



Smart elastic scheduling algorithm for virtual machine migration in cloud computing

Heba Nashaat¹ · Nesma Ashry¹ · Rawya Rizk¹ 

Published online: 11 January 2019

© Springer Science+Business Media, LLC, part of Springer Nature 2019

Abstract

Cloud Computing has the facility to transform a large part of information technology into services in which computer resources are virtualized and made available as a utility service. From here comes the importance of scheduling virtual resources to get the maximum utilization of physical resources. This paper presents two cooperative algorithms: a Smart Elastic Scheduling Algorithm (SESA) and an Adaptive Worst Fit Decreasing Virtual Machine Placement (AWFDVP) algorithm. The proposed algorithms work to dynamically distribute the cloud system's physical resources to obtain a load-balanced consolidated system with minimal used power, memory, and processing time. SESA arranges VMs in clusters based on their memory and CPU parameters' value. Then it deals with the colocated VMs that share some of their memory pages and located on the same physical machine, as a group. Then the migration decision is made based on the evaluation for the entire system by AWFDVP. This process minimizes the number of migrations among the system, saves the consumed power, and prevents performance degradation for the VM while preserving the load-balance state of the entire system. SESA reduces the power consumption in the cloud system by 28.1%, the number of migrations by 57.77%, and performance degradation by 57.1%.

Keywords Cloud computing · Colocated virtual machines · Live migration · Load balancing · Resource scheduling

1 Introduction

Cloud computing is a collection of integrated and networked hardware, software, and Internet infrastructure, called a platform. This platform hides the complexity and the details of the underlying infrastructure from users and applications by providing a simple graphical interface. In cloud systems, all hardware infrastructure

✉ Rawya Rizk
r.rizk@eng.psu.edu.eg

¹ Electrical Engineering Department, Port Said University, Port Said 42523, Egypt

elements are virtualized into virtual entities to deliver Infrastructure as a Service (IaaS) cloud model. Cloud operating system (OS), networking, and storage systems are also virtualized to deliver Platform as a Service (PaaS) cloud model. In addition, software programs, applications, and different OSs are implemented in a cloud to deliver Software as a Service (SaaS) cloud model [1].

Virtualization is considered the main enabling technology of cloud computing. It is a technique that allows running different OSs simultaneously on one physical machine (PM). These OSs are isolated from each other and from the underlying physical infrastructure by means of a special middleware abstraction called virtual machine (VM) as shown in Fig. 1. The piece of software that is responsible for creating, running, and managing these multiple VMs on PM or a pool of PMs is called hypervisor or VM kernel (VMkernel). It provides a mechanism for mapping VMs to physical resources transparently from the cloud users. So, VMkernel can be considered as a scheduler that manages VM access to the physical resources.

Scheduling the allocation of the physical and virtual resources is considered the most important challenge in virtual and cloud systems. Resource allocation process has two major levels: the physical resource allocation in the infrastructure

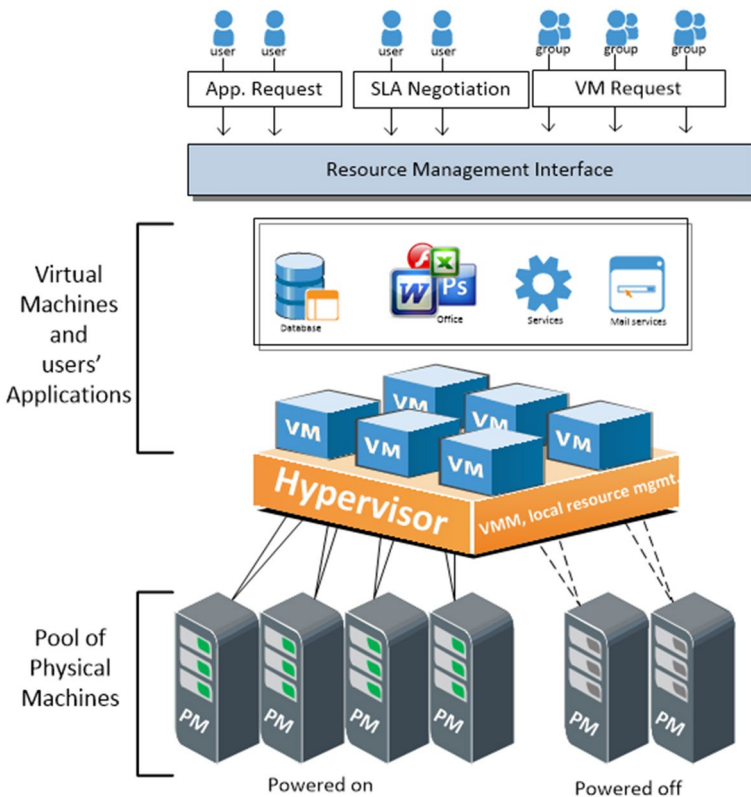


Fig. 1 Virtualization architecture

level and the virtual resource allocation in the task or application level [2, 3]. The resource allocation in infrastructure level mainly depends on VM placement whether for the first startup time of the system or by changing its place according to the available physical resources during the system running [4]. In the first startup time for the system, the VM is placed on a selected PM and gets its required resources from the PM's physical resources. In the second case, while there is a need to change the placement of VM from one PM to another, migration techniques are needed.

Live migration is a technique that allows VMs to move from one PM to another within cloud transparently while VMs are still running. It is necessary for server virtualization as it provides the virtual and cloud systems with some benefits [5, 6], which includes: (1) load balancing: It is accomplished by migrating VMs between PMs to balance the utilization of physical resources in the system and avoid overloaded PMs "hot-spot PMs" while the system is running. (2) Planned maintenance: In this case, PMs need hardware or software maintenance or update while important applications are running on VMs which are called production VMs. For high availability, these VMs must not be turned off during maintenance in order to avoid downtime of the system. (3) Consolidation: VMs on lightly loaded hosts can be packed onto fewer PMs with the consideration of meeting resource requirements and avoiding hot-spot PMs. Therefore, the freed-up PMs can be turned off for power saving or support high resource availability for new VMs.

VM placement has different perspectives in detecting where and when to migrate the VMs. Bin packing algorithm is a well-known method to allocate VMs [7]. Some heuristics studies proposed adaptive algorithms for modifying the bin packing algorithm such as First Fit Decreasing (FFD), Worst Fit Decreasing (WFD), and Best Fit Decreasing (BFD) algorithms. They mainly control the power consumption and detect hot-spot and underutilized PMs to perform balanced migrations. VMs' migrations from hot-spot and underutilized PMs to available PMs are accomplished with respect to service-level agreement (SLA). Then, Power Aware Best Fit Decreasing (PABFD) algorithm is introduced and tested for VM placement in [8]. It is a modified version of the BFD algorithm with some custom modifications that included the effect of CPU and memory utilization for VMs on the migration to load-balance the system. Some other studies focused on enhancing the amount of memory transferred during the migration by ganging the VMs to be migrated into groups based on their PM and the amount of memory shared between them. This technique is called Live Gang Migration (LGM) [9].

In this paper, two cooperative algorithms are proposed. A Smart Elastic Scheduling Algorithm (SESA) is presented as the main algorithm. SESA categorizes the migratable VMs into clusters and finds and arranges colocated VMs, which are multiple active VMs located on the same PM with some identical memory pages. An Adaptive Worst Fit Decreasing VM Placement (AWFDVP) algorithm does the VM placement process while preserving the load-balance state for the whole system. The primary objective of the proposed algorithms is to provide a solution for multiple active VMs migration that are sharing the same amount of memory at the same time for reducing the amount of memory migrated, and then reducing network traffic. The proposed algorithms dynamically adapt the VM live migration, concerning the

maximum number of migrations per PM that does not violate the SLA. In addition, they help in avoiding server sprawl to minimize energy consumption.

The rest of this paper is organized as follows: Sect. 2 presents related work on clustering techniques and scheduling live migration of VMs with the use of LGM. Section 3 introduces the proposed SESA and the associative AWFDP algorithm. Section 4 presents the implementation of the proposed algorithm with the results. Finally, the main conclusions and future work are summarized in Sect. 5.

2 Related work

Resource allocation in cloud computing has been studied from many different views [10]. As shown in Fig. 2, resource allocation could be at the application or task level, which means finding the ways to assign the available resources to the needed cloud applications or tasks running on servers over the Internet [11–15]. Resource allocation is also used to balance and scale up and scale down virtualized computer environment by automatically migrating VMs among the system's PMs [16–20].

Dynamic VM allocation challenge basically comes from the VM placement strategy. A number of approaches have been presented to solve the VM migration problems. Some of them are static resource scheduling algorithms such as round-robin scheduling [21], weighted round-robin scheduling [22], destination and source hashing scheduling [23, 24]. Others are dynamic approaches such as bin packing approach. Bin packing approximation algorithms are First Fit algorithm (FF), Best Fit algorithm (BF), First Fit Decreasing algorithm (FFD), and Best Fit Decreasing algorithm (BFD) [8].

In [25], a Modified Best Fit Decreasing (MBFD) algorithm is presented. It deals with the hot-spot as an item and the target PM as a packing bin with a scheduler that sorts hot-spot hosts in descending order. It weights CPU, memory, and network

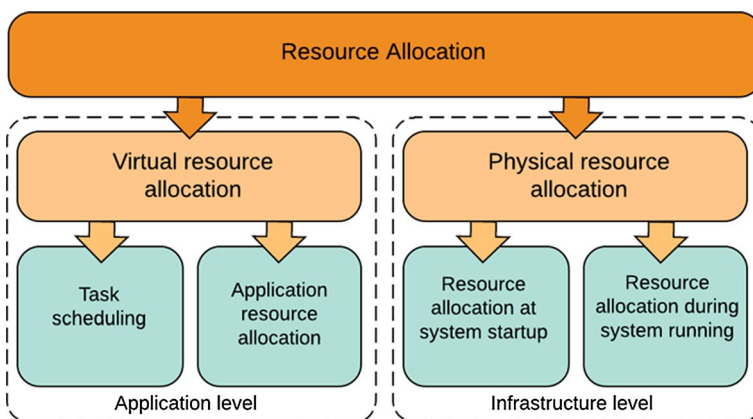


Fig. 2 Basic resource allocation hierarchy

bandwidth resources for all VMs in hot-spot PMs and sorts them in decreasing order. Then it traverses the non-hot-spot PM queue to find the most appropriate one as a migrating packing bin. It checks that the difference between the current load state of PM loading the VM and the maximum PM's threshold to avoid hot-spot is minimum. However, MBFD balances the system load; it neglects the power consumption for the system as it has no PM consolidation technique. This may lead to system sprawl.

In [8], PABFD algorithm is introduced, which is an enhanced power model based on BFD. It is a dynamic VM allocation system based on analysis of some VMs' resource usages. Strategies for power management are considered, such as dynamic VM allocation, determination of idle servers after dynamic allocation, and then switching idle servers to power-saving modes. In order to reach power saving mode, a large number of VM migrations is occurred. This large number of migrations causes performance degradation on the system and consumes the network bandwidth. The work presented in [26] modifies the VM placement and allocation algorithms to take the place of the PABFD algorithm. Some of the modified algorithms are Modified Worst Fit Decreasing VM Placement (MWFDDVP) and First Fit Decreasing with Decreasing Host VM Placement (FFDHDVP).

There are two approaches to the modification: non-cluster and cluster approaches. The clustering means grouping the objects with the same attribute values together. There are multiple cluster models such as connectivity model, centroid-based models, distribution models, density models, and graph-based model [27]. Centroid-based model is one of the best cluster models presented. The most popular centroid-based clustering algorithm is k -means algorithm. It defines a specific number of clusters K and represents them by their centroids where each centroid represents the center of its cluster. It uses an efficient way to find the number of clusters K in terms of PMs and VMs parameters from CPU and memory. Then it calculates the value of initial centroids. The common way to select the centroids is random. This leads to multiple iterations till finding the correct centroids. A way to choose initial and final centroids with single iteration was introduced in [28]. It minimizes the execution time of the algorithm. However, the clustering technique reduces the number of VM migrations a little. It does not reduce the amount of transferred data between PMs. Therefore, there is a need for a method to reduce the data transferred with the migration process.

VMs which are located on the same PM and share some memory pages are called the colocated VMs. The shared memory content is needed to be transferred only once during the migration if those colocated VMs migrated together from one PM to another. This migration is an LGM [9]. This technique optimizes memory and network bandwidth usage while migration and reduces the number of migration processes needed. LGM could be used to achieve a fast load balancing to deal with a sudden hot-spot to meet the SLA. Also, LGM is useful in consolidation cases to save energy, and mainly in cases of planned Information Technology (IT) maintenance where an entire server or rack of servers need replacement of hardware and software upgrade, so they are needed to be shut down. However, it is not easier to simultaneously migrate a large number of VMs due to limited bandwidth. Then, it needs an intelligent way to regulate the number of migrations.

Dynamic VM consolidation mechanism, called PCM, was developed and tested in [29]. It relies on modifications for the four basic steps of VM allocation presented in [26]. It starts with overloading PM detection, which distinguishes when PMs should be considered overloaded, then one or several VMs are reallocated to other PMs to reduce PM utilization. After it proposes the underloading PM detection that distinguishes when PMs should be considered underloaded, then all the VMs are consolidated to other PMs, and the PM is switched to the sleep mode. Next, for the VM selection, the most suitable VMs are chosen to be migrated from overloaded PMs. Finally, the VM placement discovers the most suitable available PM for the selected VMs. However, PCM increases complexity since it works in four levels. In addition, it doesn't take clustering into considerations.

As mentioned above, the algorithms presented in [26] are enhanced VM placement algorithms with a clustering technique. However, the amount of transferred data and the number of migrations are not preserved. Also, the system may change the load-balance state after migration. In addition, an LGM technique reduces the transferred data by identifying the identical shared memory between colocated VMs to be transferred once while those VMs are migrating, but their work does not include an algorithm to schedule the VM placement and migration process. In this paper, the method of clustering and colocating VMs while preserving the load-balance state of the system is proposed in two cooperative algorithms: SESA and AWFDVP. SESA presents a clustering colocation mechanism and AWFDVP is concerned with preserving the load-balance state of the system while doing the migration process by using the system standard deviation (STD) [30]. AWFDVP determines the best VMs' placement decision while minimizing the migration and utilization parameters. The proposed work also mitigates hot-spot PMs and prevents server sprawl to reduce power consumption and consolidate cloud system.

3 The proposed SESA

Infrastructure resource allocation techniques follow four basic steps to perform VM allocation as shown in Fig. 3. These steps are hot-spot PMs detection, Selection of Migratable VMs, listing the available PMs, and VM placement [31]. Since VM placement is one of the key challenges in this process, smart and adaptive algorithms: SESA and AWFDVP, are proposed. SESA works to minimize the number of multiple colocated VMs migrations by grouping the migratable VMs with VM clustering and LGM techniques. As the LGM increases the chance of creating hot-spot PM(s), AWFDVP algorithm is proposed for VM placement. AWFDVP relies on using the WFD algorithm, one of the algorithms used to solve the bin packing problem. AWFDVP does the actual placement for migratable VMs while mitigating hot-spot PM(s) and consequently preserving the load-balance state of the system by a STD check. This STD check is run periodically in the system in a defined time slot (DTS), and if the STD check yields unbalanced or unconsolidated system, the two cooperative algorithms are executed.

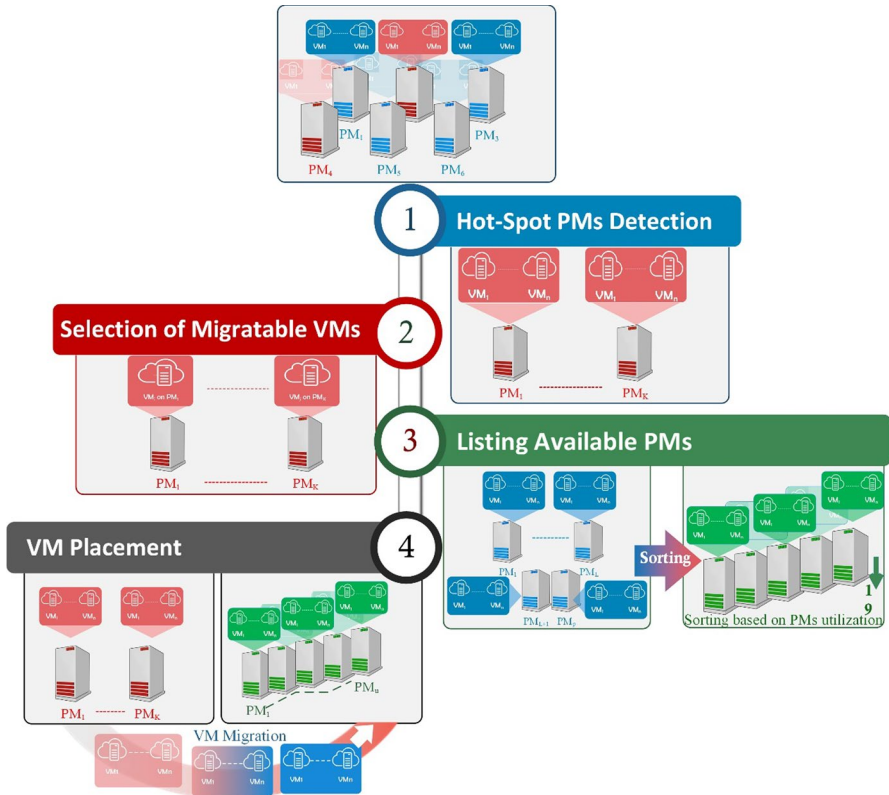


Fig. 3 VM allocation steps

3.1 VM allocation

The proposed VM allocation process depends on the first three basic steps of the resource allocation: hot-spot PMs detection, selection of VMs to be migrated, and listing the available PMs to put them as migration destinations. In addition, the fourth step is enhanced for choosing in which PM, the migratable colocated VMs will be allocated by taking care of preventing hot-spot and unload balanced state for the system. In this section, the proposed enhancement algorithms in VM allocation process: SESA and AWFDP, are presented. VM allocation algorithms are implemented and tested on CloudSim toolkit[8].

Step 1 Hot-spot PMs detection

Threshold (THR) is a common algorithm to define the hot-spot in a static way. Auto-adjustment algorithms are proposed in [8]. They utilize hot-spot detection threshold based on statistical analysis of previous data gathered during the lifetime of VMs. In addition to THR, there are four adaptive hot-spot PM detection

techniques: median absolute deviation (MAD), interquartile range (IQR), local regression (LR), and robust local regression (LRR) [8]. Once hot-spot PMs have been detected, it is essential to decide which VMs must be migrated. VM selection algorithms solve this problem.

Step II Selection of Migratable VMs

This step determines which VMs will be migrated. There are four VM selection policies: Maximum Correlation Policy (MC), Minimum Migration Time (MMT), Minimum Utilization (MU), and random selection (RS) [26, 32].

Table 1 lists all combinations of five algorithms of hot-spot PMs detection with four algorithms of VM selection, which forms twenty main algorithms to be utilized by VM placement modification algorithms illustrated in Step IV.

Step III Listing the available PMs

After excluding the hot-spot PMs, the remaining PMs are tested to exclude the underutilized ones to get all their VMs for migration and put them to sleep. Then remaining PMs, which are called available PMs, are sorted increasingly according to their utilization to be used as destination PMs for the migration process. Next, it is necessary to determine in which destination PMs the VMs will be migrated as in VM placement process.

Step IV VM placement

VM placement is to determine in which available PMs the VMs will be migrated. The proposed SESA is developed by modifying the basic *K*-means algorithm for clustering VMs with the use of LGM technique that takes into consideration the colocated VMs. SESA is calling the proposed AWFDP algorithm, which adaptively allocates the migratable VMs into PMs with a load-balance check for the system in each migration process. The proposed work is presented to automatically scale the cloud resources allocating while considering the system application SLAs, number of migrations, and power consumption. The operation of SESA begins with collecting migratable VMs into clusters by using a modified *k*-means method based on two parameters: the CPU utilization and currently allocated memory, and then grouping the colocated VMs in each cluster. After this, SESA arranges the clusters

Table 1 Main combinations of hot-spot PMs detection and VM selection algorithms

| VM selection | IQR | LR | LRR | MAD | THR |
|-----------------------|---------|--------|---------|---------|---------|
| Hot-spot PM detection | | | | | |
| MC | Iqr_mc | Ir_mc | Irr_mc | mad_mc | thr_mc |
| MMT | Iqr_mmt | Ir_mmt | Irr_mmt | mad_mmt | thr_mmt |
| MU | Iqr_mu | Ir_mu | Irr_mu | mad_mu | thr_mu |
| RS | Iqr_rs | Ir_rs | Irr_rs | mad_rs | thr_rs |

| <i>Smart Elastic Scheduling Algorithm (SESA)</i> | |
|--|---|
| 1. | <i>Input:</i> hostList, VMList, Standard Deviation threshold, <i>Output:</i> high density arranged cluster list of co-located VMs, allocation of VMs Find K points for selecting the optimal number of clusters considering two parameters (CPU, RAM) Calculating K_1 (for Parameter-> CPU) |
| 2. | $K_1_maxpoint = \text{hostlist.get_max(CPU)} / \text{VMList.get_min(CPU)}$ |
| 3. | $K_1_minpoint = \text{hostlist.get_min(CPU)} / \text{VMList.get_max(CPU)}$ |
| 4. | $K_1 = \text{average}(K_1_maxpoint, K_1_minpoint)$ Calculating K_2 (for Parameter-> RAM) |
| 5. | $K_2_maxpoint = \text{hostlist.get_max(RAM)} / \text{VMList.get_min(RAM)}$ |
| 6. | $K_2_minpoint = \text{hostlist.get_min(RAM)} / \text{VMList.get_max(RAM)}$ |
| 7. | $K_2 = \text{average}(K_2_maxpoint, K_2_minpoint)$ |
| 8. | $K = \text{average}(K_1, K_2)$ Select the initial centroid as a pair of two values (CPU, RAM) |
| 9. | centroids[1, 1] = Get_average_CPU(VMList) |
| 10. | centroids[1, 2] = Get_average_RAM(VMList) Find the remaining $K-1$ centroids, |
| 11. | for each m^{th} centroid number do, Where m takes values from 1 to $K-1$ Calculate the Euclidian distance (ED) between previous centroid and (CPU, RAM) parameters of each VM in VMList |
| 12. | for each j^{th} VM in VMList do, Where j takes values from 1 to no. of VMs in VMList |
| 13. | Ecu_dis[j] = find_Eucedian(VMList.get(j),centroids[m,1], centroids[m,2]) |
| 14. | end for Choose the Next Centroid to be (CPU, RAM) values for VM with maximum ED |
| 15. | centroids[m+1, 1] = VMList.get(get_index_forMaxValue(Ecu_dis)).get(CPU) |
| 16. | centroids[m+1, 2] = VMList.get(get_index_forMaxValue(Ecu_dis)).get(RAM) |
| 17. | end for Calculate the ED between each VMs all Clusters' centroids |
| 18. | for each j^{th} VM in VMList do, Where j takes values from 1 to no. of VMs in VMList |
| 19. | for each m^{th} centroid number do, Where m takes values from 1 to $K-1$ |
| 20. | ED[m] = find_Eucedian(VMList.get(j),centroids[m,1], centroids[m,2]) |
| 21. | end for Append VM to the Cluster with minimum ED |
| 22. | Cluster=Append_in_Cluster(get_index_forMinValue(ED), VMList.get(j)) |
| 23. | end for |
| 24. | Arrange the co-located VMs |
| 25. | for each i^{th} VMs' cluster list in Cluster do |
| 26. | arrangeBy Co-locatedVMs(Cluster.get(i)) |
| 27. | end for |
| 28. | VMList = arrangeBy HighDensityCluster(Cluster) |
| 29. | AWFDVP(hostlist,VMList, STDThr) |

Fig. 4 Pseudo-code of SESA

by their density in decreasing order. The cluster which has a lot of VMs will have the priority to be migrated first. The pseudo-code of SESA is shown in Fig. 4. Next, AWFDVP is called by SESA. It determines the best migration decisions for VMs to

```

Adaptive Worst Fit Decreasing VM placement (AWFDVP)
1. Input: hostList, VMList, Standard Deviation threshold, Output: allocation of VMs
   Initialize the variables for the Target and Destination Hosts for the last migrated VM
2. HostoldT= null
3. HostoldD= null
4. for each VM in VMList do
5.   maxPower = Double.MIN_VALUE
6.   allocatedHost = null
7.   for each host in hostList do
8.     Check if next VM has the same host for the last migrated VM
9.     if HostoldT!=gethost(VM)
10.      if host has enough resources for VM then
11.        powerAfterAllocation = getPowerAfterAllocation(host,VM)
12.        powerDiff = powerAfterAllocation -host.getPower()
13.        if powerDiff > maxPower then
14.          maxPower = powerDiff
15.          allocatedHost = host
16.          HostoldT=gethost(VM)
17.          HostoldD=host
18.        end if
19.      end if
20.    else
21.      Try to use the same host that last VM was allocated to, for the next VM
22.      host= HostoldD
23.      if host has enough resources for VM then
24.        powerAfterAllocation = getPowerAfterAllocation(host,VM)
25.        powerDiff = powerAfterAllocation -host.getPower()
26.        if powerDiff > maxPower then
27.          maxPower = powerDiff
28.          allocatedHost = host
29.          break
30.        else HostoldT==null
31.        end if
32.      end if
33.    end if
34.  end for
35.  if allocatedHost ≠ null then
36.    Allocation.add(VM,allocatedHost)
37.    for each host in hostList do
38.      Calculate all hosts volumes and add them to a list
39.      PMV= host_volume( host.getram(), host.getTotlaMips() )
40.      PMVList.add(PMV)
41.    end for
42.    Check the system load balance state by using standard deviation
43.    if STD (PMVList) < STDTHR then
44.      continue
45.    else
46.      Allocation.add(VM,Hostold)
47.      decrement VM counter by 1
48.    end if
49.  end if
50. end for
51. Return allocation

```

Fig. 5 AWFDVP load-balance VM placement algorithm

PMs by simulating the VM placement from the hot-spot PM to the available PMs except for the underutilized PMs by using an adaptive version of WFD algorithm while considering colocated VMs. AWFDVP is used to complete the colocated VM migrations, but in a condition to save the load-balance state of the system by using an STD check as illustrated in [30]. AWFDVP algorithm is presented in Fig. 5. The illustration and equations of SESA and clustering process are presented in Sect. 3.2 with a real data example to clarify the idea of how it works.

3.2 VM clustering process

As mentioned above, SESA uses modified K -means algorithm to divide the migratable VMs into clusters based on the CPU utilization and currently allocated memory. Then it uses an LGM method called “arrangeByCo-locatedVMs” to group each cluster VM into colocated VMs based on their PM. The existing MK algorithm could be illustrated in three phases: Find the number of clusters (K), calculate the K centroids, and distribute the migratable VMs according to the nearest centroid. This results in clusters which are sorted by its high density of VMs in decreasing order. In addition, SESA groups VMs in each cluster into colocated groups according to their PMs in order to load-balance and consolidate the cloud system. This operation is added as Phase 4 in the proposed algorithm. The phases of the proposed algorithm are presented as follows:

Phase 1 Find the number of clusters (K):

In order to find the number of clusters (K), two parameters are used: CPU utilization and memory allocation. It can be calculated as follows:

$$K = \frac{K_1 + K_2}{2} \quad (1)$$

where K_1 and K_2 are the number of clusters regarding CPU utilization and memory allocation, respectively. K_1 and K_2 can be calculated as:

$$K_i = \frac{\text{maxpoint} + \text{minpoint}}{2} \quad (2)$$

where maxpoint and minpoint can be formulated in Eqs. (3) and (4):

$$\text{maxpoint} = \frac{\alpha}{\sigma} \quad (3)$$

$$\text{minpoint} = \frac{\beta}{\gamma} \quad (4)$$

where α and β are the maximum and minimum available CPU MIPS among all PMs. γ and σ are the maximum and minimum currently allocated CPU MIPS among all VMs.

Phase 2 Calculate the cluster's centroids:

The initial centroid for the first cluster is usually selected randomly which is not precise. This initial centroid is the base for the other centroids' calculations. A more accurate method than random selection has been presented in [28]. This is done by taking average of all VMs' parameters; CPU utilization and memory allocation in order to calculate the initial centroid. It will be $\text{Cent}_1(\text{CPU-avg}, \text{mem-avg})$. The remaining centroids will be calculated as follows:

- (a) Calculating the Euclidean distance (ED) between all VMs' parameters and the previous centroid as follows:

$$\text{ED} = \sqrt{\sum_j^n (\text{VM}r_j - \text{cent}_g(j))^2}. \quad (5)$$

where n is the number of parameters for VMs that are considered in calculations. It is equal to 2 for VM with CPU and memory. $\text{VM}r_j$ is the value of each parameter, g is the cluster number, and cent_g is the centroid for a given cluster g .

- (b) Selecting the VM parameters with the maximum ED to be the next centroid.
 (c) Repeating steps (a) and (b) till all centroids are determined.

Phase 3 Distribute the migratable VMs according to the nearest centroid:

After finding the K centroids for the K clusters, all migratable VMs are distributed to the nearest cluster by choosing the minimum ED between the VM parameters' value and cluster's centroid as in Eq. (5).

Phase 4 Group VMs in each cluster into colocated groups:

The proposed SESA appends grouping phase to the basic K -means steps. Grouping VMs in each cluster into colocated groups depending on their PM is an essential phase to reduce the amount of data transferred during the VM placement by migrating these multiple colocated VMs together. These colocated VMs are VMs that are located on the same PM and sharing some amount of memory pages. Migrating the colocated VMs together saves network bandwidth and reduces the number of VMs' migrations. Therefore, each VM list in each cluster is grouped according to their PM and then the colocated VMs are migrated using AWFDP algorithm.

To further illustrate the idea of clustering and grouping of the colocated VMs, a real data example is presented in Fig. 6. This figure presents an example with real data excluded from the CloudSim running results. There are four PMs with two parameters: $\text{PM}_i(\text{CPU}(i)$ in MIPS, $\text{RAM}(i)$ in M bytes), as $\text{PM}_1(1510, 3500)$, PM_2

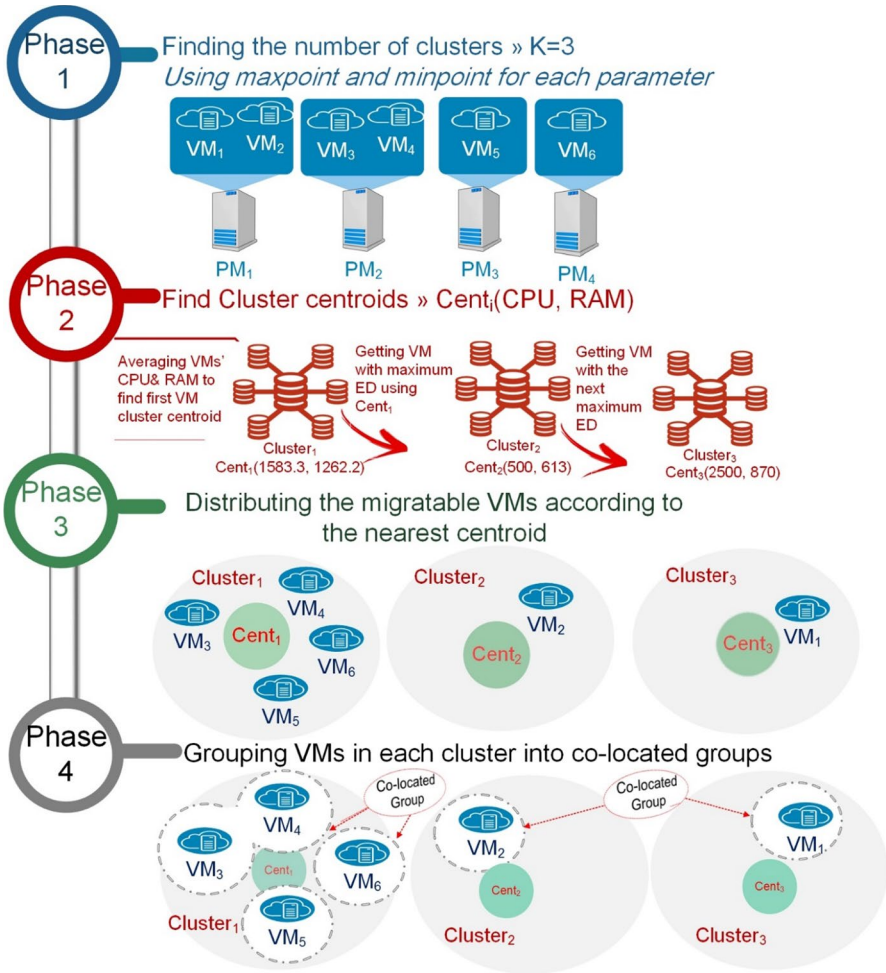


Fig. 6 SESA implementation on real VM data

(1800, 3000), PM_3 (1960, 3220), and PM_4 (1495, 3410). Each PM has one or more VMs with CPU and RAM parameters: VM_i (CPU(j) in MIPS, RAM(j) in M bytes). PM_1 has VM_1 (870, 2500) and VM_2 (612, 500). PM_2 has VM_3 (1740, 1000) and VM_4 (1740, 2000). PM_3 and PM_4 have only one VM as VM_5 (1740, 2000) and VM_6 (870, 1500), respectively. This example illustrates the proposed algorithm with the four phases as follows:

Phase 1 As previously mentioned, there are two parameters: CPU and RAM allocation. Therefore, there are K_1 and K_2 . K_1 can be calculated by finding:

- Maximum (α) and minimum (β) available CPU MIPS among all PMs. They are evaluated as $\alpha = 1960$ and $\beta = 1495$.
- Maximum (γ) and minimum (σ) currently allocated CPU MIPS among all VMs. They are computed as $\gamma = 1740$ and $\sigma = 613$.

From Eqs. (3) and (4), $\text{maxpoint} = 3.2$ and $\text{minpoint} = 0.86$. By averaging those values using Eq. (1), K_1 is computed as 2.

In the same manner, K_2 can be calculated by computing:

- Maximum (α) and minimum (β) available RAM among all PMs, which are $\alpha = 3500$ and $\beta = 3000$.
- Maximum (γ) and minimum (σ) currently allocated RAM among all VMs, which are evaluated as $\gamma = 2500$ and $\sigma = 500$.

From Eqs. (3) and (4), $\text{maxpoint} = 7$ and $\text{minpoint} = 1.2$. Using Eq. (1), K_2 is computed as 4. By applying Eq. (2), the number of clusters (K) is defined to be 3.

Phase 2 The centroids for three clusters are calculated as follows:

- Calculating initial centroid (Cent_1) for Cluster₁ by averaging all VMs' CPU RAM values. Therefore, initial centroid is set to be $\text{Cent}_1(1583.3, 1262.2)$.
- Determining the second centroid (Cent_2) for Cluster₂ by finding the ED between all VMs and Cent_1 using Eq. (5) as in Fig. 7. Then choosing the maximum ED from all VMs. Therefore, VM_2 is set as $\text{Cent}_2(500, 613)$.
- Calculating the third centroid (Cent_3) by determining the ED between all VMs and Cent_2 as in Fig. 7. The maximum ED is between VM_1 and Cent_2 . Then, VM_1 is considered as $\text{Cent}_3(2500, 870)$.

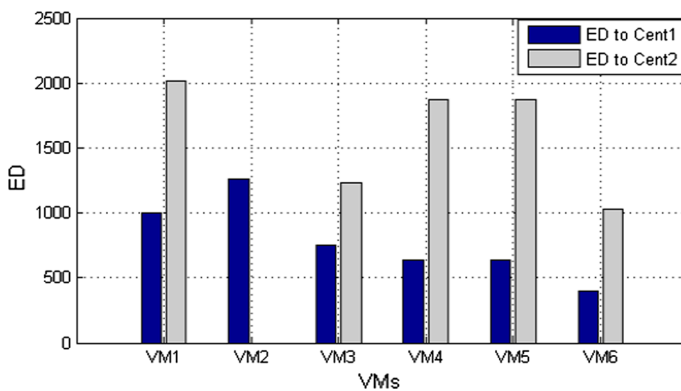


Fig. 7 ED between VMs and previous centroid to choose the next centroid

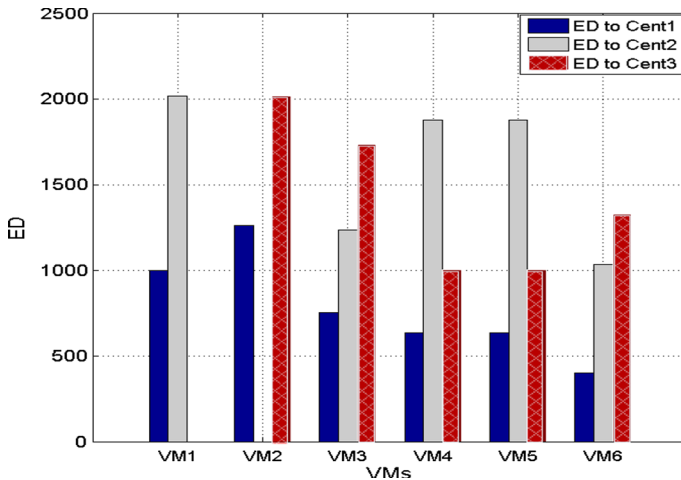


Fig. 8 ED between VMs and all centroids

Phase 3 Each VM is distributed to the nearest cluster by calculating the ED between each VM and each cluster's centroid. Then, the minimum values for VMs are chosen. Figure 8 shows ED between VMs and all centroids. For example, the most suitable cluster for VM₁ is Cluster₃. Therefore, Cluster₁ includes {VM₃, VM₄, VM₅, and VM₆}, Cluster₂ includes {VM₂}, and Cluster₃ includes {VM₁}.

Phase 4 The VMs are grouped according to their PM within each cluster as in the Cluster₁: VM₃ and VM₄ are grouped together since they are located on PM₂, while each of the other two VMs has no colocated VMs in the same cluster.

After completing the four phases of SESA, the AWFDP algorithm is called to adapt the VM allocations while considering the load-balance state of the system and avoiding creating hot-spot PM(s).

4 Implementation and results

In this section, the environment and the parameters of the simulated cloud system using CloudSim toolkit are introduced. Then the simulation results are presented.

4.1 Simulation environment

The architecture of the system is presented in Table 2. It follows the parameters of [26]. The system is implemented using CloudSim toolkit. These parameters are configured to test the performance of non-clustered and clustered approaches in [26] against the proposed cooperative algorithms SESA and AWFDP. For non-clustered and clustered approaches, three VM placement algorithms: PABFD, MWFDP, and FFDHDP, are used in comparison with SESA and its associative AWFDP algorithm. All these algorithms are combined with the twenty combinational algorithms

Table 2 Simulator parameters

| Type | Parameters | Values |
|----------------------|-------------------------------------|---|
| PMs | Number of PMs | 800 |
| | Number of PMs in categories | 400 of PM category 1 and 400 of PM category 2 |
| PM category 1 | Model | HP ProLiant ML110 G4 |
| | Processor | Intel Xeon 3040 |
| | Available memory per PM | 4 GB |
| | Number of processor cores per PM | 2 cores \times 1860 MHz |
| PM category 2 | Model | HP ProLiant ML110 G5 |
| | Processor | Intel Xeon 3075 |
| | Available memory per PM | 4 GB |
| | Number of processor cores per PM | 2 cores \times 2660 MHz |
| Storage | VM files' Datastore and users' data | Network-attached storage (NAS) |
| Data center | Number of data center sites | 500 around the world |
| Detection algorithms | Threshold for IQR | 1.5 |
| | Threshold for LR | 1.2 |
| | Threshold for LRR | 1.2 |
| | Threshold for MAD | 0.8 |
| | Threshold for THR | 2.5 |

Table 3 Trace-based PlanetLab workload days

| Workload | No. of VMs | Workload | No. of VMs |
|----------|------------|----------|------------|
| 20110303 | 1052 | 20110403 | 1463 |
| 20110306 | 898 | 20110409 | 1358 |
| 20110309 | 1061 | 20110411 | 1233 |
| 20110322 | 1519 | 20110412 | 1054 |
| 20110325 | 1078 | 20110420 | 1033 |

listed in Table 1. As mentioned above, the twenty combinational algorithms are formed from combining different hot-spot PM detection algorithms with VM selection algorithms. The test was run for a variant number of days, depending on the comparative analysis from PlanetLab online workload [33]. PlanetLab workload is a set of CPU utilization traces from PlanetLab VMs collected during 10 random days in March and April 2011, see Table 3.

The work starts by running the STD check for all PMs in the system. STD check is run every DTS equal to 5 min during the simulation. Running the check more frequent causes unnecessary overhead and increases the violation of more SLA. In addition, a DTS of less than 5 min might be too aggressive given that the cooperative algorithms: SESA and AWFDP, take on the order of 3 to 5 min to complete, depending on the number of migrations.

4.2 Simulation results

To compare the performance of all the presented algorithms, performance metrics such as power consumption (PC), number of VM migrations (NVM), performance degradation due to SLA violation (PD), SLA violation (SLAV) were chosen [34]. Then two comparative analyses were performed as follows:

(a) Comparative analysis with clustered approaches

As mentioned before, the proposed SESA is a clustered approach technique. So, SESA and the clustered algorithms in [26] were run in the CloudSim to test the effectiveness of SESA against other clustering algorithms. The proposed SESA and the cooperative AWFDP algorithm were run against the clustered algorithms: PABFD, MWFDVP, and FFDHDVP, with three different days of PlanetLab workload; “20110303,” “20110306,” and “20110309,” see Table 3. The percentage values in Table 4 present the improvement percentage for the proposed SESA against the other clustered algorithms with all twenty combinational algorithms shown in Table 1.

From Table 4, it is obvious that the cooperative algorithms SESA and AWFDP give a high improvement with all clustered VM placement algorithms reaching 60% in some cases and metrics. The four performance metrics are improved by using cooperative algorithms with almost all the twenty combinational algorithms. This is because the proposed algorithms dynamically distribute the physical resources to obtain a load-balanced system with minimal used power, memory, and processing time by arranging VMs in clusters based on their memory and CPU parameters and grouping the colocated VMs. Then, the migration decision is made. This process minimizes the number of migrations among the system, saves the consumed power, and prevents performance degradation for the VM while preserving the load-balance state of the entire system.

Figures 9, 10, 11 and 12 show a comparison of the four selected performance metrics, for non-clustered and clustered MWFDVP algorithm, since it gives the best results against SESA as shown in Table 4.

As shown in Fig. 9, SESA gives the minimum PC values with all combinational algorithms. The highest enhancement is shown in the case of using “mad_mc” combination algorithm as it reduces the PC by 8% compared with non-clustered MWFDVP and by 23% compared with clustered MWFDVP. Figure 10 shows variations in the number of migrations among the combinational algorithms. It is shown that SESA gives better results over the other two algorithms. SESA gives the lowest NVM when it is used with both “lr” and “lrr” combinational algorithms. The NVM reduction reaches 47% in clustered approach with “lr_mc” and “lrr_mc” algorithms. Figure 11 presents the performance degradation in VM applications over the cloud. It follows the behavior shown in Fig. 10

Table 4 Improvement percentages for the proposed SESA over the clustered algorithms

| | SESA improvement over PABFD | | | | SESA improvement over MWFDVP | | | | SESA improvement over FFDHDVP | | | |
|---------|-----------------------------|---------|--------|----------|------------------------------|---------|--------|----------|-------------------------------|---------|--------|----------|
| | PC (%) | NVM (%) | PD (%) | SLAV (%) | PC (%) | NVM (%) | PD (%) | SLAV (%) | PC (%) | NVM (%) | PD (%) | SLAV (%) |
| Iqr_mc | 22.4 | 26.3 | 30 | 17.1 | 20.8 | 45.8 | 50 | 0.2 | 26.8 | 57.5 | 60 | 20.7 |
| Iqr_mmt | 14.8 | 5.3 | 16.7 | 25 | 18.7 | 15.1 | 28.6 | 28.2 | 22.7 | 33 | 57.1 | 45.1 |
| Iqr_mu | 13.6 | 4.2 | 16.7 | 26.7 | 14.1 | 25.2 | 25 | 33.3 | 22.7 | 39.7 | 57.1 | 53.9 |
| Iqr_rs | 15.4 | 6.7 | 30 | 31.6 | 18 | 20.8 | 30 | 36.1 | 24.2 | 43.9 | 56.3 | 48.4 |
| Ir_mc | 18 | 57.7 | 60 | 44.4 | 21.1 | 47.8 | 57.1 | 3.8 | 28.1 | 50.8 | 62.5 | 2.9 |
| Ir_mmt | 14.4 | 22.4 | 37.5 | 34.7 | 15.5 | 5.9 | 50 | 0.2 | 25.5 | 17.3 | 40 | 9.1 |
| Ir_mu | 13 | 11.3 | 37.5 | 4.2 | 16.1 | 3 | 33.3 | 10 | 26.6 | 9.3 | 50 | 17.2 |
| Ir_rs | 11.5 | 30.7 | 50 | 38.2 | 17.8 | 20.1 | 50 | 19.2 | 25.1 | 30.7 | 42.9 | 10 |
| Irr_mc | 18 | 57.7 | 60 | 44.4 | 21.1 | 47.8 | 57.1 | 3.8 | 28.1 | 50.8 | 62.5 | 2.9 |
| Irr_mmt | 14.4 | 22.4 | 37.5 | 34.7 | 15.5 | 5.9 | 50 | 0.2 | 25.5 | 17.3 | 40 | 9.1 |
| Irr_mu | 13 | 11.3 | 37.5 | 4.2 | 16.1 | 3 | 33.3 | 10 | 26.6 | 9.3 | 50 | 17.2 |
| Irr_rs | 11.3 | 29.5 | 33.3 | 35.8 | 16.5 | 15.9 | 40 | 0.2 | 24.1 | 33.2 | 33.3 | 13.3 |
| mad_mc | 23.2 | 31.6 | 36.4 | 20 | 23.1 | 44.4 | 54.5 | 4.9 | 25.7 | 55 | 60 | 6.7 |
| mad_mmt | 14.8 | 4.5 | 16.7 | 18.8 | 18.1 | 8.6 | 37.5 | 16.3 | 22.4 | 29.7 | 57.1 | 38.8 |
| mad_mu | 14 | 5.1 | 16.7 | 26.7 | 14.5 | 20.7 | 12.5 | 24.2 | 22.3 | 39.8 | 57.1 | 41.8 |
| mad_rs | 16 | 8.7 | 30 | 28.2 | 19.1 | 25.1 | 30 | 24.7 | 6.9 | 45.1 | 58.8 | 39 |
| thr_mc | 22.9 | 25.3 | 30 | 21.6 | 21.5 | 45.1 | 50 | 25.3 | 0.4 | 46.5 | 53.3 | 18.1 |
| thr_mmt | 15.1 | 2.3 | 28.6 | 26.5 | 19.4 | 11.4 | 37.5 | 38.1 | 1.2 | 25.3 | 46.2 | 40.6 |
| thr_mu | 13.9 | 3 | 16.7 | 21.7 | 14.8 | 18.7 | 25 | 43.8 | 1.5 | 29.2 | 50 | 51.2 |
| thr_rs | 15.4 | 15.3 | 30 | 32.4 | 18.8 | 39.6 | 40 | 47.9 | 1.4 | 34.4 | 50 | 34.5 |

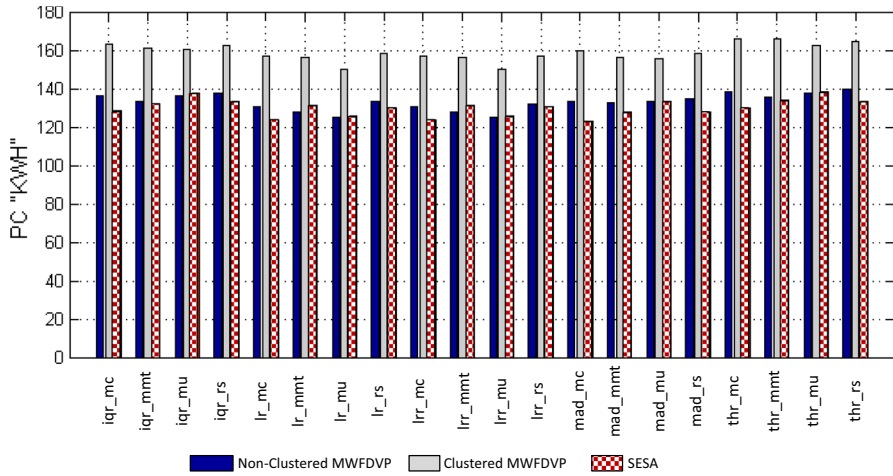


Fig. 9 Power consumption

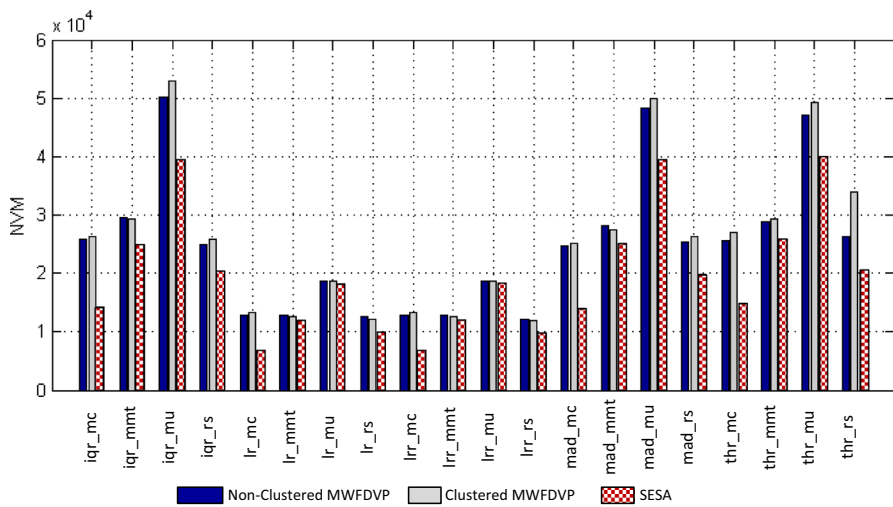


Fig. 10 Number of VM migrations

as the best reduction results come with “lr” and “lrr” combinational algorithms. It enhances the PD by 84.6% when using lr_mmt, lr_mu, lrr_mmt, and lrr_mu against the highest values with non-clustered MWFDP and by 57% when using lr_mc, and lrr_mc against the clustered MWFDP. Figure 12 shows SLAV with the different combinational algorithms. It differs in its behavior in the clustered and non-clustered approaches. However, SESA gives better results with almost all combinational algorithms.

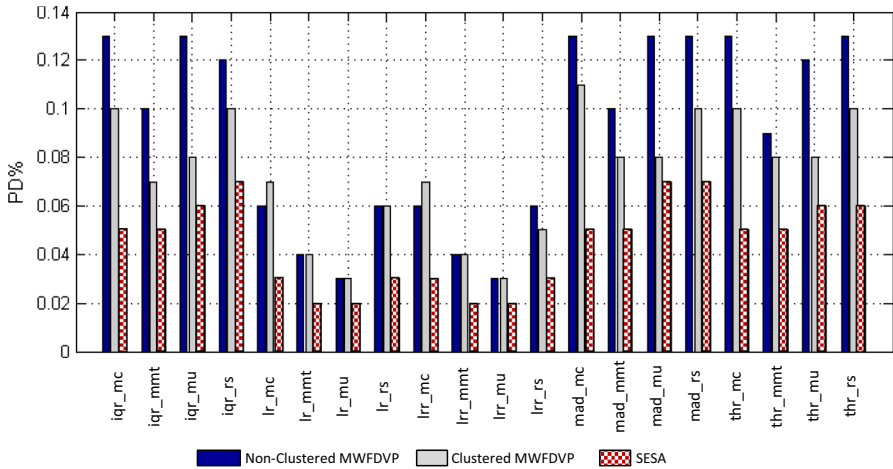


Fig. 11 Performance degradation due to VM migration

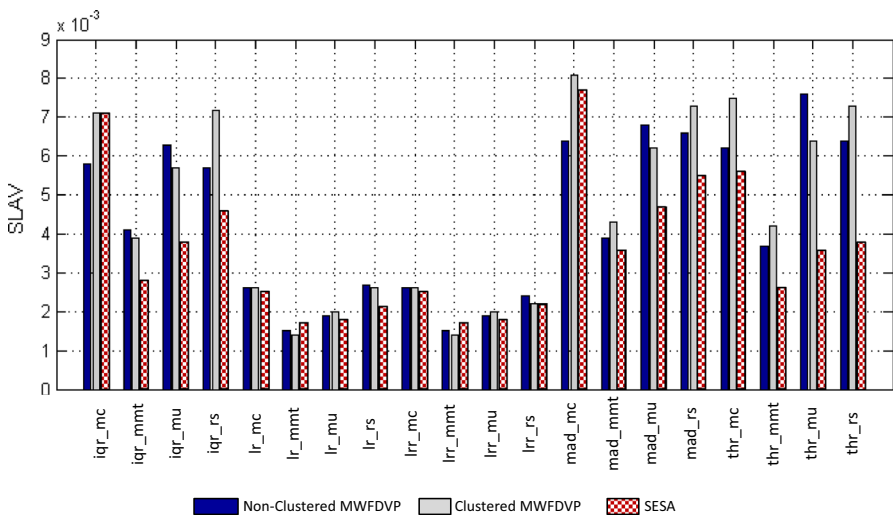


Fig. 12 SLA violation

(b) Comparative analysis with non-clustered approaches

In this section, SESA is compared with non-clustered VM placement algorithms: PABFD, MWFDVP, and FFDHDVP [26], and a non-clustered dynamic VM consolidation mechanism (PCM) [29]. In [29], PCM results were introduced with only three combinational algorithms from Table 1: lr-mc, lr-mmt, and lr-rs with non-clustered PABFD VM placement algorithm. The test was made by using the average for ten workload days' results as shown in Table 3. Therefore, lr-mc, lr-mmt, and lr-rs algorithms were considered in the comparison of SESA with

Table 5 Performance metrics for the proposed SESA against the non-clustered algorithms

| | PC | NVM | PD % | SLAV | ESV |
|--------------------------|--------|--------|-------|--------|-------|
| Non-clustered approaches | | | | | |
| PABFD | | | | | |
| lr_mc | 161.33 | 24,666 | 0.095 | 0.0061 | 1 |
| lr_mmt | 161.93 | 27,418 | 0.08 | 0.0053 | 0.87 |
| lr_rs | 160.05 | 25,118 | 0.096 | 0.0066 | 1.03 |
| MWFDVP | | | | | |
| lr_mc | 139.66 | 13,851 | 0.06 | 0.0023 | 0.32 |
| lr_mmt | 126.69 | 12,643 | 0.04 | 0.0017 | 0.22 |
| lr_rs | 141.4 | 13,710 | 0.06 | 0.0015 | 0.21 |
| FFDHDVP | | | | | |
| lr_mc | 155.66 | 17,184 | 0.08 | 0.003 | 0.47 |
| lr_mmt | 142.19 | 15,571 | 0.05 | 0.0026 | 0.37 |
| lr_rs | 153.67 | 15,627 | 0.05 | 0.0015 | 0.23 |
| SESA | | | | | |
| lr_mc | 111.58 | 4387 | 0.02 | 0.0017 | 0.189 |
| lr_mmt | 114.77 | 4455 | 0.02 | 0.0018 | 0.2 |
| lr_rs | 116.17 | 4458 | 0.03 | 0.0016 | 0.185 |
| PCM | | | | | |
| | 117.33 | 4462 | 0.039 | 0.001 | 0.105 |

non-clustered PABFD, MWFDVP, and FFDHDVP, and PCM by using the same workload days.

Table 5 shows the average of the results. It is shown from this table that lr_mc algorithm gives the best results compared with the other algorithms. Therefore, lr_mc algorithm is used to clarify the performance metrics for SESA against PCM and non-clustered PABFD, MWFDVP, and FFDHDVP algorithms in Fig. 13. It shows that SESA outperforms PCM and the other non-clustered algorithms in the all performance metrics except SLAV metric which increases slightly over PCM. Energy and SLA Violation (ESV) is another metric used in [29]. This metric evaluates the PCM based on both energy consumption and SLA violation rate. ESV is calculated by multiplying PC by SLAV values. As expected, ESV for SESA is more than for PCM since it depends on SLAV. However, ESV for SESA is better than the other non-clustered algorithms.

5 Conclusions

VM migration is an enabling technique for VM allocation process. Reducing the amount of transferred data during migration and the number of migration was a very important challenge to save cloud resources. In this paper, SESA is proposed in order to minimize the overhead of multiple colocated VMs migrations. It

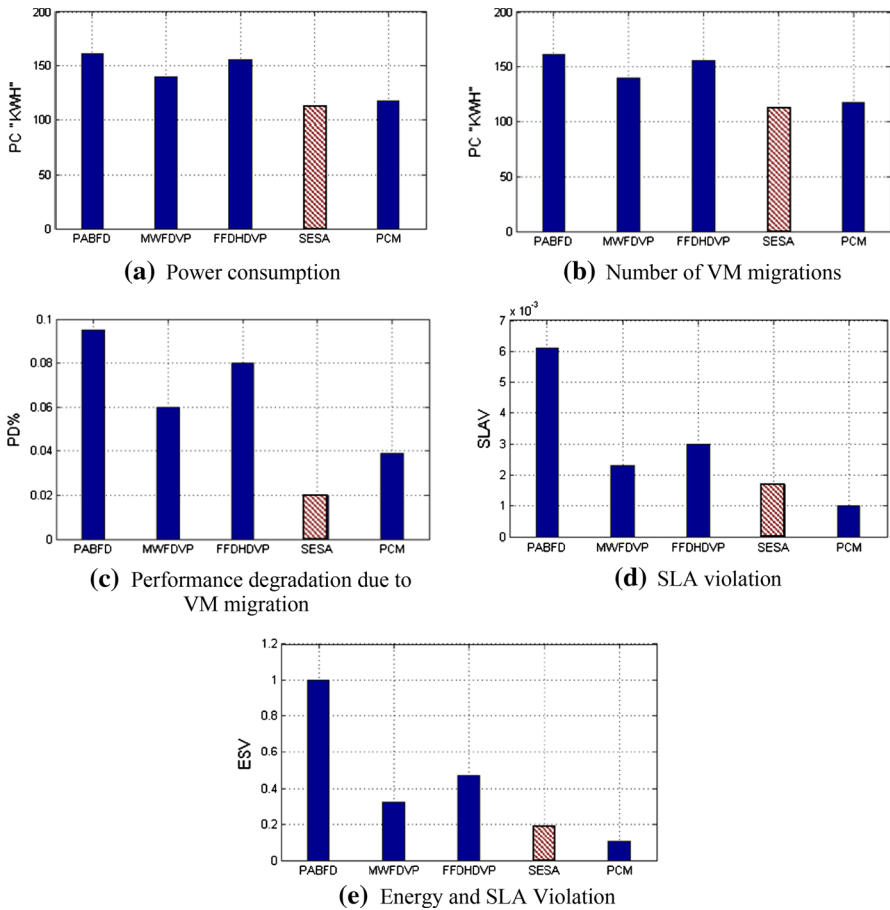


Fig. 13 Simulation results for the non-clustered algorithms

combines the proposed AWFDP VM placement algorithm with VM clustering and LGM techniques. This combinational proposed work results in a significant enhancement in the number of migrations in the system which leads to fewer data transfer. In addition, it has a good stamp in reducing the performance degradation in VM application, the PC for system's PMs and the SLA violation of the cloud system. From the simulation results, it is found that SESA gives the best performance metrics results when used with local regression-based algorithms that is used for hot-spot PM detection. As a future work, more detailed study will be introduced for LGM technique to use hashing code with deduplication algorithms in order to eliminate certain data from being transferred twice. Also, it is planned to put other performance metrics such as network bandwidth usage, the number of memory hashing iterations, and the ratio of deduplication to measure precisely the enhancement of the system.

References

1. Gorelik E (2013) Cloud computing models, comparison of cloud computing service and deployment models. The MIT Sloan School of Management and The MIT Engineering Systems, Massachusetts Institute of Technology
2. Hashem W, Nashaat H, Rizk R (2017) Honey bee based load balancing in cloud computing. *KSII Trans Internet Inf Syst (TIIS)* 11:5694
3. Gamal M, Rizk R, Mahdi H (2017) Bio-inspired load balancing algorithm in cloud computing. In: *Proceedings of the International Conference on Advanced Intelligent Systems and Informatics (AISI)*, Cairo, Egypt, pp 579–589
4. López-Pires F, Barán B (2017) Many-objective virtual machine placement. *J Grid Comput* 15(2):161–176
5. Strunk A (2012) Costs of virtual machine live migration: a survey. In: *Proceedings of IEEE 8th World Congress on Services (SERVICES)*, Honolulu, HI, USA, pp 323–329
6. Mishra M, Das A, Kulkarni P, Sahoo A (2012) Dynamic resource management using virtual machine migrations. *IEE018E Commun Mag* 50(9):34–40
7. Ren R, Tang X, Li Y, Cai W (2017) Competitiveness of dynamic bin packing for online cloud server allocation. *IEEE/ACM Trans Netw* 25(3):1324–1331
8. Beloglazov A, Buyya R (2012) Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *J Concurr Comput Pract Exp* 24(13):1397–1420
9. Deshp U, Wang X, Gopalan K (2011) Live gang migration of virtual machines. In: *Proceedings of the 20th International Symposium on High Performance Distributed Computing*, San Joes, CA, USA, pp 135–146
10. Zhen X, Weijia S, Qi C (2013) Dynamic resource allocation using virtual machines for cloud computing environment. *IEEE Trans Parallel Distrib Syst* 24(6):1107–1117
11. Sheng D, Cho-Li W (2013) Dynamic optimization of multiattribute resource allocation in self-organizing clouds. *IEEE Trans Parallel Distrib Syst* 24(3):464–478
12. Gouda KC, Radhika TV, Akshatha M (2013) Priority based resource allocation model for cloud computing (IJSETR). *Int J Sci Eng Technol Res* 2(1):215
13. Abirami SP, Ramanathan S (2012) Linear scheduling strategy for resource allocation in cloud environment. *Int J Cloud Comput Serv Archit (IJCCSA)* 2(1):9
14. Omara FA, Khattab SM, Sahal R (2014) Optimum resource allocation of database in cloud computing. *Egypt Inform J* 15(1):1
15. Abar S, Lemariner P, Theodoropoulos GK, O'Hare GMP (2014) Automated dynamic resource provisioning and monitoring in virtualized large-scale datacenter. In: *Proceedings of IEEE 28th International Conference on Advanced Information Networking and Applications (AINA)*, Victoria, Canada, BC, pp 961–970
16. Yexi J, Chang-Shing P, Tao L, Chang RN (2013) Cloud analytics for capacity planning and instant VM provisioning. *IEEE Trans Netw Serv Manag* 10(3):312–325
17. Minarolli D, Freisleben B (2014) Distributed resource allocation to virtual machines via artificial neural networks. In: *Proceedings of the 22nd Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, Torino, Italy, pp 490–499
18. Mandal U, Habib M, Shuqiang Z, Mukherjee B, Tornatore M (2013) Greening the cloud using renewable-energy-aware service migration. *J IEEE Netw* 27(6):36–43
19. Jie Z, Ng TSE, Sripanidkulchai K, Zhaolei L (2013) Pacer: a progress management system for live virtual machine migration in cloud computing. *IEEE Trans Netw Serv Manag* 10(4):369–382
20. Singh S, Chana I (2016) A survey on resource scheduling in cloud computing: issues and challenges. *J Grid Comput* 14(2):217–264
21. Rasmussen MATRV (2008) Round robin scheduling—a survey. *Eur J Oper Res* 188(3):617–636
22. Hottmar V, Adamec B (2012) Analytical model of a weighted round robin service system. *J Electr Comput Eng* 2012:374961
23. Chen B, Fu X, Zhang X, Su L, Wu D (2007) Design and implementation of intranet security audit system based on load balancing. In: *Proceedings of IEEE International Conference on Granular Computing*, Fremont, CA, USA, pp 588–588

24. Hielscher K-SJ, German R (2003) A low-cost infrastructure for high precision high volume performance measurements of web clusters. *In: Proceedings of the 13th International Conference on Computer Performance Evaluation. Modelling Techniques and Tools, Urbana, IL, USA*
25. Lu X, Zhang Z (2015) A virtual machine dynamic migration scheduling model based on MBFD algorithm. *Int J Comput Theory Eng* 7(4):278–282
26. Chowdhury MR, Mahmud MR, Rahman RM (2015) Implementation and performance analysis of various VM placement strategies in CloudSim. *J Cloud Comput* 4(1):21
27. Jain AK, Maheswari S (2012) Survey of recent clustering techniques in data mining. *Int Arch Appl Sci Technol* 3(2):68–75
28. Baswade AM, Nalwade PS (2013) Selection of initial centroids for k-means algorithm. *Int J Comput Sci Mob Comput (IJCSMC)* 2(7):161–164
29. Khoshkholghi MA, Derahman MN, Abdullah A, Subramaniam S, Othman M (2017) Energy-efficient algorithms for dynamic virtual machine consolidation in cloud data centers. *IEEE Access* 5:10709–10722
30. Ashry N, Nashaat H, Rizk R (2018) AMS: adaptive migration scheme in cloud computing. *In: Proceedings of the 3rd International Conference on Intelligent Systems and Informatics (AISI2018), Cairo, Egypt, vol 845. Springer, pp 357–369*
31. Melhem SB, Agarwal A, Goel N, Zaman M (2017) Markov prediction model for host load detection and VM placement in live migration. *IEEE Access* 6:7190–7205
32. Chang Y, Gu Ch, Luo F, Fan G, Fu W (2018) Energy efficient resource selection and allocation strategy for virtual machine consolidation in cloud datacenters. *IEICE Trans Inf Syst* E101.D(7):1816–1827
33. Beloglazov Planetlab workload traces. <https://github.com/beloglazov/planetlab-workload>. Accessed Nov 2018
34. Arianyan E, Taheri H, Sharifian S, Tarighi M (2018) New six-phase on-line resource management process for energy and SLA efficient consolidation in cloud data centers. *Int Arab J Inf Technol* 15(1):10–20