SN Computer Science (2023) 4:86
https://doi.org/10.1007/s42979-022-01504-3

ORIGINAL RESEARCH

A Reliable Client Detection System during Load Balancing for Multi-tenant Cloud Environment

Ashutosh Kumar Singh[1] · Sakshi Chhabra[2] · Rishabh Gupta[1] · Deepika Saxena[1]

Received: 7 January 2022 / Accepted: 10 November 2022 / Published online: 9 December 2022

Abstract

Cloud computing is a broad-scale distributed computing system and is a widely accepted archetype that has many luxurious features, including traffic scalability, resource allocation, accessibility, inter-communication cost, and many more. Security is considered one of the primary concerns in the cloud. This paper chooses to formulate this problem by improving the Virtual Machine (VM) allocation policy. The authors have enhanced this policy from a different perspective by maintaining the difficulty in co-tenancy between attackers and targets. A secure resource allocation mechanism has been proposed to prevent multi-tenancy attacks, where attackers and the target are co-tenants on the same server. The multi-objective approach is implemented for a secure load balancing model called reliable client detection system (RCDS). This model inquires the safe or unsafe states all along VM distribution which accomplishes and estimates the reliability of the clients as per the historical performances. When cloud data centers have received the demands to deploy the upcoming jobs, the introduced model helps to find a secure physical machine for balancing the load with avoiding threats. It is evident from the results that RCDS can effectively diminish the risks and security score when increased from 100 to 1000 numbers of cloudlets under the safe states. Performance evaluation demonstrates that RCDS achieves high throughput, avoids traffic overflow, and reduces traffic up to 33.37% in the network.

Keywords Cloud computing · Multi-tenancy · Allocation policy · Security · Resource utilization

Introduction

According to International Data Corporation (IDC), the investment made in Cloud IT infrastructure increased by 50.9% in 2020 with an amount of $37.7 billion [1]. In 2021, IDC expected that public cloud service providers (CSPs) would spend $74.6 billion on IT infrastructure for delivering services. With the incorporation of cloud infrastructure, 60–70% of the costs can be saved through data transfer by clients to the cloud. The cloud is a service-oriented computing model that facilitates resource sharing, on-demand provisioning with various computing and IT services. It further helps to reduce the initial investment, maintenance cost, and operating cost. Figure 1 illustrates the glimpse of cloud data center networks where individuals or cloud customers request their workload for the optimal task deployment under the control of cloud service providers. The resource controller holds the workload and organizes the proper scheduling of approaching applications by cloud workload management portal. The performance of the cloud data centers is improved or optimized by scheduling with resource pool chunks. It is just simple with more flexibility to make the required changes for better resource scheduling at any time or anywhere [2]. However, infrastructure resource sharing among multiple tenants establishes new security challenges. Sharing intimate information among cloud users has always been dangerous because data is spread over numerous storage devices under the same infrastructure. This type of threat, known as a multi-tenancy attack, is identified as an

✉ Rishabh Gupta
rishabhgpt66@gmail.com

Ashutosh Kumar Singh
ashutosh@nitkkr.ac.in

Sakshi Chhabra
sakshichhabra555@gmail.com

Deepika Saxena
13deepikasaxena@gmail.com

[1] Department of Computer Applications, National Institute of Technology, Kurukshetra, Haryana, India

[2] Department of Computer Applications, Panipat Institute of Engineering and Technology, Samalkha, Haryana, India

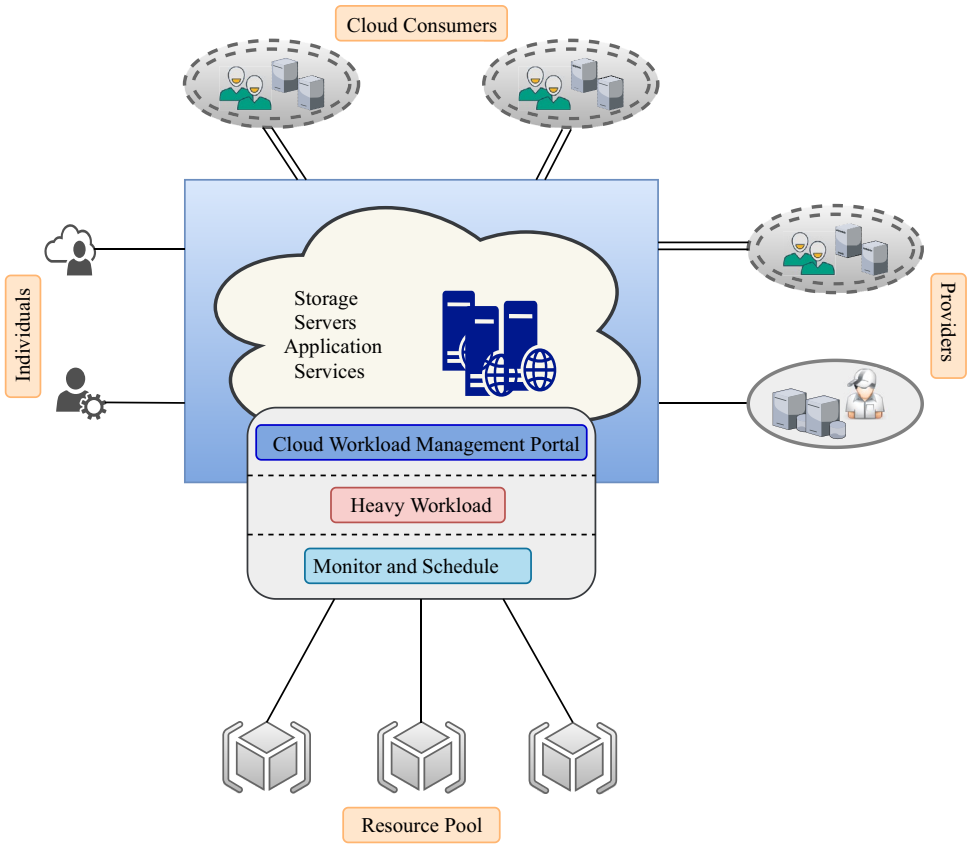

Fig. 1 View of cloud data center networks

adequate attack over the cloud infrastructure where various tenants are located on the same server. Many algorithms have been proposed to maximize the usage of resources and reduce power consumption with secure allocation during task deployment. However, security issues that defend against multi-tenancy attacks during load management are rarely addressed. These attacks can be avoided before their execution by terminating the malicious VMs if they are recognized. But it becomes very difficult to attain it in real cloud scenarios.

A secure allocation strategy called reliable client detection system (RCDS) is proposed in this paper, which helps to assign safe VMs to the upcoming jobs in a secure way to prevent malicious users along with normal users from achieving multi-tenancy. Security and Load Balancing has been considered for measuring the safety of VMs during allocation policy to defend against multi-tenancy attacks [3, 4]. It tries to minimize the number of servers shared among multiple servers. This paper suspects two states: Normal and Safe state. The normal state is not formulated any security parameters. In the case of safe state, the procedure is fixed for VM Selection and VM Placement under allocation policy for security benefits. The mapping for users and servers are conducted to reduce the number of physical hosts shared between different users. The load-balanced manager can decide the safe VMs for the deployment of upcoming jobs and ensure the minimum attack possibilities and efficient resource utilization. Time-shared or space-shared metrics are used for resource provisioning VM allocation policies. Extensive experiments have been conducted on Cloudsim, a simulated cloud environment where modeling and simulation can be done as a real cloud for attaining realistic results. In this work, we aim to provide secure and resource efficient cloud resource provisioning framework which ensures the security, quality of service (QOS) with least violation rate. We demonstrate that RCDS scheduling either time shared or time shared policy not only facilitates the cloud consumers by resources availability and improves throughput, response time etc. but also maximizes the cloud profit with less resources utilization and prevent the multi-tenancy attacks. The paper aims to propose a VM management policy which reduces the chances of ordinary VMs co-resident with malicious VMs which helps to address the security along with energy consumption and effective resource utilization. The real contribution of RCDS are summarized as follows:

- RCDS considers the dynamic request streams while simulating a real cloud environment to achieve better security during load balancing. By finding reliable clients, we try to reduce the number of co-tenant machines being executed. It also minimizes the number of working hosts which run on the same malicious hypervisor.

- RCDS guarantees the algorithm's scalability, no special changes are required in the guest hypervisor OS. The infrastructure can be expanded or secure to handle the increased load and converted into a simple load management scheme to secure load management.
- RCDS migrates running instances of the VMs from one physical host to another for optimizing cloud utilization. It inquires the qualified hosts among all, which accomplishes the optimal allocation scheme that also addresses the security aspects along with optimal VM allocation.

The remainder of this paper is organized as follows: In "Related Work", a review of related studies of the existing approaches for achieving secure load balancing in cloud data centers is presented. "Reliable Client Detection System" introduced the reliable client detection system (RCDS) with the problem definitions and some assumptions. "Performance Evaluation" discussed the performance evaluation obtained from the practical implementations. Finally, the conclusion and future directions of the work are discussed in "Conclusion". Table 1 shows the list of symbols with their descriptive terms that have been used in this manuscript.

Related Work

Several methods have been developed to prevent malicious users, which are sharing the resources with legal users [3]. These approaches are mainly concerned with reducing side-channel attacks and applying blinding techniques to minimize the amount of leaked data. However, after using the virtualized platform, service users must be aware of the security risk posed by co-resident attacks (also known as co-location attacks). A simultaneous multi-threading (SMT) free approach proposed by Yinqian Zhang [5], introduced with the extraction of fine-grained information from a victim VM running on the same server. These types of attacks happened on asymmetric microprocessing virtualized systems. The proposed approach handled several issues, including noise filtering, core migrations, and obtaining the victim's key. But, the problem is worse when processes collude. A Nomad system, which helps mitigate the arbitrary cloud side-channel attacks and provides a vector-agnostic defense against known and future side-channels proposed by Soo-Jin Moon et al. [6]. The proposed system is a probabilistic defense that keeps two tenants from being co-scheduled on the same host for long periods to operate at the cloud scheduler level. In shared cloud deployments, it captured the information leakage model by channels and required migration heuristics across VMs. Defending against fine-grained attacks with a probabilistic approach is problematic. Qian Sun [7] proposed a model called SeLance, which analyzes the security threats, and mitigates the load balancing risks. The proposed model predicts the information leakage during the entire VM migration and VM placement process. But, it still results in multiple VM migrations because the relationship of co-resident VMs is ignored. To defend against co-resident attacks in cloud computing, an approach was proposed by Han et al. [8]. This approach is based on an improved VM allocation policy to make it difficult for attackers to co-locate on VMs. The authors have used three metrics: Efficiency, Coverage, and VM_{min} to calculate the results. However, the malicious users may complete side-channel attacks to the target VM before it is classified because this approach acquires the behavior features of the tenants VMs in advance. Duan et al. [9] proposed a load balancing and multi-tenancy-oriented data center virtualization framework, which generated multi-tenancy-oriented private clouds and allowed multiple VMs to communicate with others under the physical hosts. The proposed framework was able to achieve global load balancing on the underlying physical networks. However, it only concentrated on load balancing to improve the system's performance or minimize task response time without considering the prevention of the attacks. To attain efficient resource provisioning for optimal workload allocation, a stochastic multi-tenanted framework was proposed by Zhuoyao Wang et al. [10]. The authors have focused on user-facing performance measures and devised a resource provisioning algorithm, called max-min-cloud algorithm, to reduce the mean of the stochastic response time of users' requests. The proposed framework increases the cloud's service performance, but it necessitates more CPU cores for high-performance computing tasks. Deng et al. [11] proposed a systematic framework to investigate power consumption and network delay trade-off in the cloud and fog environment. A Hungarian method and the generalized benders decomposition (GBD) approach were used to solve a workload allocation problem that was split into primary and sub-problems. Furthermore, the huge data sets obtained from various locations are heterogeneous, inconsistent, high

Table 1 List of terminologies with their explanatory terms

VM	Virtual machine	Cl_i	Clients
M_i	Memory consumption	$ph_{s,m}$	Physical host
δt	Past time	h	Host
t	Tenant	OS	Operating system
THR_F	Threshold time	R	Unknown hosts
$Favorable_{i(t)}$	Favorable VMs	$\perp^i_{s,m}$	Binary variable
ϑ	Job requests	B_s	Bandwidth
C_i	CPU consumption	$X_{s,m}$	CPU resource
CSPs	Cloud service providers	$Y_{s,m}$	Memory resource
$KnwHost_{t,i,h}(\delta t)$	Known hosts	Ⓢ	Server
$Unsafe_{t\rightarrow t)}(\delta t, tm)$	Unsafe state		

redundancy, and meaningless. A set of security-constrained energy-efficient optimization strategies was proposed by Fernandez-Cerro et al. [12] for task scheduling and hibernating VMs. A task service model was developed, which combines energy and time-based criteria to sleep idle resources. This model reduces makespan and power consumption. However, it has high computation cost because of the optimization of security operations. Singh and Kumar [13] proposed a secure and energy-aware load balancing framework called SEA-LB, which is based on the genetic algorithm (GA) approach to introduce the security concept by minimizing the number of conflicting servers along with power-saving and efficient resource utilization. The proposed framework reduces resource utilization to provide security to each VM. A drawback of this framework is that it usually leads to premature convergence. An online secure communication model for Cloud (OSC-MC) was presented in [14], which provides secure VM intercommunication and avoids illegal access-based VM threats. The proposed model minimizes network hogs and power consumption, but it cannot handle servers' over/under-loading and *VM* migration during *VM* scheduling. Saxena et al. [22] proposed a secure and multiobjective virtual machines placement (SM-VMP) framework, which unanimously achieves the feasible and optimal VMP to serve cloud users and service providers' perspectives. The proposed Whale Optimization Genetic Algorithm (WOGA), based on non dominated sorting-based genetic algorithm and evolutionary whale optimization, was used to carry out the VMP. The proposed framework reduced communication cost, security threats, and power consumption. However, the framework did not consider VM mapping to the level of numerous server clusters. Mohamed Yassin et. al. [21] presented the framework of inter-tenant attack detection for web-based running applications in SaaS cloud environment. The SQL syntactic analysis has been used to detect and prevent the attacks along with key management policy. It runs on web based AWS services and improves the accuracy, portability, integrity as well as confidentiality with minimum response time. Table 2 gives a summarized comparison of most noticeable work done on optimal and secure VM allocation techniques which is highly cited by the authors. The several classifications has been discussed i.e. Problem addressed, Outcome, Algorithm, Implementation platform and Limitation. The problem is addressed clearly with specific algorithm policies or strategy. The parameters used and improvements/limitations have discussed about each respective papers.

All existing works mentioned above worked against the resource scheduling, optimal and secure VM allocation techniques. Though different from the previous studies [25–32], the authors focus on a real-time application request types with secure VM placement with optimal resource utilization. The model allocates an appropriate resources to the VMs during secure load balancing. Through analysis of our proposed algorithm, the secure VM allocation scheme is found suitable and inquires the qualified hosts among all which accomplishes the optimal allocation scheme that also addresses the security aspects along with optimal VM allocation.

Reliable Client Detection System

Definitions

Muti-tenancy threat: The sharing of resources and services to perform simulation, computation, and storage, serving multiple consumers and client organizations, i.e., tenants, are known as multi-tenancy [23]. It appears when two cloud users adopt the same cloud as well as the same physical machine concurrently for the deployment allocated by CSP's. Cloud service providers must allow virtualization and resource sharing for multi-tenancy, as shown in Eq. (1).

$$\text{MultiTenancy} = \text{Virtualization} + \text{Resourcesharing} \quad (1)$$

Figure 2 depicts the multi-tenant threat scenario where resources are divided between the target machine and the attacker machine.

The load management helps to cut down the number of resources suggests how to allocate the upcoming requests safely so that the chances of the attackers are less. It improves the concurrent user capacity as well as the overall reliability of applications. However, the problem of deploying co-tenant VMs is curtailed by the LB allocation policy.

Hypervisor attack: The hypervisor, also known as virtualization manager is the one who helps to run multiple operating systems (OS)s on a single physical host. It is a threat where the attacker gains the program's susceptibilities to share a single hardware processor. It mostly manages the scheduling of VMs for execution and sets the configuration parameters for VM images and states. It can be seen in Fig. 2 that the attacker engaged the full hypervisor during load balancing. Whenever the resource manager sends the particular VM to the attacker's host, this VM will be easily targeted by attackers.

Favorable & non-favorable VMs: It is assumed that the intrusion detection system (IDS) takes this much THR_F time to identify the malicious ones among every co-resident VMs. If the time of two machines is being co-tenant for less than THR_F then these are unfavorable VMs. If time δt is greater or equal and the intrusion detection system did not alarm, then it is stated that there is no malicious system in these

Table 2 Summarized view of work done on most noticeable papers in optimal and secure VM allocation techniques

S no.	Problem adderessed	Outcome	Method	Implementation platform	Limitation
1.	A unified optimal resource allocation framework for networked clouds [15]	Virtual network embedding (virtual nodes and links)	Mapping revenue, mapping cost, resource utilization, virtual topology	H/w: server, router, its operating system (Windows, Linux, Android, Solaris, JUNOS, etc.), and its virtualization environment (Xen, VMware, KVM, JUNOS specific, etc.) S/w: CVI-sim, Emulation environment	Energy consumption is not considered and does not provides satisfactory solution for scalable complex problems
2.	A security and budget aware workflow scheduling in cloud [16]	Bandwidth and storage	Clustering, priortization, tasks based on data dependency	H/w: Intel Core i7-3770 CPU at 3.40 GHz with 8-GB memory S/w: Java	Execution cost and energy consumption is not considered
3.	A virtualization technology to allocate data center resources dynamically [17]	VM Monitors, DNS servers	Skewness Metric, Load prediction, energy saving	H/w: Windows Platform, 1024GB storage S/w: Cloudsim 3.0	Overload Avoidance and Green Computing for systems with multi-resource constraints
4.	A policy aware VM management scheme for cloud data centers [18]	Server Hypervisor, Fat-tree DC topology, communication cost	NP-Hard Approximate algorithm, Distributed Heuristic approach	H/w: PC with Core i7, 3.2 GHz, 8 GB RAM and Windows 10 S/w: Cloudsim	Algorithm does not work for other QoS parameters like security and energy.
5.	A security-aware resource allocation for mobile cloud computing systems [19]	Resource availability, system QOS requirements and mobile service request traffics	Linear Programming, Semi-Markov decision process, state transition probabaility	H/w: Intel core i5, 8 GB RAM S/w: Cloudsim	Algorithm does not check resource utilization
6.	A VM allocation strategy to make secure and energy efficient cloud [20]	Mapping of VM allocation, heuristic and pheromone information	NP-Hard Optimization problem, Ant colony optimization (ACO)	H/w: Linux platform,16 GB RAM, i7 processor S/w: MATLAB	Resources can fail at any time, so the resources are not balanced and the quality of the service is not maintained accordingly
7.	A secure load balancing policy of VMs in cloud [7]	Non-application-aware	Information leakage model for LB	Cloudsim, openstack, VM selection and VM Placement	Security score improves upto 46.90–81.15%
8.	An inter-tenant attack detection and prevention framework [21]	SQL-based syntactic analysis for multi-tenant SaaS	Interceptor, Correlator, ITADetector and HTTPChecker module	H/w: Intel core i5, 8 GB RAM S/w: AWS public cloud	Resources are not balanced and does not check resource utilization
9.	Proposed reliable client detection system (RCDS)	Detect safe and unsafe state, optimal VM placement, improving resource utilization and load balancing	Statistical analysis method, secure VM allocation	H/w: Intel core, 8 GB RAM, i5 processor S/w: Cloudsim, Java	Budget awareness is not considered into task scheduling in Clouds

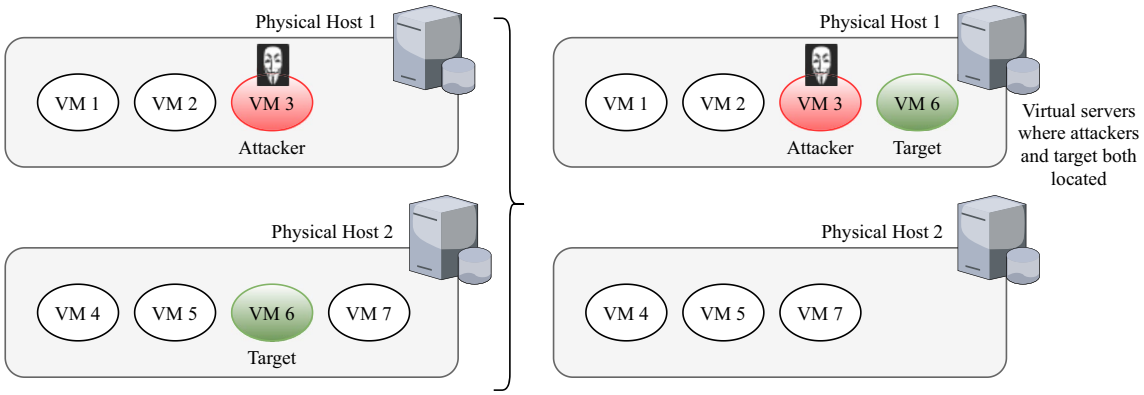

Fig. 2 Multi-tenancy attacks during load balancing

VMs and is favorable to each other where THR_F defines the VM's favorable threshold.

$$[\text{Co-tenant}_{i(t)} \times \text{Favorable}_{i(t)}](\delta t) > \prod \mathrm{THR}_F \tag{2}$$

Unknown & known host: $\mathrm{KnwHost}_{t,i,h}(\delta t)$ shows the host which is known and observed by IDS. Known hosts are known which has been Co-tenant at earlier times, and more than threshold value $\mathrm{THR}_{\mathrm{KH}}$ observed by the intrusion detection system. Similarly, if these VMs P and hosts R are co-tenant for less than $\mathrm{THR}_{\mathrm{KH}}$ in the earlier times (δt), then host R is unknown to P VM.

$$[\text{Co-tenant}_{i(t)} \times \text{Favorable}_{i(t)} \times \text{KnwHost}_{i(t)}](\delta t) > \prod \mathrm{THR}_{\mathrm{KH}} \tag{3}$$

Design and Operations

Let n Clients $\uplus = \{Cl_1, Cl_2, \cdots, Cl_n\}$, q job requests $\vartheta = \{jr_1, jr_2, \cdots, jr_q\}$ requesting to m physical hosts $\varpi_m = \{ph_{s,1}, ph_{s,2}, \cdots, ph_{s,m}, \cdots, ph_{s,\varpi}\}$ residing in server Ⓢ. Let $Ⓢ = \{s_1, s_2, \cdots, s_t, \cdots, s_{|Ⓢ|}\}$ denote as the server set with virtual machines $V = \{VM_1, VM_2, \cdots, VM_i, \cdots, VM_V\}$. A mapping $\Theta : \uplus \times \vartheta \rightarrow \varpi$ distributes each physical host from each client with specific job requests. If $\Theta_{\vartheta \times \varpi \times u} = \{\Theta_{u,jr,ph} | \Theta_{u,jr,ph} = 1\}$ then these requested jobs are allocated to the safe physical hosts ($ph_{s,m}$). Θ is the vector mapping strategy that tries to achieve the safe host for the upcoming requests for the particular deployment. It is assumed that it is done in heterogeneous nature and the main intention is to protect ($ph_{s,m}$) in this way so that co-resident attacks never harm any VMs. This detection system is formulated in dynamic environment where requirement of resources change frequently. A binary variable $\perp^i_{s,m} = 1$ indicates that this approaching VM_i is located on $ph_{s,m}$ and 0 otherwise. T is the load placement matrix and $T_{i,j}$ is the strategy of load placement between VM_i and VM_j. The following constraints should satisfies these conditions for the VM placement decision:

$$\sum_{i=1}^{V} \perp^i_{s,m} C_i \leq X_{s,m} \quad \forall s_t \in Ⓢ \quad \forall m \in \varpi \tag{4}$$

$$\sum_{i=1}^{V} \perp^i_{s,m} M_i \leq Y_{s,m} \quad \forall s_t \in Ⓢ \quad \forall m \in \varpi \tag{5}$$

where C_i and M_i represents the CPU and Memory consumption of VM_i, $X_{s,m}$ and $Y_{s,m}$ defines the total CPU and memory resource amount of $ph_{s,m}$ respectively. Equations (4) and (5) ensure that the total amount of required consumption of processors and memory resources should not exceed its total capacity. For avoiding the overflow condition of the node servers, the load constraints are as follows:

$$\frac{1}{2} \sum_{i=1}^{V} \sum_{j=1, j \neq i}^{\varpi} T_{i,j} \perp^i_{s,m} \perp^j_{t,n} \leq B_s \quad \forall s_t, s_q \in Ⓢ, \quad t \neq q \quad m \in \varpi_m, n \in \varpi_t \tag{6}$$

where B_s denotes the bandwidth of servers s_t. The coefficient (1/2) defines the VM placement pair of respective servers, i.e., the load is calculated two times. It guarantees that each task should be allocated in VMs as per respective physical hosts, which satisfies this condition in Eq. (7):

$$\sum_{s=1}^{Ⓢ} \sum_{m=1}^{\varpi} \perp^i_{s,m} = 1 \quad \forall i \in V, \quad \forall s_t \in Ⓢ, \quad m \in \varpi_m \tag{7}$$

where, $\perp^i_{s,m}$ is featured as:

$$\perp_{s,m}^{i} \in \{0,1\} \quad \forall s_t \in \text{Ⓢ}, \quad m \in \varpi_m \tag{8}$$

This $VM_{i(t)}$ defines as VM_i which is located by tenant t and $VM_{i'(t')}$ as $VM_{i'}$ located by tenant t' which is called malicious tenant. This Co-Tenant$_{t,i,t',i'}(\delta t)$ represents a boolean value which defines whether any $VM_{i(t)}$ and $VM_{i'(t')}$ are Co-Tenant at time δt. Equivalently, Co-HKnwn$_{i(t),h}(\delta t)$ also defines a boolean value which represents whether $VM_{i(t)}$ deployed in host h at time δt.

Secure VM Allocation

In this work, a secure VM allocation policy is designed to mitigate the threats of multi-tenancy attacks. The main goal is to reduce the leakage or unsafe states which occur during this proposed process. A security policy is designed for the load balancing of a virtual machine, which can effectively act on the mechanism of VM placement and VM selection. Only qualified physical hosts with adequate resources are considered when a new request is being processed. It focuses on designing an algorithm to find the optimal host with secure load balancing.

For each strategy, when the upcoming load comes for deployment, it calculates the safe or unsafe states and predicts every decision. When Cl_i send their workload job requests to the resource managers, RCDS model identifies the reliable or unreliable states. RCDS maintains a collection of all possible VM_i using Eq. (9) to calculate safe or unsafe states for all feasible migration paths, which will be further used for selection and placement of VMs.

$$\sum_{i=1,t=1}^{n,m} VM_{i(t)} = (VM_{1(t_1)}, VM_{2(t_2)}, \ldots, VM_{i(t_i)}, \ldots, VM_{n(tm)}) \tag{9}$$

Figure 3 illustrates the architecture of RCDS. The clients are sending the workload requests to the resource manager. It is basically working as the secure load balancer with findings of safe or unsafe states. The amount of resources is calculated and finds the respective qualified hosts. Then the process of VM selection will be done, and safe or unsafe states can be found and returned to the resource manager. It makes the final decision in the appraisal of both security and load balancing.

For safe states, some conceptions are initiated, and it estimates the reliability of particular tenants with the help of Co-Tenancy between those clients in historical time δt. THR_F is considered as time to identify the malicious VMs among all of the co-tenant VMs. When two VMs, P, and Q have been

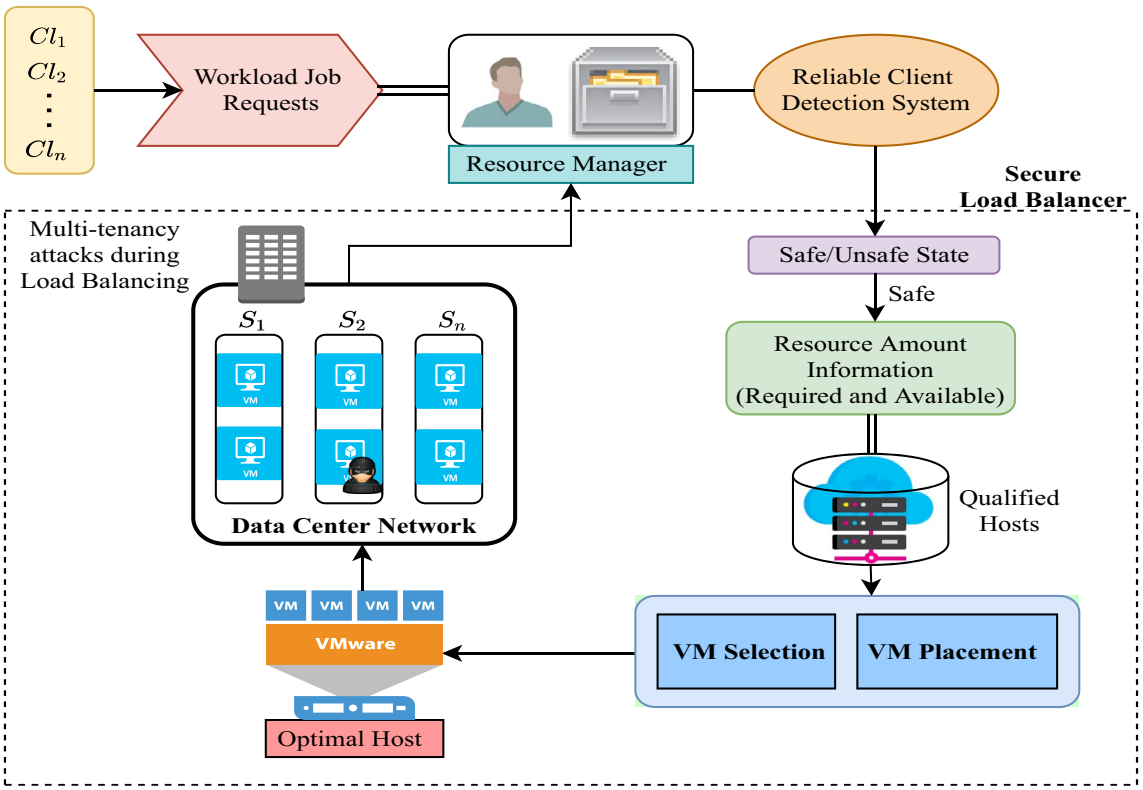

Fig. 3 Proposed architecture

co-tenant in the past δt times, and it resides parallelly for more than its threshold time THR_F, then the intrusion detection system does not alarm, and both VMs are favorable to each other, and there is no maliciousness between them as shown in Eq. (10).

$$[\text{Co-tenant}_{i(t)} \times \text{Favorable}_{i(t)}](\delta t) > \prod \text{THR}_F \tag{10}$$

These machines are ready for the deployment of tasks. For the trusty hosts, the intrusion detection system needs THR_{KH} time for finding the position of the hypervisor whether it can deploy the VM's tasks. If VM P are migrated to host R in the past δt times and P has been co-tenant in host R for less than THR_{KH}, then R is unknown host to P, and it can be the malicious one; otherwise, it is called Known Host as given in Eq. (11).

$$[\text{Co-tenant}_{i(t)} \times \text{Favorable}_{i(t)} \times \text{KnwHost}_{i(t)}](\delta t) > \prod \text{THR}_{\text{KH}} \tag{11}$$

To identify the unsafe states, all the possible ways are calculated wherever leakage can occur. $Unsafe_{i(t)\to i'(t')}(\delta t, tm)$ signifies the unsafe state from $VM_{i(t)}$ to $VM_{i'(t')}$ during time $(\delta t, tm)$. The malicious states are those where leakage occurs from $VM_{i(t)}$ to $VM_{i'(t')}$, as shown in Eq. (12).

$$\text{Unsafe}_{i(t)\to i'(t')}(\delta t, tm) = \sum (\text{Co-tenant}_{i(t),i'(t')}(\delta t) \times \text{Favorable}_{i(t),i'(t')}(\delta t)) \tag{12}$$

The unsafe states from tenant t to t', or from $VM_{i(t)}$ to host h, or tenant t to host h are shown in Eqs. (13), (14), and (15). The total unsafe states occurring in the whole cloud are calculated using Eqs. (16), and (17).

$$\text{Unsafe}_{i(t)}(\delta t, tm) = \sum_{i} \sum_{i'} \text{Unsafe}_{i(t)\to i'(t')}(\delta t, tm) \tag{13}$$

$$\text{Unsafe}_{i(t)\to i'(t')}(\delta t, tm) = \sum_{\delta t \in [\delta t - tm, \delta t]} (\text{Co-tenant}_{i(t),h}(\delta t) \times \text{KnwHost}_{i(t),h}(\delta t) \tag{14}$$

$$\text{Unsafe}_{t\to h}(\delta t, tm) = \sum_{i} \sum_{\delta t \in [\delta t - tm, \delta t]} \text{Unsafe}_{t(i)\to h}(\delta t, tm) \tag{15}$$

$$\text{Unsafe}_{i}(\delta t, tm) = \sum_{t' \in T, t' \neq t} \text{Unsafe}_{i\to i'}(\delta t, tm) + \sum_{h \in H} \text{Unsafe}_{t\to h}\delta t, tm \tag{16}$$

$$\text{Unsafe}_{\text{total}}(\delta t, tm) = \sum_{t \in T} \text{Unsafe}_{t}(\delta t, tm) \tag{17}$$

This problem for load balancing can be solved by optimizing the task deployment problem in every δt time from a long perspective. The pseudocode for finding the reliable client during load balancing (RCDS) is given in Algorithm 1.

Algorithm 1: RCDS during Load Balancing (Cl_i, VM_v, $ps_{s,\omega}$, $s_{||}$, C_i, M_i, $X_{s,m}$, $Y_{s,m}$)

Input: Requesting the jobs by clients in cloud data centres for secure load balancing in Multi-tenancy environment.
Output: A novel security policy to find safe and unsafe states.

PHList={}, QPHList={}, FPHList={}, KnwnHstPHList={}
for each client's Cl_i job request **do**
  $T_{i,j} \perp^{i}_{s,m} \perp^{j}_{t,n} \leq B_s \quad \forall s_t, s_q \in (s)$,
end for
If $\perp^{i}_{s,m}, C_i \leq X_{s,m} \; \forall \; s_t \in (s)$ **then**
  If $\perp^{i}_{s,m}, C_i \leq X_{s,m} \quad \forall \; s_t \in (s)$ **then**
    QPHList.add($ph_{s,\omega}$)
  else
    PHList.add($ph_{s,\omega}$)
  end if
end if
If two VMs are co-tenant in δt times **then**
  $[Co-Tenant_{i(t)} \times Favorable_{i(t)}](\delta t) > \prod THR_F$
  FPHList.add($ph_{s,\omega}$)
end if
If two hosts are co-tenant in δt times **then**
  $[Co-Tenant_{i(t)} \times Favorable_{i(t)} \times KnwHost_{i(t)}](\delta t) > \prod THR_{KH}$
  KnwnHstPHList.add($ph_{s,\omega}$)
end if
safe states found
for each unsafe states ε PHList **do**
  $Unsafe_i(\delta t, tm) = \sum_{t' \in T, t' \neq t} Unsafe_{i\to i'}(\delta t, tm) + \sum_{h \in H} Unsafe_{t\to h}\delta t, tm$
  $Unsafe_{total}(\delta t, tm) = \sum_{t \in T} Unsafe_t(\delta t, tm)$
end for
return *ReliableClient*

Performance Evaluation

The experiments are conducted on Cloudsim [24] simulation environment by supporting dynamic creation, and performance and efficiency of the proposed model are computed.

Experimental Setup

The experiments are conducted on a machine equipped with Intel CoreTM I5 processor clock speed and based on Intel Pentium 1.6 GHz CPU with high-performance evaluation. The simulation environment, i.e., Cloudsim, is considered to evaluate the different parameters with distinct configurations using 100 physical machines. These physical machines with

different configurations can fulfill the upcoming requirements. There are various cloudlets in the physical machine which increments/decrements for the deployment. This system is considered two allocation policies: space-shared and time-shared policy. The space-shared allocation policy is defined as the virtual machines are partitioned into a set of clusters. Every cluster is allocated as a single job and shares the memory space. The time-shared is defined as dividing the computing power by many users, and each job runs for some quantum of time. The comparison has been conducted on both the policies, i.e., time-shared and space shared with respect to cloudlet's completion time. The number of 100 physical machines is considered with different amounts of available CPU and Memory computing resources. There are 50 cloudlets running continuously on these physical machines. The parameters used in the simulation about the machines and tasks of the proposed RCDS are summarized in Table 3. Some parameters are defined in the range because of the different configurations used to represent every physical machine.

Results

MakeSpan

MakeSpan is defined as the completion time for measuring the upcoming jobs from beginning to end. It is mainly used for the length of time when workload requests are assigned to physical hosts. As evident from Table 4, the processing time of time-shared policy takes less time to execute than space-shared in case of variation in the no. of cloudlets in the safe state.

Number of Failures

This evaluation identifies the number of failure tasks in this simulation experiment during the scheduling and deployment of upcoming jobs. It happens when chosen machine couldn't fulfill their demands of the requested tasks.

We have compared the number of failures between time-shared and space-shared allocation policies for the requested number of tasks as shown in Fig. 4 in the simulated cloud network. It varies between 92 and 385 in time-shared allocation policy at the different number of requested tasks from 1000 to 5000, respectively.

Table 3 Parameters used in simulation

Parameters	Value
Host memory (MB)	204800
Host storage (MB)	10000000
Host bandwidth	100000
System architecture	× 86
Operating system	Linux
MIPS (million instructions per second)	250–350
VM image size (MB)	1000–5000
VM memory (RAM)	2048 MB
VM bandwidth	1000–2000
VMM name	Xen

Table 4 Requested tasks completion under normal state in time-shared allocation policy

Requested Tasks	Makespan (ms)	
	Time-shared	Space-shared
100	2600.36	3200.76
200	2870.23	3050.63
300	3209.84	3505.55
400	3750.26	3600.45
500	4320.24	4780.84
600	4900.21	5030.79
700	5032.40	5379.81
800	5400.80	5590.45
900	6026.19	6200.67
1000	6390.34	6470.56

Throughput Performance

Figure 5 shows the throughput measured among the various number of requested jobs by taking cloudlets as 1000, 2000, 3000, and 4000. The measure of load balancing is mainly used to analyze the actual service performance and the number of completed services per unit time. Based on

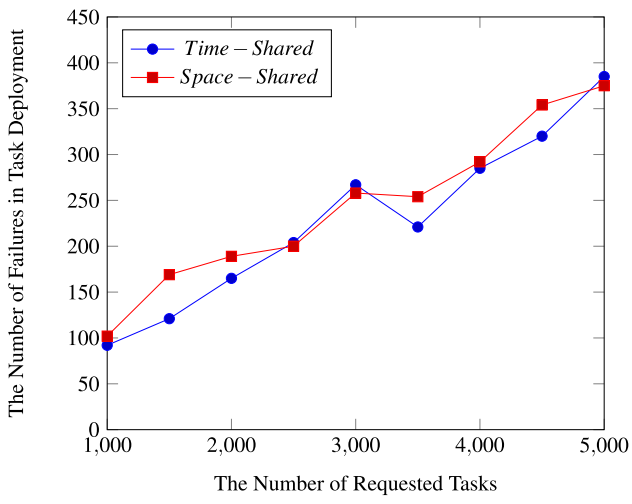

Fig. 4 Comparison of time-shared and space-shared resource allocation in terms of failures

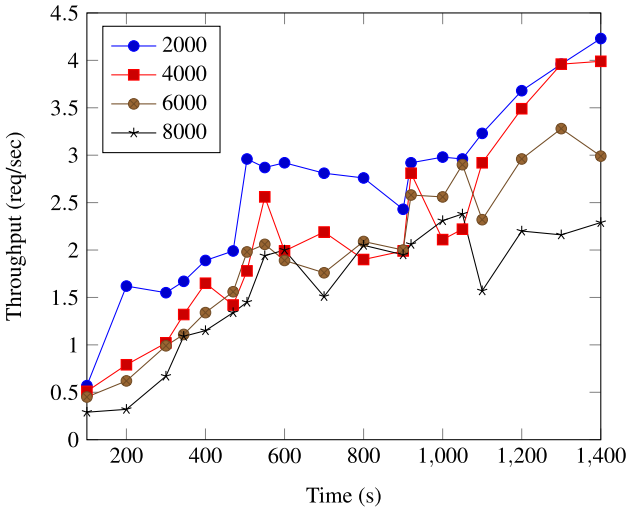

Fig. 5 Throughput in different number of requested tasks

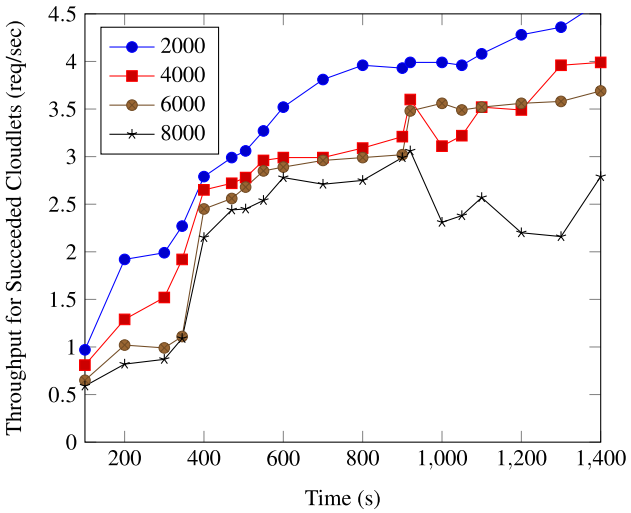

Fig. 6 Throughput in different number of succeeded requested tasks

these parameters in the cloud system, the throughput rate is calculated to evaluate the external service performance over time.

Throughput for Succeeded Cloudlets

Figure 6 shows the actual throughput, which includes an only performance that improves the responses for upcoming requests. It does not include the failure number of nodes. This picture indicates that it improves the throughput and gives much better execution than the above result. It also improves the resource utilization of the cloud data center effectively.

Number of Requested Tasks under Normal State and Safe State

Figure 7 shows the accurate cloudlet or tasks completion time for the requested tasks in time-shared scheduler policy. The number of requested tasks are varied from 100 to 1000 cloudlets. The consumption of time increases for the execution of cloudlets in time-shared scheduler allocation policy. If no. of tasks is $n = 100$, 80% of cloudlets complete their processing in time = 160.2 s while at $n = 800$, only 6% of cloudlets complete their processing in $t = 160$ s and remaining 94% cloudlets complete their processing in 320.6 or 480.8 s. Similarly, at $n = 100$, none of the cloudlets completed its processing in 160 s while a maximum of 29% of cloudlets and 71% of cloudlets completed their processing in 320.6 s and 480.8 s, respectively, in a normal state. Figure 8 shows that some of the cloudlets complete their execution more than 640.8 s, but it is quite safe than the previous graph. It shows whether it takes more time or ensures the clients share the resources without any insecurity. Tables 5 and 6 illustrate the cloudlet completion time under normal and safe states, respectively. The time-shared allocation policy is applied with respect to variation in no. of requested tasks for both the tables. There is some amount of time defined with an interval. For Table 7, if the requested tasks are 100, then the maximum of 80.00% of cloudlets completes their execution in the time interval (1–200)s and the rest of the 20% completes in the interval (201–400)s. As n is increased from 100 to 1000, an equal percentage of cloudlets complete their tasks in time interval (1–200)s, (201–400)s and (401–600)s, the percentage being 2.22%, 23.11%, and 74.66% for $n = 900$ respectively. Table 8 depicts the cloudlet completion time of cloudlets with respect to variation in no. of cloudlets (n) under attack conditions when space-shared allocation policy is applied under a normal state.

Conclusion

This paper highlights the proposed hybrid approach of secure load balancing with proper utilization of resources in the heterogeneous environment upto 200 physical machines. This proposed system helps to anticipate the safe or unsafe states for the optimal VM placements. It predicts the resource requirement and then allocates a disburse amount of VMs for the upcoming tasks and reduces the information leakage during load balancing as well as improves the security. It deals in a dynamic environment that diminishes VM downtime, improves failure rate, enables flexible resource allocation and load balancing improvement. It achieves traffic scalability improvement

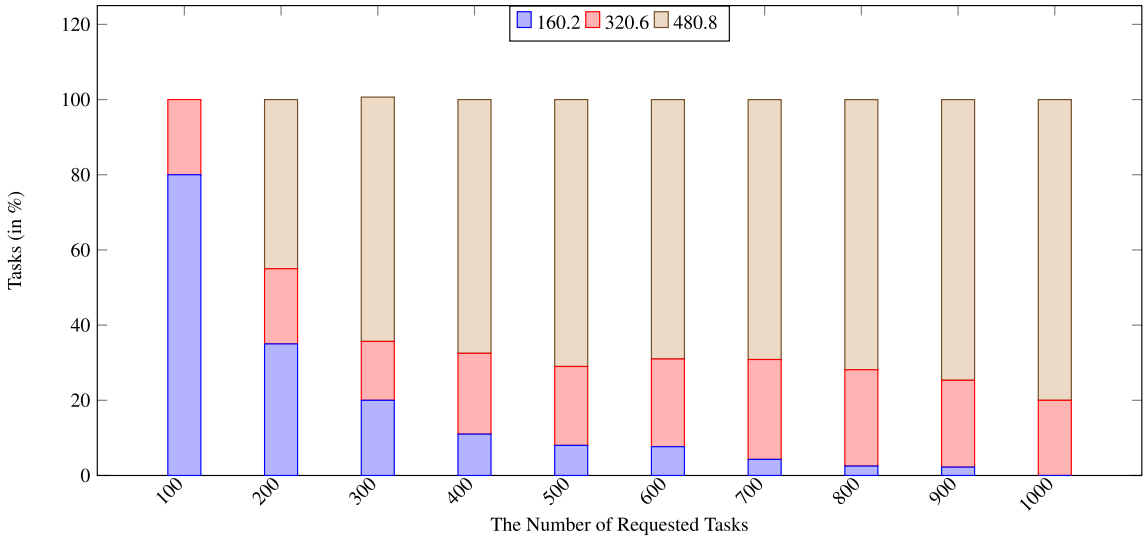

Fig. 7 Number of requested tasks under normal state

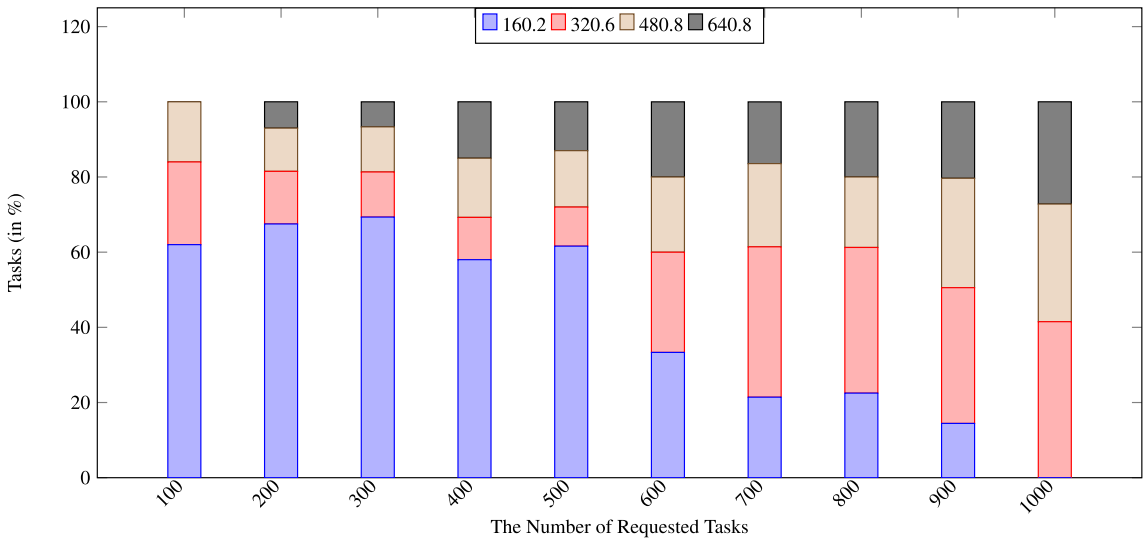

Fig. 8 Number of requested tasks under safe state

Table 5 Requested tasks completion under normal state in time-shared allocation policy

Normal state						
Requested tasks	1–200	201–400	401–600	(1–200)%	(201–400)%	(401–600)%
100	80	20	0	80.00	20.00	0.00
200	70	40	90	35.00	20.00	45.00
300	65	50	195	20.33	15.66	65.00
400	44	86	270	11.00	21.50	67.50
500	40	105	355	8.00	21.00	71.00
600	46	140	414	7.66	23.33	69.00
700	30	186	484	4.28	26.57	69.14
800	20	205	575	2.50	25.62	71.87
900	20	208	672	2.22	23.11	74.66
1000	0	200	800	0.00	20.00	80.00

Table 6 Requested tasks completion under safe state in time-shared allocation policy

Safe state								
Requested tasks	1–200	201–400	401–600	601–800	(1–200)%	(201–400)%	(401–600)%	(601–800)%
100	60	25	15	0	60.00	25.00	15.00	0.00
200	80	64	56	0	40.00	32.00	28.00	0.00
300	80	80	95	45	26.66	26.66	31.66	15.00
400	85	85	205	25	21.25	21.25	51.25	6.25
500	90	105	255	50	18.00	21.00	51.00	1.00
600	60	146	314	80	1.00	24.33	52.33	13.33
700	40	182	360	118	0.57	26.00	51.42	16.85
800	0	200	526	174	0.00	25.00	65.75	21.75
900	0	245	475	180	0.00	27.22	52.77	2.00
1000	0	275	505	220	0.00	27.50	50.50	22.00

Table 7 Requested tasks completion under safe state in space-shared allocation policy

Normal state								
Requested tasks	1–200	201–400	401–600	601–800	(1–200)%	(201–400)%	(401–600)%	(601–800)%
100	62	22	16	0	62.00	22.00	16.00	0.00
200	135	28	23	14	67.50	14.00	11.50	7.00
300	208	36	36	20	69.33	12.00	12.00	6.66
400	312	45	23	20	78.00	11.25	5.75	5.00
500	408	52	25	15	81.60	10.40	5.00	3.00
600	200	160	120	120	33.33	26.66	20.00	20.00
700	220	280	155	45	31.42	40.00	22.14	6.42
800	180	310	150	160	22.50	38.75	18.75	20.00
900	130	325	262	183	14.44	36.11	29.11	20.33
1000	0	415	313	272	0.00	41.50	31.30	27.20

Table 8 Requested tasks completion under safe state in space-shared allocation policy

Safe state										
Requested tasks	1–200	201–400	401–600	601–800	801–1000	(1–200)%	(201–400)%	(401–600)%	(601–800)%	(801–1000)%
100	50	25	25	0	0	50.00	25.00	25.00	0.00	0.00
200	85	55	45	15	0	42.50	27.50	22.50	7.50	0.00
300	123	86	58	33	0	41.00	28.66	19.33	11.00	0.00
400	200	123	60	17	0	50.00	30.75	15.00	4.25	0.00
500	100	305	80	15	0	20.00	61.00	16.00	3.00	0.00
600	80	412	102	6	0	7.50	68.66	17.00	1.00	0.00
700	60	555	85	0	0	8.57	79.28	12.14	0.00	0.00
800	52	265	285	89	109	6.50	33.12	35.62	11.12	13.62
900	32	8	585	190	125	3.55	0.08	65.00	21.11	13.88
1000	0	0	560	260	180	0.00	0.00	56.00	26.00	18.00

and compared the space shared and time shared approach for both cloudlets and VM level allocation policy. The simulation experiments have shown that the framework can successfully handle both, i.e., defend against the multi-tenancy attacks and balance the workloads, thus outperforming the existing solutions.

Author Contributions All the authors have discussed and constructed the ideas, designed the security preserving model and wrote the paper together.

Data availability Data derived from public domain resources.

Declarations

Conflict of Interest The authors have no conflict of interest regarding the publication.

Ethical Approval This article does not contain any studies with human participants or animals performed by any of the authors.

Informed Consent Informed consent was obtained from all individual participants included in the study.

References

1. Alex ME, Kishore R. Forensics framework for cloud computing. Comput Electr Eng. 2017;60:193–205.
2. Liu Y, Gong B, Xing C, Jian Y. A virtual machine migration strategy based on time series workload prediction using cloud model. Math Probl Eng. 2014;2014: 973069.
3. Ristenpart T, Tromer E, Shacham H, Savage S. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In: Proceedings of the 16th ACM conference on Computer and communications security, pp. 199–212, 2009.
4. Hu K-H, Jianguo W, Tzeng G-H. Risk factor assessment improvement for China's cloud computing auditing using a new hybrid MADM model. Int J Inform Technol Decis Mak. 2017;16(3):737–77.
5. Zhang Y, Juels A, Reiter MK, Ristenpart T. Cross-VM side channels and their use to extract private keys. In: Proceedings of the 2012 ACM conference on Computer and communications security, pp. 305–316, 2012.
6. Moon S-J, Sekar V, Reiter MK. Nomad: mitigating arbitrary cloud side channels via provider-assisted migration. In: Proceedings of the 22nd ACM sigsac conference on computer and communications security
7. Sun Q, Shen Qi, Li C, Wu Z. Selance: secure load balancing of virtual machines in cloud. In: 2016 IEEE Trustcom/BigDataSE/ISPA, pp. 662–669. IEEE, 2016.
8. Han Y, Chan J, Alpcan T, Leckie C. Using virtual machine allocation policies to defend against co-resident attacks in cloud computing. IEEE Trans Depend Secur Comput. 2015;14(1):95–108.
9. Duan J, Yang Yuanyuan. A load balancing and multi-tenancy oriented data center virtualization framework. IEEE Trans Parallel Distrib Syst. 2017;28(8):2131–44.
10. Wang Z, Hayat MM, Ghani N, Shaban KB. Optimizing cloud-service performance: efficient resource provisioning via optimal workload allocation. IEEE Trans Parallel Distrib Syst. 2016;28(6):1689–702.
11. Deng R, Lu R, Lai C, Luan TH, Liang H. Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption. IEEE Internet Things J. 2016;3(6):1171–81.
12. Fernández-Cerero D, Jakóbik A, Grzonka D, Kołodziej J, Fernández-Montes A. Security supportive energy-aware scheduling and energy policies for cloud environments. J Parallel Distrib Comput. 2018;119:191–202.
13. Singh AK, Kumar J. Secure and energy aware load balancing framework for cloud data centre networks. Electron Lett. 2019;55(9):540–1.
14. Saxena D, Singh AK. OSC-MC: online secure communication model for cloud environment. IEEE Commun Lett. 2021;25:2844–8.
15. Papagianni C, Leivadeas A, Papavassiliou S, Maglaris V, Cervello-Pastor C, Monje A. On the optimal allocation of virtual resources in cloud computing networks. IEEE Trans Comput. 2013;62(6):1060–71.
16. Zeng L, Veeravalli B, Li X. Saba: a security-aware and budget-aware workflow scheduling strategy in clouds. J Parallel Distrib Comput. 2015;75:141–51.
17. Usmin S, Irudayaraja MA, Muthaiah U. Dynamic placement of virtualized resources for data centers in cloud. In: International Conference on Information Communication and Embedded Systems (ICICES2014), pp. 1–7, 2014.
18. Cui L, Tso FP, Pezaros DP, Jia W. Plan: a policy-aware vm management scheme for cloud data centres. In: 2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC), pp. 142–151. IEEE, 2015.
19. Liu Y, Lee MJ. Security-aware resource allocation for mobile cloud computing systems. In: 2015 24th International Conference on Computer Communication and Networks (ICCCN), pages 1–8, 2015.
20. Li B, Liu P, Lin L. A cluster-based intrusion detection framework for monitoring the traffic of cloud environments. In: 2016 IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud), pages 42–45. IEEE, 2016.
21. Yassin M, Talhi C, Boucheneb H. Itadp: an inter-tenant attack detection and prevention framework for multi-tenant saas. J Inform Secur Appl. 2019;49: 102395.
22. Saxena D, Gupta I, Kumar J, Singh AK, Wen X. A secure and multiobjective virtual machine placement framework for cloud data center. IEEE Syst J. 2021;16:3163–74.
23. AlJahdali H, Albatli A, Garraghan P, Townend P, Lau L, Xu J. Multi-tenancy in cloud computing. In: 2014 IEEE 8th international symposium on service oriented system engineering, pp. 344–351. IEEE, 2014.
24. Calheiros RN, Ranjan R, Beloglazov A, De Rose CAF, Buyya R. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw Pract Exp. 2011;41(1):23–50.
25. Xiao Z, Song Weijia, Chen Q. Dynamic resource allocation using virtual machines for cloud computing environment. IEEE Trans Parallel Distrib Syst. 2012;24(6):1107–17.
26. Zhao J, Yang K, Wei X, Ding Y, Liang H, Gaochao X. A heuristic clustering-based task deployment approach for load balancing using Bayes theorem in cloud environment. IEEE Trans Parallel Distrib Syst. 2015;27(2):305–16.
27. Chhabra S, Singh AK. Dynamic hierarchical load balancing model for cloud data centre networks. Electron Lett. 2019;55(2):94–6.
28. Ma T, Jiangxing W, Yuxiang H, Huang Wanwei. Optimal VM placement for traffic scalability using Markov chain in cloud data centre networks. Electron Lett. 2017;53(9):602–4.
29. Lin C, Li G, Shan Z, Shi Y. Thinking and modeling for big data from the perspective of the I Ching. Int J Inform Technol Decis Mak. 2017;16(06):1451–63.

30. Chhabra S, Singh AK. Dynamic data leakage detection model based approach for mapreduce computational security in cloud. In: 2016 Fifth International Conference on Eco-friendly Computing and Communication Systems (ICECCS), pp. 13–19. IEEE, 2016.
31. Mi W, Qiu X, Zhang C. The analysis of security threats in structured p2p load balancing schemes. In: 2011 International Conference on Cloud and Service Computing, pp. 296–301. IEEE, 2011.
32. Liu Y, Ruan X, Cai S, Li R, He H. An optimized vm allocation strategy to make a secure and energy-efficient cloud against co-residence attack. In: 2018 International Conference on Computing, Networking and Communications (ICNC), pp. 349–353. IEEE, 2018.