Intelligent Automation & Soft Computing
DOI:10.32604/iasc.2021.019024
Article

A Multi-Objective Secure Optimal VM Placement in Energy-Efficient Server of Cloud Computing

Sangeetha Ganesan* **and Sumathi Ganesan**

Department of Information Technology, Sri Venkateswara College of Engineering, Sriperumbudur, 602117, India
*Corresponding Author: Sangeetha Ganesan. Email: gsangeethakarthik@gmail.com
Received: 30 March 2021; Accepted: 01 May 2021

Abstract: Cloud Computing has been economically famous for sharing the resources of third-party applications. There may be an increase in the exploitation of the prevailing Cloud resources and their vulnerabilities as a result of the aggressive growth of Cloud Computing. In the Cache Side Channel Attack (CSCA), the attackers can leak sensitive information of a Virtual Machine (VM) which is co-located in a physical machine due to inadequate logical isolation. The Cloud Service Provider (CSP) has to modify either at the hardware level to isolate their VM or at the software, level to isolate their applications. The hardware isolation requires changes in hardware design and it also leads to performance degradation. The existing works challenge to eliminate CSCA. But our solution prevents CSCA by securing the VM placement in the energy-efficient server. This paper proposes a heuristic secure VM placement policy by using different VM placement techniques (Least VM, Multi-Objective Ant Colony System (MOACO), Previously Assigned Physical Host First (PAPHF)) based on user behaviours to prevent co-resident. The Co-Resident Rate for the three VM placement policies (Least VM, MOACO, PAPHF) has been compared with the existing First Fit Decreasing (FFD) algorithm and Discrete Firefly Algorithm-VM Placement (DFA-VMP) policy in the CloudSim environment. The Power Consumption, Resource Wastage, and Co-Resident probability are compared with the existing FFD and DFA-VMP Systems. Our Multi-Objective Secure Optimal VM Placement (MOSOVMP) proposed system proved that the Co-Resident Rate is reduced with improved server utilization.

Keywords: Cloud computing; virtual machine placement; least VM; MOACO; PAPHF; power consumption; resource wastage; co-resident rate

1 Introduction

Side channel analysis is the famous intelligent part of the cryptanalytic attack [1–4]. The secret information is emitted to the attacker from the secure devices by analyzing its physical signals (temperature, power, radiation, heat, laser, etc.). A particular type of Side Channel (SC) attack which is related to personal computers is the Cache Side Channel (CSC) attack. The CSC attack utilizes the use of cache memory as a shared resource between different processors and it releases secret information [5,6].

With the help of the virtualization model, the cloud customers have the provision of resources on-demand on a pay-as-you-go basis. Virtualization is a technology, where the physical servers are changed to logical VMs and may be rented to different tenants. The VMs of different tenants are located on the same physical server and share the underlying hardware resources. Maximum resource utilization may be provided to improve the profit of a CSP. With an increase in its utilization rate of hardware platforms, there arise new security risks. VMs running on the same server are logically isolated from one another. The VM of the malicious user tries to side-step the logical VM isolation and acquire sensitive information about the cryptographic algorithms such as the encryption key from the co-resident [7]. Many side channels have been explored [8–10] to fetch the secret key among the VMs which is forbidden by security policies.

In CSCA, the attacker may fetch the secret information from the victim by measuring the cache usage pattern of the victim. Recent work shows that the Last-Level Cache (LLC) attacks are very powerful. In the LLC attack, the attacker can fetch the fine-grained secret information of the target user with high resolution and low noise [11]. The Prime+Probe, Evict+Time, and Flush+Reload are the special practices used in the LLC-based attacks.

Instead of controlling CSCA with the help of hardware or software isolated applications, our proposed system proactively avoids CSCA by allocating the VM of the victim and the attacker in different Physical Machines. The victim user also may try to retrieve the secret information of others. This CSCA is possible only if the attacker and the victim are co-located on a physical machine. The co-location of the attacker will be identified by the Co-resident detection system. In the proposed system, once the attacker is identified, the attacker will be migrated to other physical machines. So the co-location of a victim with the attacker is avoided.

The rest of the paper is structured as follows; Section 2 describes the Related Work of Cache Side Channel Attacks and VM placement algorithms. Section 3 explains Co-resident Attack Scenarios. Section 4 explains the objectives for secure VM placement. Section 5 explains the working methodology of our proposed work: MOSOVMP and Section 6 shows the Experimental Result.

2 Related Work

2.1 Survey on Security System

There is more hardware and software isolated applications that are proposed for LLC CSCA. In the CATalyst locking mechanism, the Cache Allocation Technology (CAT) is used to partition the cache to secure it from CSCA on the shared LLC [12]. A new cache design technique was developed [13], namely random permutation cache and partition lock cache were used to provide security against cache side-channel attack. In the dynamic page colouring technique, the caches are dynamically partitioned among the secure critical applications of cloud tenants [14]. They partitioned a cache design, in which cache architecture is divided into regions. Each region is allocated to a separate application. It also reduces the interference as compared to a single shared cache. These hardware-based solutions require changes to underlying hardware design and also create performance degradation.

In the Co-Resident Location Algorithm (CLR), all the servers are in either an open state or a close state [15]. If the selected server cannot allocate the newly created VM, then this server is marked as a closed server and a newly open server is assigned for the new VM. The Security-Based Placement (SBP) and Security-Based Selection (SBS) algorithms are proposed for compartment isolation methods to attain security-conscious VM consolidation [16]. It has been observed that this consolidation algorithm develops dynamic VM consolidation algorithms and exhibits similar characteristics compare to non-security aware VM consolidation.

In the Vickrey Clarke Groves (VCG) method, the VMs periodically migrated to solve the Co-resident problem [17]. Particularly, they confer the number of VMs to be migrated as well as the target PMs. They offer a technique to apply a VM allocation plan, to reduce the overall security issue. Conversely, regularly migrating VMs will origin additional power consumption, and may direct to performance degradation, which enlarges the probability of CSP violation of their SLA.

2.2 Survey on VM Placement

The different VM placement policy is proposed to minimize the co-resident of the attacker. Instead of using a VM placement method, a new VM placement policy technique is introduced, in which a VM placement policy will be chosen from the multiple VM placement policy pool based on the co-resident probability of attacker VM. In the secured VM placement technique, newly created VMs are deployed in the previously selected server. The Previously Selected Servers First (PSSF) technique maintained load balance by limiting the number of VM per user. The power consumption is also maintained by framing the fixed groups in the server. But, the legal user can create only a limited number of VMs. The PSSF technique reduces the co-residence of the attackers, but the server resource utilization is very less [18]. Tab. 1 highlights some of the correlated works and refers to their objectives, methodology, and limitations.

Table 1: Survey on VM placement schemes

Reference	Energy-aware	Cost-aware	Resource-aware	security-aware	Methodology	Limitations
[19]	✓				Best Fit Decrease	It can't generate an optimal solution in the entire scenario.
[20]	✓	✓	✓		Ant Colony Optimization	By increasing the number of objectives, may lead to performance degradation and time complexity.
[21]	✓	✓			Harmony Search	Experiments result depends on only random workloads.
[22]	✓				Krill Herd Algorithm	The simulation result of ESV gives a bad result compared to MBFD and GA.
[23]	✓				Cultural algorithm	The simulation result showed SLA violation
[24]	✓	✓			Best fit	The optimal VM placement cannot be assured.
[25]	✓		✓	✓	Firefly Algorithm	The co-resident attacks may occur if more VMs are created by the attacker.

3 Co-resident Attack Scenarios

Let us assume, there is an 'N' number of servers. The 'K' number of VMs started by legitimate users $\{VML_1, VML_2\ VML_3 \ldots VML_k\}$ and 'M' number of VMs started by attackers $\{VMA_1, VMA_2\ VMA_3 \ldots VMA_M\}$. The target of an attacker is a set of VMs that is created by a legitimate user i.e., Target (A) = $\{ VML_1, VML_2\ VML_3 \ldots VML_k\}$. Let Hit_VM(A) denote the VM of the attacker A that co-located with at least one of the VM of the legitimate user. i.e., Hit_VM (A) = {V, | V VM(L), Server (V) }. Hit_Target (A) denote the VMs of the target user L that is co-located with at least one VM of the Attacker A. i.e., Hit_Target (A) = {x | x}. If any of the attacker VM (VMA) is co-located with the target

VM (VML) then, the Hit_VM(A) and Hit_Target(A) will be non-empty. The hit of an attacker is measured in the following two dimensions,

(i) Efficiency: The efficiency of an attacker is defined as the number of PMs that are hosted by an attacker and co-located with at least one of the target VM (VML) divided by the total number of VMs hosted by the attacker.

$$Efficient(VM(A)) = \frac{|PM(Hit\_VM(A))|}{|VM(A)|} \tag{1}$$

(ii) Coverage: The coverage is defined as the number of target VMs co-located by the attacker VM, divided by the number of VMs allocated by the attacker.

$$Coverage(VM(A)) = \frac{|Hit\ (T\arg et(A))|}{|VM(A)|} \tag{2}$$

We now describe and analyze our secure heuristics strategy. We divide all our existing PMs into some groups (μ). The placement of VM to PM is done in two steps: (i) Group Selection and (ii) PM Selection. The probability of Co-Resident in our strategy is defined as

$$P[CoR(v_a, v_t] = P[SG(v_a, v_t)]\, X\, P[SPM(v_a, v_t)] \tag{3}$$

In Eq. (3) we have $P[SPM(v_a, v_t)] \leq 1$, therefore we attain $P[CoR(v_a, v_t)] \leq P[SG(v_a, v_t)]$. $SG(v_a, v_t)$ denotes both VMs (v_a, v_t) are allocated in the same Group. $SPM(v_a, v_t)$ denotes both VMs(v_a, v_t) are allocated in the same PM. $CoR(v_a, v_t)$ denotes The v_a is Co-Resident with v_t. $GS(\Delta, N_a, N_t)$ denotes Group Selection function Δ to select PM group for the cloud customer. $PMS(\delta, N_a, N_t)$ denotes PM selection function δ to select PM for Number of attacker VMs (N_a) and target VMs(N_t). The probability of Co-Resident is reduced by increasing the number of PM groups. PM utilization is reduced by increasing the number of PM groups. So, our proposed heuristic technique reduces the probability of Co-Resident and balancing the number of PM groups and their utilization. The PM group selection $GS(\Delta, N_a, N_t)$ is done with the help of a user behaviour database. The PM selection $PMS(\delta, N_a, N_t)$ is done with the measurement of PM resource utilization. So, the GS function reduces the Co-Resident and PMS function improves the PM resource utilization.

4 Objectives for Secure VM Placement

In our proposed MOSOVMP system, the VMs have been placed according to the following objectives: (i) Security (ii) Resource Wastage, and (iii) Power Consumption.

4.1 Security

With the help of the cloud user database and the cloud user behaviour, the attacker is identified by CSP. The CSC attacker can be detected either with the help of our proposed system (Case A) [26] or with the user behaviour (Case B). Case A: The CRR is measured as follows: step (i): The vCPU utilization and virtual memory utilization of all VMs in a PM are measured and maximum utilized VMs are identified. Step (ii): The cache miss rate is measured with help of our previous work [26] while executing a sensitive operation (cryptography operation) in a VM. If the cache miss rate of a VM is high, the vCPU and virtual memory utilization of the co-resident VMs are also high, and then the co-resident VMs is identified as attackers. Case B: In our proposed system, the cloud user is classified into three types: (i) New user (ii) Trusted user: User creates a limited number of VMs. (iii) Un-trusted user: Even the VMs already created by the tenant are in an idle state. If a tenant is creating more VMs to achieve co-resident with the target user, then the tenant is identified as an Un-trusted user. And if a tenant is terminating some VMs and

sending a request to create new VMs at the same time, then the tenant is identified as an Un-trusted user. To attain co-resident, the attackers try to place their VM with the victim VM in the following ways: (i) Brute-force approach: The attacker creates many VMs to attain co-resident. (ii) Start Time/Termination Time: If the VM created by the attacker does not co-locate with the target VM, then the attacker will terminate the existing VM and try to create new VMs to attain co-residence. (iii) Creating more VMs: The attacker creates more VMs at the same time to achieve co-residence.

4.2 Resource Wastage (RW)

The lingering resources accessible on each server may differ greatly with different VM placement strategies. To entirely utilize multi-dimensional resources, the following equation is used to measure the resource wastages:

$$RW_K = \frac{\delta + \left| W_K^{CPU} - W_K^{MEM} \right|}{U_K^{CPU} + U_K^{MEM}} \tag{4}$$

RW_K represents the resource wastage of the K^{th} server. The $U_K^{CPU} and\ U_K^{MEM}$ represent the utilization of CPU and memory resources of the K^{th} server. The $|W_K^{CPU} - W_K^{MEM}|$ represent the remaining CPU and memory wastage resources. δ is a very minute positive number and its value is fixed to be 0.0001. The idea behind Eq. (4) is to formulate the effective use of the resources.

4.3 Power Consumption (PC)

The Power Consumption of an average data centre is estimated as much as 25,000 households [27] and it is expected to be double every five years [28]. Managing the servers in an energy-efficient way is critical to CSP to reduce the PC. This has been focused on the existing works [28–33]. Ristenpart introduced a power model to formulate the power consumption of all PMs [17]. The PC of a PM is calculated with a linear relationship of CPU utilization with the help of the Eq. (5).

$$PC(PM) = \sum_{i=1}^{M} \phi_{power}\{load(Host_i)\} \tag{5}$$

The power consumption of servers is related to resources such as hard disk, CPU, and memory. Some research work specifies that the power consumption of hosts is a direct proposition with CPU utilization. Moreover, the research work also proposes that the power consumption of inactive PMs is about 70% of the power consumption below full load in Eq. (5), so the power consumption of PMs is defined as follows:

$$PC(PM_i) = \begin{Bmatrix} (PM_i^{busy} - PM_i^{idle})\, X\, U_i^{CPU} + PM_i^{idle}, 0 < U_i^{CPU} \leq 1 \\ 0 \qquad\qquad\qquad\qquad\qquad\qquad , Other \end{Bmatrix} \tag{6}$$

where U_i^{CPU} specifies the CPU utilization rate of i^{th} PM, i = [1, 2,…., M]. PM_i^{busy} *and* PM_i^{idle} represent the power consumption of i^{th} PM at full load and idle state respectively. PC (PM_i) indicates the power consumption of i^{th} PM when its CPU utilization rate is U_i^{CPU}. Therefore, the power consumption for the total data centre can be defined as follows:

$$\begin{aligned} \phi_{MIN} &= MIN \sum_{i=1}^{M} PC(PM_i) \\ &= MIN \sum_{i=1}^{M} [\delta_i X ((PM_i^{busy} - PM_i^{idle}) X \sum_{j=11}^{N} (\lambda_{ij} \cdot \psi_{ij}^{CPU}) + PM_i^{idle}] \end{aligned} \tag{7}$$

In Eq. (7) δ_i is specifies whether the i^{th} PM is in use or not. If it is in use, $\delta_i = 1$, otherwise, $\delta_i = 0$. λ_{ij} specifies that j^{th} VM is allocated on i^{th} PM. If yes, $\lambda_{ij} = 1$, otherwise, $\lambda_{ij} = 0$. ψ_{ij}^{CPU} represents CPU quantity of j^{th} VM and i^{th} PM.

5 Proposed System—MOSOVMP

In Cloud Computing, there are two ways to place the VM in Physical Machine (PM). (i) Initial Placement: after receiving a VM request from the user, based on the VM placement policy the newly created VM is placed in the PM. (ii) Migration: The VM can be moved to another PM. The Cloud Service Provider (CSP) aims to minimize the co-residence of the attacker on the premise of balancing the work loaded and maintaining the low power consumption without SLA violation. The CSP groups ($\mu = 3$) their Physical Machine as $Group_A$, $Group_B$, and $Group_C$ for Unknown user, Trusted User, and Un-trusted User respectively. The VMs of unknown users are assigned in $Group_A$ by using the Least VM placement policy. The VMs of trusted users are allocated in $Group_B$ by using the Multi-Objective Ant Colony System (MOACO) The VMs of un-trusted users are allocated in $Group_C$ by the Previously Assigned Physical Host First (PAPHF) algorithm. The proposed system working technique is given in Fig. 1. The procedure for selecting a group is given in Algorithm 1.

5.1 Least VM Policy

As the Least VM placement policy has a less co-resident rate, the new user VMs are placed in the PM of $Group_A$ by Least VM placement policy. If the new user creates only a limited number of VMs in a specific time period, then the new user will be considered as a trusted user. The VMs of the new user will be migrated to $Group_B$.

Algorithm 1: Placement of VM to PM

Input: VM_i, UserList

Output: PM for VM_i

1. uid ←user_id (VM_i) // Cloud User id
2. if (uid is in Trusted_UserList)
    a. Assign VM_i to $Group_B$ by MOACO
3. else if (uid is in Un-trusted_UserList)
    a. Assign VM_i to $Group_C$ by PAPHF
4. else // New user
    a. Assign VM_i to $Group_A$ by LeastVM
5. endif

5.2 MOACO Policy

The VMs of the trusted user is placed in $Group_B$ by MOACO policy. The MOACO policy simultaneously optimizes resource wastage (CPU and memory), and power consumption. A revised version of the ACO algorithm is designed to covenant effectively with the possible large resolution space. A feasible solution to the originated multi-objective VM placement problem is measured as a combination of all VM placements. The working procedure of the MOACO algorithm is explained in

Algorithm 2. The parameters are initialized ($\omega_l, \omega_g, q_0, \tau_0, \alpha, number\ of\ ant, number\ of\ iteration$) and all pheromone trails are located to τ_0. Each ant receives all VM requests in the iteration part and initiates VM placement to a PM. This is achieved by Pseudo-Random-Proportional (PRP) rule. It illustrates the desirability for an ant to select a specific VM as the next one to bundle to its current PM. This rule is dependent on the knowledge of existing pheromone concentration on the movement and a heuristic that directs the ants on the way to choose the most capable VMs. If an artificial ant has moved, a local pheromone (LP) is updated. Once all ants have built their solutions, then a Global Pheromone (GP) is updated with every result of the current Pareto set P. *The* initial pheromone level is measured as follows

$$\tau_0 = \frac{1}{[n \cdot (PC'(Sol_0) + RW(Sol_0))]} \tag{8}$$

where n is the number of VMs. Sol0 is the solution generated by the First Fit Decreasing (FFD) heuristic, RW(Sol0) is Resource Wastage of Sol0 and PC'(Sol0) is the power consumption of Sol0. The Sol0 is measured as follows:

$$PC'(Sol_0) = \sum_{i=1}^{M} \frac{PC_i}{PC_i^{MAX}} \tag{9}$$

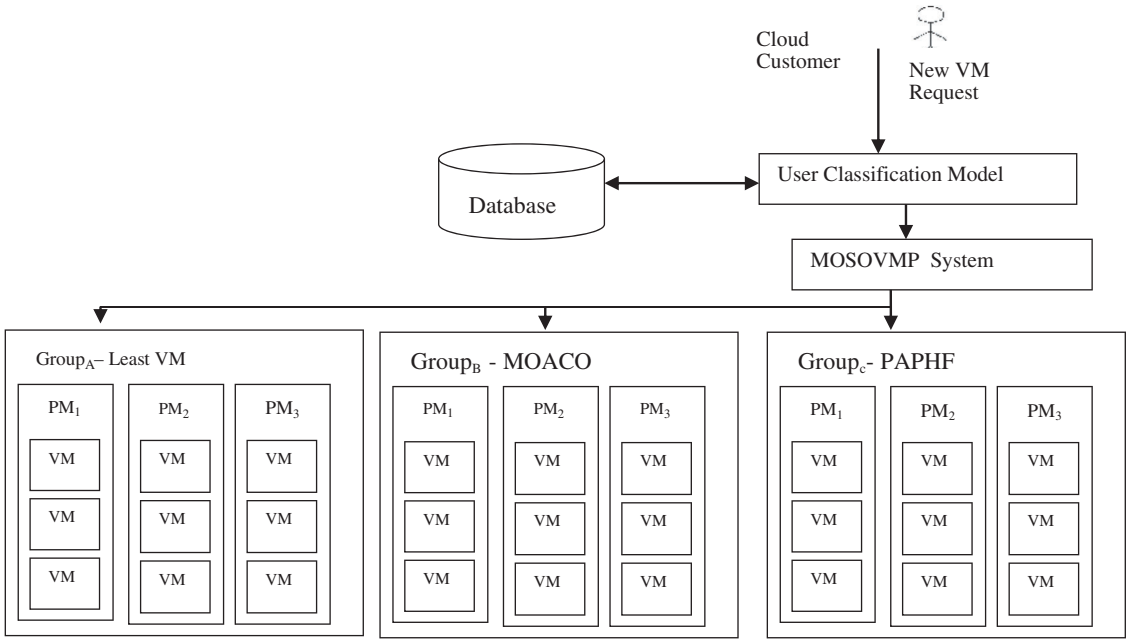

Figure 1: Proposed System—MOSOVMP workflow

The heuristic information is represented by $\mu_{i,j}$. This denotes the desirability of placing VM_i to VM_j. To perfectly assess the desirability of each placement, the heuristic solution is computed based on the current state of the ant. As this solution is measured for all ants in all movements, it may affect the efficiency. To conquer such obscurity it should be measured in a well-organized mode. Let *PL* be a list created of all the PMs. When building a solution, every ant begins with the set of all VMs to be placed in the PM and the list PL ordered randomly. It primarily places VMs one by one to the first PM in the list PL, and then places to the second PM and so on till all VMs are placed. So, while measuring the value of $\mu_{i,j}$, the permutation of VM placements from the PM_1 to PM_j is identified. The partial contribution of placing VM_i to PM_j for the PC objective and *Resource Wastage* function can be measured as follows:

$$\mu_{i,j}^{I} = \frac{1}{\delta + \sum_{K=1}^{j} \frac{PC_K}{PC_K^{MAX}}} \tag{10}$$

$$\mu_{i,j}^{II} = \frac{1}{\delta + \sum_{K=1}^{j} RW_K} \tag{11}$$

Algorithm 2: Multi-objective ant colony optimization (MOACO)

Input: $Group_B$ [PM], VM List

Output: Placement of VM_i to $Group_B$ [PM]

1. Set values of parameters, $\omega_l, \omega_g, q_0, \tau_0, \alpha, N, M$
   // N- Number of Ants; M- Number of iteration
2. Initialize
   P set to empty
   All pheromone values to τ_0
3. For i = 1 to N do // To place all VMs
   a. Initialize PL as a set of Sorted PM list (random order)
   b. For i = 1 to PL. Size do // VMs fit in the server
      1. For k = 1 to VM List. size do
         (i) Measure the desirability according to Eq. (12)
         (ii) Measure the probability according to Eq. (14)
      End for
      2. Draw q
      3. If q < = q_0 then
         Identified as a best solution
         Else Search for new solution
         Endif
      4. Update LP value according to Eq. (15)
   End for
   End for
4. For k = 1 to set P. Size do
   Update GP value according to Eq. (16)
   End for
5. End for // maximum number of iteration
6. Return set P.

Therefore the total desirability of placing VMi to PMj is measured as follows:

$$\mu_{i,j} = \mu_{i,j}^{I} + \mu_{i,j}^{II} \tag{12}$$

In the method of making placements, the ant k picks a VM_i as the next one to bundle to its current PM_j *based on* the following PRP rule.

$$i = \begin{cases} \arg MAX_{v\in\Omega_k(j)}\{\alpha X \tau_{v,j} + (1-\alpha) X \mu_{v,j}\}, & q \le q_0 \\ R & otherwise \end{cases} \tag{13}$$

The α is a parameter that permits a user to manage the relative importance of the pheromone trail and q is a random number consistently distributed in [0, 1]. If the q *value* is greater than q_0, then a new solution is searched, otherwise, an optimal solution is selected. q_0 is a fixed parameter $[0 \le q\ 0 \le 1]$ calculated by the comparative significance of exploitation of accrued information about the problem against to exploration of new travels. R is a random variable chosen based on the following PRP rule probability distribution [34], which is the probability that ant k selects to place VM_i to PM_j:

$$P_{i,j}^{k} = \begin{cases} \dfrac{\alpha X \tau_{i,j} + (1-\alpha)\ X\ \mu_{i,j}}{\sum_{s\in\Omega_k(j)} (\alpha X \tau_{s,j} + (1-\alpha) X \mu_{s,j}} & i \in \Omega_k(j) \\ 0, & Otherwise \end{cases} \tag{14}$$

Another essential component of MOVMA is the update of Pheromone Trails (PT). The PT value can raise, as ants drop pheromone. It can drop off, due to pheromone evaporation. The deposit of new pheromone is depended on the information enclosed in some good solutions that should be shown by PT and the movement included in these good solutions will be influenced by other ants constructing subsequent solutions. In the proposed MPVMP algorithm, the pheromone updating method includes two steps: a LP update and a GP update. While assigning a placement of VM_i to PM_j, an ant shrinks the PT intensity between VM_i and PM_j by concerning the following LPU rule:

$$\tau_{i,j}(T) = (1-\omega_l)\ \tau_{i,j}(T-1)\ +\ \omega_l \cdot \tau_0 \tag{15}$$

where τ_0 is the initial pheromone level and $\omega_l (0 < \omega_l < 1)$ is the LP evaporating parameter. The GP update rule is used after all ants have completed building a solution. Hence all Pareto solutions are concerned as best or optimal solutions for a multi-objective optimization problem, we assume that all non-dominated solutions have the same and highest quality and all dominated solutions must be avoided. So, the GP update is performed for every solution S of the present Pareto set (P) by concerning the following rule:

$$\tau_{i,j}(T) = (1-\omega_g)\ \tau_{i,j}(T-1)\ +\ \frac{\omega_g \cdot \gamma}{PC'(S)\ +\ RE(S)} \tag{16}$$

$$\gamma = \frac{N}{T - M(S)\ +\ 1} \tag{17}$$

Then all results conquered by the added one are removed from the external set. N represents the number of ants and M represents the number of iteration. In Eq. (17), the adaptive coefficient γ is used to manage how a solution(S) in the external set donates to pheromone information over time. This GP updating rule attempts to enhance the learning of ants.

5.3 PAPHF Policy

In $Group_C$, VMs are placed based on the PAPHF algorithm where the VMs of a specific user is only allowed into a Physical Machine. In $Group_C$, every Physical Machine has a unique user id. In PAPHF, the user id of the customer is compared with the user id of all the physical machines. If it matches any of the

user ids of the PM, then the VM is allocated into that specific PM of $Group_C$. For example, The VM of user A is allocated to a PM of $Group_C$, Where the previous VM of user A pre-exists. If it doesn't match with any of the user ids of the physical machines, then the VM is allocated into a new physical machine of $Group_A$. The VM of no two users is co-located in a PM. The PAPHF is explained in Algorithm 3. In PAPHF, the user id of the newly created VM is compared with the user id of the existing Physical Host (PH). If it is a match, then the resource request of the newly created VM is checked with the selected PH resources for compatibility. Otherwise, a new PH will be assigned for the new VM request in $Group_A$.

Algorithm 3: Previously assigned physical host first

Input: $Group_C$ [PM], Reqested_VM, PHList
Output: placement of Reqested_ VM to PM
1. N← Size[PHList]
2. UID ← Reqested_ VM(uid)
3. min_Power ←MAX
4. allocatedPH← Null
5. for each PH_i in PHList do
    a. if(UID = = PH_i(UID))
      i. if PHi has enough resources for Reqested_VM //Power Comsumption
        a. $Power_i$ ← estimatedPower(PH_i, Reqested_VM)
        b. if ($power_i$ < min_Power)
          i. allocatedPH←PH_i
        c. min_Power← $Power_i$
          endif
      endif
    endfor
6. if(allocatedPH = = Null)
    a. allocatedPH = New_PH
    b. PHi(UID) = UID
7. endif
8. return allocatedPH

6 Evaluation and Result

This research proposes to reduce the co-resident problem through different VM placement policies and also to improve server energy efficiency. The CloudSim toolkit [35] is a powerful simulation platform for the cloud computing environment. It also supports simulating various self-defined VM placement policies. Therefore, CloudSim is chosen to implement the proposed system. Each PM has one CPU core of 1000, 2500, or 3000 MIPS, and 8 GB of RAM. Each VM needs 128 MB of RAM, and one CPU core with 250, 500, 750 or 1000 MIPS. Three groups (μ) of VMs were created with a different number of the CPU

core, MIPS, and RAM. After receiving the VM request of the customer, the CSP will identify its nature. Initially, basic VM placement policies were implemented and Co-Resident was evaluated (Fig. 2).

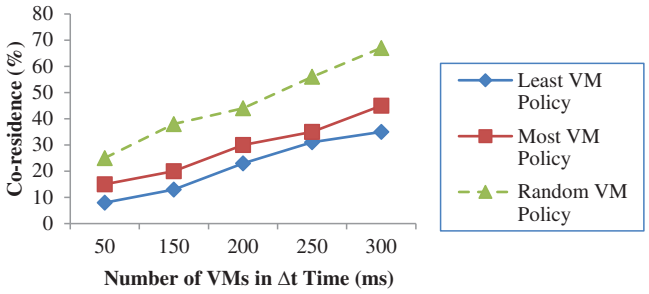

Figure 2: Co-Resident for different VM placement policy

To identify the least possible Co-Resident placement policy, a set of VMs is placed by Random VM policy, Least VM policy, and Most VM policy. In this comparison, the least VM placement policy has less co-resident value compared to other policies. So, the least VM placement policy was chosen for the unknown users in the $Group_A$. Let 'N' be the number of VM created by unknown users. Let 'M' be the number of PMs assigned by N. By using the Least VM placement policy, a new VM is placed in the server, which has a fewer number of VMs. So, the number of PM is lesser than the number of the user (M < N). A legal user L starts 20 VMs at the 24,000th second (VM (L) = 20) and a certain time (Δt) later '10' number of VMs are started by an attacker (VM(A,Δt) = 10). The attacker |VM (A, Δt)| = 1, 5, 10, 15, 20, 25, …, 100 is increased gradually at Δt = 1, 5, 10, 15, 20, 25, 30, …, 100 (ms). The experiment was repeated 50 times for various numbers of VMs with various time Δt slices. The final results presented below are the average values.

The VMs of the trusted users are placed in the PM by the MOACO placement policy. In the Multi-Objective ACO algorithm, PMs are selected based on PM energy efficiency and resource wastage. Trusted users may also try to get sensitive information about others. The co-resident detection system frequently monitors user behaviour. The attacker VMs are selected from the $Group_B$ and named as $VM_{A(B)}$. i.e., $VM_{A(B)} \in Group_B$ The set of $VM_{A(B)}$ is migrated to $Group_C$. Suitable parameter values were calculated from the initial experiments. The final parameter settings were measured to be N = 10, M = 100, $\alpha = 0.45$, $\omega_1 = \omega g = 0.35$, and $q_0 = 0.8$. The problem instances were generated randomly. The instances were a insist set of memory and CPU utilizations for 200 VMs. Every test case was repeated with 25 executions for each problem instance and the average grades over 25 independent executions are accounted. We showed five test cases with different parameters for various placement policies in Tab. 2. Based on worst scenario, the number of PMs was set to the number of VMs.

Only one VM is placed per PM in the worst scenario. After the MOACO algorithm was completed, if there were some non dominated solutions, that solution fit in to the set of non dominated solutions was randomly selected. When number of PM = 47, number of VM = 50 number of ants N = 10, number of iteration = 10, $\alpha = 0.45$, $\omega_1 = \omega g = 0.35$, $q_0 = 0.8$, and rel_paramter = 0.5, the number of paretset size is 3. When number of PM = 16, number of VM = 15 number of ants N = 10, number of iteration = 10, $\alpha = 0.45$, $\omega_1 = \omega g = 0.35$, $q_0 = 0.8$, and rel_paramter = 0.5, the number of paretset size is 9. When number of PM = 78, number of VM = 100, number of ants N = 10, number of iteration = 10, $\alpha = 0.45$, $\omega_1 = \omega g = 0.35$, $q_0 = 0.8$, and rel_paramter = 0.5, the number of paretset size is 9.

The power consumption of Least VM policy, MOACO, PAPHF, FFD and DFA-VMP is measured and showed (Fig. 3a). The power consumption of $Group_A$ and $Group_B$ is high while comparing to MOACO and DFA-VMP.

Table 2: Comparison of least VM, MOACO, PAPHF, FFD and DFA-VMP

Number of Testcase	PMs VMs Cloudlets	Number of malicious users	Number of Target users	Number of regular users	Policy	Number of active PMs	Co-resident probability	Power Consumption KW	Resource Wastage (%)
Testcase_1	10	1	1	10	Least VM	8	0.01	1525	31.2
	15				MOACO	6	0.03	1114	2.5
	20				PAPHF	8	0	1750	15.4
					FFD	7	0.02	1647	5.1
					DFA-VMP	6	0.01	1250	3.4
Testcase_2	20	3	2	15	Least VM	17	0.02	1625	29.2
	35				MOACO	14	0.03	1107	4.5
	60				PAPHF	18	0	1550	18.4
					FFD	15	0.04	1747	9.1
					DFA-VMP	15	0.04	1250	5.4
Testcase_3	50	5	4	50	Least VM	41	0.02	1825	41.2
	75				MOACO	35	0.04	1414	5.8
	100				PAPHF	44	0	1750	35.4
					FFD	40	0.06	1647	25.1
					DFA-VMP	37	0.04	1550	13.4
Testcase_4	80	10	5	100	Least VM	72	0.1	2125	52.2
	100				MOACO	65	0.5	1914	10.5
	150				PAPHF	77	0	2550	45.4
					FFD	70	0.8	2247	35.1
					DFA-VMP	68	0.6	2150	23.4
Testcase_5	100	15	10	200	Least VM	85	0.15	2675	55.2
	120				MOACO	70	0.8	2114	22.5
	180				PAPHF	90	0	2750	65.4
					FFD	75	0.89	2347	35.1
					DFA-VMP	73	0.8	2250	23.4

The VMs of the Un-trusted user are placed in the PM by PSPF placement policy. Based on the user resource requirement, the PM will be selected. The server utilization of $Group_C$ is very less. As we allocated the same PH (Physical Host) for the same user, the server utilization is lesser. Let 'N' be the number of VM created by Un-trusted users. Let 'M' be the number of PMs assigned by N. By using the PSPF placement policy, the VMs are allocated in the PM based on their PM resource utilization and previously selected PM takes the first opportunity. So, the number of PM is less than or equal to the number of the user ($M \leq N$). The power consumption and resource wastage of the PMs are measured in MOACO placement policy and compared with other placement policies (Figs. 3a and 3b). We experimented MOACO algorithm with different parameters for the same input to find suitable parameters for our problem. The number of active PMs also counted and compared for various placement policies (Fig. 3c). In DFA-VMP, the VMs are placed in PM by a discrete firefly algorithm [25]. They reduced power consumption and resource loss at the data centre. As we placed the VMs to PMs as per the user behaviour, the probability of placing malicious tenants and targeted tenants on the same physical host is very less compared to DFA-VMP. The co-resident probability is measured for the various placement policies. If the attacker creates more number of VMs, then the probability of co-resident also will be high. As the VM of the attacker is placed in $GROUP_C$ by the method of Previously Assigned Physical Host First, the co-resident is zero. But the power consumption and resource wastage are high in $GROUP_C$ (Fig. 4).

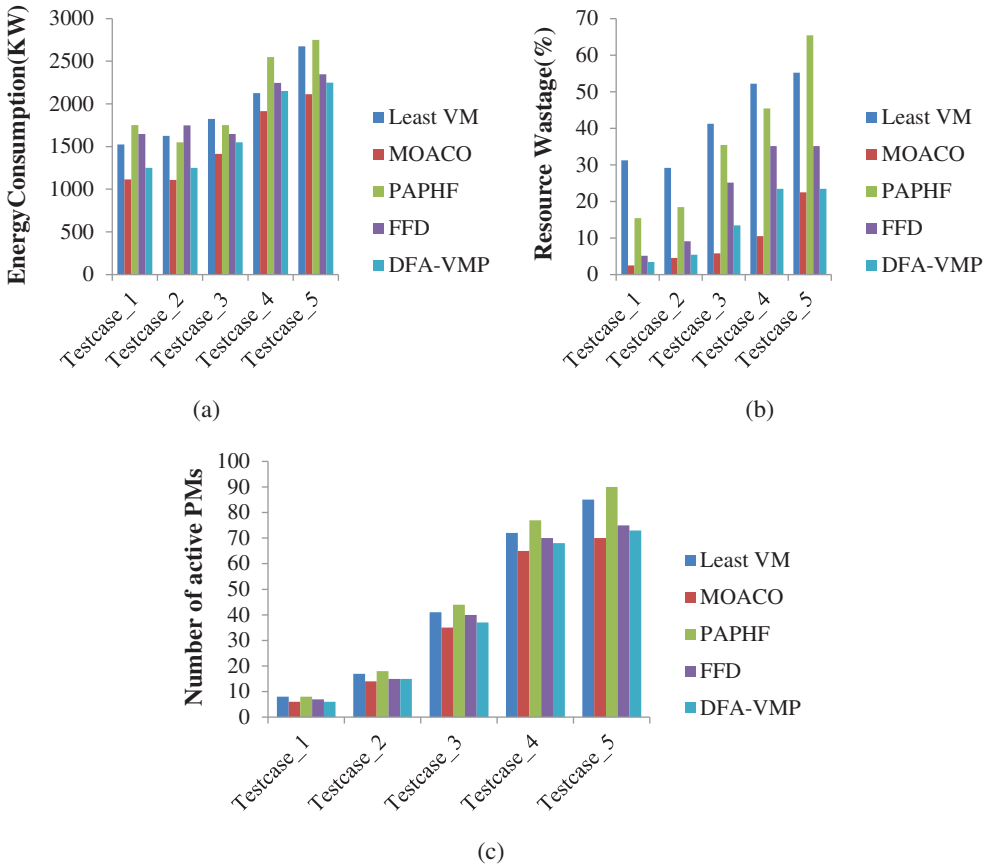

Figure 3: Comparison of Least VM, MOACO, PAPHF, FFD and DFA-VMP (a) Energy consumption (b) Resource wastage (c) Number of active VMs

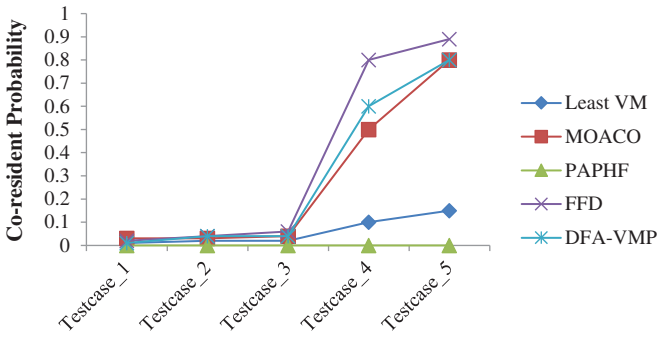

Figure 4: Comparison of Co-resident Probability for Least VM, MOACO, PAPHF, FFD and FA-VMP

7 Conclusion

This paper provides a new technique to reduce the Co-Resident of the attacker with the target user and also improves server utilization. Instead of finding a solution after co-locating the attackers with the target user, the cloud providers can mitigate the threat by minimizing the probability of attackers co-locating with the target. The existing solutions also give techniques for reducing the co-resident rate, but they have not considered the server utilization rate. Our proposed solution provides less co-resident and better server utilization. The frequent cloud CSCA attackers are identified and protected the secret information

of the victim. Even though the power consumption and resource wastage of the new users and un-trusted users is high, the secrete information of the target users is protected from the attacker.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding this study.

References

[1] Y. Han, T. Alpcan, J. Chan and C. Leckie, "Security games for virtual machine allocation in cloud computing, " in *Decision and Game Theory for Security*, *Lecture Notes In Computer Science Series*, vol. 8252, Springer International Publishing, pp. 99–118, 2013.

[2] Y. Han, J. Chan, T. Alpcan and C. Leckie, "Using virtual machine allocation policies to defend against co- resident attacks in cloud computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 14, no. 1, pp. 95–108, 2017.

[3] B. Vattikonda, S. Das and H. Shacham, "Eliminating fine grained timers in xen," in *Proc. of the Third ACM Workshop on Cloud Computing Security Workshop*, Chicago, Illinois, USA, pp. 41–46, 2011.

[4] J. Wu, L. Ding, Y. Lin, N. Min Allah and Y. Wang, "XenPump: A new method to mitigate timing channel in cloud computing," in *Proc. of the Fifth IEEE Int. Conf. on Cloud Computing*, Honolulu, HI, USA, pp. 678–685, 2012.

[5] A. Aviram, S. Hu, B. Ford and R. Gummadi, "Determinating timing channels in compute clouds," in *Proc. of the ACM Workshop on Cloud Computing Security Workshop*, Chicago, Illinois, USA, pp. 103–108, 2012.

[6] J. Wu, L. Ding, Y. Wang and W. Han, "Identification and evaluation of sharing memory covert timing channel in xen virtual machines," in *Proc. of the Fourth IEEE Int. Conf. on Cloud Computing*, Washington, DC, USA, pp. 283–291, 2011.

[7] G. Daniel, L. Valenta and Y. Yarom, "May the fourth be with you: A microarchitectural side channel attack on several real-world applications of curve25519," in *Proc. of ACM SIGSAC Conf. on Computer and Communications Security*, Dallas, Texas, USA, pp. 845–858, 2017.

[8] A. Hameed, A. Khoshkbarforoushha, R. Ranjan, P. Jayaraman, J. Kolodziej *et al.,* "A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems," *Computing*, vol. 98, no. 7, pp. 751–774, 2014.

[9] P. Graubner, "Energy-efficient management of virtual machines in eucalyptus," in *Proc. of the IEEE Fourth Int. Conf. on Cloud Computing*, Washington, DC, USA, pp. 243–250, 2011.

[10] R. Jansen and P. R. Brenner, "Energy efficient virtual machine allocation in the cloud: An analysis of cloud allocation policies," in *Proc. of the Int. Green Computing Conf. and Workshops*, Orlando, FL, USA, pp. 1–8, 2011.

[11] K. Le, R. Bianchini, J. Zhang, Y. Jaluria, J. Meng *et al.,* "Reducing electricity cost through virtual machine placement in high performance computing clouds," in *Proc. of the Int. Conf. for High Performance Computing, Networking, Storage and Analysis*, Seattle, WA, USA, pp. 1–12, 2011.

[12] K. Mills, J. Filliben and C. Dabrowski, "Comparing VM-placement algorithms for on-demand clouds," in *Proc. of the IEEE Third Int. Conf. on Cloud Computing Technology and Science*, United States, pp. 91–98, 2011.

[13] A. Beloglazov, J. Abawajy and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755–768, 2012.

[14] Y. Zhang, M. Li, K. Bai, M. Yu and W. Zhang, "Incentive compatible moving target defense against VM-colocation attacks in clouds," *IFIP Advances in Information and Communication Technology book series*, vol. 376, pp. 388–399, 2012.

[15] Y. Zhang and M. K. Reiter, "Düppel: Retrofitting commodity operating systems to mitigate cache side channels in the cloud," in *Proc. of the ACM SIGSAC Conf. on Computer & Communications Security*, Berlin, Germany, pp. 827–838, 2013.

[16] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers," *Concurrency Computation Practice Experience*, vol. 24, no. 13, pp. 1397–1420, 2012.

[17] T. Ristenpart, E. Tromer, H. Shacham and S. Savae, "Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds," in *Proc. of the ACM Conf. on Computer and Communications Security (CCS)*, Chicago, Illinois, USA, pp. 199–221, 2009.

[18] Y. C. Lee and A. Y. Zomaya, "Energy efficient utilization of resources in cloud computing systems," *Journal of Supercomputing, Springer*, vol. 60, no. 2, pp. 268–280, 2012.

[19] A. Beloglazov, J. Abawajy and R. Buyya, "'Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755–768, 2012.

[20] M. H. Malekloo, N. Kara and M. El Barachi, "An energy efficient and SLA compliant approach for resource allocation and consolidation in cloud computing environments," *Sustainable Computing: Informatics and Systems*, vol. 17, no. 1, pp. 9–24, 2018.

[21] M. H. Fathi and L. M. Khanli, "Consolidating VMs in green cloud computing using harmony search algorithm," in *Proc. of the Int. Conf. on Internet and e-Business (ICIEB)*, New York, NY, USA, pp. 146–151, 2018.

[22] M. Soltanshahi, R. Asemi and N. Shafiei, "Energy-aware virtual machines allocation by krill herd algorithm in cloud data centers," *An open access Journal, Elsevier*, Heliyon, cell press, vol. 5, no. 7, 2019.

[23] M. Mohammadhosseini, A. Toroghi Haghighat and E. Mahdipour, "An efficient energy-aware method for virtual machine placement in cloud data centers using the cultural algorithm," *Journal of Supercomputing*, vol. 75, no. 10, pp. 6904–6933, 2019.

[24] N. Garg, D. Singh and M. S. Goraya, "Power and resource-aware VM placement in cloud environment," in *Proc. of the IEEE 8th Int. Advance Computing Conf. (IACC)*, Greater Noida, India, pp. 113–118, 2018.

[25] W. Ding, C. Gu, F. Luo, Y. Chang, U. Rugwiro *et al.,* "DFA-VMP: An efficient and secure virtual machine placement strategy under cloud environment," *Peer-to-Peer Networking and Applications*, vol. 11, no. 2, pp. 318–333, 2018.

[26] G. Sangeetha and G. Sumathi, "An optimistic technique to detect cache based side channel attacks in cloud," *Peer-to-Peer Networking and Applications.* [Online]. 2020. https://doi.org/10.1007/ s12083-020-00996-1.

[27] J. M. Kaplan, W. Forrest and N. Kindler, Revolutionizing data center energy efficiency, *Technical Report.* . New York, NY, USA: McKinsey & Company, 2008.

[28] A. Beloglazov and R. Buyya, "Energy efficient resource management in virtualized cloud data centers," in *Proc. of 10th IEEE/ACM Int. Conf. on Cluster, Cloud and Grid Computing*, Melbourne, VIC, Australia, pp. 826–831, 2010.

[29] P. Graubner, "Energy-efficient management of virtual machines in eucalyptus," in *Proc. of the IEEE 4th Int. Conf. on Cloud Computing*, Washington, DC, USA, pp. 243–250, 2011.

[30] R. Jansen and P. R. Brenner, "Energy efficient virtual machine allocation in the cloud: An analysis of cloud allocation policies," in *Proc. of the Int. Green Computing Conf. and Workshops*, Orlando, FL, USA, pp. 1–8, 2011.

[31] K. Le, R. Bianchini, J. Zhang, Y. Jaluria, J. Meng *et al.,* "Reducing electricity cost through virtual machine placement in high performance computing clouds," in *Proc. of the Int. Conf. for High Performance Computing, Networking, Storage and Analysis*, Seattle, WA, USA, pp. 1–12, 2011.

[32] K. Mills, J. Filliben and C. Dabrowski, "Comparing VM-placement algorithms for on-demand clouds," in *Proc. of the IEEE Third Int. Conf. on Cloud Computing Technology and Science*, Athens, Greece, pp. 91–98, 2011.

[33] A. Beloglazov, J. Abawajy and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755–768, 2012.

[34] V. Maniezzo, "Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem," *INFORMS Journal on Computing*, vol. 11, no. 4, pp. 358–369, 1998.

[35] R. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.