Journal of Cloud Computing: Advances, Systems and Applications

RESEARCH Open Access

Energy-aware VM placement algorithms for the OpenStack Neat consolidation framework

Fikru Feleke Moges[1,2*] and Surafel Lemma Abebe[2]

Abstract

One of the main challenges in cloud computing is an enormous amount of energy consumed in data-centers. Several researches have been conducted on Virtual Machine(VM) consolidation to optimize energy consumption. Among the proposed VM consolidations, OpenStack Neat is notable for its practicality. OpenStack Neat is an open-source consolidation framework that can seamlessly integrate to OpenStack, one of the most common and widely used open-source cloud management tool. The framework has components for deciding when to migrate VMs and for selecting suitable hosts for the VMs (VM placement). The VM placement algorithm of OpenStack Neat is called Modified Best-Fit Decreasing (MBFD). MBFD is based on a heuristic that handles only minimizing the number of servers. The heuristic is not only less energy efficient but also increases Service Level Agreement (SLA) violation and consequently cause more VM migrations. To improve the energy efficiency, we propose VM placement algorithms based on both bin-packing heuristics and servers' power efficiency. In addition, we introduce a new bin-packing heuristic called a Medium-Fit (MF) to reduce SLA violation. To evaluate performance of the proposed algorithms we have conducted experiments using CloudSim on three cloud data-center scenarios: homogeneous, heterogeneous and default. Workloads that run in the data-centers are generated from traces of PlanetLab and Bitbrains clouds. The results of the experiment show up-to 67% improvement in energy consumption and up-to 78% and 46% reduction in SLA violation and amount of VM migrations, respectively. Moreover, all improvements are statistically significant with significance level of 0.01.

Keywords: Virtual machine consolidation, Virtual machine placement, Bin packing, OpenStack, OpenStack neat

Introduction

Cloud computing refers to the provisioning of computing capability as a service through the Internet. The hardware and systems software that together provide those services are referred to as a cloud [1]. Cloud providers leverage virtualization technology to provide on demand computing resources in form of virtual machines (VMs). VMs have their own operating system to manage applications. Containerization is an alternative technology which partition a server hardware into many containers that share an operating system [2].

As a cloud is realized on large-scale usually distributed data-centers, it consumes an enormous amount of energy. In 2012, energy consumption by data centers worldwide was 300 - 400 tera-watt hour, about 2% of the global electricity usage, and it is estimated to triple by 2020 [3]. VM consolidation using live VM migration [4] optimizes power utilization by running VMs in as much few servers as possible and putting the rest in sleep mode or turning them off. The research by Meisner et al. shows that turning off servers or putting them in a sleep mode saves a large amount of power [5]. According to their study, a typical HP blade server consumes 450 w at peak load, 270 w at idle state and 10.4 w at sleep mode.

Energy efficiency usually has a trade-off with quality of service which is another concern of consolidation. From the cloud customer point of view, all that matter is the fulfillment of their applications resource demand. The resource demand is usually specified as Service Level Agreement (SLA). Thus, any good consolidation algorithm should provide a well-balanced energy efficiency and SLA assurance. In addition to these concerns, there

*Correspondence: fikreyes05@yahoo.com
[1]Ethio Telecom, Addis Ababa, Ethiopia
[2]School of Electrical and Computer Engineering, Addis Ababa Institute of Technology, Addis Ababa University, Addis Ababa, Ethiopia

is a third aspect of consolidation that deals with minimizing the amount of VM migrations [6]. Unnecessary VM migrations need to be avoided as it increases the network traffic and also incurs additional cost of energy [7, 8].

To address the above consolidation issues, a number of research works are conducted [6, 9–14]. Not all researches deal with all aspects of consolidation. For example, the work in [12] particularly focuses on energy minimizing aspect of consolidation while the works in [9, 10] address minimizing SLA violation as well. Some of the works [6, 11, 13, 14] deal with all three aspects of consolidation including reducing the amount of VM migrations.

Beloglazov and Buyya (2014) developed a framework and an open-source implementation of dynamic VM consolidation for *OpenStack* cloud [14]. OpenStack is one of the most common and widely used open-source cloud management tool [15–17]. The framework is called *OpenStack Neat*. The implementation of the framework runs independently of the base OpenStack and applies the consolidation process by invoking public APIs of OpenStack. The architecture of OpenStack Neat contains four decision components:

1 Host *overload detection*: used to decide whether a host is overloaded.
2 Host *underload detection*: used to decide whether a host is underloaded.
3 *VM selection*: selects the VMs to be migrated from overloaded hosts.
4 *VM placement*: selects a host for placing the current VM to be migrated.

The VM placement algorithm of OpenStack Neat implementation is called Modified Best-Fit Decreasing (MBFD) [14]. The MBFD selects an active host with the minimum available CPU that fits the current VM. In case of a tie, the host with the smallest available RAM is chosen. Since the algorithm is based on the best-fit decreasing heuristic, it has high efficiency of consolidating VMs to a smaller number of servers. It reduces energy consumption by turning off or putting in sleep mode the rest of the servers. However, the algorithm has some drawbacks: (i) placing VM to the most utilized host increase the overload probability. The overload will intern increases SLA violations and the number of VM migrations, and (ii) in a heterogeneous cloud, the MBFD algorithm loses the benefit of favoring power-efficient servers and will be less energy efficient.

In this paper, we address the limitation of MBFD; thereby proposing an enhancement to the OpenStack Neat consolidation to improve energy efficiency, lower VM migrations and SLA violation. To improve the energy efficiency, we propose a VM placement algorithms by modifying the bin-packing heuristics considering power efficiency of hosts. Moreover, we introduce a new bin-packing heuristic, medium-fit, to reduce SLA violations and the number of VM migrations. To evaluate the proposed algorithms, we have conducted an experiment using CloudSim simulator on three cloud data-center scenarios: homogeneous, heterogeneous and default. Twenty days of workloads that run on the three data-center scenarios have been generated from traces of PlanetLab and Bitbrains clouds [18, 19]. The proposed algorithms improve all three consolidation aspects: energy efficiency, SLA violation and amount of VM migrations. The results of the experiment show up-to 67% improvement in energy consumption and up-to 78% and 46% reduction in SLA violation and amount of VM migrations, respectively.

The main contributions of this paper are the following:

- A new bin-packing heuristic called a medium-fit is defined. The medium-fit heuristic provides a well-balanced efficiency between energy consumption and SLA violation.
- In the proposed algorithms, the bin-packing heuristics are modified to consider the power efficiency of the servers. The modification has resulted in a better energy efficiency when used in heterogeneous cloud.
- The proposed algorithms can be implemented with a minor addition to a cloud configuration database. The peak power of hosts is the only additional information necessary to incorporate the proposed algorithms in the implementation of OpenStack Neat framework.

The remainder of this paper is organized as follows. In “Related work” section, we discuss the related works. The proposed VM placement algorithms are described in “Proposed work” section. In “Experiment and results” section, we present the experiment setups and discuss the result of the simulation. Finally, we conclude the paper in “Conclusion” section.

Related work

The problem of consolidation is well formulated in the works of Guazzone et al. (2012) as an optimization function [6, 20]. The objective function is a linear combination of cost of energy, VM migration and cost of performance degradation. The resulting mathematical programming is a Mixed-Integer Nonlinear Program (MINLP) which is known to be NP-hard. The only known solution to an NP-hard problem is an exhaustive search which is infeasible for dynamic cloud environment owing to its slow convergence. Some research works give a different formulation of the consolidation problem. For example, Rawas et al. (2018) formulate the consolidation problem in context of Geo-distributed data-centers [21]. Accordingly their equation considers power-effectiveness of data-centers and users to data-center communication costs in additional to cost of energy in each data-center. In all cases, the optimum solution (exhaustive search) is infeasible owning to its slow convergence.

Several approximate consolidation solutions are proposed in the literature [6, 14, 18, 22–24]. In these approximate solutions, the VM placement decision is handled with simple heuristics such as a modified form of best-fit and first-fit decreasing.

In the works of Beloglazov et al. (2014), the VM placement problem is handled by the MBFD algorithm [14]. The algorithm deals with minimizing the number of active servers and is based on a bin-packing heuristic called Best-Fit Decreasing (BFD). The default VM placement algorithm in CloudSim cloud simulator is the Power-Aware Best-Fit Decreasing (PABFD) [18]. The PABFD places the current VM on a host that fits it and the estimated increase in power is the minimum. Chowdhury et al. (2015) proposed Power-Aware Worst-Fit Decreasing (PAWFD) algorithm which favors a host whose estimated increase in power utilization is the maximum (quite the opposite of PABFD) [22]. Their experiment shows that PAWFD has better performance than their baseline algorithm, PABFD.

A comprehensive performance analysis of various VM placement algorithms is conducted by Z. Mann and M. Szabo (2017) [23]. For overload and underload detection, the authors reuse algorithms from OpenStack Neat framework. The VM placement algorithms considered for comparison include PABFD and PAWFD. The best performing algorithms are the "Guazzone" [6] and the "Shi-AC" [25] algorithms. The "Guazzone" algorithm is from the works of Guazzone et al. (2012) [6] and it applies three host selection criteria: (i) powered-on host proceeds powered-off host, (ii) within powered-on or powered-off host category, hosts are selected by decreasing size of free CPU, and (iii) in case of same CPU capacity, hosts are selected by increasing values of idle power consumption. The "Shi-AC" is from Shi et al. (2013) and assigns a VM placement by favoring a server with the largest absolute CPU capacity [25].

To address the issue of unnecessary VM migrations and an increase in SLA violation caused by heuristics that only deal with minimizing the number of servers, Farahnakian et al. (2015) proposed prediction aware VM placement [24]. The proposed algorithm called Utilization Prediction Aware Best-Fit Decreasing algorithm (UP-BFD) chooses a host based on the prediction of future resource utilization. Their simulation result shows that UP-BFD performs better than those that are not utilization prediction aware. The authors provide a whole set of prediction aware algorithms for consolidation.

Wang et al. (2016) developed an improved particle swarm optimization (PSO) algorithm to optimize virtual machine placement in national cloud data centers [26]. The optimization considers the trade-off between energy consumption and global QoS (response time, throughput, availability and reliability) guarantee for data-intensive services. To evaluate the algorithm, the authors extended CloudSim to a new simulator called FTCloudSim by adding fat-tree data center network construction module, a QoS module, and so on. Experiment results show that the proposed approach reduce energy consumption while satisfying the global QoS guarantee.

In line with the above works, this paper tries to address the consolidation problem from all three perspectives. However, instead of proposing a novel framework we built our work on existing OpenStack Neat framework and specifically improve the VM placement component. The VM placement algorithms we propose like most of the above works are based on bin-packing heuristics. The difference is that our algorithms include power efficiency of servers as an additional factor. Moreover, we defined a new bin-packing rule that is suitable to reduce SLA violation and amount of VM migrations. The complexity of bin-packing based VM placement algorithms are very simple. It is proportional to the number of VMs to the number of hosts. This is much simpler than evolutionary computation techniques that also involve the number of populations and iterations.

Problem description

An efficient VM placement algorithm is expected to allocate computing resources in such a way that it satisfies VMs' resource demand and minimizes energy utilization with the least number of VM migrations possible. As described in "Related work" section, many of the VM placement algorithms are based on bin-packing algorithms. The problem of VM placement is analogous to problem of bin-packing where items of different sizes are assigned to bins of unit size (see "Proposed work" section). The bin-packing heuristics are more suitable to homogeneous cloud environment. For heterogeneous cloud environment, however, the heuristics need to be modified to accommodate the variability in servers. Moreover, as described in "Introduction" section, efficient bin-packing algorithms like BFD, increase overload probability of servers.

The VM placement component of OpenStack Neat framework, MBFD, has no mechanism to handle the heterogeneity of servers. The PABFD algorithm, designed for the same framework, uses the estimated power utilization of servers for the current VM [18]. The idle power of the servers, however, is not taken in to consideration by PABFD. This has a negative impact on the total energy-efficiency of the algorithm.

In this paper, we address the limitation of the existing VM placement algorithms in OpenStack Neat framework with respect to energy-efficiency, VM migrations and SLA violation.

Proposed work

A bin-packing model is a natural fit to the problem of VM placement. In bin-packing, items are assigned to containers (bins) with optimization objective of minimizing the number of bins. Items are specified by sizes with values not more than the size of the bin. The bin-packing problem is well known to be NP-hard and, hence, an exact solution is unlikely to be found or is inefficient (e.g., for very dynamic problems). However, many approximation heuristics are proposed in literature [27, 28]. The most common heuristics are the following:

Next-Fit (NF) assigns an item to the recently opened bin if it fits; otherwise, it closes it and opens an empty bin.

First-Fit (FF) assigns an item to the first partially filled bin that fits it; otherwise, an empty bin is opened.

Best-Fit (BF) assigns an item to the fullest partially filled bin that fits it; otherwise, an empty bin is opened.

Worst-Fit (WF) assigns an item to the lowest partially filled bin that fits it; otherwise, an empty bin is opened. WF has a rule that opposes the BF rule.

Any-Fit (AF) any fitting rule that doesn't open an empty bin unless all partially filled bins do not fit an item. All of the above heuristic except NF belong to AF.

The worst-case asymptotic performance ratio (APR) is used to measure the packing efficiency of a given heuristic relative to the optimum in the worst case scenario. APR is defined as follows:

Let A(L) be the number of bins when algorithm A is used to pack list of items L and OPT(L) be the optimum number of bins. If, $R_A(L) = A(L)/OPT(L)$ is the ratio of the number of bins taken by A to that of the optimum, then APR is defined as [27]:

$$APR(A) \equiv \inf\{r \geq 1 : \text{ for some } N > 0, R_A(L) \leq r \forall L \text{ with } OPT(L) \geq N\}, \tag{1}$$

where r is a real number greater than or equal to the ratio $R_A(L)$.

APR of next-fit and worst-fit heuristics is 2; while that of first-fit and best-fit is 17/10 [29]. The APR of any-fit is between first-fit and next-fit. The worst-case asymptotic performance ratio is improved when the problem allows items to be presorted in decreasing order. The most important results are for First-Fit Decreasing (FFD) and Best-Fit Decreasing (BFD):

$$APR(FFD) = APR(BFD) = 11/9 \text{ [30]},$$

where FFD and BFD are algorithms that apply first-fit and best-fit rules, respectively to items sorted in decreasing order of size.

The average case performance ratio measures the expected performance ratio of approximation algorithms by considering a uniform distribution of item sizes. The average case performance ratio of both BF and FF converge to 1 asymptotically [27].

Thus, concerning minimizing the number of servers, best-fit and first-fit based algorithms will give better efficiency. However, to address the limitation of best-fit based algorithms, that is high overloading probability, we define a new bin-packing heuristic called a Medium-Fit (MF).

The medium-fit rule

The medium-fit bin-packing rule is defined as follows:

Let L_D be a desired resource utilization level of a host given by, $L_D \equiv (overload_{thr} + underload_{thr})/2$; where $overload_{thr}$ and $underload_{thr}$ are overload and underload threshold of resource utilization levels, respectively. Then, the MF rule is defined to favor a host whose resource level has a minimum distance from L_D. More precisely,

$$\text{Allocated-host} \equiv \arg\min_h |L_h - L_D|, \tag{2}$$

where L_h is the resource utilization level of host, h.

If $L_D = overload_{thr}$, then the equation will be similar to the best-fit algorithm. If on the other hand $L_D = underload_{thr}$, then it will be equivalent to the worst-fit. Hence, the name medium-fit. As an example if $overload_{thr}$ is assumed 0.9 (90%) and $underload_{thr}$ is assumed 0.3 (30%) then $L_D = 0.6$. The MF rule with items sorted by decreasing size is called a Medium-Fit Decreasing (MFD).

The reason why the MFD algorithm minimizes overload probability and at the same time minimize the number of active servers is explained in the following. Suppose that all hosts are below L_D and the underload detection algorithm selects the lowest loaded host for its VMs to be migrated. Then, the MFD algorithm takes each VM in turn and allocates it to the highest loaded host according to Eq. (2). The process is repeated– taking VMs from the lowest loaded host to highest ones– until some hosts pass the desired level, L_D. The hosts, whose VMs are all migrated, are turned off or put in sleep mode. This is minimizing the number of active servers without the highest loaded hosts passing overload-threshold. On the other hand, if some hosts are above the desired level, say by the long run underload migration process, then Eq. (2) imply that any new VM migration (from underloaded or overloaded hosts) will be allocated to a host whose load level is near L_D. In both cases, MFD minimizes the overload probability, and as a consequence reduces SLA violation and VM migrations.

Power-efficient modified heuristics

Minimizing the number of active servers is one but not the only solution to reduce energy consumption in a cloud data-center. If in addition, servers are categorized according to their power efficiency and the most efficient ones are favored, then we expect more energy-efficiency in heterogeneous cloud. For this purpose we define Power Efficiency (PE) of servers as follows:

$$PE \equiv CPU_{total}/Power_{max}, \tag{3}$$

where:

- CPU_{total} is the total processing capacity of a host
- $Power_{max}$ is the power of a host at 100% load

From the definition we can see that PE becomes higher, the higher the CPU and the lower the maximum-power. The energy efficiency in the "Lago" algorithm has the the same definition [23].

Figure 1 illustrate energy efficient heuristics for VM placement algorithms. According to the illustration, using one of the following heuristics lead to an energy efficient VM placement algorithm:

1 **Minimizing the number of active hosts** using Bin-packing heuristics such as FFD and BFD.
2 Favoring hosts with **higher power efficiency (PE)**.
3 Using an algorithm that combines the above rules.

In this research we propose a combined heuristic (option 3) for better energy efficiency. The combination is done by modifying the bin-packing rules with power-efficiency (PE).

Proposed VM placement algorithms

Using the combined heuristic illustrated in Fig. 1, we develop three algorithms: Power Efficient First-Fit Decreasing (PEFFD), Power Efficient Best-Fit Decreasing (PEBFD) and Medium-Fit Power Efficient Decreasing (MFPED). In the first two algorithms, active hosts are categorized according to their power efficiency and the most efficient ones are favored. Within the same category of hosts, FFD and BFD rules are applied. In both algorithms, if a suitable host is not found from the active hosts, the same rules are applied to the inactive hosts and turn on the selected one. In the third algorithm, the intention is to reduce the effect of overloading and also reducing energy consumption. Thus, MFPED will first apply the medium-fit rule on active servers and in case of a tie, a host with the highest PE is favored. The details of the algorithms are given in the following subsections.

The power efficient first-fit decreasing (PEFFD) algorithm

The PEFFD algorithm as shown in Algorithm 1, takes as input the VM list to be migrated and the host list to be allocated. For each VM sorted by decreasing resource demand (line 1 in Algorithm 1), the algorithm first tries to find a host that fits it and is the best (line 5-13). A host fits for a VM if it has enough resource for the VM (line 6). The best host is then determined by iteratively replacing *allocatedHost* with a better host (line 7-11). In PEFFD, a host is better than another host if its power efficiency, PE (see Eq. 3), is greater than that of the other host and the lowest indexed one is chosen to break a tie. Next, a VM and the best host are added to a *vmPlacement* map (line 15). If an active host is not found for a VM, then a host is searched from inactive host lists using the same process (line 16-31). When all VMs are exhausted the *vmPlacement* map is returned.

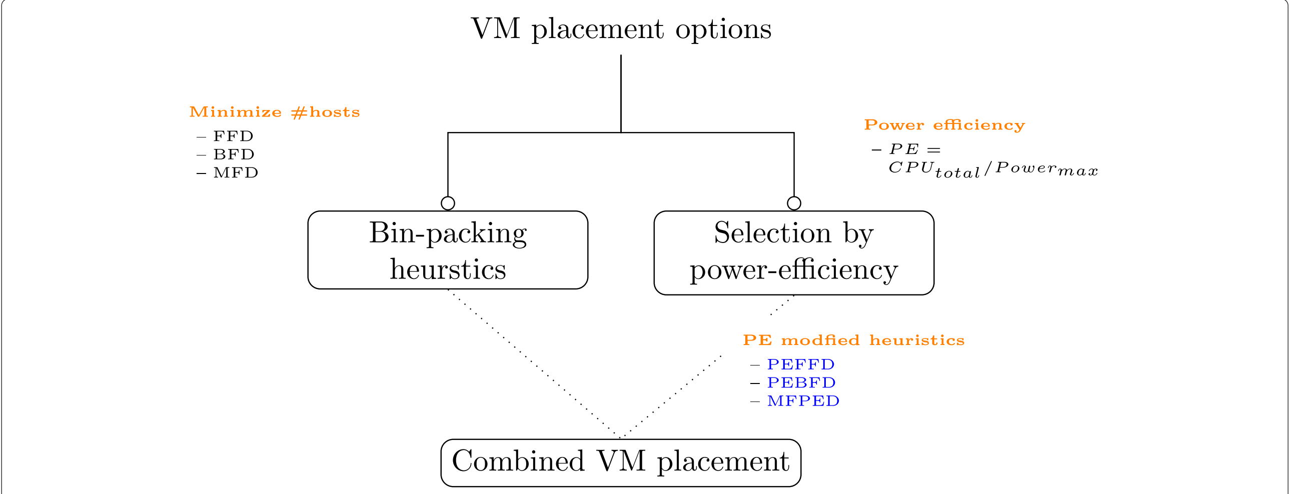

Fig. 1 Illustration of a combined heuristic. The proposed algorithms combines two VM placement heuristics: bin-packing rules and selection by power-efficiency

Algorithm 1: The Power Efficient First-Fit Decreasing

```
Input: activeHostList, inactiveHostList, vmList
Output: vmPlacement
1  sort vmList in the order of decreasing CPU utilization ;
2  foreach vm in vmList do
3      bestPowerEfficiency ← MIN ;    // MIN is a minimum number
4      allocatedHost ← NULL;
5      foreach host in activeHostList do
6          if host has enough resource for vm then
7              powerEfficiency ← getTotalCPU(host)/getMaxPower(host) ;
8              if powerEfficiency > bestPowerEfficiency then
9                  allocatedHost ← host ;
10                 bestPowerEfficiency ← powerEfficiency ;
11             end
12         end
13     end
14     if allocatedHost ≠ NULL then
15         add (allocatedHost, vm) to vmPlacement ;
16     else
           // assign allocatedHost from inactive hosts
17         bestPowerEfficiency ← MIN;
18         allocatedHost ← NULL;
19         foreach host in inactiveHostList do
20             if host has enough resource for vm then
21                 powerEfficiency ← getTotalCPU(host)/getMaxPower(host) ;
22                 if powerEfficiency > bestPowerEfficiency then
23                     allocatedHost ← host ;
24                     bestPowerEfficiency ← powerEfficiency ;
25                 end
26             end
27         end
28         if allocatedHost ≠ NULL then
29             add (allocatedHost, vm) to vmPlacement ;
30         end
31     end
32 end
Result: vmPlacement
```

The power efficient best-fit decreasing (PEBFD) algorithm
The PEBFD algorithm shown in Algorithm 2 has many similarity to the PEFFD described above. The difference is the PEBFD uses the best-fit rule instead of the first-fit rule. In PEBFD, a host is better than another host if its power efficiency, PE, is greater than that of the other host. In case two hosts have the same PE, then the one that has lower available CPU is chosen (line 12-16 in Algorithm 2).

The medium-fit power efficient decreasing (MFPED) algorithm
The MFPED algorithm is listed in Algorithm 3. In MFPED, a host is better than another host for VM allocation if its CPU utilization level distance from the desired level (as defined in (2)) is less than that of the other host (line 8-12 in Algorithm 3). In case two hosts have levels with equal distance from the desired level, then the one that has a higher PE is chosen (line 13-17).

Experiment and results
The algorithms are simulated in CloudSim, the most popular toolkit for modeling and simulating cloud environment and evaluation of resource allocation algorithms [31]. CloudSim contains Java classes for modeling the different components of a cloud. Moreover, the OpenStack Neat framework is included as power packages in CloudSim [18]. Thus, any power-aware algorithm can be simulated by extending an existing power-aware allocation policy. In this research, we extended *PowerVmAllocationPolicyMigrationLocalRegression* class and overrode the *findHostForVm* method by the proposed algorithms. Methods for overload decision (the local regression), underload decision and VM selection are inherited from the base classes.

Experiment setups
We compare the proposed VM placement algorithms with two baseline algorithms: MBFD, the VM placement in OpenStack Neat [14] and PABFD which is the default VM placement in CloudSim [18]. MBFD deals with minimizing the number of servers while PABFD chooses a host that leads to the minimum power increment for the VM to be placed. The baseline algorithms are described in (Algorithms 4 and 5).

For simulating a cloud environment, we defined three data-center scenarios (default, heterogeneous and homogeneous) described in the following subsections.

Default-scenario
The first scenario, shown in Table 1, adopts the data-center setup of Beloglazov et al. which is included in CloudSim [18]. The data-center has 800 hosts from two server models (400 hosts from each server type) and four types of VMs. The CPU capacity of the VM instances is given in millions of instruction per second (MIPS). The number of VMs and their workloads are generated from real cloud traces: from PlanetLab [18] and Bitbrains clouds [19, 32].

Algorithm 2: The Power Efficient Best-Fit Decreasing

```
Input: activeHostList, inactiveHostList, vmList
Output: vmPlacement
1  sort vmList in the order of decreasing CPU utilization ;
2  foreach vm in vmList do
3      bestPowerEfficiency ← MIN;
4      allocatedHost ← NULL;
5      foreach host in activeHostList do
6          if host has enough resource for vm then
7              powerEfficiency ← getTotalCPU(host)/getMaxPower(host) ;
8              if powerEfficiency > bestPowerEfficiency then
9                  allocatedHost ← host ;
10                 bestPowerEfficiency ← powerEfficiency ;
11             else
12                 if powerEfficiency == bestPowerEfficiency then
13                     if getAvailableCPU(host) < getAvailableCPU(allocatedHost) then
14                         allocatedHost ← host ;
15                     end
16                 end
17             end
18         end
19     end
20     if allocatedHost ≠ NULL then
21         add (allocatedHost, vm) to vmPlacement ;
22     else
           // assign allocatedHost from inactive hosts
23         bestPowerEfficiency ← MIN;
24         allocatedHost ← NULL;
25         foreach host in inactiveHostList do
26             if host has enough resource for vm then
27                 powerEfficiency ← getTotalCPU(host)/getMaxPower(host) ;
28                 if powerEfficiency > bestPowerEfficiency then
29                     allocatedHost ← host ;
30                     bestPowerEfficiency ← powerEfficiency ;
31                 else
32                     if powerEfficiency == bestPowerEfficiency then
33                         if getAvailableCPU(host) < getAvailableCPU(allocatedHost) then
34                             allocatedHost ← host ;
35                         end
36                     end
37                 end
38             end
39         end
40         if allocatedHost ≠ NULL then
41             add (allocatedHost, vm) to vmPlacement ;
42         end
43     end
44 end
Result: vmPlacement
```

Algorithm 3: The Medium-Fit Power Efficient Decreasing

```
Input: hostList, vmList
Output: vmPlacement
1  $L_D$ ← 0.6 ;  // The desired level is set to 0.6
2  sort vmList in the order of decreasing CPU utilization ;
3  foreach vm in vmList do
4      minDiff ← MAX ;  // MAX is the maximum number
5      allocatedHost ← NULL;
6      foreach host in hostList do
7          if host is active and has enough resource for vm then
8              diff ← $|getUtilization(host) - L_D|$ ;
9              if diff < minDiff then
10                 allocatedHost ← host ;
11                 minDiff ← diff ;
12             else
13                 if diff == minDiff then
                       // PE(.) is the power efficiency of a host
14                     if PE(host) > PE(allocatedHost) then
15                         allocatedHost ← host ;
16                     end
17                 end
18             end
19         end
20     end
21     if allocatedHost ≠ NULL then
22         add (allocatedHost, vm) to vmPlacement ;
23     end
24 end
Result: vmPlacement
```

For overload prediction, the local regression policy which is available in the CloudSim simulator is used. At each optimization step, the minimum loaded host is chosen for its VMs to be migrated. If the remaining hosts have enough resource to handle the VMs, the host will be turned off for saving power.

Heterogeneous-scenario

In this scenario the number of host types is increased to four by adding two quad-core IBM server models: IBM Xeon X3470 (4 X 2933 MIPS) and IBM Xeon X3480 (4 X 3067 MIPS). To make a comparable computational power as that of the Default-scenario, the number of hosts is reduced to 560 (140 of each server type). The addition of two server types creates more heterogeneity in the cloud.

Algorithm 4: The Modified Best-Fit Decreasing

```
Input: activeHostList, inactiveHostList, vmList
Output: vmPlacement
1  sort vmList in the order of decreasing average CPU utilization ;
2  foreach vm in vmList do
3      $minCPU \leftarrow MAX$;
4      $allocatedHost \leftarrow NULL$;
5      foreach host in activeHostList do
6          if host has enough resource for vm then
7              $cpu \leftarrow getAvailableCPU(host)$ ;
8              if $cpu < minCPU$ then
9                  $allocatedHost \leftarrow host$ ;
10                 $minCPU \leftarrow cpu$ ;
11             else
12                 if $cpu == minCPU$ and $getAvailableRAM(host) < getAvailableRAM(allocatedHost)$ then
13                     $allocatedHost \leftarrow host$
14                 end
15             end
16         end
17     end
18     if $allocatedHost \neq NULL$ then
19         add (allocatedHost, vm) to vmPlacement ;
20     else
           // assign allocatedHost from inactive hosts
21         $minCPU \leftarrow MAX$;
22         $allocatedHost \leftarrow NULL$;
23         foreach host in inactiveHostList do
24             if host has enough resource for vm then
25                 $cpu \leftarrow getAvailableCPU(host)$ ;
26                 if $cpu < minCPU$ then
27                     $allocatedHost \leftarrow host$ ;
28                     $minCPU \leftarrow cpu$ ;
29                 else
30                     if $cpu == minCPU$ and $getAvailableRAM(host) < getAvailableRAM(allocatedHost)$ then
31                         $allocatedHost \leftarrow host$
32                     end
33                 end
34             end
35         end
36         if $allocatedHost \neq NULL$ then
37             add (allocatedHost, vm) to vmPlacement ;
38         end
39     end
40 end
Result: vmPlacement
```

Algorithm 5: The Power Aware Best-Fit Decreasing

```
Input: hostList, vmList
Output: vmPlacement
1  sort vmList in the order of decreasing CPU utilization ;
2  foreach vm in vmList do
3      $minPower \leftarrow MAX$;
4      $allocatedHost \leftarrow NULL$;
5      foreach host in hostList do
6          if host has enough resources for vm then
7              $power \leftarrow estimatePower(host, vm)$ ;
8              if $power < minPower$ then
9                  $allocatedHost \leftarrow host$ ;
10                 $minPower \leftarrow power$ ;
11             end
12         end
13     end
14 end
15 if $allocatedHost \neq NULL$ then
16     add (allocatedHost, vm) to vmPlacement ;
17 end
Result: vmPlacement
```

Homogeneous-scenario

In this scenario only one type of host is defined in the data-center. The setup differs from the Default-scenario (Table 1) by the host type which in this case is only the HP ProLiant ML110 G5. In this scenario the power efficiency, PE, of all hosts are equal. Thus, performance improvement with respect to energy consumption is not expected from the proposed algorithms.

Table 1 Default-scenario parameters and configurations

Parameters	Configuration
Host types	HP ProLiant ML110 G4 (2 X 1800 MIPS)
	HP ProLiant ML110 G5 (2 X 2660 MIPS)
Number of hosts	800; 400 of each host type
VM types	2500 MIPS
	2000 MIPS
	1500 MIPS
	1000 MIPS
Workloads	PlanetLab (10 days of traces)
	Bitbrains (10 days of traces)
Overload decision	Local regression
Underload decision	The minimum loaded host

The data-center host, VM types and the PlanetLab traces are adopted from the configuration in CloudSim [18]

Workload traces

The experiment is performed on workload traces collected from real clouds: PlanetLab and Bitbrains [18, 32]. The PlanetLab is a cloud of global research network and the traces are collected from a monitoring system called CoMon [33]. The data contains the percentage of CPU utilization by more than a thousand VMs from servers located at more than 500 places around the world. It is collected during 10 randomly selected days in March and April 2010 [18]. The dataset is organized as one folder per day and a file in a folder contains a one day CPU utilization of a VM sampled every 5 min. The statistical characteristics of the dataset are shown in Table 2.

Bitbrains is a cloud service provider that specializes in managed hosting and business computation for enterprises [32]. The dataset of Bitbrains contains resource utilization by 1750 VMs from a distributed data-center and is available online in the Grid Workloads Archive [19]. It is organized into two folders: fastStorage traces consists of 1250 VMs and Rnd traces consists of 500 VMs. In this work, we used the fastStorage traces. The fastStorage dataset is organized as one file per a VM, each file containing 30 days of data sampled every 5 min.

To reuse the same utilization-model as that of PlanetLab available in CloudSim (*UtilizationModelPlanetLabInMemory* class), we have converted the datasets of Bitbrains to the format of PlanetLab datasets. The first 10 days of the converted datasets are used in our experiment. The statistical characteristics of the Bitbrains dataset used in our experiment are shown in Table 3.

Evaluation metrics

We adopt the evaluation metrics proposed by Beleglazov et al. [18]. The three main metrics are those that measure energy efficiency, SLA violation and VM migrations.

Table 2 Statistical characteristics of PlanetLab workloads traces [18]

Date	Number of VMs	Mean-load(%)	St.dev.(%)
03/03/2011	1052	12.31	17.09
06/03/2011	898	11.44	16.83
09/03/2011	1061	10.70	15.57
22/03/2011	1516	9.26	12.78
25/03/2011	1078	10.56	14.14
03/04/2011	1463	12.39	16.55
09/04/2011	1358	11.12	15.09
11/04/2011	1233	11.56	15.07
12/04/2011	1054	11.54	15.15
20/04/2011	1033	10.43	15.21

The percentages are relative to the configured CPU capacity of VMs

Table 3 Statistical characteristics of Bitbrains workloads traces

Date	Number of VMs	Mean-load(%)	St.dev.(%)
01/08/2013	1238	11.21	26.33
02/08/2013	1237	7.60	17.52
03/08/2013	1234	5.10	13.16
04/08/2013	1233	8.48	21.11
05/08/2013	1232	9.43	21.67
06/08/2013	1231	8.63	23.19
07/08/2013	1218	7.73	17.49
08/08/2013	1209	10.78	24.07
09/08/2013	1207	7.06	16.93
10/08/2013	1205	8.64	21.62

The percentages are relative to the configured CPU capacity of VMs

Energy efficiency is measured with the total data-center energy consumption in kwh (kilo-watt hour),

$$Energy_C \equiv \text{data-center energy consumption per day} \tag{4}$$

SLA violation due to overloading of hosts is measured by the aggregate overload time fraction (OTF) and is defined as:

$$OTF \equiv \frac{1}{N}\sum_{i=1}^{N}\frac{T_{o_i}}{T_{a_i}}, \tag{5}$$

where:

– N is the number of hosts
– T_{o_i} is the total time during which the host i has experienced the utilization of 100% leading to an SLA violation. In CloudSim simulation, T_{o_i} is counted when the CPU capacity requested exceeds the available capacity.
– T_{a_i} is the total time host i is in the active state (serving VMs)

A VM migration causes overhead on a network as well as an SLA violation as there will be service disruption during migration. In addition, VM migration incurs energy costs. In this study, however, the energy cost and its impact on the relative performance of VM placement algorithms is assumed to be negligible. As indicated in the study conducted by Huang et al. [34], in the case of consolidation the impact of VM migrations on energy cost is minimal and its value is dependent on factors such as speed of LAN [8]. Typical power consumed during a VM migration in real experiment is 1 watt on source host and 10 watts on destination host for 7 s; which gives energy cost of 2.1×10^{-5} kwh [34]. Hence, the energy consumption due to VM migrations will be negligible when compared to the total energy consumed in a data-center. Moreover, the cost of energy for migrating a VM on average is the same for both the proposed and baseline algorithms. The energy cost that might arise due to VM migrations, hence, will not be higher in case of the proposed algorithms unless the amount of VM migrations they cause is higher.

The associated metric to VM migration is denoted as *#VM migrations*.

$$\#VM\ migrations \equiv \text{The number of VM migrations in data-center per day} \quad (6)$$

In all of the metrics defined above, the lower the metric value is, the better the performance of the algorithm under consideration.

Results and discussion

The results of the experiment for each scenario are discussed in the following subsections.

Performance of algorithms in the default-scenario

In the Default-scenario all proposed algorithms give lower energy consumption, reduced SLA violation and VM migrations compared with baseline algorithms. Comparison by energy efficiency of the proposed algorithms against the baselines MBFD and PABFD are shown in Fig. 2. From the box plots of Fig. 2a, we observe that PEBFD delivers the lowest median energy consumption of 109.5 kwh in case of PlanetLab workload traces. It is followed by a closer results of MFPED and PEFFD with values 109.9 kwh and 110.3 kwh, respectively. The highest energy consumption has resulted from the baseline algorithm PABFD with a median value of 158.3 kwh followed by MBFD at 120 kwh. The improvement of proposed algorithms over baseline MBFD is 8 - 9%. In case of Bitbrains workload traces shown in Fig. 2b, the order of performance of the algorithms is not changed. The magnitude of the difference in energy consumption, however, shows significant change. For example, the improvement by proposed algorithms in this case over the baseline MBFD is 2 - 3%.

Table 4 summarizes the average performance of the algorithms in the Default-scenario with respect to all defined metrics: energy consumption, overload time fraction(OTF) and #VM migrations (see "Evaluation metrics" section). In the table, the best values are highlighted in boldface. We observe that the proposed algorithms outperform the baseline algorithms in all performance metrics. The performance difference between proposed algorithms is negligible($<$ 1%) with respect to average energy consumption while with respect to both OTF and #VM migrations, MFPED has the best performance. Compared with MBFD, MFPED improves OTF and #VM migrations by 32% and 15% respectively using PlanetLab traces. Using Bitbrians traces, MFPED improves OTF and #VM migrations by 64% and 18%, respectively over MBFD.

Thus, we conclude that, in case of the Default-scenario, the proposed algorithms improve all metrics (energy consumption, SLA violation and number of VM migrations) irrespective of the workload traces considered.

Performance of algorithms in the heterogeneous-scenario

In the Heterogeneous-scenario all proposed algorithms deliver lower energy consumption, reduced SLA violation and VM migrations compared with baseline algorithms. As shown in Fig. 3, the proposed algorithms show larger energy consumption difference over baseline algorithms than the difference in case of Default-scenario (on average 64% vs. 6%). The result supports that the power efficiency, PE, as defined by Eq. (3) is an important energy efficiency factor. From the box plots of Fig. 3a, we observe that all proposed algorithms give a median energy consumption around 35.3 kwh using PlanetLab workload traces. The

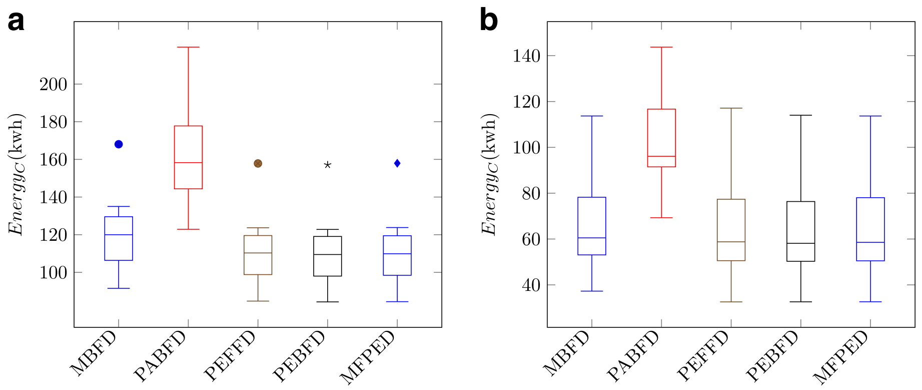

Fig. 2 Comparison by energy efficiency of algorithms in the Default-scenario. Total energy consumption in data-center with two types of dual-core HP ProLiant servers. MBFD and PABFD are the baseline algorithms and the rest are the proposed ones. **a** Results using PlanetLab traces. **b** Results using Bitbrains traces

Table 4 Average performance of algorithms in the Default-scenario

Workloads	Algorithms	*Energy*$_C$(kwh)	OTF(%)	#VM migrations
PlanetLab traces	MBFD	120.32	5.32	12919
	PABFD	161.87	6.21	28175
	PEFFD	111.13	4.06	12477
	PEBFD	**110.41**	4.21	11819
	MFPED	110.93	**3.62**	**10975**
Bitbrains traces	MBFD	67.49	6.82	8787
	PABFD	103.777	5.97	19808
	PEFFD	65.90	2.63	7612
	PEBFD	**65.17**	3.45	8527
	MFPED	65.57	**2.44**	**7216**

The best values for each metrics (defined in 4-6) are boldfaced

highest energy consumption has resulted from the baseline algorithm MBFD with a median value of 107.6 kwh followed by PABFD at 48.5 kwh. The improvement of proposed algorithms over baseline MBFD is around 67%. In case of Bitbrains workload traces shown in Fig. 3b, the lowest median energy consumption has resulted from both PEFFD and PEBFD with value 18.3 kwh. The result is followed by a very close value of MFPED at 18.6 kwh. The improvement of proposed algorithms, in this case, over the baseline MBFD is about 60%.

Table 5 presents the average performance of the algorithms with respect to all defined metrics for Heterogeneous-scenario. The performance difference between proposed algorithms is negligible ($< 0.5\%$) with respect to energy consumption. With respect to OTF and #VM migrations, MFPED has the lowest value followed by PEFFD. The improvement of MFPED over the baseline MBFD is 68% for OTF and 46% for #VM migrations.

In case of Bitbrains traces too, the proposed algorithms improve energy consumption against both baseline algorithms and the differences between them are negligible. The MFPED algorithm improves OTF and #VM migration by 78% and 40% respectively, against MBFD.

Thus, in Heterogeneous-scenario, the proposed algorithms improve all metrics (energy consumption, SLA violation and number of VM migration) by a greater amount than the improvement in case of the Default-scenario.

Performance of algorithms in the homogeneous-scenario

There is no average energy saving by the proposed algorithms against baseline MBFD in case of Homogeneous-scenario. The benefit of the proposed algorithms, in this case, is on reducing overload time fraction and number of VM migrations. In Homogeneous-scenario, as the power efficiency of all host is the same, the only power saving option for the algorithms is to minimize the number of hosts. Thus, we do not expect an improvement in energy efficiency from the proposed algorithms over MBFD. As shown in Fig. 4, both BFD based algorithms, MBFD and PEBFD, deliver nearly equal median energy consumption of 109.5 kwh and 58.2 kwh for PlanetLab and Bitbrains traces, respectively. The worst energy efficiency (highest consumption) has resulted from PABFD in both workload traces.

Table 6 presents the average result of the experiment in Homogeneous-scenario using PlanetLab traces. There is no average energy saving by the proposed algorithms against MBFD. There is, however, improvement in reducing overload time fraction and number of VM migrations. Compared with MBFD, MFPED improves OTF and #VM migrations by 19% and 9%, respectively, when traces from PlanetLab are used. Similarly, in case of Bitbrians traces, the two BFD based algorithms, MBFD and PEBFD, give almost equal energy consumption. Using Bitbrians traces,

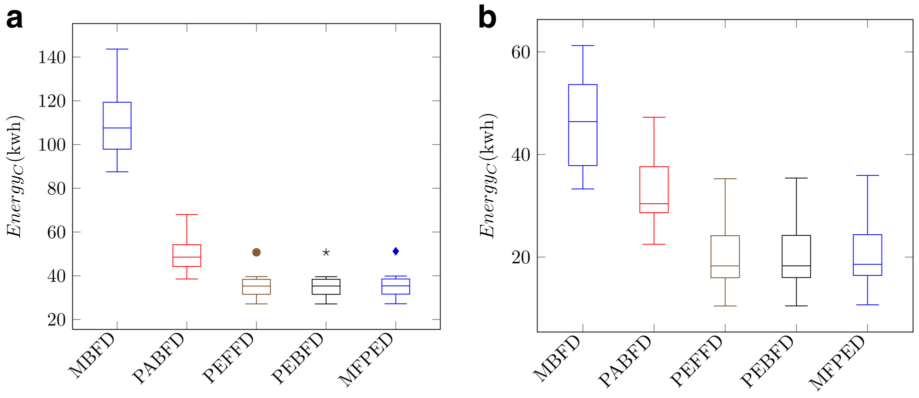

Fig. 3 Comparison by energy efficiency of algorithms in the Heterogeneous-scenario. Total energy consumption in data-center with four types of servers from dual-core HP ProLiant and quad-core IBM Xeon. MBFD and PABFD are the baseline algorithms and the rest are the proposed ones. **a** Results using PlanetLab traces. **b** Results using Bitbrains traces

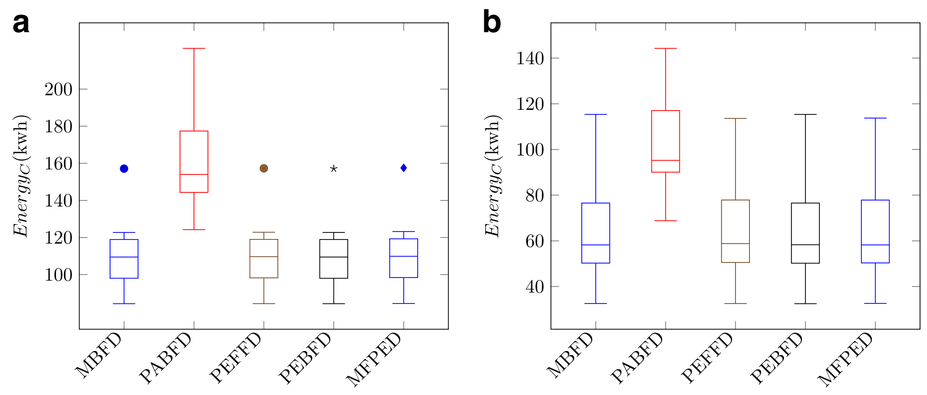

Fig. 4 Comparison by energy efficiency of algorithms in the Homogeneous-scenario. Total energy consumption in data-center with one type of server: HP ProLiant ML110 G5. MBFD and PABFD are the baseline algorithms and the rest are the proposed ones. **a** Results using PlanetLab traces. **b** Results using Bitbrains traces

MFPED improves OTF and #VM migrations by 42% and 14%, respectively over MBFD. We also note that the worst algorithm with respect to all three metrics in both workload traces is the PABFD.

Tests of significance

To verify the significance of the experiment results, we have conducted the Wilcoxon signed-rank test [35]. Two tests are presented in this Section: (i) comparing the energy consumption of a data-center under proposed algorithms with the value under MBFD (see Table 7), (ii) comparing OTF and #VM migrations value of MFPED algorithm with that of MBFD (see Table 8). The significance level $\alpha = 0.01$ is used for the test. Both tests are for Default-scenario using PlanetLab traces.

The results of the statistical tests are shown in Tables 7 and 8. P-value indicates the probability of the null hypothesis, the alternative hypothesis to the hypothesis being tested. For all tests in both tables the P-values are less than the significant level α. Which means that the improvement in all metrics are significant with confidence level of at least 99%.

Test of significance in all other improvements, such as the improvements in Heterogeneous-scenario gives similar results to the above tables.

The complexity of the algorithms

The runtime complexity of the five VM placement algorithms, described in "Proposed work" and "Experiment setups" sections, are presented as follows. For all algorithms the VMs to be migrated are sorted in decreasing order and that takes a run time of $O(m_{mig} * log(m_{mig}))$, where m_{mig} is the number of VM migrations. The algorithms then proceed in two loops: The outer loop traces the VMs to

Table 5 Average performance of algorithms in the Heterogeneous-scenario

Workloads	Algorithms	$Energy_C$(kwh)	OTF(%)	#VM migrations
PlanetLab traces	MBFD	109.31	5.51	12524
	PABFD	49.66	6.04	18160
	PEFFD	**35.58**	2.06	7452
	PEBFD	**35.58**	2.18	7853
	MFPED	35.74	**1.74**	**6731**
Bitbrains traces	MBFD	46.11	6.56	8360
	PABFD	33.29	6.45	17826
	PEFFD	**20.56**	1.82	5652
	PEBFD	20.59	2.07	5903
	MFPED	20.88	**1.48**	**5041**

The best values for each metrics (defined in 4-6) are boldfaced

Table 6 Average performance of algorithms in the Homogeneous-scenario

Workloads	Algorithms	$Energy_C$(kwh)	OTF(%)	#VM migrations
PlanetLab traces	MBFD	**110.40**	4.26	11898
	PABFD	161.43	6.26	27980
	PEFFD	110.57	4.00	12035
	PEBFD	**110.40**	4.27	11909
	MFPED	110.78	**3.60**	**10914**
Bitbrains traces	MBFD	65.29	3.19	8540
	PABFD	103.70	6.00	19625
	PEFFD	65.59	2.64	7578
	PEBFD	**65.28**	3.16	8408
	MFPED	65.39	**2.24**	**7460**

The best values for each metrics (defined in 4-6) are boldfaced

Table 7 Statistical test for the significance of energy saving by the proposed algorithms

Comparison by energy consumption,E	*P*-value
E(PEFFD) < E(MBFD)	9.8×10^{-4}
E(PEBFD) < E(MBFD)	9.8×10^{-4}
E(MFPED) < E(MBFD)	9.8×10^{-4}

be migrated and the inner loop traces the hosts to be allocated. That will take $O(m_{mig} \times n)$ runtime, where n is the number of servers. Summing the two we have a runtime complexity of $O(m_{mig} * log(m_{mig}) + m_{mig} \times n)$. As the number of virtual machines, m, usually is greater than the number of servers, the runtime complexity of all algorithms are bounded by $O\left(m^2\right)$.

Regarding space complexity, the dominant variables for all algorithms are VMs' and servers' information such as CPU and RAM. No extra information is needed for MBFD. The extra RAM for PABFD is the power utilization model of the servers which is defined by 11 power samples of servers. Thus in the worst case, PABFD need $11 * n$ extra units of memory. Likewise, the proposed algorithms need the peak power of servers to calculate the power-efficiency which takes an extra memory of n units.

Threats to validity

The main threat to the research is the experiment environment: the simulation tool and the scenarios defined. Conducting an experiment in a real cloud is infeasible considering the very expensive hardware requirement and the management overhead. That also does not facilitate the progress of research on cloud management software. On the other hand, CloudSim is typically designed to simulate a cloud environment and to evaluate a resource allocation algorithms [31]. It is widely used by research works both in industry and academia [18, 22–24]. Thus, the simulation environment is expected to provide results similar to the result in a practical environment.

Considering the cloud scenarios, we have included both homogeneous and heterogeneous data-center environment. We have also included the baseline setup available in CloudSim so that the result of the experiment can be compared with other similar works. In addition, the workloads for simulations are generated from randomly selected real cloud traces: PlanetLab and Bitbrains. In each scenario, we have conducted twenty experiments: ten experiments using PlanetLab traces and another ten experiments using Bitbrians traces. The diversity and reality of the scenarios considered enhance the evidence to support the acceptance of the experiment results.

Table 8 Statistical test for the significance of OTF and #VM migrations improvement by the proposed MFPED algorithm

Comparisons by OTF & #VM migrations(Mig)	*P*-value
OTF(MFPED) < OTF(MBFD)	9.8×10^{-4}
Mig(MFPED) < Mig(MBFD)	9.8×10^{-4}

Conclusion

One of the main challenges in cloud computing is the enormous amount of energy consumed in data-centers. Several research works are devoted to address the challenge using VM consolidation. VM Consolidation is the process of minimizing energy consumption in a cloud by allocating VMs to the fewest possible servers. The complexity of the problem of consolidation arises from the trade-off between energy saving and SLA violation.

In this research, we have addressed the problem of consolidation by improving the VM placement algorithm of OpenStack Neat framework. We have proposed VM placement algorithms by modifying bin-packing heuristics considering the power-efficiency of hosts. The proposed algorithms improve energy efficiency when compared with the baseline algorithms: MBFD and PABFD. The improvement in energy efficiency over MBFD can be up-to 67%, depending on the data-center host types and workloads. Moreover, to avoid unnecessary SLA violation and VM migrations, we defined a new bin-packing rule called a medium-fit. Compared with other VM placement algorithms, the medium-fit power-efficient decreasing (MFPED), provides a lower SLA violation and number of VM migrations. MFPED improves SLA violation and number of VM migrations against MBFD by up-to 78% and 46%, respectively, depending on the cloud scenario.

Regarding practical implementation, the only additional information necessary to implement the proposed algorithms is the peak power of hosts. For future work, we plan to expand the scope of the work to include network devices and traffic effects in the consolidation problem.

Abbreviations

APR: Asymptotic worst-case performance ratio; BFD: Best-fit decreasing; FF: First-fit; FFD: First-fit decreasing; MBFD: Modified best-fit decreasing; MF: Medium-fit; MFD: Medium-fit decreasing; MFPED: Medium-fit power efficient decreasing; MIPS: Millions of instruction per second; NF: Next-fit; OTF: Overload time fraction; PaaS: Platform as a service; PABFD: Power-aware best-fit decreasing; PE: Power efficiency; PEBFD: Power efficient best-fit decreasing; PEFFD: Power efficient first-fit decreasing; SaaS: Software as a service; SLA: Service level agreement; VM: Virtual machine; WF: Worst-fit

Acknowledgments

There is no acknowledgment from the authors' side.

Authors' contributions

The manuscript is prepared based on a thesis work by the author (FFM), under the supervision of author (SLA), for the degree of masters of science in telecommunication engineering at Addis Ababa University. Both authors have read and approved the final manuscript.

Competing interests
The authors declare that they have no competing interests.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Received: 8 August 2018 Accepted: 9 January 2019
Published online: 24 January 2019

References

1. Armbrust M, Stoica I, Zaharia M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A (2010) A view of cloud computing. Commun ACM 53(4):50. https://doi.org/10.1145/1721654.1721672. 0521865719 9780521865715
2. Sangpetch A, Sangpetch O, Juangmarisakul N, Warodom S (2017) Thoth: Automatic resource management with machine learning for container-based cloud platform. In: Proceedings of the 7th International Conference on Cloud Computing and Services Science - Volume 1: CLOSER. pp 103–111. SciTePress. https://doi.org/10.5220/0006254601030111
3. Ismaeel S, Karim R, Miri A (2018) Proactive dynamic virtual-machine consolidation for energy conservation in cloud data centres. J Cloud Comput 7(1). https://doi.org/10.1186/s13677-018-0111-x
4. Clark C, Fraser K, Hand S, Hansen JG, Jul E, Limpach C, Pratt I, Warfield A (2005) Live migration of virtual machines. In: Proceedings of the 2Nd Conference on Symposium on Networked Systems Design and Implementation Vol. 2. https://doi.org/10.1145/1251203.1251223
5. Meisner D, Gold B, Wenisch T (2007) Powernap: eliminating server idle power. ACM SIGPLAN Not 44:205–216
6. Guazzone M, Anglano C, Canonico M (2012) Exploiting vm migration for the automated power and performance management of green cloud computing systems. Lecture Notes in Computer Science: Energy Efficient Data Centers 7396: 81–92
7. Voorsluys W, Broberg J, Venugopal S, Buyya R (2009) Cost of virtual machine live migration in clouds: A performance evaluation. In: Cloud Computing. Springer, Berlin. pp 254–265
8. Strunk A, Dargie W (2013) Does live migration of virtual machines cost energy? In: 2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA). IEEE. pp 514–521. https://doi.org/10.1109/AINA.2013.137
9. Urgaonkar R, Kozat UC, Igarashi K, Neely MJ (2010) Dynamic resource allocation and power management in virtualized data centers. 2010 IEEE Netw Oper Manag Symp - NOMS 2010:479–486. https://doi.org/10.1109/NOMS.2010.5488484
10. Berral JL, Gavaldà R, Torres J (2011) Adaptive scheduling on power-aware managed data-centers using machine learning. Proc - 2011 12th IEEE/ACM Int Conf Grid Comput Grid 2011 66–73. https://doi.org/10.1109/Grid.2011.18
11. Berral J, Gavalda R, Torres J (2013) Power-aware multi-data center management using machine learning. 42nd Int Conf Parallel Process. 858–867. https://doi.org/10.1109/ICPP.2013.102
12. Moghaddam FF, Moghaddam RF, Cheriet M (2015) Carbon-aware distributed cloud: multi-level grouping genetic algorithm. Cluster Comput 18(1):477–491. https://doi.org/10.1007/s10586-014-0359-y
13. Zhang J, Zhang L, Huang H, Wang X, Gu C, He Z (2016) A Unified Algorithm for Virtual Desktops Placement in Distributed Cloud Computing. Math Probl Eng. https://doi.org/10.1155/2016/9084370
14. Beloglazov A, Buyya R (2014) Openstack neat: A framework for dynamic and energy-efficient consolidation of virtual machines in openstack clouds. Concurr Comput Pract Experience. https://doi.org/10.1002/cpe.3314
15. Wen X, Gu G, Li Q, Gao Y, Zhang X (2012) Comparison of open-source cloud management platforms: OpenStack and OpenNebula. Proc - 2012 9th Int Conf Fuzzy Syst Knowl Discov FSKD 2012:2457–2461. https://doi.org/10.1109/FSKD.2012.6234218
16. Kumar R, Gupta N, Charu S, Jain K, Jangir SK (2014) Open Source Solution for Cloud Computing Platform Using OpenStack. Int J Comput Sci Mob Comput 3(5):89–98. https://doi.org/10.13140/2.1.1695.9043
17. Mullerikkal JP, Sastri Y (2015) A comparative study of openstack and cloudstack. In: 2015 Fifth International Conference on Advances in Computing and Communications (ICACC). IEEE. pp 81–84. https://doi.org/10.1109/ICACC.2015.110
18. Beloglazov A, Buyya R (2012) Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers. Concurr Comput Pract Exp 24(13):1397–1420. https://doi.org/10.1002/cpe.1867. 1006.0308
19. Anoep S, Dumitrescu C, Epema D, Iosup A, Jan M, Li H, Wolters L The Grid Workloads Archive: Bitbrains. http://gwa.ewi.tudelft.nl/datasets/gwa-t-12-bitbrains. Accessed 20 Mar 2018
20. Guazzone M (2012) Power and performance management in cloud computing systems. Ph.d. thesis, University of Torino, Computer Science Department https://dott-informatica.campusnet.unito.it/html/theses/guazzone.pdf
21. Rawas S, Zekri A, Zaart AE (2018) Power and cost-aware virtual machine placement in geo-distributed data centers. In: Proceedings of the 8th International Conference on Cloud Computing and Services Science - Volume 1: CLOSER. SciTePress. pp 112–123. https://doi.org/10.5220/0006696201120123
22. Chowdhury MR, Mahmud MR, Rahman RM (2015) Implementation and performance analysis of various VM placement strategies in CloudSim. J Cloud Comput 4(1):1–21. https://doi.org/10.1186/s13677-015-0045-5
23. Mann ZÁ, Szabó M (2017) Which is the best algorithm for virtual machine placement optimization? Concurr Comput Pract Exp 29(10):4083. https://doi.org/10.1002/cpe.4083
24. Farahnakian F, Pahikkala T, Liljeberg P, Plosila J, Tenhunen H (2015) Utilization Prediction Aware VM Consolidation Approach for Green Cloud Computing. Proc - 2015 IEEE 8th Int Conf Cloud Comput CLOUD 2015:381–388. https://doi.org/10.1109/CLOUD.2015.58
25. Shi, L, Furlong J, Wang, R (2013) Empirical evaluation of vector bin packing algorithms for energy efficient data centers. IEEE Symp Comput Commun 9–15. https://doi.org/10.1109/ISCC.2013.6754915
26. Wang S, Zhou A, Hsu CH, Xiao X, Yang F (2016) Provision of data-intensive services through energy- and qos-aware virtual machine placement in national cloud data centers. IEEE Trans Emerg Top Comput 4(2):290–300. https://doi.org/10.1109/TETC.2015.2508383
27. Coffman EG, Garey MR, Johnson DS (1997) Approximation algorithms for bin backing: a survey. In: Approximation Algorithms for NP-Hard Problems. Springer. pp 46–93
28. Coffman EG, Csirik J, Galambos G, Martello S, Vigo D (2013) Bin packing approximation algorithms: Survey and classification. In: Handbook of Combinatorial Optimization. Springer
29. Johnson DS (1973) Near-optimal bin packing algorithms. Ph.d. thesis, Massachusetts Institute of Technology, Dept. of Mathematics http://hdl.handle.net/1721.1/57819
30. Johnson DS, Demers A, Ullman JD, Garey MR, Graham RL (1974) Worst-Case Performance Bounds for Simple One-Dimensional Packing Algorithms. SIAM J Comput 3(4):299–325. https://doi.org/10.1137/0203025
31. Calheiros RN, Ranjan R, Beloglazov A, Cesar aFDR, Buyya R (2011) CloudSim:a toolkit for modeling and simulation of cloud computing environments and evaluation of resource. Softw - Pract Exp 41(1):23–50
32. Shen S, Van Beek V, Iosup A (2015) Statistical characterization of business-critical workloads hosted in cloud datacenters. Proc - 2015 IEEE/ACM 15th Int Symp Clust Cloud, Grid Comput CCGrid 2015:465–474. https://doi.org/10.1109/CCGrid.2015.60. arXiv:1302.5679v1
33. Park KS, Pai VS (2006) Comon: a mostly-scalable monitoring system for planetlab. ACM SIGOPS Oper Syst Rev 40:65–74
34. Huang Q, Gao F, Wang R, Qi Z (2011) Power consumption of virtual machine live migration in clouds. In: 2011 Third International Conference on Communications and Mobile Computing. pp 122–125. https://doi.org/10.1109/CMC.2011.62
35. Montgomery DC, Runger GC (2003) Applied Statistics and Probability for Engineers. 3rd Edition. Wiley Press. pp 277–656