Sustainable Computing: Informatics and Systems 19 (2018) 185–203

Contents lists available at ScienceDirect

Sustainable Computing: Informatics and Systems

journal homepage: www.elsevier.com/locate/suscom

Type-aware virtual machine management for energy efficient cloud data centers

Auday Al-Dulaimy[a,*], Wassim Itani[b], Rached Zantout[c], Ahmed Zekri[a,d]

[a] Department of Mathematics and Computer Science, Beirut Arab University, Beirut, Lebanon
[b] Department of Electrical and Computer Engineering, Beirut Arab University, Beirut, Lebanon
[c] Department of Electrical and Computer Engineering, Rafic Hariri University, Beirut, Lebanon
[d] Department of Mathematics and Computer Science, Alexandria University, Alexandria, Egypt

ARTICLE INFO

Keywords:
Cloud Computing
Cloud Data Centers
Energy Efficiency
Multiple Choice Knapsack Problem
VM Consolidation
VM Placement

ABSTRACT

To meet the growing demands on cloud services and applications, a sizeable number of large scale cloud data centers, hosting thousands of heterogeneous servers, is established by cloud service providers. The ever growth in establishing cloud data centers is accompanied by consuming enormous amounts of energy. Thus, proposing efficient management approaches to reduce the energy consumption in cloud data centers becomes a top priority for ensuring the scalability of the cloud computing architecture. In general, the main source of energy over-consumption in today's data centers is due to the inefficient use of the physical servers' resources, which results in poor server utilization patterns. So, the key aspect is to utilize the physical resources optimally while serving the cloud user demands. This paper investigates the design and implementation of virtual machine management strategies for energy efficient cloud data centers. Particularly, it considers the processes of virtual machine placement and virtual machine consolidation in enhancing the energy efficiency in cloud infrastructures. While addressing the virtual machine placement problem is important, virtual machine consolidation is even more important to enable continuous reorganization of the already-placed virtual machines on the least number of physical machines. This results in reducing the number of active physical machines by leveraging live virtual machine migration enabled by the virtualization concept. Moreover, since the virtual machine migration operations consume additional energy, the frequency of VM migrations needs to be limited and controlled as well. The paper presents a distributed approach to an energy-efficient dynamic virtual machine consolidation mechanism. This approach determines, based on novel algorithms, which virtual machines to migrate, and when. Then, the placement of the virtual machines selected for migration is achieved based on a generalization of the Knapsack Problem known as the Multiple Choice Knapsack Problem. The placement process suits both static and dynamic virtual machine placement. The results of the performance evaluation demonstrate that the proposed new algorithms are able to enhance the energy efficiency in cloud data centers.

1. Introduction

Due to its innovative characteristics and services, the cloud computing model has captured significant attention among individual users, academia, industry, and even governments. The cloud computing services are offered by providing an access to a wide range of infrastructures hosted on cloud data centers. Those data centers consume extensive amounts of energy.

Moreover, as the cloud computing model has still many challenges, such as security and privacy, that are delaying its wide adoption, many approaches were proposed to overcome these challenges. But the proposed approaches themselves may consume extra energy. For example and as described in details in [1], many of the approaches proposed to solve the security and privacy issues consume energy in the encryption and decryption phases to make the processed and stored data more secure.

To address the problem of high energy consumption in cloud data centers, it is highly crucial to come up with an approach that utilizes the resources of the cloud data centers in an efficient way while keeping the number of active Physical Machines (PMs), hosting the Virtual Machines (VMs), as small as possible. In addition, investigating the VM management approaches and improving them, or proposing new approaches are necessary and seem promising in enhancing the energy efficiency of cloud data centers. One method to improve the resource

* Corresponding author.
E-mail addresses: auday.aldulaimy@gmail.com (A. Al-Dulaimy), w.itani@bau.edu.lb (W. Itani), zantoutrn@rhu.edu.lb (R. Zantout), a.zekri@bau.edu.lb (A. Zekri).

https://doi.org/10.1016/j.suscom.2018.05.012
Received 9 November 2017; Received in revised form 19 May 2018; Accepted 23 May 2018
Available online 26 May 2018

utilization, which in turn reduces the amount of consumed energy, is dynamic consolidation of VMs enabled by the virtualization concept. This paper presents a distributed approach to an energy-efficient dynamic VM consolidation mechanism. This approach determines which VMs to migrate, and when, based on new solutions. Then, the placement of VMs selected for migration is done based on a generalization of the Knapsack Problem (KP) known as the Multiple Choice Knapsack Problem (MCKP). The process of the proposed virtual machine placement strategy considers the Quality of Service (QoS) parameters stated in the Service Level Agreement (SLA). The results of the performance evaluation show that the proposed new solutions are able to enhance the energy efficiency in cloud data centers.

In addition to a detailed description on the virtualization concept in general, and on VM management in cloud computing environments, this paper tackles the problem of inefficient energy consumption in cloud data centers by proposing the following key contributions:

1) A distributed approach to energy efficient dynamic VM consolidation. The proposed approach presents novel ideas and solutions for:
   i) Deciding which VMs to be migrated and when by specifying the underloaded and overloaded PMs which host these VMs.
   ii) Selecting a VM to be migrated from the set of candidate VMs.
   iii) Selecting the PMs to host the migrated VMs.
2) A novel VM placement strategy, with a systematic mathematical formulation, that considers the types of jobs to be executed by the VMs before the process of VM placement. The strategy places the VMs' running jobs of different types and requirements on the same PM whenever possible based on the MCKP approach. MCKP suits our problem since it strives to fit items belonging to different sets in one knapsack. This strategy is applicable for both initial VM placement (Static placement), and for after migration VM placement (Dynamic placement).

The rest of this paper is organized as follows: The motivation and objectives of the paper are stated in Section 2. Section 3 provides some background material on the virtualization concept and VM management. It defines and describes the process of VM placement and consolidation. Section 4 categorizes and lists the related works of the VM placement and consolidation processes from the energy efficiency perspective. The proposed system model and the major algorithms of this paper are presented in section 5. Section 6 discusses the performance analysis of the proposed approach with its policies and algorithms. Conclusions are presented in Section 7.

2. Motivation and objectives

Recently, numerous data centers were established around the world. These data centers consume large amounts of energy. In general, the consumed energy amount is resulting in: excessive operating costs and carbon dioxide (CO_2) emissions [2].

This amount was estimated to be between 1.1% and 1.5% of the total electricity use in 2010. It has increased by 56% from 2005, and it will continue to increase in a rapid manner unless advanced energy efficient resource management algorithms are proposed [3,4].

According to the US Energy Information Administration (EIA), the World total primary energy consumption in 2015 was 575.5 Quad BTU[1] [5], which is approximately (168.6) Trillion KWh[2]. Consuming 1.1% to 1.5% from such massive amount of energy is prohibitively expensive.

Moreover, CO_2 emissions of the Information and Communication Technology (ICT) industry were accounted to be 2% of the total global emissions. As known, the CO_2 emissions affect the global warming [3].

Addressing the problem of high energy consumption is a significant issue due to its financial and environmental effects. One of the recent challenges in cloud computing is to enhance the energy efficiency of such data centers. So, it is important to improve the resource allocation algorithms and propose new management approaches which aim to enhance the energy efficiency in cloud data centers. To address the problem of high energy use, it is necessary to eliminate inefficiencies and waste in the way electricity is delivered to computing resources, and in the way these resources are utilized to serve application workloads. Usually, servers operate at 10–50% of their full capacity, leading to extra expenses on over resource provisioning. In addition, the problem of low server utilization is appearing by narrow dynamic power ranges of servers: even completely idle servers still consume about 70% of their peak power. Therefore, keeping servers underutilized is highly inefficient from the energy consumption perspective.

3. Background about virtualization and virtual machine management

Virtualization is creating a virtual version of a particular resource (e.g. CPU, memory, storage device, or network device). A single PM, which is the real hardware, can host one or more VMs. A VM is a piece of software running on a PM that emulates the properties of a separate PM.

The concept of virtualization breaks the traditional model of the PM that host a single Operating System (OS). It creates several VMs which are hosted on one PM, each VM may have its own OS. This concept is organized using hypervisor technology. A hypervisor [6], also termed as a Virtual Machine Manager (VMM), is a software that controls all PM resources, allocates resources needed by each operating system, monitors the utilization of the resources in turn, and makes sure that the guest operating systems of the VMs cannot disrupt each other.

3.1. Virtual machine management

VM Management [6] is the process of coordinated provisioning of the virtualized resources, as well as the runtime of such provisioning. This feature includes the mapping of the virtual resources to the physical ones, and the overall management of capabilities such as capacity, billing, and SLA contractual terms.

3.2. Virtual machine migration

An important issue in VM management is the VM migration process, which is the process of transferring a VM from a source to a destination PM. Basically there are two types of migrations:

Offline migration: It refers to moving a suspended VM from one host to another. It is also termed as cold migration.

Live Migration: It refers to moving a running VM from a host to another. It is also called hot migration.

In this paper, live migration is solely considered due to two main reasons: First is that live migration is more efficient from a performance point of view as it is able to transfer a VM between PMs with a close to zero downtime. Second is that live migration is the most adopted VM migration type in modern VM managers.

Although VM Live migration creates an extra load on the CPU, it has been shown that the performance overhead is low [7] compared to the saving resulting from utilizing the PMs more efficiently. By applying VM live migration, the running VMs can be dynamically consolidated to leverage fine-grained changes in the workload and keep the number of involved active PMs at the minimum during VMs execution [8].

3.3. Virtual machine live migration

VM Live migration is the process of moving a VM from one physical host to another, while the VM is still running with a close to zero

[1] BTU, which stands for British Thermal Unit, is a standard unit of energy.

[2] KWh, which stands for Kilo Watt per hour, is a standard unit of energy. Every one Quad BTU approximately equals to 293,071,000,000 KWh.

downtime [9,3]. VM Live migration has many techniques, to illustrate them, two terms need to be defined:

1) Total migration time: It represents the duration of the complete migration process from start to finish.
2) VM downtime: It represents how much time the VM takes to respond during migration.

These techniques attempt to make a trade-off between total migration time, and VM downtime. The first technique to be described is a pure *stop-and-copy* technique [10,8]. In this technique, the original VM is stopped, all pages are copied to the destination VM, and then, the new VM starts working. The advantage of this technique is its simplicity, but it has disadvantages, as the VM downtime is proportional to the amount of physical memory allocated to that VM leading to an unacceptable outage, if the VM is running a live service.

Another technique in VM migration is the *post-copy* technique [11,12]. In this technique, the essential kernel data structures are moved to the destination in a short stop-and-copy phase, before the destination VM starts. Other pages are transferred to the destination VM across the network on demand. The VM downtime is much shorter in this technique. However it is conducive to a much longer total migration time.

In the *pre-copy* migration technique [12,13], the migration is done without stopping the migrated VM. The memory pages are copied from the source machine to the destination one in an iterative manner. This iterative nature is proposed to serve moving the dirty pages (which are memory pages that have been modified in the source host since last page transfer) to the destination. Such pages must be resent to the destination host iteratively. The weakness of this technique shows when the rate of updating pages gets very high. In this case, the migration time will rise to a very high value. However, it has an advantage, since it provides the necessary updating at the destination host and hence there will be no need to re-update the pages and the destination host can be activated any time. Every VM will have some set of pages that it updates very frequently, the thing which makes them poor candidates for future pre-copy migration. However, in [14,15], modified approaches are proposed to enhance the VM management techniques from the perspective of the total migration time, and VM downtime.

3.4. Key aspects of live migration

In live migration, three important key aspects must be considered [16], these are: the CPU state, the memory state, and the storage content.

- *CPU State*: During live migration, the CPU state context of the VMs are switched from the source host to the destination one. It is a little data to be transferred, and represents the lowest limit for minimizing the live migration downtime. Transferring such kind of information from the source to the destination PM host is essential and not optional.
- *Memory Content*: The VMs memory state also needs to be transferred to the destination host. This information is greater than the CPU state. It includes the memory state of:
  1) the guest OS,
  2) all running processes within the VM.

To substantially reduce migration time, VMMs need to be specified, and transferring the contents of unused memory should be avoided. This is very crucial because transferring memory content is the main factor that effects the live migration time. In addition to avoiding transferring the contents of unused memory, compression and other techniques have the ability to speed up the transfer of memory content. Transferring memory content is also not optional in live migration.

- *Storage Content*: The Storage content is the optional part of live migration. There is no need to transfer the storage content (also called VM image), if it is accessible to both the source and the destination machines through Network Attached Storage (NAS). On this front, live migration is divided into two types:
  1) Memory live migration: It occurs when transferring the memory contents only.
  2) Storage live migration (or shared-nothing migration): If the storage cannot be accessed by the destination host, then a new storage virtual disk needs to be registered on the destination host, and the storage content needs to be synchronized from the source to the destination

Disk storage represents, by far, the greatest deal of information to be transferred, and the time to transfer the full disk image on the network can be substantial. As with memory, VMMs that can identify, and avoid transferring the contents of unused disk blocks, have the potential to greatly reduce migration time.

3.5. Migration cost

Live migration of VMs allows transferring a VM across PMs without suspension and with a short downtime. However, live migration has a negative impact on the performance of applications running in a VM during a migration. It may make each VM migration a trigger to an SLA violation; therefore, it is crucial to minimize the number of VM migrations.

The length of a live migration depends on the total size of memory used by the VM, and the available network bandwidth. Migrating only the memory content is reasonable. The images and the data of VMs are stored on an accessible shared storage, which is required to enable live migration; and hence, copying the VM's storage is not required.

One model to estimate energy consumption of VM migration cost was developed in [17] and is summarized in (1 and 2):

$$VM_{vm}^{Mig.energy} = E_{Source} + E_{Destination} \tag{1}$$

$$= (\delta_{Source} + \gamma_{Source})^{*}\, nwTRAFFIC_{Mig} + (\delta_{Dest} + \gamma_{Dest}) \tag{2}$$

where

- $nwTRAFFIC_{Mig}$ is the network switching energy
- δ_{Source} , γ_{Source} , δ_{Dest} , γ_{Dest} are model parameters to be trained in both source and destination hosts.

In addition to energy dissipation, VM migration also takes time. Therefore, time must be considered before taking any migration-related decision. The total migration time, depends on the number of VMs to be migrated, and the RAM size of each VM. However, the total number of VMs to be migrated, and the size of each VM cannot be known in advance except in a probabilistic sense [18].

The time cost of migrating of a single VM can be expressed in terms of the statistics of the CPU content size and RAM size of this VM as well as the available bandwidth at the time of migration. It is important to stress that the RAM size is the main parameter that affects the VM migration time (and consequently RAM size affects the energy since energy is propositional with time). Eq. (3) is suggested to measure the VM migration time in this paper:

$$VM_{vm}^{Mig.time} = (C_{vm} + R_{vm})/BW \tag{3}$$

where

- $VM_i^{Mig.time}$ is the time duration of VM_i migration,

- C_{vm} is the CPU content size in Bytes,
- R_{vm} is the RAM content size of this VM in Bytes, and
- BW is the available bandwidth, in Bytes/Seconds, between the source and destination PMs.

The total migration time ($TotalVM^{Mig_{time}}$) for n VMs is given in Eq. (4):

$$TotalVM^{Mig.time} = \sum_{vm=1}^{n} VM_{vm}^{Mig.time} \tag{4}$$

In evaluating $TotalVM^{Mig.time}$, the values of $VM_{vm}^{Mig.time}$ are statistically independent for all VM_i.

3.6. VM placement and consolidation

VM Placement is the process of mapping VMs to PMs. *VM Consolidation* refers to the replacement process using a fewer number of PMs, thus contributing to energy conservation.

As virtualization is a core technology of the cloud computing model. Investigating the problems of VM placement, and VM consolidation, is an essential part of this model. Investigating these two problems is an important approach to a more efficient energy use, and a better utilization of resources in cloud data centers.

To explain the VM placement and VM consolidation more, Fig. 1 shows two cases, in each case seven VMs execute on a different number of PMs (assuming that the PMs capacities can host the VMs in both cases).

Case 1 can result from:

1) Unconditional initial VM placement.
2) After finishing the execution of some VMs while other VMs are still working on the same PM.

While case 2 can result from:

1) Optimal/near optimal initial VM Placement.
2) Consolidating the VMs of Case 1 via VM migration.

It is obvious that case 2 is more energy efficient, as the number of the PMs involved in serving the running VMs is less, and the involved PMs are utilized better as well.

4. VM placement and consolidation approaches: literature review

Following are some of the approaches that have been suggested in the literature to deal with the VM placement and VM consolidation problems from the perspective of enhancing energy efficiency in cloud data centers:

Round Robin: It is the basic and most straightforward VM placement method where VMs are placed and distributed on PMs of the data center sequentially in a circular manner.

Linear Programming: It is a traditional approach to solve the problem of VM placement. The work in [19] relied on the linear and quadratic programming in proposing a VM placement algorithm to

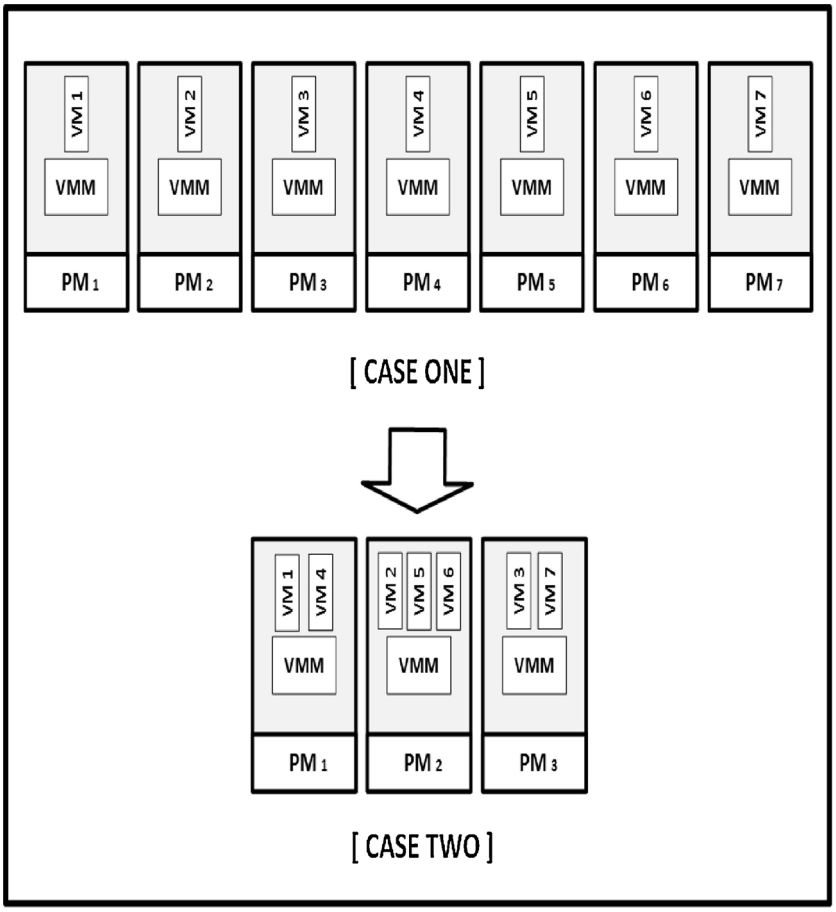

Fig. 1. VM placement and consolidation.

minimize the number of the involved PMs. The authors in [20] also proposed a linear programming approach for VMs consolidation, where VMs are allocated to a specific number of PMs. In [21], the authors modeled an energy-aware VM placement using linear programming technique to find the optimal solution from the energy-efficiency perspective.

Constraint Programming: It is another approach to solve the VM placement problem. The authors in [22] proposed a framework to maximize the utilization of the VMs resources. They defined a dynamic VM provisioning manager, and a dynamic VM placement manager, to work together within the proposed framework. VM provisioning, and VM placement, are modeled as constraint satisfaction problems. The work in [23] proposed a resource manager, called Entropy, working in homogeneous clusters. Entropy is utilized to perform dynamic consolidation based on constraint programming.

Bin Packing: The Bin packing approach is also used in solving the problem of VM placement in cloud data centers. Many studies presented the VM placement problem as a bin packing problem. In [3] and [24], novel models and algorithms were presented for distributed dynamic consolidation of VMs in cloud data centers. The applied VM placement algorithm based on a variant of bin packing called Best Fit Decreasing (BFD) bin packing. The work in [25] proposed multiple VM placement approaches to identify the parameters with the highest impact on the total energy consumption, carbon footprint, and cost. The proposed approaches, which work in geographically distributed cloud data centers, aim to maximize the energy utilization of the PMs in each data center in order to minimize the total cost. The VM placement problem was solved by considering it as a bin-packing problem with different bin sizes, where bins represent the PMs. A Best Fit Heuristic is used to deal with the process of VM placement. In [26], an energy-efficient VM consolidation algorithm, called Prediction-Based VM Deployment algorithm for energy efficiency (PVDE), was proposed. PVDE employed a linear weighted method to predict the load of the PMs and classified them based on the load into four classes for the VMs migration if necessary. The work proposed four types of VM selection algorithms to determine potential VMs to be migrated. Then, the VM placement problem is modelled as a bin packing problem with variable bin sizes and prices. A Modified Best Fit Decreasing Algorithm (MBFDA) is proposed to deal with the VM placement problem.

Knapsack Problem: This approach, as well, is employed to solve the VM placement problem. An example of using this approach was presented in [27]. The authors in [27] proposed a power-saving approach to forecast the demands on the VMs for the next period based on Holt-Winters' exponential smoothing method. Then, a knapsack algorithm is employed to place the VMs on the PMs hosted in the data center.

In addition to the previous approaches, various optimization methods such as, Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and Genetic Algorithm (GA) are used to solve the problems of VM placement and VM consolidation, in virtualized data centers.

ACO: In [28], the authors defined an initial VM placement strategy with a multi-objective optimization algorithm based on (ACO). The proposed algorithm is able to achieve an optimal solution through efficient convergence by the constantly updated pheromone. The optimal solution is selected from a set of solutions using the exclusion method. In [29], the authors designed a distributed ACO-based algorithm for solving the VM consolidation problem. The algorithm iterated over a finite number of cycles. At each cycle, an item is selected for each ant to be packed in a bin. In [30], the authors proposed a multi-objective ACO to solve the problem of VM placement. The formulated multi-objective VM placement problem represents a permutation of VM assignments. The goal is to efficiently obtain a set of non-dominated solutions that reduce the total high power consumption resulting from resource wastage. The dissertation in [31] used an ACO-based VM placement algorithm to solve the problem of considering only a single resource to evaluate the PM load, and VM resource demands, while ignoring the other resources. A consolidation algorithm based on ACO to achieve both scalability, and high data center utilization, is also proposed in [31]. In [32], the authors proposed a VM consolidation scheme that focused on balanced resource utilization of servers across different computing resources. In [33], the energy consumption during VMs migration is considered as a main factor in proposing an Energy-aware VM Consolidation approach. This approach formulated the VM Consolidation problem as a multi-objective optimization problem. The considered objectives are the number of sleeping PMs, and the memory size of the migrating VMs. Then, ACO is applied to solve the problem.

PSO: The work in [34] proposed a PSO based VM scheduling strategy for VM placement. This strategy focused on efficient VM placement, aiming to minimize the number of the PMs used. In [35], the authors proposed a model, based on PSO, to place the migrated VMs in the over-loaded PMs on other hosts, and to consolidate the under-loaded PMs, which saves power. In [36], a novel VM selection algorithm based on memory utilization, BW utilization, and VM size is proposed. After selecting the VM, the work presented a modified Discrete Particle Swarm Optimization (DPSO) method to solve the VM placement problem. In [37], the VM placement problem is modeled as a multi-objective function which considers both the resource wastage and energy consumption as two conflicting objectives. To solve the problem, a binary version of PSO with some modifications is presented.

GA: Another approach to solve this problem is to use GA. Examples of this approach were presented in [38] and [39]. The authors in [38] proposed a Genetic Algorithm for Power-Aware (GAPA) scheduling to find the optimal solution for the problem of VM placement. In the proposed algorithm, a tree structure is used to encode a chromosome of an individual job. The fitness function of GA is calculating the evaluation value of each chromosome. Using this model, each instance of the tree structure showed the VM to PM placement. In [39], a profile-based framework is proposed for dynamic VM allocation. The framework estimates the applications finishing times and then implements a dynamic assignment strategy based on a Repairing Genetic Algorithm (RGA). By employing realistic profiles for applications, VMs and PMs, RGA works in a three-layer energy management system to perform energy efficient VM placement.

In general, by studying the approaches presented in the literature to enhance the energy efficiency in virtualized cloud data centers, it could be noted that the approaches with conditional VM placement are much better compared with random VM placement approaches. The conditional VM placement approaches results in better energy efficiency.

Also, it is important to mention that examining the workload characteristics can lead to a better resource utilization, which consequently enhance the energy efficiency. The work in [40] showed the importance of investigating the workload characteristics before the process of VM placement. It presented a VM placement strategy based on the peak workload characteristics. After measuring the similarity of VMs' workload with VM peak similarity, VMs with peak workload staggering at different time are placed together. The strategy aimed to achieve better VM consolidation through VM peak similarity, resulting in better resource utilization and more energy efficient cloud data centers.

However, to the best of our knowledge, no work from the literature considers the type of the tasks/jobs/applications which are served by the VMs, nor employ the MCKP approach in solving the VM placement problem. The work presented in this paper is inspired from our previous work in [41].

5. Distributed dynamic virtual machine consolidation model

Consolidation is considered highly important in enhancing the

energy efficiency in cloud data centers. As the modern data centers do not maximize the resource utilization of their PMs, co-locating VMs allocated to the users' jobs allows to minimize the total number of the PMs which are involved in serving those jobs. Particularly, the involved PMs are not energy proportional, in other words, they consume energy even when they are idle. So, switching some of them off is worthy from the energy consumption perspective.

There are two main VM consolidation categories: static and dynamic. *Static VM consolidation* assumes empty PMs when starting the process of VMs to PMs placements. In contrast, *Dynamic VM consolidation* assumes pre-placed VMs when starting the process of VMs to PMs placements, which is a more complex process [42].

In the literature, the importance of VM consolidation on energy efficiency in cloud data centers is studied and verified. In 2007, one of the first works which applied dynamic VM consolidation to minimize energy consumption in a data center, has been performed in [43]. The energy benefits obtained by consolidating VMs using migration are explored and found that the overall energy consumption can be significantly reduced compared with the studies that ignore the VM consolidation. However, the authors did not apply any explicit algorithm for determining when it is necessary to optimize the VM placement.

In [44], the authors investigated the effects of dynamic consolidation of applications on minimizing energy consumption in the data center. They showed that dynamic consolidation influences the relationship between energy consumption and utilization of resources. When the resources utilization is low, the resource is not efficiently used leading to a higher cost in terms of the energy-performance metric. At the same time, when the resource utilization is high, this results in an increased cache miss rate, context switches, and scheduling conflicts leading to high energy consumption due to performance degradation and consequently longer execution times. Based on the results of their experiments, the authors stated that, from the energy-aware point of view, the goal of dynamic consolidation is to keep the PMs efficiently utilized, while avoiding performance degradation caused by high utilization.

It is worth mentioning here that the energy consumption resulting from live migration is a relatively small value, compared to the energy saving gains incurred from VM consolidation [45].

This section presents a distributed Dynamic VM consolidation approach for enhancing the energy efficiency in cloud data centers. The target system which relies on the one presented in [3] is demonstrated in Fig. 2, by a data center consisting of N heterogeneous *PMs*. Each node PM_i has a multi-core CPU (the CPU performance is defined in MIPS), amount of RAM, and network bandwidth. PMs do not have direct-attached storage, as the storage is provided by a Network Attached Storage (NAS), or a Storage Area Network (SAN) to enable the process of VM live migration.

Multiple independent users submit requests for the provisioning of M heterogeneous VMs which have the characterizations that meet the requirements of the PMs resources (defined in MIPS, amount of RAM, and network bandwidth). The data center has a Local Manager (LM) (which resides on each *PM*), and a Global Manager (GM) (which maintains the overall system's resource utilization of a set of *PMs* by interacting with their LMs).

In the data center, LMs and GMs work as follows:

1) In each PM, the LM detects under/over load conditions by monitoring the PM utilization.
2) The GM collects information from the LMs to maintain the overall view of the system's resource utilization.
3) The GM issues *VM* migration commands to optimize the *VM* placement, based on the information collected by the LMs.
4) VMMs perform actual migration of *VMs*. VMMs are also responsible for changing the power modes of the nodes.

As in [3], the approach to dynamic VM consolidation follows a distributed model, where the problem is divided into 4 sub-problems:

1) Host under-load detection: Determining if a host is considered to be under-loaded, so that all VMs should be migrated, and the host should be switched to a low-power mode.
2) Host overload detection: Determining if a host is considered to be overloaded, so that some VMs should be migrated to other active, or reactivated hosts, to meet the QoS requirements.

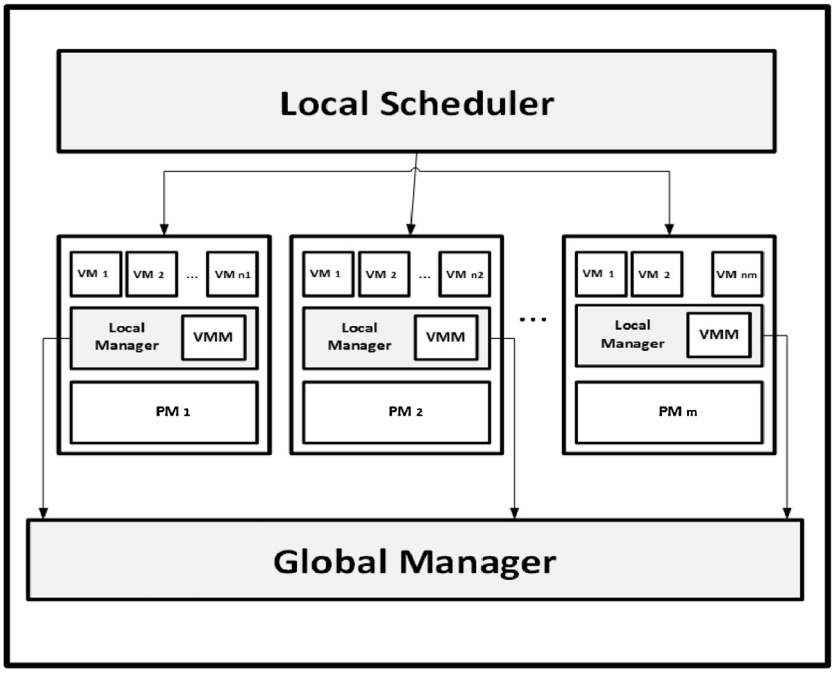

Fig. 2. The Data Center Model.

3) VM selection: Selecting VMs that would be migrated from an under/overloaded hosts.
4) VM placement: Placing VMs selected for migration to other active, or reactivated hosts.

The four sub-problems, and the proposed solution to deal with each one of them, are discussed in the next subsections.

5.1. Under-loaded physical machine detection

When the PM utilization is low, it is worthy to transfer the hosted VMs and switch the PM to off mode. Such case can be result if the job prediction is not precise, or after finishing the execution of some VMs.

To detect the under-load utilization, this study depends on the resources utilization of the node to select a threshold that specifies the under-load in that PM. Utilization of the host can be measured as follows:

$$U_{PM}(t) = \left(\sum_{i=1}^{No\ of\ Resources\ in\ PM} \frac{actual\ use\ of\ Resource_i}{total\ capacity\ of\ Resource_i} \right) * 100 \quad (5)$$

where:

$U_{PM}(t)$: *utilization of the PM at time t*

According to the PM utilization, two threshold techniques are used: static and dynamic.

1) Static Thresholding: A fixed value is set to a utilization threshold, and below this value, the PM is considered under-loaded. However, this is the simplest under-load detection technique.
2) Dynamic Thresholding: In general, a cloud computing environment has dynamic, and unpredictable workloads. Therefore, fixed values of the utilization threshold are unsuitable for such environment. The system should be able to automatically adjust its behavior depending on the workload patterns. Two methods are proposed in this study:
3) Adaptive Thresholding (AT): This study proposes an Adaptive Thresholding (AT) technique based on statistical analysis of historical data, collected during the lifetime of VMs. AT adjusts the value of the CPU utilization threshold depending on:
   - Mean Adaptive Thresholding ($MEAN_{AT}^{underload}$): Depending on the mean of the values of the minimum PMs utilization in the data center, an adaptive value is set to a utilization threshold. This value is changed frequently due to the dynamic, and the unpredictable workloads of the cloud environment.
   - Median Adaptive Thresholding ($MEDIAN_{AT}^{underload}$): As in $MEAN_{AT}^{unerload}$ but it depends on the median, rather than the mean. An adaptive value is set to a utilization threshold. This value is also changed frequently in a cloud environment.
4) Boxplot: It is one of the most frequently used graphical techniques for analyzing data sets. To create a boxplot (as the one in Fig. 3), the elements of any data set are arranged in ascending order. Then, the median value of the arranged numbers, (called in this method Q2), is calculated. Q2 divides the data into two subsets. To divide the data into quarters, the medians of these two subsets are calculated too. The median of the left subset is called Q1, while the median of the right one is called Q3. If the data set has an even number of elements, Q1 will be the average of the two middle elements. This works with the two subsets as well, if they have an even number of elements. The work in [46] explains the details of solving the boxplot method.

The boxplot thus shows information about the location, and the spread of the data by means of the median, and the interquartile range. The length of the whiskers on both sides of the box, and the position of the median within the box, are helpful in detecting any possible skewness in the data. Finally, observations, which fall outside the fences, are pinpointed as outliers. PMs of utilization less than Q2 are considered under-utilized PMs.

5.2. Overloaded physical machine detection

When the CPU utilization of a specific PM approaches 100%, the VMs allocated to this PM do not get the required CPU capacity which, in turn, may lead to performance degradation. The reason behind this observation is that: if the PM which hosts and serves VMs allocated to users' jobs is experiencing 100% utilization, then, the performance of the jobs is constrained by the PM's capacity. Therefore, the VMs are not

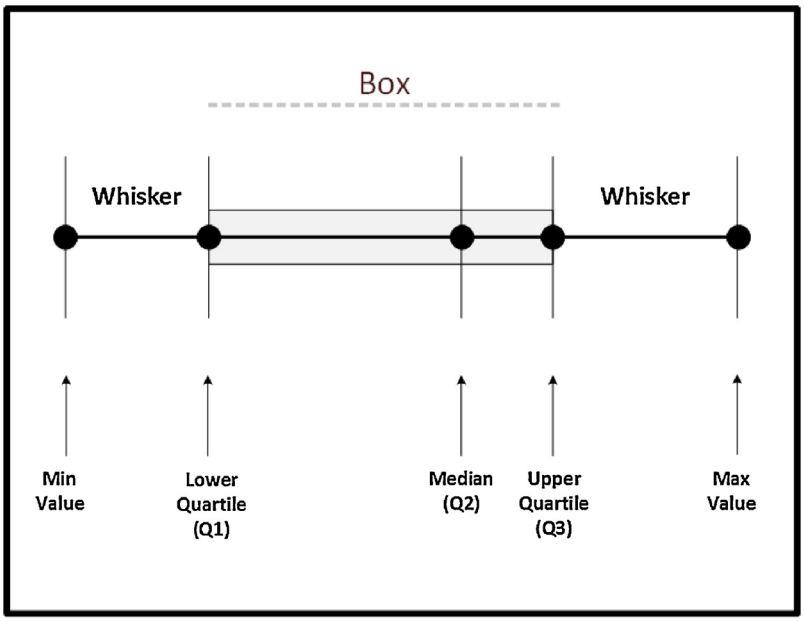

Fig. 3. Description of the Boxplot Method.

being provided with the required performance level. Such case can result if the job prediction is not precise, or as a results of unconditional placement.

As in under-load utilization detection, the overload utilization in this work depends on the resource utilization of the node to select the required threshold indicating that the PM is overloaded. The PM overload occurs when there are insufficient resources to meet the system demands. The VMs hosted on the overloaded PMs may not get the required resources under peak load conditions, and thus fail to meet the QoS of the jobs they serve.

According to the PM utilization measured by Eq. (5), two techniques are used to detect the overloaded PMs (as in under-loaded PMs detection): static and dynamic.

1) Static Thresholding: The simplest overload detection technique is to set a fixed value as a utilization threshold, and above this value, the PM is considered overloaded.
2) Dynamic Thresholding: The system should be able to automatically adjust its behavior depending on the workload patterns. Two methods are proposed in this study to detect overload utilization:
3) Adaptive Thresholding (AT) technique is proposed based on statistical analysis of historical data collected during the lifetime of VMs. AT adjusts the value of the CPU utilization threshold depending on:
   - Mean Adaptive Thresholding ($MEAN_{AT}^{overload}$): Depending on the mean of the values of the maximum PMs utilization in the data center, an adaptive value is set to a utilization threshold. This value is changed frequently due to the dynamic, and unpredictable workloads of the cloud environment.
   - Median Adaptive Thresholding ($MEDIAN_{AT}^{overload}$): Quite as in $MEAN_{AT}^{overload}$ but it depends on the median rather than the mean. An adaptive value is set to a utilization threshold. This value is also changed frequently in cloud environment.
4) Boxplot: As in under-load detection, the observations that fall outside the fences are pinpointed as outliers. PMs of utilization greater than Q4 are considered over-utilized PMs.

5.3. VM selection

Four main policies are used in this work to select the VM to be migrated, they are: The Random Sampling (RS) Policy, The Systematic Sampling (SS) Policy, The Minimum Time Cost Migration (MTCM) Policy, and The Minimum Energy Cost Migration (MECM) Policy. However, RS, MTCM and MECM policies are used in the literature, and they are included here for the purpose of comparison with the proposed SS policy.

In all policies, to meet the deadline of the jobs, the selected VM to be migrated must consider Eq. (6):

$$ExT_i^{vm,pm,d} + VM_i^{Mig.time} \leq DL_i \quad (6)$$

where:

- $ExT_i^{vm,pm,d}$ *is the total execution time for* job_i *when executed on the VM allocated to it.*
- $VM_i^{Mig.time}$ *is the time duration of migrating the VM allocated to* job_i
- DL_i *is the deadline of* job_i

The details of the VM selection policies are:

i. RS Policy: RS randomly selects a VM from a set of VMs to be migrated from the source host to the destination one.

Algorithm 1: RS policy

Input:	*Set of* $VMs = \{VM_1, VM_2, \ldots, VM_n\}$
Output:	*Selected element* VM_s
1	$VM_s = \textbf{random}\ (VMs)$
2	***return*** VM_s

The time complexity of Algorithm 1 is $O(1)$. This is due to the fact that the random VM selection method (line 1 in Algorithm 1) does not depend on the number of VMs and thus executes in a constant time algorithmic complexity.

ii. SS Policy: SS policy is similar to RS, but with a slight difference. Suppose that n VMs are numbered from 1 to n in some order. Then the set of VMs is divided into R regions (S_1, S_2, ..., S_R), each region or subset S_r has s_r elements. The total number of VMs is equal to the sum the number of region' elements, $n = \sum_{r=1}^{R} s_r$. The classic SS method selects one VM from each region S_r. The selected VMs from each region are combined in one set. From this new set, the final selected VM is chosen randomly as in RS policy.

Algorithm 2: SS Policy

Input:	*Set of* $VMs = \{VM_1, VM_2, \ldots, VM_n\}$
Output:	*Selected element* VM_s
1	*choose the number of regions (R)*
2	***for*** *each* R_i ***do***
3	$TempVM_s = Random(R_i)$
4	*Add the element* $TempVM_s$ *to the array* $TempVM$
5	$VM_s = Random\ (TempVM)$
6	***return*** VM_s

In this work, the classic SS is modified by following two directions:

- Dividing the VMs set into four subsets according to the job types, which these VMs are allocated. Then select a VM from each subset randomly to examine the effect of job type on VM migration. This direction is suggested to examine the effect of job type on VM migration process. Four sub policies result from this direction.
- Dividing the VMs set into subsets equal to the number of VMs type, offered by the provider. There are four types of VMs used in this study, and they are offered by Amazon as the "M family" (medium, large, xlarge, and 2xlarge). Then select a VM from each subset randomly. This direction is suggested to examine the effect of the VM instance type on the VM migration process. Four sub policies are result from this direction as well.

  The time complexity of Algorithm 2 is $O(R)$, where R is the number of regions (i.e. R is constant). Here again the algorithmic complexity of Algorithm 2 does not depend on the number of VMs but rather on the number of regions R which is a constant independent on the number of VMs.

iii. MTCM Policy: In this policy, all VMs to be migrated are sorted in ascending order in a queue according to the migration time of such VMs that they spend from the source to destination PMs. Then MTCM selects the VM which is on the top of the queue. The migration time is calculated using Eq. (3). In this work, the value of the available bandwidth between the source and destination PMs in Eq. (3) is assumed to be equal among all PMs. Another assumption is that there is no overhead traffic.

Algorithm 3: MTCM policy

Input:	*Set of* $VMs = \{VM_1, VM_2, \ldots, VM_n\}$
Output:	*Selected element* VM_s
1	***for*** $i = 1$ ***to*** n ***do***
2	*Calculate* $VM_i^{Mig\_time}$
3	***sort*** (VM) *such that* $VM_i^{Mig\_time} < VM_{i+1}^{Mig\_time}$
4	$VM_s = VM[1]$
5	***return*** VM_s

The time complexity of Algorithm 3 is $O(n^2 \log n)$, where n is the number of the selected VMs to be migrated. This is due to the fact that the for loop in step one iterates n times and thus takes O(n) algorithmic time to calculate the minimum migration time among the selected VMs, and the sort process in step 3 is performed in O(n log n). This renders the total time complexity to be the product of n and n log n, hence $O(n^2 \log n)$.

iv. MECM Policy: According to this policy, all VMs to be migrated are sorted in ascending order in a queue, depending on the energy such VMs consume when moved from the source PM to the destination PM. Then MECM selects the VM, which is on top of the queue. The migration time can be calculated using Equations (1 and 2). In this work, the energy consumed by VM migration is not based on Equations (1 and 2), instead, the consumed energy is estimated as a function of memory content size as energy is proportional to VM memory content size.

So, the energy consumed during the process of VM migration can be expressed as:

$$VM_{vm}^{Mig\_energy} = (C_{vm} + R_{vm}) * VM_{vm}^{Mig_{time}} \quad (7)$$

The value of C_{vm} is negligible, so it may not be considered in Eq. (7).

Algorithm 4: MPCM policy

Input:	*Set of VMs* = $\{VM_1, VM_2, \ldots, VM_n\}$
Output:	*Selected element* VM_s
1	***for*** $i = 1$ ***to*** n ***do***
2	*calculate* $VM_i^{Mig\_energy}$
3	***sort*** *(VM) such that* $VM_i^{Mig\_energy} < VM_{i+1}^{Mig\_energy}$
4	$VM_s = VM[1]$
5	***return*** VM_s

Similar to the algorithmic complexity analysis of Algorithm 3, the time complexity of Algorithm 4 is $O(n^2 \log n)$, where n is the number of the selected VMs to be migrated, as the loop in step one takes O(n) to calculate the minimum energy consumed during the migration process among the selected VMs, and the sort process in step 3 can be performed in O(n log n).

5.4. Virtual machine placement

VM placement is the process of mapping the VMs into their best fit PMs. The problem of VM placement arises in two places of the VM management process:

- **Initial VM placement** (**Static VM Placement**): It is the process of placing a set of VMs to start their execution on the PMs of the data center, considering the requirements of VMs, the capacities of PMs, and some other factors such as the consumed energy and the system performance. This type of placement has long-term effects because extensive changes in VM placement are impractical due to the overhead incurred and time consumed by multiple VM migrations. Generally, it occurs much less frequently than dynamic placement, for example, it occurs when the data centers start operation or when dynamic resource provisioning leads to unsatisfactory states. For such scenarios, the system can take a long time (relative to dynamic placement) to determine the placement, which is typically an NP-hard problem.
- **After Migration VM Placement** (**Dynamic VM Placement**): It is the process of re-placing the VMs on other PMs to balance the system conditions, or to adapt with the changes in the VM requirements. So, this type of placement is a VM consolidation. For this type of placement, the system is required to make decisions at runtime to overcome any performance degradation.

In this paper, a novel model is proposed to solve the problem of VM placement. The model is based on MCKP, which is a generalization of the classical version for KP. The proposed model is employed for both *Initial VM placement*, and *After Migration VM Placement*.

The MCKP model is chosen because it is able to target two goals:

1. Minimizing the total number of the involved knapsacks by the process of items packing (in our case, every knapsack represents one PM) which minimizes the consumed energy.
2. Maximizing the utilization of the involved PMs, resulting in combining VMs which different types of jobs on the same PM whenever possible. Different types of jobs utilize different physical resources, and in this case, all resources are keeping busy, which in turn, enhance the PM utilization

Minimizing the number of the involved PMs, which are utilized in an optimal way, leads to enhance the overall energy efficiency of the cloud data centers.

KP, and all its generalizations, is divided into two forms: **Fractional KP**, where the items can be broken into fractions, and then, the fractions are placed on the knapsacks, and **0-1 KP,** where the items cannot be divided, they are either picked as complete items to be placed on the knapsacks, or are not picked. In our case, the VMs cannot be divided. So, the KP form used in presenting the VM placement process is the 0-1 KP.

5.4.1. Physical machines representation

The set of PMs in the cloud data center is represented as follows: $PM = \{pm_1, pm_2, \ldots, pm_m\}$. Each PM has a limited capacity of the following resources: processing core(s), RAM, storage, and bandwidth. The capacities are represented as follows:

- $Capacity_{core}(j)\ \forall j \in \{1,2, ..m\}$ represents the available cores of pm_j,
- $Capacity_{RAM}(j)\ \forall j \in \{1,2, ..m\}$ represents the available RAM of pm_j,
- $Capacity_{storage}(j)\ \forall j \in \{1,2, ..m\}$ represents the available storage of pm_j,
- $Capacity_{bandwidth}(j)\ \forall j \in \{1,2, ..m\}$ represents the available bandwidth of pm_j.

So, the total capacity of the resources of any PM can be represented the vector as shown in Equation (8):

$$PM_j^{Cap} = \begin{pmatrix} Capacity_{core}(j) \\ Capacity_{RAM}(j) \\ Capacity_{storage}(j) \\ Capacity_{bandwidth}(j) \end{pmatrix} \quad (8)$$

5.4.2. Virtual machines representation

The set of VMs which will be hosted on the set of PMs of the data center is represented as follows: $VM = \{vm_1, vm_2, \ldots, vm_n\}$. The specifications of each VM are represented as follows:

$$core_i(j) = \begin{cases} c_{ij}, & when\ vm_i\ is\ mapped\ to\ pm_j \\ 0, & otherwise \end{cases}$$

$$RAM_i(j) = \begin{cases} r_{ij}, & when\ vm_i\ is\ mapped\ to\ pm_j \\ 0, & otherwise \end{cases}$$

$$storage_i(j) = \begin{cases} s_{ij}, & when\ vm_i\ is\ mapped\ to\ pm_j \\ 0, & otherwise \end{cases}$$

$$bandwidth_i(j) = \begin{cases} b_{ij}\ ,\ when\ vm_i\ is\ mapped\ to\ pm_j \\ 0\ ,\ otherwise \end{cases}$$

where the vector (c_{ij}, r_{ij}, s_{ij}, b_{ij}) represents the required number of cores, the amount of RAM, the amount of disk storage, and the amount of bandwidth for a VM_i on PM_j.

So, the requirements of an instance VM_i on an infrastructure physical machine PM_j can be represented as a vector shown in Eq. (9):

$$VM_i^{Req}(j) = \begin{pmatrix} c_{ij} \\ r_{ij} \\ s_{ij} \\ b_{ij} \end{pmatrix} \tag{9}$$

5.4.3. Virtual machine placement problem

According to the above PMs and VMs representations, the problem now is how to perform the VMs placement process for these two scenarios:

1) The demanded VMs allocated to the jobs submitted by cloud users. The VM allocation is a problem for the admission of new requests for VMs provisioned to be placed later on PMs.
2) The VMs to be migrated during execution from the PMs with under/overload utilization.

The problem of VM placement resulting from the above two cases is represented and solved based on the MCKP Method as discussed in details in the next sections.

5.5. The knapsack problem

There are different variants of the KP, which all fall under the NP-hard class of problems [47]. The knapsack problem is the one that selects a subset of n items. The selection is done in such a way that the corresponding profit sum is maximized, while their weight sum of the selected items does not exceed the capacity of a knapsack.

The problem of knapsack can be deployed to serve the problem of VM placement, and it is possible to use the fundamentals of the knapsack problem to deal with VM placement.

Each PM is considered as a knapsack, and each VM is considered as an item. The capacity of a knapsack consists of the available cores, RAM, storage, and bandwidth of each PM. Each item has profit value and weight.

Eqs. (8) and (9) described above can explain the capacity of the knapsack (PM capacity), and the weight of the items (requirement of the VMs) respectively.

5.6. The standard knapsack problem

The Knapsack problem can be described as:

Given a set of items to be placed on a limited capacity Knapsack. Each item has a profit value and a weight. The problem is to choose specific items such that the sum is maximized/minimized for specific value, without having the weight sum to exceed the capacity of the knapsack.

The standard KP can be formulated as [47]:

$$Max/Min \sum_{i=1}^{n} v_i\, x_i \tag{10}$$

Subject to

$$\sum_{i=1}^{n} r_i\, x_i \ \le\ KnapsackCapacity \tag{11}$$

$$i = 1,2,\ ...,n$$

$$x_i \in \{0,1\},$$

Eqs. (10) and (11) represent the objective and constraint functions respectively. In KP, all coefficients v, r, KnapsackCapacity are positive, and their details are as below:

- v_i represents a value associated with each item (i.e. the values of the items to be maximized/minimized).
- r_i represents the required capacity by the item
- $x_i = \begin{cases} 1\ if\ the\ item\ placed\ on\ the\ Knapsak \\ 0\ otherwise \end{cases}$

So, to solve the problem of after migration VM placement, where VMs are mapped to their best PMs under the condition of PMs limited capacity, Eqs. (12) and (13) are proposed to minimize the energy consumption:

$$Min \sum_{i=1}^{n} v_{i,j}\, x_{i,j} \tag{12}$$

Subject to

$$\sum_{i=1}^{n} VM_i^{Req}\, x_{i,j} \le\ PM_j^{Cap} \tag{13}$$

$$i = 1,2,\ ...,n$$

$$x_{i,j} = \begin{cases} 1\ if\ VM_i\ placed\ on\ the\ PM_j \\ 0\ otherwise \end{cases}$$

$$\textstyle\sum_{\forall i,j} x_{i,j} = 1$$

$$VM_i^{Mig_{time}} + Makespan\ \le\ DL_i$$

where:

- $VM_i^{Mig.time}$ is the time duration of migrating the VM allocated to job_i
- DL_i *is the deadline of* job_i *which is served by* VM_i
- $v_{i,j}$ is the amount of energy consumed when executing VM_i on PM_j. It can be estimated as in the model described in [48].

The constraint function in (13) means that the following conditions must be satisfied for all packed items:

$$core_i(j) \le capacity_{core}(j)$$

$$RAM_i(j) \le capacity_{RAM}(j)$$

$$storage_i(j) \le capacity_{storage}(j)$$

$$bandwidth_i(j) \le capacity_{bandwidth}(j)$$

Notably, the capacity of each resource can be the *total capacity* if the PM does not host any VM, or the remaining *available capacity* if the PM hosts one VM or more.

The remaining *available capacity* means that capacity of every PM which is described in Eq. (8) is modified after every VM placement process. The description of the modification is as below:

For any PM_j, the new available capacity of PM_j after placing VM_i on it is described as:

$$PM_j^{Cap.Available} = \begin{pmatrix} Capacity_{core}(j) - c_{ij} \\ Capacity_{RAM}(j) - r_{ij} \\ Capacity_{storage}(j) - s_{ij} \\ Capacity_{bandwidth}(j) - b_{ij} \end{pmatrix} \tag{14}$$

where $PM_j^{Cap.Available}$ is the resulting available capacity of PM_j after

placing the VM_i on it in the VM placement process.

It is obvious that the VM placement process in this study requires more than a single constraint in the selection and packing. The considered constraints are cores, RAM, storage, and bandwidth. This generalization of (KP) is called multidimensional KP, or d-dimensional KP (d-KP), which is represented as below [49]:

$$Min \sum_{i=1}^{n} v_i x_i \tag{15}$$

Subject to

$$\sum_{i=1}^{n} r_{id} x_i \leq KnapsackCapacity \tag{16}$$

$d = 1,2, \ldots, d$

$i = 1,2, \ldots, n$

$x_i \in \{0,1\}$

In our case $d = 4$, which represents the 4-tuple of resources requirement requested by each VM. The required resources are: (1) CPU Cores, (2) RAM, (3) Storage, and (4) BW.

5.7. The multiple choice knapsack problem

MCKP is a generalization of the classical KP. This generalization can be described as:

Given T classes ($S_1, S_2, \ldots, S_T$) of items to be placed on a limited capacity knapsack. Each item i $\in S_t$ has a profit value and a weight. The problem is to choose an item from each class such that the sum of a specific value is maximized/minimized, without having the weight sum to exceed the capacity of the knapsack.

MCKP can be formulated as [47,49]:

$$Max/Min \sum_{t=1}^{T} \sum_{i \in S_t} v_{ti} x_{ti} \tag{17}$$

Subject to

$$\sum_{t=1}^{T} \sum_{i \in S_t} r_{ti} x_{ti} \leq KnapsackCapacity \tag{18}$$

$i = 1, \ldots, n$

$i \in S_t$

$x_{ti} \in \{0,1\}$

Eqs. (17) and (18) represent the objective and constraint functions respectively. In MCKP, all coefficients v, r, and $KnapsackCapacity$ are positive, and their details are listed below:

- v_{ti} is the value associated with each i_{th} item belonging to the t_{th} class when it is placed on the knapsack (i.e. the values of the items to be maximized/minimized).
- r_{ti} represents the required capacity by the item
- $x_{ti} = \begin{cases} 1 \text{ if the item placed on the Knapsak} \\ 0 \text{ otherwise} \end{cases}$
- Subsets or Classes ($S_1, S_2, \ldots, S_T$) are mutually disjoint. Each class S_t has s_t items. The total number of items to be knapsacked (n) is equal to the sum the number of classes' items, $n = \sum_{t=1}^{T} s_t$.

Generally, in cloud data centers, VMs are owned by independent individuals or enterprises. This implies that the resulting workload is of mixed types of jobs. The mixed workload is formed by combining various types of applications, such as high-performance computing applications, and web-applications. These applications require and utilize the resources of the cloud data centers at the same time. Users establish SLAs with cloud provider to formalize and meet the QoS requirements when submitting their jobs. Compute Intensive (CI) jobs can be combined with the Data Intensive (DI) jobs, as the former mostly utilizes the compute resources, whereas the latter utilizes the non-compute resources (i.e. storage and bandwidth). This results in better PMs utilization, as all PM resources are keeping busy while servicing the user's jobs.

This paper deploys MCKP to be in line with the proposed VM placement strategy. The proposed VM placement strategy, called Mixed Type Placement (MTP) strategy, combines the VMs allocated to different types of jobs on the same PM whenever possible.

Upon their resources requirements, users' jobs can be divided into four subsets (classes) as explained in [50,51]:

- Type 1: *CI* jobs, they are the set of jobs, which highly utilize compute resources.
- Type 2: *DI* jobs, they are the set of jobs, which highly utilize storage and/or bandwidth resources.
- Type 3: *CIDI* jobs, they are the set of jobs, which utilize compute resources together with storage and/or bandwidth resources all in a high manner.
- Type 4: *NORMAL* jobs, they are the set of jobs that do not fit in any of the types specified above.

After classifying the jobs, a VM is allocated to each job. Then, VMs allocated to jobs from type 1, type 2, type 3, and type 4 are gathered in four subsets S_1, S_2, S_3 and S_4 respectively.

In the initial VM placement, MCKP is applied to the subset (S_1 and S_2), and to the subset (S_3 and S_4), to place their items (VMs), on the same knapsack (PM) wherever possible. The remaining VMs, which do not find their corresponding VMs based on MTP strategy, are combined in one set. Then, KP is applied to the items of this set to be placed on the Knapsack. As a result of such placement process, the minimum number of switched-on PMs that consume the minimum energy, and do the users' jobs, is guaranteed. At the same time, the involved PMs are used optimally, because they host VMs which request different kinds of resources. In such case, the PMs resources are kept busy and consequently the PMs utilization is maximized. The static VM placement can be illustrated in Fig. 4.

In Fig. 4, the cloud users submit their jobs (say n jobs) to the cloud provider which in turn classifies the jobs into four subsets: S_1, S_2, S_3 and S_4 of $n1$, $n2$, $n3$ and $n4$ elements respectively, such that: ($n1 + n2 + n3 + n4 = n$). After the classification process, the cloud provider allocates the proper VMs to the submitted jobs, and then, employs the MTP strategy in the VM placement process.

In the after migration VM placement, MCKP is applied to the subset (S_1 and S_2), and to the subset (S_3 and S_4), to place their items (VMs) on the same knapsack (PM) wherever possible. If the VMs do not find their corresponding VMs based on MTP strategy, the classical KP is applied to the items of this set to be placed on the Knapsack. There are many scenarios to employ the algorithms of the proposed energy efficient dynamic VM consolidation approach.

Figs. 5 and 6 explain two scenarios. In Fig. 5, the energy efficient dynamic VM consolidation approach is employed to keep the involved PMs efficiently utilized, while in Fig. 6, the approach is employed to switch off unnecessary PMs hosted in the data center.

The process of the proposed VM placement strategy is similar to MCKP. In MCKP, the set of items is classified and then, instead of taking two or more items from the same set, one item is selected from each class to be packed in the knapsack [49]. This suits both static and dynamic VM placement.

The MCKP can be rewritten as given below to represent the static (initial) VM placement:

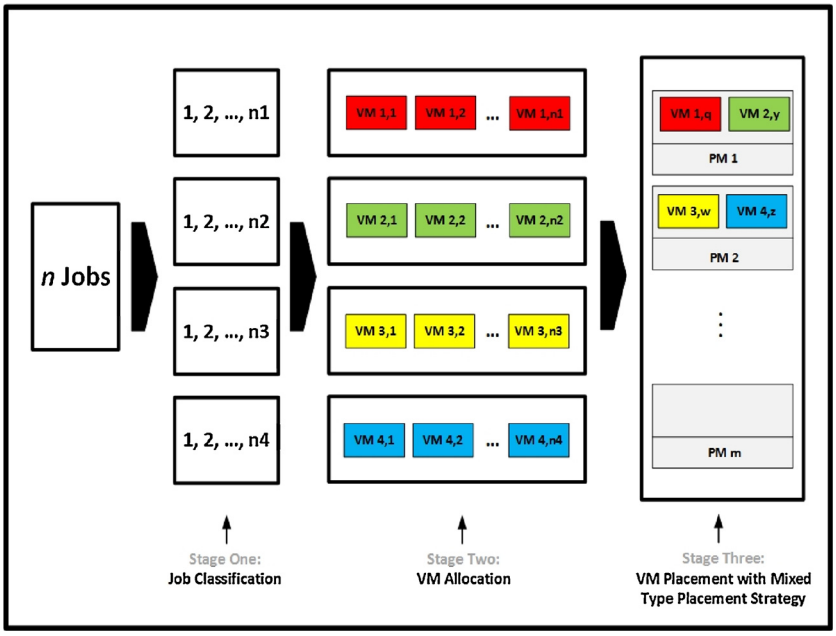

Fig. 4. Static VM Placement based on MTP strategy.

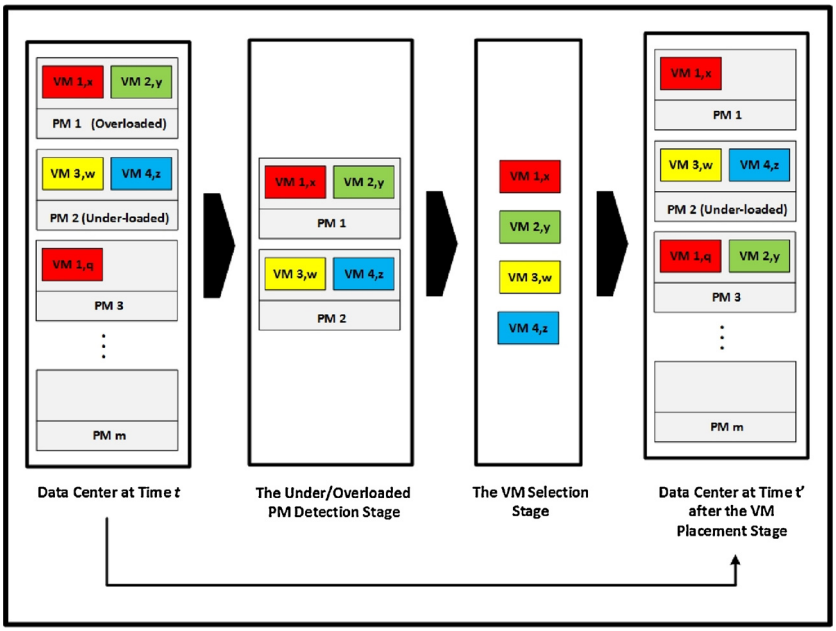

Fig. 5. Energy Efficient Dynamic VM consolidation to keep the active PMs efficiently utilized in the data center.

For every PM_j:

Place a set of VMs that consider the capacity of PM_j , *such that*:

$$Min \sum_{t=1}^{T} \sum_{i \in S_t} v_{ti,j}\, x_{ti,j} \tag{19}$$

Subject to

$$\sum_{t=1}^{T} \sum_{i \in S_t} VM_{ti}^{Req}\, x_{ti,j} \leq PM_j^{Cap} \tag{20}$$

$i = 1, \ldots, n$

$i \in S_t$

$$x_{ti,j} = \begin{cases} 1 \text{ if } VM_i \in S_t \text{ placed on the } PM_j \\ 0 \text{ otherwise} \end{cases}$$

$$\sum_{\forall ti,j} x_{ti,j} = 1$$

$Makespan \leq DL_{ti}$

Also, the MCKP can be rewritten as given below to represent the dynamic (after migration) VM placement:

On PM_j:

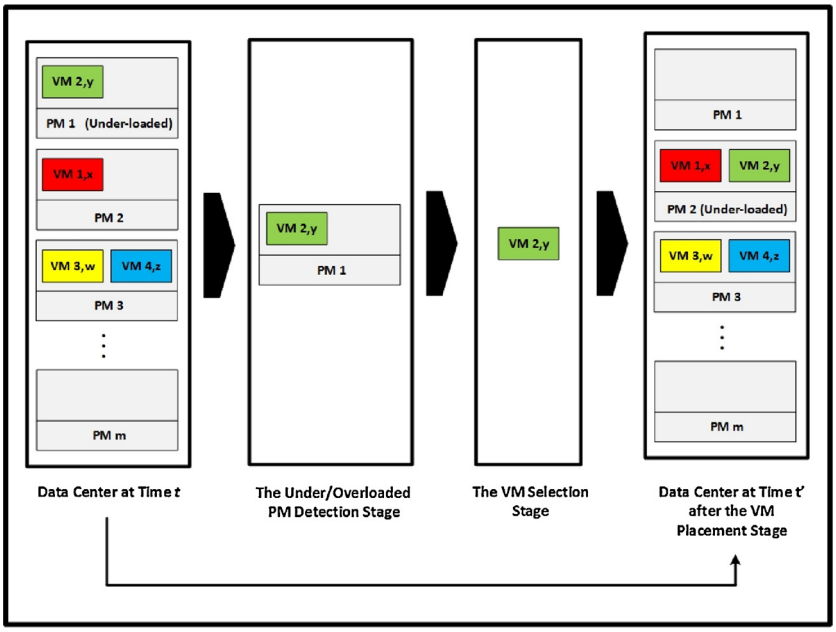

Fig. 6. Energy Efficient Dynamic VM consolidation to switch off unnecessary PMs hosted in the data center.

Place the VM_{ti} selected to be migrated, such that:

$$Min \sum_{t=1}^{T} \sum_{i \in S_t} v_{ti,j} x_{ti,j} + v_{\bar{t}\bar{i},j} \tag{21}$$

Subject to

$$\sum_{t=1}^{T} \sum_{i \in S_t} VM_{ti}^{Req} x_{ti,j} \leq PM_j^{Cap} \tag{22}$$

$i = 1, \ldots, n$

$i \in S_t$

$$x_{ti,j} = \begin{cases} 1 \text{ if } VM_i \in S_t \text{ placed on the } PM_j \\ 0 \text{ otherwise} \end{cases}$$

$$\sum_{\forall ti,j} x_{ti,j} = 1$$

$$VM_{ti}^{Mig_{time}} + Makespan \leq DL_{ti}$$

where:

- $VM_{ti}^{Mig.time}$ is the time duration of migrating the $VM_i \in S_t$
- DL_{ti} is the deadline of job_i which is served by $VM_i \in S_t$
- $v_{ti,j}$ is the value of the i_{th} item of the t_{th} class when it is knapsaked on PM_j. In our case, it is the consumed energy when placing $VM_i \in$ Class t on PM_j which host $VM_{\bar{i}} \in$ Class $\bar{t}$. While $v_{\bar{t}\bar{i},j}$ is the consumed energy when placing $VM_{\bar{i}} \in$ Class $\bar{t}$ on PM_j. The values ($v_{ti,j}$ and $v_{\bar{t}\bar{i},j}$) can be estimated as in the model described in [48].

VM_{ti} is the migrated $VM \in$ Class t, $VM_{\bar{t}\bar{i}}$ is the $VM \in$ Class $\bar{t}$ which is already hosted on the PM which is selected to host the migrated VM. Class t is the opposite of class $\bar{t}$. The opposite Classes in this work are:

VMs allocated to CI jobs are the opposites of VMs allocated to DI jobs, while VMs allocated to CIDI jobs are the opposites of VMs allocated to Normal jobs.

The constraint function (5.20) means that the following conditions must be satisfied for all migrated items:

$$core_{ti}(j) \leq capacity_{core}(j)$$

$$RAM_{ti}(j) \leq capacity_{RAM}(j)$$

$$storage_{ti}(j) \leq capacity_{storage}(j)$$

$$bandwidth_{ti}(j) \leq capacity_{bandwidth}(j)$$

Notably, the capacity of each resource can be the *total capacity* if the PM does not host any VM, or the remaining *available capacity* if the PM hosts one VM or more.

The remaining *available capacity* means that capacity of every PM which is described in Eq. 8 is modified after every VM placement process. The description of the modification can be described as follows:

For any PM_j, the new available capacity of PM_j after placing VM_i on it is described as:

$$PM_j^{Cap.Available} = \begin{pmatrix} Capacity_{core}(j) - c_{ij} \\ Capacity_{RAM}(j) - r_{ij} \\ Capacity_{storage}(j) - s_{ij} \\ Capacity_{bandwidth}(j) - b_{ij} \end{pmatrix} \tag{23}$$

where $PM_j^{Cap.Available}$ is the resulted available capacity of PM_j after placing the VM_i on it in the VM placement process.

Each item from the four subsets has a particular value and it requires some resources. The objective of the MCKP is to pick only one item (VM) from each subset and combined the picked items together on the same knapsack (PM) to minimize the total value of the picked combined items, subject to the resource constraints (capacity) of the PM.

The indexes in the proposed formulation can be separated into two parts:

- Part related to VMs (represented by variables t and i)
- Part related to PMs (represented by variable j).

The part that related to VMs can be further separated into two parts:

- Part related to the jobs types (represented by variable t)
- Part related to the index of the item (VM) in the jobs types subset (represented by variable i)

So, variable *t* is responsible for ensuring placing VMs allocated to different job types on the same PM.

Algorithms 5 and 7 are the proposed static and dynamic VM placement respectively.

Algorithm 5: Static VM placement based on MPCM.

Input	*Set of n VMs (VMs ={VM_1,VM_2,…,VM_n}), set of the available PMs (PMs={PM_1,PM_2,…PM_m})*
Output	*VM Placement*
1	***sort*** *PMs based on capacity / availability in descending order*
2	*j=1*
3	***repeat*** *for all VMs*
4	***pick*** PM_j
5	***select*** *a* VM_{CI} *and a* VM_{DI} *which results in minimum energy on* PM_j
6	***if*** *(* VM_{CI} *requirement +* VM_{DI} *requirement) <= PM_j available capacity* ***then***
7	***place*** *the selected* VM_{CI} *and* VM_{DI} ***on*** PM_j
8	*VMs = VMs – selected placed* VM_{CI} *and* VM_{DI}
9	***select*** *a* VM_{CIDI} *and a* VM_{NORMAL} *which results in minimum energy on* PM_j
10	***if*** *(* VM_{CIDI} *requirement +* VM_{NORMAL} *requirement) <= PM_j available capacity* ***then***
11	***place*** *the selected* VM_{CIDI} *and* VM_{NORMAL} ***on*** PM_j
12	*VMs = VMs – selected placed* VM_{CIDI} *and* VM_{NORMAL}
13	*j=j+1*
14	***until*** *no more VMs* ***or*** *no VMs to be placed together*
15	***if*** *there are remaining VMs which are not placed on any PM* ***then***
16	*Add the remaining VMs to* KP_{list}
17	*KPvmPLACEMENT(KP_{list},PMs)*
18	***return*** *VMs to PMs placement result*

Algorithm 5 performs the process of VM to PM placement process based on MCKP. In step 1, the algorithm sorts the PMs of the data center in decreasing order based on their capacity. Step 2 gives an index to the first available PM which is the one of the maximum capacity (index j is for the PMs). Steps 3–14 try to combine VMs allocated to CI jobs with VMs allocated to DI jobs, and VMs allocated to CIDI jobs with VMs allocated to Normal jobs on the same PM. Step 15 checks if there are more VMs which are not placed on PMs. If there are remaining VMs, step 17 combines them in one list. This list results in unequal distribution between CI and DI jobs, and between CIDI and Normal jobs. Step 17 sends the list of remaining VMs to be placed based on classical KP using Algorithm 6. Step 18 returns the results of the VM placement process.

The time complexity of Algorithm 5 is $O(n + m \log m)$, where n is the number of VMs and m is the number of PMs which will host the VMs. This complexity is obtained as the PMs sorting process in step 1 costs $O(m \log m)$, and the loop starting in step 3 iterates n times (n is the number of VMs) and, hence, costs $O(n)$. Since the PM sorting process and the loop are sequential, their total running time is the summation of the individual running times, that is: $O(n + m \log m)$.

Algorithm 6 performs the process of VM to PM placement based on classical KP. In step 1, the algorithm sorts the PMs of the data center in decreasing order based on their capacity. Step 2 gives an index to the first available PM which is the one of the maximum capacity (index j is for the PMs). Steps 3–9 try to pack a selected VM on a picked PM, the process is repeated until no more VMs remain. Step 10 returns the results of the VM placement process.

Algorithm 6: Virtual Machine Placement Algorithm Based on KP

Input	*Set of VMs (VMs ={VM_1,VM_2,…VM_n}) , set of the available PMs (PMs={PM_1,PM_2,…PM_m})*
Output	*VM placement*
1	***sort*** *PMs based on capacity / availability in descending order*
2	*j=1*
3	***repeat*** *for all VMs*
4	***pick*** PM_j
5	***select a*** *VM which results minimum energy on* PM_j
6	***if*** *the requirement of the selected* $VM_{(s)}$ *<= PM_j available capacity* ***then***
7	***place*** $VM_{(s)}$ ***on*** PM_j
8	*j=j+1*
9	***until*** *no more VMs*
10	***return*** *VMs to PMs placement result*

The time complexity of Algorithm 6 is $O(n + m \log m)$ as well, where n is the number of VMs and m is the number of PMs which will host the VMs. This complexity is obtained from two main components: (1) the PMs sorting process in step 1 which costs $O(m \log m)$, and the loop starting in step 3 which iterates n times (depending on the number of VMs) and hence costs $O(n)$. The two components are independent and thus their total running time is the summation of the respective individual running times: $O(m \log m) + O(n) = O(n + m \log m)$.

Notably, Algorithms 5 and 6 are based on the greedy algorithm concept, which sort the containers in decreasing order based on their capacity before performing the packing process. Then, they pick one by one PM to pack the VMs on it based on MCKP and KP respectively.

Algorithm 7: Dynamic VM placement Algorithm Based on MPCM.

Input:	*The selected VMs to be migrated (vmLIST) , The available PMs (pmLIST)*
Output	*VM placement for VMs in vmLIST*
1	***sort*** *pmLIST based on PMs utilization in ascending order*
2	*j = 1*
3	***for all*** *VMs* ***in*** *in vmLIST*
4	***check*** *the selected VM (*VM_s*)*
5	***case 1****: if* VM_s *is allocated to CI job*
6	***repeat***
7	***pick*** PM_j
8	***if*** *(*PM_j *hosts any VM allocated to DI job)* ***and*** *(*VM_s *requirement <= PM_j available capacity)* ***then***
9	***place*** VM_s *on* PM_j
10	***else***
11	*j = j+1*
12	***until*** VM_s *is placed on a PM*
13	***case 2****: if* VM_s *is allocated to DI job*
14	***repeat***
15	***pick*** PM_j
16	***if*** *(*PM_j *hosts any VM allocated to CI job)* ***and*** *(*VM_s *requirement <= PM_j available capacity)* ***then***
17	***place*** VM_s *on* PM_j
18	***else***
19	*j = j+1*
20	***until*** VM_s *is placed on a PM*
21	***case 3****: if* VM_s *is allocated to CIDI job*
22	***repeat***
23	***pick*** PM_j
24	***if*** *(*PM_j *hosts any VM allocated to Normal job)* ***and*** *(*VM_s *requirement <= PM_j available capacity)* ***then***
25	***place*** VM_s *on* PM_j
26	***else***
27	*j = j+1*
28	***until*** VM_s *is placed on a PM*
29	***case 4****: if* VM_s *is allocated to Normal job*
30	***repeat***
31	***pick*** PM_j
32	***if*** *(*PM_j *hosts any VM allocated to Normal job)* ***and*** *(*VM_s *requirement <= PM_j available capacity)* ***then***
33	***place*** VM_s *on* PM_j
34	***else***
35	*j = j+1*
36	***until*** VM_s *is placed on a PM*
37	***return*** *the VM placement result for vmLIST*

Table 1
VM instance types in M3 family offered by Amazon.

VM Type	CPU	Clock	vCPU	Memory	BW
M3 medium	Intel Xeon E5-2670 v2 Processors	2500	1	3750	Moderate
M3 large	Intel Xeon E5-2670 v2 Processors	2500	2	7500	Moderate
M3 xlarge	Intel Xeon E5-2670 v2 Processors	2500	3	15000	High
M3 2xlarge	Intel Xeon E5-2670 v2 Processors	2500	4	30000	High

In Algorithm 7, step 1 sorts the PMs in ascending order based on PMs utilization. This way of sorting increases the possibility of placing the VMs selected to be migrated to be placed on the PMs of minimum utilization. Placing a VM on an under-utilized PM increases the PM utilization which, in turn, enhances its energy efficiency. In other words, the sorting based on PMs utilization gives a better results from the energy efficiency perspective rather picking a random PM because the sorting process advantages in choosing the most energy-efficient PM first. Step 2 gives an index to the PMs. Steps 3 and 4 select a VM from the VMs to be migrated list. Then, the algorithm starts to check the job type which the selected VM is allocated to it. In step 5, if the selected VM is allocated to a CI job, then the algorithm moves to steps 6 to 12 to allocate the selected VM on a PM with the minimum utilization and hosts a VM allocated to a DI job at the same time, under considering the availability/capacity condition.

In step 13, if the selected VM is allocated to a DI job, then the algorithm moves to steps 14–20 to allocate the selected VM on a PM with the minimum utilization and hosts a VM allocated to a CI job at the same time, under considering the availability/capacity condition.

In step 21, if the selected VM is allocated to a CIDI job, then the algorithm moves to steps 22–28 to allocate the selected VM on a PM with the minimum utilization and hosting a VM allocated to a Normal job at the same time, considering the availability/capacity condition.

In step 29, if the selected VM is allocated to a Normal job, then the algorithm moves to steps 30–36 to allocate the selected VM on a PM with the minimum utilization and hosting a VM allocated to a CIDI job at the same time, considering the availability/capacity condition.

Analogous to the complexity analysis of Algorithms 5 and 6, the time complexity of Algorithm 7 is $O(n + m \log m)$, where n is the number of VMs and m is the number of PMs which will host the VMs. This complexity is obtained as the PMs sorting process in step 1 costs $O(m \log m)$, and the loop starting in step 3 costs $O(n)$.

6. Performance evaluation

The results in this section were obtained using CloudSim simulator in different cloud computing environment configurations. The PMs and VMs configurations are as those provided by Amazon cloud data centers. The VMs instance types, called M3 family, offered by Amazon and used in the experiments in this work are listed in Table 1.

CloudSim simulator is a powerful simulation platforms for cloud computing. To get more accurate estimation of power consumption in cloud computing environments, some works, as the one presented in [52], extended this simulator with a multi-resource scheduling and power consumption model.

The proposed algorithms and policies are applied using real workload traces from Google. The workload consists data set of different types of jobs. More details about this data are available in [53]. Using random data from this workload, the proposed PM under/overload detection algorithms, VM selection algorithms, and VM placement strategy, are simulated.

To evaluate algorithms and models proposed in this work, four sets

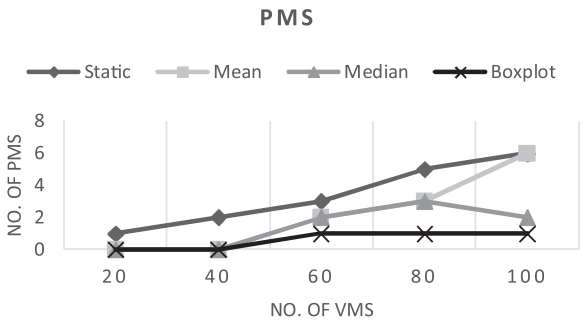

Fig. 7. The results of applying under-load detection algorithms on the number of PMs by involving part from the PMs of the data center in calculating the under-load threshold.

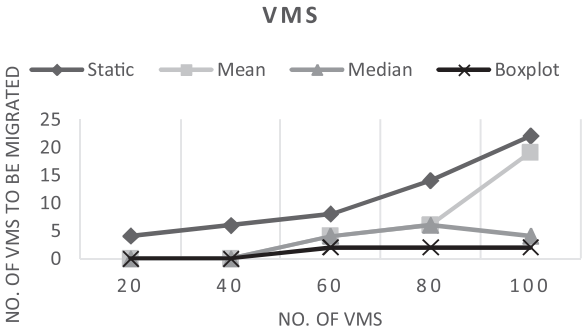

Fig. 8. The results of applying under-load detection algorithms on the number of VMs to be migrated by involving part from the PMs of the data center in calculating the under-load threshold.

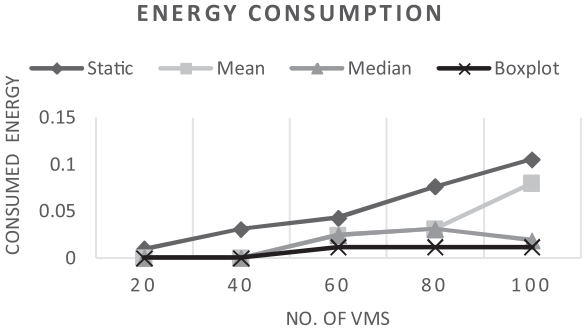

Fig. 9. The effects of applying the under-load detection algorithms on the consumed energy by involving part from the PMs of the data center in calculating the under-load threshold.

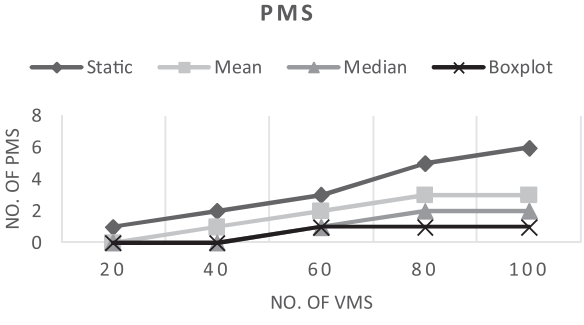

Fig. 10. The results of applying under-load detection algorithms on the number of PMs by involving all the PMs of the data center in calculating the under-load threshold.

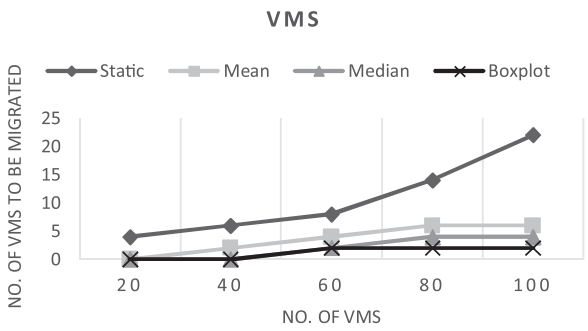

Fig. 11. The results of applying under-load detection algorithms on the number of VMs to be migrated energy by involving all the PMs of the data center in calculating the under-load threshold.

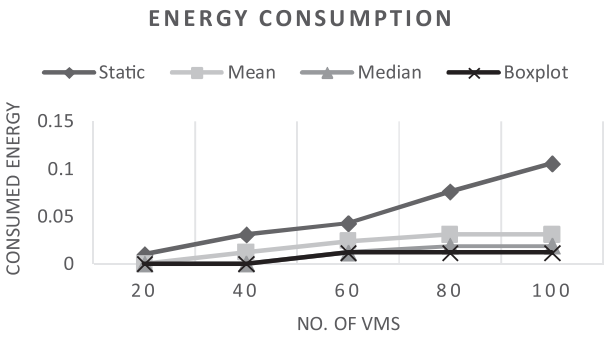

Fig. 12. The effects of applying the under-load detection algorithms on the consumed energy by involving all the PMs of the data center in calculating the under-load threshold.

of experiments are done using the real workload data described in the previous section.

The first set of experiments was done to evaluate the performance the under-load detection algorithms. The experiments are repeated for a fixed number of 25 PMs, and a various number of VMs, from 20 to 100. The static under-load threshold value is set to 30%. Two directions to calculate the dynamic threshold are used. The first direction considers only the PMs which host a specific number of VMs in calculating the under-load threshold. The second direction considers all PMs of the data center in calculating the under-load threshold.

The results of the first direction are summarized in Figs. 7–9. The figures represent the effects of applying the proposed four algorithms (Static thresholding, dynamic mean thresholding, dynamic median thresholding, and boxplot) on the number of under-loaded PMs in each algorithm, the number of VMs to be migrated which are hosted on the resulting under-loaded PMs, and the approximate energy consumption corresponding to each algorithm respectively.

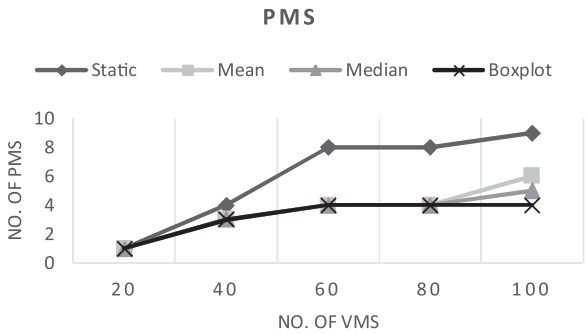

Fig. 13. The results of applying overload detection algorithms on the number of PMs by involving part from the PMs of the data center in calculating the overload threshold.

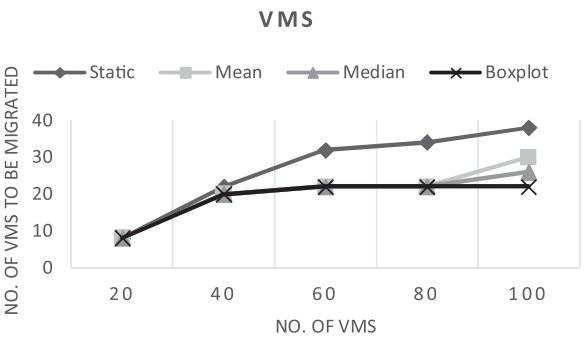

Fig. 14. The results of applying overload detection algorithms on the number of VMs to be migrated by involving part from the PMs of the data center in calculating the overload threshold.

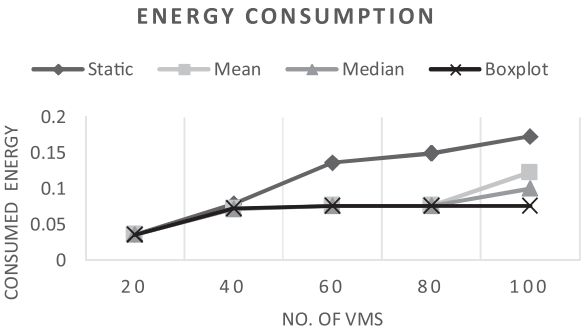

Fig. 15. The effects of applying overload detection algorithms on the consumed energy by involving part from the PMs of the data center in calculating the overload threshold.

The results of the second direction are summarized in Figs. 10–12. The figures represent the effects of applying the proposed four algorithms (Static thresholding, dynamic mean thresholding, dynamic median thresholding, and boxplot) on the number of under-loaded PMs in each algorithm, the number of VMs to be migrated which are hosted on the resulted under-loaded PMs, and the approximate energy consumption corresponding to each algorithm respectively.

By studying Figs. 7–12 the following observations can be concluded:

1) The dynamic under-load detection methods outperform the static method. Static methods are inefficient to be applied when the system workload is unknown or when it can vary over time, which is the norm in cloud computing environment.
2) The second direction of the experiments outperforms the first direction because it reflects more accurate approximations about the system conditions, including the under-load thresholds.

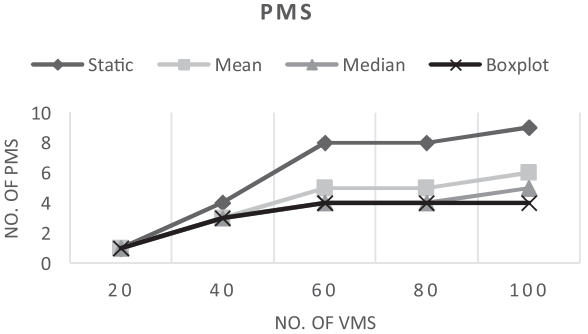

Fig. 16. The results of applying overload detection algorithms on the number of PMs by involving all the PMs of the data center in calculating the overload threshold.

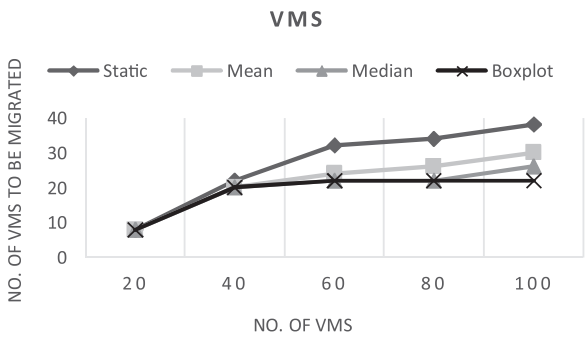

Fig. 17. The results of applying overload detection algorithms on the number of VMs to be migrated by involving all the PMs of the data center in calculating the overload threshold.

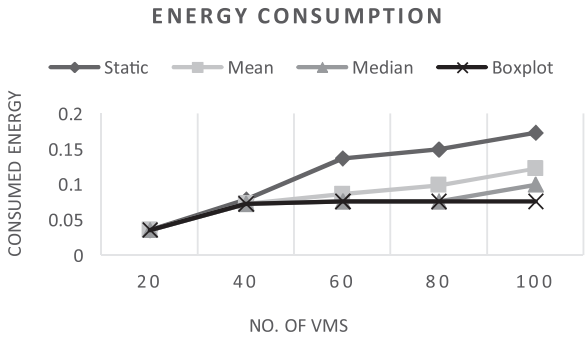

Fig. 18. The effects of applying overload detection algorithms on the consumed energy by involving all the PMs of the data center in calculating the overload threshold.

The second set of experiments was done to evaluate the performance of the host overload detection algorithms. The experiments are repeated for a fixed number of PMs equals to 25, and a various number of VMs, from 20 to 100. The static threshold value is set to 80%. In this set of experiments, two directions were used to calculate the dynamic threshold. The first direction involves only the PMs which host a specific number of VMs in calculating the overload threshold. The second direction involves all PMs of the data center in calculating the overload threshold.

The results of the first direction are summarized in Fig. 13–15 which represent the effects of applying the proposed four algorithms (Static thresholding, dynamic mean thresholding, dynamic median thresholding, and boxplot) on the number of overloaded PMs, the number of VMs to be migrated which are hosted on the resulted overloaded PMs, and the approximate energy consumption corresponding to each algorithm respectively.

The results of the second direction are summarized in Figs. 16–18 which show the effects of applying the proposed four algorithms (Static thresholding, dynamic mean thresholding, dynamic median thresholding, and boxplot) on the number of under-loaded PMs, the number of VMs to be migrated which are hosted on the resulted overloaded PMs, and the approximate energy consumption corresponding to each algorithm respectively.

By studying Figs. 13–18, the following observations can be concluded:

1) The dynamic overload detection methods outperform the static method. Static methods are inefficient to be applied when the system workload is unknown or when it can vary over time, which is the norm in a cloud computing environment.
2) The second direction of the experiments outperforms the first direction because it reflects more accurate approximations about the system conditions, including the overload thresholds.

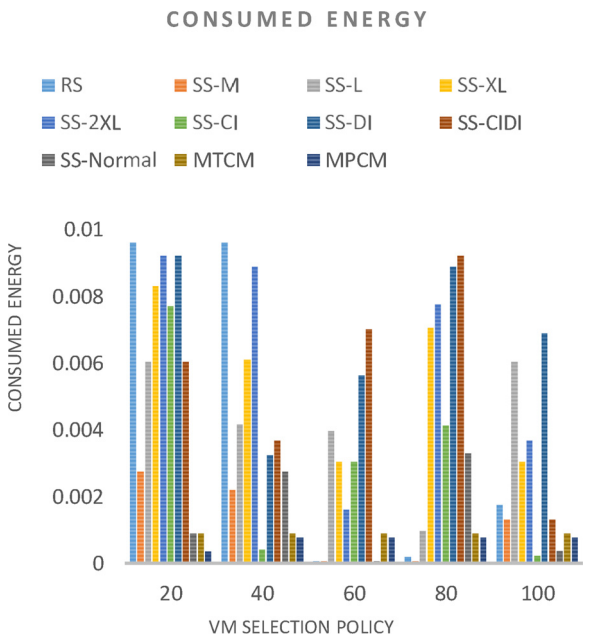

Fig. 19. The energy consumption resulted from a selected migrated VM based on different VM selection algorithms.

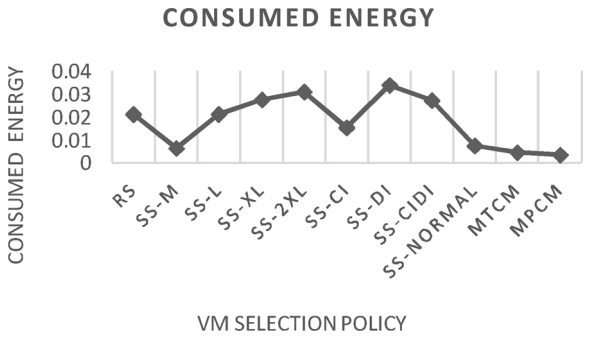

Fig. 20. The average energy consumption resulted from a selected migrated VM based on different VM selection algorithms.

Notably, the optimal/near optimal initial VM placement process results in less number of under/overloaded PMs, and as a result, less number of VMs to be migrated (the total consumed energy is reduced). The first and second sets of experiments show how it is essential to place VMs on PMs conditionally since the placement process affects the PMs utilizations.

The VM placement strategy has direct and indirect effects on the energy efficiency:

- *The direct effect* results in an initial satisfactory percentage of utilization of the PMs involved when using proposed the VM placement strategy. There is a strong relation between the utilization and energy efficiency. The better the PMs are utilized, the better is the energy efficiency obtained.
- *The indirect effect* results in saving the energy consumed from the process of migrating the VMs from the under/over utilized PMs to another hosts. VMs migrations consume energy. The initial placement results in a distribution of VMs on PMs which minimizes the subsequent migration of VMs thereby saving energy which would have been consumed by migration.

The third set of experiments was done to evaluate the performance of the algorithms which select the VM to be migrated. The VM to be migrated is selected from the first and second sets of algorithms which

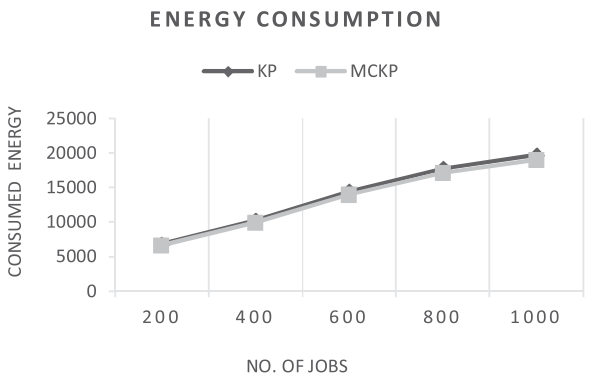

Fig. 21. The consumed energy when applying KP and MCKP in VM placement process.

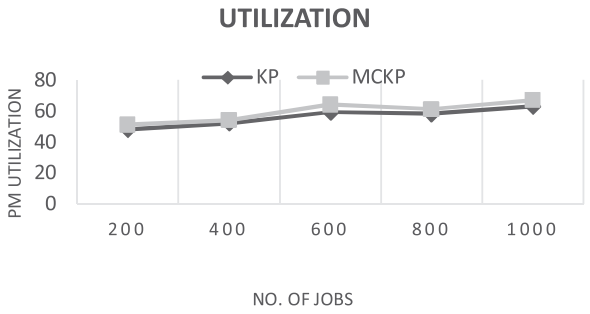

Fig. 22. The involved PMs utilization when applying KP and MCKP in the VM placement process.

specify the VMs hosted on the PMs in under/overload state. The experiments are repeated for 20 to 100 VMs. By repeating the experiments five times using different workloads, the amount of the consumed energy for each policy is illustrated in Fig. 19.

Fig. 20 illustrates the average amount of the consumed energy resulted from the five experiments for each policy.

From the energy perspective, it is clearly seen from Fig. 20 that the best VM selection algorithms is MPCM. However, MPCM algorithm may affect system performance since it requires some calculations to estimate the consumed energy resulted from the migration process.

The MTCM algorithm gives good results. The shortest time VM migration resulted when the memory content is the smallest, and the smallest memory content results in less energy. However, MTCM also may affect the system performance as it requires some calculations to estimate the time resulted from the migration process.

In the SS algorithm, the less consumed energy amount resulted when selecting both the VM of *medium* instance type configuration, and the VM that is allocated to Normal jobs.

The VM of *medium* instance type configuration is the one with the smallest memory size compared with the other available configurations (as shown in Table 1). As a result, the memory content is less compared with the other VM configurations, which consequently, results in minimum energy in VM migration process.

In general, the VMs which are allocated to Normal jobs contain less memory contents, and therefore results in less energy consumption in the VM migration process. Normal jobs do not heavily utilize memory and compute resources, and at the same time, do not utilize the bandwidth, as explained in [50] and [51].

The fourth set of experiments was done to evaluate the energy efficiency relation to the types of the jobs which are served by the VMs in the process of VM placement. Both the classical KP and the MCKP are applied to solve the problem of the initial VM placement. The experiments are repeated for a different number of VMs allocated to a number of jobs, from 200 to 1000 jobs.

Fig. 21 shows the amount of energy consumed to do a number of users' jobs using classical KP, and the proposed MCKP, which considers the jobs' types as part of VM placement strategy. Fig. 22 shows the PMs utilization when applying MCKP and KP.

The results in Figs. 21 and 22 show that applying the MCKP is more energy-efficient than the classical KP. This is because, when applying MCKP, the resources of the VMs are utilized more optimally compared to when KP is applied. By applying MCKP, the resources of the PM are keeping busy during when the PM is involved in serving the users' jobs.

In the initial VM placement, the proposed model enhances the energy efficiency by trying to minimize the total number of the involved PMs which served the users' jobs, and maximized the utilization of the involved PMs. The minimization of the involved PMs and their acceptable utilizations result in:

-
- The minimum possible number of active PMs. This minimizes the consumed energy (guaranteed by the KP optimization solver model). KP places the maximum possible items (VMs) in one knapsack (PM) ensuring that the consumed energy is minimized.
- The better utilization for the resulted involved PMs comes up because MCKP model places the VMs allocated to different types of jobs on the same PM whenever possible. Then if there is any available resources, classical KP model fits the VM on these resources. In this case, all the PM resources are kept busy because different types of jobs are placed on the same PM, and every job utilizes different kind of resources. This maximizes the PM utilization, and consequently enhances energy efficiency.

In the after migration VM placement, as in the initial placement, MCKP model places the migrated VM on a PM that hosts a VM of different kinds of jobs whenever possible. Thus, the resources of the involved PM which are not utilized well will be utilized better by keeping them busy during the PM involved period.

7. Conclusions

In this work, the problem of high energy consumption in cloud data centers is investigated from the VM management perspective. The work proposes a distributed approach to an energy-efficient dynamic VM management. The approach determines which VMs to migrate from the source PM, and when. Then, the destination PM is selected to place the VM on it.

Based on the proposed approach, the following aspects can be concluded:

1) In under/overload host detection, the proposed dynamic thresholding techniques outperform the static thresholding ones. In contrast to the systems that adopt the dynamic thresholding techniques, the systems with static thresholding techniques are unable to react to the obviously under/over utilized cases. Fixed values of the threshold (static thresholds) are not suitable for the environments with dynamic and unpredictable workloads, as in cloud computing environment. Different types of applications and jobs are submitted to the cloud computing environment to share the PMs resources. For a better performance, the system should be able to automatically adjust its behavior depending on the workload patterns of the submitted jobs.
2) In VM selection algorithms, the results vary depending on the workload. In general, the algorithms based on selecting the VM with the smallest memory size results in less energy consumption compared to the other algorithms.

3) From the energy consumption point of view, the proposed VM placement strategy, which considers the job type in the placement process, and based on the MCKP approach, outperforms the classical KP. This is due to the optimal use of the resources of the involved PMs hosted in the cloud data centers.

Data Description

The proposed algorithms and policies are applied using real workload traces provided by Google. The Google workload traces are collected from large cloud systems (about 12,500 compute nodes over 29 days). The workload traces consist of different types of jobs. Real workload traces can provide a very high level of realism when used directly in performance evaluation experiments. More details about this data are available in:

"Google Cluster-Usage Traces: Format and Schema" Version 2, By: C. Reiss and J. Wilkes, Google Inc., 2013.

The workload traces contain millions of jobs/tasks, so, using random data from traces, the proposed algorithms are simulated.

References

[1] J. Li, Y. Zhang, X. Chen, Y. Xiang, Secure attribute-based data sharing for Resource-limited users in cloud computing, J. Comput. Secur. 72 (2018) 1–12.
[2] A. Al-Dulaimy, W. Itani, A. Zekri, R. Zantout, Power management in virtualized data centers: state of the art, J. Cloud Comput. 5 (6) (2016) 1–15.
[3] A. Beloglazov, Energy-Efficient Management of Virtual Machines in Data Centers for Cloud Computing, PhD Dissertation, Department of Computing and Information Systems, The University of Melbourne, 2013.
[4] J. Koomey, Estimating Total Power Consumption By Servers in the US and the World, Lawrence Berkeley National Laboratory, 2007 Technical Report.
[5] https://www.eia.gov/, Energy Information Administration, 2017. [Online].
[6] V. Josyula, M. Orr, G. Page, Cisco system inc. Cloud Computing: Automating the Virtualized Data Center, (2012).
[7] A. Beloglazov, J. Abawajyb, R. Buyya, Energy-aware Resource allocation heuristics for efficient management of data centers for Cloud computing, Future Gener. Comput. Syst. 28 (5) (2012) 755–768.
[8] C. Clark, K. Fraser, S. Hand, J.G. Hansen, Live migration of virtual machines, 2nd Symposium on Networked Systems Design & Implementation, (2005).
[9] A. Strunk, A lightweight model for estimating energy cost of live migration of virtual machines, IEEE 6th International Conference on Cloud Computing, (2013).
[10] C. Sapuntzakis, R. Chandra, B. Pfaff, J. Chow, M. Lam, M. Rosenblum, Optimizing the migration of virtual computers NY, USA, 5th Symposium on Operating Systems Design and Implementation (2002).
[11] R. Bradford, E. Kotsovinos, A. Feldmann, H. Schioberg, Live Wide-Area migration of virtual machines including local persistent State San Diego, California, USA, 3rd International Conference on Virtual Execution Environments (2007).
[12] A. Shribman, B. Hudzia, Pre-copy and Post-copy VM live migration for memory intensive applications, Euro-Par 2012: Parallel Processing Workshops, Springer, Berlin Heidelberg, 2013, pp. 539–547.
[13] S. Akoush, R. Sohan, A. Rice, A. Moore, A. Hopper, Predicting the performance of virtual machine migration Miami, FL, USA, IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (2010).
[14] H. Jin, L. Deng, S. Wu, X. Shi, X. Pan, Live virtual machine migration with adaptive memory compression New Orleans, LA, USA, IEEE International Conference on Cluster Computing and Workshops (2009).
[15] E.P. Zawk, N.L. Thein, Improved live VM migration using LRU and splay tree algorithm, Int. J. Comput. Sci. Telecommun. 3 (3) (2012) 1–7.
[16] W. Hu, A. Hicks, L. Zhang, E. Dow, H.J.R.B. Vinay Soni, J. Matthews, A quantitative study of virtual machine live migration Miami, Florida, USA, The Cloud and Autonomic Computing Conference (2013).
[17] H. Liu, C.-Z. Xu, H. Jin, J. Gong, X. Liao, Performance and energy modeling for live migration of virtual machines, 20th International Symposium on High Performance Distributed Computing, (2011).
[18] W. Dargie, Estimation of the cost of VM migration Shanghai, China, 23rd International Conference on Computer Communication and Networks (2014).
[19] S. Chaisiri, B. Lee, D. Niyato, Optimal virtual machine placement across multiple Cloud providers Singapore, IEEE Asia-Pacific Services Computing Conference (2009).
[20] B. Speitkamp, M. Bichler, A mathematical programming approach for server consolidation problems in virtualized data centers, IEEE Trans. Serv. Comput. 3 (4) (2010) 266–278.
[21] P. Agrawal, D. Borgetto, C. Comito, G.D. Costa, J. Pierson, P. Prakash, S. Rao, D. Talia, C. Thiam, P. Trunfio, Large-scale distributed systems and energy efficiency: a holistic View, Scheduling and Resource Allocation, John Wiley & Sons, 2015, pp. 225–262.
[22] H. Van, F. Tran, J. Menaud, Performance and power management for Cloud infrastructures Miami, FL, IEEE 3rd International Conference on Cloud Computing (2010).
[23] F. Hermenier, X. Lorca, J. Menaud, G. Muller, J. Lawall, Entropy: a consolidation manager for clusters, The ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, (2009).
[24] A. Beloglazov, R. Buyya, Energy efficient Resource management in virtualized Cloud data centers, 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, (2010).
[25] A. Khosravi, L. Andrew, R. Buyya, Dynamic VM placement method for minimizing energy and carbon cost in geographically distributed Cloud data centers, IEEE Trans. Sustain. Comput. 2 (2) (2017) 183–196.
[26] Z. Zhou, H. Zhi-Gang, Y. Jun-Yang, J. Abawajy, M. Chowdhury, Energy-efficient virtual machine consolidation algorithm in Cloud data centers, J. Cent. South. Univ. 24 (10) (2017) 2331–2341.
[27] J. Cao, Y. Wu, M. Li, Energy efficient allocation of virtual machines in Cloud computing environments based on demand forecast, the 7th International Conference on Advances in Grid and Pervasive Computing, (2012).
[28] F. Ma, F. Liu, Z. Liu, Multi-objective optimization for initial virtual machine placement in Cloud data center, J. Inf. Comput. Sci. 9 (16) (2012) 5029–5038.
[29] A. Esnault, Energy-Aware Distributed Ant Colony Based Virtual Machine Consolidation in IaaS Clouds, HAL, (2012).
[30] Y. Gao, H. Guan, Z. Qi, Y. Hou, L. Liu, A multi-objective ant colony system algorithm for virtual machine placement in Cloud computing, J. Comput. Syst. Sci. 79 (8) (2013) 1230–1242.
[31] E. Feller, Autonomic and Energy-Efficient Management of Large-Scale Virtualized Data Centers, PhD Dissertation University of Rennes, ISTIC, 2013.
[32] M.H. Ferdaus, M. Murshed, R.N. Calheiros, R. Buyya, Virtual machine consolidation in cloud data centers using ACO metaheuristic Porto, Portugal, Euro-Par 2014 Parallel Proceedings of the 20th International Conference (2014).
[33] A. Aryania, H. Aghdasi, L.M. Khanli, Energy-aware virtual machine consolidation algorithm based on ant colony system, J. Grid Comput. (2018) 1–15.
[34] D. Kumar, Z. Raza, A PSO based VM Resource scheduling model for Cloud computing Ghaziabad, IEEE International Conference on Computational Intelligence & Communication Technology (2015).
[35] S. Dashtia, A. Rahmania, Dynamic VMs placement for energy efficiency by PSO in Cloud computing, J. Exp. Theor. Artif. Intell. 28 (1–2) (2015) 97–112.
[36] V. Reddy, G. Gangadharan, G. Rao, Energy-aware virtual machine allocation and selection in Cloud data centers, J. Soft Comput. (2017).
[37] A. Tripathi, I. Pathak, D. Vidyarthi, Energy efficient VM placement for effective Resource utilization using modified binary PSO, Comput. J. (2017).
[38] N. Quang-Hung, P. Nienz, N. Namz, N. Tuong and N. Thoa, 2013. "A Genetic Algorithm for Power-Aware Virtual Machine Allocation in Private Cloud," Information and Communication Technology, vol. 7804, no. Lecture Notes in Computer Science, pp. 170-179.
[39] M. Vasudevan, Y.-C. Tian, M. Tang, E. Kozan, W. Zhang, Profile-based dynamic application assignment with a repairing genetic algorithm for greener data centers, J. Supercomput. 73 (9) (2017) 3977–3998.
[40] W. Lin, S. Xu, J. Li, L. Xu, Z. Peng, Design and theoretical analysis of virtual machine placement algorithm based on Peak workload characteristics, Soft Comput. 21 (5) (2017) 1301–1314 Springer.
[41] A. Al-Dulaimy, A. Zekri, W. Itani, R. Zantout, Paving the Way for energy efficient Cloud data centers: a type-aware virtual machine placement strategy Vancouver, Canada, IEEE International Conference on Cloud Engineering (IC2E) (2017).
[42] E. Feller, C. Morin, A. Esnault, A Case for Fully Decentralized Dynamic VM Consolidation in Clouds, HAL (2012).
[43] R. Nathuji, K. Schwan, VirtualPower: coordinated power management in virtualized Enterprise systems, ACM SIGOPS Operating Syst. Rev. 41 (6) (2007) 265–278.
[44] S. Srikantaiah, A. Kansal, F. Zhao, Energy aware consolidation for Cloud computing, USENIX Workshop on Power Aware Computing and Systems, (2008).
[45] T. Hirofuchi, H. Nakada, S. Itoh, S. Sekiguchi, Making VM consolidation more energy-efficient by postcopy live migration, 2nd International Conference on Cloud Computing, GRIDs, and Virtualization, (2011).
[46] M. Hubert, E. Vandervieren, An adjusted boxplot for skewed distributions, Comput. Stat. Data Anal. 52 (12) (2008) 5186–5201.
[47] D. Pisinger, Algorithms for Knapsack Problems, PhD Dissertation University of Copenhagen, 1995.
[48] X. Fan, W. Weber, L. Barroso, Power provisioning for a warehouse-sized computer, 34th Annual International Symposium on Computer Architecture, (2007).
[49] H. Kellerer, U. Pferschy, D. Pisinger, The multiple-choice knapsack problem, Knapsack Problems, Springer, 2004, pp. 317–347.
[50] A. Al-Dulaimy, R. Zantout, A. Zekri, W. Itani, Job classification in Cloud computing: the classification effects on energy efficiency Limassol, Cyprus, IEEE/ACM 8th International Conference on Utility and Cloud Computing (2015).
[51] A. Al-Dulaimy, R. Zantout, W. Itani, A. Zekri, Job submission in the Cloud: energy aware approaches San Francisco, USA, World Congress on Engineering and Computer Science, International Association of Engineers (2016).
[52] W. Lin, S. Xu, L. He, J. Li, Multi-resource scheduling and power simulation for cloud computing, Information Sciences vol. 397–398, Elsevier, 2017, pp. 168–186.
[53] C. Reiss, J. Wilkes, Google Cluster-usage Traces: Format and Schema - Version 2, Google Inc., 2013.