

We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,100

Open access books available

149,000

International authors and editors

185M

Downloads

Our authors are among the

154

Countries delivered to

TOP 1%

most cited scientists

12.2%

Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com



Chapter

Perspective Chapter: Deep Reinforcement Learning for Co-Resident Attack Mitigation in The Cloud

Suxia Cui and Soamar Homsy

Abstract

Cloud computing brings convenience and cost efficiency to users, but multiplexing virtual machines (VMs) on a single physical machine (PM) results in various cybersecurity risks. For example, a co-resident attack could occur when malicious VMs use shared resources on the hosting PM to control or gain unauthorized access to other benign VMs. Most task schedulers do not contribute to both resource management and risk control. This article studies how to minimize the co-resident risks while optimizing the VM completion time through designing efficient VM allocation policies. A zero-trust threat model is defined with a set of co-resident risk mitigation parameters to support this argument and assume that all VMs are malicious. In order to reduce the chances of co-residency, deep reinforcement learning (DRL) is adopted to decide the VM allocation strategy. An effective cost function is developed to guide the reinforcement learning (RL) policy training. Compared with other traditional scheduling paradigms, the proposed system achieves plausible mitigation of co-resident attacks with a relatively small VM slowdown ratio.

Keywords: cloud computing, risk mitigation, resource management, co-resident attack, reinforcement learning

1. Introduction

Cloud Computing, which has its origins in expanding the Internet, aims to provide remote and scalable computing and storage resources to its customers. Users from small businesses in a locally resource-limited environment can manipulate and store large datasets for real-time processing with cloud services. The cloud platform has gradually reshaped daily lives because it has been recognized as a convenient way to transmit and store data in the big data era. Organizations can choose from the public, private, or hybrid cloud that combines the public and private deployment model features. The term “XaaS” is coined for the service-oriented architecture, emphasizing that anything can be treated as a service under the cloud computing environment. Examples of cloud delivery services include infrastructure as a service (IaaS),

platform as a service (PaaS), and software as a service (SaaS) [1]. Recently, function as a service (FaaS) further expanded the backend as a service (BaaS) offering. Under each delivery model, a cloud service provider (CSP) is responsible for allocating enough resources to maintain quality of service (QoS) to the users and protect their data from security risks.

Virtualization has been adopted by most cloud computing platforms to profit from the “pay as you go (PAYG)” model [2]. Virtualization is an idea generated from IBM’s mainframe platform in the early 1960s. After entering the twenty-first century, it was successfully utilized in cloud computing that can bring down the cost of maintaining a large-scale system. It converts a physical server into numerous VMs, rented out to several occupants [3–7]. This VM-PM relationship is illustrated in **Figure 1**.

The apparent relationship between PMs running and power consumption places a high demand on a strategy for energy minimization in this configuration [8]. Security and data privacy are other concerns for cloud computing platforms. Attackers will seek to exploit any vulnerability to achieve various malicious goals on the victim’s network, software, and databases. The co-resident attack is one of the prevalent cybersecurity risks resulting from virtualization. Ideally, two neighboring VMs are isolated from each other when running their tasks. However, in reality, each co-located VM will depend on the same PM where hardware, like CPUs or memory elements, is shared by all the VMs. Therefore, a VM’s private information may be accessed by its neighboring VM by launching side-channel attacks [9–12], as shown in **Figure 2**. Here, a hypervisor or virtual machine monitor (VMM) creates and runs VMs on a hosting PM. The arrows illustrate the route of side-channel attacks. A side-channel attack is a significant security challenge that prevents many organizations from adopting cloud computing technology. Although recently deep learning algorithms have proven to be effective in cloud resource management [13–16], few paid attention to side-channel attack avoidance at the same time.

To fill in this gap, we developed a novel deep reinforcement-learning (DRL) based dynamic VM allocation approach to optimize the trade-offs between the VM completion and the co-resident risks mitigation. The main contributions of this paper are as follows:

- Threat model design: A time-sensitive zero trust threat model is developed for co-resident vulnerability analysis. The model enables the tracking of VM co-existent pairs on the same PM.

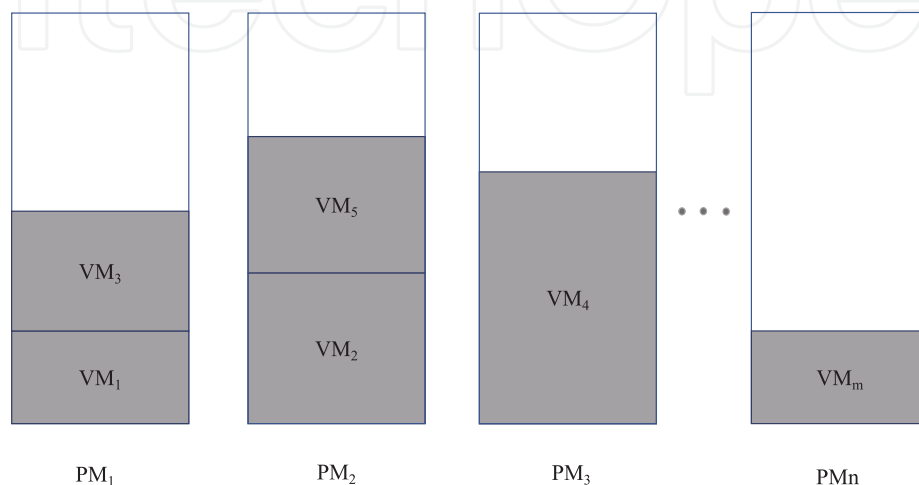


Figure 1.
VMs and PMs in a data center via virtualization.

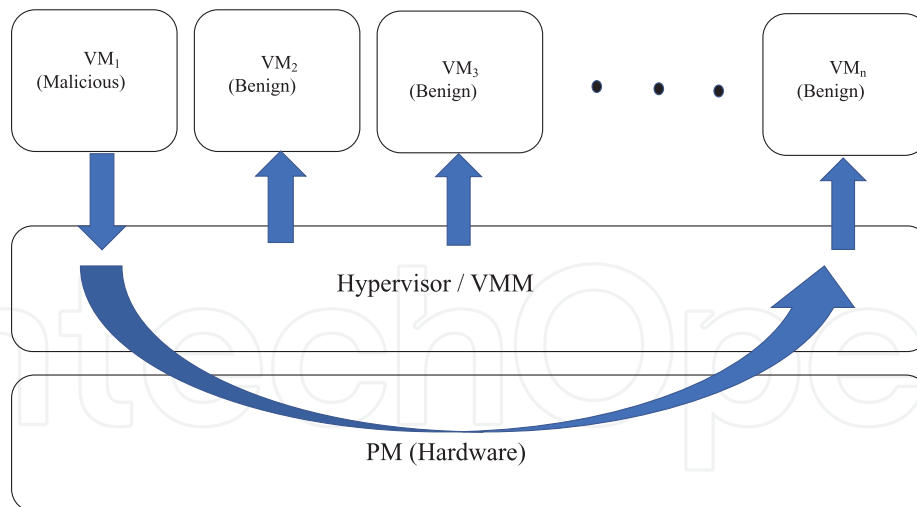


Figure 2.
Side-channel attacks in cloud computing.

- DRL adoption to co-resident risk mitigation: This article is the first to investigate the different states, actions, and rewards of DRL to fit it into cloud computing side-channel attacks. The rewards guide the VMM decision-making.
- Simulation platform implementation: The proposed system accepts tasks dynamically at runtime and analyzes cloud co-resident risks at each timestep for optimized scheduling algorithms.

2. Related works: Secure cloud resource management

Resource management alone without security consideration in a cloud computing environment is already very challenging. Multi-tenant environments allow attackers to begin a co-resident infiltration and steal the victim's information by side channels. The risks that attackers pose to VMs on the same hypervisor are growing security concerns and are being addressed differently. This section will discuss established current resource management approaches with and without security awareness. Three categories of methodologies are commonly used: Heuristic, game theory, and machine learning (ML).

The exploration of heuristics and meta-heuristics to solve nondeterministic polynomial time (NP) problems are growing amid the difficulties to solve them using traditional methods [17]. Gawali and Shinde [18] combined a modified analytic hierarchy process (MAHP), bandwidth aware divisible scheduling (BATS), and the longest expected processing time preemption (LEPT) to achieve improved performance. Qin *et al.* [20] took the idea of "ant colony optimization" from [19] and proposed a probabilistic algorithm that can simultaneously maximize the revenue of communications and minimize the power consumption of PMs. Similarly, Tawfeek *et al.* also adopted ant species' nature and presented a random optimization search approach for allocating the incoming jobs to the virtual machines [21]. The proposed method outperformed the popular first come first serve method. Patel introduced a hybrid algorithm that used a modified honeybee behavior-inspired algorithm for priority-based tasks and an enhanced weighted round-robin algorithm for non-priority-based tasks [22] to balance the workload over the cloud dynamically. When using heuristic

methods with the consideration of co-resident attacks, various policies were compared. Jia *et al.* [23] proposed a VM allocation method to optimize load balancing and reduce energy consumption and security risks by managing CPU utilization of the hosting PMs. Miao *et al.* offered two metrics to outline co-residency and conflict of the cloud [24]. Both placement and migration algorithms mediate differences between tenants to alleviate co-resident attacks in the cloud proactively. Han *et al.* formulated a set of security metrics and a quantitative model to assign new VMs to the server with the most VMs [25]. The research uncovered that the server's configuration, over-subscription, and background traffic had a substantial impact on the ability to stop attackers from co-locating with the targets.

Yang *et al.* [26] explored a simplified algorithm for energy management in cloud computing. The paper centered around establishing a mathematical model to calculate computing nodes' stability, configuring a game-theoretic cooperative model for the task of scheduling cloud computing, and examining the problem as a multi-stage sequential game. Patra *et al.* presented the task as a player and the VM as a strategy in [27]. A non-cooperative game scheduling and a task balance scheduling algorithm are compared to collect the node's average task processing speed. Therefore, it was determined that the game-theoretic algorithm proposed could improve energy management in cloud computing. In [28], the cooperative behavior of multiple cloud servers was studied. An evolutionary mechanism was presented in the hierarchical cooperative game model for VMs deployment strategy to improve the efficiency in the public cloud environment. Jia *et al.* modeled several basic VM allocation policies using game theory to achieve a quantitative analysis, while also presenting the attack effectiveness, coverage, power consumption, workload balance, and cost under the VM allocation policies and solving the mathematical solution in CloudSim [23]. Their results found that to reduce the efficiency rates for the attacker, the cloud provider should apply a probabilistic VM allocation policy. Narwal *et al.* proposed a payoff matrix and a decision tree for any number of users [29, 30]. When a unique user was selected, the choices of investing in security were assessed until equilibrium was reached. Security games are a way of blocking the attacker's ability to locate the VMs they are searching for. Han *et al.* proposed a policy pool with multiple VM allocation policies from which to select the policy that will be used with a certain probability [31].

Difficulties regarding energy efficiency in cloud computing can also be addressed using machine learning-based techniques [32]. Witanto *et al.* employed a neural network-based adaptive selector procedure to arrange the VMs on the physical servers in data centers [33]. Pahlevan *et al.* presented a hyper-heuristic algorithm to exploit both heuristic and ML-based VM allocation methods by selecting the best one during run-time [32]. Zhang *et al.* [34] suggested an auction-based resource allocation scheme to represent a machine learning classification or regression problem. They outlined machine learning classification and posed two resource allocation prediction algorithms rooted in linear and logistic regression. Liu *et al.* presented a reinforcement learning-based approach to allow complex scenarios to efficiently manage resources [35]. In order to do so, they used neural networks to grasp the goal of the research model, RL to enhance the model, and E-greedy methodology to expand the RL process. Their approach lowered job delay for hybrid scenarios. ML-based methods have been proposed to fight against co-resident attacks focusing on different factors, such as minimizing the time of a malicious VM co-location. Joseph *et al.* [36] used traditional ML algorithms, such as support vector machine (SVM), naïve bayes, and random forests to detect malware, following a self-healing methodology to power off the attacked VMs and restore them to healthy conditions. In reality, there is a concern

with the amount of time it takes to implement a solution to mitigate VMs in the event of co-resident attacks. To the best of our knowledge, no current ML-based approach succeeded in mitigating co-resident attacks based on VM mitigation, while minimizing the VM downtime.

3. Threat model

There are many approaches to fight against co-resident attacks, including hardware modification, intrusion detection, secure VM allocation, and migration. Threat model building is crucial to guide proper defense. This section goes through the study of existing models and presents our proposed threat model with detailed variable selections.

3.1 Modeling co-resident attacks

Many optimization models were proposed to fight against co-resident attacks. Abazari *et al.* suggested a multi-objective optimization method to calculate alternative responses with the least amount of threat through graphics and proper attack countermeasures [37]. Liu *et al.* considered the three main factors which lead to the likelihood of malicious VMs co-locating with normal users [38]. Berrima *et al.* used a VM placement strategy to reduce the co-location attacks with complete resource optimization. Their approach presents a trade-off between security and VM startup delay [39]. Hasan *et al.* proposed a co-resident attacks mitigation and prevention (CAMP) model to separate malicious and benign VMs by comparing existing models over data security, data survivability, and user storage overhead [5]. Other works focused on a probabilistic co-residence coverage optimization model, while combining a data partition technique that involves arranging servers randomly [40, 41].

3.2 Proposed threat model with detailed design components

Our proposed approach takes the time-sensitive risk level from co-resident attacks into account and searches for the solution to the dynamic VM allocation problem through DRL. Research shows that the co-resident attack will have a total cycle of t_3 , consisting of three stages: probe, construct, and launch. Probe and construct generate a configuration interval. This is illustrated in **Figure 3**. To avoid the attacks, the defender must take action before the launching starts. In other words, before the configuration interval t_2 is reached [42].

Our co-resident risk model is developed in a similar scenario to [43]. The choice of variables is listed in **Table 1**.

The co-resident risk indicator can be obtained through the following equations:

$$rcr(v_i, v_j) = t_s(v_i) \times CoRes(v_i, v_j) \times t_s(v_j) \quad (1)$$

$$CoResFactor = \begin{cases} \alpha_0 & \text{for } CoRes(v_i, v_j) < t_1 \\ \alpha_1 & \text{for } CoRes(v_i, v_j) \in [t_1, t_2) \\ \alpha_2 & \text{for } CoRes(v_i, v_j) \geq t_2 \end{cases} \quad (2)$$

Variables	Descriptions
N	No. of PMs
n	No. of resource slots in one PM
M	No. of VMs
X	The mapping between VMs and PMs
t_1	End of probing [42]
t_2	End of constructing/configuration interval [42]
$t_s(v_i)$	The threat score of v_i [0,1]
$CoRes(v_i, v_j)$	The co-resident duration matrix between v_i and v_j
$rcr(v_i, v_j)$	Co-resident risk indicator matrix
$CoResFactor$	VM's co-resident rewards factor

Table 1.
Variable definition.

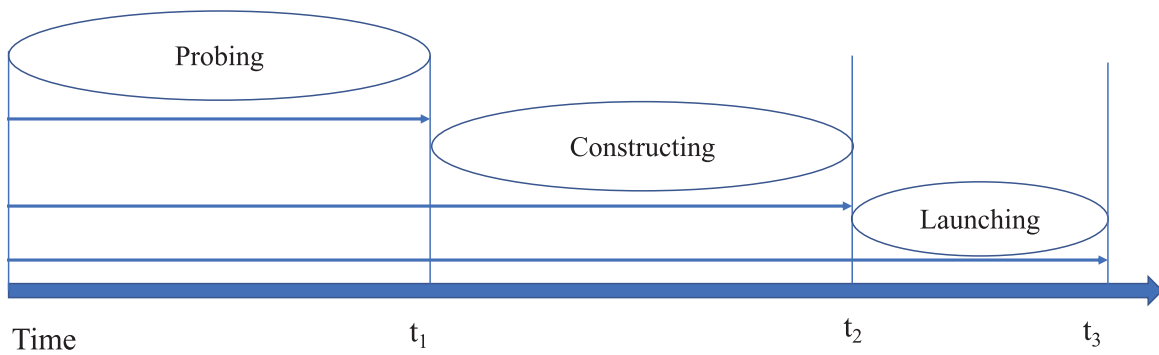


Figure 3.
The timeline of attacks [42].

The threat score, $t_s(v_i)$, reflects the potential risk to VM_i . It is a floating number between 0 and 1, with 0 representing no risk and 1 representing the highest risk. As illustrated in Eq. (1), the risk is also proportional to the co-resident duration time recorded in a matrix, $CoRes(v_i, v_j)$.

The VM's co-resident rewards factor, $CoResFactor$, is the crucial parameter in guiding RL training. It can be determined by where the co-resident attack cycle status of the VM resides. For example, the time of a co-existing VM pair on the same PM for a period of less than t_1 is considered to be safe. If there is a malicious VM, it means that it has not passed the probing stage yet. So, the risk of getting a co-resident attack is low. In this case, $\alpha_0 = 0$ is chosen. If the $CoRes(v_i, v_j)$ is between t_1 and t_2 , the system needs to be aware that if a malicious VM exists in the pair; it reaches the constructing stage and moves closer to launching the attack. So, α_1 needs to be non-zero. While the VM pair co-exists on the same PM for more than t_2 time period, an attack could be launched. This is the situation to be avoided, so that α_2 is assigned to a more aggressive number.

3.3 Assumptions

Two assumptions guide the proposed model:

1. Co-resident risk is calculated among all the active VM pairs on the same PM, and each VM is randomly created with 1 to 15-time steps of the length.
2. All the jobs are batch jobs, and all the VMs might be malicious. The goal is to mitigate co-resident risk by avoiding the VM pairs' co-resident period from reaching t_2 .

The system will be simulated under the above assumptions. The overall co-resident risk level, the number of co-resident attacks, and the VM slowdown ratio is the proposed system's evaluation metrics.

4. DRL-based VM scheduling system design and simulation

4.1 Mathematical background of RL and schematics

RL problems can be modeled as a Markov decision process (MDP) to find a policy by maximizing the accumulated rewards. An MDP has four tuples (S, A, P_a, R_a) , where S is a set of states called state space, A is a set of actions called action space, P_a is the probability of state transition from s to s' under action a , and R_a is the immediate reward right after action a . There are two major methods to solve the reinforcement learning iteration problem. One is called *value – function*, and the other is *policy – gradient*. Q-Learning is an example of *value – function*, which has a function: $Q : S \times A \rightarrow R$. Before learning begins, Q is initialized to 0 or a base value. The core of the algorithm is a Bellman equation, which updates the Q value with new information:

$$Q^{new}(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a) \right] \quad (3)$$

Here, α and γ are the learning rate and discount factor, respectively. r_t is the reward at the time step t . We adopt DeepRM [44] framework, which follows *policy – gradient* with a deep neural network added into this system to solve large-scale RL tasks. This portion of the deep RL can be illustrated in **Figure 4**.

The nature of the *policy – gradient* is to maximize the expected cumulative discount reward $E_{\pi\theta} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right]$, which can be expressed as:

$$\nabla_{\theta} E_{\pi\theta} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right] = E_{\pi\theta} \left[\nabla_{\theta} \log \pi_{\theta}(s, a) Q^{\pi\theta}(s, a) \right] \quad (4)$$

Here, $\gamma \in (0, 1]$ is a discount factor for future rewards. r_t is the reward at the time step t . The VMM picks actions based on a *policy* $\pi : \pi(s, a) \rightarrow [0, 1]$, which is defined as the probability of action a taken in the state s . A manageable number of adjustable parameters, θ , are called the policy parameter. So, the policy can be represented as $\pi_{\theta}(s, a)$, and θ will be updated via gradient descent:

$$\theta \leftarrow \theta + \beta \sum_t \nabla_{\theta} \log \pi_{\theta}(s_t, a_t) v_t \quad (5)$$

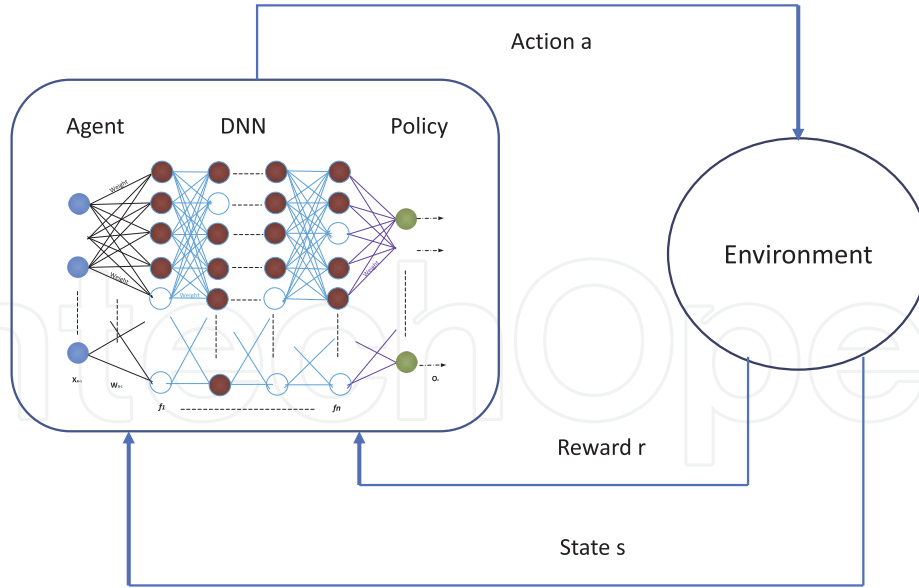


Figure 4. Reinforcement learning with policy represented via DNN [44].

where β is the step size. The corresponding expected cumulative discounted reward $Q^{\pi^{\theta}}(s, a)$ can be estimated by the empirically computed cumulative discounted reward v_t .

4.2 RL components design

Reinforcement learning is a unique type of machine learning paradigm, which has been successfully applied to task scheduling [45–49]. It contains several detailed components that need clarification. Here, we first define our state space, action space, and rewards before introducing the simulation system.

4.2.1 State space

RL is a model-free machine learning method; an agent learns from the trial-and-error process to interact with the environment. The state of the environment is defined as a vector of several components as shown in **Table 2**. They build the data structure of a VM which can be classified as:

1. Computing resources factors;
2. Security awareness factors (already introduced in **Table 1**).

The current allocation of the cluster resources can be retrieved by the mapping between VMs and resource slots available on the PM, which can be expressed as a matrix X .

4.2.2 Action space

It is assumed that VMs will be assigned to the PM if requested resources are available at each time step. The action space is defined by $\{0, 1, \dots, n\}$, where 0 means

Vector component	Values	Data type
Computing Resources		
Hosting PMs	1	Integer
VM ID	1	Integer
Ideal length of finishing	1	Float
Resources requested	2	Integer
Start time	1	Float
Finish time	1	Float
$VM_{Delayed}$	1	Float
Security		
t_s	1	Float
$CoRes$	n	Integer
rcr	n	Float

Table 2.
 Data structure of a VM.

no action taken, and 1 through n means to allocate a new VM on the n 's PM slot. After each action, the next state space is obtained by updating the recent mapping of X .

4.2.3 Rewards

A reward strategy is designed to guide the VM allocation agent toward our goal: Sufficiently utilize current resources to complete jobs on time and simultaneously minimize co-resident attacks. The reward function must be carefully designed to avoid contradiction. In the proposed system with a total of J active VMs, the rewards function consists of two terms:

1. VMs' completion factor, R_{VMD} (VM delay rewards), is defined by the accumulated $VM_{Delayed}$ from currently running VMs in the system. Here, T_j is the duration of the j th VM, v_j .
2. VMs' co-resident risk level, R_{RC} (runtime co-resident rewards), is defined by the co-resident risk indicator matrix $rcr(v_i, v_j)$.

They can be calculated accordingly as:

$$R_{VMD} = \sum_{j \in J} VM_{Delayed}(v_j) = \sum_{j \in J} \frac{-1}{T_j} \quad (6)$$

$$R_{RC} = \sum_{i, j \in J} (-1) \times rcr(v_i, v_j) / 2 \quad (7)$$

The full rewards are calculated as a weighted sum of the two terms with weights ω_1 and ω_2 . The overall rewards can be obtained by:

$$TotalRewards = \omega_1 \times R_{VMD} + \omega_2 \times R_{RC} \tag{8}$$

This rewards equation implies the objective of this novel VM allocation system is focused on:

1. Efficiently allocating VMs to minimize the VM delay time as shown in Eq. (6);
2. Aware of co-resident attacks through VM assignment correlations and take action to avoid the risk as shown in Eq. (7).

4.3 Simulation system design

4.3.1 System model

Similar to the DeepRM [44] framework, the proposed simulation system is illustrated in **Figure 5**. CPU and memory (MEM) are the two resources for limited constraint consideration. When the VM is assigned to the PM, a time step starts to count the duration of the VM. If there are other VMs simultaneously assigned to the same PM, the co-resident counter is also started to accumulate the time steps and recorded in $CoRes(v_i, v_j)$. VM requests arrive according to a Bernoulli process. The backlog queue houses all the incoming VMs waiting for allocation.

4.3.2 Co-resident duration matrix

The time steps will be recorded in the co-resident duration matrix as shown below. This small-scale example limits each CPU and MEM resource to five slots. Here, five VMs $\{VM_1, \dots, VM_5\}$ are illustrated with the life cycles marked with a start and end

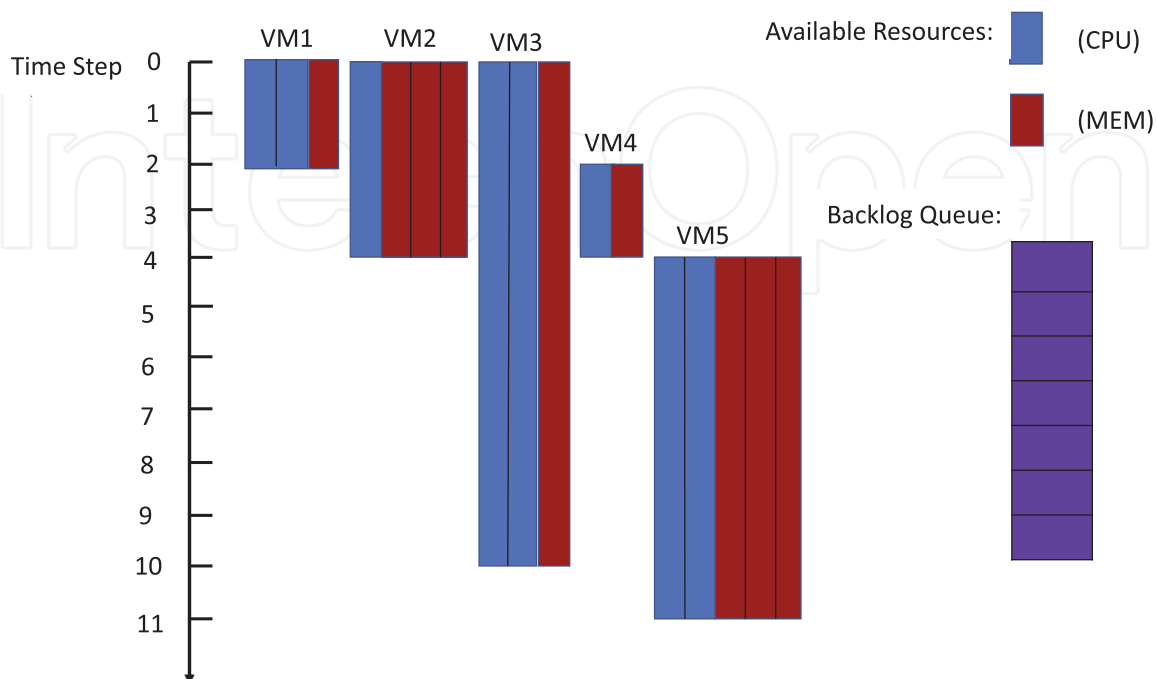


Figure 5. Resource, time steps, job slots, and backlog queue in [44].

time steps as $VM_1 : [0, 2]$, $VM_2 : [0, 4]$, $VM_3 : [0, 10]$, $VM_4 : [2, 4]$, $VM_5 : [4, 11]$. Their life cycle lengths can be represented as $\{2, 4, 10, 2, 7\}$, respectively. The overlapped time steps shown in a co-resident duration matrix are:

$$CoRes(v_i, v_j) = \begin{bmatrix} 2 & 2 & 2 & 0 & 0 \\ 2 & 4 & 4 & 2 & 0 \\ 2 & 4 & 10 & 2 & 6 \\ 0 & 2 & 2 & 2 & 0 \\ 0 & 0 & 6 & 0 & 7 \end{bmatrix}$$

The proposed zero-trust strategy means all the VMs could be malicious, so the threat scores for all VMs are set to 1 ($t_s(v_i) = 1$). Thus, the co-resident risk indicator matrix is:

$$rcr(v_i, v_j) = 1 \times CoRes(v_i, v_j) \times 1 = CoRes(v_i, v_j) \quad (9)$$

4.3.3 Configuration interval

In the simulated system, five-time steps are chosen to represent t_1 and ten-time steps to represent the configuration interval t_2 . In **Figure 5**, time is represented in a vertical direction. In order to mitigate the co-resident attacks, the system is designed to train the agent to avoid two VMs sharing the same PM for more than t_2 time interval. Based on the timeline of the attacks illustrated in **Figure 3**, different values will be assigned to the *CoResFactor* as shown in Eq. (2).

4.3.4 Risk mitigation strategies

When two VMs have overlapped time steps less than t_1 , there is a minor risk of co-resident attacks, so $\alpha_0 = 0$; when two VMs have resided on the same PM for more than t_1 time steps, but less than t_2 , co-resident attack risks start to accumulate. Thus, the first risk mitigation function is set to be: $\alpha_1 = k \times (t - t_1)$, while k has been chosen from $\{0, 0.25, 0.5, 1, 2\}$ to explore the efficiency of different choices. When two VMs have co-residence on the same PM for more than t_2 time steps, there is enough construction time for attacks to take place, so a more aggressive factor in the form of k_2 is added to the reward function. The proposed system applies the second risk mitigation function: $\alpha_2 = k \times (t - t_1) + k_2$, where k_2 has been tested in the pool of $\{0, 1, 2, 3, \dots\}$. A portion of the risk mitigation function design can be found in **Figure 6**, where all the k values are presented; only $k_2 = 0$, $k_2 = 1$, and $k_2 = 2$ on top $k = 2$ are shown on the graph.

4.3.5 Software

The system is programmed in Python with the flowchart illustrated in **Figure 7**. First, the arriving VMs are placed in a backlog queue. If the queue is not empty, the scheduling system operates to find the optimized solution to assign VMs to PMs. At each time step, the system will update the co-resident duration matrix which reflects the current risk level and will guide the choice of risk mitigation strategies.

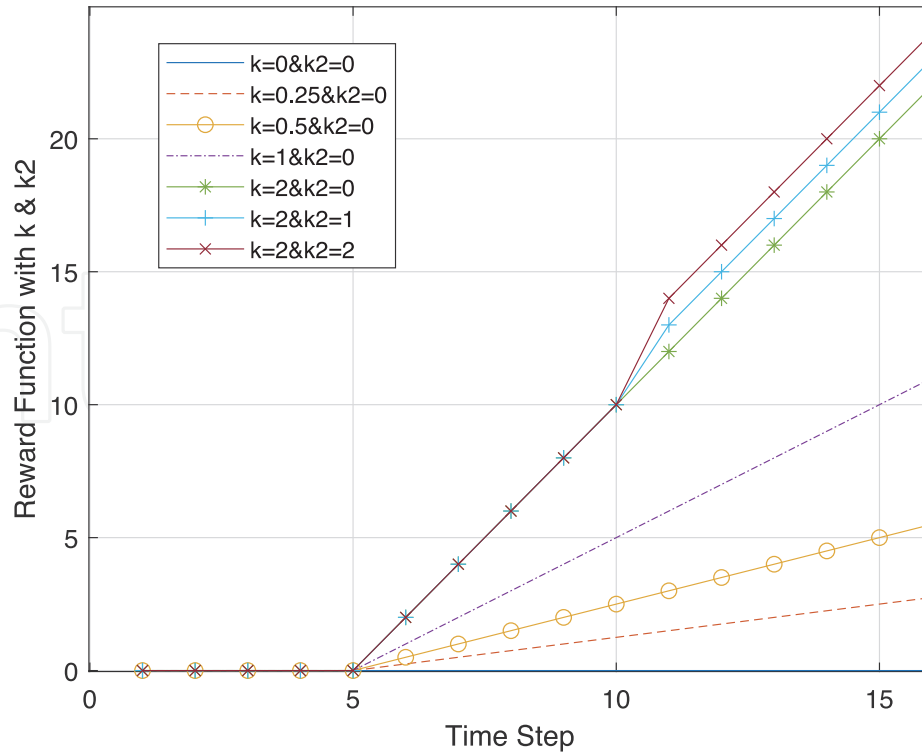


Figure 6.
Risk mitigation function design.

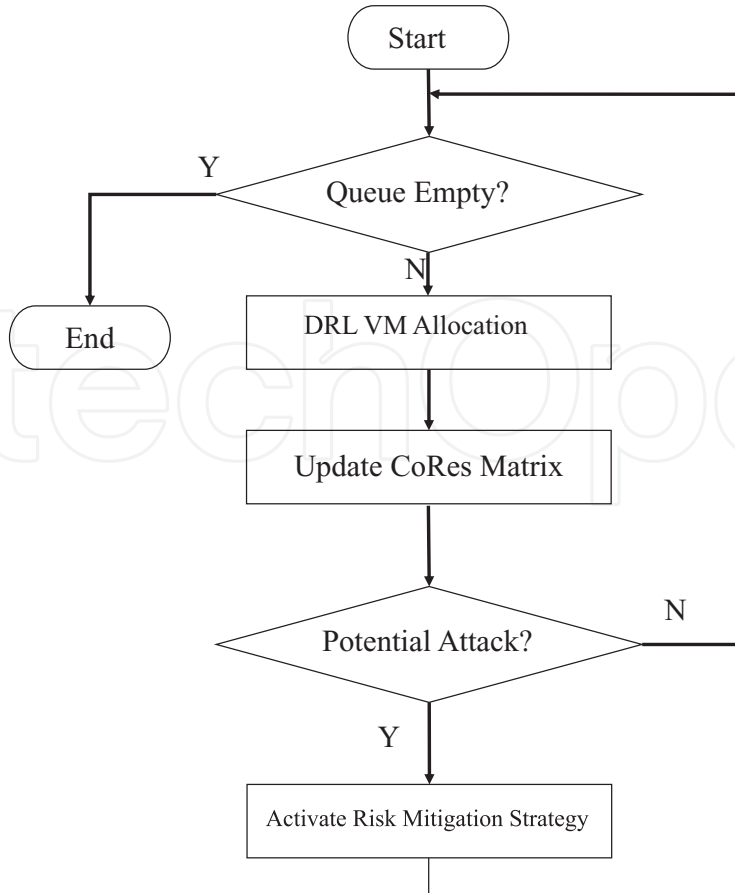


Figure 7.
The flowchart of the proposed system.

5. Simulation results analysis

The co-resident simulation program is built upon the Python-based DeepRM [44] open-source platform. The neural network is constructed by a fully connected hidden layer with 20 neurons, and a total of 89,451 parameters. Poisson distribution with a new arriving rate of 0.7 is chosen to simulate the VMs' dynamic arrival. All the results are obtained in 2500 iterations.

Our proposed system introduces co-resident risk mitigation to task scheduling by adding Eq. (7) to the total rewards calculation of Eq. (8). During the investigation, it was observed that the system performed differently, while manipulating the risk mitigation function parameters illustrated in **Figure 6**. The effectiveness of the proposed mitigation scheme can be analyzed by the RL rewards, VM slowdown ratio, and attack reduction.

5.1 Total rewards affected by risk mitigation factors

As illustrated in Eq. (8), total discounted rewards can be captured by taking both VM delay and co-resident risks into consideration. Since there is no preference between the two, ω_1 and ω_2 in Eq. (8) are both set to 1. In the first experiment, k_2 is set to 0, and k is chosen from 0, 1, and 2. The recorded total accumulated rewards in **Figure 8** explain that a smaller k value leads to a larger reward (Note: the reward is negative). When $k = 0$ there is no risk mitigation. As k increases, more mitigation influence will be placed in the system, and the total discounted reward decreases.

DeepRM provides DRL and other heuristics VM allocation methods, such as tetris, random allocation, and small jobs first (SJF), for comparison. Although those methods do not have co-resident risk mitigation features, the total discounted rewards can illustrate how severe the cybersecurity risks they are experiencing. Two user cases are

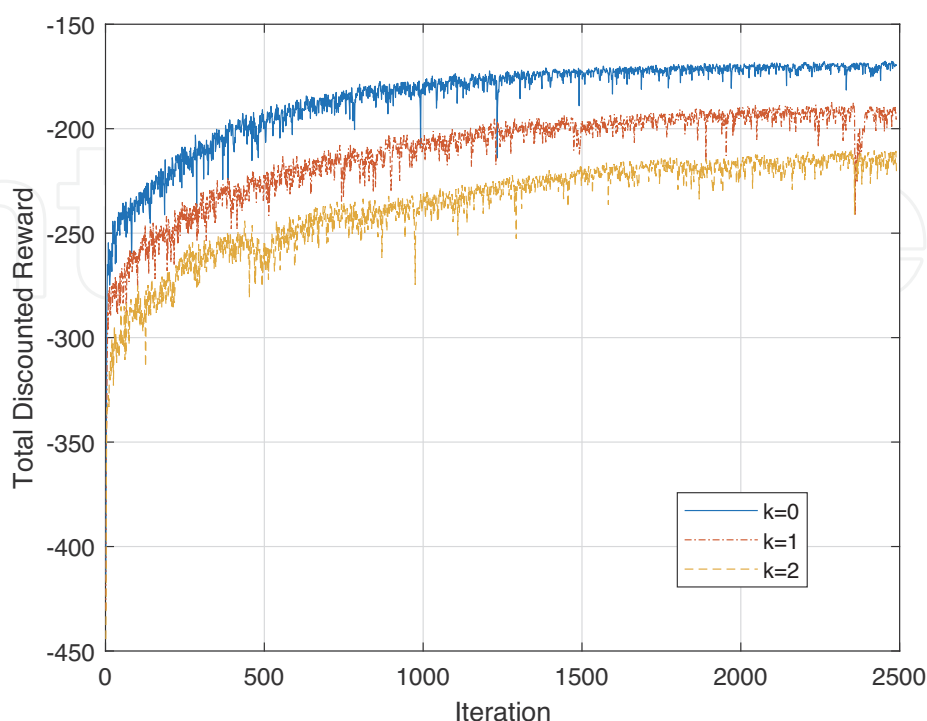


Figure 8.
The total rewards accumulated from different k values.

Methods	Tetris	Random	SJF	DRL
Case 1	-644.94	-426.81	-501.40	-320.17
Case 2	-267.88	-167.50	-187.33	-142.69

Table 3.
Total discounted rewards.

shown in **Table 3**. A negative number with a larger absolute value means a worse situation.

5.2 Slowdown ratio affected by risk mitigation factors

The metric to measure the efficiency of VM scheduling is to calculate the $VM_{Delayed}$ as defined in **Table 2**. In programming, the slowdown ratio is utilized. Each VM has its own expected life cycle shown as the “Ideal length of finishing” in **Table 2**. It also has a length marked by time step when generated. When the VM is assigned to a PM, the “Start time” is marked. At the time of finishing, a “Finish time” is recorded. The parameter *Slowdown* is calculated by Eq. (10). Ideally, if there is no delay in the execution, $Slowdown = 1$, but in the actual application, many factors can cause the delay. Thus, $Slowdown \geq 1$.

$$Slowdown = (FinishTime - StartTime) / VMLength \tag{10}$$

With an increment of k value, more rewards are generated to mitigate the potential co-resident risks through the risk mitigation function. As a matter of fact, it sacrifices the VM completion time, so the slowdown ratio increases. Experiments show that “Random” allocation of VMs has the largest average slowdown ratio. If using “Random” slowdown ratio as a baseline, the percentage of slowdown ratio reduction from the baseline data is shown in **Table 4**.

5.3 Co-resident attacks reduction by risk mitigation factors

Considering the goal of mitigating co-resident attacks, a group of experiments is conducted to represent the effectiveness of different risk mitigation function parameters under RL scenario. **Figure 9** illustrates the total counts of co-resident attacks if k and k_2 are set as in **Figure 6**. If $k = 2$ and $k_2 = 1$, the count reduces dramatically compared with $k = 0$ and $k_2 = 0$, where there is no mitigation applied.

6. Conclusions and future work

This chapter addresses the importance of cybersecurity awareness in cloud computing resource management. The proposed RL-based scheduling method takes both

Methods	Tetris	SJF	DRL ($k = 0$)	DRL ($k = 1$)	DRL ($k = 2$)
ASR	63%	60%	72%	71%	68%

Table 4.
VM slowdown ratio over random method.

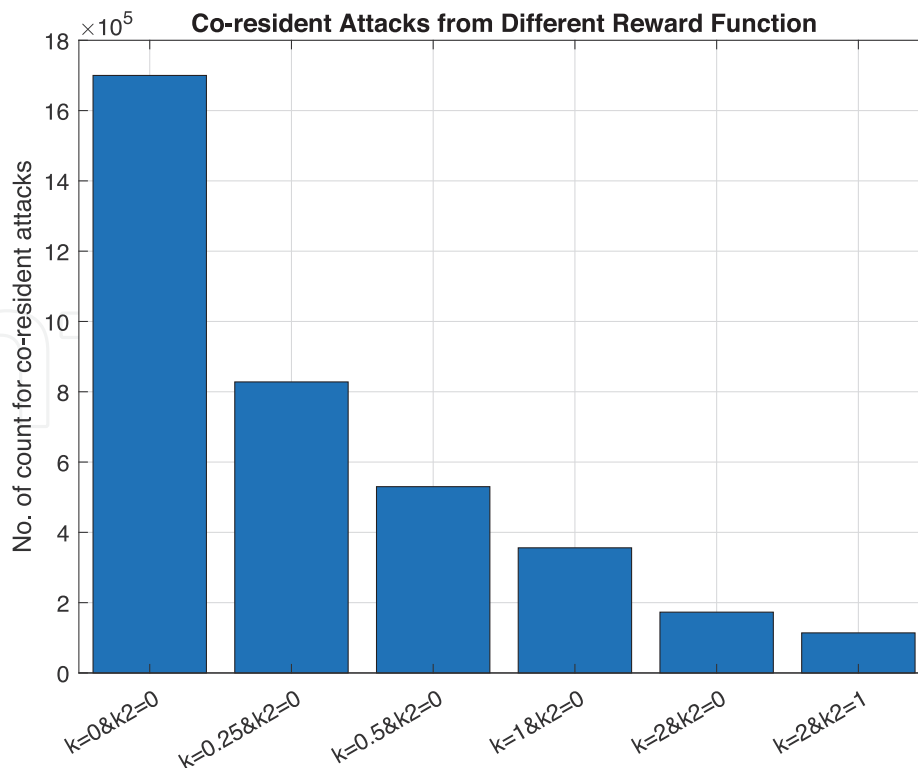


Figure 9.
The potential co-resident attacks by different k and k_2 selection strategies.

VM slowdown time and co-resident attack risks mitigation into consideration. The co-resident risk model under no-trust conditions is formed. As a result, the problem is narrowed down to minimizing the co-tenancy on the same PM among all the active VMs. Finally, a DRL-based task scheduling system is simulated with proposed risk mitigation factors.

This chapter proves that much can be explored in resource management and risk mitigation in cloud computing. It is evident that ML obtained much attention recently, and more applications are being developed in this direction. Although there is a concern about training costs under a deep learning algorithm, it outperforms other methods in adaptation to a more dynamic environment, which makes it outstanding. If designed properly, the computational burden can be shifted to off-line. The above experiment results are obtained by using MacBook Air with a 2.2GHz dual-core Intel i7 processor and 8GB memory. It takes 3 minutes per 2500 iterations to train the policies. While applying the pre-trained model to take actions during runtime testing, it will not take longer than 2 seconds for the longest allocation decision. The results show applying reinforcement learning to co-resident risk mitigation is plausible. Different mitigation strategies lead to different VM completion ratios and risk levels. The proposed strategies proved to be helpful in searching for VM allocation improvement with consideration of both VM completion constraints and co-resident risk awareness. In the future, a more in-depth investigation of the reward equation design will be conducted. A thorough search accompanied by mathematical models to discuss the convergence will be explored. An advanced cost function will be developed with resources and security constraints. Multi-agent reinforcement learning will be applied to extend the model of this research and the efficiency will be tested and compared.

Acknowledgements

This work was supported in part by the Air Force Research Laboratory and Department of Education MSEIP grant award no. P120A180114, Texas A & M Engineering Experiment Station Annual Research Conference Project (TEES TARC) Award 28-235980-00020, and the National Science Foundation grant award no. OAC 1827243. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory or the U.S. Government.

Conflict of interest

The authors declare no conflict of interest.

Author details

Suxia Cui^{1*†} and Soamar Homsi^{2†}


1 Prairie View A&M University, Prairie View, TX, USA

2 US Air Force Research Laboratory, Rome, NY, USA

*Address all correspondence to: sucui@pvamu.edu

† These authors contributed equally.

IntechOpen

© 2022 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/3.0>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. 

References

- [1] Hossain S. Chapter 1 Cloud Computing Terms, Definitions, and Taxonomy. In: *Cloud Computing Service and Deployment Models: Layers and Management*. IGI Global. 2013. pp. 1-25
- [2] BarbosaAndrea FP, Charão AS. Impact of pay-as-you-go cloud platforms on software pricing and development: A review and case study. *Computational Science and Its Applications*. 2012;**7336**: 404-417
- [3] Smith JE, Nair R. *Virtual Machines: Versatile Platforms for Systems and Processes*. Amsterdam, Netherlands: Elsevier; 2005
- [4] Tank D, Aggarwal A, Chaubey N. Virtualization vulnerabilities, security issues, and solutions: a critical study and comparison. *International Journal of Information Technology*. 2022;**14**:847–862
- [5] Hasan MM, Rahman MA. Protection by Detection: A Signaling Game Approach to Mitigate Co-Resident Attacks in Cloud. In: *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*. Honolulu, HI, USA; 2017. pp. 552-559
- [6] Ding J, Sha L, Chen X. Modeling and evaluating IaaS cloud using performance evaluation process algebra. In: *2016 22nd Asia-Pacific Conference on Communications (APCC)*. Yogyakarta, Indonesia. 2016. pp. 243-247
- [7] Addya SK, Turuk AK, Satpathy A, Sahoo B, Sarkar M. A strategy for live migration of virtual machines in a cloud federation. *IEEE Systems Journal*. 2019; **13**(3):2877-2887
- [8] Velayudhan Kumar MR, Raghunathan S. Heterogeneity and thermal aware adaptive heuristics for energy efficient consolidation of virtual machines in infrastructure clouds. *Journal of Computer and System Sciences*. 2016;**82**(2):191-212
- [9] Mthunzi SN, Benkhelifa E, Alsmirat MA, Jararweh Y. Analysis of VM communication for VM-based cloud security systems. In: *2018 Fifth International Conference on Software Defined Systems (SDS)*. 2018. pp. 182-188
- [10] Sane BO, Niang I, Fall D. A review of virtualization, hypervisor and VM allocation security: Threats, vulnerabilities, and countermeasures. In: *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*. Las Vegas, NV, USA. 2018. pp. 1317-1322
- [11] Navamani BA, Yue C, Zhou X. Discover and Secure (DaS): An Automated Virtual Machine Security Management Framework. In: *2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC)*. Orlando, FL, USA. 2018. pp. 1-6
- [12] Kong T, Wang L, Ma D, Xu Z, Yang Q, Chen K. A Secure Container Deployment Strategy by Genetic Algorithm to Defend against Co-Resident Attacks in Cloud Computing. In: *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/ SmartCity/DSS)*. Zhangjiajie, China: IEEE; 2019. pp. 1825-1832
- [13] Qiao A, Choe SK, Subramanya SJ, Neiswanger W, Ho Q, Zhang H, et al. Pollux: Co-adaptive cluster scheduling

- for goodput-optimized deep learning. In: 15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21). Virtual Conference; 2021. pp. 1-18
- [14] Xiao W, Ren S, Li Y, Zhang Y, Hou P, Li Z, et al. AntMan: Dynamic scaling on GPU clusters for deep learning. In: 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20). virtual conference; 2020. pp. 533-548
- [15] Gu J, Chowdhury M, Shin KG, Zhu Y, Jeon M, Qian J, et al. Tiresias: A GPU cluster manager for distributed deep learning. In: 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19). Boston, MA: USENIX Association; 2019. pp. 485-500
- [16] Zhao J, Rodríguez MA, Buyya R. A deep reinforcement learning approach to resource management in hybrid clouds harnessing renewable energy and task scheduling. In: 2021 IEEE 14th International Conference on Cloud Computing (CLOUD). Chicago, IL, USA; 2021. pp. 240-249
- [17] Gupta K, Katiyar V. Survey of resource provisioning heuristics in cloud and their parameters. *International Journal of Computational Intelligence Research*. 2017;**13**(5):1283-1300
- [18] Gawali MB, Shinde SK. Task scheduling and resource allocation in cloud computing using a heuristic approach. *Journal of Cloud Computing*. 2018;**7**(1):4
- [19] Dorigo M, Birattari M, Stutzle T. Ant colony optimization. *IEEE Computational Intelligence Magazine*. 2006;**1**(4):28-39
- [20] Qin Y, Wang H, Zhu F, Zhai L. A multi-objective ant colony system algorithm for virtual machine placement in traffic intense data centers. *IEEE Access*. 2018;**6**:58912-58923
- [21] Tawfeek MA, El-Sisi A, Keshk AE, Torkey FA. Cloud task scheduling based on ant colony optimization. In: 2013 8th International Conference on Computer Engineering Systems (ICCES). Cairo, Egypt; 2013. pp. 64-69
- [22] Patel KD, Bhalodia TM. An efficient dynamic load balancing algorithm for virtual machine in cloud computing. In: 2019 International Conference on Intelligent Computing and Control Systems (ICCS). Madurai, India; 2019. pp. 145-150
- [23] Jia H, Liu X, Di X, Qi H, Cong L, Li J, et al. Security strategy for virtual machine allocation in cloud computing. *Procedia Computer Science*. 2019;**147**: 140-144
- [24] Miao F, Wang L, Wu Z. A VM placement based approach to proactively mitigate co-resident attacks in cloud. In: 2018 IEEE Symposium on Computers and Communications. Natal, Brazil: IEEE; 2018. pp. 00285-00291
- [25] Han Y, Chan J, Alpcan T, Leckie C. Virtual machine allocation policies against co-resident attacks in cloud computing. In: 2014 IEEE International Conference on Communications (ICC). Sydney, NSW, Australia: IEEE. 2014. pp. 786-792
- [26] Yang J, Jiang B, Lv Z, Choo KKR. A task scheduling algorithm considering game theory designed for energy management in cloud computing. *Future Generation Computer Systems*. 2020; **105**:985-992
- [27] Patra MK, Sahoo S, Sahoo B, Turuk AK. Game theoretic approach for real-time task scheduling in cloud

- computing environment. In: 2019 International Conference on Information Technology (ICIT). Bhubaneswar, India: IEEE; 2019. pp. 454-459
- [28] Han K, Cai X, Rong H. An evolutionary game theoretic approach for efficient virtual machine deployment in green cloud. In: 2015 International Conference on Computer Science and Mechanical Automation (CSMA). Hangzhou, China; 2015. pp. 1-4
- [29] Narwal P, Singh SN, Kumar D. Predicting strategic behavior using game theory for secure virtual machine allocation in cloud. In: Networking Communication and Data Knowledge Engineering. Singapore: Springer; 2018. pp. 83-92
- [30] Narwal P, Kumar D, Singh SN. A hidden markov model combined with Markov Games for intrusion detection in cloud. *Journal of Cases on Information Technology (JCIT)*. 2019;21(4):14-26
- [31] Han Y, Alpcan T, Chan J, Leckie C. Security games for virtual machine allocation in cloud computing. In: International Conference on Decision and Game Theory for Security. Fort Worth, TX, USA: Springer; 2013. pp. 99-118
- [32] Pahlevan A, Qu X, Zapater M, Atienza D. Integrating heuristic and machine-learning methods for efficient virtual machine allocation in data centers. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. 2017;37(8):1667-1680
- [33] Witanto JN, Lim H, Atiquzzaman M. Adaptive selection of dynamic VM consolidation algorithm using neural network for cloud resource management. *Future Generation Computer Systems*. 2018;87:35-42
- [34] Zhang J, Xie N, Zhang X, Yue K, Li W, Kumar D. Machine learning based resource allocation of cloud computing in auction. *Comput Mater Continua*. 2018;56(1):123-135
- [35] Liu Z, Zhang H, Rao B, Wang L. A reinforcement learning based resource management approach for time-critical workloads in distributed computing environment. In: 2018 IEEE International Conference on Big Data (Big Data). Seattle, WA, USA: IEEE; 2018. pp. 252-261
- [36] Joseph L, Mukesh R. To detect malware attacks for an autonomic self-heal approach of virtual machines in cloud computing. In: Fifth International Conference on Science Technology Engineering and Mathematics (ICONSTEM). Chennai, India: IEEE; 2019. pp. 220-231
- [37] Abazari F, Analoui M, Takabi H. Multi-objective response to co-resident attacks in cloud environment. *International Journal of Information and Communication Technology Research*. 2017;9(3):25-36
- [38] 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: International Conference on Computing, Networking and Communications (ICNC). Maui, HI, USA: IEEE; 2018. pp. 349-353
- [39] Berrima M, Nasr AK, Ben RN. Co-location resistant strategy with full resources optimization. In: Proceedings of the 2016 ACM on Cloud Computing Security Workshop. Hofburg Palace, Vienna, Austria; 2016. pp. 3-10
- [40] Levitin G, Xing L, Dai Y. Co-residence based data vulnerability vs. security in cloud computing system with

random server assignment. *European Journal of Operational Research*. 2018; **267**(2):676-686

[41] Xing L, Levitin G. Balancing theft and corruption threats by data partition in cloud system with independent server protection. *Reliability Engineering & System Safety*. 2017;**167**:248-254

[42] Zhang Y, Li M, Bai K, Yu M, Zang W. Incentive compatible moving target defense against vm-colocation attacks in clouds. In: *IFIP International Information Security Conference*. Heraklion, Crete, Greece: Springer; 2012. pp. 388-399

[43] Wang X, Wang L, Miao F, Yang J. SVMDF: A secure virtual machine deployment framework to mitigate co-resident threat in cloud. In: *2019 IEEE Symposium on Computers and Communications (ISCC)*. Barcelona, Spain; 2019. pp. 1-7

[44] Miao H, Alizadeh M, Menache I, Kandula S. Resource management with deep reinforcement learning. In: *HotNet '16: Proceedings of the 15th ACM Workshop on Hot Topics in Networks*. Atlanta, Georgia, USA. 2016. pp. 50-56

[45] Mao H, Schwarzkopf M, Venkatakrisnan SB, Meng Z, Alizadeh M. Learning Scheduling Algorithms for Data Processing Clusters. In: *Proceedings of the 2019 ACM Special Interest Group on Data Communication (SIGCOMM)*. Beijing, China; 2019. p. 270-288

[46] Tuli S, Ilager S, Ramamohanarao K, Buyya R. Dynamic scheduling for stochastic edge-cloud computing environments using A3C learning and residual recurrent neural networks. In: *IEEE Transactions on Mobile Computing*. 2022 March;**21**(3):940-954

[47] Tuli S, Poojara SR, Srirama SN, Casale G, Jennings NR. COSCO: Container Orchestration Using Co-Simulation and Gradient Based Optimization for Fog Computing Environments. *IEEE Transactions on Parallel and Distributed Systems*. 2022;**33**(1):101-116

[48] Paeng B, Park IB, Park J. Deep reinforcement learning for minimizing tardiness in parallel machine scheduling with sequence dependent family setups. *IEEE Access*. 2021;**9**(10):1390-1401

[49] Asheralieva A, Niyato D, Xiong Z. Auction-and-learning based lagrange coded computing model for privacy-preserving, secure, and resilient mobile edge computing. In: *IEEE Transactions on Mobile Computing*. 2021;early access. pp. 1-2