Transactions on Machine Learning and Artificial Intelligence - Vol. 10, No. 6
Publication Date: December, 25, 2022
DOI:10.14738/tmlai.106.13419.
Jason, S. (2022). Real-time Virtual Machine Energy-Efficient Allocation in Cloud Data Centers Using Interval-packing Methods. Transactions on Machine Learning and Artificial Intelligence, 10(6). 15-34.

Real-time Virtual Machine Energy-Efficient Allocation in Cloud Data Centers Using Interval-packing Methods

Sebagenzi Jason
Department of Information Technology
AUCA University, Kigali 2461, Rwanda

ABSTRACT

The reduction of power consumption, which can lower the operation costs of Cloud providers, lengthen the useful life of a machine, as well as lessen the environmental effect caused by power consumption, is one of the critical concerns for large-scale Cloud applications. To satisfy the needs of various clients, Virtual Machines (VMs) as resources (Infrastructure as a Service (IaaS)) can be dynamically allocated in cloud data centers. In this research, we study the energy-efficient scheduling of real-time VMs by taking set processing intervals into account, with the providers' goal of lowering power consumption. Finding the best solutions is an NP-complete problem when virtual machines (VMs) share arbitrary amounts of a physical machine's (PM) total capacity, as demonstrated in numerous open-source resources. Our strategy treats the issue as a modified interval partitioning problem and takes into account configurations with dividable capacities to make the problem formulation easier and assist save energy. There are presented both exact and approximate solutions. The proposed systems consume 8–30% less power than the existing algorithms, according to simulation data.

Keywords: Cloud computing; Cloud data centers; resource scheduling; fixed processing intervals; modified.

INTRODUCTION

Based on a number of recent developments in virtualization, Grid computing, Web computing, utility computing, and related technologies, cloud computing is developing. Through the Internet or an internal network, cloud computing offers platforms and applications on demand [1]. Google App Engine [2], IBM Blue Cloud [3], Amazon EC2 [4], and Microsoft Azure [5] are a few examples of new cloud computing platforms. Software can be shared, allocated, and aggregated via cloud computing, together with computational and storage network resources. The concealment and abstraction of complexity, the efficient utilization of remote resources, and virtualized resources are a few of the major advantages of cloud computing. Since there are still many difficult problems to be solved, cloud computing is still regarded as being in its infancy [1,6-8]. Youseff et aldetailed .'s ontology of cloud computing's five main layers—cloud applications (SaaS), the cloud software environment (PaaS), cloud software infrastructure (IaaS), the software kernel, and hardware (HaaS)—is established in their paper published in Youseff et aljournal, .'s Cloud Computing, [9], and it is used to illustrate how these layers are related to one another and how they depend on earlier technologies. We concentrate on Infrastructure as a Service (IaaS) in cloud data centers in this study. There has been a lot of study done on cloud data center-related topics. The main problems and solutions in cloud

computing are outlined by Armbrust et al. [1]. Grid computing and Cloud computing are contrasted by Foster et al. [8]. By taking into account dynamic traffic models, Tian [10] introduces multi-dimensional algorithms for cloud data centers. One of the essential services in cloud computing is IaaS. Creating an on-demand resource management system for IaaS in cloud environments is crucial.

Regarding cloud architecture, Liu et al[11] .'s GreenCloud architecture intends to cut down on data center power usage while maintaining user-perceived performance. Using the suggestions made in its open-source incubator for Cloud standards, DMTF [12] concentrates on standardizing interactions between Cloud environments. The Eucalyptus open-source Cloud-computing system is introduced by Nurmi et al. [13]. A dynamic and integrated load-balancing approach for resource scheduling in Cloud data centers is proposed by Tian et al. [14].

Beloglazov et al. [6] provide a taxonomy and assessment of energy-efficient data centers and Cloud computing, and Garg et al. [15] introduce a Green Cloud framework for enhancing the carbon efficiency of Clouds. A state-of-the-art research study on Green Cloud computing is provided by Jing et al. [16], who also identify three hot research areas. The interrelationships between energy consumption, resource usage, and performance of aggregated workloads are studied by Srikantaiah et al. [17]. By condensing active activities, Lee et al. [18] offer two online heuristic algorithms for resource-efficient use in Cloud computing systems. In addition to reducing the overall number of migrations, Beloglazov et al. [19] investigate offline allocation of virtual machines (VMs) using updated best-fit bin-packing techniques. In a Xen virtualized system, Liu et alinvestigation .'s of performance and energy modeling for live VM migration and evaluation of models using five sample workloads. In order to automatically distribute resources and decrease the energy usage of web-service applications, Guazzone et al. [21] develop a two-level control approach. In order to provision VMs in Cloud data centers, Kim et al. [22] model a real-time service as a real-time VM request and employ dynamic voltage frequency scaling techniques. Real-time VM scheduling taking fixed processing intervals into account is still not well researched, unlike dynamic voltage frequency scaling approaches.

The scheduling of resources is crucial in cloud data centers. The allocation and migration of VMs with full life cycle limitations, which is sometimes overlooked, is one of the difficult scheduling difficulties in cloud data centers [23].

In order to address the aforementioned major concerns, we provide in this study a framework for real-time VM scheduling in IaaS that takes set processing intervals into account. The following are the key goals of this essay:

• Giving a unified picture to make managing many heterogeneous physical machines (PMs) and virtual machines (VMs) with diverse combinations easier. Through this single access point, administrators and users will then be able to more easily manage and keep an eye on their expanding collections of VMs.

• Taking into account the distribution of VMs with their set processing intervals (full life cycles). The majority of research studies frequently ignore this. The challenge becomes more challenging when critical capacity and real-time restrictions are taken into account.

• Using traditional interval scheduling and bin-packing approaches to create scheduling plans for offline and online environments. Our models take into account various intervals sharing a PM's total capacity during sometimes if the capacity restriction is satisfied, which sets them

apart from conventional interval scheduling problems (ISPs). Our models take into account the VM life cycle, which sets them apart from conventional bin-packing problems (BPPs).
• Theoretical proofs are used to create optimal and approximative energy-saving methods for the offline environment, where the scheduler is aware of every request in advance.
• For the online context, where the scheduler only knows the current requests, approximate techniques with delay and minimum migration are proposed in light of the observation that the total number of PMs and their time at power-on influence total power usage.
• Some current results that take into account the energy impact under various configurations and energy models are based on a single, basic set of VM and PM setups. Through simulations, we demonstrate how alternative setups and models might result in different energy consumption. To make the scheduling issue simpler and use less energy, the proposed divisible capacity arrangement of VMs and PMs is crucial.

This paper's remaining material is organized as follows: The suggested Green Cloud architecture and key elements for resource scheduling in Cloud data centers are presented in Section 6.2. The problem is stated in Section 6.3, which also introduces the suggested scheduling techniques. The performance evaluation of several algorithms in various setups is shown in Section 6.4. Related work on energy-efficient scheduling is presented in Section 6.5. Conclusion of Section 6.6.

GREENCLOUD ARCHITECTURE

Figure 6.1 suggests the layered design for GreenCloud. At the top layer, there is a web portal where users may choose resources and submit requests. In essence, it is a unified representation of the few preconfigured VM types available to users. After user requests are submitted, they move on to the CloudSched level, which is in charge of selecting the best data centers and PMs depending on user requirements. This layer has the capacity to control a sizable number of Cloud data centers made up of thousands of PMs. This layer allows for the use of various scheduling algorithms based on client attributes in various data centers. The lowest tier of the cloud contains resources like PMs and VMs, each of which has a specific allocation of CPU, memory, storage, and bandwidth. Virtual management is primarily in charge of keeping track of all VMs in the system, including their status, necessary capacities, hosts, arrival timings, and departure times, at the Cloud resource layer.

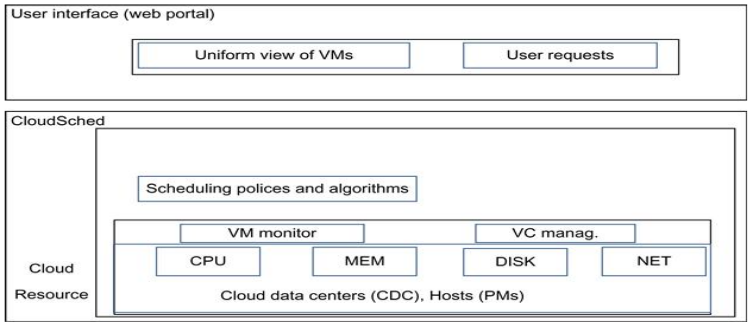


FIGURE 6.1 Proposed GreenCloud architecture.

Cloud data center resources

Only computational resources are taken into account in this work. The hosts (PMs) in a data center are in charge of controlling virtual machines throughout their life cycles. In a cloud, a

URL: http://dx.doi.org/10.14738/tmlai.106.13419

host is a component that stands in for a real computing node. It receives preconfigured RAM, storage, a scheduling policy for allocating VMs, and processor power (expressed, for example, in GHz or Million Instructions Per Second). A cluster or data center can also be created by connecting a number of hosts.

A uniform view for different types of VMs

There may be an excessive number of various kinds of PMs, VMs, and their mixtures. If a standard perspective is not offered, cloud service providers will encounter a vast number of issues. We demonstrate that it is feasible to have an unified view of various VM types using the frequently used example of Amazon EC2. The eight VM types as listed in the Amazon EC2 web documentation are shown in Table 6.1. Information on Amazon EC2's hardware setup is not available. On the basis of compute units, we can still create three different sorts of PMs. In a PM with 268.4 GB of RAM, for instance, in a genuine Cloud data center, It is possible to give 2 1690 GB storage and 16 3.25 unit cores. It is feasible to create a consistent understanding of several VM types in this or a related manner. For heterogeneous virtualization technologies (like Xen, KVM, and VMWare), this type of classification offers a standard picture of virtualized resources and has significant positive effects on managing and allocating VMs.

Table 3: 8 types of VMs in Amazon EC2

Memory (GB)	Compute units	Storage (GB)	API name	VM type
1.875	1 (1 cores×1 units)	211.25	m1.small	1-1(1)
7.5	4 (2 cores×2 units)	845	m1.large	1-2(2)
15.0	8 (4 cores×2 units)	1690	m1.xlarge	1-3(3)
17.1	6.5 (2 cores×3.25 units)	420	m2.xlarge	2-1(4)
34.2	13 (4 cores×3.25 units)	845	m2.2xlarge	2-2(5)
68.4	26 (8 cores×3.25 units)	1690	m2.4xlarge	2-3(6)
1.7	5 (2 cores×2.5 units)	422.5	c1.medium	3-1(7)
7.0	20 (8 cores×2.5 units)	1690	c1.xlarge	3-2(8)

Table 2: Divisible configuration of three types of PMs (an example)

PM type	CPU (compute units)	Memory (GB)	Storage (GB)
Type1	16	30.0	3380
Type2	52	136.8	3380
Type3	40	14.0	3380

Table 3. Random configuration of three types of PMs (an example)

PM type	CPU (compute units)	Memory (GB)	Storage (GB)
Type1	13	37.0	3000
Type2	53	137.8	4000
Type3	47	17.0	2000

Real-time VM request model

Because it makes use of virtualization, the cloud computing environment is an appropriate solution for real-time VM service [22]. Appropriate VMs are allotted when customers ask for the execution of their real-time virtual machines in a cloud data center.

Example 1

The interval vector vmRequestID can be used to denote a real-time VM request (VM typeID, start time, finish time, requested capacity). Figure 6.2's vm1(1, 0, 6, 0.25) indicates that the requested VM is of Type1 (equivalent to integer 1), with start times of 0 and 6, meaning that it completed at the sixth slot following start times of 0, and 25% of the entire capacity of Type1 PM. Similar representations can be used for other requests. Figure 6.2 depicts the life cycles of VM allocation utilizing two PMs and a slotted-time window arrangement, with PM#1 hosting VMs 1, 2, and 3 and PM#2 hosting VMs 4, 5, and 6. Keep in mind that each interval must adhere to the total capacity restriction.

URL: http://dx.doi.org/10.14738/tmlai.106.13419

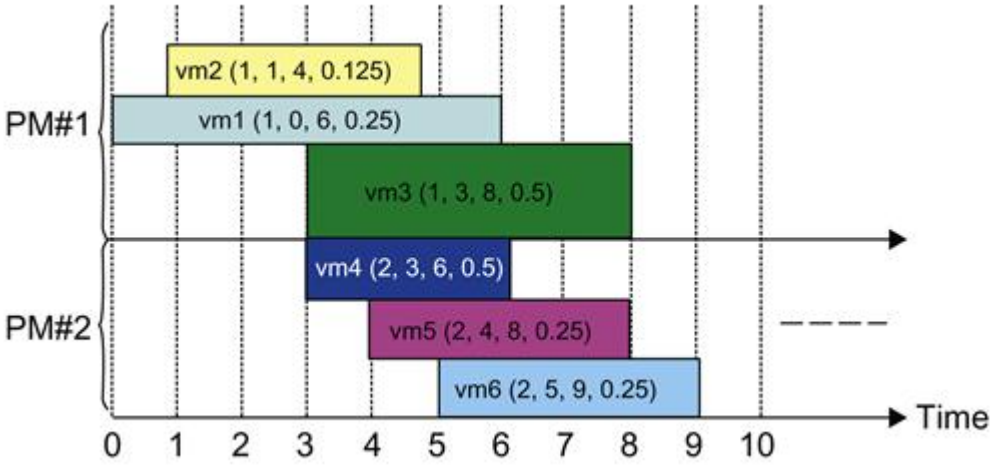


FIGURE 2. VM allocations using two PMs

Energy-efficient real-time scheduling

Problem description

We use a modified ISP as our model for the issue of real-time VM scheduling. In Ref. [24] and its references, you can find additional explanation and analysis regarding fixed ISPs. We outline a solution to the modified interval scheduling problem and compare the outcomes to well-known, current solutions.

a collection of requests (1, 2,..., n), where the ith request is linked to a capacity requirement (ci) and represents a time period beginning at si and ending at fi. Based on the following assumptions, the objective of energy-efficient scheduling is to satisfy all objectives with the least amount of running PMs and overall running times possible:

1. Except as otherwise noted, all data are deterministic, and time is formatted in slotted windows. Figure 6.3 illustrates how we divide the complete time interval [0,T] into slots of identical length (s0), where the sum of the slots is equal to T/s0 (in integer form). One slot's start time si and finish time fi are both integers. Afterward, a request's interval might be represented in slot format using (start time, finish time). For instance, if s0=5 min, then an interval [16,19] has a start time at the third slot and a finish time at the tenth slot, respectively. The actual time taken to complete this request is (103)5=35 minutes.
2. Each work stands alone. Other than what the start and finish times imply, there are no priority restrictions.
3. Each request's necessary capacity is a positive real value between 0 and 1. Keep in mind that a single PM's capacity is standardized to be 1. It can become multidimensional when taking into account various resources, such as CPU, memory, or storage, for instance.
4. Assume that each VM request is given to a single PM when it is processed. Therefore, unless otherwise specified, stopping a request and continuing it on another machine is not permitted (such as when using migration).
5. Every PM is constantly accessible (each machine is continuously available in the range [0,)).
6. Assume that each VM request uses the entire available capacity when it is assigned (the worst-case scenario). When a VM requests VMi=(CPU, memory,

storage)=[0.25,0.25,0.25] during interval [0,5], for instance, if the total capacity of a PM is normalized to be 1—all capacity of (CPU, memory, storage)=[1,1,1]—then the scheduler allocates 0.25 of the total capacity of a PM to VMi (i.e., the system assumes that the VMi occupies 25% of the total.

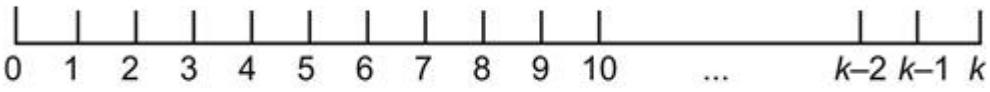


FIGURE 3. Time in slotted format

The following definitions are provided to aid in formally understanding the issue:
[Definition 1. Traditional interval scheduling with fixed processing time]. A collection of requests with the ith request corresponding to a period of time with the start and end points si and fi. Each request requires a 1 capacity (i.e., occupying the whole capacity of a machine during the fixed processing time).

[Definition 2. Interval scheduling with capacity sharing (ISWCS)]. The main distinction from conventional interval scheduling is that multiple requests may share a resource (namely, a PM) as long as the total capacity of all requests allotted to the resource at any given moment does not exceed the maximum capacity that the resource is capable of providing.

[Definition 3. Sharing compatible intervals for ISWCS]. The capacity of a PM can be shared by a subset of periods that have the complete required capacity but do not at any time exceed it.

The power model, the capacity configuration of VMs and PMs, and the power usage policies all have a significant impact on the energy consumption of all VMs and PMs. Here is an introduction to them.

The linear power consumption model of a server
The majority of the energy used in data centers is used by the cooling, networking, and disk storage systems. The authors of Ref. [25] offer the following power consumption model for blade servers:

$$P = 14.45 + 0.236U_{\text{cpu}} + (4.47E-8)U_{\text{mem}} + 0.0028U_{\text{disk}} + (3.1E-8)U_{\text{net}} \quad (6.1)$$

where Ucpu, Umem, Udisk, and Unet represent, respectively, the utilization of the CPU, memory, hard drive, and network interface. As can be observed, additional elements like RAM, hard drives, and network interfaces barely affect overall power consumption. The authors of Ref. [19] discover that CPU utilization is frequently proportional to the total load on the system and suggest the power model shown in Eq. (6.2):

$$P(u) = kP_{\max} + (1-k)P_{\max}u \quad (6.2)$$

where k is the percentage of power consumed by an idle server (studies show that on average it is roughly 70%), u represents CPU usage, and Pmax is the maximum power consumed when the server is fully employed. In comparison to other resources like memory, disk storage, and

URL: http://dx.doi.org/10.14738/tmlai.106.13419

network devices, the CPU accounts for the majority of energy usage, which is the subject of this paperr. We employ the power model specified in Eq. in this work (6.2). Eq. (6.3) is formed by further simplifying equation (6.2):

$$P = P_{\min} + (P_{\max} - P_{\min})u \tag{6.3}$$

where Pmin represents the power used by the given PM when its CPU use is nil (when the PM is idle without any VM running). Because workloads vary in a real setting, the CPU's utilization may fluctuate over time. Consequently, CPU usage is a function of time and is denoted by the symbol u. (t). As a result, Eq. (6.4) may be used to describe the total energy consumed by a PM (Ei) as an integral of the power consumption function over time:

$$E_i = \int_{t_0}^{t_1} P(u(t))\mathrm{d}t \tag{6.4}$$

Ei=P(u) (t1t0) if u(t) is a constant across time, such as if average utilization is adopted, where u(t)=u.

1. A cloud data center's overall energy usage is calculated using Equation (6.5):

$$E_{\mathrm{DC}} = \sum_{i=1}^{n} E_i \tag{6.5}$$

It represents the total energy used by all PMs. It should be noted that the energy usage of all VMs running on PMs is included.

2. During the testing period, all PMs had a total power-on time of:

$$\mathrm{Total_Time} = \sum_{i=0}^{..} \mathrm{PM}_i\mathrm{_Poweron_Time} \tag{6.6}$$

where PMi Power-on Time is the total time, the ith PM has been powered on.

For comparison, we'll suppose that each virtual machine uses all of the requested CPU resources.

If a PM's current CPU usage is u and it increases to u′ after creating a VM, the energy increase brought on by the VM is designated as Evm and may be calculated using Eq. (6.7), where u represents the rise in CPU utilization as a result of allocating a VM:

$$\begin{aligned} E_{\mathrm{vm}} &= P \times (t_1 - t_0) - P' \times (t_1 - t_0) \\ &= (P' - P) \times (t_1 - t_0) \\ &= (P_{\min} + (P_{\max} - P_{\min})u' - (P_{\min} - (P_{\max} - P_{\min})u)) \\ &(P_{\max} - P_{\min}) \times (u' - u) \times (t_1 - t_0) \\ &(P_{\max} - P_{\min}) \times \Delta u \times (t_1 - t_0) \end{aligned} \tag{6.7}$$

Formally, our task of scheduling VMs in real-time to reduce total energy consumption (Min iEi) becomes a multidimensional combinatorial optimization problem with constraint fulfillment (meeting capacity and time restrictions), making it an NP-complete problem [26,27].

Theorem 1
In a heterogeneous case, the decision version of a real-time VM scheduling problem is NP-complete.

Remark
Different kinds of PMs and VMs are taken into consideration in the case of a heterogeneous arrangement. The decision variant of this problem, which asks whether there is a workable schedule, is demonstrated to be NP-complete in Reference [23].

We take into account the heterogeneous case while mapping various VM types to corresponding PM types in order to reduce complexity (i.e., we simplify the heterogeneous case to a homogeneous case as given in Tables 6.1–6.3). The discussion and findings in the following are all predicated on a homogeneous instance.

Capacity configuration of VMs and PMs
Following our observation that alternative VM and PM capacity configurations affect the complexity of the issue and overall energy consumption, we will look at two distinct capacity combinations.

Random capacity configuration of VMs and PMs
The capabilities (CPU, memory, storage, and other parameters) of the VMs and PMs are set at random in this scenario, which is referred to as the random capacity case. This problem can be changed into a dynamic multidimensional BPP or interval-packing problem, which is known to be an NP-complete problem, if we need to take into account CPU, memory, storage, life cycles, and other aspects of VMs and PMs. See for example Refs. [5,26,27].

Lemma 1
According to the literature, there is no optimal method for minimizing the overall number of power-on PMs in offline scheduling in the random capacity situation.

Remark
Here, by taking CPU, memory, storage, life cycle, and other elements into account, our real-time VM scheduling problem may be converted into a conventional multidimensional (interval) BPP. It is simple to demonstrate that the real-time VM scheduling problem is an NP-complete problem by reducing a well-known NP-complete problem—the multidimensional BPP—to our issue [5,26,27]. Although there is no exact answer that can be found in polynomial time, there are certain approximations that can be made to reduce the overall number of power-on PMs.

Divisible capacity configuration of VMs and PMs
As shown in Tables 6.1 and 6.2, (CPU, memory, and storage) capacity is handled as a whole in this situation. For instance, the combined CPU, memory, and storage of the VM Types 1-1, 1-2, and 1-3, respectively, are 1/16, 1/4, and 1/2 of the combined CPU, memory, and storage of the PM Type 1. In a similar vein, the combined capacities of VM Types 2-1 and 2-2 are 1/8 and 1/4,

URL: http://dx.doi.org/10.14738/tmlai.106.13419

respectively, of the combined capacities of PM Type 2. The combined capacities of VM Type 3-1 and 3-2 are, respectively, 1/8 and 1/2 of the combined capacities of PM Type 3. The strongly divisible capacity case refers to this situation.

[Definition 4 Strongly divisible capacity of VMs and PMs]. The series of different capacities s1>s2>...>si>si+1>... assumed by VMs (the number of VMs for each capacity is arbitrary) is such that for any i>1, si+1 precisely divides si. This is known as a divisible sequence. If the capacities of the items in a list L form a divisible sequence, then we can say that the list has a divisible item capacity. Additionally, we state that the pair (L, C) is weakly divisible if L has divisible item capacities and strongly divisible if the biggest item capacity in L precisely divides the capacity C if L is a list of items and C is a PM's entire capacity [5].

Lemma 2
There is an ideal method for minimizing the total number of power-on PMs in offline scheduling in the situation of substantially divisible capacity.

Proof
The entire capacity of a VM in the case of strongly divisible capacity, as shown in Tables 6.1 and 6.2, equals (1/16, 1/8, 1/4, or 1/2) of the total capacity of a PM, and the capacities of all VMs belonging to the same type form a strongly divisible sequence. As a result, our real-time VM scheduling issue may be reduced to a standard one-dimensional interval-packing issue, or BPP. For offline scheduling, the First-Fit Decreasing (FFD) or Best-Fit Decreasing (BFD) method yields the best result [5]. Additionally, the issue in this instance may be turned into an ISP, demonstrating that there are sufficient PMs to accommodate all requests [27].

Lemma 3
The asymptotic worst case approximation ratio for minimizing the total number of power-on PMs in online scheduling in the substantially divisible capacity situation is 2.384 (in comparison to the offline optimal solution).

Remark
The proofs provided in Coffman et al. [28] can be used to achieve the proof of Lemma 3.
Numerous previous findings are applicable because, in the strongly divisible capacity situation, the multidimensional problem (taking into account CPU, memory, storage, and other parameters) is reduced to a one-dimensional BPP or interval-packing problem. On this, the discussion and simulation that follow are predicated.

The power usage policies
Strategy one: idle servers turned off
Assume a PM is turned off during a testing period when it is idle to reduce energy usage. During the testing phase, it is possible that each PM will be turned on and off numerous times.

Example 2
Given a testing period of 0, 1000, a PM is turned on in three intervals of 2, 100, 209, 235, and 7,89, 1000, with average utilization of 0.5 for each interval, Pmax of 300 W, and Pmin of 200 W, respectively. This results in a total power-on time for the PM of 335 slots: 100, 2, 235, 2, 09,

1000, 7,89, and 1000. The total energy consumption is (200+(300200)0.5)3355/60/1000=6.979 kW h if each slot lasts for 5 minutes.

Lemma 4
When turning off idle servers, reducing the overall PM count does not necessarily entail reducing the overall energy use.

Proof
It is clear from the linear power model of Eqs. (6.3)–(6.7) that a PM's total energy consumption depends on both its average utilization and the duration of its power-on times, rather than just on the overall number of PMs employed. The total power usage also depends on the aggregate power-on time of all PMs when VM requests are the same and, consequently, average utilization of all PMs is the same. Therefore, the goal of reducing overall energy consumption is to reduce both the total number of PMs used and the amount of time those PMs are powered on.

Remark
Minimizing the aggregate busy time of all machines (PMs) is an NP-complete problem [26,27]. Since there is currently no known optimal scheduling method in this situation, we suggest an approximation algorithm termed the modified interval scheduling algorithm (MFFI).

Strategy two: idle servers not turned off
In this scenario, idle servers are not turned off but can be put into sleep mode to conserve energy while still taking into account server dependability and preventing too many server transitions or traffic vibrations.

In Example 2, Assume that servers are always on and use Pmin of power while they are idle. The average usage is 3350.5/1000=0.1675, and the total power-on time is 1000 slots. The overall energy consumption is equal to 18.06 kW h or (200 (300200)0.167510005/60/1000). From Example 2, it is clear that various approaches to dealing with idle servers can have an impact on the overall energy consumption.

Lemma 5
If idle servers are not turned off, the task of reducing overall energy usage can be reduced to finding the smallest number of PMs.

Proof
(1). Assume that during the scheduling process, idle servers are turned on (albeit they can be put into sleep mode). Since the Pmin values are the same, Eqs. (6.3)–(6.7) show that the total energy consumption is dependent on average utilization and operating times. (2). Ei should be set to (Pmin+(PmaxPmin)u)t=(+ui)t, where ui represents the average PMi consumption. If there are m homogenous PMs, then E=iEi=mt+(iui)t represents the overall energy consumption. (3). The total time t (the period of time from start-up to the present) is the same for all scheduling if the number of PMs used in the two scheduling solutions is the same. *The same amount of VM requests, which equals the same utilization for all PMs, explains why the β(Σiui)t is the same.* (4). If two scheduling processes use different numbers of PMs—say scheduling#1 uses *m* homogeneous PMs and scheduling#2 uses *m+1* homogeneous PMs—then the only difference is $m\alpha t$, $(m+1)\alpha t$, obviously $m\alpha t < (m+1)\alpha t$ when $\alpha t > 0$ (which is true). This

URL: http://dx.doi.org/10.14738/tmlai.106.13419

means that employing more PMs will result in a higher overall energy consumption (notice that this is not true if assumption is that idle servers are turned off). The evidence is now complete.6.3.2 Four offline and online scheduling algorithms

We suggest the following four offline and online scheduling algorithms:
[Definition 5. Modified Interval Partitioning First-Fit Algorithm (MFF)]. The requests are given to the algorithm in any sequence. Each request is made with the intention of being placed in the first PM (with the lowest index) that can handle it. It launches a new PM and installs the VM on it if no nonempty PMs are discovered. It should be noted that MFF is online with regard to VM requests because it doesn't use any data from requests that come after the one being processed.

[Definition 6. Modified Interval Partitioning First-Fit Increasing Algorithm (MFFI)]. Before applying MFF to the VM requests for the ISWCS problem, the requests are sorted in order of rising start times. An offline scheduling algorithm is MFFI. Figures 6.4 and 6.5 show the pseudo code of MFF and MFFI algorithms, respectively, also called ONWID and OFWID in this paper.

Lemma 6
The MFFI algorithm, as depicted in Figure 6.5, has a temporal complexity of O(n max(m, logn)), where n is the number of VM requests and m is the number of PMs.

Remark
Lemma 6's proof is simple if you use the pseudocode in Figure 6.5.

Lemma 7
The MFF algorithm in Figure 6.4 has an O(nm) time complexity, where n is the number of VM requests and m is the number of PMs.

Remark
The proof for *Lemma 7* is straightforward, so details are omitted.
[Definition 7. Offline MFFI with delay (OFWD)]. Considering that some requests may conflict with one another in terms of time and capacity constraints, delaying the start of some requests may result in fewer PMs overall. This algorithm is largely the same as MFFI (OFWID), however OFWD can employ MFFI after delaying the beginning of VM queries for a period of time (with a threshold).

[Definition 8. Online MFF with delay and migration (ONWD)]. Similar to ONWID, but with the ability to move VMs between PMs and postpone some VM start times. The overall number of PMs and their power-on times can be decreased thanks to this. Only when the system's overall burden is low does migration occur, which involves selecting VMs from the PMs with the lowest (or second-lowest) loads and moving them to other PMs using the MFF method.

Input: VM requests (indicated by their required VM type ID, start time, finish time, and requested capacity), interval of start time and finish time of request *i* is denoted as I_i
Output: Assign a PM ID to each request and allocate an interval for each request.

```
1.  d=0;
2.  for j = from 1 to n do
3.      foreach I_i that precedes I_i
4.          if they are not overlapped or overlapped but still
    can share resources of an allocated PM do
5.              allocate I_i to the PM
6.          else
7.              start a new PM;
8.             d=d+1;
9.             allocate I_i to PM d
10.         endif
11.     endforeach
12. endfor
```

FIGURE 6.4 Modified interval partitioning first-fit algorithm (MFF)

Input: VM requests indicated by their (required VM type IDs, start times, ending times, requested capacity), the interval of start time and ending time of request *i* is denoted as I_i
Output: Assign PM IDs to all requests and allocated intervals for all requests

```
1.  Sort intervals by start-times in increasing order,
    breaking ties arbitrarily
2.    Let I_1, I_2.... I_n denote the intervals in this order
3.    d=0;
4.  for j = from 1 to n do
5.   foreach I_i that precedes I_i in sorted order
6.   if they are not overlapped or overlapped but still
    can share resources of an        allocated PM do
7.        allocate I_i to the PM hosts I_i
8.   else
9.     start a new PM; d=d+1; allocate I_i to PM d
10.  end
11.  endeach
12. endfor
```

FIGURE 6.5 Modified interval partitioning first-fit increasing (MFFI)

PERFORMANCE EVALUATION

This section explains how to compare several scheduling algorithms; methodology, metric, algorithm, and outcome analyses are offered.

URL: http://dx.doi.org/10.14738/tmlai.106.13419

Methodology

There is currently no tool available that can be used to carry out the comparisons suggested in this chapter. As a result, a java discrete simulator is put into practice for comparisons. To all comparison algorithms, the same set of inputs (VM requests) is applied. A program creates the collection of inputs before writing them into a text file. Then, while online algorithms read one record (request) at a time, offline algorithms employ all of these inputs simultaneously. All simulation results are based on the VM and PM configurations with divisible capacities that are listed in Tables 6.1 and 6.2.

Metrics

1. All simulations use the linear energy model (Eqs. (6.3)–(6.7)), however our computational findings hold for any convex power function.
2. The entire amount of energy used by a cloud data center.
3. The total number of PMs that are turned on for the duration of the test.
4. The total amount of time that each PM was powered on during the testing period.

Algorithms

In Section 6.3, the four potential methods are previously described. These are the other two algorithms:

Round Robin (Round): One of the most popular scheduling methods, the Round Robin assigns VM requests to each PM in turn (e.g., by Eucalyptus [13] and Amazon EC2 [4]). The simplicity of this algorithm's implementation is a benefit.

*Modified Best-Fit Decreasing (MBFD)***:** A bin-packing algorithm is the MBFD algorithm. For the one-dimensional BPP, it is demonstrated that BFD uses no more than 11/9 OPT +1 bins (where OPT is the number of bins provided by the best solution). The MBFD algorithm initially arranges all of the virtual machines (VMs) in decreasing order of CPU usage before allocating each VM to the host with the lowest increase in power consumption as a result of the assignment.

Inputs settings and results analysis

We regard the VM and PM configurations in Tables 6.1 and 6.2 to be strongly divisible capacity scenarios. For various types of PMs, Table 6.4 also provides a distinct Pmin and Pmax.

Table 6.4. Three types of PMs with power consumptions

PM type	CPU (compute units)	Memory (GB)	Storage (GB)	P_{min} (W)	P_{max} (W)
Type1	16	30.0	3380	210	300
Type2	52	136.8	3380	420	600
Type3	40	14.0	3380	350	500

For comparison, we consider that all virtual machines (VMs) occupy the entire specified capacity (as the worst-case scenario). Based on Amazon EC2, eight different types of virtual machines (VMs) are taken into consideration in this situation. Each type of VM has an equal

amount of arrivals (requests), which total 1000. (i.e., 125). All queries have exponential service times and follow the Poisson arrival pattern. The maximum intermediate period, the mean inter-arrival interval, and the maximum request duration are all set at 50, 100, 200, 400, and 800 slots, respectively. Each slot is equal to 5 min. For instance, if a virtual machine (VM) has 20 slots of required service time, its real length is 20 x 5 or 100 minutes. Simulations are conducted six times for each set of inputs (requests), and all of the results presented in this chapter are the average of the six iterations. The maximum delay is capped at 50 slots.

Assuming idle servers turned off

Total energy usage is shown in Tables 6.5 and 6.6, assuming that idle servers are turned off and idle servers are not, respectively. Only the online scheduling algorithm ONWD applies migration; the total number of migrations is listed in the last column of both tables.

Table 6.5. Total energy consumption (idle servers turned off)

Total energy consumed in a DC (kW h)	RR	OFWID	OFWD	MBFD	ONWID	ONID	Migrations
maxdur.=50	655.6	465.9	438.4	495.1	476.0	459.8	0
maxdur.=100	1210.7	813.3	781.7	890.8	824.7	796.8	1
maxdur.=200	2312.1	1478.3	1444.4	1620.0	1492.3	1458.8	4
maxdur.=400	4011.4	2731.3	2676.0	2957.3	2762.3	2708.3	12
maxdur.=800	7508.2	5190.9	5117.2	5559.6	5209.8	5168.9	21

URL: http://dx.doi.org/10.14738/tmlai.106.13419

Table 6. Total energy consumption (idle servers NOT turned off)

Total energy consumed (kW h)	Round	ONWID	OFWID	MBFD	OFWD	ONWD	Migrations
maxdur.=50	2793.4	1918.4	1918.4	1918.4	1305.9	1305.9	0
maxdur.=100	3534.5	2790.7	2659.5	2397.0	1784.5	1784.5	2
maxdur.=200	5029.9	3542.4	3411.2	3411.2	3017.4	3017.4	5
maxdur.=400	8353.4	6034.7	6034.7	6034.7	5290.9	5290.9	16
maxdur.=800	16918.7	9962.5	9831.2	9568.7	9568.7	9568.7	30

While all other parameters remain the same, Figure 6.6 displays the overall energy usage (in percentages relative to Round Robin) for the six algorithms as the maximum length of VMs varies from 50 to 800 slots. As a benchmark, Round Robin's total energy consumption of 100% is used to compare the total energy consumption of other algorithms to. Then Round Robin's energy usage is compared to that of other algorithms. For all cases, Round>MBFD>ONWID>OFWID>ONWD>OFWD for the total energy consumption calculated using Equations 6.3–6.7. Compared to MBFD and Round Robin, ONWID, ONWD, OFWD, and OFWID often use 2-8% less power each. Because ONWID and OFWID use the best available capacity sharing and require less total power-on time, they outperform MBFD in terms of performance.

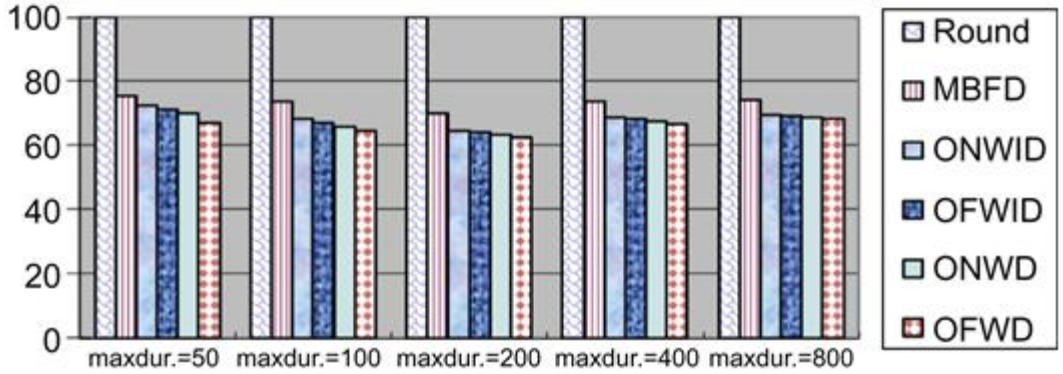


FIGURE 6.6 Total energy consumption comparing to Round Robin (%) when varying maximum duration of VM requests (idle servers turned off).

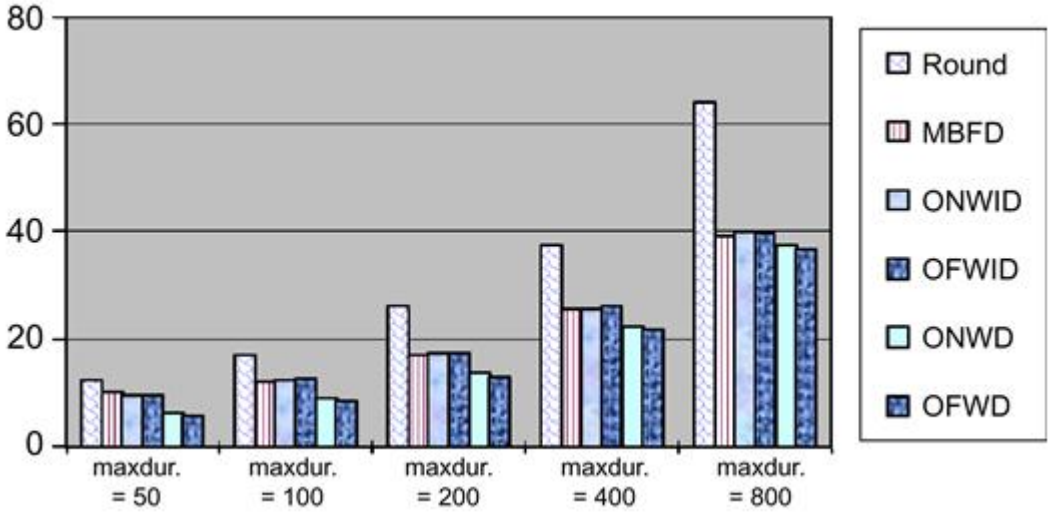


FIGURE 6.7 Total number of PMs used (idle servers turned on)

Figure 6.8 demonstrates the cumulative power-on time of six algorithms (in minutes). Keep in mind that if a PM stays idle for a certain amount of time, it is always assumed to be turned off in simulations. Using Eq, the total power-on time for all PMs is calculated (6.6). As can be shown, Round is always superior to MBFD, ONWID, OFWID, ONWD, and OFWD in terms of total power-on time. This explains why the overall pattern of energy use is compatible with the predictions made by theory and Lemma 3's proof.

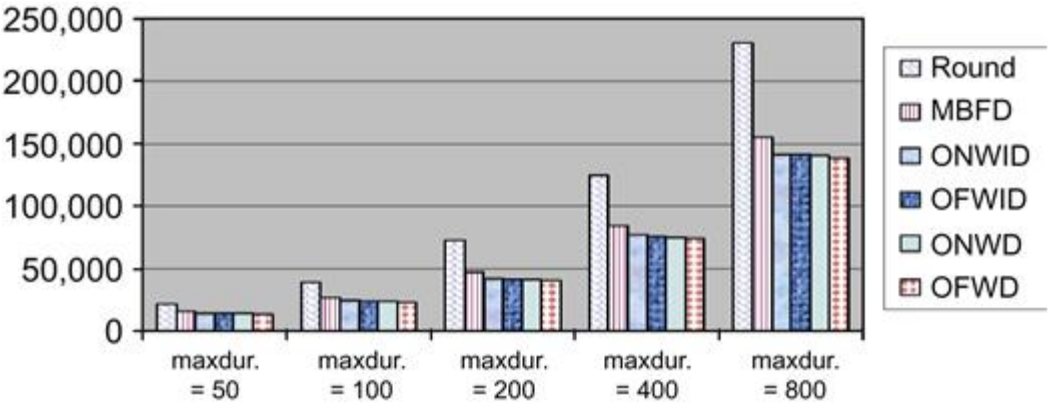


FIGURE 6.8 Total power-on time (minutes) of all PMs (idle servers turned off).

Assuming idle servers not turned off

While all other parameters remain the same, Figure 6.9 displays the overall energy usage (in percentages relative to Round Robin) for the six algorithms as the maximum length of VMs varies from 50 to 800 slotsRound>ONWID>OFWID>MBFD>OFWD>ONWD with regards to total energy consumption as determined by Equations (6.3)- (6.7). Compared to MBFD and Round Robin, ONWD and OFWD use, respectively, around 5-20% and 40% less electricity. The adoption of delays or migrations results in a decrease in the overall number of PMs, which is why ONWD and OFWD function better. The total number of PMs employed across various algorithms is shown in Figure 6.10. These findings support the theoretical findings and Lemma 4's proof.

URL: http://dx.doi.org/10.14738/tmlai.106.13419

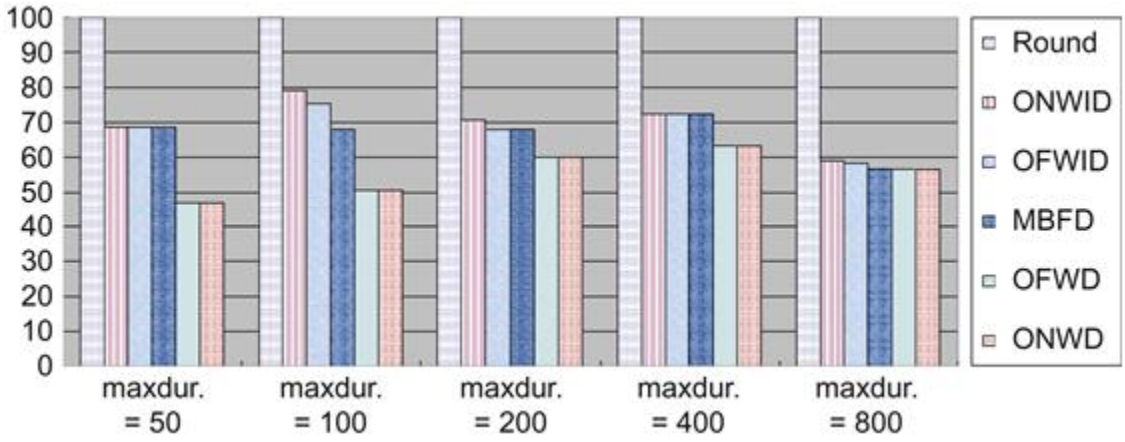


FIGURE 6.9 Total energy consumption comparing to Round Robin in percentages (idle servers not turned off)

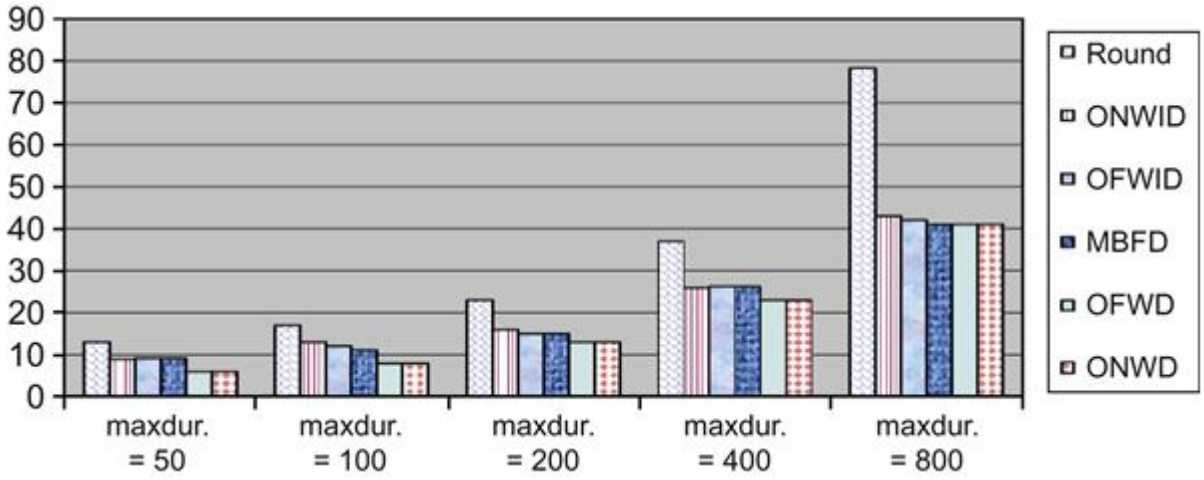


FIGURE 6.10 Total number of PMs used (idle servers not turned off)

Impact of varying the total number of VM requests

Additionally, we varied the total number of VM requests while fixing the total number of each sort of PM. The average arrival rate divided by the average service rate was used to calculate the system load. The same outcomes were seen as before. These outcomes are not presented here due to page restrictions.

RELATED WORK

Consideration of the allocation and migration of VMs with full life cycle constraints, which is frequently disregarded, is one of the difficult scheduling difficulties in cloud data centers [23]. Offline allocation of VMs using updated best-fit bin-packing techniques is a topic Beloglazov et al. [19] consider. A real-time service is modeled by Kim et al. [22] as a real-time VM request, and dynamic voltage frequency scaling techniques are used. Load balance and energy efficiency are combined by Matthew et al. [29] who also suggest an ideal offline method and an ideal online algorithm. Constrained mixed-integer programming is how Rao et al. [30] models the issue and suggests a rough resolution. In order to reduce overall costs, Lin et al. [31] presents online and offline algorithms for data centers that turn off unused servers.

CONCLUSIONS

In this paper, we offer updated interval partitioning models-based scheduling techniques that take into account the entire life cycle of heterogeneous VM types. We have demonstrated that by using these scheduling strategies, cloud data centers can use less energy overall. Additional research in the following areas needs to be done:

1. Energy usage during migration transitions being taken into account
2. The gathering and examination of energy usage information from actual cloud data centers
3. The combination of load balance, energy efficiency, and other features

Our upcoming research will look into scheduling plans that take these factors into account.

References

[1]. Armbrust M, Fox A, Griffith J, et al. *Above the Clouds: a Berkeley view of Cloud computing* Berkeley: EECS Department, University of California; 2009.

[2]. Google App Engine. <http://code.google.com/intl/zh-CN/appengine/>.

[3]. IBM (2007) blue cloud, <http://www.ibm.com/grid/>.

[4]. Amazon EC2. <http://aws.amazon.com/ec2/>.

[5]. Microsoft Inc., Windows Azure, <http://www.microsoft.com/windowsazure>; December 2013.

[6]. Beloglazov A, Buyya R, Lee YC, Zomaya A. In: Amsterdam, The Netherlands: Elsevier; 2011;Zelkowitz M, ed. *A taxonomy and survey of energy-efficient data centers and Cloud computing systems, advances in computers*. vol. 82 ISBN: 978-0-12-385512-1.

[7]. Boss G, et al. Cloud computing, IBM Corporation white paper, <http://download.boulder.ibm.com/ibmdl/pub/software/dw/wes/hipods/Cloud_computing_wp_final_8Oct.pdf >; November 2007.

[8]. Foster I, et al. Cloud computing and grid computing 360-degree compared [R]. IEEE International Workshop on Grid Computing Environments (GCE) 2008, co-located with IEEE/ACM Supercomputing, 2008.

[9]. Youseff L, et al. Toward a unified ontology of Cloud computing. In: The proceedings of grid computing environments workshop, GCE'08, 2008.

[10]. Tian WH. Adaptive dimensioning of Cloud datacenters. In: The proceedings of IEEE the eighth international conference on dependable, autonomic and secure computing (DASC-09), Chengdu, China; December 12–14, 2009.

[11]. Liu L, Wang H, Liu X, et al. *GreenCloud: a new architecture for green data center*. *Proceedings of the sixth international conference industry session on autonomic computing and communications industry session, ICAC-INDST'09* New York, NY: ACM; 2009; p. 29–38.

[12]. Distributed Management Task Force Inc., Interoperable Clouds: A white paper from the open cloud standards Incubator, <www.dmtf.org/about/policies/disclosures.php>; November 2009.

[13]. Nurmi D, et al. The Eucalyptus open-source Cloud-computing system. In: Proceedings of ninth IEEE international symposium on cluster computing and the grid, Shanghai, China, 2008.

[14]. Tian WH, Zhao Y, Zhong YL, Xu MX, Jing C. A dynamic and integrated load-balancing scheduling algorithm for Cloud data centers. In: the proceedings of CCIS 2011, Beijing.

[15]. Garg SK, Yeo CS, Buyya R. GreenCloud framework for improving carbon efficiency of Clouds. In: Proceedings of the 17th international European conference on parallel and distributed computing (EuroPar 2011, LNCS, Springer, Germany), Bordeaux, France; August 29–September 2, 2011.

[16]. Jing S, Ali S, She K, Zhong Y. State-of-the-art research study for green Cloud computing. *J Supercomputing, Special Issue on Cloud Computing 2011.* 2013;65(1):445–468.

URL: http://dx.doi.org/10.14738/tmlai.106.13419

[17]. Srikantaiah S, Kansal A, Zhao F. Energy aware consolidation for Cloud computing. In: Proceedings of the 2008 conference on power aware computing and systems.

[18]. Lee Y, Zomaya AY. Energy efficient utilization of resource in Cloud computing systems. *J Supercomput*. May 2012;60(2):268–280.

[19]. Beloglazov A, Abawajy J, Buyya R. Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing. *Future Gener Comput Syst*. May 2012;28(5):755–768.

[20]. Liu H, Xu C, Jin H, Gong J, Liao X. Performance and energy modeling for live migration of virtual machines. In: The proceedings of HPDC'11, June 8–11, San Jose, CA; 2011.

[21]. Guazzone M, Anglano C, Canonico M. Energy-Efficient Resource Management for Cloud Computing Infrastructures. In the proceedings of CloudCom, 2011.

[22]. Kim K, Beloglazov A, Buyya R. Power-aware provisioning of virtual machines for real-time Cloud services, Concurrency and Computation: Practice and Experience, vol. 23, Number 13, New York, NY: Wiley Press; September 10, 2011. p. 1491–505. ISSN: 1532-0626.

[23]. Kolen AWJ, Lenstra JK, Papadimitriou CH, Spieksma FCR. Interval scheduling: a survey, Published online 16 March 2007 in Wiley InterScience (<www.interscience.wiley.com>).

[24]. Kovalyov MY, Ng CT, Cheng E. Fixed interval scheduling: models, applications, computational complexity and algorithms. *Eur J Operational Res*. 2007;178(2):331–342.

[25]. Economou D, Rivoire S, Kozyrakis C, Ranganathan P. Full-system power analysis and modeling for server environments, 2006. Stanford University/HP Labs workshop on modeling, benchmarking, and simulation (MoBS) June 18, 2006.

[26]. Garey R, Johnson DS. *Computing and intractability: a guide to the theory of NP-completeness* San Francisco, CA: W.H. Freeman; 1978.

[27]. Kleinberg J, Tardos E. *Algorithm design* Pearson Education 2005; ISBN: 0321295358.

[28]. Coffman Jr EG, Garey MR, Johnson DS. Bin-packing with divisible item sizes. *J Complexity*. 1987;3:406–428.

[29]. Mathew V, Sitaraman RK, Shenoy P. Energy-aware load balancing in content delivery networks. In: The proceedings of INFOCOM, 2012.

[30]. Rao L, Liu X, Xie L, Liu WY. Minimizing electricity cost: optimization of distributed Internet data centers in a multi-electricity-market environment. In INFOCOM, 2010.

[31]. Lin M, Wierman A, Andrew LLH, Thereska E. Dynamic right- sizing for power-proportional data centers. In: Proceedings of the IEEE INFOCOM, Shanghai, China; 2011. p. 10–5.