Sustainable Computing: Informatics and Systems 20 (2018) 48–55

Contents lists available at ScienceDirect

Sustainable Computing: Informatics and Systems

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

Energy-efficient VM-placement in cloud data center

Sambit Kumar Mishra[a], Deepak Puthal[b,**], Bibhudatta Sahoo[a], Prem Prakash Jayaraman[c], Song Jun[d], Albert Y. Zomaya[e], Rajiv Ranjan[d,f,*]

[a] National Institute of Technology Rourkela, India
[b] University of Technology Sydney, Australia
[c] Swinburne University of Technology, Australia
[d] Chinese University of Geosciences, China
[e] The University of Sydney, Australia
[f] Newcastle University, Newcastle upon Tyne, United Kingdom

ARTICLE INFO

Article history:
Received 15 July 2017
Received in revised form 31 October 2017
Accepted 4 January 2018
Available online 2 February 2018

Keywords:
Cloud computing
Energy consumption
VM consolidation
Task scheduling
Makespan

ABSTRACT

Employing cloud computing to acquire the benefit of cloud by optimizing various parameters that meet changing demands is a challenging task. The optimal mapping of tasks to virtual machines (VMs) and VMs to physical machines (PMs) (known as VM placement) problem are necessary for advancing energy consumption and resource utilization. High heterogeneity of tasks as well as resources, great dynamism and virtualization make the consolidation issue more complicated in the cloud computing system. In this paper, a complete mapping (i.e., task VM and VM to PM) algorithm is proposed. The tasks are classified according to their resource requirement and then searching for the appropriate VM and again searching for the appropriate PM where the selected VM can be deployed. The proposed algorithm reduces the energy consumption by depreciating the number of active PMs, while also minimizes the makespan and task rejection rate. We have evaluated our proposed approach in CloudSim simulator, and the results demonstrate the effectiveness of the proposed algorithm over some existing standard algorithms.

1. Introduction

Cloud is an emerging multi-disciplinary research area with the aim of analyzing the vast amount of data and user requests to extract actionable information. Cloud computing has become the suitable platform for big data processing due to its on-demand elasticity or flexibility, extremely low-latency and massively parallel processing architecture [1]. Almost all IT-industries needs the assistance from the flexible cloud-computing platforms, supported by millions of physical hosts and devices spread in more number of data centers. Cloud computing system is underpinned by virtualization technology that virtualizes the physical cloud resources. The virtual cloud environment increases the throughput as well as scalability of the system. The virtual resources in the cloud system are known as virtual machines (VMs). These VMs are mapped to different user requests for the execution of input tasks. Resource management becomes even more complex when resources are oversubscribed, and the cloud users are not cooperative. To handle these situations, the cloud service provider (CSP) has to follow a proper scheduling mechanism to delivered services. The negotiation between the CSP and the user is technically termed as service level agreement (SLA). The SLA is a part of quality of service (QoS). We use PM and host interchangeably in the rest of the paper.

A large number of advances achieved in developing energy-efficient servers and networking devices. Up to 20% of energy conservation can be achieved through data centers which save up to 30% on cooling energy requirements additionally [2]. A standard cloud deployment consumes a significant amount of energy, and in turn, increases carbon dioxide (CO_2) level indirectly. The energy issue is a big challenge, and therefore, many cloud providers have focused on conserving energy by advancing energy-efficient techniques and protocols [3]. The cloud computing is expressed as an inherently energy-efficient platform due to the scalable view of its resources and multitenant ability [4]. Low-quality VM placement in data centers leads to the massive energy consumption [5–7].

The allocation of service requests to a set of virtual machines running on different hosts while achieving the terms and conditions stated in the SLAs and without degrading the QoS is referred to as the service allocation problem [8–10]. In this paper, we concentrate on the problem of adaptively allocating VMs to physical machines (PMs) in the data center, in the context of unpredictable dynamic workloads. Precisely, this involves delivering decisions

* Corresponding author at: Newcastle University, Newcastle upon Tyne, United Kingdom.
** Corresponding author.
E-mail address: rranjans@gmail.com (R. Ranjan).

https://doi.org/10.1016/j.suscom.2018.01.002
2210-5379/

such as when to allocate VMs, which VMs to relocate; which VMs are assigned to which PMs, which PMs can be turned off. The VMs are clustered according to their type of resources, and then the clustering output is used for the decision of customized VM instances. These decisions can be made with the objective of saving energy consumption of the system by changing the state of idle PMs to disable. Our proposed solution reduces the resource wastage in the cloud system with the help of virtualization technique and efficient allocation policies. We have established various sub-types of VM according to their resource capability. The total input load of the data center is the finite number of tasks where each task involves with several VMs for execution. In this work, our aim is to assign an input task to the existing VM otherwise create VM according to the task and assigned the newly created VM to an active host.

The major contributions of this work are as follows:

- Heterogeneous cloud resources and the input tasks of a cloud system are modeled, and an energy-aware resource allocation platform is introduced to optimize the energy consumption of cloud data centers.
- Propose a task-based VM-placement algorithm (ETVMC) to reduce the energy consumption, minimize the makespan of the system, and reduce the task rejection rate.
- An experimental evaluation to validate the proposed solution utilizing the CloudSim as simulation framework.

The remaining of the paper is prepared as follows. Section 2 outlines an overview of some related work; Section 3 reviews the problem statement; Section 4 describes the cloud system model and architecture. Section 5 presents the proposed energy model for the consolidation problem. Section 6 illustrates our proposed work to optimize various parameters in a heterogeneous computing environment. Section 7, presents simulation results and shows the effectiveness of our algorithm compared to existing standard algorithms. Section 8, concludes the work.

2. Related work

Various algorithms have previously been introduced pointing to the scheduling issue in the cloud computing environment. Researchers and companies have done lots of research on this problem to gain a relationship between energy consumption and system performance. From an extensive study, we have recognized two distinguishing issues to optimize the energy consumption of the cloud system. Firstly, the allocation of tasks to the appropriate VMs [8,9,11,12]. Secondly, the VM placement in the cloud [9,13–15]. The literature expressed two distinctive models for the scheduling of tasks. Those are Bag of Task (BoT) [16,17], and Direct Acyclic Graph (DAG) [18]. All BoT-based applications are independent parallel task [19]. The DAG-based applications are organized as graph where nodes (tasks) are connected by edges. In the graph, the node weight denotes resource requirements and edge weight denotes requested file size. The VM selection for a specific task is based on (a) the specified SLA (e.g., cost constraint), (b) the CSP to enhance profit, (c) other objectives like makespan minimization, throughput maximization, energy optimization, etc.

A bi-criteria scheduling approach has been proposed to schedule DAG-based applications in the cloud to optimize running time and cost [18]. Their experimental result shows that their proposed scheduler is superior to an approach called join the shortest queue [20]. Their proposed work does not guarantee the optimal resource utilization. Achar et al. [17] have proposed a scheduling algorithm for BoT-based applications. Initially, they set some priorities to the tasks and VMs, and after that they clustered tasks as well as VMs. Then, their approach selects an appropriate VM cluster for a specific task cluster They have compared their algorithm with first come first serve which shows high resource utilization with low makespan.

The VM placement approaches can be static [21] or dynamic [13,22–25]. In static approach, the allocation is not changed after the decision once made. But, in dynamic approach, the allocation of VM to a physical machine (PM) may change at the time of execution. The information regarding the actual load is used in dynamic approach while the information is not available to static approach. The dynamic approach is more relevant in this work. Various allocation decisions of VM to PM, based on dynamic VM placement techniques have been proposed to save energy consumption [23,26]. In this paper, we have tried to increase the profit of the service providers by the optimal allocation of VM to PM as well as the task to VM that take into account the energy usage is minimum.

Mosa and Paton [25] have proposed a dynamic genetic algorithm based VM placement approach to optimize the energy usage along with fewer SLA violation rate. They have considered CPU time as the resource to represent the utility function. The load of a VM is the CPU time requirement of all the tasks assigned to that VM. According to the load of the VM, the migration will be performed. Arianyan et al. [27] have focused on consolidation problem to enhance resource utilization and efficiency to cause energy conversion in cloud data centers. They have proposed new resource management technique according to three parameters: energy consumption, live migration, and SLA violation. They split the whole problem into two sub-problems. Firstly, obtain the overloaded hosts, and secondly, find a suitable VM for migration by using multi-criteria decision-making techniques.

A solution for the consolidation of VMs based on Bernoulli trials has been presented by Carlo et al. [28]. The solutions have two principal characteristics that include a self-organizing and adaptive method that makes it efficient for handling the large data centers. By using Bernoulli trials and local information, the server determines the possibility of the execution applications. One of the best advantages of the solution is the balancing of CPU-bound and memory-bound applications. Vakilinia et al. [29] have proposed a VM placement platform to minimize the power consumption of the data center. In that, one scheduler works on expected load and another on unpredictable load.

A novel taxonomy for the resource allocation in the cloud is presented by Hameed et al. [9]. They have identified various open research challenges correlated with energy efficient allocation resources in a data center. The scheduling problem of scientific workflows in the cloud system is addressed by Casas et al. [8] and employed a genetic-based algorithm. The scheduler optimized the makespan and monetary cost of the system for computational and data-intensive scientific tasks [8]. An entropy-based VM placement algorithm is proposed by Chen et al. [13] to reduce the energy and thermal cost, also minimizes the number of hot spots in the system. Their algorithm consolidates with Dynamic Voltage Frequency Scaling (DVFS) to optimize the energy consumption of the system.

From this related work, we remark that most schedulers generate scheduling plans based on a fixed number of resources (e.g., VMs). Some schedulers are designed for a BoT type application with no dependencies between tasks. This factor prevents them from being used in scheduling of user requests (tasks) on clouds. In order to address the limitations of existing scheduling solutions, we have proposed ETVMC, an algorithm to optimize various performance metrics. In summary, ETVMC (1) has a well-organized structure to efficiently map VMs, (2) enables service providers to optimize the energy consumption and reduce task rejection rate of the system, and (3) enables cloud users to optimize execution time and monetary cost.

3. Problem statement

VM-placement plays an important role for efficient utilization of energy in cloud data center. There are m finite heterogeneous hosts in the cloud system. Each host is in one state among the two states: active or sleep state and all hosts are in sleep state initially. There are k types of virtual machines classified based on their resources (like processing speed, main memory, secondary storage, bandwidth). Similar to the VM types, tasks are also classified so that tasks of the same tasks group can fit to one type of virtual machines. The task manager has complete information about the tasks in the queue and also about the incoming tasks. Therefore, according to the need as mentioned by the task manager for the incoming tasks, new VMs are created in some specific hosts. So, the appropriate assignment of new virtual machines to some hosts is an assignment problem and a well known NP-complete problem. The sub-optimal solution for the mentioned assignment problem with the aim of optimized energy consumption is the primary objective of this work.

4. System model and architecture

In this section, we present a scheduling model for a Cloud environment that includes the host model, VM model and task model in a Cloud data center as shown in Fig. 1. The cloud system has m heterogeneous hosts (say set $H = \{Host_1, Host_2, \ldots, Host_m\}$). These hosts are heterogeneous in terms of their resource capacities. The tasks originated from the cloud users over the Internet is referred to as service request. Users request services from the cloud service provider (CSP). They submit their tasks (heterogeneous in nature) to the task queue of the cloud system. The task manager classifies those tasks to four different categories of tasks (CPU-intensive, Memory-intensive, I/O-intensive, and Communication-intensive) to map tasks to virtual machines. The tasks manager has also known about the types of virtual machines and also its sub-type. There are three types of VM: VM_{Type1} is fit for CPU-intensive tasks, VM_{Type2} is fit for memory-intensive tasks, VM_{Type3} is fit for I/O-intensive tasks, and VM_{Type4} is fit for communication-intensive tasks. Within a specific VM-type, a finite set of VM-sub-type are available (where VM_{ij} represents jth VM of ith type). All the tasks from the three sub-queues are submitted to the host manager. The host manager also has the complete information about the VM-type. In this system, the host manager is responsible for the creation of VMs (means the creation of a specific type of VM on the top of a specific host) to meet certain service level agreements (SLAs).

Initially, all hosts of the system are in the sleep state. The host manager chooses a VM-type according to the task-type and creates an appropriate VM in a host where energy and makespan to be optimized. If any active host has sufficient resources for the creation of required VM, then create the VM on the top of the active host with minimum energy consumption. On the other hand, if no active host can provide resources for the creation of required VM, then the host manager goes for another host which is in the sleep state and change its status to the active state and create the required VM. Here, the deadline of the task is considered as SLA. The service delivered to the cloud users based on hard-deadline constraint. If the SLA between the cloud user and CSP permits to wait for some time (i.e., the time required to complete the execution of previously assigned tasks to the target VM) for the execution of the new task, then the new task is in the local queue of that VM. When there are no tasks in the local queue of a VM, then the VM is deallocated from the host and the resources are free. I/O-intensive task mostly waits for the network, filesystem, and database [15].

5. Energy model

We have considered m independent heterogeneous, uniquely addressable computing entity (hosts) in a cloud system. We have a set of $V = \{V_1, V_2, \ldots, V_m\}$, $(p \times m)$ heterogeneous VMs, where $V_j = \{vm^x_{j1}, vm^x_{j2}, vm^x_{j3}, \ldots, vm^x_{jp_j}\}$ and p_j is the number of VMs running on jth host ($p = p_1 + p_2 + \cdots + p_j + \cdots + p_m$). Here, vm^x_{jk} is the kth VM of jth host of x-type and $x = \{1, 2, 3, 4\}$, i.e., $x = 1$ means the VM is of CPU-type, $x = 2$ means the VM is of memory-type, $x = 3$ means the VM is of I/O-type, and $x = 4$ means the VM is of communication-type. Let there be $T = \{t_1, t_2, \cdots, t_n\}$, n heterogeneous service requests (tasks), where each task t_i has a service length L_i in terms of a million instructions (MI).

The energy consumption by a task t_i, $1 \le i \le n$ is represented by E_{ijk}, i.e., the energy consumed by ith task on jth VM of kth host. This energy consumption is calculated by adding the energy consump-

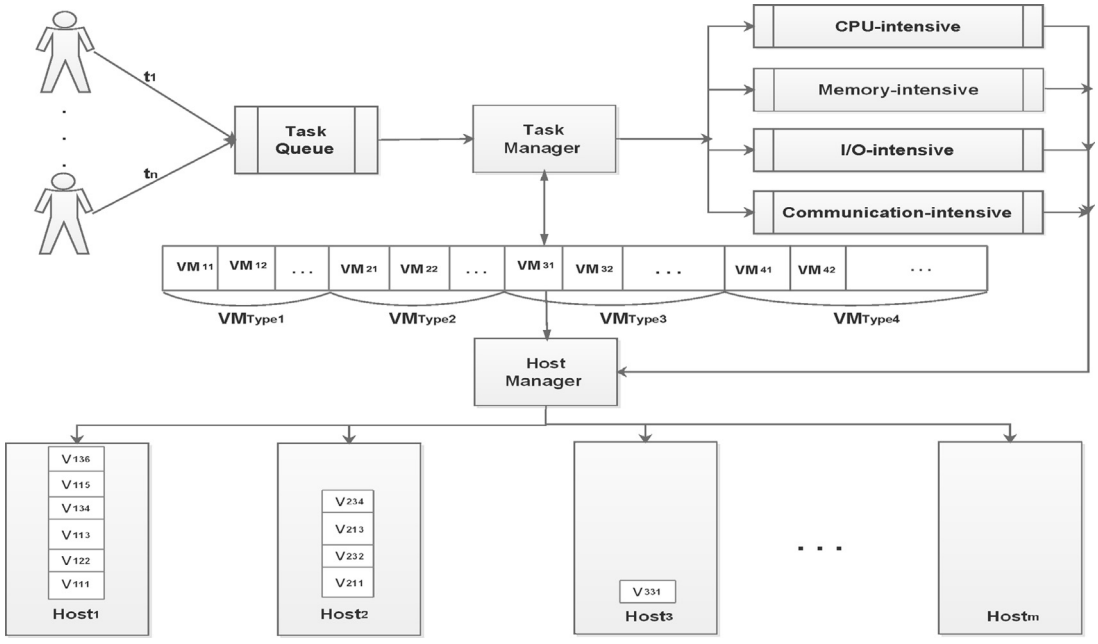

Fig. 1. Cloud system model.

tion due to required data transfer (E_{ijk}^{tran}) and the energy consumed due to the execution of the task (E_{ijk}^{proc}).

$$E_{ijk} = E_{ijk}^{tran} + E_{ijk}^{proc} \tag{1}$$

The required file transfer for the *ith* task consumed some amount of energy which is calculated by the product of the sum of transfer time of all required files with the average power consumption for the transfer of data (P_{avg}). Let the power consumed by *kth* active host is denoted as P_{active_k}. Hence, the average power consumed by the host (P_{avg}) in data center can be calculated as $P_{avg} = (\sum_{k=1}^{m} P_{active_k})$.

$$E_{ijk}^{tran} = \left\{ \sum_{l=1}^{h} Time^{tran}(f_{ijk}^{l}) \right\} \times P_{avg} \tag{2}$$

The energy consumed by the execution of *ith* task on *jth* VM (*x*-type) of *kth* host is the product of the execution time of task (ET_{ijk}^{x}) and the average power consumption of *jth* VM (*x*-type) of *kth* host, i.e., P_{jk}^{x}.

$$E_{ijk}^{proc} = ET_{ijk}^{x} \times P_{jk}^{x} \tag{3}$$

The energy consumption of a VM is the sum of energy consumption of the same type of VM on the same host by all assigned tasks is represented as E_{jk}^{x}, $x = \{1, 2, 3, 4\}$.

$$E_{jk}^{x} = \sum_{i=1}^{n} \left\{ ET_{ijk}^{x} \times TAT_{ij} \right\} \tag{4}$$

TAT is the temporarily allocated task matrix for all the tasks on different VMs of each host. Each host has their *TAT* matrix and is represented as follows.

$$TAT_{ij} = \begin{cases} 0, & \text{If } t_i \text{ is not assigned to } j^{th} \text{ VM of that host} \\ 1, & \text{If } t_i \text{ is assigned to } j^{th} \text{ VM of that host.} \end{cases} \tag{5}$$

The energy consumed by a physical host is the sum of energy consumed by all VMs on that host (in the active state) and the energy consumed by the background applications (in an idle state). The energy consumed by the *kth* host is represented as E_k and estimated as follows.

$$E_k = Idle_k \times P_{Idle_k} + \sum_{j=1}^{p_j} E_{jk}^{x} \tag{6}$$

Here, P_{Idle_k} is the energy consumed by the *kth* host in idle state and $Idle_k$ is the amount of time the *kth* host is in idle state in millisecond, which is calculated using the following equation.

$$Idle_k = Makespan - \sum_{k=1}^{m} \sum_{j=1}^{p_j} \sum_{i=1}^{n} ET_{ijk} \tag{7}$$

The makespan of the system is the maximum amount of time required by a host to execute all input tasks in the system. In other words, the maximum amount of time a host is in active state.

$$Makespan = \max_{1 \le k \le m} \left\{ \sum_{j=1}^{p_j} \sum_{i=1}^{n} ET_{ijk} \right\} \tag{8}$$

The total energy consumption of the system is the sum of energy consumed by all the hosts using the following equation.

$$E = \sum_{k=1}^{m} E_k \tag{9}$$

The objective is to minimize the energy consumption in a data center. This energy optimization problem in the data center with *m* hosts can be presented as follows.

$$Minimize \left\{ \sum_{k=1}^{m} E_k \right\} \tag{10}$$

6. Energy-aware Task-based Virtual Machine Consolidation Algorithm

In this paper, we have presented the complete path to consolidate virtual machines to physical machines based on input task. A task can be represented as 5-tuples i.e., $t_i = \{L_i, d_i, M_i, IO_i, \lambda_i\}$. Here, L_i is the length of the *ith* task in terms of Million Instruction (MI). d_i is the deadline of *ith* task in terms of second. M_i is the main memory requirement of *ith* task in terms of *Mb*. IO_i is the input/output (IO) requirement of *ith* task. λ_i is the bandwidth requirement of *ith* task in terms of *Mb*.

The VM consolidation algorithm is presented in Algorithm 1. A finite number of tasks (with *n* number of task set, *T*) along with their deadlines *D*, the host set *H* with *m* hosts, and the VM-types according to their resource capabilities. This algorithm releases the makespan of the system and total energy consumption. Makespan is the time taken by the host to perform execution of all input tasks. Here, the objective is to minimize this makespan value along with the energy consumption of the cloud system.

Algorithm 1. Energy-aware Task-based Virtual Machine Consolidation (*ETVMC*)

Input: Task set: $T = \{t_1, t_2, \ldots, t_n\}$, Deadline of task: $D = \{d_1, d_2, \ldots, d_n\}$, Host set: $H = \{Host_1, Host_2, \ldots, Host_m\}$, VM Types: $VM_{Type} = VM_{Type1}, VM_{Type2}, VM_{Type3}, VM_{Type4}\}$.
Output: Makespan, Energy.

```
Update TQ ⟵ SortQTask(T, D) using Algorithm 2.
[Q_a, Q_b, Q_c, Q_d] ⟵ ClassifyTasks(TQ) using Algorithm 3.
for each task t_i ∈ T do
    FreeVMs() Algorithm 4
    FreeHosts() Algorithm 5
    vm ⟵ SelectVMType(t_i, Type(t_i. T)) Using Algorithm 6
    h ⟵ SelectHost(vm) Using Algorithm 7
    Allocate t_i to vm deployed on host h.
end for
FreeVMs() Algorithm 4
FreeHosts() Algorithm 5
```

In step-1 of Algorithm 1, the sub-algorithm (Algorithm 2) is called by providing the set of tasks and their deadlines as input. The Algorithm 2 (*SortTask*) sort all the tasks in ascending order of their deadlines. The *RemoveMin* function in step-2 of Algorithm 2 will remove the task with minimum deadline value from the array of tasks and store in the queue, *TQ* to the step-1 of Algorithm 1. The need of Algorithm 2 is same as building a Min-heap of the tasks on the basis of their deadlines. The Min-heap is built based on the deadline as key value so that task with least deadline to be removed first.

Algorithm 2. *SortQTask*

Input: Task set: $T = \{t_1, t_2, \ldots, t_n\}$, Deadline of task: $D = \{d_1, d_2, \ldots, d_n\}$.
Output: *TQ*.

```
for i = 1 to n do
    TQ[i] ⟵ RemoveMin(T_i)
    RemoveMin(T_i) will remove the task which has minimum d_i value.
end for
Return TQ
```

The step-2 of Algorithm 1 calls the sub-algorithm (Algorithm 3) to classify all the tasks into four categories. The sorted task queue, *TQ* is provided as input to the algorithm. Also the resource requirement and parameters of tasks, RPT_i along with their lower and upper bounds are provided to the Algorithm 3. Here, $RPT_i = \{L_i, d_i, M_i, IO_i, \lambda_i\}$. *CL* and *CU* are the lower and upper bounds for the

length of the tasks respectively. *DL* and *DU* are the lower and upper bounds for the deadline of the tasks respectively. *ML* and *MU* are the lower and upper bounds for the main memory requirement of the tasks respectively. *IOL* and *IOU* are the lower and upper bounds for the IO requirement of the tasks respectively. *BL* and *BU* are the lower and upper bounds for the bandwidth requirement of the tasks respectively. The Algorithm 3 returns four task sets: $CPU_{intensive}$, $Memory_{intensive}$, $IO_{intensive}$, $Communication_{intensive}$. In step-1, the upper bound for the processing speed (UC) of the processor is calculated. A loop is started from step-2 and ended at step-14, and in *ith* iteration, the *ith* task is placed in one of the four categories of task. Here, the value of w_{c_i}, w_{m_i}, w_{io_i}, w_{λ_i} are in between 0 and 1. The summation of all these value is 1, i.e., $w_{c_i} + w_{m_i} + w_{io_i} + w_{\lambda_i} = 1$. In step-6, x_i is calculated and then all w_{c_i}, w_{m_i}, w_{io_i}, w_{λ_i} values are updated by multiplying x_i. *Max* is the maximum value among those updated 4 values. Based on the *Max* value, the task category is determined.

Algorithm 3. *ClassifyTasks*

Input: Task set: $T = \{t_1, t_2, \ldots, t_n\}$, Resource requirement and parameters of tasks: $RPT_i = \{L_i, D_i, M_i, IO_i, \lambda_i\}$, Range of all parameters of tasks: CPU: {*CL*, *CU*}, Deadline: {*DL*, *DU*}, Main Memory: {*ML*, *MU*}, I/O: {*IOL*, *IOU*}, Bandwidth: {*BL*, *BU*}.
Output: $CPU_{intensive}$, $Memory_{intensive}$, $IO_{intensive}$, $Communication_{intensive}$.

1: $UC \longleftarrow CU/DU$
2: **for** each task $t_i \in T$ **do**
3: $C_i \longleftarrow \frac{L_i}{D_i}$
4: $w_{c_i} \longleftarrow \frac{C_i}{UC}$, $w_{m_i} \longleftarrow \frac{M_i}{MU}$
5: $w_{io_i} \longleftarrow \frac{IO_i}{IOU}$, $w_{\lambda_i} \longleftarrow \frac{\lambda_i}{BU}$
6: $x_i \longleftarrow \frac{1}{w_{c_i}+w_{m_i}+w_{io_i}+w_{\lambda_i}}$
7: $w_{c_i} = x_i \times w_{c_i}$, $w_{m_i} = x_i \times w_{m_i}$
8: $w_{io_i} = x_i \times w_{io_i}$, $w_{\lambda_i} = x_i \times w_{\lambda_i}$
9: $Max = \{w_{c_i}, w_{m_i}, w_{io_i}, w_{\lambda_i}\}$
10: $t_i \in CPU_{intensive}$ iff $w_{c_i} = Max$
11: $t_i \in Memory_{intensive}$ iff $w_{m_i} = Max$
12: $t_i \in IO_{intensive}$ iff $w_{io_i} = Max$
13: $t_i \in Communication_{intensive}$ iff $w_{\lambda_i} = Max$
14: **end for**
15: **Return** $CPU_{intensive}$, $Memory_{intensive}$, $IO_{intensive}$, $Communication_{intensive}$

A loop starts from the step-3 and ends at step-9 of Algorithm 1. This loop will run for *n* (number of tasks) iterations. The free virtual machines those are in the active state are transformed to sleep/off state and resources of those VMs are handover to the corresponding host. The active host set with their VM lists are provided as input to Algorithm 4. This algorithm updates or reduces the number of active VMs (*AV*) of the system. Here, a checking will be done for each VM to update the VM set *AV*.

Algorithm 4. *FreeVMs*()

Input: Active host set: $AH = \{Ah_1, Ah_2, \ldots, Ah_k\}$, VM set: $AV = \{Av_{11}, Av_{12}, \ldots, Av_{1p}, \ldots, Av_{21}, Av_{22}, \ldots, Av_{2p}\}$.
Output: Updated *AV*.

for each active host $Ah_i \in AH$ **do**
  for each VM $Av_{ij} \in Ah_i$ **do**
    if Av_{ij} is idle **then**
      Deallocate resources of Av_{ij} to Ah_i.
    end if
  end for
end for

After performing FreeVMs in Algorithm 4, FreeHosts will be done in Algorithm 5. Here also the host set (*AH*) with the active VM set (*AV*) are provided as input. The *AH* set will be updated through this algorithm. The loop from step-1 to step-5 will convert the state of the host from active to sleep if that host is in the idle state at that point of time. Then, an another loop will be there from step-7 to step-18, and in those steps, the state of the host will be changed based on VM-migration. If all the virtual machines on a less loaded host can be migrated to other active hosts, then all the current state VMs are migrated to other active hosts and then the state of that host is changed to the sleep state.

Algorithm 5. *FreeHosts*()

Input: Active host set: $AH = \{Ah_1, Ah_2, \ldots, Ah_k\}$, Active VM set: $AV = \{Av_1, Av_2, \ldots, Av_p\}$.
Output: Updated *AH*.

for each active host $Ah_i \in AH$ from Ah_k to Ah_1 **do**
  if Ah_i is idle **then**
    Transform the host Ah_i from idle state to sleep state.
  end if
end for
$VMStatus = 0$
for each active host $Ah_i \in AH$ from Ah_k to Ah_1 **do**
  for each VM $Av_{ij} \in Ah_i$ **do**
    if Migration of Av_{ij} to $AH - Ah_i$ **then**
      $VMStatus = VMStatus + 1$.
      $Migr_{ij} \longleftarrow$ targeted host ID
    end if
  end for
  if $VMStatus == j - 1$ **then**
    Migrate all VMs using $Migr_{ij}$
    Transform the host Ah_i from idle state to sleep state.
  end if
end for

The step-6 of Algorithm 1 will choose an appropriate VM-type based on the task type and the resource requirement of the task. The task along with the task type (or queue type of the task) is passed as input to the Algorithm 6. We have considered four types of VM: VM_{Type1}, VM_{Type2}, VM_{Type3}, VM_{Type4} for $CPU_{intensive}$, $Memory_{intensive}$, $IO_{intensive}$, $Communication_{intensive}$. These VM-types are provided as input to Algorithm 5 along with their sub-types. There are *k* number of VM sub-types within one VM-type for one type of tasks. The VM-types within a type (VM_{Type1}) are in ascending of their resource capacity (*RC*). It means that $RC(VM_{Type11}) \leq RC(VM_{Type12}) \leq \cdots \leq RC(VM_{Type1k})$. This property is following by all VM-type sets. For example, if the required resource of a task is not available in $VM_{Type}11$, then the checking procedure is forwarded to $VM_{Type}12$ and if the resource requirement of the task is matched with the $VM_{Type}12$ resource, then the algorithm stops there and return $VM_{Type}12$. This is the way to choose a VM-type for all four types of tasks.

Algorithm 6. *SelectVMType*()

Input: task *t*, task type *TT*, $VM_{Type} = \{VM_{Type1}, VM_{Type2}, VM_{Type3}, VM_{Type4}\}$, VM sub-types: $VM_{Type1} = \{VM_{Type11}, VM_{Type12}, \ldots, VM_{Type1k}\}$, $VM_{Type2} = \{VM_{Type21}, VM_{Type22}, \ldots, VM_{Type2k}\}$, $VM_{Type3} = \{VM_{Type31}, VM_{Type32}, \ldots, VM_{Type3k}\}$, $VM_{Type4} = \{VM_{Type41}, VM_{Type42}, \ldots, VM_{Type4k}\}$.
Output: *VMType*

All VM sub-types are sorted within VM_{Type}.
for each VM sub-type $VM_{TypeTTi} \in VM_{TypeTT}$ **do**
  if *t* is fit in $VM_{TypeTTi}$ **then**
    $VMType \longleftarrow VM_{TypeTTi}$
    Return *VMType* and Stop.
  end if
end for

The step-7 of Algorithm 1 perform the selection of best-matched host according to the VM-type. The type of VM found in step-6 is passed to the Algorithm 7 (*SelectHost*) along with the active state host set ($AH = \{Ah_1, Ah_2, \ldots, Ah_k\}$) and sleep state host set ($SH = \{Sh_1, Sh_2, \cdots, Sh_r\}$). Here, the $RC(Ah_1) \leq RC(Ah_2) \leq \cdots \leq RC(Ah_k)$, and $RC(Sh_1) \leq RC(Sh_2) \leq \cdots \leq RC(Sh_r)$, and the total host $m = k + r$. This Algoritm-7 returns the host where the new VM is deployed. A loop from step-1 to step-6 of Algorithm 7 searches the active state host where the specified VM can deploy. If there is no active state host where the VM can create, then searching for the sleep state host in the loop from step-7 to step-13. Here, we have proposed two heuristic algorithms (Algorithm 6 and Algorithm 7) to select VM type and to select host respectively. The main aim is that we are trying to find an active state host instead of sleep state host.

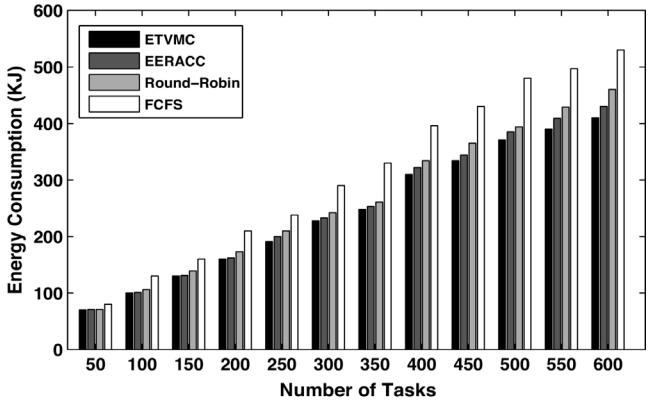

Fig. 2. Comparison of energy consumption of the system for ETVMC, EERACC, Round-Robin and FCFS, when number of VMs is fixed and the number of tasks is varied.

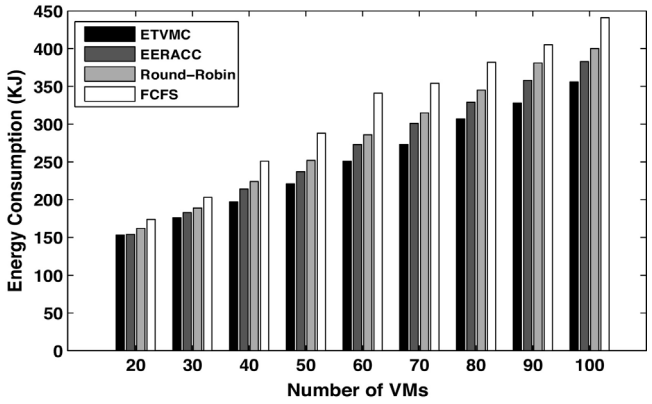

Fig. 3. Comparison of energy consumption of the system for ETVMC, EERACC, Round-Robin and FCFS, when number of tasks is fixed and the number of VMs is varied.

Algorithm 7. *SelectHost*()

Input: VM_{Type}, Active host set: $AH = \{Ah_1, Ah_2, \ldots, Ah_k\}$, Sleep host set: $SH = \{Sh_1, Sh_2, \ldots, Sh_r\}$.
Output: *Host*

```
for each Ah_i ∈ AH do
  if VM of VM_Type is fit in Ah_i then
    Host ⟵ Ah_i
    Return Host and Stop.
  end if
end for
for each Sh_i ∈ SH do
  if VM of VM_Type is fit in Sh_i then
    Host ⟵ Sh_i
    Transform state of Sh_i to active state.
    Return Host and Stop.
  end if
end for
```

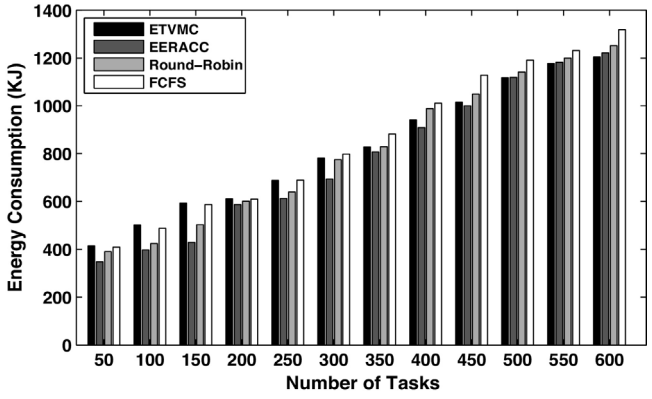

Fig. 4. Comparison of makespan of the system for ETVMC, EERACC, Round-Robin and FCFS, when number of VMs is fixed and the number of tasks is varied.

7. Experimental results

The experiments are carried out with the help of CloudSim 3.03 simulator. Xen is used as the virtual machine monitor (hypervisor). The proposed Energy-aware Task-based Virtual Machine Consolidation (ETVMC) algorithm is implemented in Java and tested on a Dell workstation with Intel i7 3.07 GHz CPU and 24 GB memory. Our simulations are performed over a set of heterogeneous cloud resources, i.e., hosts as well as VMs, and heterogeneous input service requests. The resource requirement and the length of the service requests are also generated randomly. In each test case, the number of hosts is fixed and the number of VMs varies from 20 to 200. The resources for the VMs are generated randomly and the resources of all the VMs deployed on a host is less than the resource capacity of that host. To evaluate the efficiency of our proposed algorithm, we have compared the proposed ETVMC algorithm with FCFS, Round-Robin, and EERACC proposed in [29] algorithms in terms of energy consumption of the system.

To evaluate the performance, we have compared the aggregate values for the energy consumption of all VMs with the energy consumption values estimated for the whole server one by one. There are 10 different numbers of runs of all the algorithms for a specific set of input tasks in the simulation to avoid transient anomalies. We have considered four ranges of the task length (input file size of the task) for one set of input task. The ranges are [3000–5000], [6000–8000], [8000–9500], and [9500–11,000]. Each input task set has 25% of tasks from one range of task length.

The results as presented in Figs. 2–7 demonstrate that ETVMC algorithm performs better than all existing compared algorithms. The proposed algorithm gives better average energy consumption than FCFS, Round-Robin, and EERACC algorithms. The energy consumption of the system increases linearly for all approaches as shown in Figs. 2 and 3. The bar-graphs show a small gap between energy consumption of all the algorithms when the number of tasks or VMs is less whereas they show a big difference when the number of tasks or VMs increases.

The makespan of the system for the proposed algorithm is worst among all other compared algorithms when the number of input tasks is less. This is because ETVMC algorithm always trying to use less number of hosts and those hosts should use maximally. But in general, the number of input tasks to a cloud system is much more and during those cases, ETVMC algorithm gives satisfactory makespan value as shown in Fig. 4. The number of tasks is fixed to 300 and for this case, the proposed algorithm does not perform well as shown in Fig. 5. But, for the less number of VMs, ETVMC gives satisfactory makespan value. Therefore, for the case of a large number of tasks and less number of available resources, our proposed algorithm have a better makespan as compared to others.

The task rejection rate is the rate of tasks rejected by the system. The task rejection rate is related to the load of each VM. Because, when the load of individual VM is more, the rate of task rejection rate also increases. The rejection rate also measures the system scalability. The task rejection rate is inversely proportional to the scalability features. It means if the rejection rate is more, the system is less scalable, and vice-versa. Fig. 6 shows the task rejection rate for all four algorithms including our proposed one when the number of task varies and the number of VMs is fixed. Fig. 7 shows the task rejection rate for all the algorithms when the number of VM varies and the number of task is fixed. From the simulation results,

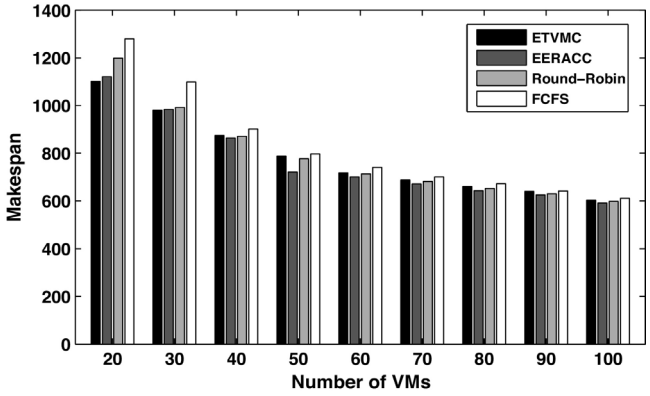

Fig. 5. Comparison of makespan of the system for ETVMC, EERACC, Round-Robin and FCFS, when number of tasks is fixed and the number of VMs is varied.

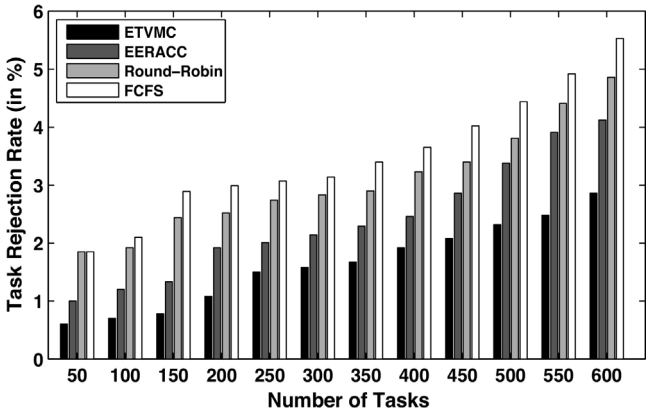

Fig. 6. Comparison of task rejection rate of the system for ETVMC, EERACC, Round-Robin and FCFS, when number of VMs is fixed and the number of tasks is varied.

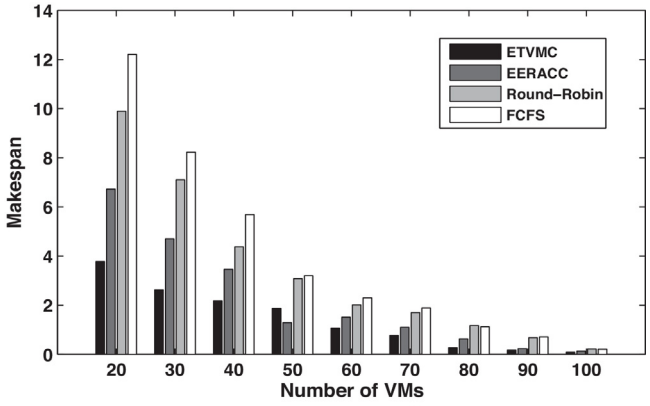

Fig. 7. Comparison of task rejection rate of the system for ETVMC, EERACC, Round-Robin and FCFS, when number of tasks is fixed and the number of VMs is varied.

we remark that the proposed algorithm ETVMC perform superior as compared to FCFS, Round-Robin, and EERACC algorithms.

8. Conclusion

We have presented a task-based VM-placement algorithm (ETVMC) by introducing heterogeneous tasks, VMs, and hosts in the cloud system. The goal is to efficiently allocate tasks to VMs and then VMs to hosts so that the allocation minimizes energy consumption, makespan, and task rejection rate. Its solution performance is compared to FCFS, Round-Robin, and EERACC [29] algorithms. We have got results for systems where resource requirements of service requests may vary dynamically during their service time. The simulation results show that ETVMC is preferred to those algorithms.

References

[1] D. Puthal, B. Sahoo, S. Mishra, S. Swain, Cloud computing features, issues, and challenges: a big picture, 2015 International Conference on Computational Intelligence & Networks (CINE 2015) (2015) 116–123, http://dx.doi.org/10.1109/CINE.2015.31.
[2] L. Minas, B. Ellison, Energy Efficiency for Information Technology: How to Reduce Power Consumption in Servers and Data Centers, Intel Press, 2009.
[3] S. Zeadally, S.U. Khan, N. Chilamkurti, Energy-efficient networking: past, present, and future, J. Supercomput. (2012) 1–26, http://dx.doi.org/10.1007/s11227-011-0632-2.
[4] D. Kliazovich, P. Bouvry, Y. Audzevich, S.U. Khan, GreenCloud: a packet-level simulator of energy-aware cloud computing data centers, J. Supercomput. (2010) 1–5, http://dx.doi.org/10.1109/GLOCOM.2010.5683561.
[5] L. Wang, S.U. Khan, J. Dayal, Thermal aware workload placement with task-temperature profiles in a data center, J. Supercomput. (2012) 1–24, http://dx.doi.org/10.1007/s11227-011-0635-z.
[6] L. Wang, S.U. Khan, Review of performance metrics for green data centers: a taxonomy study, J. Supercomput. 63 (3) (2013) 639–656, http://dx.doi.org/10.1007/s11227-011-0704-3.
[7] P. Sarwesh, N.S.V. Shet, K. Chandrasekaran, Effective integration of reliable routing mechanism and energy efficient node placement technique for low power IoT networks, Int. J. Grid High Perform. Comput. 9 (4) (2017) 16–35, http://dx.doi.org/10.4018/IJGHPC.2017100102.
[8] I. Casas, J. Taheri, R. Ranjan, L. Wang, A.Y. Zomaya, GA-ETI: An enhanced genetic algorithm for the scheduling of scientific workflows in cloud environments, J. Comput. Sci. https://doi.org/10.1016/j.jocs.2016.08.007.
[9] A. Hameed, A. Khoshkbarforoushha, R. Ranjan, P.P. Jayaraman, J. Kolodziej, P. Balaji, S. Zeadally, Q.M. Malluhi, N. Tziritas, A. Vishnu, et al., A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems, Computing 98 (7) (2016) 751–774, http://dx.doi.org/10.1007/s00607-014-0407-8.
[10] J. Tao, J. Kolodziej, R. Ranjan, P. Prakash Jayaraman, R. Buyya, A note on new trends in data-aware scheduling and resource provisioning in modern HPC systems, Future Gener. Comput. Syst. 51 (C) (2015) 45–46, http://dx.doi.org/10.1016/j.future.2015.04.016.
[11] S.K. Mishra, D. Puthal, B. Sahoo, S.K. Jena, M.S. Obaidat, An adaptive task allocation technique for green cloud computing, J. Supercomput. (2017) 1–16, http://dx.doi.org/10.1007/s11227-017-2133-4.
[12] W. Sun, T. Sugawara, Heuristics and evaluations of energy-aware task mapping on heterogeneous multiprocessors, IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW) (2011) 599–607, http://dx.doi.org/10.1109/IPDPS.2011.209.
[13] X. Chen, Y. Chen, A.Y. Zomaya, R. Ranjan, S. Hu, CEVP: Cross entropy based virtual machine placement for energy optimization in clouds, J. Supercomput. 72 (8) (2016) 3194–3209, http://dx.doi.org/10.1007/s11227-016-1630-1.
[14] A. Zhou, S. Wang, C.-H. Hsu, M.H. Kim, K.-s. Wong, Network failure–aware redundant virtual machine placement in a cloud data center, Concurr. Comput. Pract. Exper. 29 (24) (2017) e4290, http://dx.doi.org/10.1002/cpe.4290.
[15] S. Wang, A. Zhou, C.-H. Hsu, X. Xiao, F. Yang, Provision of data-intensive services through energy-and QoS-aware virtual machine placement in national cloud data centers, IEEE Trans. Emerg. Topics Comput. 4 (2) (2016) 290–300, http://dx.doi.org/10.1109/TETC.2015.2508383.
[16] L. Deng, Q. Yu, J. Peng, Adaptive scheduling strategies for cloud-based resource infrastructures, Secur. Commun. Netw. 5 (10) (2012) 1102–1111, http://dx.doi.org/10.1002/sec.425.
[17] R. Achar, P.S. Thilagam, D. Shwetha, H. Pooja, et al., Optimal scheduling of computational task in cloud using virtual machine tree, Third International Conference on Emerging Applications of Information Technology, IEEE (2012) 143–146, http://dx.doi.org/10.1109/EAIT.2012.6407881.
[18] H. Kloh, B. Schulze, R. Pinto, A. Mury, A bi-criteria scheduling process with CoS support on grids and clouds, Concurr. Comput. Pract. Exper. 24 (13) (2012) 1443–1460, http://dx.doi.org/10.1002/cpe.1868.
[19] C. Anglano, M. Canonico, Scheduling algorithms for multiple bag-of-task applications on desktop grids: a knowledge-free approach, in: International Symposium on Parallel and Distributed Processing, IPDPS 2008, IEEE, 2008, pp. 1–8, http://dx.doi.org/10.1109/IPDPS.2008.4536445.
[20] H.-C. Lin, C. Raghavendra, An approximate analysis of the join the shortest queue (JSQ) policy, IEEE Trans. Parallel Distrib. Syst. 7 (3) (1996) 301–307, http://dx.doi.org/10.1109/71.491583.
[21] A. Wolke, B. Tsend-Ayush, C. Pfeiffer, M. Bichler, More than bin packing: dynamic resource allocation strategies in cloud data centers, Inform. Syst. 52 (2015) 83–95, http://dx.doi.org/10.1016/j.is.2015.03.003.
[22] A. Beloglazov, J. Abawajy, 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, http://dx.doi.org/10.1016/j.future.2011.04.017.

[23] D. More, S. Mehta, P. Pathak, L. Walase, J. Abraham, Achieving energy efficiency by optimal resource utilisation in cloud environment, 2014 IEEE International Conference on Cloud Computing in Emerging Markets (2014) 1–8, http://dx.doi.org/10.1109/CCEM.2014.7015479.
[24] Z. Xiao, W. Song, Q. Chen, Dynamic resource allocation using virtual machines for cloud computing environment, IEEE Trans. Parallel Distrib. Syst. 24 (6) (2013) 1107–1117, http://dx.doi.org/10.1109/TPDS.2012.283.
[25] A. Mosa, N.W. Paton, Optimizing virtual machine placement for energy and SLA in clouds using utility functions, J. Cloud Comput. 5 (1) (2016) 17, http://dx.doi.org/10.1186/s13677-016-0067-7.
[26] Y. Ngoko, C. Cérin, P. Gianessi, C. Jiang, Energy-aware service provisioning in volunteers clouds, Int. J. Big Data Intell. 2 (4) (2015) 262–284, http://dx.doi.org/10.1504/IJBDI.2015.072171.
[27] E. Arianyan, H. Taheri, S. Sharifian, Novel heuristics for consolidation of virtual machines in cloud data centers using multi-criteria resource management solutions, J. Supercomput. 72 (2) (2016) 688–717, http://dx.doi.org/10.1007/s11227-015-1603-9.
[28] C. Mastroianni, M. Meo, G. Papuzzo, Probabilistic consolidation of virtual machines in self-organizing cloud data centers, IEEE Trans. Cloud Comput. 1 (2) (2013) 215–228, http://dx.doi.org/10.1109/tcc.2013.17.
[29] S. Vakilinia, B. Heidarpour, M. Cheriet, Energy efficient resource allocation in cloud computing environments, IEEE Access, https://doi.org/10.1109/ACCESS.2016.2633558.