Cluster Computing (2019) 22:635–647
https://doi.org/10.1007/s10586-018-2869-5

Network-aware energy saving multi-objective optimization in virtualized data centers

Motassem Al-Tarazi[1] · J. Morris Chang[2]

Received: 8 April 2018 / Revised: 19 September 2018 / Accepted: 15 November 2018 / Published online: 21 November 2018

Abstract
With the current growth of data centers, improving energy saving is becoming more important to cloud service providers. The data centers architectural design and the advancement of virtualization technologies can be exploited for energy saving. In this paper, we studied the energy saving problem in data centers using virtual machines placement and live migration taking to account the status of the network links load. The problem was formulated as multi-objective integer linear program, which solvable by CPLEX, to minimize the energy consumed by the servers and minimize the time to migrate virtual machines. To overcome CPLEX high computation, a heuristic algorithm is introduced to provide practical and efficient virtual machines placement while minimizing their migration overhead to the network. The heuristic is evaluated in terms of energy consumed and performance using a real data center testbed that is stressed by running Hadoop Hibench benchmarks. The results where compared to the ones obtained by distributed resource scheduler (DRS) and the base case. The results show that the heuristic algorithm can save up to 30% of the server's energy. For scalability and validity of optimality, the results of the heuristic were compared to the ones provided by CPLEX where the gap difference was less than 7%.

Keywords Data centers · Multi-objective optimization · Energy saving · Virtual machine placement

1 Introduction

The growth and popularity of cloud computing services is leading toward the rise of large-scale data centers. Current data centers sizes tend to have tens to hundreds of thousands of servers in order to provide massive and sophisticated services, such as web searching, cloud storage, online social services, and scientific computing [1]. The growth of data centers made it one of the most energy consumed categories in the world. The United States Environmental Protection Agency (EPA) reported that the total electricity used by data centers in 2010 was about 1.3% of all electricity used in the world [2] and it is expected to reach 8% by 2020 [3]. The problem of saving data centers energy is important and challenging for cloud service providers especially with current data center designs and the advancement of virtualization technologies.

Extensive research has been done in the literature to provide solutions to overcome the high data centers energy consumption problem. As the highest source of energy consumption in data centers, most researchers focus their approaches and techniques on finding solutions to the server-side data center energy saving problem [4–7].

The limitations and drawbacks of the approaches and techniques provided in the literature can be categorized into one of the following five cases: first, they fail to satisfy all server resources requirements (CPU, memory, network, and disk), at the same time, in their solution [8–12]. Such techniques will address only the considered resources leaving others as potential performance bottlenecks. Second, some techniques do not scale to the size of current data centers due to lack of computational efficiency. Third, providing solutions just for the initial placement ignoring

✉ Motassem Al-Tarazi
altarazi@iastate.edu

J. Morris Chang
chang5@usf.edu

[1] Computer Science Department, Iowa State University, Ames, IA 50011, USA

[2] Department of Electrical Engineering, University of South Florida, Tampa, FL 33647, USA

the load variations that might happen afterwards [13–17]. Four, the evaluations of some techniques were carried out through simplified simulations running workloads that do not represent real data centers daily workloads. Finally, most techniques try to minimize data center power consumption through virtual machine (VM) migrations while ignoring its effect on network links. This may result in moving VMs over links that are already congested/near congested, leaving the data center network vulnerable to sudden traffic surges. To address this problem, some techniques have proposed a network-aware energy saving techniques [18, 19], however, there solutions were mainly focused on the distance between servers (represented by hop count) and the migration cost effect on the source and destination servers without considering the current traffic on network links.

In this paper, we propose a weighted sum multi-objective optimization for data centers energy saving taking into account the effect of virtual machines migration on network links. The multi-objective optimization is part of a framework that monitors the state of the data center by collecting run time utilization data for servers' resources (CPU, memory, network, and disk). It uses them as an input for the multi-objective optimization. The multi-objective optimization will provide a new virtual machines placement that assure maximum energy saving with minimum effect on the underlying network. Live migration commands will take place to adjust the placement of the virtual machines into their designated destinations based on the optimization solution. Finally, unused servers are set into standby mode.

For large-scale data centers, the running time for the multi-objective optimization, which is solvable via CPLEX, is computationally inefficient. For example, the multi-objective optimization runs for more than 37 h to provide a solution for a data center with 1500 virtual machines. So, for practical implementation on large scale data centers, a two-phase greedy heuristic algorithm is introduced. The first phase targets finding an initial feasible placement for the virtual machines that satisfies all the resource utilization constraints with minimum migration time. After finding an initial feasible placement, the second phase tries to find an optimal/near optimal solution to efficiently save the data center energy without violating the utilization constraints. To achieve this solution, the heuristic algorithm consolidates virtual machines to a small subset of servers that satisfy their requirements and put unused servers to standby mode. The heuristic searches for the best routes to consolidate virtual machines with minimum effect on the network links. The heuristic continuously monitors the state of each virtual machine and present a new placement if a resource violation occurs or a better solution can be obtained. Live migration moves virtual machines between servers with minimum down time.

To evaluate the efficiency, applicability, scalability, and optimality/near optimality of the proposed framework and the heuristic algorithm, extensive experiments were conducted. The evaluation was divided into two parts: First, experiments on a testbed data center, that is built using VMware vSphere 5.5 suite, were conducted to evaluate performance and energy saving.

Hadoop 2.7.3 multi-node cluster is deployed on the testbed to mimic real data centers environment, while the testbed is stressed using different workloads from Hibench [20]. The framework is compared to the base case, where no energy saving mechanism is used, as well as VMwares' distributed resource scheduler (DRS). The experiments show that the proposed framework can save energy up to 30% while achieving better performance with minimum effect on the data center network. Second, to evaluate scalability and validity of optimality of the heuristic algorithm, the solutions of the heuristic algorithm were compared to the optimal ones provided by CPLEX. The comparison shows that the gap between the optimal energy consumption and the ones provided by the heuristic algorithm is at most 7%. Meanwhile, the heuristic algorithm can reduce the computation time significantly compared to CPLEX.

The list of contributions in this paper is as follows:

- We propose a dynamic virtual machine framework with the objective to minimize energy consumption and virtual machines migration effect on the network. The proposed framework actively monitors workload runtime fluctuations and provides dynamic placement solutions.
- A multi-objective optimization formulation for server-side energy saving and time to migrate virtual machines is introduced. The optimization considers all servers resources in its solution (CPU, memory, network, and disk) such that energy wastage and performance bottlenecks caused by resource wastage are eliminated.
- We present a two-stage greedy heuristic algorithm that achieves near-optimal energy saving and low computational complexity. This heuristic is practical solution for large size data centers.
- The heuristic algorithm was evaluated on a real testbed data center. The heuristic was compared with industry leading design VMware's DRS and the base case to demonstrate the effectiveness of the heuristic algorithm in regard of performance and energy saving.

The rest of the paper is organized as follows. Section 2 reviews previous related works. Section 3 introduces the proposed system framework. Section 4 formulates the multi-objective power saving problem. Section 5 shows the

proposed heuristic algorithm. Section 6 discusses how the testbed data center was implemented and presents the experimental results, and finally Sect. 7 concludes the paper.

2 Related works

Many approaches have been proposed to deal with the data centers energy saving problem. A number of researchers proposed designs of new topological structures that provide energy conservation while preserving performance. Examples may include flatted butterfly [21], Pcube [22], Small-World [23], NovaCube [24], 3D Torus based CamCube [25], Nano Data Centers [26], and Proteus [27]. The primary drawback of these new topologies is that they cannot be applied to existing data centers as they require specific hardware and software capabilities.

Other researchers focus on saving energy of the data center network (DCNs). They found optimization problems for current DCNs and propose different techniques and heuristics to solve them. The main idea is to switch the network traffic to a subset of switches and turn off unused devices. Many approaches use such technique such as ElasticTree [28], Carpo [29], REsPoNse [30], GreenTE [31], Merge network [32], and many others [33–36]. The main concerns in these studies include: the trade-off between energy saving and network performance and how to deal with sudden traffic surges. It should be noted that the amount of energy to be saved by these DCNs techniques is much less compared to the data canter server's energy saving techniques.

Most researchers focus their efforts toward server-side energy saving since the servers are the most energy consuming devices in the data centers and with the advancements of virtualization technologies which provide great opportunities for energy saving. Some studies target only static placement [13–17], these studies consider the initial placement of virtual machines ignoring workloads fluctuation. Other studies suggest live migration for dynamic virtual machine placement [10, 37, 38]. Such studies ignore the overhead produced by the live migration and its effect on the network. This will lead to placement solutions that require virtual machines to be migrated over congested links or to long distances. For that reason, some researchers propose network-aware virtual machine placement mechanisms [18, 19], they consider the hop-count between the source and destination hosts for migration, the cost of the migration, inter-related virtual machines, and the power consumed during the migration process to minimize the migration overhead and avoiding long distance migrations. The major drawback of these mechanisms is that they don't consider the current status of the network. Thus, they might migrate virtual machines through routes that are shorter, but already congested or almost congested.

This work overcomes previous studies drawbacks. It takes advantages of the virtualization technologies, uses live migration for dynamic placement while considering all servers resources (CPU, Memory, Network, and Disk). The work also considers the current network status, thus, migrating virtual machines to the most suitable servers with minimum effect on the network. The work is applicable since it was applied to a real data center testbed and evaluated using the widely use Hibench benchmark suite.

3 System model

Figure 1 illustrates the three major modules of the proposed framework (Resources measurements module, multi-objective optimization module, and Next placement module).

The resources measurements module is responsible for the continuous monitoring of the data center and for collecting virtual machines resources utilization (CPU, memory, network, and disk). Furthermore, the module also extracts the network current traffic matrix.

The multi-objective optimization module is a weighted sum integer linear program that takes the collected virtual machines resource utilization and the network traffic matrix as an input. The optimization will provide a Pareto solution that minimizes the energy consumed by the data center, meanwhile minimizes the effect of virtual machines migration on the network. One of the results of this module is a migration matrix, which includes the virtual machines that need to be moved from their current hosting server to another target server.

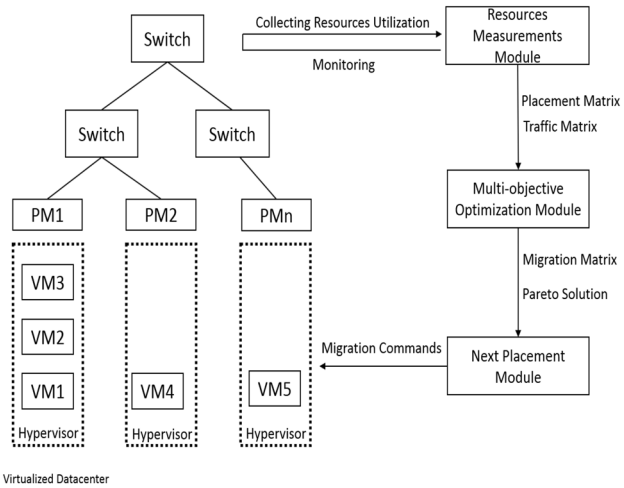

Fig. 1 Proposed system framework

The last module is the next placement module, which is responsible for sending live migration commands based on the solution provided by the optimization. After all live migration commands completed successfully, the module will put unused servers to standby mode.

4 Problem formulation

In this section, we present the weighted sum multi-objective optimization formulation to minimize the power consumed by servers and the effect of migrating virtual machines on the data center network. The virtual machines migration effect is calculated by finding the time needed for a virtual machine to travel from a source server to a destination server using the current network traffic. Consider a data center $G = (PM, E)$ where PM is the set of all physical machines (servers) and E is the set of all links. The formulation is divided to two parts: minimizing power consumed by servers and the network.

The power consumed by a single physical machine follows the model proposed by [39, 40], expressed by Eq. 1. The model shows that the server average power is approximately linear with respect to CPU utilization. The model has been proven to be accurate for large scale data centers.

$$P^{server} = P^{active} + P^{dynamic} \cdot Util^{CPU} \tag{1}$$

where P^{server} is the total power consumed by the server, $P^{dynamic}$ is the dynamic power consumption of the CPU, $Util^{CPU}$ is the average CPU utilization, and P^{active} is the power consumption when the CPU is idle.

For the servers' part, a server $p \in PM$ can provide a set of resources bounded by an upper utilization threshold (U_p^{CPU}, U_p^{Mem}, U_p^{Net}, and U_p^{Disk}). Let VM be the set of all virtual machines to be hosted by physical machines. Each virtual machine $\upsilon \in VM$ requests a specific amount of resources (denoted by $\upsilon m_\upsilon^{CPU}$, $\upsilon m_\upsilon^{Mem}$, $\upsilon m_\upsilon^{Net}$, and $\upsilon m_\upsilon^{Disk}$) to be consumed. A physical machine can host many virtual machines as long as its resource utilization thresholds are not violated.

A physical machine can be turned into standby mode if there is no active v hosted by p. A physical machine in standby mode consumes $P^{standby}$, meanwhile, an active physical machine consumes P^{active} in addition to the power consumed by each virtual machine hosted by that physical machine P^{vm}. We use decision variables On_i to denote the current power status of physical machines (i.e. active or standby mode) and $M_{\upsilon p}$ to present the current placement of virtual machines on physical machines.

A virtual machine v can move from one physical machine to another either to minimize power consumption or to solve resource utilization threshold violation. A $g_{\upsilon p}$ is a decision variable to denote which virtual machine is migrated and its target physical machine.

With the notations summarized in Table 1, we can formulate our problem as a weighted sum multi-objective integer linear program that is solvable by CPLEX as the following: the ILP takes the physical machines PM, the virtual machines VM, utilization thresholds for resources U_p^c, virtual machines current resource utilizations υm_υ^c, the power of physical machines in active and standby modes $P^{standby}$ and P^{active}, and the power of each virtual machine P^{vm} as inputs.

The constraints are divided into four categories: placement, power, resource, and network. Equations 2 and 3 shows the placement constraints, they assure the correct placement of virtual machines on their designated physical machines. Equation 2 states that each virtual machine has to be and can only be served by one physical machine. Equation 3 illustrates that if a virtual machine decided to migrate from its current physical machine to a new one, then the value of its current placement $M'_{\upsilon p}$ and next placement should change (i.e. $M'_{\upsilon p} \neq M_{\upsilon p}$).

$$\sum_{p \in PM} M_{\upsilon p} = 1, \quad \forall \upsilon \in VM \tag{2}$$

$$M_{\upsilon p} - M'_{\upsilon p} \times M_{\upsilon p} = g_{\upsilon p}, \quad \forall \upsilon \in VM, p \in PM \tag{3}$$

Equations 4 and 5 present the power constraints, they state that a physical machine operates in active mode if and only if it needs to serve an active virtual machine. Specifically, Eq. 4 assures that a physical machine can be turned into standby mode only if there is no virtual machine placed on it. Likewise, Eq. 5 shows that a virtual machine can only be placed on an active physical machine.

$$On_p \leq \sum_{\upsilon \in VM} M_{\upsilon p}, \quad \forall p \in PM \tag{4}$$

$$M_{\upsilon p} \leq On_p, \quad \forall \upsilon \in VM, p \in PM \tag{5}$$

The power consumed by a physical machine is calculated in Eq. 6. It is based on the status of the physical machine and the virtual machines hosted on that machine.

$$P(p) = P^{active} On_p + P^{standby}(1 - On_p) + P^{vm} \sum_{\upsilon \in VM} M_{\upsilon p} vm_\upsilon^{CPU} \tag{6}$$

Equations 7–9 introduce the resources utilization constraints. Since virtual machines deployed on a physical machine require some amount of resources, these resources should not exceed a specific utilization level from the resources offered by the physical machines (in this paper, we consider the utilization threshold = 70%). In this

Table 1 Definition of important symbols

Symbol	Definition
PM	Set of all physical machines
VM	Set of all virtual machines
E	Set of all links
vm_v^c	Utilization of resource c by VM v
U_p^c	Utilization threshold of resource c at PM p
i, j	A link connects two nodes i and j
$P^{standby}, P^{active}$	Power consumed by a PM in standby and active modes, respectively
P^{vm}	Power consumed by a VM
bd^e	Bandwidth of link e
vbd^e	Bandwidth consumed by virtual machines on link e
$sizeof(v)$	Size of VM fingerprint
γ	Tuning variable for weighted-sum
M_{vp}	Virtual machines placement matrix
On_p	Physical machine power mode matrix
g_{vp}	Virtual machines migration matrix

formulation, all types of resources are considered (CPU, memory, network, and disk).

$$\sum_{v \in VM} (M_{vp} \times vm_v^{CPU}) \leq U_p^{CPU}, \quad \forall p \in PM \tag{7}$$

$$\sum_{v \in VM} (M_{vp} \times vm_v^{Mem}) \leq U_p^{Mem}, \quad \forall p \in PM \tag{8}$$

$$\sum_{v \in VM} (M_{vp} \times vm_v^{Disk}) \leq U_p^{Disk}, \quad \forall p \in PM \tag{9}$$

For network resources, previous studies focus their effort to the network bandwidth consumed by physical machines, ignoring the bandwidth of all network links, in order to simplify their formulation. In this formulation, we consider the pipe model to express bandwidth constraints on all network links.

Suppose that τ is the current communication matrix between VMs where $\tau_{v,w}^e$ is 1 if the virtual machines v and w are communicating through link e with bandwidth $bd_{v,w}^e$. Equation 10 calculates bandwidth consumed by all virtual machines communicating through link e. Equation 11 ensures that the bandwidth passing through link e is less than the capacity of that link by a utilization threshold.

$$vbd^e = \sum_{v,w \in VM} \sum_{p,d \in PM} M_{vp} M_{wd} min(vm_v^{Net}, vm_w^{Net}) \cdot \tau_{v,w}^e bd_{v,w}^e \tag{10}$$

$$\sum_{e \in E} \frac{vbd^e}{bd^e} \leq U_e^{Net} \tag{11}$$

To be able to solve the problem using software solver such as CPLEX, we need to linearize the bandwidth capacity constraint in Eq. 10 which is in bi-linear form [41]. The problem can be linearized by introducing variables $x_{vwpd} \in [0, 1]$ that verify the following constraints:

$$x_{vwpd} \leq M_{vp}, \quad \forall v, w \in VM, \ \forall p, d \in PM \tag{12}$$

$$x_{vwpd} \leq M_{wd}, \quad \forall v, w \in VM, \ \forall p, d \in PM \tag{13}$$

$$M_{vp} + M_{wd} - 1 \leq x_{vwpd}, \quad \forall v, w \in VM, \ \forall p, d \in PM \tag{14}$$

It can be seen that for a given $M_{vp}, M_{wd} \in \{0, 1\}$, $M_{vp} \times M_{wd} = x_{vwpd} \in [0, 1]$. So, Eq. 10 can be rewritten as:

$$vbd^e = \sum_{v,w \in VM} \sum_{p,d \in PM} x_{vwpd} \cdot min(vm_v^{Net}, vm_w^{Net}) \cdot \tau_{v,w}^e bd_{v,w}^e \tag{15}$$

By replacing Eq. 10 with Eqs. 12–15, the problem becomes ILP. It should be noted that this linearization is valid only if M_{vp} are integer variables.

The second part of the multi-objective optimization focuses on the time required for a virtual machine to migrate. For a virtual machine v to be migrated, Eq. 16 calculates *Troute* which is the time required for v to pass through the set of links that form a route between the source host p and destination host d (i.e. $E*$). The time for a virtual machine to travel through a link depends on the virtual machine size and current link traffic.

The virtual machine size plays an essential role in calculating the migration time. The virtual machine size represents its state information; this includes its current memory contents and all information that uniquely defines and identifies the virtual machine. The memory contents include the data and instructions of the operating system and the applications that are in the memory. The defining and identification information consist of all the data that

maps to the virtual machine hardware elements such as BIOS, I/O devices, CPU, MAC addresses for the Ethernet cards, chip set states, registers...etc. Generally, memory contents are very large compared to the state defining and identification data, thus, we will consider the size of the virtual machine as the size of its memory contents. For a predefined set of routes (r_n) between the source host p and destination host d, equation 17 chooses the minimum routing time to migrate a virtual machine v to its target.

$$Troute(\upsilon, p, d, E*) = \sum_{e \in E*} \frac{sizeof(\upsilon)}{U_e^{Net} bd^e - \upsilon bd^e}, E* \subseteq E \quad (16)$$

$$T(\upsilon, p) = min(Troute(\upsilon, p, d, r_1), ..., Troute(\upsilon, p, d, r_n)) \quad (17)$$

Lastly, Eq. 18 presents the objective function to minimize the power consumed by physical machines as well as the time to migrate virtual machines. The function uses a weighted sum tuning variable $\gamma \in (0, 1]$. It is an indication of to what extent the cloud provider is willing to sacrifice some the energy saving to decrease migration overhead on the network. γ can be set to large value to indicate that the cloud provider is very keen on energy saving and to a small value when the provider is more concerned about the migration effect on the network. If γ is set 1, the problem became an energy saving problem without taking the migration overhead into consideration. It is nonsensical to set γ to 0, as the problem will not allow any virtual machine migration to occur.

$$Minmize \quad \gamma \sum_{p \in PM} P(p) + (1 - \gamma) \sum_{p \in PM} \sum_{\upsilon \in VM} T(v, p) g_{vp} \quad (18)$$

Since multi-objective integer linear programming is NP-hard, the proposed formulation is not practical for large data center networks. Thus, it can be used as a benchmark tool to evaluate practical heuristic approaches.

5 Heuristic approach

To overcome the exponential increase in CPLEX computation time, a heuristic algorithm solving the data center energy-saving problem was developed. In data center environment, workload demands fluctuate frequently. For that reason, heuristic algorithm is preferred to solve our optimization model in real time. Algorithm 1 and 2 illustrate the two-stage heuristic pseudocode, it takes similar inputs as in CPLEX and it is implemented using java programming language with vSphere SDK. The output includes the next placement matrix M, the physical machines power mode matrix On, and the migration matrix g.

Algorithm 1 Heuristic Algorithm

```
Stage 1: Finding initial feasible solution
Input: PM, E, VM, U^CPU, U^Mem, U^Net, U^Disk, vm^CPU, vm^Mem, vm^Net, vm^Disk, P_active, P^standby, P^vm, bd^e, vbd^e
Output: M, On, g
for PM is active do
    if resource exceeds p ∈ PM utilization threshold then
        Sort VMs in p in descending of the resource
        Find topVMs to solve the violation
        for v ∈ topVMs do
            Find targetPMs
            for p ∈ targetPMs do
                Calculate time to migrate v to p
            end for
            Record the lowest time to migrate v to p
        end for
        Migrate VMs with the lowest scores
        if The violation solved then
            Return
        end if
    end if
end for
if No feasible solution exists then
    Adopt alternative strategy to handle violation
end if
```

Stage 1 starts with the objective of finding an initial placement that satisfies all virtual machines requirements. The algorithm traverses all physical machines searching for a resource utilization violation. A virtual machine placement solution is considered feasible if all virtual machines resource requirements are satisfied and all physical machines consumed resources are within the physical machine resource utilization threshold. If there exists a physical machine that violate utilization threshold for one of its resources, the solution is considered unfeasible and countermeasure operations should be applied to solve the violation.

Algorithm 2 Heuristic Algorithm - Continue

```
Stage 2: Improving energy-efficiency
loop
    list = Sort PM based on energy consumption
    for the first p ∈ list do
        Check consolidation options
        if consolidation options are available then
            Calculate time to migrate vms from p
            Migrate to the lowest time
            Turn p to Standby mode
            Update list
        else
            Remove from p from list
        end if
        if list is empty then
            return M, On, g
        end if
    end for
end loop
```

For a physical machine with a resource utilization violation, the algorithm sorts all the virtual machines currently

placed on that physical machine in descending order based on that resource type. Then, finding the set *topVMs*, which includes the virtual machine or the set of virtual machines where their migration will solve the resource violation.

For each element in *topVMs*, a *targetPMs* set is formed where each element in it represents a physical machine that is a potential destination. To avoid high computational running time, since this set might contain large number of elements considering today's data centers sizes, the search for a target physical machine will be within a rack or a pod. Next, the migration time for each potential destination is measured based on the network current traffic and the virtual machine size, and the lowest migration time is recorded such that there is no conflict on the network routes. Finally, the virtual machine(s) with the lowest recorded time will be migrated to its destination. The process will be repeated for all physical machines until a feasible solution is reached. If no feasible solution exists, other alternative strategies can be adopted to solve the resource utilization violation.

After an initial feasible solution is found, stage 2 of the heuristic algorithm aims to improve the current energy consumption while presenting minimal overhead to the network links. The adopted strategy moves virtual machines to a subset of physical machines and puts unused physical machines to standby mode.

This stage begins by calculating the power currently consumed by each physical machine and sorting them, into *list*, in ascending order based on their power consumption. For the first physical machine p in the ordered *list*, the algorithm searches for available consolidation options in order to move all virtual machines currently residing on that physical machine. The consolidation options are the set of physical machines that can handle the virtual machines currently hosted by p without violating the resource utilization thresholds (Eqs. 7– 9). For large data centers, this set might become very large, so the consolidation options candidates are limited to the hosts within p's pod or rack. If found, for each virtual machine, calculate the migration time to move it to the target physical machine with the lowest migration time. Note that the communication matrix will be updated after each vm migration to avoid conflicts on the network route. When all virtual machines hosted by the physical machine migrated to their target physical machine(s), the physical machine will be set to standby mode, removed from *list*, and *list* will be updated. If no consolidation option found, the physical machine p will be removed from *list*. This iterative process is repeated until *list* is empty; thus, no further improvement can be made to the solution and the algorithm returns the next placement matrix, the physical machines power matrix, and the migration matrix.

Overall, the computation complexity for the proposed heuristic algorithm is $\mathcal{O}(PM^2 VM \log PM \log VM)$.

6 Evaluation and experimental evaluation

This section presents the evaluation of our proposed heuristic algorithm. The evaluation is conducted to show that the proposed heuristic algorithm is efficient, applicable, scalable, and can achieve considerable amount of energy saving to the data center while maintaining network performance. The evaluation is divided into two parts: in the first part, a three-node data center testbed is built and stressed through fluctuation workloads using Hibench 6.0. Hibench is a benchmark developed by Intel to evaluate the performance of MapReduce jobs running in data centers for both Hadoop and Spark. As benchmark loads are running, the proposed heuristic will adjust the locations of the virtual machines in order to avoid physical machines resource utilization limit violation and to save energy.

The heuristic algorithm results were compared to the ones obtained by the VMware's distributed resource scheduler (DRS) [7]. DRS is used to manage the placement of the virtual machines within a cluster. DRS focuses on balancing the load across all physical machines by calculating the cluster imbalance score I_c and made VM migration decisions to minimize or maintain it under a given threshold. The imbalance score is the standard deviation of the load over all physical machines. DRS periodically (every 5 min by default) invokes a greedy hill-climbing algorithm to calculate the cluster imbalance score and make migration decisions.

An extended feature of DRS called distributed power management (DPM) [42] is used to save energy by moving virtual machines from lightly loaded physical machines and puts physical machines with no virtual machines into standby mode. DPM periodically search each physical machines' resources utilization and provide recommendations for energy saving if the load for a physical machine is lower than a predefined threshold. Furthermore, the heuristic is also compared to the base case, where no virtual machine migration is allowed. The evaluation was conducted using the same workload for all designs and each test was repeated three times and the average was recorded.

In the second part, we investigate the computation time and energy consumption of the optimal solutions obtained by CPLEX 12.7 and compare them to the ones achieved by our proposed heuristic. This will prove the scalability and optimality/near optimality of our proposed heuristic algorithm.

6.1 Testbed setup

The testbed data center is built using VMware vSphere suite 5.5 to prove the applicability, energy efficiency, and performance effectiveness of the proposed framework.

Currently, the testbed is configured using three physical machines to host virtual machines. Each physical machine is equipped with a quad-core 3.4 GHz Intel i7 processor and 32 GB of memory. The physical machines use a 1 Gbps private network for communication and virtual machines migrations. Figure 2 shows the testbeds' network topology. An ESXi 5.5 hypervisor is running on each physical machine for deploying and serving virtual machines. Note that the energy consumed by the physical machines were measured using Kill a watt [43] energy monitoring device.

MainCenter is a virtual server that has a vCenter tool to manage and control all events happened on the data center such as initiating migration commands and enter/exit a physical machine from standby mode. Furthermore, the vCenter collects runtime performance measurements and saves them to a Microsoft SQL Server 2005 database. Another virtual server called DNS provides domain name, Active directory domain, and network storage services. Both virtual servers are running Windows server 2008 R2 with 4 GB of memory and 60 GB of storage.

Furthermore, a 13 Linux Ubuntu 16.04 LTE virtual machines are deployed in the data center testbed. They are equipped with an iSCSI network storage that is accessible by all physical machines. The virtual machines share a 1 TB iSCSI storage and they use 1 Gbps vMotion network for live migration. A Hadoop multi-node cluster is configured using Apache 2.7.3 to evaluate the testbed using MapReduce benchmarks. The Hadoop cluster runs in default settings and it includes one master node (name node) and 12 slave nodes (data nodes). Each node has a 2 GHz CPU capacity, 4 GB of memory, and 40 GB of storage.

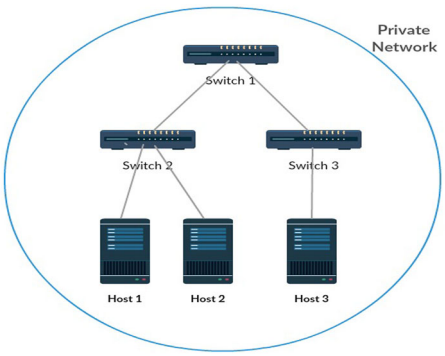

Fig. 2 Testbed network topology

To increase the utilization of a single node, Docker containers is used. Dockers containers help in running distributed applications within a Linux instance. This technology is becoming popular for applications in cloud environment since there is no need for deploying and managing new virtual machines, thus, reducing overhead. The estimation of a virtual machine resource usage is based on its history resource usage. Finally, it should be noted that we use a 1-min threshold for initiating virtual machines migrations.

6.2 Testbed results

Figure 3 shows the network bandwidth consumed by the testbed servers when managed by VMware's DRS. For a 15 min period, the results illustrate that servers managed by DRS show large variation in network bandwidth consumption for each server. For example, server three only consumes up to 13% of its network bandwidth. Meanwhile, server two consumes no lower than 51% and up to 82% of its network bandwidth. This large variation happens since DRS makes migration decisions to maintain the imbalance score I_c, which is mainly based on CPU and memory resources, while network resources are being ignored.

This situation might produce a bottleneck that affect the performance of the data center, especially if multiple virtual machines that require high network bandwidth and low CPU and memory resources are placed on the same server. Thus, the server will have available CPU and memory resources, but it will not be able to host new virtual machines. This problem is known as resource contention problem.

On other the hand, Fig. 4 shows the network bandwidth consumed by the testbed servers when managed by our heuristic algorithm. Starting with high variations of

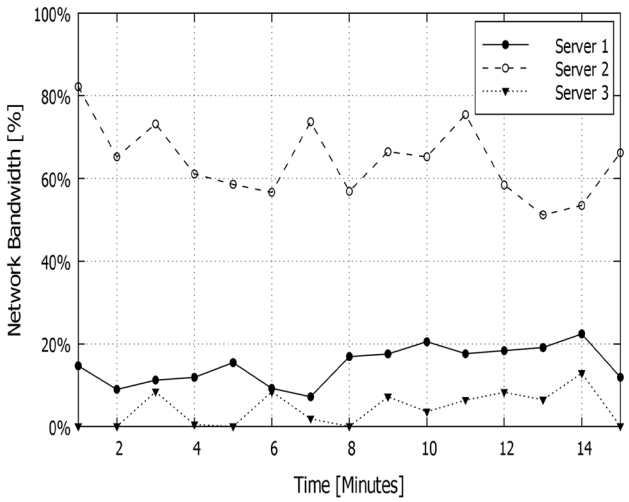

Fig. 3 Network bandwith consumption using DRS

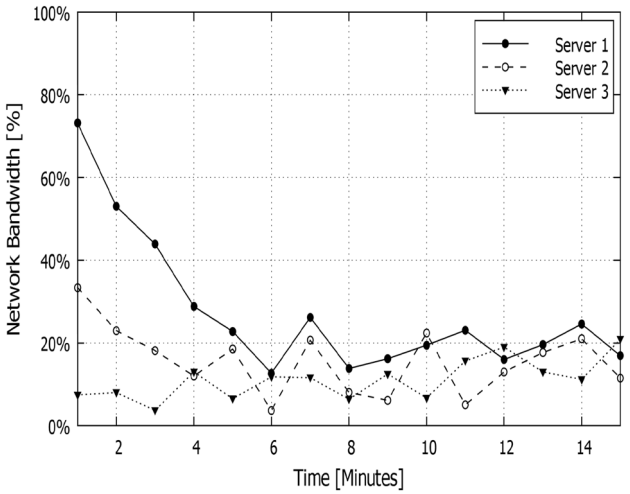

Fig. 4 Network bandwith consumption using heuristic algorithm

network bandwidth consumption, the heuristic algorithm detects this variation and provides new placement to overcome any potential bottleneck. The heuristic algorithm invokes migration commands to balance the network resource (it also considers balancing CPU, memory and disk resources) with minimum overhead since the virtual machines are migrated to the target machine with minimum migration time. In Fig. 4, within 5 min, server 1 network bandwidth consumption is decreased from 73% to around 23% and the heuristic algorithm maintains the network bandwidth consumption balancing between the testbed servers afterwards.

For the evaluation of the Hadoop cluster configured on the testbed, Hibench benchmarks: Wordcount, TeraSort, PageRank, and Kmeans are used [20]. Figure 5 shows the energy consumed by the testbed when running each benchmark for the base case, DRS, and the heuristic algorithm. The results clearly show that the heuristic

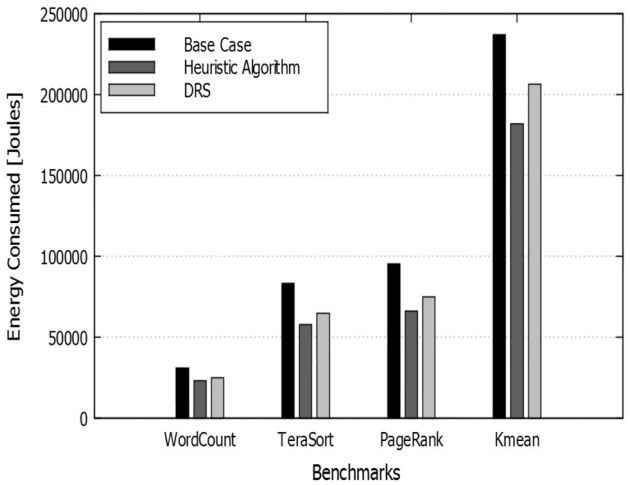

Fig. 5 Testbed energy consumption

algorithm outperforms the base case and DRS for all benchmarks. For the Wordcount benchmark, which is MapReduce job used to count the occurrence of each word in a randomly generated text, the heuristic algorithm and DRS have almost the same energy consumption while the base case consumes more energy. The energy saved by the heuristic is small, this is because of the overhead introduced by the virtual machines live migration. Terasort benchmark sorts a randomly generated text. The heuristic algorithm consumes around 57.6 KJ while DRS and base case consume 64.8 and 83 KJ, respectively. To evaluate web searching, PageRank benchmark is used. It is an implementation of Google's web page ranking algorithm. The PageRank MapReduce job is to rank 500000 web pages using three iterations. Using PageRank benchmark, the heuristic algorithm consumes less energy than DRS and the base case.

For all benchmarks, it is obvious that the base case cannot deal with the resource contention problem, thus, it needs more time to finish the jobs and consumes more energy. Furthermore, DRS reacts periodically to detect and solve the resource contention problem (every 5 min), so the servers need to wait until the DRS is invoked. These servers will suffer extra energy to be consumed. The heuristic algorithm detects the contention problem and solve it quickly via live migration. The live migration will introduce extra overhead with small performance degradation for the application on the migrated virtual machine during the migration process. The heuristic algorithm migrates virtual machines with minimum migration time to reduce such overhead. Finally, Kmean benchmark is a MapReduce job for machine learning. The job is to cluster 20 dimensions, 20 million samples into five clusters with $K = 10$ and maximum iterations is 5. The heuristic algorithm consumes around 182 KJ of energy compared to 206.4 KJ for DRS and 236.9 KJ for the base case.

Similarly, Fig. 6 shows the computational running time for the base case, DRS, and the heuristic algorithm when running Hibench benchmarks: WordCount, TeraSort, PageRank, and Kmeans. Like the consumed energy, the heuristic algorithm is more efficient than the base case and DRS in terms of computational efficiency for all tested benchmarks.

The number of virtual machines migrations is an indication of the data center stability. In this experiment, we ran the Hibench workloads (Wordcount, TeraSort, PageRank, and Kmean) all at once, and record the number of virtual machine migrations. Table 2 show the number of virtual machine migrations for the testbed using the Base case, DRS, and the Heuristic algorithm. Since the Base case do not use VM migration, the number of migrations is 0. The heuristic algorithm considers all resources in its

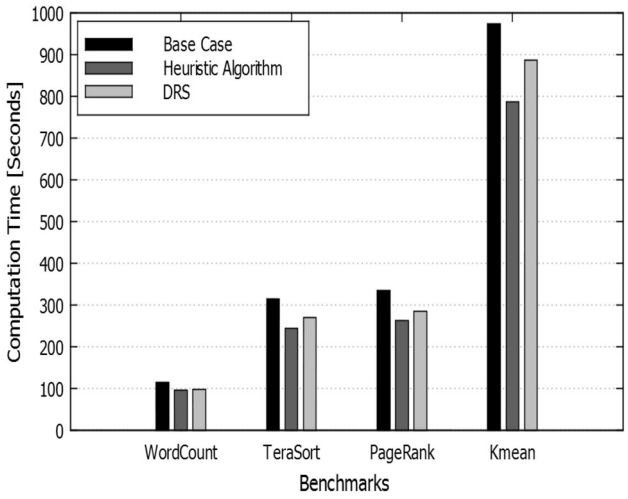

Fig. 6 Testbed computational running time

Table 2 Comparison of the number virtual machines migrations

Approach	Base Case	DRS	Heuristic
No. of VM migrations	0	23	15

solution; thus, it needs a smaller number of migrations compared to DRS.

From the evaluation, it could be concluded that the heuristic algorithm is efficient in terms of energy saving and performance. Also, the heuristic algorithm can detect and solve contention problems for all server resources. Lastly, it should be noted that the testbed is using shared storage and storage migration is not implemented in this framework yet.

6.3 CPLEX versus heuristic algorithm results

We implement the heuristic algorithm using Java programming language. Furthermore, a linearized version of the proposed multi-objective formulation is solved using CPLEX 12.7 [44]. CPLEX results provide the optimal solutions that are taken as a benchmark to evaluate how optimal are the solutions provided by the proposed heuristic algorithm. The experiments were conducted using synthetic data on an identical platform; a Linux machine with 32 Intel Xeon CPUs E5-2650 @ 2.00 GHz and 256 GB of memory.

To show the validity of optimality for our proposed heuristic algorithm, we compared the difference gap between the results provided by the heuristic algorithm with the optimal ones obtained by CPLEX. We found that the energy consumption values of our proposed algorithm are fairly close to the optimum ones for all the cases under consideration.

Figure 7 shows the differences between CPLEX and the heuristic algorithm in terms of energy consumption. The comparison was conducted for data centers hosting virtual machines ranging from 10 VMs to 1500 VMs (3 to 300 PM see Table 3 for details) with 5 min running period. The results show that the gap between the optimal energy consumption and the ones obtained by the heuristic algorithm is less than 7% for all cases. For example, a data center hosting 750 virtual machines will consume around 6461.8 KJ in the optimal case, while using the heuristic algorithm it will consume around 6925 KJ with a 6.7% difference gap between them. Although the proposed heuristic algorithm provides solutions that are slightly less than the optimal, it is much more computationally efficient.

The proposed heuristic algorithm demonstrates high computational efficiency compared to CPLEX as shown in Table 3. The growth of computational time for the proposed algorithm increases linearly with the size of the data center, whereas the growth of computational time in CPLEX increases exponentially.

There is a slight difference between the solutions obtained by CPLEX and proposed algorithm; however, solving the problem in CPLEX will introduce high computational cost. As the size of the data center goes up and hosts more VMs, in contrast with the significant boost in computation time for CPLEX, the proposed algorithm solves the problem efficiently. For example, CPLEX needs more than 940 s to find the optimal solution for a data center hosts 500 VMs. Meanwhile, the heuristic algorithm needs only 2 s. Moreover, for a data center that hosts 1500 VMs, CPLEX needs more than 37 h to provide the optimal solution while the proposed heuristic algorithm can obtain a solution in less than one minute.

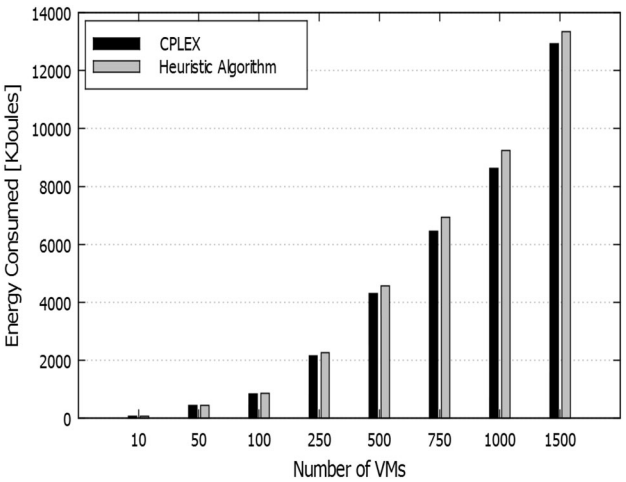

Fig. 7 Comparison of energy consumption between the proposed heuristic algorithm and CPLEX

Table 3 Comparison of computational time between the proposed heuristic algorithm and CPLEX (in seconds)

No. of VMs	No. of PMs	Heuristic	CPLEX
10	3	0.004	0.09
50	10	0.084	1.02
100	20	0.17	4.25
250	60	0.457	87.92
500	120	2.06	946.48
750	180	4.683	4873.08
1000	240	14.158	21816.22
1500	300	54.454	136040.35

The ratios of the solving time of CPLEX to that of the heuristic algorithm are considerable. They demonstrate the applicability and scalability for our proposed heuristic especially for large-scale (exascale) data centers.

7 Conclusion and future works

The current growth of data centers sizes make energy saving problem important for cloud service providers. The development of virtualization technologies provides opportunities for energy saving. In this paper, we present a framework for managing and controlling virtual machines placement on physical servers to reduce the energy consumed by data centers. Furthermore, the framework considers the current status of the network when making migration decisions. The problem was formulated as a multi-objective ILP to reduce consumed energy and minimizing migration time. The problem solution succeeded to calculate the minimum energy and migration time; however, it showed high computational complexity. Thus, for implementation purposes to large data centers a two-stage heuristic algorithm is proposed. The heuristic monitors the physical machines and virtual machines resources and reacts if a resource threshold violation occurs or a better solution is found considering the network status. The heuristic algorithm was evaluated using a real data center testbed against DRS and the base case in terms of performance and energy saving. For the cases under consideration, it was found that the heuristic algorithm can save energy while maintaining performance and introducing minimum virtual machines migration overhead to the network links. Moreover, the heuristic algorithm solutions were compared to the optimal ones obtained by CPLEX. The solutions were fairly close to the optimum ones and the heuristic algorithm provide a much better computational running time.

For future works, the proposed framework can be evaluated by a larger testbed with different virtualization platforms such as Xen. Furthermore, the proposed formulation can be part of a joint optimization that saves network and server sides of the data center.

References

1. Hammadi, A., Mhamdi, L.: A survey on architectures and energy efficiency in data center networks. Comput. Commun. **40**, 1 (2014)
2. Koomey, J., Oakland, C.A.: A Scalable, Commodity Data Center Network Architecture. Analytics Press, Berkeley (2011)
3. Gao, P.X., Curtis, A.R., Wong, B., Keshav, S.: It's not easy being green. ACM SIGCOMM Comput. Commun. Rev. **42**(4), 211 (2012)
4. Zhang, Q., Cheng, L., Boutaba, R.: Cloud computing: state-of-the-art and research challenges. J. Internet Serv. Appl. **1**(1), 7 (2010)
5. Tso, F.P., Hamilton, G., Oikonomou, K., Pezaros, D.P.: Implementing scalable, network-aware virtual machine migration for cloud data centers. In: 2013 IEEE Sixth International Conference on Cloud Computing (CLOUD), IEEE, pp. 557–564 (2013)
6. Shanmuganathan, G., Gulati, A., Holler, A., Kalyanaraman, S., Padala, P., Zhu, X., Griffith, R., et al.: Towards proactive resource management in virtualized datacenters. VMware Labs (2013)
7. VMware Infrastructure: Resource management with VMware DRS. VMware Whitepaper **13** (2006)
8. Kansal, N.J., Chana, I.: An empirical evaluation of energy-aware load balancing technique for cloud data center. Clust. Comput. 1–19 (2017)
9. Duggan, M., Duggan, J., Howley, E., Barrett, E.: A network aware approach for the scheduling of virtual machine migration during peak loads. Clust. Comput. **20**(3), 2083 (2017)
10. Bobroff, N., Kochut, A., Beaty, K.: Dynamic placement of virtual machines for managing sla violations. In: 10th IFIP/IEEE International Symposium on Integrated Network Management, IM'07, IEEE, pp. 119–128 (2007)
11. Hermenier, F., Lorca, X., Menaud, J.M., Muller, G., Lawall, J.: Entropy: a consolidation manager for clusters. In: Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (ACM), pp. 41–50 (2009)
12. Nguyen Van, H., Dang Tran, F., Menaud, J.M.: Autonomic virtual resource management for service hosting platforms, In: Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing (IEEE Computer Society), pp. 1–8 (2009)
13. Cardosa, M., Korupolu, M.R., Singh, A.: Shares and utilities based power consolidation in virtualized server environments. In: IFIP/IEEE International Symposium on Integrated Network Management, IM'09, IEEE, pp. 327–334 (2009)
14. Li, X., Qian, Z., Lu, S., Wu, J.: Energy efficient virtual machine placement algorithm with balanced and improved resource utilization in a data center. Math. Comput. Modell. **58**(5), 1222 (2013)
15. Mills, K., Filliben, J., Dabrowski, C.: Comparing vm-placement algorithms for on-demand clouds. In: IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom), IEEE, pp. 91–98 (2011)
16. Srikantaiah, S., Kansal, A., Zhao, F.: Energy aware consolidation for cloud computing. In: Proceedings of the 2008 Conference on

Power Aware Computing and Systems, vol. 10, pp. 1–5. San Diego (2008)
17. Xu, J., Fortes, J.A.: Multi-objective virtual machine placement in virtualized data center environments. In: Proceedings of the 2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing (IEEE Computer Society), pp. 179–188 (2010)
18. Yang, T., Lee, Y.C., Zomaya, A.Y.: Energy-efficient data center networks planning with virtual machine placement and traffic configuration. In: 2014 IEEE 6th International Conference on Cloud Computing Technology and Science (CloudCom), IEEE, pp. 284–291 (2014)
19. Zhang, Z., Hsu, C.C., Chang, M.: Cool cloud: a practical dynamic virtual machine placement framework for energy aware data centers. In: 2015 IEEE 8th International Conference on Cloud Computing (CLOUD), IEEE, pp. 758–765 (2015)
20. Wood, T., Shenoy, P.J., Venkataramani, A., Yousif, M.S.: Black-box and Gray-box strategies for virtual machine migration. NSDI **7**, 17–17 (2007)
21. Abts, D., Marty, M.R., Wells, P.M., Klausler, P., Liu, H.: Multi-objective virtual machine placement in virtualized data center environments. In: ACM SIGARCH Computer Architecture News (ACM), vol. 38, pp. 338–347
22. Huang, L., Jia, Q., Wang, X., Yang, S., Li, B.: Pcube: Improving power efficiency in data center networks. In: 2011 IEEE International Conference on Cloud Computing (CLOUD), IEEE, pp. 65–72
23. Shin, J.Y., Wong, B., Sirer, E.G.: Small-world datacenters, In: Proceedings of the 2nd ACM Symposium on Cloud Computing (ACM), p. 2 (2011)
24. Wang, T., Su, Z., Xia, Y., Qin, B., Hamdi, M.: NovaCube: A low latency Torus-based network architecture for data centers. In: 2014 IEEE Global Communications Conference IEEE, pp. 2252–2257 (2014)
25. Abu-Libdeh, H., Costa, P., Rowstron, A., O'Shea, G., Donnelly, A.: Symbiotic routing in future data centers. ACM SIGCOMM Comput. Commun. Rev. **40**(4), 51 (2010)
26. Valancius, V., Laoutaris, N., Massoulié, L., Diot, C., Rodriguez, P.: Greening the internet with nano data centers. In: Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies (ACM), pp. 37–48 (2009)
27. Singla, A., Singh, A., Ramachandran, K., Xu, L., Zhang, Y.: Proteus: a topology malleable data center network. In: Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks (ACM), p. 8 (2010)
28. Heller, B., Seetharaman, S., Mahadevan, P., Yiakoumis, Y., Sharma, P., Banerjee, S., McKeown, N.: ElasticTree: Saving energy in data center networks. In: NSDI, vol. 10, pp. 249–264
29. Wang, X., Yao, Y., Wang, X., Lu, K., Cao, Q.: Carpo: Correlation-aware power optimization in data center networks. In: 2012 Proceedings IEEE on INFOCOM (IEEE), pp. 1125–1133
30. Vasi, N., Bhurat, P., Novakovi, D., Canini, M., Shekhar, S., Kosti, D.: Identifying and using energy-critical paths. In: Proceedings of the Seventh Conference on emerging Networking Experiments and Technologies (ACM), p. 18
31. Zhang, M., Yi, C., Liu, B., Zhang, B.: GreenTE: Power-aware traffic engineering. In: 18th IEEE International Conference on Network Protocols (ICNP), IEEE, pp. 21–30 (2010)
32. Carrega, A., Singh, S., Bolla, R., Bruschi, R.: Applying traffic merging to datacenter networks. In: Proceedings of the 3rd International Conference on Future Energy Systems: Where Energy, Computing and Communication Meet, p. 3 (2012)
33. Wang, L., Zhang, F., Hou, C., Aroca, J.A., Liu, Z.: Incorporating rate adaptation into green networking for future data centers. In: 2013 12th IEEE International Symposium on Network Computing and Applications (NCA), IEEE, pp. 106–109 (2013)
34. Shang, Y., Li, D., Xu, M.: Greening data center networks with flow preemption and energy-aware routing. In: 19th IEEE Workshop on Local and Metropolitan Area Networks (LANMAN), IEEE, pp. 1–6 (2013)
35. Shang, Y., Li, D., Xu, M.: Energy-aware routing in data center network. In: Proceedings of the First ACM SIGCOMM Workshop on Green networking (ACM), pp. 1–8
36. Wang, T., Xia, Y., Muppala, J., Hamdi, M.: Achieving energy efficiency in data centers using an artificial intelligence abstraction model. In: IEEE Transactions on Cloud Computing, vol. 1, p. 99 (2015). https://doi.org/10.1109/TCC.2015.2511720
37. Liu, L., Wang, H., Liu, X., Jin, X., He, W.B., Wang, Q.B., Chen, Y.: GreenCloud: a new architecture for green data center, In: Proceedings of the 6th International Conference Industry Session on Autonomic Computing and Communications Industry Session (ACM), pp. 29–38 (2009)
38. Teng, F., Deng, D., Yu, L., Magoulès, F.: An energy-efficient vm placement in cloud datacenter. In: 2014 IEEE Intl Conf on High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC, CSS, ICESS), IEEE, pp. 173–180 (2014)
39. Meisner, D., Wenisch, T.F.: Peak power modeling for data center servers with switched-mode power supplies. In: Proceedings of the 16th ACM/IEEE International Symposium on Low Power Electronics and Design (ACM), pp. 319–324 (2010)
40. Chou, Y., Fahs, B., Abraham, S.: Microarchitecture optimizations for exploiting memory-level parallelism. In: ACM SIGARCH Computer Architecture News, vol. 32, IEEE Computer Society, p. 76 (2004)
41. Dai, X., Wang, J.M., Bensaou, B.: Energy-efficient virtual machines scheduling in multi-tenant data centers. IEEE Trans. Cloud Comput. **4**(2), 210 (2016)
42. Gulati, A., Holler, A., Ji, M., Shanmuganathan, G., Waldspurger, C., Zhu, X.: Vmware distributed resource management: design, implementation, and lessons learned. VMware Tech. J. **1**(1), 45 (2012)
43. kill a watt meter—electricity usage monitor. http://www.p3international.com/products/p4400.html. Accessed 29 Aug 2018
44. IBM ILOG CPLEX. Users manual for cplex 12.7 (2016)

Motassem Al-Tarazi received his B.S. and M.S. degrees in computer information systems and computer science from Jordan University of Science and Technology, Jordan. He is currently working toward his Ph.D. degree in computer science at Iowa State University. His research interests include cloud computing, wireless network, communication network and Internet technology. He is a student member of IEEE.

J. Morris Chang a professor in the Department of Electrical Engineering at the University of South Florida. He received the Ph.D. degree from the North Carolina State University. His past industrial experiences include positions at Texas Instruments, Microelectronic Center of North Carolina and AT&T Bell Labs. He received the University Excellence in Teaching Award at Illinois Institute of Technology in 1999. His research interests include: cyber security, wireless networks, and energy efficient computer systems. In the last five years, his research projects on cyber security have been funded by DARPA. Currently, he is leading a DARPA project under Brandeis program focuses on privacy-preserving computation over Internet. He is a handling editor of Journal of Microprocessors and Microsystems and the Associate Editor-in-Chief of IEEE IT Professional. He is a senior member of IEEE.