Cluster Computing
https://doi.org/10.1007/s10586-021-03341-0

An efficient energy-aware approach for dynamic VM consolidation on cloud platforms

Minhaj Ahmad Khan[1]

Received: 1 January 2021 / Revised: 4 June 2021 / Accepted: 9 June 2021

Abstract
The cloud computing environments rely heavily on virtualization that enables the physical hardware resources to be shared among cloud users by creating virtual machines (VMs). With an overloaded physical machine, the resource requests by virtual machines may not be fulfilled, which results in Service Level Agreement (SLA) violations. Moreover, the high performance servers in cloud data centers consume large amount of energy. The dynamic VM consolidation techniques use live migration of virtual machines to optimize resource utilization and minimize energy consumption. An excessive migration of virtual machines may however deteriorate application performance due to the overhead incurring at runtime. In this paper, we propose a normalization-based VM consolidation (NVMC) strategy that aims at placing virtual machines in an online manner while minimizing energy consumption, SLA violations, and the number of VM migrations. The proposed strategy uses resource parameters for determining over-utilized hosts in a virtualized cloud environment. The comparative capacity of virtual machines and hosts is incorporated for determining over-utilized hosts, while the cumulative available-to-total ratio (CATR) is used to find under-utilized hosts. For migrating virtual machines to appropriate hosts, the VM placement uses a criteria based on normalized resource parameters of hosts and virtual machines. For evaluating the performance of VM consolidation, we have performed experimentation with a large number of virtual machines using traces from the PlanetLab workloads. The results show that the NVMC approach outperforms other well-known approaches by achieving a significant improvement in energy consumption, SLA violations, and number of VM migrations.

Keywords Cloud computing · VM consolidation · Energy consumption · Resource utilization · SLA violations

1 Introduction

Cloud computing continues to evolve as a main paradigm for delivery of diverse IT services to end users. Its efficient mechanism for provision of services encompasses access to computational, storage, networking, and software resources. Through its economical, scalable, and elastic infrastructure, it benefits the users as well as the organizations providing services [6, 8, 22]. For provision of services, the data centers in cloud computing are equipped with high performance servers and other hardware resources. The resource usage of data centers is characterized with high monetary costs including operational and power consumption costs. In this regard, the virtualization technology helps in mapping multiple virtual machines to a physical machine, thereby amortizing the operational costs and the capital investment made for purchase of servers.

For efficient and reliable provision of services to end users, the QoS requirements negotiated with cloud users are documented and formalized as Service Level Agreements (SLAs). The non-fulfillment of performance requirements results in SLA violations which must be avoided to ensure QoS guarantees [4, 11, 23, 27]. While executing user applications or workloads, the resource demand for virtual machines may result in over-utilization of physical machines. Moreover, in data centers, the physical servers and their cooling systems account for a high share of power consumption [30, 36, 38]. The data centers equipped with a large number of IT devices

✉ Minhaj Ahmad Khan
mik@bzu.edu.pk

[1] Bahauddin Zakariya University, Multan, Pakistan

consume significant amount of power which ultimately converts to heat thereby requiring cooling equipment that also needs electricity. High energy consumption also gives rise to concerns on Carbon Dioxide (CO_2) emission which is a serious hindrance in attaining a green computing environment [3, 5, 14, 21, 24, 34, 43]. The high performance servers are known to consume significant amount of energy even for applications with limited workloads [15, 38].

What's the need of live migration?

For effective utilization of cloud resources, the dynamic VM consolidation and placement techniques attempt to consolidate virtual machines to minimum possible physical machines. This is accomplished through live migration of virtual machines to adequate hosts. The rest of the servers are turned to low-energy sleep or hibernate modes. The excessive live migration of virtual machines may however deteriorate performance. An effective VM consolidation strategy must therefore optimize resource utilization while minimizing energy consumption, SLA violations and the number of VM migrations. A few approaches including linear programming, genetic algorithms, bin packing, and constraint programming have been proposed in the literature to address the issues related to VM consolidation.

The classical bin-packing problem requires items of different sizes to be packed into bins, while minimizing the number of bins. The problem of VM mapping is modelled by using virtual machines as items that are to be packed into minimum number of bins that represent physical machines. The evolutionary approaches, in contrast, perform population based meta-heuristic optimization to select solution from search space containing possible solutions. These approaches include genetic algorithms (GAs), PSO, and ACO to optimize parameters through computation of fitness for possible solutions. The genetic algorithm uses evolutionary operators including crossover, mutation, and selection on a collection of possible solutions. The VM mappings are performed by using fitness criteria based on customized parameters. The PSO algorithm uses the concept of swarm intelligence by adjusting velocity and position of particles while updating local and global optimum values. The ant-colony optimization (ACO) is also a bio-inspired technique that simulates the behavior of ants for solving optimization problems. The VMs are mapped to physical hosts using pheromone trails and updating them at each iteration to fit virtual machines in physical hosts. In contrast to the above-mentioned approaches, this paper suggests a normalization-based VM consolidation (NVMC) approach for dynamically consolidating virtual machines. The proposed approach minimizes energy consumption and SLA violations by reducing the number of live VM migrations. For minimizing energy consumption, the virtual machines are migrated from under-utilized hosts to switch then in sleep/hibernate mode. The SLA violations which may incur due to host over-utilization or excessive live migration of virtual machines are limited by using comparative capacity based criteria for identification of over-utilized hosts. The overall approach works efficiently by incurring a small overhead at runtime and outperforms other approaches in terms of minimizing energy consumption, SLA violations and number of VM migrations.

Overall, this paper makes the following contributions:

- Problem formulation for the dynamic VM consolidation problem under given constraints
- Algorithm for normalization-based dynamic consolidation of virtual machines aimed at minimizing energy consumption, SLA violations, and number of VM migrations
- Performance evaluation of the proposed algorithm along with the state-of-the-art approaches using diverse configurations of virtual machines and user workloads

The rest of the paper is organized as follows. Section 2 presents and analyzes techniques proposed to address the VM consolidation problem. The context of our proposed approach including the environment and problem formulation is described in Sect. 3. Section 4 describes the proposed normalization-based VM consolidation (NVMC) algorithm. The parameters and configurations used for experimentation are given in Sect. 5 along with results obtained for various implementations. Section 6 concludes the paper with major findings and future research directions.

2 Related work

VM consolidation techniques have gained significant importance with the evolution of the virtualization technology. Several approaches have been proposed in the literature including linear programming, heuristics, and meta-heuristics based algorithms to perform various phases of VM consolidation including the detection of over-utilized and under-utilized hosts, VM migration and VM placement. These approaches target diverse goals such as optimizing energy consumption, resource utilization, SLA violations and number of VM migrations. A succinct analysis of these approaches has been presented in this section.

Various approaches using set of decisions variables, objective function and constraints through linear programming and constraint programming solvers have been implemented. A comparative analysis of LP, constraint-programming and heuristic approaches for VM consolidation is shown in Table 1. Speitkamp and Bichler [40] propose approach for server consolidation by assigning virtual servers to physical servers. The static server

Table 1 A comparative analysis of VM consolidation strategies in terms of approach, resources, and objectives

References	Approach	Resources	Objectives
Speitkamp and Bichler [40]	LP-relaxation based algorithm	CPU and memory	Minimization of server costs using constraints specifying server capacity
Dupont et al. [13]	Constraint Prog. based framework	CPU, RAM and storage	Optimizing energy efficiency and Carbon emissions
Zhang et al. [47]	Constraint Prog.	CPU, RAM and bandwidth	Optimizing cost function (related to resource utilization) for VM provisioning and minimizing number of physical nodes for VM packing
Beloglazov and Buyya [5]	Heuristic algorithms	CPU, memory and bandwidth	Minimizing SLA violations and energy consumption by finding overloaded and underloaded hosts, selecting their VMs for migration and placing VMs using power-aware best-fit decreasing heuristic
Ding et al. [11]	Heuristic algorithms	CPU, memory and bandwidth	Minimizing energy consumption, migration cost and SLA violations by detecting overloaded and underloaded hosts and placing VMs through heuristics
Yadav et al. [45]	Heuristic algorithms	CPU and bandwidth	Detecting overloaded host using gradient-descent based regression and correlation precentage, and bandwidth-aware VM selection
Lin et al. [28]	Heuristic algorithm	CPU	Minimizing power consumption using dynamic Round-Robin and First-Fit based approaches
Mastroianni et al. [31]	Heuristic using Bernoulli trial	CPU and RAM	Consolidation using assignment and migrations based on trial success probabilities
Chen et al. [9]	Heuristic for stochastic bin packing	CPU and memory	Minimizing active servers using first-fit decreasing, best-fit decreasing and history-aware bin packing algorithms
Hsieh et al. [23]	Heuristic algorithm	CPU	Reduce energy cost and number of active hosts by predicting CPU utilization
Dong et al. [12]	Heuristic alogirthm using hierarchical clustering	CPU, memory, storage and network interface	Minimizing energy, migration and link utilization costs using clustering based on minimum-cut
Al-Dulaimy et al. [1]	Heuristic algorithm	CPU, RAM, storage and bandwidth	Minimizing energy consumption using multiple choice Knapsack problem based heuristic
Azizi et al. [2]	Heuristic algorithm	CPU and RAM	Minimizing power consumption and resource usage using multi-dimensional resource usage model
Li et al. [25]	Heuristic algorithm	CPU and RAM	Minimizing energy consumption using multi-dimensional space partitioning

assignment problem is then solved using linear programming relaxation based heuristic. Their approach performs data preprocessing to characterize data patterns that are subsequently used for deriving estimators and minimizing the number of parameters required for optimization. Similarly, Dupont et al. [13] propose a framework for consolidation through energy-aware allocation of virtual machines. The constraint programming is used to find solution of the VM placement problem while considering the idle power and power consumption required for each virtual machine usage. The approach suggested by Zhang et al. [47] uses constraint programming for virtual resource allocation. The problem of resource allocation is represented as a set of variables and constraints. For provisioning and placement of virtual machines, the existing resources and workload requirements are considered for evaluating a satisfaction level which is used for optimization through constraint programming. The linear programming and other related approaches are however considered inefficient as the number of variables increases.

Several approximation and heuristic algorithms have been developed to perform VM consolidation. Beloglazov and Buyya [5] suggest several algorithms for dynamic VM consolidation aiming at minimizing energy consumption and SLA violations. The overloaded hosts are found using statistical measures of median of absolute values, interquartile range, local regression, and local robust regression. The host with minimum resource utilization is tagged as underloaded host. For migration, several policies are proposed to select virtual machines randomly, having the minimum migration time or the maximum correlation. The VM placement approach uses power-aware best-fit decreasing (PABFD) algorithm that allocates host requiring minimum power consumption. Ding et al. [11] propose host overload detection for VM consolidation using available computational capacity of host with the highest

performance to power ratio (PPR). For live migration, the virtual machines requiring minimum data transfer are selected to be transferred to other hosts. Similarly, the under-utilized hosts are detected using Z-score values based on the CPU utilization and PPR value. The VM placement strategy considers allocation of VMs to hosts with higher PPR values. Another similar approach of VM consolidation by Yadav et al. [45] uses resource utilization based heuristic for detecting over-utilized hosts and selects VMs from these hosts that minimize VM migration time. The dynamic round-robin algorithm proposed by Lin et al. [28] restrains servers from accepting more VMs if any of the VMs has completed its job and allows to shutdown the machines after migration. Their hybrid algorithm uses first-fit during rush hours while resorting to round-robin for consolidation during non-rush hours.

Mastroianni et al. [31] use Bernoulli trial whose success probability depends upon actual resource utilization and the utilization threshold. A low probability indicates the host to be over-utilized or under-utilized. The approach uses threshold values to be set by data center administrators for gradual migration of virtual machines. Chen et al. [9], in contrast, use the concept of effective sizing of virtual machines incorporated for server consolidation. The effective sizing estimates aggregate resource requirements by considering VM resource requirements and its correlation with resource requirements of other VMs. Similarly, for resource allocation and minimizing the number of physical machines, other bin-packing based algorithms have also been devised [39].

Markov Chain based models have also been incorporated for different phases of VM consolidation. Hsieh et al. [23] propose approaches for host overload and host underload detection using Gray-Markov based forecasting model. The Markov chain is used to determine error in forecasting. The overload and underload host detection algorithms use CPU utilization history for prediction and make decision based on specified thresholds. Similarly, another approach in [4] uses Markov Chain model for optimizing delay between VM migration while using workloads whose state transition delays are exponentially distributed.

Several strategies transform consolidation steps to other problems with known efficient solutions. Dong et al. [12] present an approach that places VMs on physical machines considering physical resources and VM requirements while using a hierarchical clustering algorithm based on min-cut for traffic between VMs. Ghobaei-Arani et al. [20] use algorithm based on best-fit decreasing to reduce energy consumption and minimize SLA violations in cloud data centers. A learning automata based actions are used for selection of hosts while considering energy consumption for allocation. Al-Dulaimy et al. [1] use the static and dynamic thresholding for resource in order to find underloaded and overloaded hosts, while the VM placement is mapped as the multiple choice knapsack problem to cope with multiple resource constraints.

A dimension-aware approach by Azizi et al. [2] optimizes resource wastage and power consumption for allocating virtual machines to physical machines. Their algorithm uses multi-dimensional model for resource usage to categorize usage states into domains. The host capacity and resource usage factor are used for allocating virtual machines and replacing them for balancing the utilization of resources. The strategy by Li et al. [25] also uses a multi-dimensional space model to represent resource utilization for VM placement. The distances of usage states are categorized into pre-defined domains that are subsequently used to determine resource leakage and priorities for resource allocation. Zhang and Ansari [48] have however shown the approach with clustering of dominant resources to perform similar to dimension-aware approaches with low complexity.

The evolutionary algorithms that use population based meta-heuristic optimization have been widely used despite their complexity and requirements for tuning of parameters. A comparison of meta-heuristic approaches for VM consolidation is shown in Table 2. Li et al. [26] suggest algorithms simulating artificial bee colony foraging behavior to address the issues of energy consumption and quality of service. The VM consolidation is addressed through optimization of multiple objectives including the number of migrations, energy consumption, and host overload probability while considering the CPU, RAM, and bandwidth resources. Mi et al. [32] propose another mechanism for consolidation through dynamic reconfiguration of virtual machines aimed at improving resource utilization and minimizing power consumption. The approach uses genetic algorithm with selection, crossover, and mutation operations being applied to the population containing solutions. The fitness function uses computation of power consumption and CPU usage to improve overall utilization while conserving energy.

Wu et al. [44] use genetic algorithm based approach for dynamic consolidation while optimizing migration cost and power saving. Their genetic algorithm uses swapping and the best-fit heuristic based operations while optimizing fitness function that is based on migration cost and power saving. The approach by Ye et al. [46] also incorporates an energy efficient evolutionary algorithm for optimizing allocation of VMs to physical machines. Other approaches given in [27, 33] propose genetic algorithms where the population contains possible mappings between VMs and physical machines. Their objective functions attempt to optimize expected energy consumption of physical machines in addition to other parameters such as SLA

Table 2 A comparative analysis of meta-heuristic based VM consolidation strategies in terms of approach, resources, and objectives

References	Approach	Resources	Objectives
Li et al. [26]	Artificial Bee Colony based algorithm	CPU, memory and bandwidth	Minimizing power consumption, number of migrations and overload probability
Mi et al. [32]	Genetic algorithm	CPU	Maximizing CPU utilization and minimize power consumption
Wu et al. [44]	Genetic algorithm	CPU and memory	Maximizing consolidation score based on migration cost and power savings
Ye et al. [46]	Genetic algorithm	CPU, memory and bandwidth	Minimizing energy consumption and load variance, while maximizing robustness and resource utilization
Mosa and Paton [33]	Genetic algorithm	CPU, memory and bandwidth	Maximimizing profit while minimizing costs related to energy, SLA violations and migration
Tarahomi et al. [41]	Micro-genetic algorithm	CPU and memory	Minimizing power consumption and SLA violations
Torre et al. [42]	FFD and NSGA-II	CPU and memory	Minimizing resource wastage, resource overcommitted ratio and migrations cost
Gharehpasha et al. [19]	Salp Swarm and Sine-Cosine with chaotic functions	CPU, memory, storage and bandwidth	Minimizing power consumption, resource wastage and SLA violations
Gao et al. [18]	Ant Colony Optimization	CPU and memory	Minimizing resource wastage and power consumption using multi-objective ant colony system
Ferdaus et al. [17]	Ant Colony Optimization	CPU, memory and network I/O	Maximizing resource utilization and minimizing power consumption
Shabeera et al. [37]	Ant Colony Optimization	CPU, memory and storage	Optimizing sum of PM distances for VM and data placement
Liu et al. [29]	Ant Colony Optimization	CPU and memory	Minimizing number of active servers for VM placement using Ant Colony System
Farahnakian et al. [16]	Ant Colony Optimization	CPU, memory and network I/O	Optimizing migration plan while minimizing number of active PMS using ant colony system

violations and/or related costs. The approach proposed by Tarahomi et al. [41] uses an agile version of genetic algorithm with reduced steps for allocating virtual machines. Their fitness function uses total power consumption that is based on CPU utilization. In contrast, Torre et al. [42] propose algorithm based on NSGA-II with multiple populations that evolve independently for placement of virtual machines. The initial population is divided into two generations that are later merged to find the best offsprings using the ranks and crowd distance metrics. Similarly, Gao et al. [18] use criteria based on resource wastage and power consumption for minimization. The algorithm takes as input the resource demands and thresholds of resource utilization to generate non-dominated solutions for mapping virtual machines to hosts.

Other population-based meta-heuristics using ant-colony optimization (ACO) have also been used for addressing VM consolidation. Ferdaus et al. [17] use ACO with multi-dimensional vector packing while considering resource wastage and power consumption as main parameters for optimization. Similarly, Shabeera et al. [37] propose to allocate VMs to physical machines that are close to data. The adjacent physical machines having accumulative capacity equal to the resource demand are selected for allocation using ACO. Gharehpasha et al. [19] propose a hybrid approach for improved exploration and exploitation using the Sine-Cosine and Salp Swarm algorithms. These algorithms are combined with chaotic functions to search for solutions while optimizing power consumption, resource wastage and SLA violations. Liu et al. [29] propose ant-colony system based approach for consolidation using CPU and RAM resources. The approach attempts to minimize the number of active servers. A preference value computed from the pheromone between two VMs is used for selection of server for a virtual machine. Farahnakian et al. [16] incorporate categorization based on CPU utilization and energy for input to ACO based algorithm that is used for generating migration plan. These population-based meta-heuristics require a large number of customized parameters for exploring the search space, thereby making them inappropriate for online VM consolidation, as addressed in this paper.

3 Context and problem formulation for dynamic VM consolidation on cloud platforms

Modern data center operators rely heavily on virtualization that enables execution of multiple virtual machines on a physical machine for effective utilization of physical resources. A cloud environment containing a large number of high performance computing equipment suffers from energy inefficiency due to inadequate usage of cloud resources. An analysis of the utilization of high performance servers in cloud environments shows their utilization to rarely approach 100% [5]. On the one hand, the under-utilization of cloud resources decreases revenue of cloud service providers, and on the other hand, the servers despite being in idle state consume significant amount of their peak power [15]. Moreover, the extra power for cooling the servers with high power consumption incurs monetary cost and also deteriorates the green computing environment [36, 38].

The under-utilization of resources is addressed through virtualization that enables to host multiple instances of virtual machines on a physical machine. Consequently, the cost of operations, power consumption and greenhouse effects mitigate significantly. The under-utilized hosts may then be switched to low-power modes (hibernate/sleep) by migrating remaining virtual machines from under-utilized hosts to any other host.

A generic view of the system model used for consolidation of virtual machines is shown in Fig. 1. The virtual machine consolidation techniques aim at finding minimum number of physical hosts for virtual machines and generate migration maps to be used for live migration. The consolidation through excessive migration of virtual machines may however deteriorate performance and response time of applications. Moreover, the Service Level Agreements (SLAs) established between cloud service providers and customers are violated while attempting to minimize energy through VM consolidation.

To cope with the above-mentioned issues, this paper proposes a novel approach for dynamic consolidation of virtual machines. The proposed NVMC approach consolidates VMs by improving live migration while minimizing energy consumption and SLA violations. The energy consumption is minimized by identifying under-utilized hosts and migrating their VMs to other hosts. The resource capacity of hosts and virtual machines is used to find over-utilized hosts and migrate the virtual machines subsequently. For the VMs to be migrated, the target hosts are found while ensuring to meet the required resource constraints. The VM placement phase uses the criteria based on normalized resource parameters of hosts and virtual machines, and a migration map is subsequently generated for live migration.

For dynamic VM consolidation, we assume a cloud environment with m physical hosts, each having q resource parameters whose capacity is represented by P. Each virtual machine is characterized with resource requirements χ, for placement on a host. With n virtual machines to be allocated to a host, the following constraints must be fulfilled:

$$\sum_{i=0}^{n-1} \chi_{ij} \leq P_j \forall j = 0, 1, \ldots, q-1 \tag{1}$$

The execution of user workloads through virtual machines results in power consumption by physical machines which are usually high performance servers. With the resource requirement such as CPU exceeding the capacity of the host, an SLA violation occurs. The physical machines may be over-utilized due to a large number of virtual machines being hosted on the machines. To cope with the increasing resource demand, a few virtual machines need to be migrated from over-utilized hosts. Similarly, the virtual machines from the under-utilized hosts must be migrated to bring those hosts to idle state, thereby making hosts to consume negligible power. Assuming $C^e_{i,k}$, $C^s_{i,k}$ and $C^v_{i,k}$ to be respectively the costs of energy consumption, SLA violations and live VM migrations corresponding to host k at i-th instance of time, the total cost C_{total} becomes:

$$C_{total} = \sum_{i=1}^{t} \left(\sum_{k=1}^{m} C^e_{i,k} + \sum_{k=1}^{m} C^s_{i,k} + \sum_{k=1}^{m} C^v_{i,k} \right), \tag{2}$$

where t is the total time which may be divided into multiple time frames. The dynamic consolidation determines migration maps describing the mapping of virtual machines to hosts for which the total cost C_{total} is minimized.

The SLA violations (SLAV) are measured as a product of the time percentage of a host being active during full utilization of CPU and the CPU utilization incurred due to migration [5, 7], as given below:

$$SLAV = \frac{1}{q} \sum_{i=1}^{m} \frac{T^f_i}{T^a_i} * \frac{1}{n} \sum_{i=1} n \frac{U^m_i}{U^f_i} \tag{3}$$

where, T^f is the time for full CPU utilization, T^a is the time during which the host is active, U^m is the CPU utilization during migration, and U^f is the total CPU utilization requested by a virtual machine.

4 Normalization-based VM consolidation algorithm

In a virtualized environment, the cloud service providers attempt to optimize resource utilization to meet cloud users' demands. Migration of virtual machines from over-utilized and under-utilized hosts helps in meeting SLA constraints and in reducing the energy consumption. A large number of VM migrations may however affect the quality-of-service (QoS) by increasing SLA violations. Through consolidation, the VM migrations need to be leveraged to find a trade-off between performance and energy. The normalization-based VM consolidation (NVMC) algorithm aims at generating migration plan that consolidates virtual machines to physical machines while minimizing energy consumption, number of VM migrations and SLA violations.

Algorithm 1 Normalization-based VM Consolidation (NVMC) Algorithm

1: /* Let $\forall_{i=0}^{m-1}\ H_i$ be the set of m physical hosts and let $\forall_{i=0}^{n-1}\ V_i$ be the set of n virtual machines. Let $\forall_{j=0}^{q-1}\ P_j(h)$ be the set of q resource parameters for each host $h \in H$. Let ϕ represent the set of available hosts. */
2: $\phi \leftarrow H$ // Initialize available hosts
3: **for** $j = 0, 1, \ldots, q-1$ **do**
4: $P_j^{max} \leftarrow$ Double.MIN_VALUE, $P_j^{min} \leftarrow$ Double.MAX_VALUE
5: **for** $i = 0, 1, \ldots, m-1$ **do**
6: **if** $P_j(H_i) > P_j^{max}$ **then**
7: $P_j^{max} \leftarrow P_j(H_i)$
8: **else if** $P_j(H_i) < P_j^{min}$ **then**
9: $P_j^{min} \leftarrow P_j(H_i)$
10: **end if**
11: **end for**
12: **end for**
13: /* Let R reresent a map with host as a key and a resource weight as value. Let W_j, $\forall j = 0, 1, \ldots, q-1$ represent the resource weight values for parameters */
14: **for** $i = 0, 1, \ldots, m-1$ **do**
15: **for** $j = 0, 1, \ldots, q-1$ **do**
16: **if** $P_j^{max} == P_j^{min}$ **then**
17: $W_j(H_i) \leftarrow$ Double.MIN_VALUE
18: **else**
19: $W_j(H_i) \leftarrow \frac{P_j(H_i) - P_j^{min}}{P_j^{max} - P_j^{min}}$
20: **end if**
21: **end for**
22: R.put$\left(H_i, \sum_{j=1}^{q} W_j(H_i)\right)$
23: **end for**
24: $V_o \leftarrow \epsilon$, $V_u \leftarrow \epsilon$
25: **for** $i = 0, 1, \ldots, m-1$ **do**
26: flag $\leftarrow$ false
27: **while** isOverUtilized(H_i) **do**
28: flag $\leftarrow$ true
29: $\Upsilon \leftarrow$ virtual machine to be migrated with minimum migration time
30: $V_o \bigcup \Upsilon$
31: Remove virtual machine Υ from host H_i
32: **end while**
33: **if** (flag == true) **then**
34: $\phi \leftarrow \phi$ - H_i
35: **end if**
36: **end for**
37: $\psi_o \leftarrow$ findVmPlacement (V_o, ϕ)
38: **while** (h=getUnderUtilizedHost(ϕ)) && ($h \neq$ NULL) **do**
39: $\phi \leftarrow \phi$ - h
40: **for** each virtual machine Υ on host h **do**
41: $V_u \leftarrow V_u \bigcup \Upsilon$
42: Remove virtual machines Υ from host h
43: **end for**
44: **end while**
45: $\psi_u \leftarrow$ findVmPlacement (V_u, ϕ)
46: Return $\psi_o \bigcup \psi_u$

The NVMC algorithm (Algorithm 1) takes as input the sets of m hosts (H) and n virtual machines (V). The hosts are equipped with a set of q resource parameters (P) whose utilization is demanded by virtual machines executing user applications. Initially, the NVMC algorithm computes resource weights through normalization that scales resource parameters for both physical hosts and virtual machines. The scaled values are subsequently used in detecting over-utilized hosts while comparing with a threshold value, as given in Algorithm 2.

The set of available hosts ϕ is initiaized at step 2. The maximum and minimum values of resource parameters are determined corresponding to each host using steps 3–12. For each host, the resource weights are computed and placed in a map R, using loops in steps 14–23. The resource weights represent normalized values that are scaled to exist in interval [0,1]. The map R is filled with the host as key and the sum of its resource weights as the value. The sets of virtual machines executing on over-utilized and under-utilized hosts are represented by V_o and V_u, respectively. The *isOverUtilized* algorithm (Algorithm 2) is invoked at step 27 to check whether the input host is over-utilized in terms of resources. For each over-utilized host, the virtual machines are selected, added to the set V_o, and then removed from the host until the host is no longer over-utilized. The set of available hosts ϕ is subsequently updated, at step 34. The *findVMPlacement* algorithm (Algorithm 4) is then invoked to determine appropriate hosts for the virtual machines in the set V_o. The algorithm returns the migration map ψ_o, containing virtual machines and their corresponding hosts. In the next step, the under-utilized hosts are found by using the function *getUnderUtilizedHost* (Algorithm 4). All the virtual machines from the underutilized hosts are then added to the set V_u in steps 40–43. The algorithm *findVMPlacement* is once again invoked to determine the migration map ψ_u, representing mapping of virtual machines to hosts. The maps ψ_o and ψ_u are then merged to generate final migration map that is returned by the NVMC algorithm.

Algorithm 2 *isOverUtilized* Algorithm

Input: Host h
Output: Value true or false
1: // Let $\forall_{j=0}^{q-1}\chi_j(v)$ be the resources requested by a virtual machine v and let T be the threshold value to be used for detecting over-utilized host.
2: **for** each virtual machine v running on host h **do**
3: $\chi_j^{max}(v) \leftarrow$ Double.MIN_VALUE, $\forall j = 0, 1, \ldots, q-1$
4: $\chi_j^{min}(v) \leftarrow$ Double.MAX_VALUE, $\forall j = 0, 1, \ldots, q-1$
5: **for** $j = 0, 1, \ldots, q-1$ **do**
6: **if** $\chi_j(v) > \chi_j^{max}(v)$ **then**
7: $\chi_j^{max}(v) \leftarrow \chi_j(v)$
8: **end if**
9: **if** $\chi_j(v) < \chi_j^{min}(v)$ **then**
10: $\chi_j^{min}(v) \leftarrow \chi_j(v)$
11: **end if**
12: **end for**
13: **end for**
14: /* Let S represent the sum of normalized values for each resource requested by the virtual machines on the host h */
15: $S \leftarrow 0$
16: **for** each virtual machine v running on host h **do**
17: **for** $j = 0, 1, \ldots, q-1$ **do**
18: **if** $\chi_j^{max}(v) == \chi_j^{min}(v)$ **then**
19: $W_j(v) \leftarrow 0$
20: **else**
21: $W_j(v) \leftarrow \frac{\chi_j(v) - \chi_j^{min}(v)}{\chi_j^{max}(v) - \chi_j^{min}(v)}$
22: **end if**
23: $S \leftarrow S + W_j(v)$
24: **end for**
25: **end for**
26: $W_h \leftarrow R$.get(h) // Retrieve resource weight from the map R
27: **R**eturn $\frac{S - W_h}{q} \geq T$

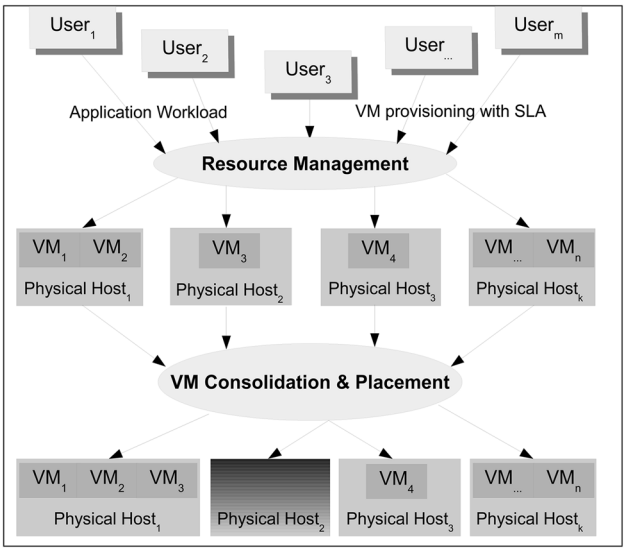

Fig. 1 Overview of VM consolidation on cloud platforms

The *isOverUtilized* algorithm (Algorithm 2) takes as input a host h and returns whether a host is over-utilized, thereby requiring the virtual machines to be migrated to other hosts. The maximum and minimum values for each resource requested by the virtual machine are initialized at steps 3–4. Subsequently, these values are updated in an iterative manner (steps 5–12) to find values for all virtual machines running on the host h. The values for requested resources are normalized to exist in the interval [0,1] using steps 18–21. The sum of the normalized resource weights is accumulated for all the virtual machines. The step 26 retrieves the resource weight corresponding to the host h from the map, as stored by the *NVMC* algorithm. The *isOverUtilized* algorithm then computes the difference of the normalized resource weights of the host and its virtual machines, corresponding to a single resource. It then returns true or false depending upon the computed value exceeding the threshold value T.

Algorithm 3 *findVmPlacement* Algorithm

Input: Set of virtual machines V to be migrated, Set of available hosts ϕ
Output: Map (ψ) containing hosts and virtual machines
1: Sort virtual machines V w.r.t. CPU utilization in descending order
2: /* Let $A_k(h)$ represent value of available resource k on host $h \in \phi$, and let $\chi_k(v)$ represent the value of resource k demanded by virtual machine $v \in V$. $P_k(h)$ represents the capacity of resource k on host h, while μ represents CDTR values */
3: $\beta \leftarrow$ null // host to be allocated
4: **for** each virtual machine $v \in V$ **do**
5: $\mu_{min} \leftarrow$ Double.MAX_VALUE
6: **for** $j = 0, 1, 2, \ldots, |\phi| - 1$ **do**
7: flag $\leftarrow$ true
8: **for** $k = 0, 1, 2, \ldots, q - 1$ **do**
9: **if** $A_k(\phi_j) + \chi_k(v) > P_k(\phi_j)$ **then**
10: flag $\leftarrow$ false
11: break
12: **end if**
13: **end for**
14: **if** (flag == true) **then**
15: $\mu(\phi_j, v) \leftarrow \sum_{k=0}^{q-1} \frac{\chi_k(v)}{P_k(\phi_j)}$
16: **if** $\mu(\phi_j, v) < \mu_{min}$ **then**
17: $\beta \leftarrow \phi_j$
18: $\mu_{min} \leftarrow \mu(\phi_j, v)$
19: **end if**
20: **end if**
21: **end for**
22: ψ.put(v,β)
23: Update available resources $A_j(\beta)$, $\forall j = 0, 1, \ldots, q - 1$
24: **end for**
25: Return ψ

The *findVmPlacement* algorithm (Algorithm 3) takes as input the set of virtual machines and the set of available hosts to generate a map ψ, containing virtual machines and the hosts on which these virtual machines should be hosted. The virtual machines are initially sorted w.r.t their CPU utilization. The host to be allocated to a virtual machine is initialized at step 3. The loop at steps 4–24 finds host for each virtual machine and subsequently adds both the virtual machine and host to the map ψ. The algorithm uses the notion of cumulative demand-to-total ratio (CDTR), represented as μ. Corresponding to a host h and a virtual machine v, it is computed as follows:

$$\mu(h,v) = \sum_{k=0}^{q-1} \frac{\chi_k(v)}{P_k(h)}, \tag{4}$$

where $\chi(v)$ represents the set of q resources requested by the virtual machine v, and $P(h)$ represents the capacity of resources of host h. The minimum CDTR value μ_{min} is initialized at step 5. The suitability of host for a virtual machine is found at steps 7–13, using resources available on host ϕ_j, the resources χ requested by the virtual machine, and the total resources P existing on the host. The CDTR value for the host ϕ_j, represented as $\mu(\phi_j)$, is computed at step 15. The host with the minimum CDTR value, β, is then found and subsequently added to the map ψ along with the virtual machine v. The map ψ containing mappings of virtual machines to hosts is then returned by the algorithm.

$$\lambda(h) = \sum_{k=0}^{q-1} \frac{A_k(h)}{P_k(h)}, \tag{5}$$

where A represents the values for available resources and $P(h)$ represents the capacity of resources existing on host h.

The minimum value of CATR, represented as λ_{min} and the under-utilized host are initialized at step 1 and 2, respectively. For each available host, the CATR values are determined at step 4. Subsequently, the minimum CATR value is found using steps 5–8. The algorithm then returns the host with the minimum CATR value.

Considering q resource parameters, m hosts and n virtual machines, the loops at steps 3–12 and 14–23 have time complexity of $O(q * m)$. The loop at steps 25–36 and 38–44 for finding over-utilized and under-utililized hosts, both work with the complexity of $O(m * n * q)$. The invocations of VM placement algorithms also incur a cost of $O(n * m * q)$. The overall complexity of the algorithm therefore becomes $O(m * n * q)$. Limiting the resource parameters of the cloud environment to be small constants, i.e. $q << n$ and $q << m$, the complexity of the proposed algorithm reduces to $O(n * m)$.

Moreover, the steps 2–23 (highlighted) in the NVMC algorithm are performed offline, prior to dynamic online consolidation, thereby further enhancing the overall efficiency of the proposed algorithm.

Algorithm 4 *getUnderUtilizedHost* Algorithm

Input: Set of available hosts ϕ
Output: The under-utilized host β
1: $\lambda_{min} \leftarrow$ Double.MAX_VALUE
2: $\beta \leftarrow$ null
3: **for** $j = 0, 1, 2, \ldots, |\phi| - 1$ **do**
4: $\quad \lambda(\phi_j) \leftarrow \sum_{k=0}^{q-1} \frac{A_k(\phi_j)}{P_k(\phi_j)}$
5: $\quad$ **if** $\lambda(\phi_j) < \lambda_{min}$ **then**
6: $\quad\quad \lambda_{min} \leftarrow \lambda(\phi_j)$
7: $\quad\quad \beta \leftarrow \phi_j$
8: $\quad$ **end if**
9: **end for**
10: **return** β

The *getUnderUtilizedHost* algorithm (Algorithm 4) takes as input the set of available hosts ϕ and returns the under-utilized host β. The algorithm uses the notion of cumulative available-to-total ratio (CATR), represented as λ. Corresponding to a host h, having q number of resource parameters, it is computed as follows:

5 Experimentation setup and results

For experimentation, the proposed NVMC approach has been implemented in the CloudSim framework [7]. Our experiments have been performed with a PowerDatacenter containing 800 hosts. The NVMC approach considers CPU, memory, bandwidth, and storage resources while

Table 3 Experimental setup and configuration parameters

Experimentation platform			
Framework	Simulation platform		Operating system
CloudSim-4.0 Framework [7]	Intel Core i7-720q		64-bit Windows
Hosts configuration			
Power models		Host RAM	Number of cores
HP Proliant-ML110-G4 with Intel Xeon 3040 & HP Proliant-ML110-G5 with Intel Xeon 3075		4 GB	02
Core frequency (MHz)	Host BW		Host storage
1860, 2660 MHz	1 Gbps		1 TB
Virtual machines configuration			
VM CPU (MIPS)	Number of cores		RAM (MB)
2500, 2000, 1000, 500	01		870, 1740, 1740, 613
VM bandwidth	VM size (MB)		Hypervisor
100 Mbps	2500		Xen

placing virtual machines on hosts. The user requests are set to execute on virtual machines with the workloads being characterized with resource requirements from the real system traces obtained from the CoMon [35] project, a monitoring system for the PlanetLab testbed [10]. The configurations used for experimentation containing hosts and virtual machine parameters are given in Table 3.

For virtual machines, the Amazon EC2 instances are used that are categorized into micro (500 MIPS), small (1000 MIPS), extra large (2000 MIPS) and high-cpu (2500 MIPS). The performance results of the proposed NVMC algorithm have been compared with the dynamic energy-aware consolidation (EAC) implemented in the CloudSim framework [5, 7] while using interquartile range (IQR), Median Absolute Deviation (MAD) and Local Regression Robust (LRR) strategies for host overload detection. The online deterministic algorithms for dynamic VM consolidation implemented in the CloudSim framework are shown to achieve significant competitiveness[1] over optimal offline algorithms.

The IQR strategy uses statistical measure of dispersion as the difference of quartiles for setting upper threshold of utilization for finding over-utilized hosts. The MAD strategy uses a customized threshold for CPU utilization to detect over-utilized hosts. It controls the frequency of VM consolidation to impact the number of VM migrations and SLA violations. The LRR strategy attempts to fit a polynomial to the data of CPU utilization in order to estimate the future utilization. It uses the robust estimation method with weights that are computed iteratively to fit data. The safety parameter and estimated utilization are then used to detect overloaded hosts and perform VM migrations subsequently.

The strategies used for comparison, their parameters, the workloads and the number of virtual machines used for evaluation are given in Table 4. The PlanetLab workloads (W1–W6) contain date-wise traces of execution (representing utilization of resources at several instances) on cloud platform, as obtained for different days of March and April (2011). These workloads, set to work with diverse configurations of virtual machines, are widely used for evaluating the performance of dynamic VM consolidation strategies.

5.1 Metrics for performance evaluation

For evaluation, we use the metrics of energy consumption, SLA violations (SLAV), number of VM migrations, Cost Impact Factor (CIF) and Overall Performance Enhancement (OPE). The first metric represents the energy consumed by physical machines of the data center while executing user workloads. The SLA violation results are computed as the product of the SLA time per active host and performance degradation due to migration, as given in

[1] With upper bound for competitive-ratio being $1 + (m * c/(2 * (m + 1)))$, where m is the maximum number of virtual machines that may be allocated to a host demanding maximum CPU capacity, and c is the cost of SLA violations.

Table 4 Strategies, their parameters and workloads used for evaluation

Evaluated workloads and strategies with parameters					
Workloads and number of VMs					
W1		W2		W3	
2011/03/03	1052	2011/03/06	898	2011/03/22	1516
W4		W5		W6	
2011/04/03	1463	2011/04/09	1358	2011/04/20	1033
Strategies and parameters					
EAC_IQR = > 1.0, 2.0, 3.0		EAC_MAD = > 1.0, 2.0, 3.0		EAC_LR = > 1.0, 1.2, 1.4	
EAC_THR = > 0.6, 0.8, 1.0		NVMC = > 3.0, 3.5, 4.0			

Eq. 3. The third metric represents the number of VM migrations performed from one physical machine to other during consolidation.

Let E, S and V represent the energy consumption, SLA violations, and number of VM migrations, respectively. The normalized values of energy consumption, SLA violations and number of VM migrations, represented respectively, by E_j^N and S_j^N, and V_j^N corresponding to j-th consolidation strategy are computed as:

$$E_j^N = 1 + \frac{(E_j - E_{min})(k-1)}{E_{max} - E_{min}}, \tag{6}$$

$$S_j^N = 1 + \frac{(S_j - S_{min})(k-1)}{S_{max} - S_{min}}, \tag{7}$$

$$V_j^N = 1 + \frac{(V_j - V_{min})(k-1)}{V_{max} - V_{min}}, \tag{8}$$

where k is the total number of strategies considered for evaluation. The Cost Impact Factor (CIF) representing the combined incurred costs for a strategy may be computed as:

$$CIF_j = E_j^N + S_j^N + V_j^N, \quad \forall j = 1, 2, \ldots, k. \tag{9}$$

The Overall Performance Enhancement (OPE) achieved by the NVMC strategy over any other j-th strategy is described as the ratio of CIF values, and is computed as:

$$OPE_{NVMC} = \frac{CIF_j}{CIF_{NVMC}}. \tag{10}$$

5.2 Performance results

For energy consumption using workloads W1, W2 and W3, the performance results are shown in Fig. 2. The NVMC strategy significantly minimizes energy consumption. Using NVMC_4.0, the minimum energy 91.58 kWh, 68.6 kWh and 94.20 kWh is consumed for workloads W1, W2 and W3, respectively. The performance of NVMC variants is followed by the EAC_LRR_1.0 consolidation strategy. In terms of energy consumption, the NVMC_4.0 variant performs 1.42, 1.43 and 1.54 times better than the EAC_LRR_1.0 strategy.

The number of VM migrations performed through consolidation for workloads W1, W2 & W3 are shown in Fig. 3. The dynamic consolidation using NVMC results in minimum number of VM migrations. The performance of NVMC variants is followed by the performance of EAC_MAD_3.0 for workload W1. Similarly, for the workloads W2 and W3, the EAC_LRR_1.0 and EAC_IQR_1.0 strategies perform better than other energy aware consolidation strategies. Overall, the NVMC_4.0 variant outperforms all other variants. It performs 10.33, 8.84 and 6.95 times better than the best performing strategies EAC_MAD_3.0, EAC_LRR_1.0 and EAC_IQR_1.0 for workloads W1, W2 and W3, respectively.

The percentage of SLA violations incurred during execution of workloads W1, W2 and W3 is shown in Fig. 4. As shown in the figure, the NVMC_4.0 strategy outperforms other strategies by producing lowest number of SLA violations. For the workload W1, the performance of NVMC_4.0 variant is followed by NVMC_3.5 and EAC_IQR_2.0. Similarly, for the workload W2, the performance of NVMC variants is followed by the performance of EAC_MAD_3.0. For the workload W3, however, the NVMC_4.0 performs better than all other strategies, and is followed by the performance of EAC_MAD_3.0 and EAC_IQR_2.0.

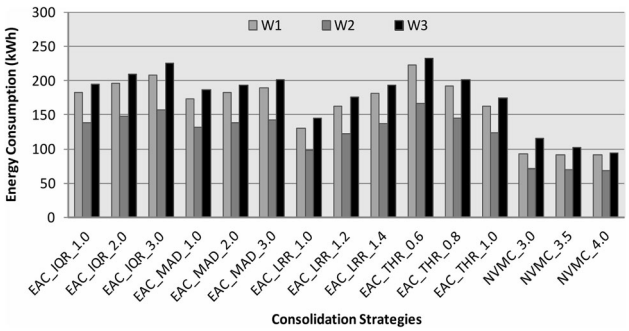

Fig. 2 Energy consumption (kWh) by physical machines for workloads W1, W2, & W3 corresponding to different VM consolidation strategies

Overall, for the workloads W1, W2 and W3, the results for the energy consumption, number of VM migrations and SLA violations show that the NVMC strategy outperforms other strategies by significantly optimizing all the objectives.

Using workloads W4, W5 and W6, the results for energy consumption are shown in Fig. 5. The results show that the NVMC strategy significantly minimizes energy consumption. Using NVMC_4.0, the minimum energy 118.81 kWh, 98.88 kWh and 68.24 kWh is consumed for workloads W4, W5 and W6, respectively. The NVMC variants outperform other strategies in terms of energy consumption. The performance of NVMC variants is followed by EAC_LRR_1.0 strategy. In terms of energy consumption, the NVMC_4.0 variant performs 1.49, 1.46 and 1.61 times better than the EAC_LRR_1.0 strategy.

Figure 6 shows the number of VM migrations performed during consolidation. The NVMC strategy outperforms other strategies by producing minimum number of VM migrations. The performance of NVMC variants is followed by the performance of EAC_MAD_3.0 for workloads W4 and W5, and EAC_IQR_1.0 for workload W6. Overall, the NVMC_4.0 variant outperforms all other variants. It performs 10.17, 9.22 and 7.36 times better than the variants performing best among other strategies for workloads W4, W5 and W6, respectively.

The results of percentage of SLA violations incurred during execution of workloads W4, W5 and W6 are shown in Fig. 7. As shown in the figure, the NVMC_4.0 outperforms other strategies by producing lowest number of SLA violations. The performance of NVMC_4.0 strategy is followed by EAC_THR_0.8, EAC_MAD_3.0 and EAC_IQR_1.0 for the workloads W4, W5 and W6, respectively.

It is evident that the best performing energy aware strategies do not perform consistently in optimizing all the objectives. In contrast, the NVMC strategy is able to minimize the energy consumption, number of VM migrations and SLA violations, simultaneously.

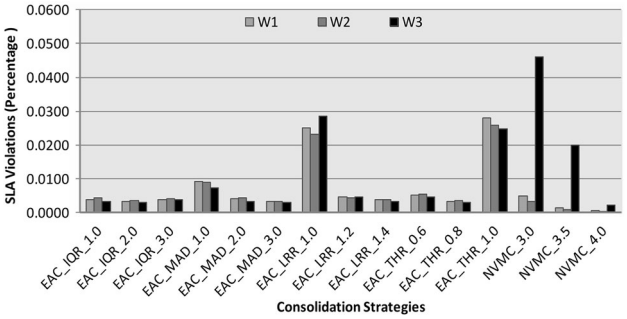

Fig. 4 Percentage of SLA violations incurred during execution of workloads W1, W2, & W3 corresponding to VM consolidation strategies

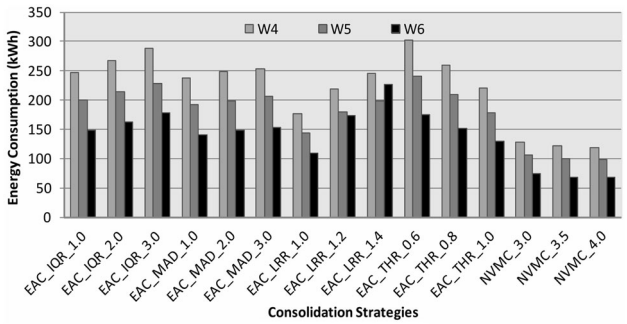

Fig. 5 Energy consumption (kWh) by physical machines for workloads W4, W5, & W6 corresponding to different VM consolidation strategies

Figure 8 shows the Cost Impact Factor (CIF) results, as computed by Eq. 9, for the VM consolidation strategies. It represents the collective cost for comparative evaluation while considering all three objectives of minimizing energy consumption, number of VM migrations and SLA violations. It is evident that the NVMC strategy performs better than all other strategies, while the NVMC_4.0 produces the lowest value of CIF. Among other energy-aware consolidation strategies, the EAC_LRR_1.2, EAC_IQR_1.0 and EAC_MAD_2.0 perform better than other strategies.

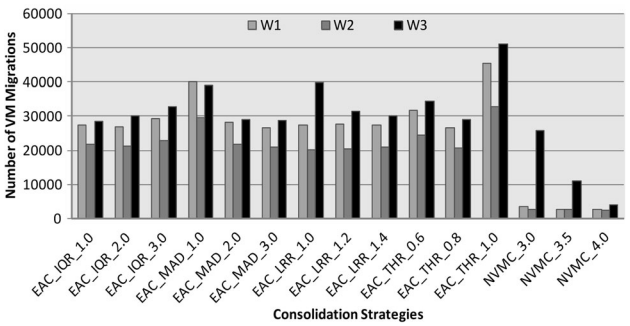

Fig. 3 Number of VM migrations performed through VM consolidation strategies using workloads W1, W2, & W3

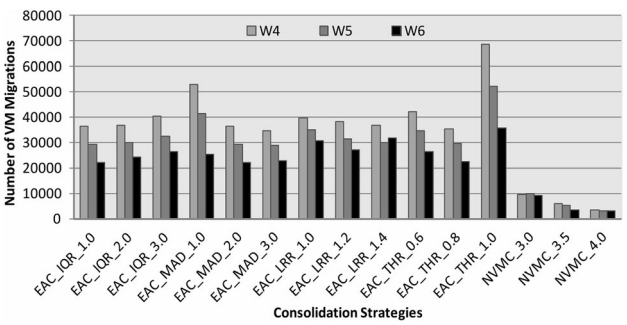

Fig. 6 Number of VM migrations performed through VM consolidation strategies using workloads W4, W5, & W6

5.3 Analysis of results

The cumulative values of energy consumption, number of VM migrations and SLA violations for all workloads corresponding to threshold values are shown in Table 5. The cumulative results show that for energy consumption, the EAC_IQR_1.0, EAC_MAD_1.0, EAC_LRR_1.0, EAC_THR_1.0 and NVMC_4.0 variants perform better than variants with other threshold values. Similarly, for the number of VM migrations, the EAC_IQR_1.0, EAC_MAD_3.0, EAC_LRR_1.2, EAC_THR_0.8 and NVMC_4.0 variants produce better results. For the SLA violations, the EAC_IQR_2.0, EAC_MAD_3.0, EAC_LRR_1.4, EAC_THR_0.8 and NVMC_4.0 show better results than other variants.

The Overall Performance Enhancement (OPE) values as computed using Eq. 10 obtained by the NVMC_3.0 variant are shown in Fig. 9. The OPE values correspond to the performance enhanced in terms of all three objectives of energy consumption, number of VM migrations, and SLA violations. As shown in the figure, the NVMC_3.0 performs better than all other energy-aware strategies. On average, it performs 1.63 times better than all other strategies.

Figure 10 shows the OPE values obtained by the NVMC_3.5 variant over other energy-aware variants. As shown in the figure, the NVMC_3.5 performs better than all other energy-aware strategies by producing average OPE value of 2.70.

The OPE values obtained by the NVMC_4.0 variant are shown in Fig. 11. The NVMC_4.0 variant outperforms all energy-aware consolidations strategies. Moreover, it performs better than other NVMC variants as well. On average, it performs 3.61 times better than the energy-aware consolidation strategies.

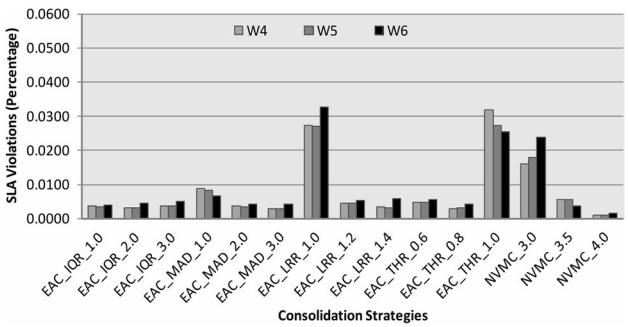

Fig. 7 Percentage of SLA violations incurred during execution of workloads W4, W5, & W6 corresponding to VM consolidation strategies

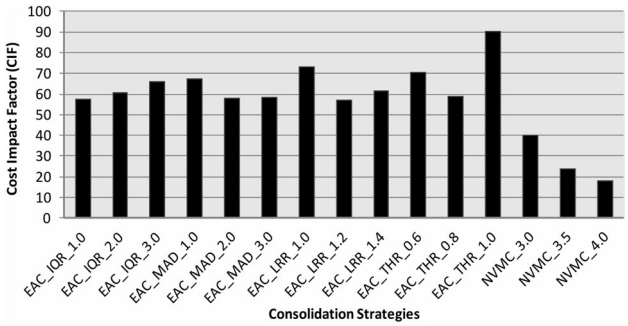

Fig. 8 Cost impact factor (CIF) computed using Eq. 9, corresponding to the VM consolidation strategies

Table 5 Performance analysis in terms of cumulative values of energy consumption, number of VM migrations, and SLA violations for all workloads

Cumulative performance results					
	EAC_IQR 1.0	EAC_MAD 1.0	EAC_LRR 1.0	EAC_THR 0.6	NVMC 3.0
E	1110.23	1064.63	803.72	1341.59	589.79
V	165,689	227,889	192,793	193,836	60,976
S	0.02259	0.04955	0.16404	0.03034	0.11227
	2.0	2.0	1.2	0.8	3.5
E	1197.97	1108.39	1036.01	1158.62	554.37
V	169,224	166,984	175,879	163,509	31,113
S	0.02065	0.02295	0.02798	0.02015	0.03767
	3.0	3.0	1.4	1.0	4.0
E	1286.98	1148.62	1182.42	989.27	540.31
V	184,027	162,636	176,911	286,175	18,473
S	0.02424	0.01955	0.02322	0.1635	0.00685

6 Conclusion

For cloud service providers, the energy efficiency and quality-of-service (QoS) have become pivotal in order to reduce energy costs, support green computing environments, and meet user requirements as per Service Level Agreement (SLA). The VM consolidation techniques aim at improving energy efficiency by finding a trade-off between energy and performance. These techniques consolidate VMs to minimum number of hosts to save energy while maintaining SLA, through migration of VMs from over-utilized and under-utilized hosts. The complexity of finding optimal placement for virtual machines makes heuristic-based approaches more appropriate for dynamic VM consolidation. In this paper, we propose a normalization-based VM consolidation (NVMC) approach that aims

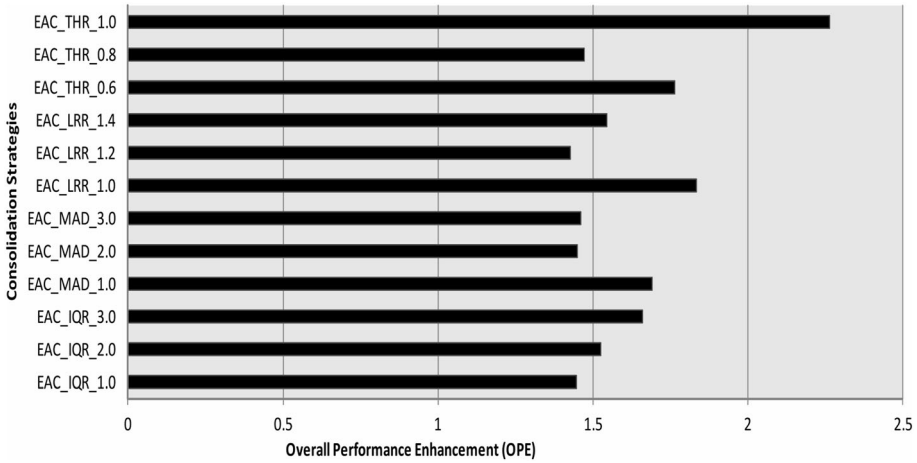

Fig. 9 Overall performance enhancement (OPE) obtained by NVMC_3.0 over other approaches

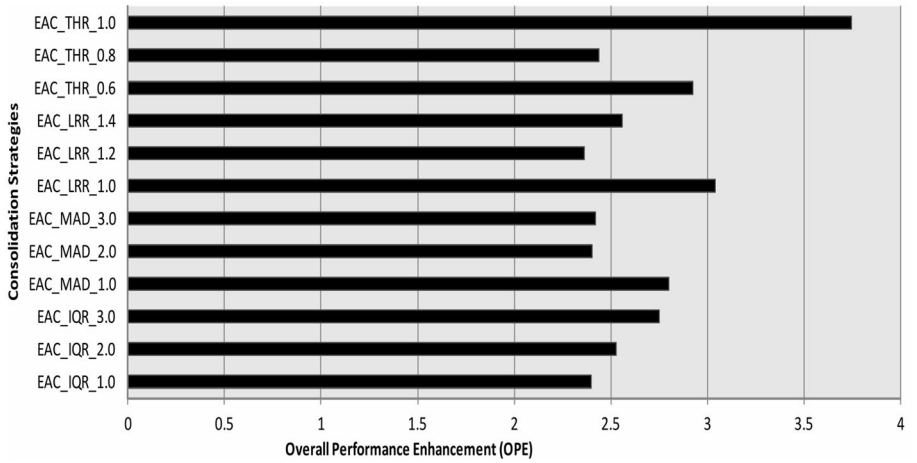

Fig. 10 Overall performance enhancement (OPE) obtained by NVMC_3.5 over other approaches

at minimizing energy consumption, number of VM migrations and SLA violations during execution of user applications. Our approach uses normalized values of resource parameters for consolidation. The over-utilized hosts are found by considering criteria based on virtual machines and physical machines resources. The approach uses the notions of cumulative available-to-total ratio (CATR) and demand-to-total ratio (CDTR) for finding under-utilized hosts and subsequent placement of virtual machines.

We have performed large scale experimentation using real world data obtained from execution traces of

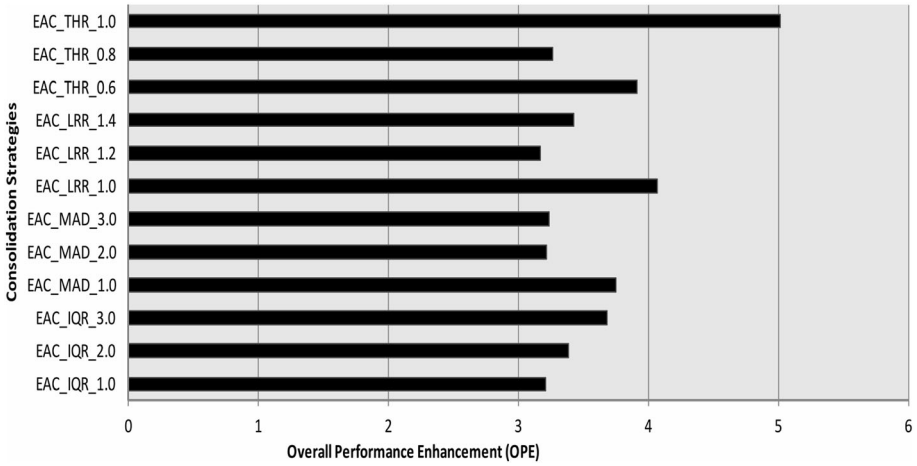

Fig. 11 Overall performance enhancement (OPE) obtained by NVMC_4.0 over other approaches

PlanetLab virtual machines. The experimentation results show that the proposed NVMC approach outperforms other VM consolidation strategies. It significantly optimizes the energy consumption, number of VM migrations and SLA violations. The NVMC approach is able to achieve 1.61, 10.33 and 6.82 times improvement over other energy-aware strategies with best performance corresponding to energy consumption, number of VM migrations, and SLA violations, respectively. Moreover, the NVMC variants NVMC_3.0, NVMC_3.5 and NVMC_4.0 obtain average overall performance enhancement (OPE) values of 1.63, 2.70 and 3.61, respectively.

The NVMC approach currently uses threshold value for detection of over-utilized hosts. We intend to extend the approach to dynamically adapt the threshold value through estimation based on regression analysis. By maintaining performance history in terms of energy and SLA violations, the runtime adaptation of threshold value would allow to further enhance the performance of consolidation. It would however require to consider a trade-off between performance and estimation overhead.

Data availability The datasets used for experimentation in this research work are available with the well-known CloudSim simulator that may be accessed from the repository: https://github.com/Cloudslab/cloudsim/releases.

References

1. Al-Dulaimy, A., Itani, W., Zantout, R., Zekri, A.: Type-aware virtual machine management for energy efficient cloud data centers. Sustain. Comput. Inform. Syst. **19**, 185–203 (2018) https://doi.org/10.1016/j.suscom.2018.05.012. http://www.sciencedirect.com/science/article/pii/S2210537917304249
2. Azizi, S., Li, D., et al.: An energy-efficient algorithm for virtual machine placement optimization in cloud data centers. Clust. Comput. **23**, 3421–3434 (2020). https://doi.org/10.1007/s10586-020-03096-0
3. Belkhir, L., Elmeligi, A.: Assessing ict global emissions footprint: trends to 2040 & recommendations. J. Clean. Prod. **177**, 448–463 (2018)
4. Beloglazov, A., Buyya, R.: Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints. IEEE Trans. Parallel Distrib. Syst. **24**(7), 1366–1379 (2012)
5. Beloglazov, A., Buyya, R.: 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 (2012)
6. Buyya, R., Broberg, J., Goscinski, A.M.: Cloud Computing Principles and Paradigms. Wiley Publishing, New York (2011)
7. Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A.F., Buyya, R.: Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw. Pract. Exp. **41**(1), 23–50 (2011). https://doi.org/10.1002/spe.995
8. Calheiros, R.N., Toosi, A.N., Vecchiola, C., Buyya, R.: A coordinator for scaling elastic applications across multiple clouds. Future Gener. Comput. Syst. **28**(8), 1350–1362 (2012). https://doi.org/10.1016/j.future.2012.03.010. http://www.sciencedirect.com/science/article/pii/S0167739X12000635. Including Special sections SS: Trusting Software Behavior and SS: Economics of Computing Services
9. Chen, M., Zhang, H., Su, Y., Wang, X., Jiang, G., Yoshihira, K.: Effective vm sizing in virtualized data centers. In: 12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops, pp. 594–601 (2011). https://doi.org/10.1109/INM.2011.5990564
10. Chun, B., Culler, D., Roscoe, T., Bavier, A., Peterson, L., Wawrzoniak, M., Bowman, M.: Planetlab: an overlay testbed for broad-coverage services. SIGCOMM Comput. Commun. Rev. **33**(3), 3–12 (2003). https://doi.org/10.1145/956993.956995
11. Ding, W., Luo, F., Han, L., Gu, C., Lu, H., Fuentes, J.: Adaptive virtual machine consolidation framework based on performance-to-power ratio in cloud data centers. Future Gener. Comput. Syst. **111**, 254–270 (2020). https://doi.org/10.1016/j.future.2020.05.004. http://www.sciencedirect.com/science/article/pii/S0167739X19307769
12. Dong, J., Wang, H., Cheng, S.: Energy-performance tradeoffs in iaas cloud with virtual machine scheduling. China Commun. **12**(2), 155–166 (2015). https://doi.org/10.1109/CC.2015.7084410
13. Dupont, C., Schulze, T., Giuliani, G., Somov, A., Hermenier, F.: An energy aware framework for virtual machine placement in cloud federated data centres. In: 2012 Third International Conference on Future Systems: Where Energy, Computing and Communication Meet (e-Energy), pp. 1–10 (2012). https://doi.org/10.1145/2208828.2208832
14. Duy, T.V.T., Sato, Y., Inoguchi, Y.: Performance evaluation of a green scheduling algorithm for energy savings in cloud computing. In: 2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), pp. 1–8. IEEE (2010)
15. Fan, X., Weber, W.D., Barroso, L.A.: Power provisioning for a warehouse-sized computer. ACM SIGARCH Comput. Archit. News **35**(2), 13–23 (2007)
16. Farahnakian, F., Ashraf, A., Pahikkala, T., Liljeberg, P., Plosila, J., Porres, I., Tenhunen, H.: Using ant colony system to consolidate vms for green cloud computing. IEEE Trans. Serv. Comput. **8**(2), 187–198 (2015). https://doi.org/10.1109/TSC.2014.2382555
17. Ferdaus, M.H., Murshed, M., Calheiros, R.N., Buyya, R.: Virtual machine consolidation in cloud data centers using aco metaheuristic. In: Silva, F., Dutra, I., Santos Costa, V. (eds.) Euro-Par 2014 Parallel Processing, pp. 306–317. Springer International Publishing, Cham (2014)
18. Gao, Y., Guan, H., Qi, Z., Hou, Y., Liu, L.: A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. J. Comput. Syst. Sci. **79**(8), 1230–1242 (2013)
19. Gharehpasha, S., Masdari, M., Jafarian, A.: Power efficient virtual machine placement in cloud data centers with a discrete and chaotic hybrid optimization algorithm. Clust. Comput. **24**(2), 1293–1315 (2021)
20. Ghobaei-Arani, M., Rahmanian, A.A., Shamsi, M., Rasouli-Kenari, A.: A learning-based approach for virtual machine placement in cloud data centers. Int. J. Commun. Syst. **31**(8), e3537 (2018). https://doi.org/10.1002/dac.3537. https://onlinelibrary.wiley.com/doi/abs/10.1002/dac.3537. E3537 IJCS-17-0421.R1
21. Hameed, A., Khoshkbarforoushha, A., Ranjan, R., Jayaraman, P.P., Kolodziej, J., Balaji, P., Zeadally, S., Malluhi, Q.M., Tziritas, N., Vishnu, A., et al.: A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems. Computing **98**(7), 751–774 (2016)
22. Herbst, N.R., Kounev, S., Reussner, R.: Elasticity in cloud computing: what it is, and what it is not. In: Proceedings of the

10th International Conference on Autonomic Computing ({ICAC} 13), pp. 23–27 (2013)
23. Hsieh, S.Y., Liu, C.S., Buyya, R., Zomaya, A.Y.: Utilization-prediction-aware virtual machine consolidation approach for energy-efficient cloud data centers. J. Parallel Distrib. Comput. **139**, 99–109 (2020). https://doi.org/10.1016/j.jpdc.2019.12.014. http://www.sciencedirect.com/science/article/pii/S074373151930190X
24. Laganà, D., Mastroianni, C., Meo, M., Renga, D.: Reducing the operational cost of cloud data centers through renewable energy. Algorithms **11**(10), 145 (2018)
25. 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. Model. **58**(5), 1222–1235 (2013). https://doi.org/10.1016/j.mcm.2013.02.003. http://www.sciencedirect.com/science/article/pii/S0895717713000319. The Measurement of Undesirable Outputs: Models Development and Empirical Analyses and Advances in mobile, ubiquitous and cognitive computing
26. Li, Z., Yan, C., Yu, L., Yu, X.: Energy-aware and multi-resource overload probability constraint-based virtual machine dynamic consolidation method. Future Gener. Comput. Syst. **80**, 139–156 (2018) https://doi.org/10.1016/j.future.2017.09.075. http://www.sciencedirect.com/science/article/pii/S0167739X16307476
27. Li, Z., Yu, X., Yu, L., Guo, S., Chang, V.: Energy-efficient and quality-aware vm consolidation method. Future Generat. Comput. Syst. **102**, 789–809 (2020)
28. Lin, C., Liu, P., Wu, J.: Energy-efficient virtual machine provision algorithms for cloud systems. In: 2011 Fourth IEEE International Conference on Utility and Cloud Computing, pp. 81–88 (2011). https://doi.org/10.1109/UCC.2011.21
29. Liu, X.F., Zhan, Z.H., Deng, J.D., Li, Y., Gu, T., Zhang, J.: An energy efficient ant colony system for virtual machine placement in cloud computing. IEEE Trans. Evol. Comput. **22**(1), 113–128 (2016)
30. Mastelic, T., Oleksiak, A., Claussen, H., Brandic, I., Pierson, J.M., Vasilakos, A.V.: Cloud computing: survey on energy efficiency. ACM Comput. Surv. **47**(2), 1–36 (2014). https://doi.org/10.1145/2656204
31. Mastroianni, C., Meo, M., Papuzzo, G.: Probabilistic consolidation of virtual machines in self-organizing cloud data centers. IEEE Trans. Cloud Comput. **1**(2), 215–228 (2013). https://doi.org/10.1109/TCC.2013.17
32. Mi, H., Wang, H., Yin, G., Zhou, Y., Shi, D., Yuan, L.: Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers. In: 2010 IEEE International Conference on Services Computing, pp. 514–521 (2010). https://doi.org/10.1109/SCC.2010.69
33. Mosa, A., Paton, N.W.: Optimizing virtual machine placement for energy and sla in clouds using utility functions. J. Cloud Comput. **5**(1), 17 (2016)
34. Mytton, D.: How much energy do data centers use? (2020). https://davidmytton.blog/how-much-energy-do-data-centers-use/
35. Park, K., Pai, V.S.: Comon: a mostly-scalable monitoring system for planetlab. SIGOPS Oper. Syst. Rev. **40**(1), 65–74 (2006). https://doi.org/10.1145/1113361.1113374
36. Salimian, L., Safi, F.: Survey of energy efficient data centers in cloud computing. In: Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing, UCC'13, pp. 369–374. IEEE Computer Society, USA (2013)
37. Shabeera, T., Madhu Kumar, S., Salam, S.M., Murali Krishnan, K.: Optimizing vm allocation and data placement for data-intensive applications in cloud using aco metaheuristic algorithm. Eng. Sci. Technol. Int. J. **20**(2), 616–628 (2017). https://doi.org/10.1016/j.jestch.2016.11.006. http://www.sciencedirect.com/science/article/pii/S2215098616304232
38. Shehabi, A., Smith, S., Sartor, D., Brown, R., Herrlin, M., Koomey, J., Masanet, E., Horner, N., Azevedo, I., Lintner, W.: United states data center energy usage report. Tech. rep., Lawrence Berkeley National Lab.(LBNL), Berkeley, CA (United States) (2016)
39. Song, W., Xiao, Z., Chen, Q., Luo, H.: Adaptive resource provisioning for the cloud using online bin packing. IEEE Trans. Comput. **63**(11), 2647–2660 (2014). https://doi.org/10.1109/TC.2013.148
40. Speitkamp, B., Bichler, M.: A mathematical programming approach for server consolidation problems in virtualized data centers. IEEE Trans. Serv. Comput. **3**(4), 266–278 (2010). https://doi.org/10.1109/TSC.2010.25
41. Tarahomi, M., Izadi, M., Ghobaei-Arani, M.: An efficient power-aware VM allocation mechanism in cloud data centers: a micro genetic-based approach. Clust. Comput. **24**, 919–934 (2021). https://doi.org/10.1007/s10586-020-03152-9
42. Torre, E., Durillo, J.J., de Maio, V., Agrawal, P., Benedict, S., Saurabh, N., Prodan, R.: A dynamic evolutionary multi-objective virtual machine placement heuristic for cloud data centers. Inf. Softw. Technol. **128**, 106390 (2020). https://doi.org/10.1016/j.infsof.2020.106390. http://www.sciencedirect.com/science/article/pii/S0950584919302101
43. Wu, C.M., Chang, R.S., Chan, H.Y.: A green energy-efficient scheduling algorithm using the dvfs technique for cloud datacenters. Future Gener. Comput. Syst. **37**, 141–147 (2014). https://doi.org/10.1016/j.future.2013.06.009. Special Section: Innovative Methods and Algorithms for Advanced Data-Intensive Computing Special Section: Semantics, Intelligent processing and services for big data Special Section: Advances in Data-Intensive Modelling and Simulation Special Section: Hybrid Intelligence for Growing Internet and its Applications
44. Wu, Q., Ishikawa, F., Zhu, Q., Xia, Y.: Energy and migration cost-aware dynamic virtual machine consolidation in heterogeneous cloud datacenters. IEEE Trans. Serv. Comput. **12**(4), 550–563 (2019). https://doi.org/10.1109/TSC.2016.2616868
45. Yadav, R., Zhang, W., Kaiwartya, O., Singh, P.R., Elgendy, I.A., Tian, Y.C.: Adaptive energy-aware algorithms for minimizing energy consumption and sla violation in cloud computing. IEEE Access **6**, 55923–55936 (2018)
46. Ye, X., Yin, Y., Lan, L.: Energy-efficient many-objective virtual machine placement optimization in a cloud computing environment. IEEE Access **5**, 16006–16020 (2017). https://doi.org/10.1109/ACCESS.2017.2733723
47. Zhang, L., Zhuang, Y., Zhu, W.: Constraint programming based virtual cloud resources allocation model. Int. J. Hybrid Inf. Technol. **6**(6), 333–344 (2013)
48. Zhang, Y., Ansari, N.: Heterogeneity aware dominant resource assistant heuristics for virtual machine consolidation. In: 2013 IEEE Global Communications Conference (GLOBECOM), pp. 1297–1302 (2013). https://doi.org/10.1109/GLOCOM.2013.6831253

Minhaj Ahmad Khan completed his MS and PhD degrees in Computer Science from University of Versailles, France. He also holds a post-doc from University of Bordeaux-I, France. He is currently working as Associate Professor at Bahauddin Zakariya University, Multan. His research interests include Cloud Computing, High Performance Computing, Computer Networks, and Code Optimization. He has several publications in prestigious journals and conferences on these topics.