no resource contention on the selected host *i*
Q_i	The number of wVMs on host *i*
R_i	The number of eVMs on host *i*
ovm	The sum of the minimum values of R_j and $c_{j,CPU} - 1$ for all other hosts *j*
ovm′	The sum of the maximum values of 0 and $R_j - c_{j,CPU} + 1$ for all other hosts *j*
boot_time	Average VM boot time on host *i*

training a random forest regression tree, and the time complexity for prediction. The time complexity is primarily determined by three factors: (1) the number of training instances, (2) the number of features, and (3) the average depth of the decision tree. The overall time complexity for training a random forest regression is $\mathcal{O}(n\_estimators * m * \log(n))$, where *n_estimators* is the number of trees in the forest, *m* is the number of features, and log(n) is the average depth of the decision tree. In this study, we set the value of *n_estimators* to 1000. Meanwhile, the prediction time complexity is $\mathcal{O}(n\_estimators * \log(n))$ as we apply each new data point to all trees in the forest.

After the training phase, the models were tested with the test dataset (the remaining data). The accuracy of each prediction model is calculated using the following Eq. (3). The percentage of the prediction difference is divided by the actual time. As an example, the accuracy calculation for the random forest regression can be seen in Fig. 8.

$$accuracy = (1 - \frac{|actual\ time - pred\ time|}{actual\ time}) * 100\% \tag{3}$$

Figure 8 shows the random forest regression model performance (accuracy) in each case, where the random forest regression accuracy increased with an increase in the training dataset size. When the training dataset size was 35% of the original dataset, the model accuracy reached 95%. These datasets are available in our git repository [47].

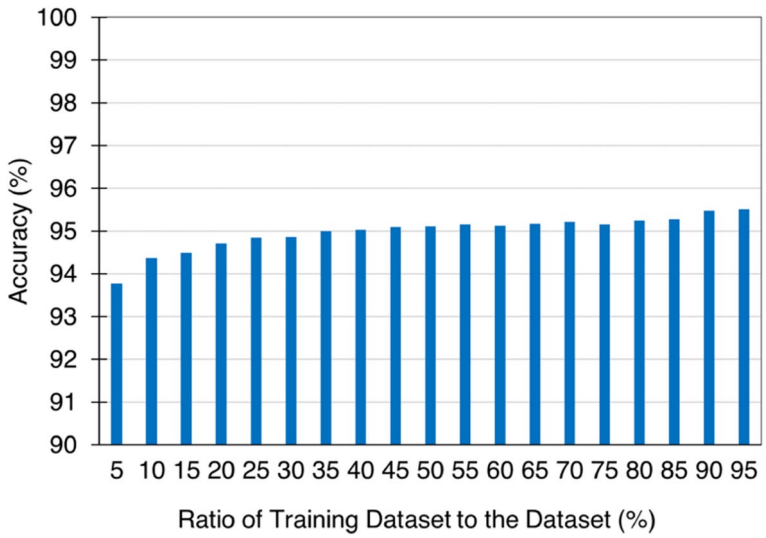

Fig. 8 RF model accuracy for each case, with training dataset sizes ranging from 5% to 95% of the original dataset (in 5% increments)

Performance evaluation

To verify the scalability and performance of the VM boot time prediction models, including a rule-based model and three ML-based models, we performed two sets of experiments in two clusters of different scales with four and seven computing hosts based on the computing architecture shown in Fig. 2. The experimental design in the two clusters is shown in Figs. 9 and 10, as follows:

For each cluster, we performed two sets of experiments. The first experiment, called the single-host experiment, aimed to validate the prediction accuracy with known training dataset values. In the first experiment, we compared the average accuracy of ML-based models using different training dataset sizes. We used the datasets collected as explained in Environment setup section and used 1%, 3%, 5%, 20%, 40%, 60%, and 80% of the dataset as the training datasets and the rest as the test datasets, respectively.

In the second experiment, the average accuracy of each model outside the range of known values was compared. In the second experiment, also called the multi-host experiment, the prediction target was the VM boot time of Compute 4, which was not included in the dataset. By removing the host (Compute 4), new data may be available on the new machine. Therefore, in the second experiment, we aimed to determine whether the model could predict the boot time on Compute 4. Each ML-based model was trained by using 75% of the dataset. Meanwhile, for the rule-based model by Nguyen et al. [24], we only used the coefficient, as shown in Table 4. All experiments followed the experimental protocol described in Experimental protocol section.

Experiments using four compute hosts

In this experiment, we used clusters with four compute hosts (Compute 1, 2, 3, and 4). The dataset used is described in Environment setup section. Additionally, the multi-host experiment in this cluster included cases with 0-6 eVMs on each other host (Compute 1, 2, and 3) and 1-6 eVMs on Compute 4, for a total of 2058 cases. The accuracy of each VM boot time prediction model in the cluster with four compute hosts is shown in Fig. 11.

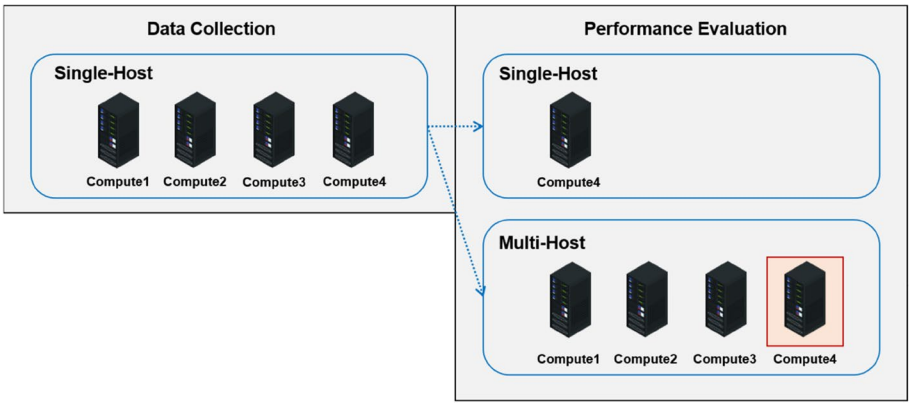

Fig. 9 Experiments in the cluster with four hosts, where Compute 4 is the prediction target for the multi-host evaluation

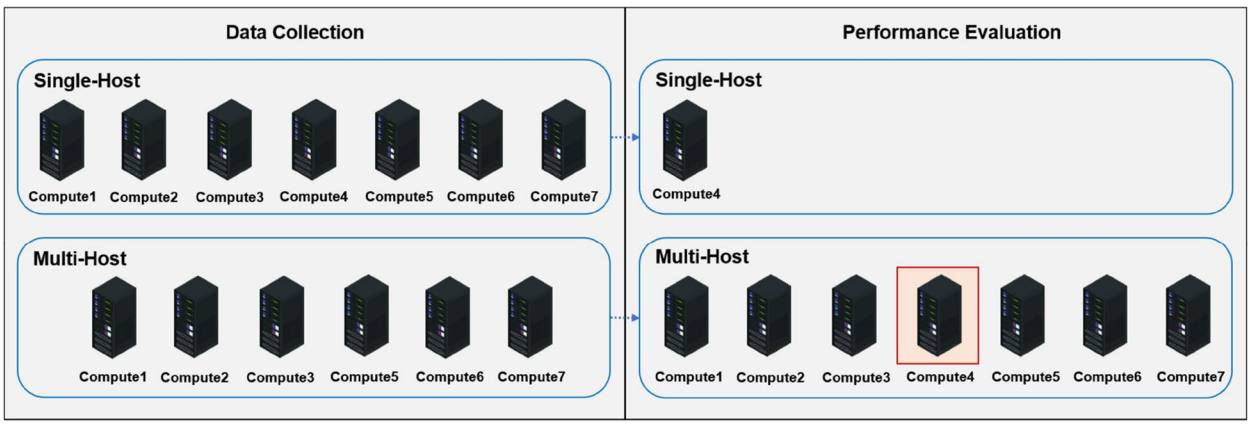

Fig. 10 Experiments in the cluster with seven hosts, where Compute 4 is the prediction target for the multi-host evaluation

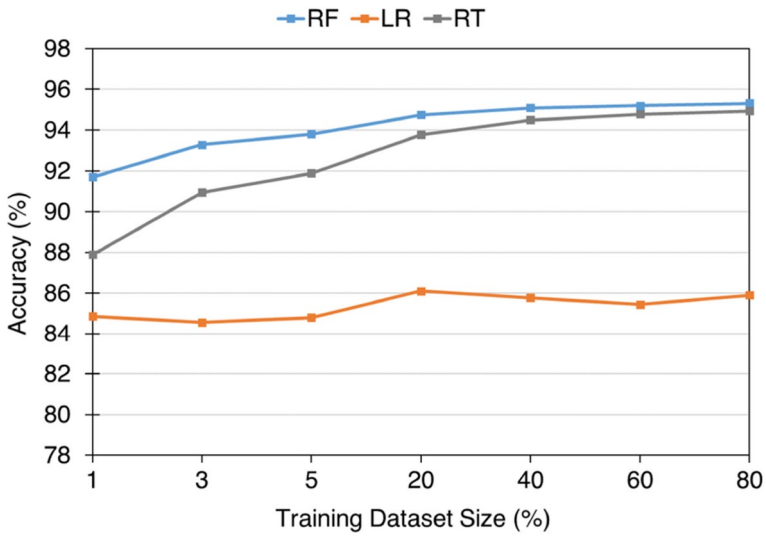

Fig. 11 The average accuracy of ML-based models on the cluster with four compute hosts. (RF: the proposed random forest regression model, LR: the multiple linear regression model, and RT: the regression tree model)

In addition, the actual VM boot times for the cases in the multi-host experiment are available in our git repository [47].

Figure 11 presents the accuracy comparison among ML-based prediction models in the first experiment. This shows that the proposed random forest regression outperforms the linear regression and regression tree models. Random forest regression required only a small amount of data (1% of the dataset) to achieve an average accuracy of 91.55%. Furthermore, random forest regression can achieve an average accuracy of 95.32% using 80% of the training dataset.

Experiments using seven compute hosts

In this experiment, we added three new compute hosts (Compute 5, 6, and 7) to the computing architecture described in Experiments using four compute hosts section. The specifications of these new hosts are listed in Table 6. We limited each host to a maximum of six eVMs. In this cluster, we also have two situations for the experiment: single-host and multi-host experiments with the settings mentioned in Environment setup section.

Similar to the results shown in Experiments using four compute hosts section (Fig. 11), Fig. 12 shows that the random forest regression can still outperform all ML-based prediction models in the cluster with seven hosts regardless of the size of the training dataset. Random forest regression can achieve an average accuracy of 95.16% and 96.57% using datasets of 1% and 80%, respectively.

Discussion

Experimental results summary

In this study, we performed comparative experiments on the performance of four prediction models in clusters of four and seven hosts. The experimental results are summarized in Fig. 13 and Table 7.

Table 7 and Fig. 13 show the performance comparison of four prediction models in clusters of four and seven compute hosts. While the size of the data in a cluster grows as the number of hosts increases, our experiment results indicate that the number of hosts has a minimal impact on the prediction accuracy of random forest regression. The proposed random forest regression has the highest accuracy in the experiment with two clusters of four and seven compute hosts. The random forest regression achieves 94.76% and 96.59% accuracy in four and seven host clusters, respectively. The other two ML-based models, which are regression tree and linear

Table 6 The specifications of the new hosts in the second cluster

Host role	CPU	Memory	Disks	Operation System
Compute 5	Intel i7-6700 (Quad-core)	32 GB	1024 GB HDD	Ubuntu 16.04 Server
Compute 6	Intel E3-1220v6 (Quad-core)	16 GB	1024 GB HDD	Ubuntu 16.04 Server
Compute 7	Intel E3-1220v6 (Quad-core)	16 GB	1024 GB HDD	Ubuntu 16.04 Server

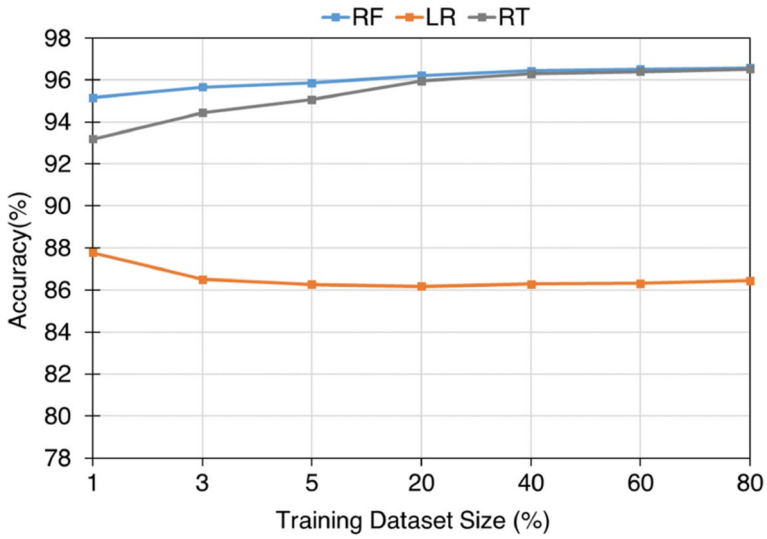

Fig. 12 The average accuracy of each ML-based model in the first experiment on the cluster with seven compute hosts. (RF: the proposed random forest regression model, LR: the multiple linear regression model, and RT: the regression tree model)

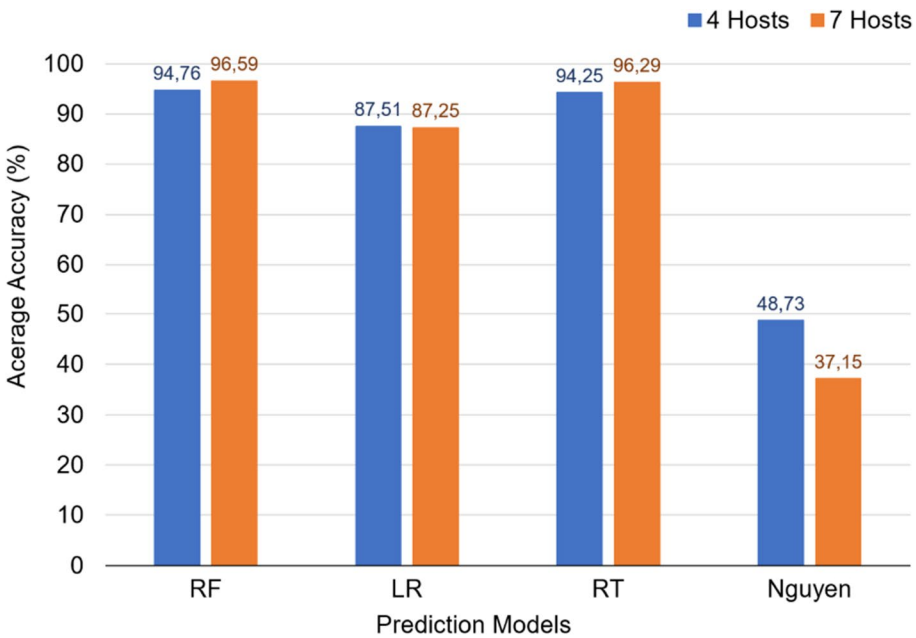

Fig. 13 Average accuracy comparison of four prediction models on the clusters with four and seven hosts. (RF: the proposed random forest regression model, LR: the multiple linear regression model, and RT: the regression tree model)

Table 7 The accuracy of each VM boot time prediction model in clusters with four and seven hosts

Cluster	Experiment	Accuracy (%)			
		Rule-based (Nguyen et al.'s)	Random Forest	Regression Tree	Linear Regression
4 Hosts	Multi-Host	48.73	94.76	94.25	87.51
7 Hosts	Multi-Host	37.15	96.59	96.29	87.25

regression, also perform well in both clusters. The regression tree is the second best, which achieves 94.25% and 96.29% accuracy in four and seven clusters, respectively. Nguyen et al.'s model performs well in the single-host experiment in both clusters with four and seven hosts. However, in the multi-host experiment, the accuracy of Nguyen et al.'s model decreased from 48.73% to 37.15%.

The differences in the accuracy of Nguyen et al.'s model in single-host and multi-host experiments can be attributed to the following reasons. First, Nguyen et al.'s model uses local storage to store VM images. Therefore, the effects of host competition were not considered in the model. Meanwhile, different hosts compete for resources on shared storage, which affects VM boot time. Second, Nguyen et al.'s model does not consider the number of CPU cores of the host. VM boot times are affected if the number of processes is equal to the number of CPU cores.

Based on the values in Table 7, each model's range of accuracy values is presented in Table 8. Table 8 presents the minimum and maximum accuracy values for each model. In the cluster with four hosts, the rule-based model has the lowest minimum and maximum accuracy of 29.85% and 97.50%, respectively. Meanwhile, the proposed random forest regression has the highest minimum and maximum accuracy of 68.66% and 100%, respectively. The regression tree has a similar performance to the random forest regression but with a lower minimum accuracy of 50.19%. Linear regression has a relatively good performance with a slight difference in accuracy values compared to the random forest regression and regression tree. It has the minimum and maximum accuracy of 49.61% and 99.99%. In the cluster with seven hosts, the minimum accuracy of Nguyen et al.'s model decreases from 29.85% to 22.39%. The minimum accuracy of linear regression decreases from 49.61% to 15.61%. Meanwhile, it has similar maximum accuracy with regression tree and random forest regression of 100%.

We have conducted the Wilcoxon test to test whether the random forest regression has a significant accuracy difference from other prediction models on the four and seven-host clusters. This test is commonly used in cloud computing research to determine whether there is a significant difference between the means of the two populations [48–50]. The significance test results are presented in Table 9. As seen in Table 9, the *p-values* in the test for the multi-host experiment in the clusters with four and seven hosts are <0.05. It indicates a significant difference between the proposed random forest regression and all other compared algorithms.

Random forest regression outperforms the other prediction models for the following reasons:

1. **First,** random forest regression uses an ensemble learning method that combines multiple decision trees. Its ensembled decision trees decrease overfitting by reducing the risk of individual tree bias. Moreover, the ensembled decision tree can capture complex relationships to provide high accuracy.
2. **Second,** random forest regression assesses the importance of valuable features for feature selection. This mechanism provides a deeper understanding of the impact of different features and increases the ability of random forest regression to handle the complex relationship of data.

Table 9 *P*-values in the Wilcoxon test corresponding to the average boot time of different models

Cluster	Experiment	*p*-Values of Each VM Boot Time Prediction Model		
		Rule-based (Nguyen et al.'s)	Regression Tree	Linear Regression
4 Hosts	Multi-Host	$< 1 \times 10^{-5}$	1.43×10^{-5}	$< 1 \times 10^{-5}$
7 Hosts	Multi-Host	$< 1 \times 10^{-29}$	9.06×10^{-29}	$< 1 \times 10^{-29}$

Table 8 The range of accuracy values for four prediction models

Cluster	Data Range	Accuracy of Each VM Boot Time Prediction Model (%)			
		Rule-based (Nguyen et al.'s)	Random Forest Regression (RF)	Regression Tree (RT)	Linear Regression (LR)
4 Hosts	Min	29.85%	68.66%	50.19%	49.61%
	Max	97.50%	100%	100%	99.99%
	Avg	48.73%	94.76%	94.25%	87.51%
7 Hosts	Min	22.39%	79.69%	76.91%	15.61%
	Max	88.08%	100%	100%	100%0
	Avg	37.15%	96.59%	96.29%	87.25%

3. **Third,** random forest regression offers robustness over noisy data and outliers, making it suitable for real-world datasets that may have imperfections.

Limitations

Despite its superior performance, the proposed random forest regression model has several limitations:

- **Experiment on a Small-Scale Cluster:** In this study, we used a small-scale cluster for the experimental environment. Meanwhile, a real-world cloud environment typically has a larger scale. Data collection in such an environment is more time-consuming and difficult. For the seven-host cluster, the data collection process requires a lot of time due to many VM placement cases. In total, there are 823,542 cases, each of which takes 3 minutes and 30 seconds to collect, and thus, it takes thousands of days to collect all of the data. Therefore, we used 1% of the overall cases, which is 8,235 cases. For the 8,235 cases, approximately 28 days were required for data collection. For such case coverage, the proposed random forest regression model could still perform well when the cluster scale was increased from four to seven hosts. However, the case coverage should be validated to prove its effectiveness in a larger cluster.
- **Clean Workload:** We use compute hosts with clean or low background workloads. We assume that this factor can contribute to VM boot time prediction for the experiment.
- **Impact of Different Cloud Architecture:** The ML-based models must be retrained at different cloud architectures. Some features in the proposed model may have different weights on prediction accuracy. For example, host competition may not be an important factor in a cloud architecture that does not use a shared storage system for VM booting.

Several strategies can be implemented to address the limitations:

- **Ensemble Techniques:** The ensemble technique combines multiple decision trees. Each decision tree is trained on a bootstrap subset of the data (bagging) and uses a random subset of features for each split (feature bagging) to make a prediction. New data will be passed through each tree in the forest, and the predictions from each tree are combined to produce the final prediction with better accuracy.
- **Hyperparameters Tuning:** The hyperparameters of random forest regression can be tuned to improve predictive ability and processing speed [51]. The number of trees built by the algorithm before taking the average of the predictions should be increased to increase the predictive ability of random forest regression. Then, the bootstrap randomness of the samples used to build the tree can be tuned to increase prediction speed. However, as the predictive ability increases, it consumes more resources and slows down the computation. Therefore, these hyperparameters should be carefully tuned.
- **Data Sampling Technique:** To handle a large dataset, we can use a small subset of data during model development and testing (data sampling). This technique can decrease the time required for model training [52]. However, the data sample should be ensured that it is representative to provide good accuracy.
- **Parallel Processing:** A large amount of data can result in lengthy and difficult model training. One of the preferable solutions is to process large datasets in parallel across multiple machines. By utilizing multiple machines, the processing time speed can be increased [53, 54].
- **Use More Complex Algorithms:** A combination of machine learning algorithms [55] or more complex prediction models that can handle large datasets more effectively may be necessary in certain cases. For example, Reinforcement Learning (RL) methods or Deep Learning (DL) models, such as Recurrent Neural Network (RNN) or Convolutional Neural Networks (CNN) [55–58], can be considered for VM boot time prediction.

Applicability to real-world environment

The applicability of random forest regression can be described as follows:

- **Scalability:** Regarding data size, random forest regression is a suitable option for predicting VM boot time in real-world cloud environments because of its scalability for handling both small and large datasets. Our experimental results show that random forest regression can provide an average accuracy of 96.59% using a smaller dataset size using only 1% case coverage of 823,542 of the total cases. However, it is crucial to maintain efficient case coverage as the cluster size increases. Therefore, validating the case coverage and performance of random forest regression in a larger cluster is necessary.
- **Computational Overhead:** The random forest regression ensembles decision trees; therefore, model training can be computationally expensive when dealing with many decision trees. In our experiments, the training time was still tens of seconds, and

the inference time was approximately 1 ms. However, random forest regression may not be as efficient as rule-based models in terms of inference time, because the inference time is only the time to calculate the equation for the rule-based model. If the rule-based method can improve its accuracy, it can be a better VM boot time prediction solution. However, its accuracy is too low.

In conclusion, random forest regression is generally applicable in a real cloud environment, especially for VM boot time prediction. Its scalability, ensemble nature, and ability to handle various data types make it a preferable model for VM boot time prediction. However, the computational overhead should be considered, especially during model training. The training time can be prolonged, which leads to high computational costs. Moreover, storing such a large dataset requires a significant amount of memory.

Conclusions and future work

Accurate VM boot time prediction is essential to increase the effectiveness and efficiency of resource management, decrease costs, and increase the performance of a cloud platform. In this study, we analyzed several factors that influence VM boot time prediction: the number of CPU cores of a host, the number of VMs, and the competition between hosts. We used these factors to build a random forest regression model for VM boot time prediction. We then compared its performance to a rule-based prediction model and two ML-based regression models, linear regression and regression tree. We performed two sets of experiments in clusters with four and seven hosts. According to the experimental results, random forest regression outperformed the other prediction models by providing an average accuracy of 94.76% and 96.59% in four and seven host clusters by using only a small case coverage of 1%. Our findings have several practical implications as follows:

- Using the most important factors to predict VM boot time accurately, we can guarantee the resource, cost, and energy allocation effectiveness. For example, in high availability (HA), an accurate VM boot time prediction allows us to choose the least VM boot time to evacuate failed VMs on a cluster.
- By providing insight into the performance of different models, users can choose the best model for VM boot time prediction. Implementing better prediction models can help maintain resource efficiency, cost-effectiveness, and overall performance of cloud platforms.

The VM boot time can vary significantly depending on a specific configuration and environment. Further research can address this issue and propose potential solutions for improving VM boot time prediction.

- **Consideration of Different Prediction Models:** Reinforcement Learning (RL) methods and popular Deep Learning (DL) models, such as Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN), can be considered in VM boot time prediction. Currently, these models have been implemented for predictions in many cloud-computing-related studies to obtain refined prediction models [55–58].

  – RL uses agents to make decisions by interacting with the environment and then receives feedback through rewards or penalties [58]. It has several advantages, such as flexibility in decision-making in a dynamic and complex environment where explicit programming or rules may be challenging. RL agents are also suitable for tasks where optimal strategy may evolve since they learn from experience and can adapt to environmental change. However, RL can be challenging to set up because the accuracy depends on the state, action, and gain functions. Moreover, training RL models can be more computationally expensive than ML-based models [58].
  – DL is inspired by the structure and function of the human brain, and it learns patterns and representations using an artificial neural network. DL has several advantages, including adaptability and performance on large datasets [59]. However, DL can be computationally expensive and complex during training and inference, making it unsuitable for resource-constrained environments or real-time requirements. Furthermore, DL typically requires a large amount of data, and its performance may suffer when the available data is limited [59].

- **Implementing Different Virtualization Technologies and Environments:** Different virtualization technologies, such as deduplication and boot-from-snapshots, can be considered in future studies. In addition, the arrangement of network components, such as switches and routers, is also related to VM boot time prediction. Moreover, a larger-scale environment could be considered to validate whether the proposed approach is scalable. Therefore, exploring those aspects of VM boot time prediction can be a potential future direction.

Authors' contributions
Ridlo Sayyidina Auliya was responsible for paper writing, validation, paper survey, and experiments; Yen-Lin Lee was responsible for paper writing, algorithm implementation, paper survey, and experiments; Chia-Ching Chen was responsible for conceptualization and validation; Deron Liang was responsible for problem definition and project supervision; Wei-Jen Wang was responsible for conceptualization and paper editing.

Funding
This study was supported by the National Science and Technology Council of Taiwan, under grants 111-2221-E-008-061- and 111-2221-E-008-059-, and the Software Research Center, National Central University.

Availability of data and materials
The datasets used in this manuscript are available on our GitHub repository [47].

Declarations

Ethics approval and consent to participate
Not applicable.

Competing interests
The authors declare no competing interests.

Received: 16 August 2023 Accepted: 23 March 2024
Published online: 04 April 2024

References

1. Govindaraju Y, Duran-Limon HA, Mezura-Montes E (2021) A regression tree predictive model for virtual machine startup time in IaaS clouds. Cluster Comput 24:1217–1233. https://doi.org/10.1007/s10586-020-03169-0
2. García-Valls M, Cucinotta T, Lu C (2014) Challenges in real-time virtualization and predictable cloud computing. J Syst Archit 60(9):726–740. https://doi.org/10.1016/j.sysarc.2014.07.004
3. Alhazmi K, Sharkh MA, Shami A (2018) Drawing the cloud map: Virtual network provisioning in distributed cloud computing data centers. IEEE Syst J 12(2):1480–1491. https://doi.org/10.1109/JSYST.2015.2484298
4. Amazon (2023) Amazon EC2. https://aws.amazon.com/ec2/. Accessed May 2022
5. Microsoft (2023) Microsoft Azure. https://azure.microsoft.com/en-us. Accessed May 2022
6. Linux (2023) Kernel Virtual Machine. https://www.linux-kvm.org/page/Documents. Accessed Aug 2023
7. VMWare vSphere (2023) VMWare vSphere. https://docs.vmware.com/en/VMware-vSphere/index.html. Accessed Aug 2023
8. da Rosa Righi R, Rodrigues VF, Da Costa CA, Galante G, De Bona LCE, Ferreto T (2015) Autoelastic: Automatic resource elasticity for high performance applications in the cloud. IEEE Trans Cloud Comput 4(1):6–19. https://doi.org/10.1109/TCC.2015.2424876
9. Ghobaei-Arani M, Souri A, Baker T, Hussien A (2019) ControCity: An autonomous approach for controlling elasticity using buffer management in cloud computing environment. IEEE Access Pract Innov Open Solutions 7:106912–106924. https://doi.org/10.1109/ACCESS.2019.2932462
10. Lee YC, Zomaya AY (2012) Energy efficient utilization of resources in cloud computing systems. J Supercomput 60:268–280. https://doi.org/10.1007/s11227-010-0421-3
11. Zhang F, Liu G, Fu X, Yahyapour R (2018) A survey on virtual machine migration: Challenges, techniques, and open issues. IEEE Commun Surv Tutorials 20(2):1206–1243. https://doi.org/10.1109/COMST.2018.2794881
12. Kim MH, Lee JY, Raza Shah SA, Kim TH, Noh SY (2021) Min-max exclusive virtual machine placement in cloud computing for scientific data environment. J Cloud Comput 10:2. https://doi.org/10.1186/s13677-020-00221-7
13. Çağlar İ, Altılar DT (2022) Look-ahead energy efficient VM allocation approach for data centers. J Cloud Comput 11:11. https://doi.org/10.1186/s13677-022-00281-x
14. Javadpour A, Nafei A, Ja'fari F, Pinto P, Zhang W, Sangaiah K (2023) An intelligent energy-efficient approach for managing IoE tasks in cloud platforms. J Ambient Intell Human Comput 14:3963–3979. https://doi.org/10.1007/s12652-022-04464-x
15. Javadpour A, Sangaiah AK, Pinto P, Ja'fari F, Zhang W, Abadi AMH, Ahmadi H (2023) An energy-optimized embedded load balancing using DVFS computing in cloud data centers. Comput Commun 197:255–266. https://doi.org/10.1016/j.comcom.2022.10.019
16. Javadpour A, Wang G, Rezaei S (2020) Resource Management in a Peer-to-Peer Cloud Network for IoT. Wirel Pers Commun 115:2471–2488. https://doi.org/10.1007/s11277-020-07691-7
17. Zhou Z, Shojafar M, Alazab M, Abawajy J, Li F (2021) AFED-EF: An energy-efficient VM allocation algorithm for IoT applications in a cloud data center. IEEE Trans Green Commun Netw 5(2):658–669. https://doi.org/10.1109/TGCN.2021.3067309
18. Zhou Z, Li F, Zhu H, Xie H, Abawajy JH, Chowdhury M (2020) An improved genetic algorithm using greedy strategy toward task scheduling optimization in cloud environments. Neural Comput Appl 32:1531–1541. https://doi.org/10.1007/s00521-019-04119-7
19. Zhou Z, Abawajy J, Chowdhury M, Hu Z, Li K, Cheng H, Alelaiwi AA, Li F (2018) Minimizing SLA violation and power consumption in Cloud data centers using adaptive energy-aware algorithms. Futur Gener Comput Syst 86:836–850. https://doi.org/10.1016/j.future.2017.07.048
20. Zhou Z, Shojafar M, Abawajy J, Yin H, Lu H (2021) ECMS: An edge intelligent energy efficient model in mobile edge computing. IEEE Trans Green Commun Netw 6(1):238–247. https://doi.org/10.1109/TGCN.2021.3121961
21. Kampa T, El-Ankah A, Grossmann D (2023) High Availability for virtualized Programmable Logic Controllers with Hard Real-Time Requirements on Cloud Infrastructures. In: 2023 IEEE 21st International Conference on Industrial Informatics (INDIN), Lemgo, Germany, 2023. pp. 1-8. https://doi.org/10.1109/INDIN51400.2023.10218014
22. Nitu V, Olivier P, Tchana A, Chiba D, Barbalace A, Hagimont D, Ravindran B (2017) Swift birth and quick death: Enabling fast parallel guest boot and destruction in the xen hypervisor. ACM SIGPLAN Not 52(7):1–14. https://doi.org/10.1145/3140607.3050758
23. Costache S, Parlavantzas N, Morin C, Kortas S (2013) On the use of a proportional-share market for application slo support in clouds. In: Euro-Par 2013 Parallel Processing: 19th International Conference, Aachen, Germany, August 26-30, 2013. Proceedings 19. Springer Berlin Heidelberg, pp 341–352. https://www.hal.inserm.fr/INRIA/hal-00821558. Accessed Aug 2023
24. Nguyen TL, Lebre A (2017) Virtual machine boot time model. In: 2017 25th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP). Presented at the 2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP), St. Petersburg, Russia. https://doi.org/10.1109/PDP.2017.58
25. Abrita SI, Sarker M, Abrar F, Adnan MA (2019) Benchmarking vm startup time in the cloud. In: Benchmarking, Measuring, and Optimizing: First BenchCouncil International Symposium, Bench 2018, Seattle, WA, USA, December 10-13, 2018, Revised Selected Papers 1. Springer International Publishing, pp 53–64. https://doi.org/10.1007/978-3-030-32813-9_6
26. Mao M, Humphrey M (2012) A performance study on the VM Startup time in the cloud. In: 2012 IEEE Fifth International Conference on Cloud Computing. Presented at the 2012 IEEE 5th International Conference on Cloud Computing (CLOUD), Honolulu, HI, USA. https://doi.org/10.1109/CLOUD.2012.103
27. Wu H, Ren S, Garzoglio G, Timm S, Bernabeu G, Chadwick K, Noh SY (2016) A reference model for virtual machine launching overhead. IEEE Trans Cloud Comput 4(3):250–264. https://doi.org/10.1109/TCC.2014.2369439
28. IBM (2023) Boot from Volume. https://www.ibm.com/docs/es/cic/1.1.1?topic=planning-boot-from-volume. Accessed Oct 2023
29. OpenStack (2023) Images and Instances. https://docs.openstack.org/glance/train/admin/troubleshooting.html. Accessed Oct 2023
30. Block87 (2021) Booting ISO's in OpenStack Environments https://blog.andyserver.com/2021/06/booting-iso-in-openstack-environments/. Accessed Oct 2023
31. Crago SP, Dunn K, Eads P, Hochstein L, Kang DI, Kang M, Walters JP (2011) Heterogeneous cloud computing. In: 2011 IEEE International Conference on Cluster Computing. Presented at the 2011 IEEE International Conference on Cluster Computing (CLUSTER), Austin, TX, USA. https://doi.org/10.1109/CLUSTER.2011.49

32. Zahran M (2016) Heterogeneous computing: Here to stay. Queue 14:31–42. https://doi.org/10.1145/3028687.3038873
33. Crago SP, Walters JP (2015) Heterogeneous cloud computing: The way forward. Computer 48(1):59–61
34. Parthasarathi R (2018) Warehouse-Scale Computers in Computer Architecture: Engineering and Technology. https://www.cs.umd.edu/~meesh/411/CA-online/chapter/warehouse-scale-computers/index.html. Accessed May 2022
35. Razavi K, Razorea LM, Kielmann T (2014) Reducing VM Startup Time and Storage Costs by VM Image Content Consolidation. In: Euro-Par 2013: Parallel Processing Workshops. Euro-Par 2013. Lecture Notes in Computer Science, vol 8374. Springer, Berlin, Heidelberg. https://comsec.ethz.ch/wp-content/files/dihc13.pdf. Accessed Aug 2023
36. Schmidt M, Fallenbeck N, Smith M, Freisleben B (2010) Efficient distribution of virtual machines for cloud computing. In: 2010 18th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), Pisa, Italy. https://doi.org/10.1109/PDP.2010.39
37. OpenStack (2023) Launch an instance from a volume.https://docs.openstack.org/nova/zed/user/launch-instance-from-volume.html. Accessed May 2022
38. OpenStack (2023) OpenStack Documentation. https://docs.openstack.org/zed/. Accessed May 2022
39. 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 Exper 41:23–50. https://doi.org/10.1002/spe.995
40. Saxena D, Gupta R, Singh AK, Vasilakos AV (2023) Emerging VM Threat Prediction and Dynamic Workload Estimation for Secure Resource Management in Industrial Clouds. IEEE Trans Autom Sci Eng. https://doi.org/10.1109/TASE.2023.3319373
41. Li Y, Ou D, Jiang C, Shen J, Guo S, Liu Y, Tang L (2020) Virtual machine performance analysis and prediction. In: 2020 International Conference on Communications, Computing, Cybersecurity, and Informatics (CCCI), Sharjah, United Arab Emirates. https://doi.org/10.1109/CCCI49893.2020.9256518
42. Gao J, Wang H, Shen H (2020) Machine learning based workload prediction in cloud computing. In: 2020 29th international conference on computer communications and networks (ICCCN). IEEE, pp 1–9. https://doi.org/10.1109/ICCCN49398.2020.9209730
43. Moreno-Vozmediano R, Montero RS, Huedo E, Llorente IM (2019) Efficient resource provisioning for elastic cloud services based on machine learning techniques. J Cloud Comput 8(1):1–18. https://doi.org/10.1186/s13677-019-0128-9
44. RightScale (2017) RightScale 2017 State of the Cloud Report Uncovers Cloud Adoption Trends. https://www.globenewswire.com/news-release/2017/02/15/1208194/0/en/RightScale-2017-State-of-the-Cloud-Report-Uncovers-Cloud-Adoption-Trends.html. Accessed May 2022
45. Bolte M, Sievers M, Birkenheuer G, Niehorster O, Brinkmann A (2010) Non-intrusive virtualization management using libvirt. In: 2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010), Dresden. https://doi.org/10.1109/DATE.2010.5457142
46. Both D (2020) Linux Boot and Startup. In: Using and Administering Linux, vol 1. Apress, Berkeley, pp 451–490. https://link.springer.com/book/10.1007/978-1-4842-5049-5
47. Lee YL (2022) Repository for experimental data related to average VM boot time. https://github.com/Ncu-software-research-center/NCU-VMDataset. Accessed Aug 2023
48. Bezdan T, Zivkovic M, Bacanin N, Strumberger I, Tuba E, Tuba M (2022) Multi-objective task scheduling in cloud computing environment by hybridized bat algorithm. J Intell Fuzzy Syst 42(1):411–423. https://doi.org/10.3233/JIFS-219200
49. Putrada AG, Abdurohman M, Perdana D, Nuha HH (2023) EdgeSL: Edge-Computing Architecture on Smart Lighting Control with Distilled KNN for Optimum Processing Time. IEEE Access. https://doi.org/10.1109/ACCESS.2023.3288425
50. Thakur A, Goraya MS (2022) RAFL: A hybrid metaheuristic based resource allocation framework for load balancing in cloud computing environment. Simul Model Pract Theory 116:102485. https://doi.org/10.1016/j.simpat.2021.102485
51. Probst P, Wright MN, Boulesteix AL (2019) Hyperparameters and tuning strategies for random forest. Wiley Interdiscip Rev Data Min Knowl Disc 9(3):e1301. https://doi.org/10.1002/widm.1301
52. Paing MP, Pintavirooj C, Tungjitkusolmun S, Choomchuay S, Hamamoto K (2018) Comparison of sampling methods for imbalanced data classification in random forest. In: 2018 11th Biomedical Engineering International Conference (BMEiCON). IEEE, pp 1–5. https://doi.org/10.1109/BMEiCON.2018.8609946
53. Chen J, Li K, Tang Z, Bilal K, Yu S, Weng C, Li K (2016) A parallel random forest algorithm for big data in a spark cloud computing environment. IEEE Trans Parallel Distrib Syst 28(4):919–933. https://doi.org/10.1109/TPDS.2016.2603511
54. Genuer R, Poggi JM, Tuleau-Malot C, Villa-Vialaneix N (2017) Random forests for big data. Big Data Res 9:28–46. https://doi.org/10.1016/j.bdr.2017.07.003
55. Zhou Z, Shojafar M, Alazab M, Li F (2022) IECL: an intelligent energy consumption model for cloud manufacturing. IEEE Trans Ind Inform 18(12):8967–8976. https://doi.org/10.1109/TII.2022.3165085
56. Leka HL, Fengli Z, Kenea AT, Hundera NW, Tohye TG, Tegene AT (2023) PSO-Based Ensemble Meta-Learning Approach for Cloud Virtual Machine Resource Usage Prediction. Symmetry 15(3):613. https://doi.org/10.3390/sym15030613
57. Nam S, Yoo JH, Hong, JWK (2022) VM Failure Prediction with Log Analysis using BERT-CNN Model. In 2022 18th International Conference on Network and Service Management (CNSM). IEEE, pp 331–337. https://doi.org/10.23919/CNSM55787.2022.9965187
58. Nian R, Liu J, Huang B (2020) A review on reinforcement learning: Introduction and applications in industrial process control. Comput Chem Eng 139:106886. https://doi.org/10.1016/j.compchemeng.2020.106886
59. Jauro F, Chiroma H, Gital AY, Almutairi M, Shafi'i MA, Abawajy JH (2020) Deep learning architectures in emerging cloud computing architectures: Recent development, challenges and next research trend. Appl Soft Comput 96:106582. https://doi.org/10.1016/j.asoc.2020.106582

Publisher's Note